THE UNIVERSITY
of EDINBURGH

Data Cleaning with Variational Autoencoders

Simão Fernandes Lopes Marques Eduardo

Doctor of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
The University of Edinburgh

2022

Abstract

A typical data science or machine learning pipeline starts with data exploration; then data engineering (wrangling, cleaning); then moves towards modelling (model selection, learning, validation); and finally model visualization or deployment. Most of the datasets used in industry are either structured or text based. Two relevant instances of structured datasets are: graph data (e.g. knowledge graphs), and tabular data (e.g. excel sheets, databases). However, image datasets are increasingly used in industry and have similar pipeline steps.

This thesis explores the data cleaning problem, where two of its main steps are outlier detection and subsequent data repair. This work focuses on outliers that result from corruption processes that are applied to a subset of instances belonging to an original clean dataset. The remaining instances unaffected by corruption, or before corruption, are called inliers. The outlier detection step finds which data instances have been corrupted. The repair step either replaces the entire instance with a clean version, or imputes the values of specific features in that instance that are deemed corrupted. In both cases, an ideal repair process restores the underlying inlier instance, before having been corrupted by errors.

The main goal is to devise machine learning (ML) models that automate both outlier detection and data repair, with minimal supervision by the end-user. In particular, we focus on solutions based on variational autoencoders (VAEs), because these are flexible generative models capable of providing repairs as samples or reconstructions. Moreover, the reconstruction provided by VAEs also allow for the detection of corrupted feature values, unlike classic outlier detection methods. Since the training dataset is corrupted by outliers, the key aspect to good performance in detection and repair is model robustness to data corruption, which prevents overfitting to errors. If the model overfits to errors, then it is difficult to distinguish inliers from outliers, therefore degrading performance. In this thesis two novel generative models are proposed for this task, to be used in different contexts.

The two most common types of errors are either of random or systematic nature. Random errors corrupt each instance independently using an unknown distribution, exhibiting no clear anomalous pattern across outlier instances. Systematic errors result from nearly deterministic transformations that occur repeatedly in the

data, exhibiting a clear pattern across outliers. Overall, this means high capacity models like VAEs more easily overfit to systematic errors, which compromises outlier detection and repair performance. This thesis focuses on point outliers as they are the most commonly found by practitioners. Point outliers are those that can be identified by only evaluating said instance individually, without the context of other instances (e.g. space, time, graphs).

The first model proposal devises a novel unsupervised VAE that is robust to random errors for mixed-type (e.g. categorical, continuous) tabular data. This first model is called the *Robust Variational Autoencoder* (RVAE). We introduce this robustness by designing a decoder architecture that downweighs the contribution of corrupted feature values (cells) during training. Unlike traditional methods, besides providing which instances are outliers, the novel model provides which cells have been corrupted improving model interpretability. It is shown experimentally that the novel model performs better than baselines in cell outlier detection and repair, and is robust against initial hyper-parameter selection.

In the second model proposal the focus is on detection and repair in datasets corrupted by systematic errors. This second model is called the *Clean Subspace Variational Autoencoder* (CLSVAE). The nature of systematic errors makes them easy to learn, and thus easy to overfit to. This means that if they are numerous in a dataset, then unsupervised methods will have difficulty distinguishing between inliers and outliers. A novel semi-supervised VAE is proposed that only requires a small labelled set of inliers and outliers, thus minimizing end-user intervention. The main idea is to learn separate latent representations for inliers and systematic errors, and only use the inlier representation for data repair. The novel model is shown to be robust to systematic errors, and it registers state-of-the-art repair in image datasets. Compared to the baselines, the novel model does better in challenging scenarios, where corruption level is higher or the labelled set is very small.

Lay summary

Data science and machine learning practitioners leverage data from companies and universities to build useful models that can make predictions. These predictions can then be used for the benefit of analytics, decision making, smart consumer electronics, banking, online advertisements and even social networks feeds. These models build upon years of knowledge in the fields of statistics, algebra, calculus and other fields of mathematics. However, often these practitioners deal with data that is of poor quality.

The data can be messy in its format and not conform well to the software used to develop and deploy such models. These issues have to be resolved so that the data can be used by the modelling software; but also so that the data can be stored in existing databases for later. Moreover, the data may also have been corrupted with data examples that do not reflect what the actual untainted data would be like. Those corrupting data examples are called outliers; whilst inliers are those that are normal and do not have that effect. These outliers can have a negative impact on the prediction power of the aforementioned models, which can lead to poor analytics and decision making. In practice data corruption issues may also need to be resolved before data storage.

In this work we focus on outliers that were caused by previous inliers having been compromised by corrupting errors. These errors have diverse origins, e.g. typos, mislabelling of categories, noise in scientific instruments, unwanted image watermarks, camera sensors being broken, human error in data entry. As a result, for this type of outliers it is often possible to revert back and restore the underlying inlier. This task is called data repair.

The process of resolving either data format messiness or data corruption is called *data cleaning*. In this thesis we focus on *resolving the data corruption issue*, which entails detecting the outliers and then repairing them. Generally this has been done manually by the practitioner, often with substantial effort. In the real world, it is common for data scientists to spend much more of their time in data cleaning tasks compared to model development.

In this thesis we propose two novel models that can automate the process of data cleaning for the data corruption issue, i.e. outlier detection and subsequent data repair. By using these proposed models the burden on the practitioner is

lessened with automation, and only a monitoring effort is needed. The family of models that were chosen for this task are deep generative models. The reason was that these are very powerful at generating new data examples that can be used as repairs. These models use a deep learning approach, which is a set of math building blocks for modelling. In the last few years deep learning has been the dominating paradigm in machine learning, producing outstanding predictions and results.

The first model proposal is more hands off, i.e. unsupervised. It only requires the user to set a few initial values to define model behavior in terms of data cleaning. This model is great for unexpected errors that do not repeat constantly throughout the data. The second model requires more effort by the practitioner, but also allows more control over the data cleaning process. The user is required to provide a few data examples of what types of outliers it wants to repair, i.e. semi-supervised. This model is great for errors that affect large parts of the data and repeat constantly throughout. These types of errors produce outliers that tend to be more difficult to remove with solutions like the first model proposal.

Acknowledgements

I would like to thank my supervisor Dr. Charles Sutton, who was fundamental in all of this work. His patience and willingness to support me was pivotal, and my discussions with him were always very insightful. I would also like to thank Dr. Chris Williams, my second supervisor, who helped me quite a bit by providing different perspectives with a keen eye for detail. Dr. Alfredo Nazabal and Dr. Kai Xu were my co-authors and research partners, I've learnt a lot from them. Both were always supportive, keen to help me and with many good suggestions. I would also like to thank my family, my parents and especially my younger brother Afonso. They were always there with great advice, and were my bedrock when I was going through difficult times. Finally, I would like to thank everyone I've met at the University of Edinburgh, especially my colleagues at the CUP research group. I would also like to thank my friends and colleagues at the Center for Doctoral Training in Data Science for all the great memories.

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Simão Fernandes Lopes Marques Eduardo)*

Table of Contents

Chapter 1

Introduction

1.1 Motivation

In the real world, machine learning (ML) and data science practitioners often have to deal with corrupt or messy datasets when developing or deploying models. Moreover, it is common for datasets that have been designated for storage or merging with existing databases to have corruption issues. Besides data corruption other issues affecting datasets might be: units of measure issues, row or column header name issues, file format issues, instance duplication, data is split across several files, imputing known missing entries, removing irrelevant features or data, or even feature transformation.

In general, the machine learning pipeline (Hapke & Nelson, 2020; Xin et al., 2021; Ilyas & Rekatsinas, 2020; Orr et al., 2021) can be described as having several different stages:

- ***Data Exploration*** is where data is explored using visualization tools, like OpenRefine or Trifacta, and other exploratory data analysis methods (e.g. Q-Q plots, boxplots, scatter plots, histograms, dimensionality reduction). It allows the practitioner to detect any problematic issues in the data, make some early choices about potential models to develop, and even label part of the data if needed.
- ***Data Engineering*** is where data is modified and transformed in order to conform with subsequent stages, in particular for model training. This is also where any issues with the data are sorted. We can define two types of

data modification steps:

1. *Data Wrangling* this can be seen as the process of translating or mapping data from one raw format to another, and preparing the data to be accessed for model development or for later stages. This can include feature transformation, merging of different data files, and even data loading scripts for compute clusters.

2. *Data Cleaning* this is the process of detecting and addressing data quality issues and inconsistencies. Some typical issues include: data de-duplication, data imputation, fixing units of measure issues, outlier detection and subsequent data repair if needed. These issues may need to be fixed so as to not affect model training, and subsequent deployment on real world data.

- ***Model Development*** is where the practitioner chooses between different types of models, and tries a range of different hyperparameter values for each model. This can be seen as trial-and-error process, or a continuous improvement of modelling assumptions by the practitioner. Broadly, it encompasses the following steps: model selection, model training, and model validation.

- ***Model Visualization*** is where test set predictions by the chosen model are visualized, and additional metrics are taken into account. Here a final decision about the model is made. Otherwise, another round of model development might take place.

- ***Model Deployment*** is where the model is finally deployed into a production environment. Scalability issues become a concern, and the model is continuously monitored to make sure performance is acceptable.

We will be focusing on the *data cleaning* step of the data engineering stage. *Our focus is on the cleaning of datasets that have been corrupted by outliers, and thus have to be repaired before being usable.* These outlier instances have a diverse provenance, and they can be of different types (Section 2.2.1). In this work we will tackle outliers that result from a data instance having its feature values altered (errors) by some corruption process. There are other causes for outliers besides corruption, even so corruption-based outliers are quite common in practice. Hence, they have recently become relevant for both the ML and database research

communities (Krishnan et al., 2016; Chu et al., 2016; Ruff et al., 2021; Liu et al., 2020; Hendrycks & Dietterich, 2019). Furthermore, we focus on *point outliers* as they are the most common in practice (Section 2.2.1). Point outliers are those that can be identified by themselves, without the context of other instances.

This thesis will focus on the task of *automating the data cleaning* step, particularly for *outlier detection and the subsequent data repair*. To that effect, we will propose novel solutions based on machine learning. Few models in literature provide an end-to-end solution, i.e. combining outlier detection and data repair, which is one of the main aspects of this work. In fact, most models focus on either outlier detection or data repair. However, this ignores the fact that a practitioner often wants to repair the corrupt dataset after outlier detection has taken place.

Data repair is often necessary when there is a need to store new data into existing databases. This can be critical data than needs to be kept, or because it will be used later on for downstream tasks. For instance, banking transactions, client or product data in a company database, or even citizen health records. This has been one of the focuses of the database research community (Fan, 2015), where data repair procedures are used to maintain data quality in existing databases.

In some cases it can make sense to just remove or ignore the outliers without the burden of data repair. In fact, this is the only option if the outliers are not due to having some of their features corrupted. In this case there is no repair possible. Further, in some cases the fraction of outliers in the data is so small that removing them will have little impact on the training of an ML model. In fact, ignoring outliers has been explored for adversarial outliers and systematic outliers (see Section 2.3.3.4), this has been called *data sanitization*.

Nonetheless, in several cases the amount of corruption present in a dataset can be substantial. Thus ignoring the outliers would result in large parts of the dataset not being used, which will impact negatively the diversity of data available to train an ML model. This could also mean that more powerful ML models that require significant amounts of data cannot be trained properly, e.g. *deep learning*. In addition, some authors (Krishnan et al., 2016; Liu et al., 2021) have shown that data repair can have a positive impact on the performance of ML models in downstream applications. However, in other cases the outlier instance is just too corrupted and does not have a recognizable repair. In this case, it is impossible to

produce a repair.

If the purpose of a data repair process is a downstream task, then it should be assessed first if outlier removal suffices. Sometimes direct removal of outliers results in little to no negative impact on the downstream task. If this is the case, then effort can be spared trying to repair the data. The typical example is when very large datasets are available, such that removing a small portion of the data has little impact on ML model generalization.

Even so, in *several cases data repair may yield better results for downstream tasks.* For instance, if in the presence of imbalanced datasets, sometimes outlier removal may affect disproportionately a particular minority data class. In this case, it is possible downstream model generalization may be affected and give rise to unwanted biases. Additionally, in those cases where the dataset is already small, then outlier removal may seriously impair training and lead to poor downstream model generalization. Another case is when different future downstream tasks may be expected to use the same cleansed dataset. Here one may not be able to measure the impact right away of outlier removal, as different ML models may present different sensitivities to that process. Further, as stated before, cases where corruption affects most of the data would lead to a very small training dataset for the downstream model. Hence possibly negatively affecting model generalization.

Lastly, *at test time* some ML models may also need to repair data instances before making a prediction. Indeed, sometimes corruption can mangle the data instances to a point where data repair is needed for the predictor to be able to recognize it. It can lead to low confidence predictions, and sometimes to completely wrong predictions with high confidence. This sensitivity is often both model and corruption dependent, and would need analysis before a decision could be made on using data repair. Alternatively, a downstream predictor may rather have a *rejection option*, where a prediction on some test data instance is not made if its below a certain confidence level.

1.2 Problem Setting

We assume a tabular or image dataset has been corrupted by noise, and a machine learning practitioner (or user) needs to cleanse the dataset. The cells of a tabular

dataset, or pixels of an image dataset, are potentially corrupted with arbitrary noising processes appropriate for each feature type. Note that the instances in tabular dataset are *rows*, whilst for image datasets they are *images*. The first objective in this work is the detection of anomalous instances that have been corrupted, i.e. *outliers* or *row outliers* for tabular data. This task is known as *outlier detection* or *anomaly detection*. Then, in some cases, we also want to detect which cells (or pixels) have been specifically corrupted by errors. This has the advantage of allowing the user to know why that instance is an outlier, thus improving the *interpretability* of the outlier detection process. We call this *cell outlier detection*. Moreover, we make the assumption that most of the instances are inliers, as this conforms to most cases in practice (Ruff et al., 2021; Chandola et al., 2009). In this thesis we only explore *point outliers* (see Section 2.2.1), which means an instance can be considered an inlier or outlier by itself. In other words, there is no need to inspect other data instances within some context (e.g. space proximity, time-series, connections in a graph). Point outliers are the most common type of outlier in literature and in practice. In fact, most traditional outlier detection methods focus on this type of outlier.

The second objective in this work is to repair the corrupted cells (or pixels) in each outlier instance, such that the end result is a clean dataset unaffected by corruption. This is called *data repair*. This is done by either generating new values for the corrupt cells only, or reconstructing the instance altogether (i.e. all cells). Either option is valid depending on the context, but repairing the corrupt cells only tends to be the most applicable. This is akin to *data imputation*, but in our case the model for this task is trained on a corrupted dataset. So these new cell values need to be *close enough to the underlying ground-truth* for *data repair* to be good, as measured by some metric or evaluated qualitatively. Furthermore, one might need to estimate which cells have been corrupted by errors before we can repair them. Interestingly, this is the task of cell outlier detection. Though in some rare cases the corrupt cells mask may already be known.

Having said that, the *overall goal* of this thesis is to *devise models that perform outlier detection (OD) and data repair, with little or no user intervention*. Often, we will refer to this as *automatic detection and repair*. Moreover, sometimes repair needs substantial user intervention, if little to none is used we refer to this as *automated repair*. Generally, some intervention may still be needed in the model,

for such things as hyper-parameter selection (e.g. learning rates, neural network architectures) or even setting a threshold for outlier detection (user may want to control this aspect directly). In some cases, it may be useful to use a small labelled subset of the data to supervise the model. In this case, user intervention may also be needed to build this labelled set if not obtained otherwise. *Obtaining a labelled set can be done during the data exploration stage of the ML pipeline.*

In this work, we explore two types of corruption that are quite relevant in practice, which are *random errors* and *systematic errors* (see Section 2.2.1). Random errors affect an instance by corrupting the cells independently, and using an unknown distribution. For continuous features, a common example of this type of error are those well-modelled by additive noise with zero-mean. Systematic errors result from a nearly deterministic corruption of the cells, which has been applied repeatedly throughout the data instances. Examples include watermarks or deterministic pixel corruption (e.g. artifacts) in images; additive offsets or replacement by default values (e.g. NaN) in sensor data and tabular data.

Consequently, random errors and systematic errors have different impacts on machine learning models, and thus may require different data cleaning solutions. Machine learning models can overfit to either of these errors if they are not made robust to them. If a model overfits to corruption then its ability to perform outlier detection or data repair becomes compromised. This is because it may not be able to distinguish between inliers and outliers. Systematic errors are quite problematic in this sense, because unlike random errors they are far more easily learnt by a model.

1.3 Solving the Problem

Classic models for outlier detection only focus on finding outlier instances, often forgetting to identify which cells are to blame. The latter is not only important in terms of interpretability, but also since detecting outlier cells may be needed for data repair. These classic outlier detection models have been mostly proposed by the statistics, ML and data mining communities (Section 2.2.3). These models tend to be shallow, i.e. lack of modelling capacity, and thus have difficulty modelling complex data distributions. This is the case for a lot of unstructured data like image or text data; but also in the case of tabular data where nonlinear

dependencies between features are not properly modelled by shallow methods. In general, these models also have scalability issues when applied to large datasets, though some have specific versions to tackle this. Furthermore, little thought is given to the subsequent step of data repair by most ML methods that are proposed for outlier detection.

The database research community has also proposed several methods for data cleaning (Sections 2.2.4 and 2.3.2), and some are capable of both outlier detection and data repair. However, these typically rely on the use of either logic rules or user-defined programs (e.g. scripts) to define data quality constraints. These can either describe what constitutes a clean dataset, or the actual errors corrupting the data. One issue with this type of solution is that it requires substantial effort on the part of the practitioner, since this data quality knowledge need to be codified into logic rules or programs. This can be very time consuming, and it requires the practitioner to be skilled in the type of logic used (e.g. first-order logic) or a programming language. Alternatively, these logic rules can be distilled from a master dataset that is clean by using rule mining software. But in most cases in the real world a clean or master dataset is not available to the practitioner. Another issue is one of scalability, since in practice it is quite difficult to fully describe the clean patterns of a large dataset by just using logic rules.

Data imputation (Section 2.3.3) models proposed in statistics and machine learning research also have difficulty being applied here. These models rely on some other method having been trained to perform cell outlier detection, as they need the mask of cells to be imputed. The outlier detection method would have to be trained on corrupt data, and it is possible some corrupted cells would be missed. This could endanger the training of the data imputation model later on if the same data were to be used, as these models are not robust to corruption. Further, even if all corrupt cells are identified, not a lot of clean data instances may be left for proper training of the data imputation model. In addition, a lot of data imputation models assume large amounts of clean data are available for training. Similar arguments can be made about image inpainting models (Section 2.3.3.2), which suffer from similar issues.

1.3.1 Why Variational Autoencoders?

A potential solution for our task should be an ML model that is flexible enough to fit complex data distributions; and be applicable to different types of data, e.g. image data or tabular data. In fact, tabular datasets often encompass features with different data types, e.g. continuous and categorical, which is often referred to as mixed-type data. This is an important requirement since we aim to provide good fidelity in terms of data repair estimates. Deep learning models fit this requirement as they have shown their potential as very powerful function approximators in the last few years. Moreover, recently deep learning has started to gain traction for outlier detection (Ruff et al., 2021).

Given our task of combined outlier detection and data repair, deep generative models seem to be quite appropriate as a solution. Broadly, a generative model is an ML model that is trained to fit a data distribution. Deep generative models use neural networks as function approximators, and thus they are extremely flexible. They were chosen for their ability to reconstruct and sample data, which is necessary for data repair. However, because of their high capacity they can easily overfit to corrupting errors. Hence, it is very important to make sure that our deep generative model is robust to the type of corruption found in the data.

In this thesis we focused on Variational Autoencoders (VAEs). These were preferred for their simplicity in terms of implementation, and stable training regimes. Other models like Generative Adversarial Networks (GANs) are more complicated in terms of implementation, and often have very unstable training regimes. Hence, VAEs were chosen as a first incursion into *robust deep generative models* for the task of automating outlier detection and repair. Moreover, since VAEs are reconstruction-based models (Section 2.2.3.6) they allow for the granularity of cell outlier detection.

1.3.2 Thesis Contributions

Having decided on the type of model for our task of automated data cleaning, this thesis introduces two novel VAEs that are robust to data corruption. The first model proposal focuses on data cleaning for random errors. The second proposal focus on systematic errors. All models explored in this thesis apply to *point outliers*, as defined in Section 2.2.1.

In more detail, the contributions of this thesis are the following:

- In Chapter 3, the novel ***Robust Variational Autoencoder* (RVAE)** is proposed. This unsupervised VAE model is designed to perform combined outlier detection and data repair in the presence of random error corruption. One of the novelties is the exploration of the cell outlier detection task, which can be important for subsequent data repair or for interpretability. When the model was first published, to our knowledge it was the only robust deep VAE model for mixed-type tabular data. This deep generative model learns a distribution of the underlying inlier data, whilst isolating the outlier cells, and thus downweighting their contribution to the training loss. RVAE learns the probability of each cell being an outlier, which allows the balancing of different likelihood models (e.g. categorical and continuous) in the instance anomaly score. This makes RVAE much more suitable for outlier detection in mixed-type data. RVAE outperformed or matched competitor model performance in outlier detection (cell and row) and in data repair. Experiments were carried out using several tabular datasets from UCI ML database, with different corruption levels and error types. The model was also shown to be robust against initial value choices for its main hyperparameter α, for moderate corruption amounts.
- In Chapter 4, the novel ***Clean Subspace Variational Autoencoder* (CLSVAE)** is proposed. This semi-supervised VAE is designed to perform data cleaning in the presence of systematic error corruption. Systematic errors result from nearly deterministic transformations (plus potentially some noise) that occur repeatedly in the data, e.g. specific image pixels being set to default values or watermarks. Consequently, models with enough capacity easily overfit to these errors, making outlier detection and repair difficult. Since it is difficult to isolate these types of outliers using unsupervised models, we propose using user interaction in the form of semi-supervision. In order to make it easier to the user, we limit supervision to only a few labelled examples, particularly the type of outliers (systematic errors) the user wishes to repair. This is far more practical than using logic rules or programs, since these require expert knowledge and much more effort by the user. Seeing as a systematic outlier is a combination of patterns of a clean instance with patterns of a systematic error, the main insight

is that inliers can be modelled by a smaller representation (subspace) in a model compared to outliers. So, the main idea behind CLSVAE is to partition the latent space and model inlier and outlier patterns separately. Experiments provided use three image datasets in scenarios with different levels of corruption and labelled set sizes. CLSVAE is effective with much less labelled data compared to previous related models (two of them state-of-the-art VAEs), often with less than 2% of the data. In fact, for data repair with just 0.25% of labelled data CLSVAE registers a relative error decrease of 58% compared to the closest baseline.

Work Disclosure

Here we disclose the work split between authors for each content chapter.

- Disclosure for Chapter 3 (RVAE). This chapter is based on the paper *Robust Variational Autoencoders for Outlier Detection and Repair of Mixed-Type Data* published in *AISTATS 2020* (Eduardo et al., 2020). This is joint work with Dr. Alfredo Nazabal, Dr. Christopher K. I. Williams and Dr. Charles Sutton, where I was first author. My contribution to this work was the generative model; part of the inference scheme (RVAE-CVI); outlier detection and repair procedures; vast majority of coding for RVAE and baselines; code for most of the experiments; writing significant part of the paper. Dr Alfredo Nazabal partly developed the inference scheme (RVAE-CVI); code for experiments on standard OC-SVM and ABDA models; general code review; helped in repair metrics coding; helped in writing the paper; and steadfast supervision. Dr. Christopher Williams and Dr. Charles Sutton provided much needed supervision, and guidance in all things.

- Disclosure for Chapter 4 (CLSVAE). This chapter is based on a pre-print called *Repairing Systematic Outliers by Learning Clean Subspaces in VAEs*, and is available online on OpenReview[1] and arXiv[2]. This is joint work with Dr. Kai Xu, Dr. Alfredo Nazabal, and Dr. Charles Sutton, where I was first author. The co-authors provided helpful guidance and supervision, and helped write the pre-print. My contribution was proposing the generative model, though helpful suggestions were made by co-authors; the inference

[1] `https://openreview.net/forum?id=kHNKTO2sYH`
[2] `https://arxiv.org/abs/2207.08050`

procedure; coding for CLSVAE and all baselines; all the experiments; writing most of the pre-print.

Chapter 2

Background

2.1 Problem Setting: Notation and Definitions

Consider a tabular or image dataset $\mathcal{X}$ that has been corrupted by noise. This dataset, or most of it, will be used as a training set for a model that performs both *outlier detection* (OD) and *data repair*. The instances in this corrupted dataset $\mathcal{X}$ are numbered using $n \in \{1, \cdots, N\}$, and we can define it as $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^{N}$. Further, each instance $\boldsymbol{x}_n$ is comprised of several features (or pixel positions) numbered using $d \in \{1, \cdots, D\}$. Each cell (or pixel) x_{nd} in the dataset can be continuous (real) $x_{nd} \in \mathbb{R}$, or categorical $x_{nd} \in \{1, .., C_d\}$ with C_d the number of unique categories of feature d. Usually, in an image dataset all the features will have the same feature type, i.e. all continuous or all categorical. If no more information is given, e.g. labelled data, then this is called an *unsupervised setting*.

In some cases we might have access to a small labelled set, i.e. a *trusted set*, which is a subset of the overall dataset $\mathcal{X}$. In essence, we have the labelled part of the data $\mathcal{X}_l$, and the unlabelled part $\mathcal{X}_u$. We can write the overall dataset as $\mathcal{X} = \mathcal{X}_u \cup \mathcal{X}_l$, where we have $\mathcal{X}_u = \{\boldsymbol{x}_n\}_{n=1}^{N_u}$ and $\mathcal{X}_l = \{\boldsymbol{x}_n\}_{n=1+N_u}^{N_l+N_u}$. Hence, the overall size of the dataset is $N = N_l + N_u$. In this case, each $\boldsymbol{x}_n \in \mathcal{X}_l$ is associated with a label $y_n \in \{0, 1\}$, which indicates whether $\boldsymbol{x}_n$ is an inlier ($y_n = 1$) or an outlier ($y_n = 0$). We define the set of labels as $\mathcal{Y}_l = \{y_n\}_{n=1+N_u}^{N_l+N_u}$ and thus the trusted set is formally defined by $(\mathcal{X}_l, \mathcal{Y}_l)$. This scenario is called a *semi-supervised setting*.

Often we leave out the subscript n to simplify notation, for example, an instance

in the dataset (e.g. a row, or an image) may be represented as: $\boldsymbol{x}$ is $\boldsymbol{x}_n$; or $\boldsymbol{z}$ is $\boldsymbol{z}_n$; or y is y_n. Further, a cell (or pixel) of an instance can also be represented as: x_{nd} is x_d. As such, we can define an instance as a vector with D features as follows: $\boldsymbol{x}_n = [x_{n1} \dots x_{nd} \dots x_{nD}]$ is $\boldsymbol{x} = [x_1 \dots x_d \dots x_D]$.

Our problem assumes that *corruption is due to errors that change the ground-truth values of cells (or pixels).* These errors may originate from several different processes, and they may be either *random* or *systematic* in nature (see Section 2.2.1). Those instances that have been affected by corruption in its cells (or pixels) are called *outliers*; whilst those left unchanged are called *inliers*. Typically, a corrupted dataset tends to have more inliers than outliers, but the amount of outlier instances can range from small to large. Note that these type of outliers fall within the scope of *point outliers* (see Section 2.2.1), which is the main focus of this thesis.

We can model an arbitrary corruption process through a general transformation f_{cr}, which represents the corruption applied to the cells of a clean instance. Assuming the clean ground-truth of an instance is $\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}$, then $\boldsymbol{x} \in \mathcal{X}$ is defined by: if an inlier ($y = 1$) then $\boldsymbol{x} = \tilde{\boldsymbol{x}}$; otherwise, if an outlier ($y = 0$) then $\boldsymbol{x} = f_{cr}(\tilde{\boldsymbol{x}})$. Further, in case of an outlier then f_{cr} will only change a subset of cells, such that for some feature d: for clean cells we have $x_d = \tilde{x}_d$; and for dirty cells we have $x_d = [f_{cr}(\tilde{\boldsymbol{x}})]_d$. Note that $[\,\cdot\,]_d$ is just to make it clear we mean the d^{th} element of the resulting vector, i.e. select the d^{th} cell. Lastly, we will often call a dataset with only inlier instances an *inlier dataset*, or *clean dataset*. Likewise, we call a dataset that has been corrupted with outliers a *corrupt dataset*, or *dirty dataset*.

2.2 Outlier Detection Task

In this section, we discuss the outlier detection task in more detail. We define it formally and review prior literature on the topic.

Traditional outlier detection (OD), also called anomaly detection, focuses on finding which data instances from a test set do not belong to normal data. In this context, normal data is just all the patterns that are expressed by inliers. Outliers are those that do not fit within the diversity seen in normal data. As seen in Hawkins (1980), an outlier is *an observation that deviates so much from the other observations as to arouse suspicion that a different mechanism generated it.* As

such, this assumes normal data is generated by a process (mechanism) different from that which generated the outlier. In practice, sometimes user intervention is needed to help identify the normal data. Note that normal data and clean data are the same thing, though in traditional outlier detection the term normal data is more popular. In our problem notation inliers are assumed to be the majority in the data, which is a common assumption amongst classic OD methods (Emmott et al., 2015). Indeed, several OD methods tend to assume outliers are those exhibiting rare patterns (Chandola et al., 2009; Ruff et al., 2021). Yet, sometimes outliers are numerous and this assumption does not hold. In this case, methods should be tailored to the type of outliers seen, or use some kind supervision to overcome this.

More generally, in the recent survey Ruff et al. (2021) the authors specify that an outlier *is an observation that deviates considerably from some concept of normality.* This concept of *normality* can be expressed by some distribution $p^+(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathcal{X}$ such that it absolutely models the *ground-truth law of normal behavior.* This *law of normal behavior* is inherently subjective and specific to each application or context. For instance, the biases of a practitioner or a downstream task can constrain what is normal. Therefore, according to Ruff et al. (2021), an outlier is a data instance $\boldsymbol{x} \in \mathcal{X}$ that lies in a low probability region under $p^+(\boldsymbol{x})$. In other words, we can say that an outlier is an instance that is sampled from a generative process that is not considered normal.

The fact that a particular instance is rare or unusual does not make it automatically an outlier, as it can still abide the *law of normal behavior.* For instance, in a data exploration setting the practitioner is responsible for defining what is normal. In this case an instance can be flagged as unusual, but the practitioner can still decide it to be normal. Indeed, the variability within normal data (inliers) can be quite large in some applications (e.g. biometric data, image data), and the *deviation* to other normal instances larger than to some outlier instances. Besides, in image data a corruption process (e.g. compression artifacts) can be quite frequent, and still not abide to the *law of normal behavior.* In this case all instances affected by corruption should be flagged as outliers despite being frequent. Alternatively, in the context of out-of-distribution outliers (see Section 2.2.1), it is assumed only the diversity seen during training abides to the *law of normal behavior.* Thus no assumption on rarity makes sense in this context. Another inductive bias that

can decide whether an instance might be an outlier is the impact that instance may have on the performance of a downstream task model. Indeed, here the *law of normal behavior* is constrained by the downstream application the dataset will have. Once more, here nothing is assumed about the rarity of an instance. Still, for a lot of cases in outlier detection (Chandola et al., 2009; Ruff et al., 2021) the rarity assumption seems useful and practical, even if only as a starting point.

A different perspective for characterizing an instance as an outlier is to observe the impact it has on model learning. Specifically, we can say the data instance has *high influence* on the ML model if removing said instance from the train set (leave-one-out retraining) would yield clearly different model parameter values. Conversely, *low influence* instances are those that have little impact on parameter values if removed from training. Generally instances indicating high influence can be either outliers or just rare (high leverage). If dealing with predictor models, then outliers are those that the model cannot predict with confidence if removed from the train set. Otherwise, rare (high leverage) instances are those that when removed from the train set, and despite qualitatively diverging from the majority of other instances, the model can predict the instance with some confidence. Therefore, just obtaining the *high influence instances from a dataset is not enough for properly detecting an outlier*. Thus further filtering would be needed to account for false positives – e.g. like user inspection.

For most practical cases, when measuring influence through leave-one-out retraining it becomes too computationally expensive. Fortunately, a famous proxy for the leave-one-out strategy is the *influence function* (Cook & Weisberg, 1980), which has been explored in *robust statistics* (Huber, 2004) for simple convex models. Moreover, this has been recently extended to non-convex models like deep neural networks (Koh & Liang, 2017; Hara et al., 2019). This resolved some tractability issues, though estimation of data instance influence is still computationally intensive. Once more, an influential instance is not necessarily an outlier.

Another similar approach is to use the concept of *memorization* by the model, which is particularly relevant in deep neural networks (Arpit et al., 2017). The main idea is that the only way for outliers and rare instances to be predicted by the model is to be *memorized* (or overfitted) rather than learnt through model generalization. This can be observed through a leave-one-out strategy, where one can measure if there is a difference in prediction when said data instance is

in instead of out of the train set. If there is a marked difference in prediction confidence, then the instance is said to be memorized. Since measuring this through a leave-one-out strategy is computationally intractable in most cases, recent literature has focused on tractable estimators for memorization metrics (Arpit et al., 2017; Feldman & Zhang, 2020). Again, a memorized instance is not necessarily an outlier, since it can be just atypical.

Several different assumptions can be made about the dataset setup. In our problem we assume that outlier instances are due to corruption (Ruff et al., 2021; Liu et al., 2020), but other processes may generate outliers – e.g. merging of two different datasets from different sources, or in banking data a set of transactions that are deemed fraud or atypical.

Another assumption is whether a curated training dataset made only of normal data is available. Some OD methods only perform well if this is guaranteed, for instance a *Standard Variational Autoencoder* (Kingma & Welling, 2014; An & Cho, 2015) or *One-Class Support Vector Machines* (OC-SVM) (Schölkopf et al., 1999). Though OC-SVMs can have their hyperparameters adjusted to deal with this, by using a labelled validation set. Still, generally *One-Class Classification* (OCC) methods (Moya & Hush, 1996) like OC-SVM should be trained using only normal data. If methods are trained in this way, then literature often refers to it as *novelty detection*. We note that some literature (Villa-Pérez et al., 2021) defines the methods that train on normal data only as semi-supervised learning. In our work, we take the perspective that semi-supervision includes labelled outliers – e.g. (Ruff et al., 2019).

A different assumption is whether any supervision is provided for training, i.e. labelled inliers and outliers. If labels are provided for all the instances, then this becomes a standard classification problem. Standard methods like support vector machines (SVMs) (Cortes & Vapnik, 1995) or logistic regression can be applied. More advanced methods like deep learning classification (Litjens et al., 2017) can be applied, specially for unstructured datasets (e.g. medical imaging). Though *class imbalance* issues need to be taken into account, since typically outliers are the minority class. If only a few labels exist, then semi-supervised methods can be applied. One option could be *semi-supervised Variational Autoencoders* (see Section 2.5.3.2), or the deep autoencoder in Ruff et al. (2019). For classic (non-deep learning) semi-supervised OD methods see Görnitz et al. (2013); Liu &

Zheng (2006); Ruff et al. (2021). Still, most of the OD methods are unsupervised Aggarwal (2016).

Several good surveys exist for traditional outlier detection methods. In Hellerstein (2008) statistical methods and robust estimators are explored for numerical only data. In Emmott et al. (2015) a survey and meta-analysis is provided on typical outlier detection from the machine learning and data mining communities. It describes proper benchmarking methodologies and provides an ontology for outlier detection contexts. An earlier and influential survey about outlier detection and outlier types can be seen in Chandola et al. (2009). More recently Ruff et al. (2021) compares classic (shallow) outlier detection models with deep learning ones, and best practices on when to apply each type. It also provides an empirical comparison on several methods on a few datasets. Lastly, the book Aggarwal (2016) gives an overall good overview of the field of outlier detection.

2.2.1 Types of Outliers in Data

Here we present a possible characterization of outlier types that are commonly found in data. In this thesis we mainly focus on tabular and image data. However, outliers are also present in other types of data like natural language, time-series, or sensor data.

Given the typical taxonomy found in several surveys (Ruff et al., 2021; Chandola et al., 2009), we can define the following general categorization for outliers:

- ***Point Outliers*** (Krishnan et al., 2016; Hendrycks & Dietterich, 2019) Data instances are considered outliers if by themselves they do not fit what is defined as clean or normal data. This class of outliers is the *most common* in outlier detection literature. *The typical corruption-based outliers explored in this thesis tend to fit this class.* The traditional outlier detection method here is only concerned about evaluating each instance on its own.
- ***Context Outliers*** Data instances are only considered outliers depending on a specific context such as time (Gupta et al., 2013), space (Zheng et al., 2017), or connections on a graph (Akoglu et al., 2015). For example, an individual instance in a time-series can be considered an outlier if it deviates substantially from previous inlier instances in that series. Likewise, a node in a graph is an outlier instance if it deviates substantially from its inlier

neighbors. Here outlier detection methods evaluate the instance in question and its neighborhood (instances) to make an assessment about the outlierness of the instance.

- ***Group Outliers*** (Chalapathy et al., 2018b; Muandet & Schölkopf, 2013; Kasieczka et al., 2021) Data instances are considered outliers as a set or group, and not individually like *point outliers* or *context outliers*. In fact, instances by themselves may be considered inliers, but taken as a group they are outliers. For instance, in a banking dataset a set of transactions might look suspicious (outliers) when seen as a whole. Yet, alone each transaction would seem quite normal. Outlier detection methods here evaluate instances in groups, either already given or inferred later, and the decision about outlierness is made for the entire set.

The most explored type of outliers from a traditional perspective are the *point outliers*. Generally, literature tends to explore outliers that are not due to error corruption, and thus have no underlying inlier (repair). For instance, these can be a result of a merge between a normal data source and an outlier data source from web crawling. These can be of two types: *random outliers* have no obvious pattern across the outliers in the data; *structured or systematic outliers* (Diakonikolas et al., 2018; Ruff et al., 2019) have a specific pattern that repeats across the outliers in the data. The latter outliers are more difficult to handle for both outlier detection and data repair. This is due to structured outliers being much easier to overfit to. Some examples of systematic outliers that are not due to corruption are: specific data classes from a merged data source that are undesirable (e.g. pictures of dogs in an all cat dataset); transactions in a banking dataset that have a specific pattern considered anomalous or fraudulent; intrusion detection in a dataset of computer network connections, where "bad" connections (outliers) can be of several specific types (e.g. KDD CUP 99[1]); or even *out-of-distribution outliers* in a test dataset (see definition below in the section).

Corruption-based Outliers (Liu et al., 2020; Ruff et al., 2021; Hendrycks & Dietterich, 2019) This thesis focuses on outliers due to corruption, which are less explored in literature. By and large these are *point outliers*, and our focus will be on these. These type of outliers are due to the corruption of a subset of data instances, which were previously considered clean or inliers. This means that it

[1] `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

is possible to find a process that reverses this corruption, and thus restores the underlying inlier. This reverse process is called repair. Usually, for each instance, only some cells in a row (or pixels in an image) are corrupted, whilst the reminder cells are left intact. Different types of corruption exist, in this thesis we focus on two types: *random errors* or *systematic errors*. These can be due to storage or transmission issues; data-entry issues; improper loading or transformation of databases; crowdsourcing of data; or merging two different sources of data, e.g. different data formats; and others.

Broadly we can define the following error corruption types:

- ***Random Errors*** (Liu et al., 2020; Khademi et al., 2021; Krishnan et al., 2016; Hendrycks & Dietterich, 2019) Outliers are created by a corruption transformation affecting each instance independently using an unknown distribution. Typically the cells (or pixels) corrupted are selected independently at random as well. A simple example is Additive white Gaussian noise (AWGN). This type of error exhibits no clear anomalous pattern across the outlier instances. Hence the errors cannot be properly predicted by an ideal model, though a very flexible model can overfit the errors. *These outliers can be repaired via regularization or data-reweighting*, see Section 2.5.2 and model proposal in Chapter 3 for model examples. For a more in-depth discussion on the impact of these errors in generative models please see Sections 2.5 and 2.5.1.2.

  Some examples: additive noise with zero mean (e.g. AWGN), due to sensor or measurement error; impulse noise in images (e.g. salt-and-pepper noise); shot noise in images; Poisson-Gaussian noise in medical imaging; random category change due to mislabelling (in categorical feature).

- ***Systematic Errors*** (Liu et al., 2020; Krishnan et al., 2016; Lew et al., 2021; Broaddus et al., 2020; Aigrain et al., 2017) (Boyat & Joshi, 2015, section 2.10) Outliers result from a nearly deterministic transformation (plus potentially some noise) applied repeatedly to a subset of instances in the data. Further, typically the same cells (or pixels) are affected, but not always as there can be some positional randomness. This also fits the classic definition of systematic error as seen in measurement errors of scientific instruments in physics (Taylor, 1997). A simple example is a specific typo

in a categorical feature that is repeated across several instances. Another example is the occlusion or missingness of patches of pixels in the same position across several images in a dataset. As we can see, this type of error exhibits a specific anomalous pattern common to several outlier instances. Hence these errors could be predicted by an ideal model. *If these outliers are frequent in the data, then this may cause some outlier detection methods to more easily overfit and learn the errors.* This is problematic for the performance of both outlier detection and data repair processes. *These outliers can be repaired via semi-supervised latent disentanglement models*, see Section 2.5.4 and model proposal in Chapter 4. Once more Sections 2.5 and 2.5.1.2 discuss the impact of errors in models.

Some examples: watermarks; position-based artifacts on images (e.g. errors due to camera sensors, or medical imaging sensors); replacement by default values in data transformation processes (e.g. NaN's) or sensor data (e.g. 0's); mislabelling due to deterministic change of categories (e.g. a format issue, or data-entry issues).

More generally The definition above covers a major portion of point outliers that are originated by systematic error corruption (Liu et al., 2020; Krishnan et al., 2016; Aigrain et al., 2017). Hence in this thesis the focus was on this type of systematic error. However, there can still be outliers due to systematic errors that do not conform to this definition. For example, systematic errors that corrupt every single feature (cell or pixel) in a data instance are not covered. This can be the case for images that are blurred in all pixels by some systematic pattern. Another example are systematic errors that result from the combination of several underlying inlier instances from the original dataset. This can be the case when transmission errors occur in video images and there is an overlap between several frames. Lastly, as discussed before, we note to the reader that systematic outliers can also be due to different processes other than error corruption (and thus have no repair).

- ***Adversarial Errors*** (Diakonikolas et al., 2018; Liu et al., 2020) This corruption is due to an attacker corrupting data instances with the goal of fooling a machine learning model – e.g. wrong classification decisions. This is done with malicious intent, in order to exploit the model at a later date.

Out-of-Distribution Outliers (Yang et al., 2021; Ren et al., 2019) These outliers have become more important recently in the field of deep learning reliability and safety. Data instances are outliers if they belong to a different type of dataset than the one the model was trained on. In fact, these outliers express patterns that are not found in the original training dataset. These outliers are also *point outliers*.

Once more, *this thesis will be focusing on outliers that are due to corruption*. In particular both *random errors* and *systematic errors* are explored. In fact, *all models and scenarios explored in this thesis focus on point outliers*.

2.2.2 Problem Definition for Outlier Detection

Given the notation in Section 2.1, the task of traditional outlier detection (OD) on dataset $\mathcal{X}$ is that of finding the ground-truth labels $y \in \mathcal{Y}$, for each instance $\boldsymbol{x} \in \mathcal{X}$. Assume there exists some ideal model g_r^θ that has been trained using $\mathcal{X}$, where θ expresses the parameters of the model. Then this ideal model is *robust* to corruption for this task if it is able to obtain the labels y of set $\mathcal{Y}$ correctly. In reality this may not be possible, hence a model should strive to detect the majority of outliers in $\mathcal{X}$, i.e. instances with ground-truth $y = 0$.

In practice, the model g_r^θ should be able to produce a score $\mathcal{A}^\theta$, which will be used to rank instances. Again, θ reflects the dependency on model parameters. Usually $\mathcal{A}^\theta(\boldsymbol{x})$ is termed an *anomaly score* for $\boldsymbol{x}$. A higher (scalar) value for $\mathcal{A}^\theta(\boldsymbol{x})$ means that instance $\boldsymbol{x}$ is more likely to be an outlier. Accordingly, we can formally define the outlier detection process as

$$y = \begin{cases} 0, & \text{if} \quad \mathcal{A}^\theta(\boldsymbol{x}) \geq \gamma \\ 1, & \text{if} \quad \mathcal{A}^\theta(\boldsymbol{x}) < \gamma \end{cases} \tag{2.1}$$

and as such we can define the set of outlier instances as $\mathcal{O} = \{\boldsymbol{x} \in \mathcal{X} | \mathcal{A}^\theta(\boldsymbol{x}) \geq \gamma\}$. In an ideal setting, score $\mathcal{A}^\theta$ and threshold γ are jointly optimized such that all outliers are captured in $\mathcal{O}$. In reality, score $\mathcal{A}^\theta$ has limited flexibility due to the underlying model g_r^θ in practice. Further, we often see in practice γ being user-defined as to control OD performance. The setting of γ will often represent a tradeoff between *precision* and *recall* as represented by a point in the *precision-recall* curve; though a good score $\mathcal{A}^\theta$ will be able to mitigate this aspect.

Now for the task of cell outlier detection, we need to obtain a score that allows to rank cells of an instance in terms of being corrupt, i.e. *cell outliers*. We can define a different score $\mathcal{A}_d^\theta$ for each feature d, which allows us to decide if a cell x_d of instance $\boldsymbol{x}$ is indeed a cell outlier. Once again, this score is derived from the same underlying model g_r^θ, which has been trained on a corrupted dataset. Like before, a higher (scalar) value for $\mathcal{A}_d^\theta(\boldsymbol{x})$ means that x_d is more likely to be a cell outlier. Note that this score depends on the entire instance $\boldsymbol{x}$, instead of just on cell x_d. The reasoning is that the *context provided by the other cells in the instance may be needed to provide a good decision*, i.e. a score. Therefore, for some outlier instance $\boldsymbol{x} \in \mathcal{O}$, we can define the set of cell outliers therein as $\mathcal{C} = \left\{d \in \{1, \cdots, D\} \mid \mathcal{A}_d^\theta(\boldsymbol{x}) \geq \gamma_d\right\}$, where γ_d can be tuned for each feature d.

The vast majority of literature focuses on just finding the outlier data instances. Nonetheless, by neglecting cell outlier detection a method fails to provide which cells are to blame for the anomalous behavior. This can be useful for interpretability, but also later in data repair. Moreover, we note that all *anomaly scores* defined here and used throughout this thesis only apply to the task of detecting *point outliers* (see Section 2.2.1).

2.2.3 Classic Methods: Machine Learning and Data Mining

In this section, we discuss classic outlier detection methods from the statistics, machine learning (ML) and data mining (DM) communities. A more in-depth discussion can be found in Aggarwal (2016) or Ruff et al. (2021). The vast majority of these methods can only be applied to *point outliers* (see Section 2.2.1), though in some cases modifications can be made.

2.2.3.1 Statistical Methods

Most of the early work in outlier detection was done by statisticians. The main idea behind this approach is that in a probability distribution representing normal data the outlier instances should be in the low probability regions. Respectively inlier instances should be in high probability regions. Therefore, after a probability distribution is fitted to the training dataset, an inference test decides whether some instance belongs to the distribution. If not, then the instance is an outlier.

These techniques assume inliers are generated by a process that is modelled

by a probability distribution p with parameters θ. Typically, the parameters of these models are estimated via *maximum likelihood estimation* (MLE). A technique is considered parametric if it makes strong assumptions about the data distribution. Contrarily, the technique is considered non-parametric if the probability distribution is chiefly determined by the training data.

Lastly, we note that some simple statistical methods can be used for the practitioner visually detecting outliers. For instance, a histogram may be used to detect bins that are anomalous, and their points marked as outliers. Further, box-plots and Q-Q (quantile-quantile) plots can aid in detecting outliers visually. However, this requires manual intervention by the user, and may become problematic for large datasets.

Statistical Tests and Distances These techniques usually pick a specific distribution to model normal data, and then either evaluate a distance or perform a statistical test. The most simple example of this type of technique is using a Gaussian distribution as a model for the generative process of normal data. In this case mean μ and variance σ are estimated via MLE estimators. The anomaly score is then given by the distance between the unseen instance x and the mean μ. This number can be reported as the number of standard deviations σ, as expressed by the *z-score*. If the distance is bigger than 3σ then the instance is marked as an anomaly. The reasoning is that most of the probability mass is within the region $\mu \pm 3\sigma$, i.e. around 99%.

On the other hand, for multivariate data the *Mahalanobis distance* can similarly be used to measure the distance of instance $\boldsymbol{x}$ to its mean μ, but correct for volume and direction using the covariance matrix Σ. However in high dimensions this anomaly score may become meaningless when it comes to outlier detection, due to it revolving around a single value. This is due to the probability mass of a high dimensional Gaussian distribution being concentrated in "shell" surrounding the mean.

More sophisticated options include the *Grubb's Outlier Test* (Tietjen & Moore, 1972) for univariate data, or *Dixon's Q Test* (Dean & Dixon, 1951) in case some prior knowledge about outliers is known. Both these statistical tests assume normal data is generated by a Gaussian distribution. For both a sample statistic is computed, and then according to confidence level a threshold defines if the

instance is an outlier or not. Lastly, if trying to detect multiple outliers, then *Rosner's Test* (Rosner, 1983) may be used instead.

Likelihood or Density Estimation These type of techniques usually learn a probability distribution on the training data, using either parametric or non-parametric models. Then an anomaly score for some test instance is given by the *negative log* of the probability distribution $p_\theta(\boldsymbol{x})$, i.e. the *negative log-likelihood* (NLL).

A good example of a non-parametric method is *kernel density estimation* (KDE), and then using an NLL score for the test instance. However, this method works best if the training data as few or no outliers. Otherwise, these can contaminate the density estimation process and assign probability mass where none should exist (outlier regions). A *robust KDE* (RKDE) method (Kim & Scott, 2012) was developed with this in mind, thus providing a KDE that is less sensitive to outliers in the training data.

Another popular approach is to use a *Gaussian Mixture Model* (GMM) for density estimation, and again use the NLL score. This is a parametric approach, where the Gaussian component parameters and membership probabilities need to be learnt. If outliers are present in the training data then some caution should be taken if they are numerous. If the outliers can be represented by a cluster (if it can be identified) then that GMM component should be excluded from the NLL score. This can happen with *structured outliers*. In this case, cluster membership probability can also be used, using its negative log. However, for dispersed outliers in the training set, other strategies should considered if they are numerous. For instance, an *ensemble of GMMs* (eGMM) is proposed in Emmott et al. (2015) where the authors fit several GMM models with varying number of components. Then they average across GMMs models to obtain the final distribution, like in other ensemble methods. This averaged distribution is used by the NLL anomaly score.

A different option which can be quite practical is to use *histogram-based* density estimation. This is a very simple non-parametric approach. If the histogram is obtained by using uncorrupted training data, then we should see if a test instance falls within the boundaries of any bins. If not, then the instance is an outlier. Alternatively, if training on corrupted data, then the small size of some bins may

determine whether those instances are outliers.

Moreover, in the field of *robust statistics* (Huber, 2004; Barron, 2019) authors have proposed different *heavy-tailed distributions* in order to make training parametric models less sensitive to outliers. This is important when a significant amount of outliers are present, or in the case of structured outliers (e.g. due to systematic errors). In this case, heavy-tailed distributions may allow for better density estimation of the normal data, and thus a better NLL anomaly score. For more details, we discuss this further in the context of generative modelling using VAEs, see Section 2.5.2.

Statistical Depth These types of methods organize the data instances into convex hull layers, where outliers are instances in the outer most layers. In essence, outlier instances are at the border of the (training) data space. These methods exploit the concept of *statistical depth* (Mosler, 2013) from non-parametric statistics, where the most central region serves as a median for the data. Some good examples of such models are Kwok & Ng (1998); Fernández-Francos et al. (2017); Staerman et al. (2020).

2.2.3.2 Spatial Proximity Methods

These algorithms have been developed mostly by the data mining community. The main idea is that the *outlierness* of some test instance depends principally on the instances in its neighborhood (proximity). Particularly, it assumes that inliers have dense neighborhoods, with several data instances in its proximity. Contrarily, outliers are far apart from their neighbors. In order to produce an anomaly score, mainly two different ways exist to account for instances in the proximity. These are either *distance-based* or *density-based*. Given their nature, these are considered non-parametric methods. This strategy may not work well in the case of *structured outliers*, since these may cluster together.

Distance-based These models judge a test instance based on the distance to its neighbors. For instance, one can use the *k-nearest neighbors* (KNN) distance to obtain an anomaly score for a test instance. In this case, the top h data instances are those whose distance to their k^{th} nearest neighbor is greatest. However, the average of all k nearest neighbors could also be used. A good scalable example of this type of model is Ghoting et al. (2008), which uses a KNN-type distance. An

older model can be seen here Bay & Schwabacher (2003). However, KNN distance can perform badly in datasets where data clusters may have large variation in density.

Density-based These models were created to correct the above issue with varying density of clusters in distance-based models. Instead of using an anomaly score based on distance, they use a relative concept of density. No actual density estimation is needed, for instance a surrogate like euclidean distance can be used. The idea here is to compare the density of a test instance to the density of its local neighbors. The ratio of these two will define the anomaly score for a test instance. The main assumption is that inliers and the instances in its neighborhood will have similar densities. Conversely, the density for an outlier is considerably different compared to its neighbors. Some examples of this type of model are the *Local Outlier Factor* (LOF) (Breunig et al., 2000), LOCI (Papadimitriou et al., 2003) and DBSCAN (Schubert et al., 2017).

2.2.3.3 Clustering Methods

The main idea is to use a clustering method to segment the dataset into clusters, according to some similarity or distance measure. The assumption is that similar data instances will group to the same clusters. Hence, clusters of inlier data should be separable from outliers. After clustering the data, the centroids representing the normal data should be chosen. Either manually, or using some labelled data (e.g. semi-supervision). Otherwise, one can also train the clustering method using normal data only if available. The anomaly score is thus the average distance of the test instance to the chosen centroids. An example of an *hard-clustering option* can be seen here (K-means) Chawla & Gionis (2013), and a *soft-clustering option* can be found here (GMM) (Kuusela et al., 2012).

2.2.3.4 Kernel Methods

These methods have mostly been developed by the machine learning community. Methods in this category rely on the *kernel trick*, which maps each instance to a high-dimensional feature space, i.e. *kernel space*. These methods are based on SVMs (Support Vector Machines), and so they learn a decision boundary in kernel space that separates normal data from potential outliers. Accordingly, data instances that are outside of the decision boundary will have positive residuals,

whilst interior instances will have negative residuals. Anomaly scores are defined by the residuals after each test instance is projected onto the decision boundary. Usually the user needs to pick a kernel to be used, and a typical one is the Radial Basis Function (RBF).

Two models are the most prominent, those being the *One-Class Support Vector Machine* (OC-SVM) and the *Support Vector Data Description* (SVDD). Note that both of these models are often trained in an OCC setting, where only normal data is used. In this case literature often refers to this as *novelty detection*. Though the term *outlier detection* still applies generally.

OC-SVM (Schölkopf et al., 1999) This method uses an SVM type architecture to learn a decision boundary in kernel space that separates the training data from the origin. This decision boundary is given by an hyper-plane. The fraction of data instances that are allowed to violate this boundary is controlled by an hyperparameter associated with the slack variables of OC-SVM. In fact, these instances are the outliers. Therefore, making OC-SVM robust to the presence of outliers in the training dataset.

SVDD (Tax & Duin, 2004) This method also uses an SVM type architecture to learn the smallest hyper-sphere (decision boundary) in kernel space that encapsulates the normal data. This model is usually used in an OCC setting, where only normal data is provided for learning. Therefore it assumes training data is clean, without corruption. Of course this is not true in most cases. So there is an hyperparameter to allow a fraction of the training data to be ignored when learning the decision boundary. Moreover, if the kernel used by SVDD is the RBF then essentially this becomes a version of OC-SVM.

2.2.3.5 Projection-based Methods

These algorithms rely on random projections of the dataset in order to compute the anomaly score. They are mostly prominent in the data mining and machine learning communities. Two algorithms are discussed: *Isolation Forests* (IF) (Liu et al., 2008), and the *Lightweight Online Detector of Anomalies* (LODA) (Pevnỳ, 2016).

Isolation Forests (Liu et al., 2008) The main assumption here is that outlier instances are rare and quite different from normal data. IFs determine if instances

are outliers if they can be readily isolated by random axis-parallel splits. These random splits are carried out in an *isolation tree*. Like Random Forests, IFs is an ensemble of several isolation trees, each a highly random decision tree. The decision points for the random decision trees are chosen by selecting a feature uniformly at random, and then choosing splitting thresholds uniformly at random in the feature value range. Since outliers are easily isolated by IFs, then outliers should have substantially shorter paths in the tree (from root node). Hence, the anomaly score is defined as the average depth of each data instance across trees in the ensemble. Isolation Forests have proven to be one of the most effective outlier detection methods (Emmott et al., 2015). They are great off-the-shelf with few hyperparameters, and have great scalability.

LODA (Pevnỳ, 2016) This is a recently proposed model that relies on projecting the data onto several random Gaussian noise vectors. This is done through the inner product of each data instance and a random noise vector. The main idea is to produce several weak outlier detection models via density estimation (histograms), one for each random projection of the dataset. Note that technically other density estimation methods can also be used. The anomaly score is just the average NLL across the ensemble of weak density estimators. Random projections here not only make cheaper the cost of density estimation in high dimensions, but also make density estimation more robust to outliers when used in ensembles.

2.2.3.6 Reconstruction-based Methods

These methods perform *dimensionality reduction* on the training dataset, and aim to reconstruct back the same data. They may use a training loss that enforces this identity mapping from data instance to data reconstruction, or use some kind of matrix factorization technique. Since these models usually learn to reconstruct normal data well, they can be used to detect outliers by *failing* to properly reconstruct them under the learnt model. The anomaly score typically used is the reconstruction error, which measures the distance between the original instance and the reconstructed one. A higher distance means that instance might be an outlier. ***These models unlike the other classic methods in this section allow for cell outlier detection***. In other words, it is possible to discover which feature(s) have caused the instance to be an outlier.

Classic methods include *Principal Component Analysis* (PCA) (Jolliffe & Cadima,

2016), and any of its variants like *probabilistic PCA* (PPCA) (Tipping & Bishop, 1999) or *kernel PCA* (KPCA) (Mika et al., 1998). The classic PCA is a linear model, whilst KPCA allows for non-linearity through the *kernel trick*. A *robust PCA* (RPCA) (Candès et al., 2011) has also been proposed, and it provides robustness to the presence of outliers in the training dataset. Unlike other PCA methods, which typically will need to train on normal data only to work well. We address RPCA and related models in more detail in Section 2.5.2, and a deep learning variant in Section 2.5.2.2.

Autoencoders (Goodfellow et al., 2016) are also in this model category, offering a non-linear version of PCA through the application of neural-networks (deep learning). In fact, an autoencoder with a linear encoder and decoder will mimic the behavior of classic PCA. Further, in a similar way, VAE (Kingma & Welling, 2014) is the non-linear version of PPCA.

2.2.4 Database Systems Methods

The database research community has also considered the problem of outlier detection, specifically for tabular data and relational databases. The community has not only focused on the problem of traditional outlier detection, i.e. finding outlier instances, but also on the issue of cell outlier detection. The community has also had more focus than the ML community on outliers that are due to corruption of a clean dataset. In this case, outlier detection is often called *error detection*. Most of the work has focused on logic rules type methods as applied to outlier detection, or even data repair. A good survey about the field can be found in Ilyas & Chu (2015), and a shorter one in Chu et al. (2016).

Generally two types of methods have been explored: *rules-based* and *pattern-based*. Though in last few years the community has also been incorporating machine learning into to their solutions.

Rules-based These models rely on logic-based data quality rules that are either supplied by the user, or obtained by using some software (a *rule miner*) to distill them from clean data. These rules in literature are often called *integrity constraints* (ICs), and tend to be a subset of *first-order logic*. These rules specify constraints on feature values that only hold for inliers. They can capture errors such as instance duplication, missing values and inconsistencies in feature values (cell

outliers). Most often a program is written that enforces these rules on a tabular dataset. This program then finds a violation of the rule by a set of instances, and potentially may highlight the outlier cells that do not conform to the rule. Still inferring *which cells are to blame* is a complex problem, and the rule set may not be specific enough to provide that level of detail. Popular examples include *functional dependencies* (FDs) (Kolahi & Lakshmanan, 2009), *conditional functional dependencies* (CFDs) (Fan et al., 2008) and *denial constraints* (DCs) (Chu et al., 2013b). All of these represent a subset of first-order logic, where the most flexible one are DCs. Both FDs and CFDs express a "if-then" relationship (dependence) between features of a dataset. CFDs are more specific since they only apply when certain feature values appear in the data instance.

A well rounded survey on just integrity constraints for data cleaning can be found in Fan (2015), where both instance and cell outliers are explored. Examples of rule miners can be found in (Fan et al., 2010) for CFDs, and in (Chu et al., 2013a; Pena et al., 2019) for DCs.

The main advantages of the rules-based approach are:

- ***Interpretability*** If the user is a subject-matter expert then the violated rules can potentially explain why the instance is an outlier. For instance, in the case of a CFD it can be seen as a violation of a dependence relationship between features.
- ***Specificity*** Rule-based approaches allow the description of what constitutes an outlier in specific detail, which allows for guarantees that if a violation exists then the outlier will be reported. Machine learning methods for outlier detection usually do not allow for such guarantees, even in the case of training data with a large amount of labelled examples.
- ***Summarization*** These methods often operate by having a set rules that describe the normal data, thus summarizing all its defining patterns. Alternatively, the rule set may only summarize the outlier detection process.

Nonetheless, there are practical issues when trying to use rule-based systems. The more pressing issues are:

- ***Sourcing the Rules*** Obtaining the rules can be difficult, and either one is a subject-matter expert that can write first-order logic to define the rules;

or instead, a *rule miner is used on some uncorrupted training data to obtain these rules.* This uncorrupted data is often called a clean master database.

- ***What cell to blame?*** Assume the outlier process is due to corruption, and one wishes to perform cell outlier detection given a set of rules. It is not always possible to specify which cell(s) are to blame once an outlier instance is found to violate the rules. This is because several different combinations of cells being corrupted may have originated the rule violation. We may never know which without manual inspection, if that. However, when several rules are applicable to the outlier instance, then it may be possible to identify the corrupt cell(s) through logical inference.

Pattern-based These methods rely on user-defined transformation, which can be defined in a scripting language or some domain specific language (DSL). These transformations are in essence programs that provide a way to detect outliers, either by defining what is an outlier or instead an inlier. Sometimes the user may also define how to transform an outlier into a inlier – i.e. data repair. As such, they are far more broad in their coverage than integrity constraints. These can define semantic, statistical or even logic patterns to be applied to the dataset. They can be used in interactive fashion to let the user explore the dataset and perform manual data cleaning – i.e. *data wrangling* task.

Typical use cases would be a script in a DSL that defines: the correct format for a categorical feature; a change in date-time format in column for tabular data; an upper or lower bound in a continuous feature; a check on whether continuous feature is in correct scale unit; a check on whether missing value entries are present.

A great example of this type of method is the Data Wrangler (Kandel et al., 2011). Another examples is Google's OpenRefine (Verborgh & De Wilde, 2013). KATARA (Chu et al., 2015) offers an automated framework for data cleaning with semantic patterns instead of traditional rules. In this case, semantic patterns are obtained from DBpedia. NADEEF Dallachiesa et al. (2013) also allows for user-defined transformations.

Frameworks-based (Holistic) There is also a tradition of work on end-to-end frameworks, or software, that focuses on combining different existing outlier detection methods to provide a novel solution, which can be superior to just a

single model. These can *combine machine learning, data quality rules and pattern-based approaches together*. This contrasts with machine learning research which mostly focus on novel model approaches. Often in framework papers, practical considerations like deployment and computational complexity are taken more seriously. Moreover, in the same spirit of end-to-end frameworks, some works also tackle the problem of outlier detection and data repair together.

Some examples of these types of frameworks are DBoost (Mariet et al., 2016), which integrates several classic ML outlier detection methods; NADEEF (Dallachiesa et al., 2013) integrates different types of rules (e.g. CFDs) and user defined transformations for cell outlier detection and data repair; BoostClean (Krishnan et al., 2017) proposes an ensemble of methods via *statistical boosting* for cell outlier detection, and for repair using user-defined functions.

2.2.5 Deep Learning Methods

Outlier detection can prove quite challenging when nonlinear or hierarchical dependencies between features are present in the dataset. An example is the dependencies between pixels in an image, and how groups of pixels are arranged. For instance, an image of a human face has specific dependencies between pixels. High dimensional data is also challenging, since scalability of the method becomes important and learning abstractions is often needed for classification, e.g. outlier detection. Further, in high dimensions typical statistical distances start to be ineffective, e.g. Mahalanobis distance. All of these are problems for classic ML methods in outlier detection. Unsurprisingly, Isolation Forests which is a nonlinear method tends to outperform a lot of other classic methods. It also scales quite well in high dimensions and size of the dataset. The reality is that most classic outlier detection methods are linear, or lack the ability to capture hierarchical dependencies between features. This makes classic methods not as competitive for high dimensional tabular or image data, more details in Zimek et al. (2012). A good discussion about classic methods (shallow) and deep models is found in Ruff et al. (2021). Relatively recent surveys about deep learning for outlier detection are found in Ruff et al. (2021); Pang et al. (2021); Chalapathy & Chawla (2019).

One effective way to learn hierarchical or nonlinear relationships between features is to use neural networks, i.e. deep learning. Another aspect is that deep learning is quite flexible and can be applied to different types of data. Examples of deep

learning models exist from tabular data, to images, to speech, to natural language, and even music data. General surveys about deep learning and its applications are Dong et al. (2021), and Pouyanfar et al. (2018). A great book on the topic is Goodfellow et al. (2016). Deep learning for tabular data is discussed in Borisov et al. (2021), where pros and cons are presented. The survey also discusses which tabular data tasks are better modelled by deep learning.

As discussed early on (in Section 2.2) supervised, semi-supervised, and unsupervised scenarios are possible. The latter two being the most common. For the supervised scenario, a deep learning classifier should be used. In the semi-supervised scenario often this means an autoencoder model combined with some classifier or classic outlier detection method. For instance, in Ruff et al. (2019) a semi-supervised model relies on autoencoder pre-training, and then reuses the encoder network for the actual method. In the unsupervised scenario an autoencoder model has typically been used, which may or may not be combined with a classic method.

The autoencoder (AE) (Goodfellow et al., 2016) is a popular and effective deep learning method for *nonlinear dimensionality reduction*, or *representation learning*. A probabilistic version is the variational autoencoder (VAE) (Kingma & Welling, 2014). A significant amount of recent outlier detection methods rely on using the latent space of an AE, or VAE, as input to a classic outlier detection method. This works well if the inliers and outliers are somewhat separable in latent space. If these methods are trained end-to-end they are usually called *hybrid methods*, where the AE is seen as a feature extractor. Some unsupervised examples include combining a GMM with an AE latent space in DAGMM (Zong et al., 2018a); combining One-Class SVM with an AE latent space in AE-OCSVM (Nguyen & Vien, 2018b); and combining robust subspace recovery (RSR) in AE latent space in RSRAE (Lai et al., 2019).

In these hybrid models the anomaly score can be defined by the chosen classic method. However, most autoencoder models use the reconstruction error, or likelihood in VAEs, as the anomaly score. For instance, Dehaene et al. (2019); An & Cho (2015); Wang et al. (2017b) use a reconstruction-type score.

Deep One-Class Classification (OCC) models have also been developed. These can be trained on corrupted data, as long as corruption level is low enough. Though

ideally normal data (inliers only) should be used if available. These methods learn a discriminative decision boundary, like their classic counterparts. Although, these use neural networks to learn very flexible feature maps, which are ideal for capturing nonlinear or hierarchical dependencies. Unlike their classic counter parts, where the kernel feature map is shallow and fixed before training. For instance, Ruff et al. (2018) learns a data enclosing hyper-sphere in feature space like SVDD; and Chalapathy et al. (2018a) learns a hyper-plane in feature space like OC-SVM.

Lastly, deep learning models have been developed for outlier detection in the presence of systematic error corruption. As an example, recently the Transformer based model in Liu et al. (2020) was proposed. This model can also tackle adversarial outliers, which are the result of corruption meant to fool the model. That said, the field of adversarial attacks is out of scope for this thesis.

2.3 Data Repair Task

In this section, we discuss the data repair task in more detail. We define it formally and review prior literature on the topic. Literature from database, statistics, and machine learning communities is explored.

An issue with several of these models is the reliance on either encoding prior knowledge in the model, or substantial labelled data or a clean dataset to train on. Some examples of prior knowledge are: data-specific or corruption-specific prior distributions or architectures; logic rules or data schemas; scripts that define data transformations. *The novel models proposed in this thesis have integrated outlier detection and data repair. Further, they do not require large curated datasets for training or encoding complex prior knowledge.*

2.3.1 Problem Definition for Data Repair

Given the notation in Section 2.1, the task of data repair is that of finding the ground-truth inlier $\tilde{\boldsymbol{x}}$ for some outlier instance $\boldsymbol{x} \in \mathcal{O}$. This assumes the outlier detection task has been carried out, and thus the set of outliers $\mathcal{O}$ has been found. In practice, finding the ground-truth $\tilde{\boldsymbol{x}}$ is generally not possible. Instead, the repair task tries to provide an estimate that is close enough, i.e. $\hat{\boldsymbol{x}} \approx \tilde{\boldsymbol{x}}$. The outlier instance is the result of the application of a *corruption transformation*, i.e.

$\boldsymbol{x} = f_{cr}(\tilde{\boldsymbol{x}})$. Hence, the model g_r^θ will learn how to reverse such transformation and provide an estimate $\hat{\boldsymbol{x}}$. Here we say reverse since the corruption transformation f_{cr} may not itself be invertible.

Generally, taking the above into account, we can define the *repair transformation* as $\hat{\boldsymbol{x}} = g_r^\theta(\boldsymbol{x})$. In practice, the models g_r^θ that are used to estimate an instance repair $\hat{\boldsymbol{x}}$ tend to have an associated loss function (e.g. deep generative models). After all, this may be the same loss function used to train the model. Therefore, after model training, repair estimates are usually produced by optimizing this loss function given outlier $\boldsymbol{x}$. This process will produce a point-wise estimate for $\hat{\boldsymbol{x}}$, however other inference strategies could be used – e.g. MCMC methods like pseudo-Gibbs sampling as described in (Rezende et al., 2014), and used in Section 3.8.7 of Chapter 3; or instead (Mattei & Frellsen, 2019). Assuming $\mathcal{F}_\theta$ is the loss function, for some ideal model g_r^θ we can define the *repair transformation* as

$$\hat{\boldsymbol{x}} = g_r^\theta(\boldsymbol{x}) = \arg\min_{x'} \mathcal{F}_\theta(\boldsymbol{x}'; \boldsymbol{x}), \tag{2.2}$$

where a lower loss function value corresponds to a more plausible repair estimate under the trained model. This plausibility relates to both how clean the repair instance is, but also how realistic it might be according to other inlier data instances. Again, note that the loss $\mathcal{F}_\theta$ depends explicitly on model parameters θ. That said, one very important point is that g_r^θ must be *robust* to the corrupted training dataset. After training, this is reflected in $\mathcal{F}_\theta$ where small loss values should be seen for inliers, and large loss values for outliers. In other words, this means that g_r^θ places more importance in modelling inlier data than on outlier data.

Similarly, we can express this repair transformation under a probabilistic model framework. In fact, here a point-wise estimate corresponds to either *maximum likelihood* (MLE) or a *maximum a posterior* (MAP) estimate. In this case, we can define $\mathcal{F}_\theta(\boldsymbol{x}'; \boldsymbol{x}) = -\log p_\theta(\boldsymbol{x}'|\boldsymbol{x})$ and thus the *repair transformation* is given as

$$\hat{\boldsymbol{x}} = g_r^\theta(\boldsymbol{x}) = \arg\max_{x'} \log p_\theta(\boldsymbol{x}'|\boldsymbol{x}), \tag{2.3}$$

where inliers have high likelihood values under distribution $p_\theta(\boldsymbol{x}'|\boldsymbol{x})$, and outliers have low likelihood.

2.3.2 Database Systems Methods

Quite a few works from the database community take an end-to-end approach to data cleaning, and present a solution for both outlier detection and repair. This is motivated by wanting to provide a real solution to the problem of corrupted data in practice, which includes both outlier detection and repair. A good survey is Ilyas & Chu (2015), and more detailed book on the topic is Ilyas & Chu (2019).

Most methods fall in the framework (holistic) category, see Section 2.2.4, where several approaches are combined: logic rules, transformations by a user or another source, machine learning or statistical methods. This is important to point out, since generally logic-based rules (integrity constraints) by themselves are not able to provide a repair proposal for a corrupt cell. Instead, they use other user-defined transformations, statistical measures, or even a uncorrupted master dataset to copy from. Having said that, the biggest advantage is that any proposed cell repair needs to be logically consistent with the integrity constraints in place.

It is important to point out that *almost no method here can be applied directly to our problem setting* (see Section 2.1). In this thesis, we assume a corrupt training dataset is either given as is, or with a small labelled set of inliers and outliers. This is because we aim to minimize required user know-how and intervention for method usage. In addition, we aim to develop models that can be applied generally, not just tabular datasets. Therefore, obtaining rules or scripts for use in data cleaning is not practical in our setup. Particularly, most of these rules can only be obtained by substantial user interaction, or access to (clean) master datasets.

These frameworks usually consider three types of repair operations (Fan, 2015): exclude from dataset; delete and replace by a similar instance from the dataset, or from a master dataset; use a cost-function to measure how different the repaired instances are from the corrupted ones. The latter has become much more common recently (Fan, 2015; Ilyas & Chu, 2019), and provides a way to enforce a total cost budget. This allows the user to cap how much change the corrupted dataset undergoes during repair. The cost-functions can be similarity measures, string edit distances, or even statistical distance measures.

Relevant Frameworks

Now we briefly discuss a few relevant works. **ERACER** (Mayfield et al., 2010) is a data-driven approach for missing value imputation, using a Bayesian Network to represent the clean data.

NADEEF (Dallachiesa et al., 2013) performs repair through user-defined transformations, and repairs are logically consistent with data quality rules.

LLUNATIC (Geerts et al., 2013) is a data cleaning framework that considers different kinds of integrity constraints, uses database schemas, and applies a cost-function based approach to repair.

SCARE (Yakout et al., 2013) proposes cell value repairs via a Dependency Network (graphical model) (Heckerman et al., 2000) learnt using clean master data. The proposed cell repairs maximize the log-likelihood of the graphical model, and the user can define a budget for the entire repair process.

BoostClean (Krishnan et al., 2017) is a framework that provides repair by using ensembles of common ML models and user-defined transformations. These ensembles are learnt via boosting (machine learning). It requires a validation dataset that is considered clean (uncorrupted) by the user. This validation set is then used to improve the performance of the ensemble in the data repair task. In essence, each weak model will propose a cell repair, after which a decision is made.

HoloClean (Rekatsinas et al., 2017) proposes a probabilistic model for data repair that incorporates several *signals*. These *signals* range from integrity constraints, dictionaries, knowledge bases, descriptive statistics, user-defined transformations, and others. Emphasis was given to denial constraints and descriptive statistics, which need to be sourced by the user. This probabilistic model is similar to Markov Logic Networks (Richardson & Domingos, 2006), a type of graphical model that incorporates first-order logic. HoloClean has become a popular framework for data repair, and until recently registered top performance compared to other frameworks.

2.3.3 Statistical and Machine Learning Methods

Data repair as defined in Section 2.3.1, where the model is learnt on corrupt data, has not been explored that much. Most existing machine learning models able to

repair data either assume:

1. ***Complex prior assumptions or model architectures that only apply to certain types of corruption, or in certain types of data.*** For instance, prior distributions for specific types of corruption in images; or data descriptions (schemas, logic, etc) for tabular data.
2. ***Require large amounts of labelled data*** *(inliers vs outliers).*
3. ***Require clean data*** *(without corruption) to train on.*

2.3.3.1 Specific to Tabular Data

Particularly, very few works in machine learning have focused on data repair for tabular data, where the training data is corrupted. Recently, some works have focused on melding probabilistic models with either logic rules or user-written programmatic descriptions of clean data. For instance, HoloClean (Rekatsinas et al., 2017) is an example of probabilistic relational models for data repair, after erroneous cells have been detected. On the other hand, PClean Lew et al. (2021) defines a probabilistic programming language and inference for repair of already detected erroneous cells. In fact, in PClean the program written by the user defines a probabilistic relational model (Friedman et al., 1999), such that the dataset schema is encoded.

From a different perspective, if the erroneous cells have been detected and the majority of data is uncorrupted, then *data imputation* methods can be used. Note that this is generally not possible without some method to detect said corrupt cells. Furthermore, this outlier detection method would need to be trained on corrupted data. Having said that, over the years several works have focused on the problem of imputing missing values for tabular data. Classic work includes Dempster et al. (1977) that defines the Expectation Maximization algorithm for missing data at random. Another classic example is the use of Markov Chain Monte Carlo (MCMC) for the same purpose (Gilks et al., 1995). Recent papers on more traditional probabilistic methods include Su et al. (2011); Kropko et al. (2014). In terms of deep learning techniques, we point the reader towards Section 2.4 where examples of generative models for data imputation are provided.

2.3.3.2 Specific to Image Data

For image data, several tasks are related to data repair. Two of the most relevant are *image denoising* and *image inpainting*. The SOTA in these two tasks is currently dominated by deep learning models. Often the recently proposed machine learning models can tackle both of these tasks, but often rely on large amounts of labelled data or training on clean data. A good example is the recent model for image denoising in Wan et al. (2020), which relies on the availability of a large curated dataset. One way to bypass the above lack of data is to make strong assumptions about the type of noise to repair. For instance, assuming the noise is Poisson-Gaussian distributed (Khademi et al., 2021) (medical imaging). Another example is to assume the image noise is mostly limited to high frequency signals (Bao et al., 2013), and thus a low-pass filter is applied. However, the proposed models in this thesis aim to be broadly applicable, and thus do not make this specific type of assumptions about noise.

The task of image denoising is to remove the noise from an image at test time, and generate a new image with the correct pixel values. This new image should conform to what is considered normal or clean data. A survey about the topic is Tian et al. (2020).

The task of image inpaiting is similar in spirit with that of data imputation for tabular data. Here the missingness mask is usually known apriori, and the task is to impute missing pixels such that the overall generated image conforms with clean data. A survey about the topic is Jam et al. (2021).

2.3.3.3 Regularization and Data-Reweighting

These types of models either use strong regularization of model parameters, or use data reweighting for the training dataset. In general these tend to be unsupervised generative models, and most of the current ones are deep learning models. The data-reweighting methods usually control the importance of an instance towards the training loss or likelihood. In other words, potential outliers should be downweighted in the training loss. Both regularization and data reweighting try to mitigate the effects of corruption in the learning procedure, and thus avoid overfitting to outliers. A good example of regularization is the ℓ_2 norm penalty (Golub et al., 1999) for model parameters. For data-reweighting, a classic example

is the RPCA Candès et al. (2011); Zhao et al. (2014) family of models. A more detailed discussion about these models is seen in Section 2.5.2, although specific for AE / VAE models. The novel model proposed in Chapter 3 can be seen as part of this family of models.

A related method is to use the RANSAC (Fischler & Bolles, 1981) algorithm to continuously isolate outliers, and then retrain the underlying model. Every iteration outlier detection is performed, and outliers removed from the training dataset. Therefore, with successive iterations of RANSAC, the underlying model would only learn from inliers. Hence, in the end model parameters are unaffected by outliers or corruption. This type of approach can be very expensive computationally, as the model needs to be retrained several times for the dataset. This is particularly problematic for deep generative models, which often take a long time to train. Therefore, this has not been explored that much in deep learning.

2.3.3.4 Data Sanitization

A different option that has been explored is to just remove the outlier instance from the dataset. Like in the database systems research (see Section 2.3.2), in some cases this is acceptable or even the right decision. In literature this is often called *data sanitization*. For instance, outliers caused by adversarial attacks often need to be removed from the training data. Otherwise the model will overfit to the corruption, and thus make it vulnerable to malicious intent. This technique has also been used to remove instances affected by systematic errors. RANSAC (Fischler & Bolles, 1981) can be seen as an example of this technique if no repairs are generated by the final model. Instead, the outliers are removed and the process terminated. Other model examples which are far superior in terms of robustness to outliers cane be found in Koh et al. (2018); Diakonikolas et al. (2018); Liu et al. (2020).

2.3.3.5 Conditional Generative Models

If labels for inliers and outliers are available, then conditional deep generative models can be used for data repair. Usually, labelled data is limited and thus semi-supervised models are more realistic than supervised ones. A discussion on semi-supervised generative models can be found here Ouali et al. (2020). After a conditional model has been trained, the repair process would reconstruct

the outlier without any corruption patterns by using label information. This is done by flipping the label value to inlier. In Section 2.5.3 we discuss in detail conditional VAEs, which are a type of conditional deep generative models, for both semi-supervised and supervised settings.

2.3.3.6 Latent Disentanglement Models and ICA

Generative models like VAEs learn a latent representation of the data. In theory, in latent space the attributes of an instance can be manipulated before reconstruction by a decoder (generator). This would allow to repair an outlier instance by removing any corruption patterns. This is difficult to do without knowing which latent variables to manipulate and what their effect on the reconstruction is.

Disentanglement models learn a latent space where each variable, or set of variables, is assigned a particular attribute of the data. To guarantee that latent variables (or set of) model different data attributes these models often try to guarantee that the variables are statistically independent. Therefore, it should be possible to separate in the latent space of a VAE the variables controlling if corruption is present and its type, and the variables controlling the attributes of clean data (inliers). Hence, the repair process through latent variable manipulation can be more easily obtained. In Section 2.5.4 we discuss these models in more detail, and present a SOTA baseline to be used in this thesis. The novel model proposed in Chapter 4 can be seen as part of this family of models.

Classically, in signal processing or time-series data these models have been used to isolate and remove noise sources. However, this usually requires a *lot of labelled data* or *prior knowledge* about the clean data or the corruption process. The ICA (Independent Component Analysis) (Hyvarinen & Morioka, 2016) model has been used for these tasks in the past.

2.4 Deep Generative Models

Before moving to specify in more detail VAE models for our task, we briefly discuss different types of deep generative models. We remark that most of the research about deep generative models has been for image, text and speech data. Still, relatively few works have focused on mixed-type tabular datasets. Note that mixed-type datasets are those where different feature types co-exist,

e.g. continuous and categorical tabular dataset (from csv files, excel sheets, or relational databases).

Deep generative models are neural networks with many hidden layers trained to approximate complicated, high-dimensional data distributions. One type of deep generative models are the variational autoencoders (VAEs). The choice to use and develop upon VAEs (Kingma & Welling, 2014) for our task (i.e. outlier detection and data repair) is due to these being ubiquitous, generally easy to understand and to implement. Thus as a first effort into *robust generative models* for automatic data repair it makes sense to start simple, but also to provide an extension onto an already popular and extensively deployed model. In fact, recently VAEs have proved competitive against more sophisticated generative models, producing diverse high quality samples and reconstructions. Some examples of these state-of-the-art (SOTA) VAE models are VQVAE-2 (Razavi et al., 2019b), VDVAE (Child, 2020), VQGAN (Esser et al., 2021), MUSE (Chang et al., 2023).

Broadly, one can identify five types of deep generative models:

1. ***Variational Autoencoders (VAE)*** (Kingma & Welling, 2014, 2019); generally these models *offer good sample diversity and sample speed. The sample quality in general tends to be inferior to other SOTA generative models, but recent VAE models are quite competitive.* Quality tends to be related to *how good* the samples are as it relates to true data samples; usually measured by FID scores or by human evaluation of the samples. Diversity tends to be measured by the negative log-likelihood (NLL), and is a measure of how different are the samples produced by the model. Speed is related to the time / compute complexity of synthesizing a sample. The VAE uses amortized variational inference, where neural encoders parameterize variational distributions, in conjunction with neural decoders that express generative model of the VAE. Training is generally quite stable, however *posterior collapse* (latent space) (Dai et al., 2020) may be a problem, restricting sample diversity and representation learning. This tends to happen especially with powerful decoders, e.g. autoregressive PixelCNN (Van den Oord et al., 2016). That said it can be avoided by using some strategies, e.g. KL annealing (Fu et al., 2019) or others like Razavi et al. (2019a); Mathieu et al. (2019); Alemi et al. (2018). One common criticism about standard VAEs is that their samples can be blurry, often lacking sharpness or detail.

Although this can be vastly mitigated in modern architectures, the recent work in (Bredell et al., 2022) proposes a more universal approach. The authors improve sample quality of VAEs by incorporating a de-blurring convolution operation in the decoder output – i.e. in the covariance of decoder distribution. VAEs are still the subject of many research papers, and some recent SOTA models are quite popular. VQVAE-2 (Razavi et al., 2019b) uses stacked discrete latent space codebooks (like K-means) that are optimized during training, which leads to better diversity with quite good sample quality at test time (though post-processing is needed). The sampling speed tends to be slower as compared to other VAE or GAN models. VDVAE (Child, 2020) uses a sparse Transformer (Vaswani et al., 2017) in the decoder to sequentially combine latent embeddings producing high quality samples. Note that the VDVAE is a type of Hierarchical VAE (Zhao et al., 2017b), where each layer of the encoder / decoder is modelled as a latent variable in a *top-down inference* architecture. Unlike VQVAE-2 the latent bottleneck is not discrete, and thus the sampling is quite fast. It performs as well as other SOTA autoregressive models. More recently, Efficient VDVAE (Hazami et al., 2022) was proposed registering a speed up in training convergence of about 2.6 fold with improved stability, and saving up to 20 times in memory load. VQGAN (Esser et al., 2021) is better than the aforementioned models, combining VQVAE with Transformer neural architectures, plus it uses a discriminator like GANs (realism constraint) to improve on sample quality. The sample quality is as good as the model BigGAN (Brock et al., 2018), one of the best GAN models. Recently MUSE (Chang et al., 2023) has been proposed for the task of text-to-image generation. This model leverages two key concepts: i) masked image modelling with Transformers; ii) VQGAN encoder and decoder architectures are used for both semantic tokenization and high-resolution image generation. MUSE registers performance very close to SOTA performance in terms of image generation. Further, MUSE appears to be significantly faster at sampling than diffusion models (e.g. Latent Diffusion Models, Imagen) and traditional autoregressive models.

2. ***Generative Adversarial Networks (GAN)*** (Goodfellow et al., 2014; Jabbar et al., 2022); generally these *offer great sample quality with high sample speed, but poor sample diversity.* In fact, *training is complex and*

inherently unstable and can be computationally expensive (Brock et al., 2018). GANs tend to be *very flexible models, as no explicit likelihood is needed*. The model defines a generator network that produces synthetic samples; in a way that it fools a discriminator network trying to find which samples are from the dataset, and which are synthetic. Accordingly, this translates to a min-max optimization problem that defines an *adversarial training procedure*. This adversarial procedure is the source of high quality samples, but also training instability. Further, many times it leads to *mode collapse* or lack of convergence (in training). The issue of mode collapse in GANs translates to lack of sample diversity. Examples of models with overall great sample quality can be seen in StyleGAN v2 (Choi et al., 2020), StyleGAN v3 (Karras et al., 2021), InfinityGAN (Lin et al., 2021), and BigGAN (Brock et al., 2018). Yet typically in these models the sample diversity is not on par with SOTA VAE models – e.g. see (Esser et al., 2021). The authors in (Karras et al., 2020) develop a technique termed *adaptive discriminator augmentation* that stabilizes training of GANs in limited data regimes, whilst still offering great sample quality. This approach can be applied to both BigGAN and StyleGAN models. More recently, StyleGAN-XL (Sauer et al., 2022) sets a new SOTA for GANs by improving the modelling, but mostly by improving the training strategy for StyleGANs. Showcasing great performance in high-resolution image synthesis for large diverse datasets.

3. ***Autoregressive Models (AutoReg)*** (Van Oord et al., 2016; Germain et al., 2015); generally these models *offer good sample quality and diversity (NLL), though sampling speed is quite slow*. These models are characterized by sequentially generating each pixel (feature) when producing a sample. For each feature, the model conditions on the ones already generated. Moreover, the joint distribution for (exact) likelihood-based training is given by the product of conditional probability distributions (for each feature), as in Germain et al. (2015); Uria et al. (2016); Papamakarios et al. (2017) these can be parametrized by neural networks. Nowadays, many other models incorporate *AutoReg* architectures into their neural architectures or even VAE priors, e.g. based on PixelCNN (Van den Oord et al., 2016). On a related note, Transformers have since started to be used (see VQGAN (Esser

et al., 2021)), which allow for non-sequential processing of features (pixels). This can make training more parallelizable and allow for better modelling of long dependencies between features. Transformers rely on multi-head self-attention layers, which in practice provide increased performance on sample diversity and improved speed, see Sparse Transformer with DistAug (Jun et al., 2020) and MaskGIT Chang et al. (2022). The former model (Jun et al., 2020) takes a Sparse Transformer (Child et al., 2019) and applies transformation functions for more aggressive (in its distortion) data augmentation, registering quite good FID scores. Note that the Sparse Transformer was especially designed for long sequence generation. The more recent MaskGIT model is better at image generation than Sparse Transformer with DistAug. MaskGIT learns to predict randomly masked tokens by attending to tokens in all directions. At inference time, MaskGIT begins with generating all tokens of an image simultaneously, and then refines the image iteratively from the previous image generation. This type of masked image modelling has been used in other model types, like the SOTA VAE model MUSE (Chang et al., 2023) already discussed.

4. ***Energy-based Models (EBM)*** (LeCun et al., 2006; Song & Kingma, 2021); generally these *offer good sampling quality and diversity, with slow sampling speed.* The model *training is stable compared to VAEs and especially GANs, but computationally expensive.* EBMs are typically trained via maximum likelihood estimation (MLE), through gradient-based optimization. The model is defined by the unnormalized probability distribution (energy), which provides significant model flexibility compared to VAEs. However, it also means the partition function needs to be approximated through sampling, or at least a few fantasy samples need to be synthesized from the energy. This *makes training computationally expensive and difficult to scale*, as sampling relies on Markov Chain Monte-Carlo (MCMC), and chain mixing issues become a concern. Contrastive Divergence (Carreira-Perpinan & Hinton, 2005) tries to mitigate this. Some of these issues seem to have been improved dramatically recently by using Langevin dynamics MCMC (Du & Mordatch, 2019; Du et al., 2020), especially in high dimensional datasets (e.g. images). Recently VAEBM (Xiao et al., 2020) composes a hierarchical VAE with an EBM model for image generation. The EBM

modelling part helps to exclude non-data like regions (realism constraint), and helps refine image samples for increased quality. On the other hand, the VAE model part helps speed up the MCMC updates by reparametrizing them in a pre-trained VAE latent space. Other authors have proposed *score matching* training methodologies for EBMs (Song & Kingma, 2021), further improving on all aspects (quality, speed and diversity).

5. ***Normalizing Flows (NF)*** (Papamakarios et al., 2021; Rezende & Mohamed, 2015); generally these *offer good sample quality and diversity, but sampling speed is slow.* Further, usually *training is not very scalable* to high dimensional datasets, e.g. high resolution images. Flow-based models restrict their architectures to invertible neural networks, i.e. have a reversible transform. This is necessary so as to apply the *change of variable trick* to the data probability distribution in a computationally tractable fashion – i.e. determinant of Jacobian being constant or cheaply computed. In turn, this allows for exact MLE training via gradient-descent methods, by using the probability distribution of the latent variables in the log-likelihood. Some recent SOTA flow-based models can be found in Ho et al. (2019); Kingma & Dhariwal (2018); Durkan et al. (2019); Chen et al. (2019); Mahajan et al. (2020). However, given current advancements in other generative model families, and how computationally expensive flow-based models still are, the development has been slower for the image generation task.

6. ***Diffusion or Score-based Models (DiffM)*** (Ho et al., 2020; Song & Ermon, 2019); generally these *offer the best sample quality and sample diversity, though sampling speed is slower* than GANs and VAEs. Although, recent works have significantly improved sampling and training speed of diffusion models. Recently, these models have become a topic of interest, and are an active research topic. Diffusion models are inspired by non-equilibrium thermodynamics, as they define a Markov chain of diffusion steps that slowly add random noise to data. The reverse of this diffusion (probabilistic) process, which is learnt by a neural network (decoder), synthesizes any desired samples from noise (latent space). The latent space unlike VAEs has the same dimensionality as the original data. Thus it is not an ideal model for representation learning. Recently, a connection between diffusion models and EBMs trained via *score matching* has been made in (Song & Kingma, 2021;

Song & Ermon, 2019). So these can also be called *score-based models*. Using score matching for inference greatly improves speed of training and sampling. The model DDPM v2 (Nichol & Dhariwal, 2021) represented a big jump in performance in terms of sample quality and diversity. Though sampling was still quite slow compared to models like StyleGAN. But still faster than typical EBMs or AutoReg models. More recently, Latent Diffusion Models (LDMs) (Rombach et al., 2022) proposed using a pre-trained autoencoder to perform the diffusion process in latent space, thus improving on sample quality and speed (training and inference). Alternatively, another option is to use Subspace Diffusion (Jing et al., 2022) models for training and sampling speed-up. LDMs are currently the *best SOTA model for sample quality and diversity* for image generation and related tasks (e.g. text-to-image). A popular version of LDMs for the text-to-image task is Stable Diffusion. However, sampling and training speed is still quite expensive compared to models like SOTA VAEs (e.g. MUSE (Chang et al., 2023)). Still, there is a lot of work trying to speed up sampling and training in LDM type models. For instance, Distilled Stable Diffusion (Meng et al., 2022) presents a 20 times speed-up compared to regular Stable Diffusion in sampling time, and is able to generate quality samples in 2 to 4 denoising steps.

Essentially, one can rearrange the above models into two classes:

1. ***Likelihood-based Generative Models: VAE, AutoReg, NF, EBM, DiffM.*** These models learn the probability distribution function directly through (approximate) maximum likelihood. These include autoregressive models and normalizing flows that allow for exact evaluation and maximization of the likelihood function. It also includes VAEs that evaluate and maximize a surrogate of the likelihood function, i.e. a lower-bound of the log-likelihood.

2. ***Implicit Generative Models: GAN.*** These models are those where the probability distribution is implicitly expressed by a model of its sampling process. In this case, the most well-known example are models of the GAN family, where samples from the data distribution are constructed by transforming random Gaussian noise via a neural network (generator). A discriminator network is used to help in training the model.

2.4.1 Deep Generative Models for Mixed-Type Tabular Data

Although most generative model types above can be adapted for mixed-type tabular data; in reality, few papers have explored this type of structured dataset thoroughly. Most models focus on unstructured datasets like image, speech, or natural language data. Here we showcase some relevant generative models specific for (mixed-type) tabular data.

Some papers have focused on the task of tabular data synthesis, where it seems GAN architectures have been most successful. One of the earliest works was Xu et al. (2019) proposing two models TVAE and CTGAN, where the former is a VAE and the latter a GAN. The authors found that CTGAN outperformed all other alternatives in terms of sample quality; including TVAE and traditional (i.e. non-deep learning) Bayesian models. The same research group has recently developed a framework to compare performance generative models for tabular data synthesis (see SDGym repository [2]).

From the database community Table-GAN (Park et al., 2018) was proposed for tabular data synthesis with privacy concerns in mind. The goal is to generate tabular data samples that does not leak sensitive or private information from individuals, i.e. *differential privacy*. PATE-GAN (Jordon et al., 2018) was also proposed as a GAN for tabular data synthesis with differential privacy in mind. Similarly, CTAB-GAN (Zhao et al., 2021) proposes a conditional GAN that improves on certain typical problems in tabular datasets, e.g. data imbalances and long tail issues. Privacy concerns were also evaluated in the paper. DTGAN (Kunar et al., 2021) improves upon CTAB-GAN by proposing a conditional Wasserstein GAN with superior data synthesis (sample quality) whilst enforcing differential privacy constraints.

From another perspective, some papers have focused on devising generative models for *data imputation* in mixed-type tabular datasets. These works are obviously more related to our task of data repair. However, these models have the enormous advantage of training on clean datasets, which are without corruption. Further, the missing entries (or cells) are properly identified before training happens. A good example of an earlier VAE model for imputation is HI-VAE (Nazabal et al., 2020), where the pattern of missingness is assumed to be at random. Another perspective

[2] `https://github.com/sdv-dev/SDGym`

is the model GAIN (Yoon et al., 2018), which proposes a GAN for tabular data imputation when cell values are missing at random. The model MIWAE (Mattei & Frellsen, 2019) proposes a VAE that improves upon HI-VAE by using tighter lower-bound to the data likelihood. This is inspired by the importance-weighted autoencoder (IWAE) estimator in Burda et al. (2016). The result is superior data imputation (or sample) quality. Recently Not-MIWAE (Ipsen et al., 2020) improves on the previous work by assuming that cell missingness patterns may not be at random, and that there is benefit in overtly modelling the missingness pattern as well. GINA (Ma & Zhang, 2021) introduces a VAE model that improves on Not-MIWAE for the same task, also learning the missingness pattern. However, it provides a data imputation process that is less biased (due to missingness pattern); and finally provides guarantees on the identifiability of the true data generating process.

2.5 Deep Generative Modelling with Variational Autoencoders

Here we present in detail a class of deep generative models that have become ubiquitous in machine learning, and are powerful enough for our task of *automatic outlier detection (OD) and data repair*. This class of models is the variational autoencoder (VAE) (Kingma & Welling, 2014) and its subvariants. A good overview of representation learning using deep VAEs is found in Tschannen et al. (2018), and a more recent survey in Kingma & Welling (2019). Once more, all the models and anomaly scores as presented here only apply to detection and repair of *point outliers* (see Section 2.2.1). Besides the standard VAE, we introduce unsupervised and supervised / semi-supervised versions relevant to our task. We briefly discuss latent space *disentanglement* VAEs (Locatello et al., 2019a) and their applicability to this task, and present a semi-supervised state-of-the-art (SOTA) model. Although not quite a VAE, we also discuss an unsupervised deep autoencoder (AE) that extends the robust PCA model (Candès et al., 2011), i.e. robust principal component analysis (RPCA). In fact, RPCA has been used in literature for both outlier detection and data repair (Zhao et al., 2014; Wang et al., 2017b), where the training data is corrupted. Further, the purpose of RPCA is to provide a PCA model that has reduced sensitivity to outliers in the training set,

i.e. *a model robust to dataset corruption.*

Likewise, the *novel models developed in this thesis*, as well as *several explored baselines*, have the *objective of providing a model robust to corrupted data.* However, not all baselines will be able to perform outlier detection and repair, sometimes only detection is possible. Nonetheless, all the AE models in this section will be able to do both detection and repair, but not all will be robust to corrupted data (e.g. standard VAE). For the models here that are robust, there are two major ways to accomplish this: *i)* using *strong enough regularization or data reweighting* in the model, such that it focuses on modelling inlier data and mostly ignores outliers – e.g. RPCA or Akrami et al. (2019a) or Chapter 3 (RVAE); *ii)* using *supervision / semi-supervision by labelling inliers and outliers*, such that the model learns to distinguish them and models the inlier data well – e.g. Kingma & Welling (2014); Joy et al. (2020) or Chapter 4 (CLSVAE). Typically, option *i)* is fine for *random error* corruption in reasonable amounts, whilst *ii)* is preferred when *systematic errors* are present. This is because *systematic errors* exhibit specific patterns that are repeated across the dataset, and thus are easier to overfit to by unsupervised models. Though models using supervision are not necessarily immune.

There is a another option, specially if the amount of outliers is low, which is to apply a RANSAC (Fischler & Bolles, 1981) type algorithm to a VAE. However, like we discussed in Section 2.3.3, methodologies like RANSAC can be computationally expensive due to model retraining; but also standard deep generative models can easily overfit to outliers, which makes RANSAC fail due to poor outlier detection by the model. Therefore, a RANSAC approach is generally not explored.

For the models in this section we use the notation in Section 2.1. The problem definition for outlier detection is found in Section 2.2.2, and the one for data repair is in Section 2.3.1. Once more, we omit the subscript n for convenience.

2.5.1 Standard Variational Autoencoders (VAEs)

A common approach to unsupervised outlier detection is to build a generative model $p(\boldsymbol{x})$ that models the distribution of clean data (inliers only). A powerful class of deep generative models are variational autoencoders (VAEs) (Kingma &

Welling, 2014), which model $p(\boldsymbol{x})$ as

$$p(\boldsymbol{x}) = \int d\boldsymbol{z}\; p(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z}), \tag{2.4}$$

where $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \prod_{d=1}^{D} p_\theta(x_d|\boldsymbol{z})$ and $p_\theta(x_d|\boldsymbol{z})$ is the conditional likelihood of feature d, $\boldsymbol{z} \in \mathbb{R}^K$ is the latent representation of instance $\boldsymbol{x}$, and $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$ is an isotropic multivariate Gaussian prior.

To handle mixed-type data, e.g. tabular datasets, one can use a conditional likelihood $p_\theta(x_d|\boldsymbol{z})$ that is different for each feature type. For real features $p_\theta(x_d|\boldsymbol{z}) = \mathcal{N}(x_d|m_d(\boldsymbol{z}), \sigma_d)$, where each σ_d represents the standard deviation of feature d and these are parameters learnt during model training. For categorical features $p_\theta(x_d|\boldsymbol{z}) = f(\boldsymbol{a}_d(\boldsymbol{z}))$, where $\boldsymbol{a}_d(\boldsymbol{z})$ is an unnormalized vector of probabilities for each category and f is the softmax function. All $m_d(\boldsymbol{z})$ and $\boldsymbol{a}_d(\boldsymbol{z})$ are parametrized by feed-forward neural networks.

Otherwise, for image datasets, if the pixel values are continuous then usually we can model these using $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_\theta(\boldsymbol{z}), \sigma_{\boldsymbol{x}}^2 \boldsymbol{I})$ where $\sigma_{\boldsymbol{x}}$ is a scalar parameter that is learnt. If the pixels are categorical variables, then pixel values have a set number of categories, e.g. a black-and-white image has two categories. In this case each pixel d is modelled by $p_\theta(x_d|\boldsymbol{z}) = f(\boldsymbol{a}_d(\boldsymbol{z}))$ as seen above.

Since exact inference for $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ is generally intractable, a variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is used; in VAEs this is also known as the encoder. It is modelled by a Gaussian distribution like so

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mu_\phi(\boldsymbol{x}), \Sigma_\phi(\boldsymbol{x})), \tag{2.5}$$

where $\mu_\phi(\boldsymbol{x})$ and $\Sigma_\phi(\boldsymbol{x})$ are feed-forward neural networks, and $\Sigma_\phi(\boldsymbol{x})$ defines a diagonal covariance matrix. VAEs are trained by ***maximizing*** the lower bound on the marginal log-likelihood called the *evidence lower bound (ELBO)*, given by

$$\begin{aligned} \log p(\boldsymbol{x}) \geq \mathcal{L}_{\theta,\phi}(\boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right], \\ &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(x|\boldsymbol{z})\right] - \; D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \\ &= \sum_{d=1}^{D} \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(x_d|\boldsymbol{z})\right] - \; D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \end{aligned} \tag{2.6}$$

where the neural network parameters of the decoder θ and encoder ϕ are learnt with a gradient-based optimizer (e.g. Adam (Kingma & Ba, 2014)). Accordingly,

the optimization problem defining model training is thus given by

$$\min_{\theta,\,\phi} \; -\frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{X}} \mathcal{L}_{\theta,\phi}(\boldsymbol{x}). \tag{2.7}$$

In practice, it is common to use a modified version of the ELBO. This version introduces a coefficient η that controls the contribution of the D_{KL} term to the ELBO loss. This is expressed as

$$\mathcal{L}_{\theta,\phi}(\boldsymbol{x}) = \sum_{d=1}^{D} \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(x_d|\boldsymbol{z})\right] - \eta\, D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})), \tag{2.8}$$

where for $\eta = 1$ we obtain the standard ELBO in eq. (2.6). This version can be used for latent space disentanglement like in β-VAE (Burgess et al., 2018), where η is fixed to some value (different than 1) during training. Alternatively, it can be used for annealing of the D_{KL} term (Fu et al., 2019), where η changes its value at each epoch during training, usually monotonically increasing. The annealing procedure can help in preventing posterior collapse, and obtaining better latent representations. It can sometimes also improve reconstruction quality.

Outlier Detection and Repair Process

A typical *anomaly score* for VAEs, see outlier detection task definition (section 2.2.2), is to use the reconstruction loss (negative log-likelihood) of the autoencoder (Dehaene et al., 2019; An & Cho, 2015; Wang et al., 2017b). This is quite common as an anomaly score for *point outliers*, which is the main focus of this thesis. As such, we can define the anomaly score as

$$\mathcal{A}^\theta(\boldsymbol{x}) = -\sum_{d}^{D} \log p_\theta(x_d|\boldsymbol{\mu}_\phi(\boldsymbol{x})) \quad \boldsymbol{x}\in\mathcal{X}, \tag{2.9}$$

and thus the outlier set of instances is $\mathcal{O} = \left\{\boldsymbol{x}\in\mathcal{X}|\; \mathcal{A}^\theta(\boldsymbol{x}) \geq \gamma\right\}$, where γ is given by the user or tuned using a validation set. In the case of cell outlier detection a different score needs to be used, following Section 2.2.2, for VAEs a cell anomaly score using the reconstruction is defined as

$$\mathcal{A}^\theta_d(\boldsymbol{x}) = -\log p_\theta(x_d|\boldsymbol{\mu}_\phi(\boldsymbol{x})) \quad \boldsymbol{x}\in\mathcal{X}, \tag{2.10}$$

and so the cell (pixel) x_d is considered an outlier if $\mathcal{A}^\theta_d(\boldsymbol{x}) \geq \gamma_d$, where γ_d is either tuned via validation set or given by the user.

In terms of the repair task, following Section 2.2.2, we need to define the repair transform that allows to clean the instance. Accordingly, for VAEs we can define a point-wise estimate (MAP) from the VAE decoder as follows

$$\hat{\boldsymbol{x}} = g_r^{\theta}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}'} \log p_\theta(\boldsymbol{x}'|\boldsymbol{\mu}_\phi(\boldsymbol{x})) \quad \boldsymbol{x} \in \mathcal{O}, \tag{2.11}$$

Indeed, this is just the standard autoencoder reconstruction, where for continuous features this corresponds to $m_d(\boldsymbol{\mu}_\phi(\boldsymbol{x}))$; and for categorical features this is just the category with higher probability from $f(\boldsymbol{a}_d(\boldsymbol{\mu}_\phi(\boldsymbol{x})))$. Note that when repairing $\boldsymbol{x}$ we can replace the entire instance by $\hat{\boldsymbol{x}}$; conversely, we can just repair the cells in $\boldsymbol{x}$ that are outliers (i.e. x_d where $d \in \mathcal{C}$, in Section 2.2.2) replacing them by the appropriate cells $\hat{x}_d$. If dealing with image datasets, then we can simplify the repair transformation as follows

$$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x})) \quad \boldsymbol{x} \in \mathcal{O}. \tag{2.12}$$

2.5.1.1 Clarifying the Repair Process Formulation

Now we show how to obtain the formulation for the repair estimate in eq. (2.11). This is an optional section for the reader, and some may think it is already clear enough. That said, this is included here for the sake of being thorough. The VAE is a probabilistic model, we can obtain a point-wise estimate for $\hat{\boldsymbol{x}}$ by following eq. (2.3) in Section 2.3.1. Accordingly, we can specify $\log p(\boldsymbol{x}'|\boldsymbol{x})$ from eq. (2.3) and further simplify it as follows

$$\begin{aligned} \log p_\theta(\boldsymbol{x}'|\boldsymbol{x}) = \log \mathbb{E}_{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\left[p_\theta(\boldsymbol{x}'|\boldsymbol{z})\right] &\geq^a \mathbb{E}_{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}'|\boldsymbol{z})\right] \\ &\approx^b \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}'|\boldsymbol{z})\right] \propto^c \log p_\theta(\boldsymbol{x}'|\boldsymbol{\mu}_\phi(\boldsymbol{x})) \end{aligned} \tag{2.13}$$

where (a) uses the property of Jensen's inequality; (b) is obtained by approximating intractable distribution $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ by the variational analogue $q_\phi(\boldsymbol{z}|\boldsymbol{x})$; and (c) uses $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \delta\{\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x})\}$ for tractability. Note that $\delta\{.\}$ is the Dirac delta distribution. Therefore, we can use the simplification from above to obtain the repair estimate

$$\hat{\boldsymbol{x}} = g_r^{\theta}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}'} \log p_\theta(\boldsymbol{x}'|\boldsymbol{x}) \approx \arg\max_{\boldsymbol{x}'} \log p_\theta(\boldsymbol{x}'|\boldsymbol{\mu}_\phi(\boldsymbol{x})), \tag{2.14}$$

which is thus used in eq. (2.11) as the MAP estimate.

2.5.1.2 Standard VAE not Robust to Corrupt Data

One issue with the standard VAE model is that it has no means to downweigh the importance of outlier instances on the training loss (in this case the *negative ELBO*). The end result is that for a high capacity VAE, as seen by a bigger amount of layers or units in the encoder / decoder neural networks, the model will easily overfit to the outliers; and thus learning the errors as if they were clean patterns of an inlier instance.

Therefore, since both inlier and outlier instances may end up having the same reconstruction loss values, distinguishing them becomes harder, and thus outlier detection is compromised. This is worsened when the outliers in the corrupted dataset are more frequent, which means there is greater gain for the training loss to learn the dirty patterns (errors) that constitute outliers.

Moreover, since the VAE has overfitted to the outliers, then its repair process is also compromised. The VAE model, as reflected by the training loss, will assume outliers are indeed part of the clean subset of the data. Accordingly, the decoder (generator) will reconstruct back the outlier instance with very little changes. This results in a poor quality repair, where the errors are still present.

2.5.1.3 Using the Likelihood as Anomaly Score

Generally, it is common to see the likelihood (or negative log-likelihood) of the data being used as an anomaly score. Indeed, several models tackling the problem of traditional outlier detection have used it successfully in the past (Ruff et al., 2021). Nonetheless, it should be noted that using the likelihood as an anomaly score has some known issues, mostly seen under unsupervised generative models.

One case is that of out-of-distribution outliers (see Section 2.2.1), where it can present very poor performance. This issue with out-of-distribution outliers was first properly shown in Nalisnick et al. (2018) for deep generative models that rely on *maximum likelihood estimation* either exactly (e.g. Normalizing Flows, Autoregressive) or through surrogates losses (e.g. VAEs). The authors believe that this is a fundamental limitation of high-dimensional likelihoods as scores, such as the ones used in some image datasets. The models tested seemed to be focusing on low-level statistics rather than high-level semantics when trying to score an outlier instance.

Another known issue is with outlier instances that cluster together (e.g. group outliers, see Section 2.2.1). In Koh et al. (2018) the authors discuss how loss-based (e.g. log-likelihood) can overfit to these outliers, as it would decrease the average negative log-likelihood loss significantly during training. Hence, the anomaly score produced could be compromised if proper steps are not taken, like regularization or data sanitization techniques.

More recently, the authors in Lan & Dinh (2020) presented a theoretical framework showcasing through the lens of reparameterization several issues with the likelihood as an anomaly score. The paper focuses on several adversarial or failure cases for this type of anomaly score. The authors remark that outlier detection can be an ill-posed problem, and without proper inductive biases (e.g. supervision) some of these failure modes can happen. Furthermore, the authors suggest that *density ratio scores* that compare estimated inlier and outlier distributions to be more robust, and can mitigate several of the issues. Interestingly, the unsupervised RVAE model (see Chapter 3) proposes a ratio score like this with good performance in cell outlier detection.

2.5.2 Unsupervised AEs: Regularization or Data Reweighting

In this section we present two unsupervised autoencoder (AE) models. Each one represents an example of a type of unsupervised autoencoder that can be used in our task. Three assumptions are typically made when using these models: inliers are more frequent in the dataset relative to outliers, so regularization or data reweighting focuses model learning on more general patterns like those of inliers; corruption is random and has no specific pattern that repeats throughout the dataset, i.e. *random errors*; if corruption has a specific repeatable pattern and not random, i.e. *systematic errors*, then the amount of corruption in the dataset should be low (few outliers).

Regularization

The first modelling option is to use an autoencoder where *strong regularization* is applied to make the model robust against corruption. Generally regularization is applied to the model parameters, e.g. neural network weights, which hopefully forces the modelling efforts to focus on inlier data.

One of the most popular approaches is to apply weight decay to the autoencoder neural networks, i.e. Tikhonov regularization (Golub et al., 1999) or ℓ_2 norm penalty on the weights. We explore this option in more detail in Section 2.5.2.1 under the name *VAE-L_2*. Several other interesting approaches to regularized autoencoders exist in literature. *Deterministic autoencoders* (Ghosh et al., 2019) generalize VAEs and introduce a penalty term based on ℓ_2 norm of the gradient of the decoder w.r.t its input. *Contractive autoencoders* (Rifai et al., 2011), use a penalty term based on ℓ_2 norm of the Jacobian (second order derivative) of the encoder activations, tough one could apply it to the decoder instead as well (see Ghosh et al. (2019)). *Sparse (variational) autoencoders* (Ng et al., 2011) use a penalty term that enforces sparsity on the activations of the hidden layers, either activating or deactivating hidden unit contributions. *Denoising (variational) autoencoders* (Vincent et al., 2010) start by injecting corruption to inlier instances, and thus generating synthetic outliers. Then during training the autoencoder learns to reconstruct the inlier using the synthetic outlier. In addition, injecting noise to the hidden layers in an autoencoder has also been explored as means for regularization (Poole et al., 2014). *Concrete (variational) autoencoders* (Abid et al., 2019) learn a discrete number of latent features (patterns) that are combined by the decoder to reconstruct a data instance. As such, a feature selector layer picks which latent features are used by the decoder. The number of features is an hyperparameter, which can be small enough to enforce regularization. In (Shu et al., 2018) regularization is applied to VAEs by restricting the capacity of the encoder. One of the methods proposed therein is a *weight-normalization technique*, defining a family of amortized inference functions for the encoder. A theoretical connection between this proposal and denoising variational autoencoders is provided. Nonetheless, a simpler way to restrict capacity in VAEs is to just decrease the latent space dimensionality.

Data Reweighting

The second modelling option is to use *data reweighting* so as to isolate the contribution of the outliers towards autoencoder training. These models typically introduce robustness to corruption by modifying the loss function, or likelihood of a generative model, such that *outliers are down-weighted in importance (or ignored)* and *inliers up-weighted*. This means the model focuses on learning inlier data, however models have *different ways to down-weight or isolate outliers* from

inlier modelling.

Generally, three options exist for *data reweighting. One option is to use the training loss to rank instances, and isolate the outliers by removing them from subsequent model training epochs.* Those instances with high training loss get removed, usually a fraction of the entire dataset (hyperparameter). This then avoids overfitting to outliers. The Transformer model (deep learning) in Liu et al. (2020) is a good example, where outliers are removed from training early on in the first few epochs. Theoretically a RANSAC (Fischler & Bolles, 1981) version of a VAE would fit here, however as stated before, there are several practical complications that have precluded this for deep autoencoders (see intro. of Section 2.5).

A second option is to use an adjacent model or variable to account for outlier contributions, this means the principal model focuses on modelling inlier data. For instance, RPCA type models that are autoencoders fit this type quite well. The *Deep RPCA* (Zhou & Paffenroth, 2017) which defines an additional variable to model corruption is a good example. We explore *Deep RPCA* in detail in Section 2.5.2.2, since we use it as a baseline. The model we propose in Chapter 3 (RVAE) is an example of a VAE where its generative model has a specific model component for outliers. Moreover, very recently a probabilistic version of Deep RPCA was proposed in Aizenbud et al. (2021), the problem of outlier detection was explored.

A third option is to use a training loss that is less sensitive to outliers. These loss functions tend to attenuate or truncate contributions from large deviations relative to where most of the data is located. Since we assume most of the data are inliers, then these large deviations are relative to inlier data. In this context, large deviations are caused by outliers. As a result, this type of loss helps the model avoid overfitting to outliers during training. A relevant example is the *absolute error* (MAE) loss that can attenuate larger deviations. MAE is used instead of the problematic *squared error* (MSE) loss, which is typically used in autoencoder reconstruction, since MSE loss is dominated value-wise by large deviations (outliers) resulting in overfitting. A synthesis between these two error losses (MSE and MAE) is given by the Huber loss (Huber, 1992) for robust regression, which is effectively less sensitive to outliers.

Classically, in robust statistics (Huber, 2004), another way is to use heavy-tailed distributions to attenuate outlier contributions – e.g. *t-Student* or *Laplace* or *Cauchy* distributions. However, these can only be applied to continuous feature datasets. Similar to MAE, heavy-tailed distributions when used in the negative log-likelihood loss will make large devitions from inlier data have a lower loss value than typical distributions (i.e. Gaussian). This means gradients (of the loss) are less affected by outliers, and hence gradient-descent like training is less sensitive to outliers. The exploration of heavy-tailed distributions to a wide variety of tasks was explored in Barron (2019), where it was also applied to image synthesis using VAEs.

In Akrami et al. (2019a) a VAE is proposed that uses a β-divergence measure to define a reconstruction loss that attenuates outliers. The β-divergence Regli & Silva (2018b) is a general class of robust divergences that can be used in variational learning when outliers are present.

The model in Wang et al. (2017b) uses a VAE as a recurrent unit (like recurrent neural network) to produce successive reconstructions, until the image is denoised. It does so by iteratively decreasing the variance of the Gaussian likelihood. In essence, outliers are represented by a Gaussian with a larger variance compared to inliers, in a way similar to a heavy-tailed distribution.

2.5.2.1 VAE-L_2

Here we present a regularized version of the standard unsupervised VAE (Kingma & Welling, 2014) that uses the common ℓ_2 regularization (weight decay). Since this is an *unsupervised* model, a practitioner applying this model cannot easily provide any type of supervision to help the model distinguish between outliers and inliers.

The main assumption is that by applying regularization this will have enough of an effect to have the VAE mostly focus on modelling the inlier data. In fact, if we assume a VAE with enough network capacity, then without regularization the model will surely overfit to the errors present in the outliers from the training set. To that end, the practitioner must increase the *strength* of the regularization enough to constrain the weights of the autoencoder network weights.

For this model, this is done by increasing the weight of the regularization term,

which is given by the hyperparameter λ_{ℓ_2}. However, this hyperparameter has to be tuned carefully: too little strength leads to the model still overfitting to the errors; too much strength leads to a poor quality repair since detail might be lost in the decoder reconstruction. This latter issue is due to constraining the neural network weights to a point where the model collapses to mean behavior.

In this model, we regularize both the encoder and decoder weights via ℓ_2 regularization term. This is a standard approach for VAEs using weight decay. Hence, we can write the modified ELBO as

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \eta D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right) + \lambda_{\ell_2} \sum_{w_i \in \mathcal{W}_{\text{AE}}} ||w_i||^2, \quad (2.15)$$

where θ and ϕ are the parameters for the decoder and encoder respectively, and η is used for D_{KL} annealing. The w_i are the parameters for a layer indexed by i, belonging to the set of all VAE layer parameters $\mathcal{W}_{\text{AE}}$. We use this baseline in Chapters 3 and 4.

Note that λ_{ℓ_2} is tuned by using a small validation set, where inliers and outlier instances are known (i.e. y labels). Mostly, the user is looking towards optimizing outlier detection performance (AVPR, Average-Precision), but can also take into account repair metrics (SMSE, Standardized Mean Square Error). Getting a large enough value for λ_{ℓ_2} is important, however too much regularization can degrade the quality of the repair, which is why checking repair performance is important. However ground-truth values $\tilde{\boldsymbol{x}}$ are generally not available as part of the problem setting (see Section 2.1), making this difficult to evaluate besides qualitatively. In Chapters 3 and 4, the hyperparameter λ_{ℓ_2} was tuned using a small labelled set, where repair performance was also taken into account. This was critical for good performance.

Outlier Detection and Repair Process

In terms of the outlier detection process, the VAE-L_2 model uses the same anomaly scores as defined for the standard VAE (see Section 2.5.1). Specifically, the anomaly scores defined in eq. (2.9) and eq. (2.10). In terms of the repair process, again this model uses the same repair transformation as defined for the standard VAE (see Section 2.5.1). The reader should see eq. (2.11) and eq. (2.12) for this.

2.5.2.2 Deep RPCA (or Robust Deep Autoencoder)

Now we explain in more detail the *Deep RPCA* (Zhou & Paffenroth, 2017), which is a type of *robust* unsupervised deep AE. Here we use the name *Deep RPCA* as it reflects its nature in terms of modelling, though its authors named it *Robust Deep Autoencoder*. This model extends the RPCA (robust principal component analysis) by incorporating an autoencoder for the modelling the clean data (inliers). Otherwise, the core idea remains the same, which is to learn a combined model of the underlying inlier (repair estimate) and an error contribution if the instance is an outlier. In fact, RPCA models the error explicitly in order to *downweigh* outlier instance contribution to the modelling of the repair estimate, i.e. *data reweighting*. Deep RPCA is used in Chapter 3 (RVAE) as a baseline.

Classic RPCA (Candès et al., 2011) is a model that expresses the corrupted data as the addition of a low-rank matrix representing the inlier data (repair term), and a sparse matrix representing additive noise (error term). This however tends to work best for continuous features, and not so much for categorical ones (see Chapter 3). Accordingly, this is commonly written as $\boldsymbol{x} = \hat{\boldsymbol{x}} + \boldsymbol{s}$ where $\boldsymbol{x} \in \mathcal{X}$ is the potentially corrupt data instance; $\hat{\boldsymbol{x}}$ is the repair term, as we have seen before; and $\boldsymbol{s}$ is the error term, which is a nearly zero vector if $\boldsymbol{x}$ is an inlier, or clearly non-zero if an outlier. We see that RPCA uses $\boldsymbol{s}$ to isolate any error deviation to the repair estimate $\hat{\boldsymbol{x}}$, and hence the repair estimate should be close to the ground-truth inlier, i.e. $\hat{\boldsymbol{x}} \approx \tilde{\boldsymbol{x}}$.

Like in many applications, it is useful to use a matricial notation for RPCA. The instance vectors $\boldsymbol{x} \in \mathcal{X}$ are stacked vertically to form a data matrix $\mathbf{X}$. Similarly, we have for error term $\boldsymbol{s}$ the matrix $\mathbf{S}$, and for repair term $\hat{\boldsymbol{x}}$ the matrix $\hat{\mathbf{X}}$. Hence, one can express the dataset modelling as $\mathbf{X} = \hat{\mathbf{X}} + \mathbf{S}$. Using a non-zero vector for $\boldsymbol{s}$ tends to be penalized in RPCA, which reflects the assumption that outliers are less common than inliers. In essence, this corresponds to constraining $\mathbf{S}$ to be a sparse matrix in terms of its rows. Moreover $\hat{\mathbf{X}}$ is modelled by a low-rank matrix, like in standard PCA, which defines a lower dimension approximation of the data.

A common formulation (Candès et al., 2011) for the learning procedure (optimiza-

tion problem) for RPCA is

$$\begin{aligned} \min_{\hat{\mathbf{X}},\, \mathbf{S}} \quad & \|\hat{\mathbf{X}}\|_* + \lambda \|\mathbf{S}\|_1 \;, \\ \text{s.t.} \quad & \|\mathbf{X} - \hat{\mathbf{X}} - \mathbf{S}\|_F^2 = 0 \end{aligned} \tag{2.16}$$

where $\| \, . \, \|_1$ is the ℓ_1 norm applied to the matrix element-wise, which is used to control how sparse $\mathbf{S}$ is through λ; $\| \, . \, \|_*$ is the nuclear norm, i.e. sum of the singular values of the matrix, which is used to obtain a low-rank matrix much like PCA data reconstruction; $\| \, . \, \|_F$ is the Frobenius norm, which is used to enforce the constraint that $\mathbf{X} = \hat{\mathbf{X}} + \mathbf{S}$.

In Deep RPCA (Zhou & Paffenroth, 2017) the data repair matrix $\hat{\mathbf{X}}$ (or $\boldsymbol{x}$) is actually modelled via an autoencoder. Therefore, the learning procedure (optimization problem) for Deep RPCA is now defined as

$$\begin{aligned} \min_{\theta,\, \phi,\, \mathbf{S},\, \hat{\mathbf{X}}} \quad & \|\hat{\mathbf{X}} - \mathcal{D}_\theta(\mathcal{E}_\phi(\hat{\mathbf{X}}))\|_2 + \lambda \, \mathcal{R}(\mathbf{S}) \;, \\ \text{s.t.} \quad & \mathbf{X} - \hat{\mathbf{X}} - \mathbf{S} = \mathbf{0} \end{aligned} \tag{2.17}$$

where $\mathcal{E}_\phi(.)$ and $\mathcal{D}_\theta(.)$ are respectively the encoder and decoder neural networks of the autoencoder, and ϕ and θ their parameters to be learnt; further, $\mathcal{R}(\mathbf{S})$ is the regularization loss for the error term, which has its strength controlled by the scalar hyperparameter λ. The value of λ controls the level of sparsity of matrix $\mathbf{S}$, which is tuned or user-defined according to the amount of corruption present in the dataset. A large value for λ assumes the dataset $\mathbf{X}$ has a low amount of corruption, i.e. few outliers, increasing the penalty incurred for using $\mathbf{S}$. Likewise, the reverse is true, a smaller value for λ should be used when large amounts of corruption are present. For proper tuning of λ one should use a small labelled set of outliers / inliers, which is used to gauge outlier detection performance.

The type of regularization loss $\mathcal{R}(\mathbf{S})$ used will dictate how outlier detection is performed, and it should be adjusted to the type of corruption in the data. Firstly, note that S_{nd} is an element of matrix $\mathbf{S}$, and S_n or $\boldsymbol{s}$ is a row of $\mathbf{S}$. The set of all rows of matrix $\mathbf{S}$ is defined as $\mathcal{S}^r = \{\boldsymbol{s} \mid \boldsymbol{s} \text{ is a row of } \mathbf{S}\}$. The set of all elements (entries) of matrix $\mathbf{S}$ is defined as $\mathcal{S}^e = \{S_{nd} \mid S_{nd} \text{ is an entry of } \mathbf{S}\}$. In Deep RPCA two types of regularizers are used, the first is defined by

$$\mathcal{R}(\mathbf{S}) = \|\mathbf{S}\|_1 = \sum_{n=1}^{N} \sum_{d=1}^{D} |S_{nd}| \;, \tag{2.18}$$

which should be used when the corruption affecting cells (pixels) is distributed with no particular pattern. This is also the standard regularizer used in classic RPCA, see eq. (2.16). Another and more interesting regularizer is defined by

$$\mathcal{R}(\mathbf{S}) = \|\mathbf{S}\|_{2,1} = \sum_{n=1}^{N} \sqrt{\sum_{d=1}^{D} |S_{nd}|^2} \,, \tag{2.19}$$

and its most useful when data corruption is due to the presence of outlier instances, which means that dirty cells are concentrated in specific data instances. Further, it assumes most instances are inliers, and thus made up of clean cells. This second regularizer is more relevant for the problem setting we are trying to solve in this thesis (see Section 2.1).

Training the Model

The optimization algorithm used to train Deep RPCA (Zhou & Paffenroth, 2017) is an hybrid version of the original Alternating Direction Method of Multipliers algorithm (ADMM). The classic ADMM (Boyd et al., 2011) is proximal optimization method, ideally used when parts of the training loss are *non-smooth* functions (e.g. ℓ_1 regularizer). It alternates between a typical gradient descent update to the solution, and one or more updates using proximal operators. In the case of Deep RPCA the proximal operators reflect the type of regularizers used, e.g. those in eq. (2.18) and eq. (2.19). The gradient descent step updates the autoencoder parameters ϕ and θ; and the proximal operators update $\hat{\mathbf{X}}$ and $\mathbf{S}$.

Outlier Detection and Repair Process

In terms of outlier detection, one must define an anomaly score for cells and another for instances. According to Zhou & Paffenroth (2017), and assuming $\mathcal{R}(\mathbf{S})$ is given by 2.19, the cell anomaly score is

$$\mathcal{A}_d^{\theta}(\boldsymbol{s}) = |S_{nd}|^2 \quad S_{nd} \in \mathcal{S}^e \,, \tag{2.20}$$

where according to (Deep) RPCA formulation for each $\boldsymbol{x}$ we have an associated $\boldsymbol{s}$. On the other hand, for traditional outlier detection, we need an anomaly score for instances. In Deep RPCA, assuming $\mathcal{R}(\mathbf{S})$ is given by 2.19, the instance or row anomaly score is given by

$$\mathcal{A}^{\theta}(\boldsymbol{s}) = \sqrt{\sum_{d=1}^{D} |S_{nd}|^2} \quad S_n \in \mathcal{S}^r \,. \tag{2.21}$$

The repair estimate $\hat{\boldsymbol{x}}$ is obtained directly by solving the optimization process in eq. (2.17). Thus the optimization problem in eq. (2.17) is the repair process, where $\hat{\boldsymbol{x}}$ is just a row of matrix $\hat{\mathbf{X}}$.

2.5.3 Supervised and Semi-supervised VAEs

In this section, VAE models that have at least some supervision are discussed. The model can either be fully supervised, where all labels $y \in \mathcal{Y}$ are known for the entire training set instances $\boldsymbol{x} \in \mathcal{X}$. Alternatively, only a smaller labelled set is provided, i.e. a *trusted set*. In our problem setting, this trusted set is a subset of the overall training set $\mathcal{X}$. Our problem setting only allows for small labelled sets, as to consider *little user intervention*, though supervised models can be useful as baselines. An instance in this small *trusted set* is given as $(\boldsymbol{x}, y) \in \mathcal{X}_l \times \mathcal{Y}_l$, where inlier ($y = 1$) and outlier ($y = 0$) instances are labelled.

The main idea behind using these models is to exploit any labels y provided for the dataset by the user. This supervision should help the model recognize outliers, learn the patterns of the errors, and distinguish them from inlier instances. This should allow not only for outlier detection, but also a repair process where changing the label to $y = 1$ in the model should generate a repair estimate. *This is particularly relevant when outliers are difficult to detect, e.g. systematic errors which unsupervised generative models can more easily overfit to.* Using semi-supervision improves outlier detection and data repair significantly in this case, which is explored in Chapter 4. In fact, unsupervised robust generative models still are at a clearly increased risk to overfitting to systematic errors. This is because systematic errors result from nearly deterministic transformations (plus potentially some noise) that occur repeatedly in data, unlike random errors.

As a reminder, the reader should take a look at the problem setting notation in Section 2.1. Finally, some models may always need a small labelled set, e.g. validation set, for hyperparameter tuning model.

2.5.3.1 CVAE (Supervised)

The CVAE *(Conditional VAE)* model (Kingma et al., 2014; Sohn et al., 2015) simply concatenates the label y to both the input of the encoder network, and the input of the decoder. This models needs the entire training set to be labelled, i.e.

$(\mathcal{X}, \mathcal{Y})$. The CVAE provides a valuable supervised baseline, where the ground-truth for y is observed for all instances $\boldsymbol{x}$. However, this is typically not possible, as the user would have to label or otherwise provide y for each $\boldsymbol{x}$ for a potentially large dataset. This means this model is not very practical, i.e. *little user intervention* in our problem setting. Once more, our problem setting (Section 2.1) only allows for small trusted sets.

The ELBO for the CVAE model is given by

$$\log p(\boldsymbol{x}, y) \geq \mathcal{L}(\boldsymbol{x}, y) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},y)} \left[p_\theta(\boldsymbol{x}|\boldsymbol{z}, y) \right] - \eta D_{KL} \left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, y) || p_\sigma(\boldsymbol{z}) \right), \quad (2.22)$$

where θ and ϕ are the decoder and encoder parameters, and η can be used for D_{KL} annealing. Note that we have $q_\phi(\boldsymbol{z}|\boldsymbol{x}, y) = \mathcal{N}(\boldsymbol{z}|\mu_\phi(\boldsymbol{x}, y), \Sigma_\phi(\boldsymbol{x}, y))$; and the decoder has the same type of distribution as the standard VAE (Section 2.5.1), apart from conditioning on y.

Furthermore, in a standard CVAE we usually have $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. However, we found empirically that stronger regularization for the encoder was *needed* for CVAE to do well in repair. Otherwise, it would often overfit to errors in outliers (of the systematic type), see Chapter 4. Hence, we use a modified prior $p_\sigma(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ where $\sigma \in [0.1, 0.8]$, thus enforcing stronger regularization (lower message capacity) on latent code $\boldsymbol{z}$. Theoretically, one might think that the encoder / decoder weights may scale in order to cancel out any change in the variance σ. Thus, the optimum model after training on the ELBO loss would always be the same. In practice, this is not the case. For further discussion into this please see Section 4.6.3.1. Even though it is not standard, we also tried modifying the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to not depend on y, but always ended up with worse performance than $q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)$ in data repair.

Therefore, the training loss for the CVAE is defined by

$$\min_{\theta,\, \phi} \; -\frac{1}{N} \sum_{(\boldsymbol{x},y) \in \mathcal{X} \times \mathcal{Y}} \mathcal{L}(\boldsymbol{x}, y) \quad . \quad (2.23)$$

Outlier Detection and Repair Process

In terms of outlier detection, once again an *anomaly score* based on the reconstruction (negative log-likelihood) of CVAE is used. Hence, for traditional outlier detection we have

$$\mathcal{A}^\theta(\boldsymbol{x}) = -\sum_d^D \log p_\theta(x_d|\boldsymbol{\mu}_\phi(\boldsymbol{x}, y = 1), y = 1) \quad \boldsymbol{x} \in \mathcal{X} \,, \quad (2.24)$$

where the outlier set of instances is $\mathcal{O} = \{\boldsymbol{x} \in \mathcal{X} | \ \mathcal{A}^{\theta}(\boldsymbol{x}) \geq \gamma\}$, where γ is given by the user or tuned using a validation set. Again, note that $y = 1$ is the inlier label, which is selected for this likelihood-based score. The idea here is that outliers $\boldsymbol{x}$ should have low likelihood if one conditions the reconstruction distribution to $y = 1$ (inliers). As a result, the anomaly score of an outlier should be high in that case.

Similarly, we can define the anomaly score for cell outlier detection. For some feature d in instance $\boldsymbol{x}$ the score for each cell (or pixel) is

$$\mathcal{A}_d^{\theta}(\boldsymbol{x}) = -\log p_{\theta}(x_d | \boldsymbol{\mu}_{\phi}(\boldsymbol{x}, y = 1), y = 1) \quad \boldsymbol{x} \in \mathcal{X} \ , \tag{2.25}$$

and so the cell x_d is considered an outlier if $\mathcal{A}_d^{\theta}(\boldsymbol{x}) \geq \gamma_d$, where γ_d is either tuned via validation set or given by the user.

Now one must the define the *repair process* using this model. We follow the CVAE attribute manipulation strategy from (Klys et al., 2018), and thus provide a repair estimate for an outlier ($y = 0$). This will produce a point-wise estimate (MAP) for $\hat{\boldsymbol{x}}$. Conceptually, we have the following steps: one encodes $\boldsymbol{x}$ using the original label ($y = 0$); then one switches the label in the latent space ($y = 1$); finally one used the decoder to generate the repair. In other words, the repair estimate is given by

$$\hat{\boldsymbol{x}} = g_r^{\theta}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}'} \ \log p_{\theta}(\boldsymbol{x}' \mid \boldsymbol{\mu}_{\phi}(\boldsymbol{x}, y = 0), \ y = 1) \quad \boldsymbol{x} \in \mathcal{O} \ , \tag{2.26}$$

which can be simplified in the case of an image dataset with continuous features as follows

$$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_{\theta}(\boldsymbol{\mu}_{\phi}(\boldsymbol{x}, y = 0), y = 1) \quad \boldsymbol{x} \in \mathcal{O}. \tag{2.27}$$

2.5.3.2 M2 Model: Standard Semi-Supervised VAE

The classic version of the semi-supervised VAE (Kingma et al., 2014) assumes a labelled subset of the training data is available. Hence in the training data we have a trusted set $\mathcal{X}_l \times \mathcal{Y}_l$, and an unlabelled set $\mathcal{X}_u$ that tends to encompass the vast majority of data instances. This model is often referred to as the M2 model, since that is the original name from the authors. The M2 model shares several characteristics with the CVAE, and can be seen as its semi-supervised version. For this reason often CVAE will register the same or better performance than the M2 model.

The ELBO of the semi-supervised VAE is given by two terms, which account for the semi-supervised setting. The first term $\mathcal{L}(\boldsymbol{x})$ is the ELBO for the unlabelled part of the data, corresponding to $p(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathcal{X}_u$. The second term $\mathcal{L}(\boldsymbol{x}, y)$ is the ELBO for the trusted set, corresponding to $p(\boldsymbol{x}, y)$ where $(\boldsymbol{x}, y) \in \mathcal{X}_l \times \mathcal{Y}_l$.

The generative model for the semi-supervised VAE (M2 model, (Kingma et al., 2014)) is $p_\theta(\boldsymbol{x}|\boldsymbol{z}, y)p(\boldsymbol{z})p(y)$. Note that $p(y) = \text{Bernoulli}(y|\alpha)$ is just the prior distribution of the labels, which defines a prior belief on how corrupt the training dataset is. Specifically, α reflects the fraction of instances in the training set believed to be inliers. For instance, an α close to 1 means that the dataset is mostly without outliers. Further, just like in standard VAE we have $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. The decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z}, y)$ has the same definition as in Section 2.5.1, apart from being conditioned on y. The typical variational model used for this VAE model is $q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)q_\phi(y|\boldsymbol{x})$, where $q_\phi(\boldsymbol{z}|\boldsymbol{x}, y) = \mathcal{N}(\boldsymbol{z}|\mu_\phi(\boldsymbol{x}, y), \Sigma_\phi(\boldsymbol{x}, y))$ and $q_\phi(y|\boldsymbol{x}) = \text{Bernoulli}(y|\pi_\phi(\boldsymbol{x}))$.

According to Kingma et al. (2014), the labelled term of the ELBO is

$$\mathcal{L}(\boldsymbol{x}, y) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},y)}\left[p_\theta(\boldsymbol{x}|\boldsymbol{z}, y)\right] - \eta D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)||p(\boldsymbol{z})\right) + \eta \log p(y) \ , \quad (2.28)$$

where once more note that θ and ϕ are decoder and encoder neural network parameters. The unlabelled term of the ELBO is

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}) =& \mathbb{E}_{q_\phi(y|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x},y)}\left[p_\theta(\boldsymbol{x}|\boldsymbol{z}, y)\right] \\ &- \eta\mathbb{E}_{q_\phi(y|\boldsymbol{x})}\left[D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)||p(\boldsymbol{z})\right)\right] - \eta D_{KL}\left(q_\phi(y|\boldsymbol{x})||p(y)\right) \ .\end{aligned} \quad (2.29)$$

Combining the two terms the overall *negative* ELBO is obtained as follows

$$\mathcal{I}^{\phi,\,\theta}_{-\text{ELBO}} = -\frac{1}{N}\left[\sum_{\boldsymbol{x}\in\mathcal{X}_u}\mathcal{L}(\boldsymbol{x}) \ + \sum_{(\boldsymbol{x},y)\in\mathcal{X}_l\times\mathcal{Y}_l}\mathcal{L}(\boldsymbol{x}, y)\right] \ . \quad (2.30)$$

However, if we look at the overall negative ELBO we notice that the variational distribution $q_\phi(y|\boldsymbol{x})$, also known as classifier, is not being encouraged to predict the labels y given by the trusted set. Thus the overall negative ELBO should not be used alone as a training loss.

As a result, in Kingma et al. (2014) an additional loss term to train the classifier on the trusted set labels is proposed. This term is the *cross-entropy loss* between the true labels $y \in \mathcal{Y}_l$ and $q_\phi(y|\boldsymbol{x})$. This loss term is defined as

$$\mathcal{I}^{\phi}_{\text{CE}} = -\frac{1}{N_l}\sum_{(\boldsymbol{x},y)\in\mathcal{X}_l\times\mathcal{Y}_l} y\log q_\phi(y=1|\boldsymbol{x}) \ + \ (1-y)\log\left(1-q_\phi(y=1|\boldsymbol{x})\right) \ . \quad (2.31)$$

Therefore, combining all loss terms together for the training loss, we define the optimization problem as

$$\min_{\phi,\,\theta} \;\; \mathcal{I}^{\phi,\,\theta}_{-\text{ELBO}} \;+\; \beta\, \mathcal{I}^{\phi}_{\text{CE}}\,, \tag{2.32}$$

where the scalar hyperparameter β controls the amount of up-sampling or importance relative to the ELBO terms. In the case of quite small trusted sets, which is the case for our problem setting, β tends to be moderately large.

Outlier Detection and Repair Process

The outlier detection anomaly scores defined for CVAE in Section 2.5.3.1 can be used here as well. Specifically, we have the anomaly score for cells in eq. (2.25), and the one for instances in eq. (2.24).

However, since a classifier $q_\phi(y|\boldsymbol{x})$ is available we can use it to define an alternative anomaly score for instance $\boldsymbol{x}$. If the classifier is found to be calibrated and the VAE training successful, then it should be strongly considered. In practice, if available we found this type of score to be superior in performance to one based on the decoder likelihood (e.g. like in eq. (2.24)). This type of score is defined simply as

$$\mathcal{A}^{\phi}(\boldsymbol{x}) = -\log q_\phi(y=1|\boldsymbol{x}) \quad \boldsymbol{x} \in \mathcal{X}\,, \tag{2.33}$$

representing the negative log probability of the instance $\boldsymbol{x}$ being an inlier ($y=1$). Like previous anomaly scores, a higher value means instance $\boldsymbol{x}$ is more likely to be an outlier. Again the set of outlier instances is given by $\mathcal{O} = \big\{\boldsymbol{x} \in \mathcal{X} |\; \mathcal{A}^{\phi}(\boldsymbol{x}) \geq \gamma\big\}$. Assuming the classifier is calibrated, we can use $\gamma = -\log(0.5)$ instead of tuning it or having the user define it.

In terms of the data repair process, this is the same exact procedure as defined for CVAE, in Section 2.5.3.1. Particularly, one should look at formulations in eq. (2.26) or (2.27).

2.5.3.3 VAEGMM: Alternative to M2 Model (Sparse Semi-Supervised)

This formulation is based on Willetts et al. (2020), specifically the GM-DGM (Gaussian Mixture Deep Generative Model) presented therein. The authors focused on the problem of applying a semi-supervised VAE in severe cases of sparse semi-supervision (e.g. some classes do not even have labels). The *sparseness*

refers to how small the labelled set is, just *like in our problem setting with quite small trusted sets*. In sparse semi-supervision, this formulation is relevant since posterior collapse of $q_\phi(y|\boldsymbol{x})$ can occur for the original M2 model (Kingma et al., 2014). This in turn leads to poor performance in classification and clustering tasks. Hence, one can think of Willetts et al. (2020) as an improvement on M2 model for the issue above. Below we present the version of GM-DGM from Willetts et al. (2020) applied to our problem setting, which we refer to as VAEGMM. The name VAEGMM reflects the nature of its modelling more explicitly.

Generative Model

The generative model is defined as

$$p(\boldsymbol{x}, \boldsymbol{z}, y) = p_\theta(\boldsymbol{x}|\boldsymbol{z})\ p_\tau(\boldsymbol{z}|y)\ p_\alpha(y), \tag{2.34}$$

where

$$p_\alpha(y) = \text{Bernoulli}(y|\alpha), \tag{2.35}$$

$$p_\tau(\boldsymbol{z}|y) = y\ \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \sigma^2_{y=1}\boldsymbol{I}) + (1-y)\ \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \sigma^2_{y=0}\boldsymbol{I}), \tag{2.36}$$

and decoder $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is the same as in standard VAE (see Section 2.5.1). This model defines a 2-component *Gaussian Mixture Model* w.r.t. $\boldsymbol{z}$, and $\tau = \{\sigma_{y=1}, \sigma_{y=0}\}$ and $\sigma_{y=1} < \sigma_{y=0}$. A sensible range for τ is: $\sigma_{y=1} = [0.2, 1]$; $\sigma_{y=1} = [2, 8]$. This prior defines a 2-component *Richter distribution* (Gales & Olsen, 1999a; Quinn et al., 2008), expressing a type of *heavy-tailed distribution* on $\boldsymbol{z}$. In the context of robust statistics Huber (2004), this type of distribution can be used as a means to robustify (regularize) Barron (2019) the parameters of a model to outliers. The main idea here is that inliers $\boldsymbol{z}|y = 1$ will be regularized more strongly compared to outliers $\boldsymbol{z}|y = 0$, as reflected by $\sigma_{y=1} < \sigma_{y=0}$. In our experiments, we initially *tried learning the parameters of the Gaussian components* for $\boldsymbol{z}$, as suggested in Willetts et al. (2020). However we obtained *much better results by fixing their parameters*, i.e. mean vector and covariance matrix, particularly when it came to outlier detection in small trusted sets. Like in CVAE, see Section 2.5.3.1, one might think that changing the variances of the prior for $\boldsymbol{z}$ has limited or no impact. Since the encoder / decoder weights may scale in order to cancel the changes in $\sigma_{y=1}$ or $\sigma_{y=0}$. In practice, this is not the case, and we registered good performance by scaling these. For further discussion into variance scaling (of $\boldsymbol{z}$ prior) and its

impact please see Section 4.6.3.1. Lastly, once again α reflects the initial belief on the fraction of clean data.

Variational Model

The variational model uses the standard formulation provided in Willetts et al. (2020), which is the same as in the original M2 model (Kingma et al., 2014). Accordingly, the encoders

$$q(\boldsymbol{z}, y|\boldsymbol{x}) = q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)\ q_\phi(y|\boldsymbol{x}), \tag{2.37}$$

where

$$q_\phi(y|\boldsymbol{x}) = \text{Bernoulli}(y|\pi_\phi(\boldsymbol{x})), \tag{2.38}$$

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}, y) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}, y), \boldsymbol{\Sigma}_\phi(\boldsymbol{x}, y)), \tag{2.39}$$

such that $\pi_\phi(\boldsymbol{x})$, $\boldsymbol{\mu}_\phi(\boldsymbol{x}, y)$ and $\boldsymbol{\Sigma}_\phi(\boldsymbol{x}, y)$ are neural networks.

Training Loss

The ELBO (Evidence Lower Bound) for the unlabelled part of the dataset $\mathcal{X}_u$ is

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}) =& \mathbb{E}_{q_\phi(y|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x},y)}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \\ &- \eta\mathbb{E}_{q_\phi(y|\boldsymbol{x})}\left[D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)||p_\tau(\boldsymbol{z}|y))\right] - \eta D_{KL}(q_\phi(y|\boldsymbol{x})||p_\alpha(y)),\end{aligned} \tag{2.40}$$

which can be rewritten in a different fashion as

$$\mathcal{L}(\boldsymbol{x}) = \pi_{\phi(\boldsymbol{x})}\mathcal{G}(\boldsymbol{x}, y=1) + \left(1 - \pi_{\phi(\boldsymbol{x})}\right)\mathcal{G}(\boldsymbol{x}, y=0) - D_{KL}(q_\phi(y|\boldsymbol{x})||p_\alpha(y)), \tag{2.41}$$

where

$$\mathcal{G}(\boldsymbol{x}, y) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x},y)}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \eta D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}, y)||p_\tau(\boldsymbol{z}|y)), \tag{2.42}$$

and $\pi_{\phi(\boldsymbol{x})} = q_\phi(y=1|\boldsymbol{x})$ is the probability a data instance is clean.

Accordingly, the ELBO for the labelled part of the dataset (trusted set) $\mathcal{X}_l \times \mathcal{Y}_l$ is as follows

$$\log p(\boldsymbol{x}, y) \geq \mathcal{L}(\boldsymbol{x}, y) = \mathcal{G}(\boldsymbol{x}, y) + \eta \log p_\alpha(y). \tag{2.43}$$

Taking the above formulations for the ELBOs of the unlabelled and labelled (trusted set) subsets, then these can be used to form the dataset ELBO. The

negative ELBO loss for the M2 model also applies here(see eq. (2.30)). Further, like for M2 model we need to encourage $q_\phi(y|\boldsymbol{x})$ to predict the labels of the trusted set. Thus a cross-entropy loss like eq. (2.31) is needed. Lastly, putting it all together the overall training loss is exactly the one for M2 model, as defined in eq. (2.32).

Outlier Detection and Repair Process

After training the model, we can then proceed with the outlier detection and data repair processes. Like before we need an anomaly score for cells and another for instances. As usual, we can define likelihood-based scores using the decoder of this VAE model. The anomaly score for a cell of instance $\boldsymbol{x}$ is defined by

$$\mathcal{A}_d^\theta(\boldsymbol{x}) = -\log p_\theta(x_d \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}, y=1)) \quad \boldsymbol{x} \in \mathcal{X} \ , \tag{2.44}$$

whilst the anomaly score for the whole instance is

$$\mathcal{A}^\theta(\boldsymbol{x}) = -\sum_d^D \log p_\theta(x_d \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}, y=1)) \quad \boldsymbol{x} \in \mathcal{X} \ , \tag{2.45}$$

where the set of outlier instances is $\mathcal{O} = \left\{\boldsymbol{x} \in \mathcal{X} | \ \mathcal{A}^\theta(\boldsymbol{x}) \geq \gamma\right\}$, and cell x_d from $\boldsymbol{x} \in \mathcal{O}$ is dirty if $\mathcal{A}_d^\theta(\boldsymbol{x}) \geq \gamma_d$. However, since this is a semi-supervised model we have access to classifier $q_\phi(y|\boldsymbol{x})$, which when used as a score tends to obtain better detection performance. Therefore, the anomaly for the whole distance using the classifier is

$$\mathcal{A}^\phi(\boldsymbol{x}) = -\log q_\phi(y=1 \mid \boldsymbol{x}) \quad \boldsymbol{x} \in \mathcal{X} \ , \tag{2.46}$$

just like in the case of the M2 model. Once more, assuming the classifier is well calibrated we can use $\gamma \approx -\log(0.5)$, instead of tuning it or manually setting it. Moving on the repair process, we need to provide a point-wise estimate for the repair. For this model, this simply means conditioning on $y = 1$ as we want to generate a inlier like instance. As such, we generally have that

$$\hat{\boldsymbol{x}} = g_r^\theta(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}'} \ \log p_\theta(\boldsymbol{x}' \mid \boldsymbol{\mu}_\phi(\boldsymbol{x}, y=1)) \quad \boldsymbol{x} \in \mathcal{O} \ , \tag{2.47}$$

and for the case of images with continuous pixels we can just simplify to

$$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_\theta(\boldsymbol{\mu}_\phi(\boldsymbol{x}, y=1)) \quad \boldsymbol{x} \in \mathcal{O} \ . \tag{2.48}$$

2.5.4 Latent Space Disentanglement in VAEs

A class of VAE models that have garnered some attention in the last few years are those that try to *disentangle the latent space*. The idea is that a latent variable, or a set of them, will map to a particular *attribute (or characteristic, or factor)* of the instance. Equally as important, these latent variables (or set of) should be statistically independent, thus guaranteeing the modelling of different data attributes. If this happens, then the VAE has its latent space *disentangled*, or has a *disentangled* representation. A good overview of disentanglement methods in the context of VAEs is found in Locatello et al. (2019a) and Tschannen et al. (2018), though more recent surveys may exist. In Locatello et al. (2019b) the authors give a complementary overview for semi-supervised disentangled VAEs.

What is a Disentangled Representation?

The essential task of disentanglement (in VAEs) is to learn statistically independent non-linear latent features (i.e. variables) that together completely describe the data generation process. In practice, it is also common to relax the independence aspect to cover sets or groups of independent latent variables, instead of just individual variables.

In our context, an instance is composed by several attributes, where each attribute can result in different feature (or pixel) values depending on its *latent label*. For instance, consider an image dataset of human faces, an instance is made of several attributes like *eyes*, *mustache*, *nose type*, *hair*, etc. Each attribute may have different *latent labels*, e.g. *hair* may have the labels [blonde, brunette, black]; or *mustache* may have the labels [yes, no]. One can relax this concept and allow the *latent label* to be continuous instead of discrete, allowing for much more detailed control of the image attribute. Note that an attribute is also known as *factor* or *characteristic* in literature.

Modelling the attributes through specific latent variables allows the user to have detailed control of the generative process, or reconstruction process. Because the latent variables are *disentangled*, this means a change on a particular attribute should not change other attributes when generating samples (or reconstruction), if we assume other latent variables retain their values. In our example above, this means we can modify the *hair* label changing a specific latent variable value

without modifying other attribute labels like *mustache* or *eyes*. Once more, this is due to the property of statistical independence between latent variables, which disentanglement models in VAEs try to guarantee (Lopez et al., 2018; Khemakhem et al., 2020).

Relevance to Data Repair

Theoretically, for the problem of data repair, a VAE with a disentangled representation would model attributes related to errors in a specific set of latent variables. Therefore, after encoding the outlier instance into latent space, the values of latent variables related to errors can be modified such that upon reconstruction it produces a repair. This means that error attributes have been set to be *disabled* or *off*, by choosing appropriate values for those latent variables. Further, all other attributes not related to the presence of errors are maintained, and so the instance is reconstructed without errors (repair). Moreover, by checking which values are in the error related latent variables after encoding, a model can potentially classify the instance as an outlier or inlier. In other words, perform outlier detection.

Generally, this *latent space disentanglement* perspective about outlier detection and data repair has not been explored in literature. In Chapter 4 we explore this topic in depth, and propose our own solution (*Clean Subspace VAE*, CLSVAE). Specifically, we show the effectiveness of our solution on dataset corruption by systematic errors. This makes sense since systematic errors have specific repeatable patterns, which should theoretically be encoded well in a set of latent variables.

Models from Literature

According to Bengio et al. (2013); Higgins et al. (2016) disentanglement is an important part of representation learning, which can be useful for downstream tasks – e.g. regression, classification, clustering. In Locatello et al. (2019a) the authors present theoretical results that put into question the abilities of existing methods to actually disentangle the latent space successfully. However, counterarguments to these theoretical results have also appeared in literature, such as in Mathieu et al. (2019) and the recent works on *identifiable deep generative models* (Hälvä et al., 2021; Khemakhem et al., 2020; Moran et al., 2022). The authors in Locatello et al. (2019a) argue that one of the main practical objectives of unsupervised VAE disentanglement should be interpretability and fairness. Further, the authors

suggest that inductive biases and labelled data (e.g. semi-supervision) play an important role in maximizing the usefulness of disentanglement. The authors in Hälvä et al. (2021); Khemakhem et al. (2020) also suggest that inductive biases or labelled data are often necessary if we are to obtain an indentifiable model – i.e. able to disentangle and obtain the true latent factors of the generative process. Moreover, in Moran et al. (2022) the concepts of *anchor features* and *sparsity* used jointly are seen as sufficient to guarantee model identifiability.

Most of the earlier VAE disentanglement models focused on unsupervised approaches. The β-VAE (Higgins et al., 2016) was one of the first models, where the coefficient β controls weight of the D_{KL} term relative to the reconstruction term. The idea is that β as an hyperparameter can adjust VAE disentanglement, specifically for $\beta \geq 1$. The work in Burgess et al. (2018) provides further study of β-VAE within an information bottleneck framework. It explains the tradeoff between reconstruction quality and a disentangled representation, and the role of β in adjusting it. It proposes a way to anneal β in a way that progressively increases the bottleneck capacity so that the encoder focuses on learning one attribute at a time. A generalization of β-VAE is proposed in Mathieu et al. (2019), where the aim is to make disentanglement more principled and study the impact of hyperparameters.

Nonetheless, many of the unsupervised models have focused on augmenting the ELBO of the VAE (see eq. 2.6) with regularizers encouraging disentanglement of the latent variables. Some methods propose a regularizer based on the decomposition of the typical D_{KL} found in the ELBO of a VAE. This D_{KL} can be decomposed in two terms: mutual information (MI) between $\boldsymbol{x}$ and $\boldsymbol{z}$; and the D_{KL} divergence between the aggregate posterior $q_\phi(\boldsymbol{z})$ and the standard prior $p(\boldsymbol{z})$. For instance, such models include InfoVAE (Zhao et al., 2017a), β-TCVAE and FactorVAE (Kim & Mnih, 2018) (Total Correlation as surrogate for MI penalty), DIP-VAE (Kumar et al., 2017) (matching the moments of aggregate posterior and prior). Another class of methods introduces regularizers to the ELBO that promote independence between sets of latent variables. This is in contrast with most methods that focus on independence between individual latent variables. Some examples are HSIC-VAE (Lopez et al., 2018) and HF-VAE (Esmaeili et al., 2019). A different perspective is to change the VAE prior structure of $\boldsymbol{z}$ to induce disentanglement; for instance, Tonolini et al. (2019) proposes a prior on the latent

variables based on sparsity (i.e. Spike and Slab distribution).

More recently some works (iVAE) Khemakhem et al. (2020); Hälvä et al. (2021) focus on melding nonlinear ICA (Independent Component Analysis) (Bach & Jordan, 2002; Hyvarinen & Morioka, 2016) with VAEs. The original ICA is a linear model that focuses on the separation of a mixture of signals into their independent and separate sources. Indeed, this is related to VAE disentanglement since the aim is to find statistically independent attributes. These works also reinforce the fact that inductive biases (e.g. neural architectures, weak-supervision) or semi-supervision (i.e. a few labelled instances) are needed for effective latent disentanglement, just like in ICA; otherwise, the problem might be intractable. The importance of inductive biases or explicit supervision is also discussed in Locatello et al. (2019a).

Having said that, inductive biases have been explored in literature for the purpose of disentanglement. These explore things like relational information, cause-effect of the variation between attributes, grouping information, amongst others. Some examples can be found in Bouchacourt et al. (2018); Li & Mandt (2018); Ruiz et al. (2019); Hsu et al. (2017).

Most of the semi-supervised VAE models in literature focus on disentangling a few latent variables (attributes) that are partly observed (few labels), whilst the remaining variables (attributes) remain entangled. More recent examples can be seen in CCVAE (Joy et al., 2020) and DIVA (Ilse et al., 2020), and older models seen in Paige et al. (2017); Lopez et al. (2018); Bouchacourt et al. (2018). Further, fully supervised models can be found in CSVAE (Klys et al., 2018) and Fader Networks (Lample et al., 2017). From another perspective, a fully supervised GAN (Generative Adversarial Network) version is seen in StarGAN v2 (Choi et al., 2020). On the other hand, some models assume all ground-truth attributes (latent variables) are labelled, e.g. the semi-supervised version in Locatello et al. (2019b).

The type of VAE disentanglement models most useful for our problem, specially when a trusted set (some labelled data) is available, is the semi-supervised kind. Those particularly important are the models that can handle only using very few labelled instances, i.e. *sparse* semi-supervision. This exact scenario is explored in Chapter 4. For this reason we pick CCVAE (Joy et al., 2020) as a SOTA baseline

for our problem of outlier detection and data repair. We explain CCVAE in detail below, within the context of our problem.

2.5.4.1 CCVAE (Semi-Supervised Latent Disentanglement)

Here we present a recent semi-supervised latent disentanglement VAE, the CCVAE (Joy et al., 2020), which provides reasonable performance at outlier detection and data repair. This model was explored in the context of systematic error repair as a baseline, see Chapter 4 for details. Moreover, Joy et al. (2020) compares CCVAE to M2 model in several tasks, with CCVAE clearly performing much better at classification and conditional generation.

In Joy et al. (2020) the latent space $\boldsymbol{z}$ is split into two subspaces: the style (or agnostic) part $\boldsymbol{z}_{\setminus c}$ that is meant to model unlabelled patterns present in the instance; the characteristics part $\boldsymbol{z}_c$ that is meant to model the labelled attributes (also called characteristics). Formally we have $\boldsymbol{z} = \left[\boldsymbol{z}_c \,;\, \boldsymbol{z}_{\setminus c}\right]$ where $[\,;\,]$ is the concatenation operation. An *attribute* in the context of latent disentanglement models could be for instance *mustache* vs. *no mustache* in a human face, *color of hair*, *skin color*, or even *inlier* vs. *outlier*. Therefore, changing the value of $\boldsymbol{z}_c$ when reconstructing an existing data instance at test time is usually called *attribute manipulation*, since this changes how the attribute is reflected in the reconstruction of the instance.

Following Joy et al. (2020) and its implementation, our problem is modelled by defining $\boldsymbol{z}_c$ as a single variable associated with binary label y. Basically, $\boldsymbol{z}_c$ can be seen as a latent representation or embedding encoding whether the data instance has been corrupted or not. In this context, if $y = 1$ that means $\boldsymbol{z}_c$ should have a value that reflects the absence of errors when generating $\boldsymbol{x}$, i.e. an inlier; if $y = 0$, then that means $\boldsymbol{z}_c$ should have a value that reflects the presence of errors when generating $\boldsymbol{x}$, i.e. an outlier.

Generative Model

From Joy et al. (2020), and using a similar notation, we have

$$p(\boldsymbol{x}, \boldsymbol{z}, y) = p_\theta(\boldsymbol{x}|\boldsymbol{z})\ p_\psi(\boldsymbol{z}_c|y)\ p(\boldsymbol{z}_{\setminus c})\ p(y), \tag{2.49}$$

$$p(y) = \text{Bernoulli}(y|\alpha), \tag{2.50}$$

$$p(\boldsymbol{z}_{\setminus c}) = \mathcal{N}(\boldsymbol{z}_{\setminus c}|\mathbf{0}, \mathbf{I}), \tag{2.51}$$

$$p_\psi(\boldsymbol{z}_c|y) = \mathcal{N}(\boldsymbol{z}_c|\boldsymbol{\mu}_\psi(y), \boldsymbol{\sigma}^2_\psi(y)), \tag{2.52}$$

where $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is a neural network decoder, like in Section 2.5.1, and θ are its parameters. Once more, α expresses the prior belief about the fraction of inliers in the dataset. The above generative model expresses a two-component mixture model on $\boldsymbol{z}_c$ with y as gating variable, see eq. (2.52). In fact, we have a mean and variance for each y value as it pertains to $\boldsymbol{z}_c$ where: $\boldsymbol{\mu}_\psi(y=1)$ and $\boldsymbol{\sigma}_\psi(y=1)$ for inliers; and similar for outliers ($y=0$).

Variational Model

The variational distribution is factorized as

$$q(y, \boldsymbol{z}|\boldsymbol{x}) = q_{\varphi,\phi}(y|\boldsymbol{x})\ q_{\varphi,\phi}(\boldsymbol{z}|\boldsymbol{x}, y), \tag{2.53}$$

$$q_\varphi(y|\boldsymbol{z}_c) = \text{Bernoulli}(y|\pi_\varphi(\boldsymbol{z}_c)), \tag{2.54}$$

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_\phi(\boldsymbol{x}), \boldsymbol{\sigma}^2_\phi(\boldsymbol{x})), \tag{2.55}$$

$$q_{\varphi,\phi}(y|\boldsymbol{x}) = \int q_\varphi(y|\boldsymbol{z}_c) q_\phi(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}, \tag{2.56}$$

$$q_{\varphi,\phi}(\boldsymbol{z}|\boldsymbol{x}, y) = \frac{q_\varphi(y|\boldsymbol{z}_c) q_\phi(\boldsymbol{z}|\boldsymbol{x})}{q_{\varphi,\phi}(y|\boldsymbol{x})}, \tag{2.57}$$

where $q_\varphi(y|\boldsymbol{z}_c)$ and $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ are neural network based encoders, with φ and ϕ as neural network parameters.

Training Loss

The training loss in Joy et al. (2020) for CCVAE is based on the ELBO, with one term for the labelled part of data (trusted set), and a second term for the unlabelled part. Below we quickly present the ELBO loss and the optimization problem, for a detailed description please see the original paper (Joy et al., 2020).

The labelled part of the ELBO is

$$\mathcal{L}_{\text{CCVAE}}(\boldsymbol{x}, y) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{q_\varphi(y|\boldsymbol{z}_c)}{q_{\varphi,\phi}(y|\boldsymbol{x})}\log\left(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})\; p_\psi(\boldsymbol{z}_c|y)\; p(\boldsymbol{z}_{\setminus c})}{q_\varphi(y|\boldsymbol{z}_c)\; q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right)\right] + \\ + \beta\;\log q_\varphi(y|\boldsymbol{x}) + \log p(y)\;, \tag{2.58}$$

where β is the hyperparameter controlling amount of up-sampling and importance relative to other terms, like in M2 model (section 2.5.3.2). In Joy et al. (2020), for their application, they found that setting $\beta = 1$ brought good results and found no need to tune it further. In our problem setting, we found that we obtained better performance by using larger values for β. This is probably because our application is different, i.e. outlier detection and subsequent repair, and since the trusted sets (labelled sets) used in our problem setup are quite small.

An important point about CCVAE is that the term that encourages the classifier $q_\varphi(y|\boldsymbol{z}_c)$ to learn the labels of the trusted set emerges naturally from the labelled ELBO – see eq. (2.58). As a result, one does not need to add a cross-entropy term like in M2 model (section 2.5.3.2).

The unlabelled part of the ELBO is

$$\mathcal{L}_{\text{CCVAE}}(\boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})q_\varphi(y|\boldsymbol{z}_c)}\left[\log\left(\frac{p_\theta(\boldsymbol{x}|\boldsymbol{z})\; p_\psi(\boldsymbol{z}_c|y)\; p(\boldsymbol{z}_{\setminus c})\; p(y)}{q_\varphi(y|\boldsymbol{z}_c)\; q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right)\right]\;. \tag{2.59}$$

Lastly, the training procedure is defined by

$$\min_{\theta,\,\phi,\,\varphi,\,\psi}\sum_{\boldsymbol{x}\in\mathcal{X}_u}\mathcal{L}_{\text{CCVAE}}(\boldsymbol{x}) \;+\sum_{(\boldsymbol{x},y)\in\mathcal{X}_l\times\mathcal{Y}_l}\mathcal{L}_{\text{CCVAE}}(\boldsymbol{x}, y)\;. \tag{2.60}$$

Outlier Detection and Repair Process

After training, we proceed to outlier detection and data repair. We need to define a score for use in detection, and for that we use the classifier given by the variational model. The same was done in Joy et al. (2020) for classification tasks, but one could use a decoder likelihood-based score. The former tends to have better performance, if available. Hence, for instance outlier detection, we have the score

$$\mathcal{A}^{\varphi,\,\phi}(\boldsymbol{x}) = -\log\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[q_\varphi(y=1|\boldsymbol{z}_c)\right] \quad \boldsymbol{x}\in\mathcal{X}, \tag{2.61}$$

which is the approximate negative log-probability of an instance being an inlier. The user may define the threshold γ or tune it. However, assuming the classifier is somewhat calibrated, then one can use the a $\gamma \approx -\log(0.5)$.

If one needs to detect cell outliers, one can use a decoder based anomaly score like in other VAEs. For CCVAE, this can be defined for some feature d as

$$\mathcal{A}^{\theta,\,\psi}(\boldsymbol{x}) = -\log p_\theta(x_d \mid \left[\boldsymbol{\mu}_\psi(y=1); \boldsymbol{\mu}_\phi(\boldsymbol{x})_{\setminus c}\right]) \quad \boldsymbol{x} \in \mathcal{X}\ , \tag{2.62}$$

where the mean for $\boldsymbol{z}_c$ corresponding to an inlier ($y = 1$) is used, i.e. $\boldsymbol{\mu}_\psi(y = 1)$. The encoding from non-labelled instance characteristics is reused, i.e. $\boldsymbol{\mu}_\phi(\boldsymbol{x})_{\setminus c}$.

From the perspective of CCVAE, the repair of an outlier instance is just *attribute manipulation* via $\boldsymbol{z}_c$ subspace. Once the appropriate $\boldsymbol{z}_c$ is found, then one uses the decoder for reconstruction obtaining a repair. Under our problem definition, automated repair is very much like *conditional generation* as seen in Joy et al. (2020), where samples for $\boldsymbol{z}_c$ are drawn from the conditional prior whilst reusing $\boldsymbol{z}_{\setminus c}$ obtained from encoding the outlier instance. Specifically, we have $\boldsymbol{z} = \left[\boldsymbol{z}_c \sim p_\psi(\boldsymbol{z}_c|y)\ ;\ \boldsymbol{z}_{\setminus c}\right]$, where depending on the y value it forces the generated samples to have, or not have, the presence of the attribute (e.g. inlier / outlier). In our case, we are interested in automated repair, and thus limiting human interaction apart from building the trusted set. That means that exploring $\boldsymbol{z}_c$ with user interaction to pick the best repair (reconstruction) for each outlier instance is not realistic. So we wish to obtain the most likely reconstruction under $y = 1$, i.e. the *maximum a posteriori*, thus defining an automated repair for the outlier instance. Therefore, the repair for an image instance with continuous features is given by

$$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_\theta\left(\left[\boldsymbol{\mu}_\psi(y=1)\ ;\ \boldsymbol{\mu}_\phi(\boldsymbol{x})_{\setminus c}\right]\right) \quad \boldsymbol{x} \in \mathcal{O}, \tag{2.63}$$

where $\boldsymbol{\mu}_\psi(y = 1)$ is the mean for the inlier component of $p_\psi(\boldsymbol{z}_c \mid y)$, and $\boldsymbol{\mu}_\phi(\boldsymbol{x})_{\setminus c}$ is the mean of $q_\phi(\boldsymbol{z}_{\setminus c} \mid \boldsymbol{x})$, which excludes the characteristic (labelled) latent subspace. Similarly, for mixed-type datasets a repair procedure that uses a MAP estimate of the decoder can be easily defined – as seen in eq. (2.47).

Chapter 3

Robust VAEs for Outlier Detection and Repair of Mixed-Type Data

3.1 Motivation: How does it fit into the thesis?

The main motivation behind this model proposal was dealing with random error corruption in data cleaning. The main goal was to develop a model that is unsupervised and requires as little user intervention. The focus is on the problem of unsupervised cell outlier detection and repair in mixed-type tabular data, for the case of *point outliers* (Section 2.2.1). Traditional methods are concerned only with detecting which rows in the dataset are outliers. However, identifying which cells are corrupted in a specific row is an important problem in practice, and the very first step towards repairing them. The Robust Variational Autoencoder (RVAE) is introduced as a deep generative model that learns the joint distribution of the clean data while identifying the outlier cells, allowing for their repair later on. RVAE explicitly learns the probability of each cell being an outlier, balancing different likelihood models in the row outlier score, making the method suitable for outlier detection in mixed-type datasets. Experimentally it is shown that not only RVAE performs better than several SOTA methods in cell outlier detection and repair for tabular data, but also that is robust against the initial hyperparameter selection.

3.2 Introduction

The existence of outliers in real world data is a problem data scientists face daily, so outlier detection (OD) has been extensively studied in the literature (Chandola et al., 2009; Emmott et al., 2015; Hodge & Austin, 2004). The task is often *unsupervised*, meaning that we do not have annotations indicating whether individual cells in the data table are clean or anomalous. Although supervised OD algorithms have been proposed (Lee et al., 2018; An & Cho, 2015; Schlegl et al., 2017), annotations of anomalous cells are often not readily available in practice. Instead, unsupervised OD attempts to infer the underlying clean distribution, and explains outliers as instances that deviate from that distribution. It is important to focus on the joint distribution over features, because although some outliers can be easily identified as anomalous by considering only the marginal distribution of the feature itself, many others are only detectable within the context of the other features (Chandola et al., 2009, section 2.2). Recently deep models have outperformed traditional ones for tabular data tasks (Klambauer et al., 2017), capturing their underlying structure better. They are an attractive choice for outlier detection, since they have the flexibility to model a wide variety of clean distributions. However, outlier detection work has mostly focused on image datasets, repairing dirty pixels instead of cells in tabular data (Wang et al., 2017b; Zhou & Paffenroth, 2017; Akrami et al., 2019b).

Outliers present unique challenges to deep generative models. First, most work focuses on detecting anomalous data rows, without detecting which specific cells in a row are problematic (Redyuk et al., 2019; Schelter et al., 2018). However, not enough care is given to cell granularity, which means it is often difficult to properly *repair* the dirty cells, e.g. if there are a large number of columns or when the data scientist is not a domain expert. Work on cell-level detection and repair often focuses on real-valued features, e.g. images (Zhou & Paffenroth, 2017; Wang et al., 2017b; Schlegl et al., 2017), or does not provide a principled way to detect anomalous cells (Nguyen & Vien, 2018a). Second, tabular data is often *mixed-type*, including both continuous and categorical columns. Although modelling mixed-type data has been explored before (Nazabal et al., 2020; Vergari et al., 2019), difficulty arises when handling outliers. Standard anomaly scores are based on the probability that the model assigns to a cell, but these values are not comparable between likelihood models, performing poorly for mixed-type data.

Finally, the effect of outliers in unsupervised learning can be insidious. Since deep generative models are highly flexible, they are not always robust against outliers (Hendrycks & Dietterich, 2019), overfitting to anomalous cells. When the model overfits, it cannot identify these cells as outliers, because it has modelled them as part of the clean distribution, and consequently, most repair proposals are skewed towards the dirty values, and not the underlying clean ones.

The main contributions of this chapter are: (i) *Robust Variational Autoencoder (RVAE)*, a novel fully unsupervised deep generative model for cell-level OD and repair for mixed-type tabular data, for the case of *point outliers* (see Section 2.2.1). It uses a two-component mixture model for each feature, with one component for clean data, and the other component that robustifies the model by isolating outliers. (ii) *RVAE* models the underlying clean data distribution by down-weighting the impact of anomalous cells, providing a competitive anomaly score for cells and a superior estimate of cell repairs. (iii) A hybrid inference scheme for optimizing the model parameters, combining amortized and exact variational updates, which proves superior to standard amortized inference. (iv) *RVAE* allows to present an anomaly score that is commensurate across mixed-type data. (v) *RVAE* is robust to the selection of its hyperparameters, while other outlier detection methods suffer from the need to tune their parameters to each specific dataset.

3.3 Related Work

There is relevant prior work in the field of outlier detection and robust inference in the presence of outliers, a good meta-analysis study presented in Emmott et al. (2015). Most prior models apply to *point outliers* (see Section 2.2.1) as this is the most common type of outlier, and the type we explore in this thesis. Different deep models have been applied to this task, including autoencoders (Zong et al., 2018b; Nguyen & Vien, 2018a; Zhou & Paffenroth, 2017), VAEs (An & Cho, 2015; Wang et al., 2017b) and generative adversarial networks (Schlegl et al., 2017; Lee et al., 2018). In Nalisnick et al. (2018) the authors show that deep models trained on a dataset assign high likelihoods to instances of different datasets, which is problematic in outlier detection. We identify outliers during training rather than from a fully-trained model, down-weighting their effect on parameter learning. Earlier in training, the model had less chance to overfit, so it should be easier to

detect outliers.

Most closely related to our model are methods based on **robust PCA (RPCA) and autoencoders**. They focus on unsupervised learning in the presence of outliers, even though most methods need labelled data for hyperparameter tuning (Candès et al., 2011; Zhou & Paffenroth, 2017; Zong et al., 2018b; Nguyen & Vien, 2018a; Xu et al., 2018; Akrami et al., 2019b). RPCA-based alternatives often assume that the features are real-valued, and model the noise as additive with a Laplacian prior. A problem in RPCA-type models is that often the hyperparameter that controls the outlier mechanism is dataset dependent and difficult to interpret and tune. In Wang et al. (2017b), the authors proposed using a VAE as a recurrent unit, iteratively denoising the images. This iterative approach is reminiscent of the solvers used for RPCA. However, their work is not easily extended to mixed likelihood models and suffers from the same problems as VAEs when computing row scores (Section 3.5.3).

Robust Variational Inference. Several methods explore robust divergences for variational learning in the presence of outliers applied to supervised tasks (Regli & Silva, 2018a; Futami et al., 2018). These divergences have hyperparameters which are dataset dependent, and can be difficult to tune in unsupervised outlier detection; in contrast, the α hyperparameter used in RVAE is arguably more interpretable, and experimentally robust to misspecification. Recently a VAE model using one of these divergences in the decoder was proposed for down-weighting outliers (Akrami et al., 2019b). However, in contrast to our model, they focused on image datasets and are not concerned with cell outliers. The same hyperparameter tuning problem arises, and it is not clear out to properly extend to categorical features.

Bayesian Data Reweighting. Wang et al. (2017a) propose an approach that raises the likelihood of each observation by some weights and then infer both the latent variables and the weights from corrupted data. Unlike RVAE, these weights are only defined for each instance, so the method cannot detect cell-level outliers. Also, the parameters of the model are trained via MCMC instead of variational inference, making them more difficult to apply in deep generative models.

Classifier Confidence. Several methods explore adding regularization to improve neural network classifier robustness to outliers (Lee et al., 2018; Hendrycks et al.,

2019). Neural network based classifiers tend to be over-confident, and can overfit to spurious patterns. Regularization can come in form of adding uniform noise over the categories of target labels (label tempering); or even adding an entropy-based term to the cross-entropy cost in order to avoid over-confidence. However, the regularization hyperparameters are not interpretable and often require a validation dataset to tune them. Other works like Hendrycks & Gimpel (2017), use the confidence of the predicted distribution as a measure of outlier detection.

3.4 Problem Setting

The work in this chapter assumes an *unsupervised training setting* where one only has access to dataset $\mathcal{X}$, where a data instance is given by $\boldsymbol{x} \in \mathcal{X}$. Notation for the unsupervised problem setting is found in Section 2.1. The problem definition for the *outlier detection task* is in Section 2.2.2, and the definition for the *data repair task* is in Section 2.3.1. We only consider outliers that are *corruption-based* of the *random error* kind, which fit within the scope of *point outliers* (see Section 2.2.1). Note that if possible we simplify the notation by removing the instance index n. For instance, w_{nd} is same as w_n, or x_d is same as x_{nd}, or $\boldsymbol{x}_n$ is same as $\boldsymbol{x}$.

Cells in the dataset are potentially corrupted with an unknown noising process appropriate for the feature type. The objective in this work is not only detecting the anomalous instances in the dataset, termed *row outliers*, but also determining the specific subset of cells that are anomalous, termed *cell outliers*, proposing potential *repair* values for them.

In this chapter we aim to improve the standard VAE model to be *robust* to corruption, and perform well at point outlier detection and repair. The *Standard VAE* model definition for *mixed-type data* can be found in Section 2.5.1 in the Background chapter.

3.5 Proposal: Robust Variational Autoencoder (RVAE)

To improve VAEs for outlier detection and repair, we want to make them more robust by automatically identifying potential outliers during training, so they are down-weighted when training the generative model. We also want a cell-level anomaly score which is comparable across continuous and categorical attributes.

We can achieve both goals by modifying the generative model of the VAE.

Here we define the novel *robust variational autoencoder (RVAE)*, a deep generative model based on a two-component mixture model likelihood (decoder) per feature, which isolates the outliers during training. RVAE is composed of a clean component $p_\theta(x_d|\boldsymbol{z})$ for each feature d, explaining the clean cells, and an outlier component $p_0(x_d)$, explaining the outlier cells. A mixing variable $w_d \in \{0, 1\}$ acts as a gate to determine whether cell x_d should be modelled by the clean component ($w_d = 1$) or the outlier component ($w_d = 0$).

We define the marginal likelihood of the mixture model under for $\boldsymbol{x} \in \mathcal{X}$ as[1]

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{w}} \int d\boldsymbol{z}\; p(\boldsymbol{z})\; p(\boldsymbol{w})\; p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{w}), \tag{3.1}$$

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{w}) = \prod_{d=1}^{D} p_\theta(x_d|\boldsymbol{z})^{w_d}\; p_0(x_d)^{1-w_d}, \tag{3.2}$$

where $\boldsymbol{w} \in \{0, 1\}^D$ is modelled by a Bernoulli distribution

$$p(\boldsymbol{w}) = \prod_{d=1}^{D} \text{Bernoulli}(w_d|\alpha), \tag{3.3}$$

and $\alpha \in [0, 1]$ is a parameter that reflects our belief about the cleanliness of the data. To approximate the posterior distribution $p(\boldsymbol{z}, \boldsymbol{w}|\boldsymbol{x})$, we introduce the variational distribution

$$q_{\phi,\pi}(\boldsymbol{w}, \boldsymbol{z}|\boldsymbol{x}) = q_\phi(\boldsymbol{z}|\boldsymbol{x}) \prod_{d=1}^{D} q_\pi(w_d|\boldsymbol{x}), \tag{3.4}$$

with $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ defined in equation 2.5 and $q_\pi(w_d|\boldsymbol{x}) = \text{Bernoulli}(w_d|\pi_d(\boldsymbol{x}))$. The probability $\pi_d(\boldsymbol{x})$ can be interpreted as the predicted probability of cell x_d being clean. This approximation uses the mean-field assumption that $\boldsymbol{w}$ and $\boldsymbol{z}$ are conditionally independent given $\boldsymbol{x}$. Furthermore, a similar mean-field assumption is made about the dimensions of $\boldsymbol{w}$ being conditionally independent given $\boldsymbol{x}$. Finally, the ELBO for the RVAE model can be written as

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}) = &\sum_{d=1}^{D} \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\pi_d(\boldsymbol{x}) \log p_\theta(x_d|\boldsymbol{z}) + (1 - \pi_d(\boldsymbol{x})) \log p_0(x_d)\right] \\ &- D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - D_{KL}(q_\pi(\boldsymbol{w}|\boldsymbol{x})||p(\boldsymbol{w})),\end{aligned} \tag{3.5}$$

[1]Mixture models can also be written in product form using mixing variables w_d (Bishop, 2006, Section 9, page 431), as we adopt here.

where the overall training loss to ***minimize*** is

$$\mathcal{I} = -\frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{X}} \mathcal{L}(\boldsymbol{x}). \tag{3.6}$$

Examining the gradients of equation 3.6 helps to understand the robustness property of the RVAE. The gradient of $\mathcal{I}$ with respect to the model parameters θ is given by

$$\nabla_\theta \, \mathcal{I} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{d=1}^{D} \pi_d(\boldsymbol{x}) \, \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\nabla_\theta \log p_\theta(x_d|\boldsymbol{z})\right]. \tag{3.7}$$

We see that $\pi_d(\boldsymbol{x})$ acts as a weight on the gradient. Cells that are predicted as clean will have higher values of $\pi_d(\boldsymbol{x})$, and so their gradients are weighted more highly, and have more impact on the model parameters. Conversely, cell outliers with low values of $\pi_d(\boldsymbol{x})$ will have their gradient contribution down-weighted. A similar formulation can be obtained for the encoder parameters ϕ.

Lastly, we note that the mean-field assumption on $\boldsymbol{w}$ (in equation 3.4) is inspired by the *random error* setting on the outliers. Indeed, this is the problem setting explored in this chapter. Random errors corrupt cells independently, and hence there is no need to model correlations between different cell (feature) errors. Therefore, one can make the assumption that $\boldsymbol{w}$ can be modelled by a factorized distribution on its D features. However, if the errors were of systematic or structured nature, then modelling correlations between different dimensions of $\boldsymbol{w}$ would be relevant and appropriate. Still, the modelling of correlations between cells errors can perhaps be better achieved through a subspace in $\boldsymbol{z}$. Specifically, as it comes to modelling efficiency. In fact, in Chapter 4 we explore a possible solution (CLSVAE) to the problem of systematic errors through latent modelling in $\boldsymbol{z}$ space.

3.5.1 Outlier Model

The purpose of the outlier distribution $p_0(x_d)$ is to explain the outlier cells in the dataset, removing their effect in the optimization of the parameters of clean component p_θ. For categorical features, we propose using the uniform distribution

$$p_0(x_d) = \frac{1}{C_d}\,. \tag{3.8}$$

Such a uniform distribution assumption has been used in multiple object modelling (Williams & Titsias, 2003) as a way to factor in pixel occlusion. In Chemudugunta et al. (2006) a similar approach for background words is proposed. For real features, we standardize the features to have mean 0 and standard deviation 1. We use an outlier model based on a broad Gaussian distribution[2]

$$p_0(x_d) = \mathcal{N}(x_d \mid 0, S) \ , \tag{3.9}$$

with $S > 1$. Outlier cells modelled by the outlier component will be further apart from $m_d(\boldsymbol{z})$ relative to clean ones.

Although more complex distributions can be used for $p_0(x_d)$, we show empirically that these simple distributions are enough to detect outliers from a range of noise levels (Section 3.6). Furthermore, RVAE can easily be extended to handle other types of features (Nazabal et al., 2020): for count features we can use a Poisson likelihood, where the outlier component p_0 would be a Poisson distribution with a large rate; for ordinal features we could have an ordinal logit likelihood, where p_0 can be a uniform categorical distribution.

3.5.1.1 Assumptions and Prior Work on Outlier Model

Broad distributions and robust statistics. In this chapter, it is assumed we are dealing with random error corruption as a source of dirty cells, and thus the outliers. The corruption is moderate or low, and therefore most of the instances are inliers and their cells are clean. Under these circumstances, it is common in classic robust statistics (Barron, 2019) to use heavy-tailed distributions to mitigate the contributions of these outliers to the training of parametric models. Some examples of heavy-tailed distributions are *t-Student*, *Laplace*, *Cauchy*, or *Richter* (Gales & Olsen, 1999b). The reasoning is that outlier values (i.e. dirty cells) will deviate from the more common inliers, and thus outliers will be modelled by the tail-end of the distribution. Indeed, the negative log-likelihood loss using a heavy-tailed distribution will make larger deviations of outlier cells have a lower loss value than typical distributions (e.g. Gaussian). Hence, the likelihood is now less sensitive to outliers, and consequently the model parameters are as well. In fact, in gradient-based training the outliers will have a much smaller magnitude compared to inliers, and thus contribute less to parameter training. However, these assumptions may not work for systematic error corruption (see Chapter 4).

[2]This is standard (Quinn et al., 2009; Gales & Olsen, 1999b)

In the case of RVAE, we used a heavy-tailed distribution based on a 2-component *Richter* distribution (Gales & Olsen, 1999b), where the objective is to mitigate outlier contribution to the training of $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. In this case, the heavy-tail behavior is modelled by $p_0(x_d)$ where the outliers are captured. Indeed, the *Richter* distribution has been used for capturing rare or outlier instances in speech recognition in its broader Gaussian components (Gales & Olsen, 1999b). In (Quinn et al., 2009) a similar 2-component *Richter* distribution is defined for time-series data, where novelty instances are captured by a broader component.

Outlier model dependency on $\boldsymbol{x}$. Given the current assumptions on the outliers (i.e. random errors), it is possible for RVAE to use a $p_0(x_d)$ where its parameters are not dependent on the original $\boldsymbol{x}$ for simplification purposes. In fact, for continuous features the mean parameter of $p_0(x_d)$ can be 0 since the data is standardized – i.e. mean of the original feature is subtracted. For a categorical feature no notion of a mean parameter exists, so a broad distribution can be modelled by a uniform across categories. In both cases this yielded good results for us in the experiments. However, in cases where continuous feature standardization does not work as well, it is possible to set the mean of $p_0(x_d)$ to the mean of $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ and hence obtain a proper *Richter* distribution. In this case, the dependency of $p_0(x_d)$ parameters on $\boldsymbol{x}$ becomes obvious. Moreover, using a *stop-gradient* operator for reusing the decoder mean might be useful to prevent gradients of dirty cells to affect the AE parameters (θ and ϕ).

Outlier model dependency on $\boldsymbol{z}$. Since the assumption is that random errors are corrupting cells independently in the data, there should not be any correlation of the cell errors between multiple data features. Hence, there is no need to make $p_0(x_d)$ dependent on $\boldsymbol{z}$ as error modelling between different features is not relevant here. On the other hand, for systematic error corruption (see Chapter 4) cell errors affecting multiple features will be correlated, and therefore modelling that correlation through $\boldsymbol{z}$ can be quite important for outlier detection and repair. In fact, the CLSVAE model in Chapter 4 takes that assumption.

3.5.2 Inference

We use a hybrid procedure to train the parameters of RVAE that alternates amortized variational inference using stochastic gradient descent for ϕ and θ, and coordinate ascent over π. When we do not amortize π, but rather treat each

$\pi_d(\boldsymbol{x}) \in [0, 1]$ as an independent parameter of the optimization problem, then an exact solution for $\pi_d(\boldsymbol{x})$ is possible when ϕ and θ are fixed. Optimizing the ELBO equation 3.5 w.r.t. $\pi_d(\boldsymbol{x})$, we obtain an exact expression for the optimum[3]

$$\hat{\pi}_d(\boldsymbol{x}) = g\left(r + \log\frac{\alpha}{1-\alpha}\right), \tag{3.10}$$

where

$$r = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log\frac{p_\theta(x_d|\boldsymbol{z})}{p_0(x_d)}\right], \tag{3.11}$$

where g is the sigmoid function. The first term in equation 3.10 represents the density ratio r between the clean component $p_\theta(x_d|\boldsymbol{z})$ and the outlier component $p_0(x_d)$. When $r > 0$ it will bias the decision towards assuming the cell being clean, conversely $r < 0$ it will bias the decision towards the cell being dirty. Such a ratio r has arisen in the literature (Hido et al., 2011; Yamada et al., 2017). usually between a distribution trained on clean (or labelled) data and the test-set distribution, where one is performing outlier detection. The second term in equation 3.10 represents our prior belief about cell cleanliness, defined by $\alpha \in [0, 1]$. Higher values of α will skew the decision boundary towards a higher $\hat{\pi}_d(\boldsymbol{x})$, and vice-versa. This coordinate ascent strategy is common in variational inference for conjugate exponential family distributions (see e.g. Jordan et al. 1999). We term this model RVAE-CVI (Coordinate ascent Variational Inference) below.

Alternatively, $\pi_d(\boldsymbol{x})$ can be obtained using amortized variational inference. However, two problems arise in the process. First, an inference gap is introduced by amortization, leading to slower convergence to the optimal solution. Second, there might not be enough outliers in the data to properly train a neural network to recognize the decision boundary between clean and dirty cells. We term this model RVAE-AVI (Amortized Variational Inference). RVAE inference is summarized in Algorithm 1, for both the coordinate ascent version (RVAE-CVI) and the amortized version (RVAE-AVI). We used Adam (Kingma & Ba, 2014) as the gradient-based optimizer (line 15).

Note that in line 14 of Algorithm 1 for RVAE-CVI it is important to clarify that *no gradients* of θ are being passed through the estimate of $\hat{\pi}_{md}$. Indeed, if using RVAE-CVI the $\hat{\pi}_{md}$ estimate step in line 13 always uses the *stop-gradient* operation for all gradients of θ and ϕ. This is by design, as a coordinate ascent step for $\hat{\pi}_{md}$ should assume θ and ϕ are static parameters during inference, and

[3]The derivation of equation equation 3.10 is provided in the Additional Notes (Section 3.7.2).

vice-versa. Thus $\hat{\pi}_{md}$ should not allow for gradients being passed through when applying (stochastic) gradient descent on θ and ϕ. Empirically, we tried both the option of having gradients pass through, and the current one of stopping them. Generally, from a repair performance perspective, the best option was to use *stop-gradient*. In fact, in higher corruption scenarios this was more obvious has letting the gradients through would lead to RVAE-CVI more easily overfit to errors.

Algorithm 1 RVAE Inference

procedure RVAE(η learning rate, M batch size, T number epochs, α prior value)
  if RVAE-AVI = True **then**
    Define NN parameters: $\Psi = \{\phi, \theta, \tau\}$;
  else if RVAE-CVI = True **then**
    Define NN parameters: $\Psi = \{\phi, \theta\}$;
  Initialize Ψ;
  for $1, ..., T$ **do**
    Sample mini-batches $\{\boldsymbol{x}_m\}_{m=1}^{M} \sim p(\boldsymbol{x})$;
    Evaluate $p_\theta(x_{md}|\boldsymbol{z}_m)$ and $p_0(x_{md})$ $\forall m, d$;
    if RVAE-AVI = True **then**
      Evaluate encoder $\pi_\tau(\boldsymbol{x}_m)$;
    else if RVAE-CVI = True **then**
      Infer $\hat{\pi}_{md}, \forall m, d$ using eq. equation 3.10
    $g_\Psi \leftarrow \nabla_\Psi \mathcal{I}(\Psi, \pi(\boldsymbol{x}_m), \alpha)$ using eq. equation 3.6;
    $\Psi \leftarrow \text{Optimizer}(\Psi, g_\Psi, \eta)$;

3.5.3 Anomaly Scores for Outlier Detection

A natural approach to determine which cells are outliers in the data is computing the likelihood of the cells under the trained model. In a VAE, the scores for row and cell outliers would be

$$\textbf{Cell: } -\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(x_d|\boldsymbol{z})\right], \quad \textbf{Row: } -\sum_{d=1}^{D} \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(x_d|\boldsymbol{z})\right], \tag{3.12}$$

where a higher score means a higher outlier probability. However, likelihood-based anomaly scores present several problems, specifically for row scores. In mixed-type

datasets categorical features and real features are modelled by probability and density distributions respectively, which have different ranges. Often this leads to continuous features dominating over categorical ones. With the RVAE we propose an alternative anomaly score based on the mixture probabilities $\hat{\pi}_d(\boldsymbol{x})$

$$\textbf{Cell: } -\log \hat{\pi}_d(\boldsymbol{x}), \quad \textbf{Row: } -\sum_{d=1}^{D} \log \hat{\pi}_d(\boldsymbol{x}), \tag{3.13}$$

where again a higher score means a higher outlier probability. Notice that the row score is just the negative log-probability of the row being clean, given by $\hat{\pi} = \prod_{d=1}^{D} \pi_d(\boldsymbol{x})$. These mixture-based scores are more robust against some features or likelihood models dominating the row anomaly score, making them more suitable for mixed-type datasets.

3.5.4 Repair Process for Dirty Cells

Cell repair is related to missing data imputation. However, this is a much **harder** task, since the positions of anomalous cells are not given, and need to be inferred. After the anomalous cells are identified, a robust generative model allows to impute them given the dirty row directly. In general, repair under VAE-like models can be obtained via maximum a posteriori (MAP) inference,

$$\hat{x}_d = \arg\max_{x_d} p_\theta(x_d|\boldsymbol{\mu}_\phi(\boldsymbol{x})), \tag{3.14}$$

where $\hat{x}_d$ is the proposed repair for cell d of some dirty instance $\boldsymbol{x}$. Note that the underlying inlier value, or ground-truth value, for the cell is given by $\tilde{x}_d$. In the case of RVAE, $p_\theta(x_d|\boldsymbol{z})$ is the clean component responsible for modelling the underlying clean data, see equation 3.2. This reconstruction is akin to robust PCA's clean component. In practice, for real features $\hat{x}_d = m_d(\boldsymbol{z})$, the mean of the Gaussian likelihood, and for categorical features $\hat{x}_d = \arg\max_c f(a_{dc}(\boldsymbol{z}))$, the highest probability category. Note that index $c \in \{1, ..., C_d\}$ refers to a unique class for categorical feature d; where the total number of categories is C_d. Other repair strategies are discussed in the Additional Results (Section 3.8.7).

3.6 Experiments

We showcase the performance of RVAE and baseline methods, for both the task of identifying row and cell outliers and repairing the corrupted cells in the data[4].

[4] `https://github.com/sfme/RVAE_MixedTypes/`

Four different datasets from the UCI repository (Lichman, 2013), with a mix of real and categorical features, were selected for the evaluation (see Additional Notes, Section 3.7.1). We compare RVAE with ABDA (Vergari et al., 2019) on a different outlier detection task in the Additional Results (Section 3.8.6).

3.6.1 Corruption Process

All datasets were artificially corrupted in both training and validation sets. This is a standard practice in outlier detection (Futami et al., 2018; Redyuk et al., 2019; Krishnan et al., 2016; Natarajan et al., 2013), and a necessity in our setting, due to the scarcity of available datasets with labelled cell outliers. No previous knowledge about corrupted cell position, or dataset corruption proportion is assumed.

For each dataset, a subset of cells are randomly selected for corruption, following a two-step procedure: a) a percentage of rows in the data are selected at random to be corrupted; b) for each of those selected rows, 20% of features are corrupted at random, with different sets of features being corrupted in each select row. For instance, a 5%-20% scenario means that 5% of the rows in the data are randomly selected to contain outliers, and for each of these rows, 20% of the features are randomly corrupted, leading to 1% of cells corrupted overall in the dataset.

We will consider for the experiments five different levels of row corruption, $\{1\%, 5\%, 10\%, 20\%, 50\%\}$, leading to five different levels of cells corrupted across the data, $\{0.2\%, 1\%, 2\%, 4\%, 10\%\}$. We repeat this process five times, and provide in the next sections the results for the aggregate of all datasets for one of those experiments. The full disclosure of the results per dataset can be found in the Additional Results (Section 3.8), including results with error bars (Section 3.8.5).

Real features: Additive noise is used as a noising process, with dirty cell values obtained as $x_d = \tilde{x}_d + \zeta$, with $\zeta \sim p_{noise}(\mu, \eta)$. Note that the noising process is performed before standardizing the data. Four different noise distributions p_{noise} are explored: *Gaussian noise* ($\mu = 0$, $\eta = 5\hat{\sigma}_d$), with $\hat{\sigma}_d$ the statistical standard deviation of feature d; *Laplace noise* ($\mu = 0$, $\eta = \{4\hat{\sigma}_d, 8\hat{\sigma}_d\}$); *Log-Normal noise* ($\mu = 0$, $\eta = 0.75\hat{\sigma}_d$); and a *Mixture of two Gaussian noise components* ($\mu_1 = -0.5, \eta_1 = 3\hat{\sigma}_d$, with probability 0.6 and $\mu_2 = 0.5, \eta_2 = 3\hat{\sigma}_d$ with probability 0.4).

Categorical features: The noising process is based on the underlying marginal

(discrete) distribution. We replace the cell value by a dirty one by sampling from a *tempered categorical distribution*[5] (and excluding the current clean category):

$$x_{dc} \sim \frac{p_c(\tilde{x}_d)^\beta}{\sum_{c=1}^{C_d} p_c(\tilde{x}_d)^\beta}, \tag{3.15}$$

with the range $\beta = [0, 0.5, 0.8]$. Notice that, when $\beta = 0$, the noise process reduces to the uniform distribution, while when $\beta = 1$, the noising process follows the marginal distribution.

3.6.2 Evaluation Metrics

In the anomaly detection experiments, we use Average Precision (AVPR) (Salton & McGill, 1986; Everingham et al., 2014), computed according to the anomaly scores for each method. AVPR is a measure of area under the precision-recall curve, so higher is better. For cell outliers we report the macro average of the AVPR for each feature in the dataset[6]. In the repair experiments, different metrics are necessary depending on the feature types. For real features, we compute the Standardized Mean Square Error (SMSE) between the estimated values $\hat{x}_{nd}$ and the original ground truth in the dirty cells $\tilde{x}_{nd}$, normalized by the empirical variance of the ground truth values:

$$SMSE_d = \frac{\sum_{n=1}^{N_c^d} (\tilde{x}_{nd} - \hat{x}_{nd})^2}{\sum_{n=1}^{N_c} (\tilde{x}_{nd} - \overline{x}_d)^2}, \tag{3.16}$$

where $\overline{x}_d$ is the statistical mean of feature d (across all instances $n \in N$) and N_c^d is the number of corrupted cells for that feature[7]. For categorical features, we compute the Brier Score between the one-hot representation of the ground truth $\tilde{x}_{nd}$ and the probability simplex estimated for each category in the feature:

$$Brier_d = \frac{1}{2N_c} \sum_{n=1}^{N_c^d} \sum_{c=1}^{C} (\tilde{x}_{ndc} - p_c(x_{nd}))^2, \tag{3.17}$$

where $p_c(x_{nd})$ is the probability of category c for feature d, $\tilde{x}_{ndc}$ the one-hot true value for category c, and C the number of unique categories in the feature. We used the coefficient $\frac{1}{2}$ in the Brier score to limit the range to $[0, 1]$. We name both metrics as SMSE below for simplicity, but the correct metric is always used for each type.

[5] Also known as *power heuristic* in importance sampling.

[6] The AVPR macro average is defined as the average of the AVPR for all the features in a dataset.

[7] In our experiments $\overline{x}_d = 0$ in practice, since the data has been standardized before using any method

3.6.3 Competing Methods

We compare to several standard outlier detection algorithms. Most methods are only concerned about row outlier detection, whilst only a few can be used for cell outlier detection. For more details on hyperparameter selection and network settings for *RVAE* and competitor methods, see the Additional Notes (Section 3.7.3). In addition, the outlier detection and repair process for *VAEs* and *Deep RPCA* is described in the Background chapter, in Section 2.5.1 and Section 2.5.2.2 respectively. The *VAE-ℓ_2* model in particular is described in Section 2.5.2.1. All models are trained once using all the data instances (rows), once the hyperparameter setting has been selected.

Exclusively row outlier detection. We consider *Isolation Forest (IF)* (Liu et al., 2008), an outlier detection algorithm based on decision trees, which performed quite well in the extensive comparison of Emmott et al. (2015); and *One Class Support Vector Machines (OC-SVM)* (Schölkopf et al., 1999) using a radial basis function kernel. These traditional methods were discussed in Section 2.2.3 of Background chapter.

Row and cell outlier detection. We compare to

- (*i*) estimating the *Marginal Distribution* for each feature and using the negative log-likelihood as the anomaly score. For real features we fit a Gaussian mixture model with the number of components chosen with the Bayesian Information Criterion. The maximum number of components is set at 40. For categorical features, the discrete distribution is given by the normalized category frequency.

- (*ii*) a combination of *OC-SVM and Marginal Distribution* for each feature. We use Platt scaling to transform the anomaly score of OC-SVM for each row (to obtain log-probability), and then combine it with marginal log-likelihood of each feature. This score, a combined log-likelihood, is then used for cell outlier detection.

- (*iii*) *VAEs* with ℓ_2 regularization and anomaly scores given by equation 3.12, additional details in the Background chapter (Section 2.5.2.1).

- (*iv*) *DeepRPCA* (Zhou & Paffenroth, 2017), an unsupervised model inspired by robust PCA. The data $\mathbf{X}$ is divided in two parts $\mathbf{X} = \hat{\mathbf{X}} + \mathbf{S}$, where $\hat{\mathbf{X}}$

is a deep autoencoder reconstruction of the clean data, and $\mathbf{S}$ is a sparse matrix containing the estimated outlier values. Anomaly scores for rows are given by the Euclidean norm $\sqrt{\sum_{d=1}^{D} |S_{nd}|^2}$, whilst cell scores are given by $|S_{nd}|^2$, where $S_{nd} \in \mathcal{S}^e$. For more details on DeepRPCA and its outlier detection or repair process please see Section 2.5.2.2 in Background chapter. For implementation details please see Additional Notes, in Section 3.7.3, where the handling of categorical is also discussed.

- (v) a set of *Conditional Predictors* (*CondPred*), where a neural network parametrizing $p_\theta(\boldsymbol{x}_n)$ is employed for each feature in the data given the rest. This can be seen as a pseudo-likelihood model given by $p_\theta(\boldsymbol{x}_n) \approx \prod_d p_\theta(x_{nd}|\boldsymbol{x}_{n\setminus d})$. A deep version was found to work better than linear in all metrics. However, ℓ_2 regularization is necessary to prevent overfitting, and the model is overall much slower to train than VAE.

Repair: We compare to *VAE*, *DeepRPCA*, *Marginal Distribution* method and *Conditional Predictor* (*CondPred*) method for repairing dirty cells (same model parameters as in outlier detection). We use equation 3.14 for all VAE-based methods. For DeepRPCA the estimate $\hat{\mathbf{X}}$ is used. For CondPred the estimate is $\hat{x}_{nd} = \arg\max_{x_{nd}} p_\theta(x_{nd}|\boldsymbol{x}_{n\setminus d})$, with $\boldsymbol{x}_{n\setminus d}$ meaning all features in $\boldsymbol{x}_n$ except x_{nd}. The Marginal Distribution method takes x_{nd} and uses as estimate the mean of the closest GMM component in the real line. For RVAE, results using a different inference strategy (pseudo-Gibbs sampling, (Rezende et al., 2014)) are provided in Additional Results (Section 3.8.7).

Additional Details

We note that a leave-one-out retraining could have been applied to the CondPred baseline. This is a strategy that can be used in predictor type models (i.e. regression or classification) for outlier detection. However, in practice we trained CondPred on all instances (rows) together just once.

This leave-one-out retraining would exclude each instance (row) in the dataset from training one at a time, and then that trained model would try and predict said instance. In this case, if the instance could not be predicted with confidence, then it would be considered an outlier. Note that we would have to retrain the model once for each instance. The main idea behind this is that the concept of an

inlier would have been learnt by the model during training using the remainder data. Hence, the model should be able to reliably predict inliers at test time. This strategy can be readily applied cell-wise for a row, where a predictor $p_\theta(x_{nd}|\boldsymbol{x}_{n\setminus d})$ is used to predict a particular cell.

Nevertheless, in practice CondPred used the negative log-likelihood of the predictor to measure the outlierness of a cell, which is a common anomaly score. Since we heavily regularized each predictor, the effect of outliers on the training of the models is mitigated (Arpit et al., 2017), which in our experiments yielded good outlier detection performance. In fact, the purpose of RVAE is also to mitigate the contribution of outliers to model parameters (Section 3.5).

Although a leave-one-out retraining would most likely lead to a better performing baseline in terms of outlier detection, it is not clear it would perform better at repair. For instance, the predictor being called upon to repair the cell could have low confidence in proposing a cell value. Moreover, the cost of training several deep or even linear models for each row in a moderately sized dataset is very costly in practice. Once more, we would have to train a predictor for each feature in the dataset, which makes it even more intractable. Note that CondPred is already the most computationally expensive model in the experiments. However, in smaller datasets and using shallower models this type of leave-one-out strategy could be considered as a possibility. In fact, all models that present a log-likelihood for each cell could in theory apply this leave-one-out strategy for outlier detection (e.g. VAEs).

3.6.4 Hyperparameter Selection for Competing Methods

In order to tune the hyperparameters for the competing methods, we reserved a validation set with known inlier / outlier labels and ground truth values. This validation set was **not** used by the RVAE method. Thus the performance obtained by the competitor methods is an *optimistic* estimate of their performance in practice. In contrast, RVAE method had *no access* to the validation set. Note also that RVAE-CVI is robust to the selection of its parameter α in equation 3.10, as we will show in Section 3.6.8. In Figure 3.1 we compare the performance of the conditional predictor method and VAE, with respect to RVAE-CVI when ℓ_2 regularization is not used, and when the best ℓ_2 regularization value is used for each dataset. We term RVAE-CVI-nll our model with anomaly score as defined

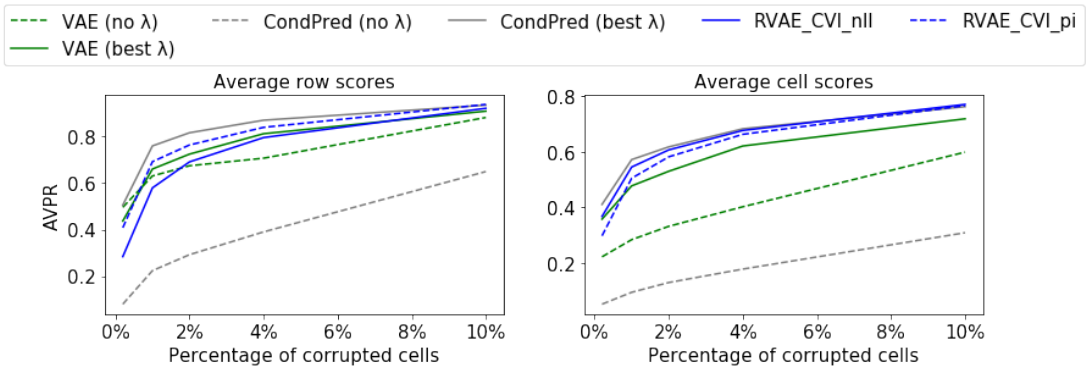


Figure 3.1: Row and cell outlier detection performance (higher means better) of VAE and CondPred methods without L2 regularization and best choice of λ.

in equation 3.12 and RVAE-CVI-pi our model with anomaly score as defined in equation 3.13. We can observe clearly that a significant gap exists in the performance of these competitor methods when not fine-tuned, making explicit the reliance of these methods on a labelled validation set. In the rest of the experiments we will use the best possible version of each competitor method.

3.6.5 Outlier Detection Results

We compare the performance of the difference methods in outlier detection, both at row and cell levels. We focus on Gaussian noise ($\mu = 0, \eta = 5\sigma_d$) for real features and uniform categorical noise, i.e. $\beta = 0$ in equation 3.15, relegating results on other noise processes scenarios to Section 3.6.7. In Figure 3.2 we show the average outlier detection performance across all datasets for all outlier detection models in terms of both row (left figure) and cell outlier detection (right figure). We relegate RVAE-AVI results to the Additional Results (Section 3.8.3), since RVAE-AVI is worse than RVAE-CVI in general. More results on the outlier detection for each dataset are also available in the Additional Results (Section 3.8.1 and Section 3.8.5). In the right figure, we observe that RVAE-CVI is performing similar to the conditional predictor method on cell outlier detection while being consistently better than the other methods. Additionally, it performs comparatively well in row outlier detection, being similar to the conditional predictor at higher noise levels. We remind the reader that RVAE-CVI does not need a validation set to select its parameters. This means that RVAE-CVI is directly applicable for datasets where no ground truth is available, providing a comparable performance to other methods where parameter tuning for each dataset is necessary. Figure 3.2

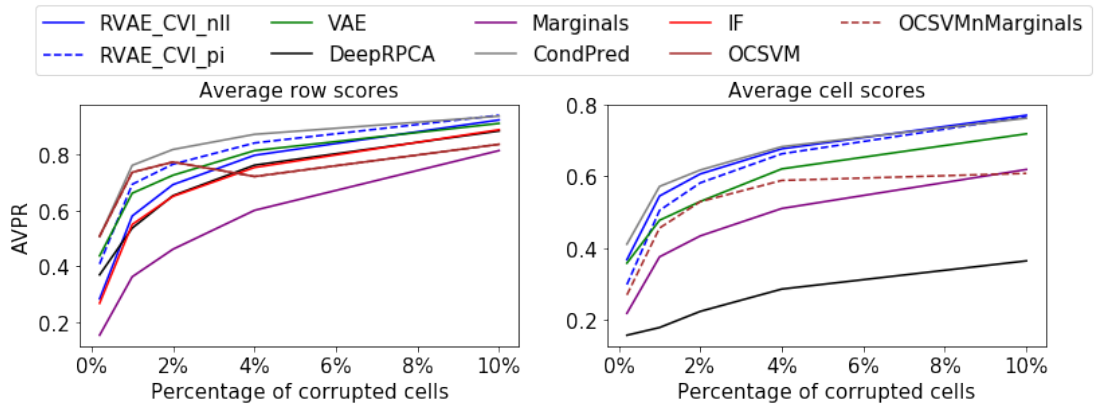


Figure 3.2: Row and cell outlier detection scores for the average of the four datasets in 5 different cells corruption levels. Left: AVPR at row level. Right: AVPR at cell level.

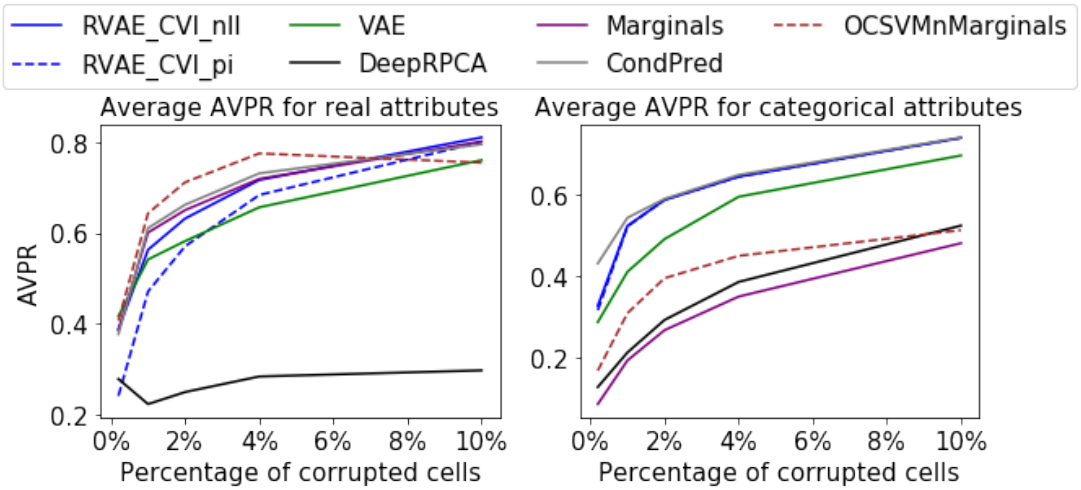


Figure 3.3: Average AVPR over all the features in the four datasets partitioned by type. Left: AVPR for real features. Right: AVPR for categorical features

(left figure) also confirms our hypothesis (Section 3.5.3) on the proper score to compute row outliers. We can see in the upper figure that RVAE-CVI using scores based on estimate $\hat{\pi}_d(\boldsymbol{x})$, as per score equation 3.13, are better for row outlier detection compared to averaging different feature log-likelihoods equation 3.12. Further analysis of the outlier detection performance of each model for the different feature types is shown in Figure 3.3. While the model based on estimating the marginal distribution works well for real features, it performs poorly on categorical features. Similarly the method combining OCSVM and the marginal estimator detects outliers better than the other methods in real features and low noise levels, but performs poorly for categorical features. In contrast, RVAE performs

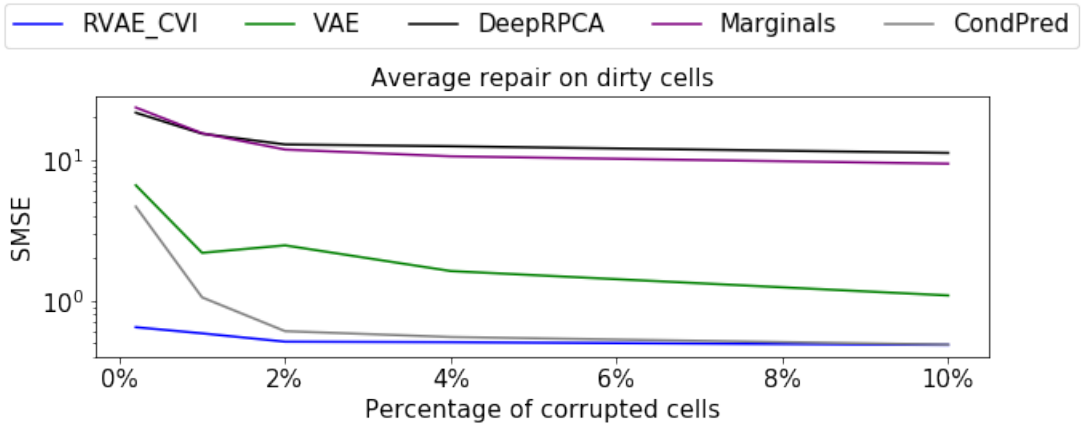


Figure 3.4: SMSE computed over the dirty cells in all datasets (lower means better). It shows the average over the four datasets for 5 different noised cells percentages. Y-axis is provided in log-scale.

comparatively better across different types than the other models, with comparable performance to the conditional predictor.

3.6.6 Data Repair Results

In this section, we compare the ability of the different models to repair the corrupted values in the data. We use the same noise injection process as in Section 3.6.5. Figure 3.4 shows the average SMSE repair performance across datasets for all models when repairing the dirty cells in the data (more details in the Additional Results, Sections 3.8.2 and 3.8.5). We can observe that RVAE-CVI outperforms the other models for all the different cell corruption scenarios, being of particularly significance in lower cell corruption regimes. This is significantly important since all the comparator methods required hyperparameter selection and still performed worse than RVAE-CVI. Also, in Figure 3.5 we can see the repair performance of different models according to the types of features in the data. Notice that RVAE-CVI is consistently better than the other models across real features while being slightly worse on categorical features.

3.6.7 Robustness to Noising Processes

Figure 3.6 shows the performance of the different models across different combinations of noise processes for all datasets and noise corruption levels (three other noise processes are covered in the Additional Results, Section 3.8.4). We

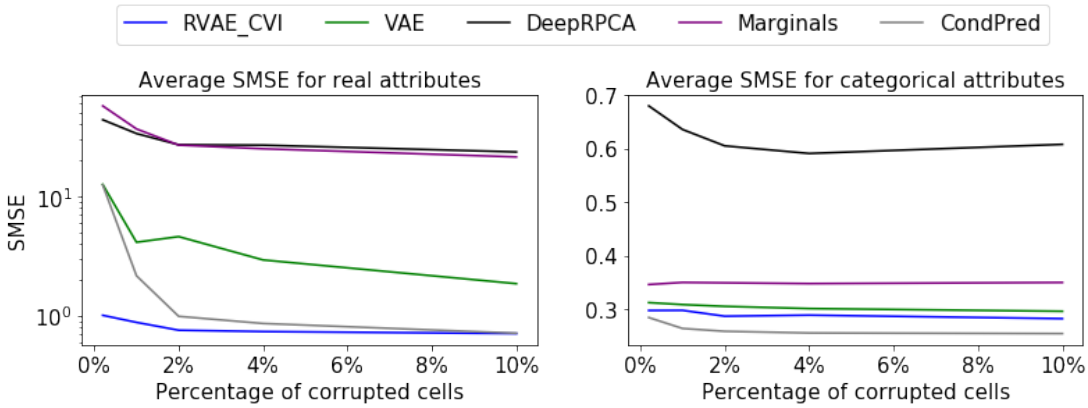


Figure 3.5: Average SMSE over all the features in the four datasets according to their type. Left: AVPR for real features. Right: AVPR for categorical features

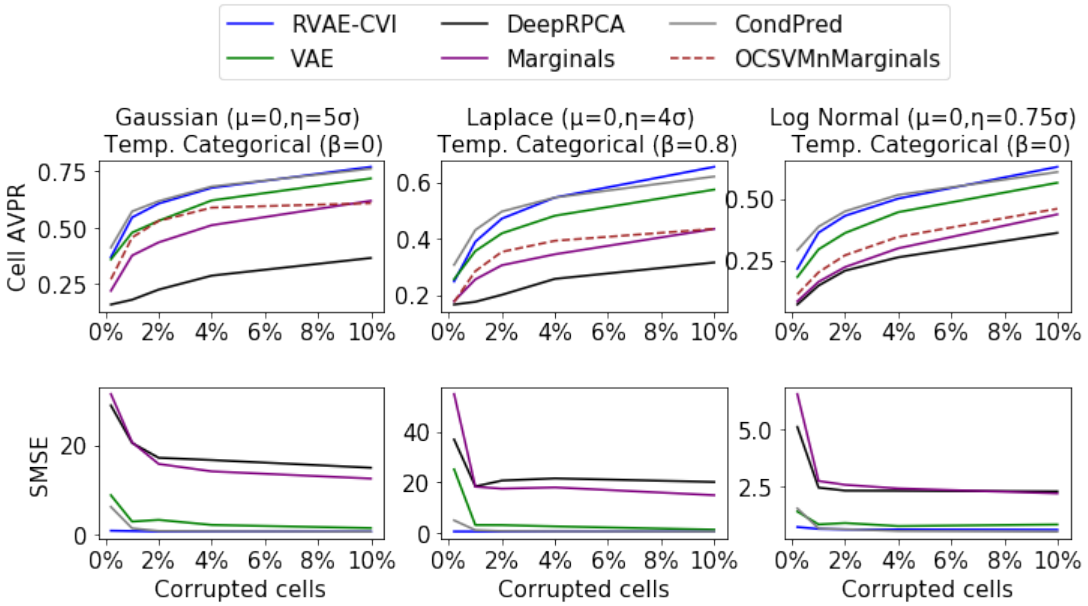


Figure 3.6: Effect of three different noising processes. Upper figures: average cell outlier detection across datasets. Lower figures: average SMSE on the dirty cells

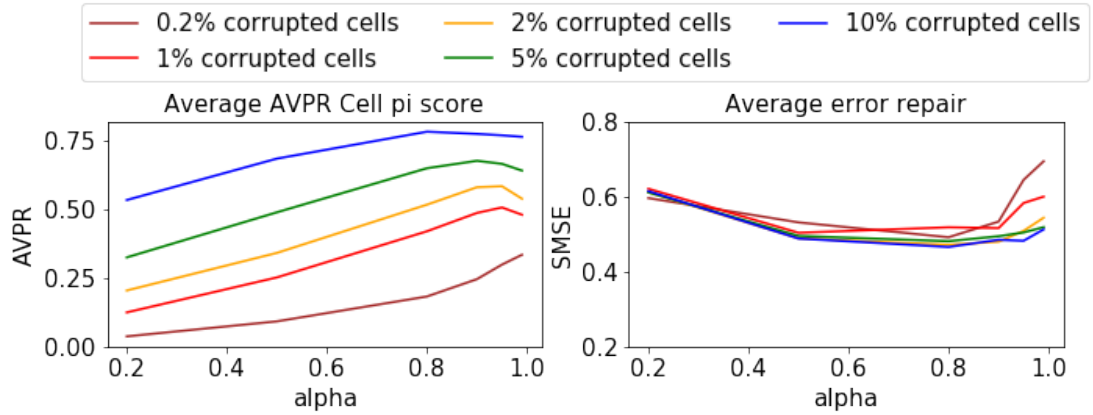


Figure 3.7: RVAE-CVI performance with different choices for α. Left: average cell AVPR over the datasets. Right: average repair over dirty cells

notice that all the models perform consistently across different types of noise. RVAE-CVI performs better in repair for low-level noise corruption, while providing competitive performance in outlier detection. Also, our choice of outlier models on Section 3.5.1 does not have a negative effect on the ability of RVAE to detect outliers and repair them. Different noise processes define what is feasible to detect and repair.

3.6.8 Robustness to Hyperparameter Values

In this section, we examine the robustness of RVAE inference to the choice α, and study its effect in both outlier detection and repair of dirty cells. We have analyzed values of α in the set $\{0.2, 0.5, 0.8, 0.9, 0.99\}$ and evaluated RVAE-CVI in all datasets under all levels of cell corruption and the noising process of Sections 3.6.5 and 3.6.6. Figure 3.7 shows the performance of RVAE-CVI in both outlier detection (left figure) and repair (right figure) across different values of α. Larger values of α lead in general to a better outlier detection performance, with a slight degradation when we approach $\alpha = 1$. Repair performance is consistent across different choices of α, but values closer to 0 or 1 lead to a degradation when repairing dirty cells.

3.7 Additional Notes

In this section we provide additional details and notes about the datasets and implementation details of the models used in the experiments (Section 3.6). We

also discuss hyperparameter selection and data pre-processing. The derivation of the coordinate step update for $\hat{\pi}_d(\boldsymbol{x})$ in eq. (3.10) is also found here.

3.7.1 Dataset details

Dataset	Rows	Real features	Categorical features
Wine	6497	12	1
Adult	32561	5	10
Credit Default	30000	14	10
Letter	20000	0	17

Table 3.1: Properties of the tabular datasets employed in the experiments.

3.7.2 Derivation of Coordinate Step for Weights

Here we derive eq. 3.10, which is the exact expression for $\hat{\pi}_d(\boldsymbol{x})$ to be used in coordinate ascent optimization. Given eq. (3.5) and eq. (3.6), we can write the bound $\mathcal{I}$ on $\sum_{\boldsymbol{x}\in\mathcal{X}} \log p(\boldsymbol{x})$ with respect to $\pi_d(\boldsymbol{x})$ as

$$
\begin{aligned}
\mathcal{I} \;\propto\; & \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{d=1}^{D} \pi_d(\boldsymbol{x}) \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(x_d|\boldsymbol{z})] \\
+ \; & \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{d=1}^{D} (1-\pi_d(\boldsymbol{x}))\, \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_0(x_d)] \\
- \; & \pi_d(\boldsymbol{x}) \log \frac{\pi_d(\boldsymbol{x})}{\alpha} \\
- \; & (1-\pi_d(\boldsymbol{x})) \log \frac{1-\pi_d(\boldsymbol{x})}{1-\alpha}
\end{aligned}
$$

The derivative of this bound w.r.t. $\pi_d(\boldsymbol{x})$ can be easily computed:

$$
\begin{aligned}
\frac{\partial\, \mathcal{I}}{\partial \pi_d(\boldsymbol{x})} \;=\; & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(x_d|\boldsymbol{z})] \\
- \; & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_0(x_d)] \\
- \; & \log \frac{\pi_d(\boldsymbol{x})}{\alpha} + \log \frac{1-\pi_d(\boldsymbol{x})}{1-\alpha}
\end{aligned}
$$

Evaluating $\frac{\partial \mathcal{I}}{\partial \pi_d(\boldsymbol{x})} = 0$ and solving for $\pi_d(\boldsymbol{x})$, we obtain the coordinate update for the weights:

$$\hat{\pi}_d(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\left(\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log \frac{p_\theta(x_d|\boldsymbol{z})}{p_0(x_d)}] + \log \frac{\alpha}{1-\alpha}\right)\right)},$$

which *is the sigmoid function* applied to the *expected log density ratio* between the clean model and the outlier model plus the logit of the prior probability.

3.7.3 Additional details for RVAE and Competing Methods

- **Data Pre-Processing:** For all models and competitor methods the real features were standardized, i.e. subtracting by the empirical mean and dividing by standard deviation. One-hot encoding for categorical features was used depending on the method, as defined below.
- **Validation Set**: 10% of each dataset was separated from the rest of the data to be employed as a validation set, with known ground truth of the corrupted cells, for hyperparameter selection on all baselines. Our RVAE model does not use this validation set in any of the experiments.
- **Hyperparameter Selection:** The criterion used for hyperparameter selection on all baselines was the AVPR in the outlier detection task registered in the validation set. The exception is the Marginals Distribution baseline, where the number of components is chosen via BIC score.

3.7.3.1 RVAE, VAE, DeepRPCA and Conditional Predictor methods

- **Architecture:** For VAE, RVAE and DeepRPCA, we used an intermediate hidden layer in both encoder and decoder, size 400. The latent space dimension was chosen to be size 20. In the CondPred baseline, we found that a deep version of the base conditional predictor was superior than a linear one in both outlier and repair metrics. Two inner layers of dimension 200 and 50 for each predictor were employed, which made this model substantially slower than all autoencoder baselines. The non-linear activation used throughout was ReLU (Rectified Linear Unit).
- **Optimization:** We used the Adam optimizer as provided in Pytorch to train the encoder and decoder parameters, for all VAE-based models. In the case of RVAE, VAE and CondPred models we minimized their respective

negative losses. In CondPred, each conditional predictor had its own Adam optimizer, we found this to work better. The initial learning rate used in experiments was 0.001. All models ran for 100 epochs on all datasets, noise levels and noise processes. Since access to a validation set is impossible in a unsupervised learning setting, no standard early stopping can be defined.

In the case of DeepRPCA, we use Adam to train the encoder and decoder parameters, as in the original paper. The optimization process used to obtain data matrix $\hat{\mathbf{X}}$, and noise matrix $\mathbf{S}$, was carried out using ADMM (Alternating Method of Multipliers). We use row structured $\ell_{2,1}$ version of DeepRPCA for outlier detection as it performed better. In order for the ADMM optimization procedure to work, in terms of categorical reconstruction loss we follow the work in (Udell et al., 2016) (Section 6, Categorical PCA), using cross-entropy loss to aggregate the different one-hot dimensions. This yielded better experimental results than one-vs-all type aggregation. All models ran for 20 ADMM iterations, each using 10 intermediate epochs of Adam to train the autoencoder component $\hat{\mathbf{X}}$. All the above are in accordance to DeepRPCA paper (Zhou & Paffenroth, 2017). It should be noted that, in our experiments, running more ADMM iterations eventually led to performance degradation, even after an extensive hyperparameter search and optimizer tuning.

- ℓ_2 **Regularization (Weight Decay):** We used the weight decay option of the Adam optimizer in Pytorch. We performed a grid search over the values $\lambda_{\ell_2} = [0, 0.1, 1, 5, 10, 100]$, each run for 100 epochs, and chose the best on the validation set The search was performed for each dataset in Table 1. For VAE, the best performance was obtained with we $\lambda_{\ell_2} = 0.1$ in the Letter dataset, $\lambda_{\ell_2} = 1$ in the Adult dataset and $\lambda_{\ell_2} = 10$ in the Wine and Credit Default datasets. For the conditional predictors, the best performance was obtained for $\lambda_{\ell_2} = 1$ in Adult, Credit default and Letter datasets, and $\lambda_{\ell_2} = 5$ in the Wine dataset. For RVAE-CVI and RVAE-AVI no regularization was needed.

- **Categorical Encoding:** VAE, RVAE and CondPred models we used categorical embedding matrices to codify the categorical features at the input level of the encoder. The dimensionality used in all experiments was size 50, as it provided generally good results. For CondPred, embeddings were not

shared between individual feature predictors. In the case of DeepRPCA we had to use one-hot encoding, as this was the only way to make the ADMM procedure to work properly, given the projection step (using proximity operator). This relies on subtracting the noise matrix $\mathbf{S}$ from the data matrix $\mathbf{X}$, which is non-trivial using embedding representations. One-hot encoding is standard in PCA-type models when dealing with categorical features.

- **DeepRPCA hyperparameter:** The coefficient that regulates how many of the data-points (cells) will be represented by sparse matrix $\mathbf{S}$ was chosen from the range $\lambda = [0.001, 0.01, 0.1, 1]$. The best outlier detection performance was obtained for 0.01 in Wine and Adult datasets and 0.1 in Credit Default and Letter datasets.

- **RVAE (hyperparameters):** The value for the prior probability α was set to 0.95 throughout (it is fair to assume in general that most of the data is clean). A full evaluation on its effect on the performance of the model was conducted in the main text. In the case of the hyperparameter $\mathbf{S}$ of the outlier model for real features, we used 2 throughout, with good results. This was the setting used for all RVAE-based models in the experiment section, and the validation set was not employed at any time while selecting parameters.

- **Encoder of the weights for RVAE-AVI:** We used a feed-forward neural network with the same architecture as the one specified above for the encoder of VAE, which parameterizes the variational distribution of the latent space. In this case the $\pi_{\tau\, d}(\boldsymbol{x})$'s are parameterized directly by a neural network, where τ defines the neural network parameters (see Algorithm 1). An intermediate hidden layer of size 400 was used. Hence, no coordinate optimization procedure was performed.

3.7.3.2 OC-SVM

We use a scikit-learn implementation, with RBF (radial basis function) kernel. We conducted an hyperparameter search on both ν and γ, from 0 to 1 in intervals of 0.1. The best performance for all the datasets was obtained with $\nu = 0.2$ and $\gamma = 0.1$, on the validation set.

3.7.3.3 Marginal Method

The Marginal method has no hyperparameters to tune, apart from the maximum number of Gaussian Mixture Model components that can be selected by BIC score. We found a maximum of 40 components to be sufficient.

3.7.3.4 OCSVM + Marginals Method

We employed a combination of both the OCSVM and Marginals implementations described above. The parameters were selected based on the previous details ($\nu = 0.2$,$\gamma = 0.1$ and maximum number of components of the GMM set to 40).

3.7.3.5 Isolation Forest

We use scikit-learn implementation. A maximum number of samples of 50% of the size of the datasets, and a contamination parameter of 0.2 seemed to work best for all the scenarios. Again, these parameters were selected using the validation set.

3.7.3.6 Timing Information on Models

The linear models (OCSVM, Marginals) and Isolation Forest in this chapter have a computing time in terms of a few minutes in a desktop CPU (3 GHz) with 8GB of memory. For instance, Isolation Forest or OCSVM can take about 3-8 minutes depending on the size of the dataset, and usually Isolation Forest takes more time. The model OCSVM + Marginals usually takes a couple more minutes than the other simpler models, since both linear models have to be trained, and then include Platt scaling for the OCSVM score.

The deep learning models take significantly longer than the standard models above-mentioned. The VAE and RVAE models take an order of magnitude more than the standard ones. Using a desktop CPU with 8 GHz it can take about 40 minutes for RVAE, which can be longer depending on the size of the dataset. Using a GPU (GeForce TITAN X) we can reach a lower computing time of about 10-20 minutes. DeepRPCA will take longer when using a GPU at about 35 minutes, since the ADMM procedure overall is costly. The most expensive model in terms of computation time is CondPred and is strongly dependent on the dimensionality of the dataset (number of features), since for each feature an additional neural network model is trained. For the experiments carried out, CondPred can easily

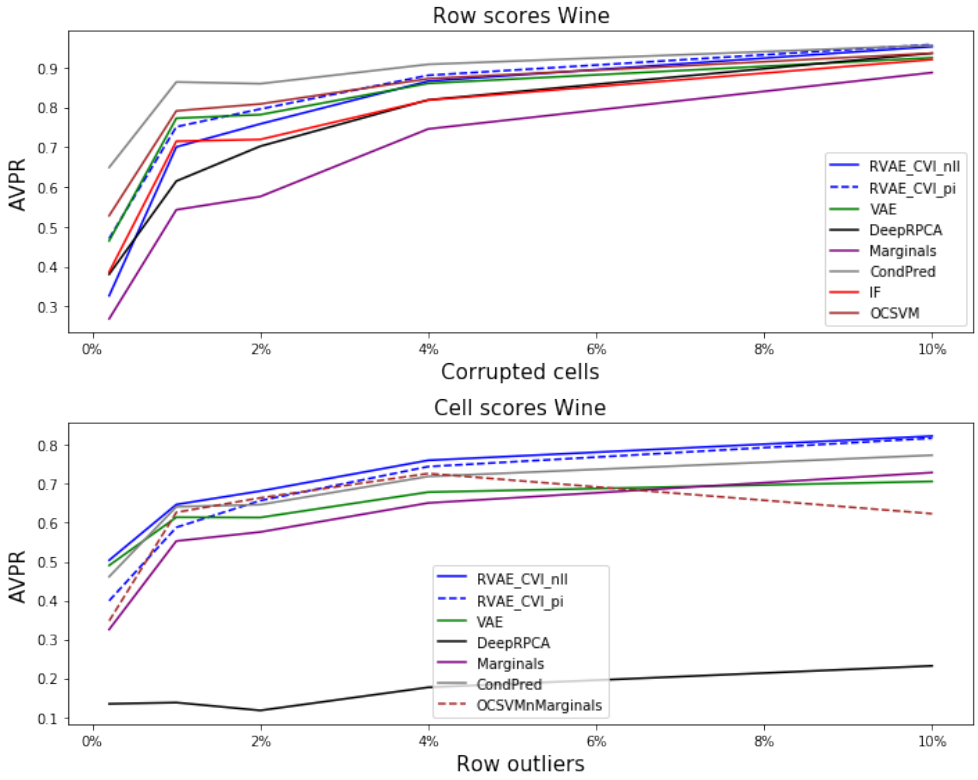


Figure 3.8: Row and cell outlier detection scores on Wine dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

take two times as long as VAE and RVAE models. Generally, it can take up to an hour sometimes when using a GPU.

3.8 Additional Results

The purpose of this section is to provide a **complement of results to the main results sections** (Sections 3.6.5 and 3.6.6) and show the reader the complete set of experiments. As such, we present additional figures for the experiments carried out in this chapter.

3.8.1 Outlier detection additional details

In this section, we present the full disclosure of all the models in both row and cell outlier detection in each of the datasets of the experiments, in Figures 3.8-3.11

Notice that RVAE-CVI is stable across datasets and noise corruption levels,

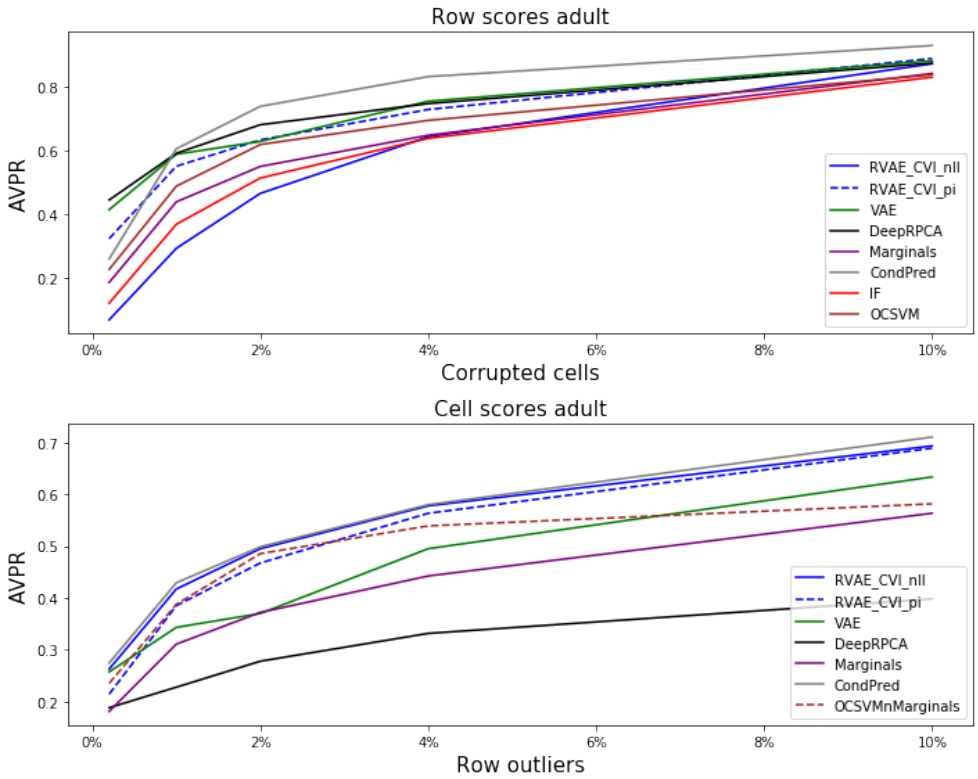


Figure 3.9: Row and cell outlier detection scores on Adult dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

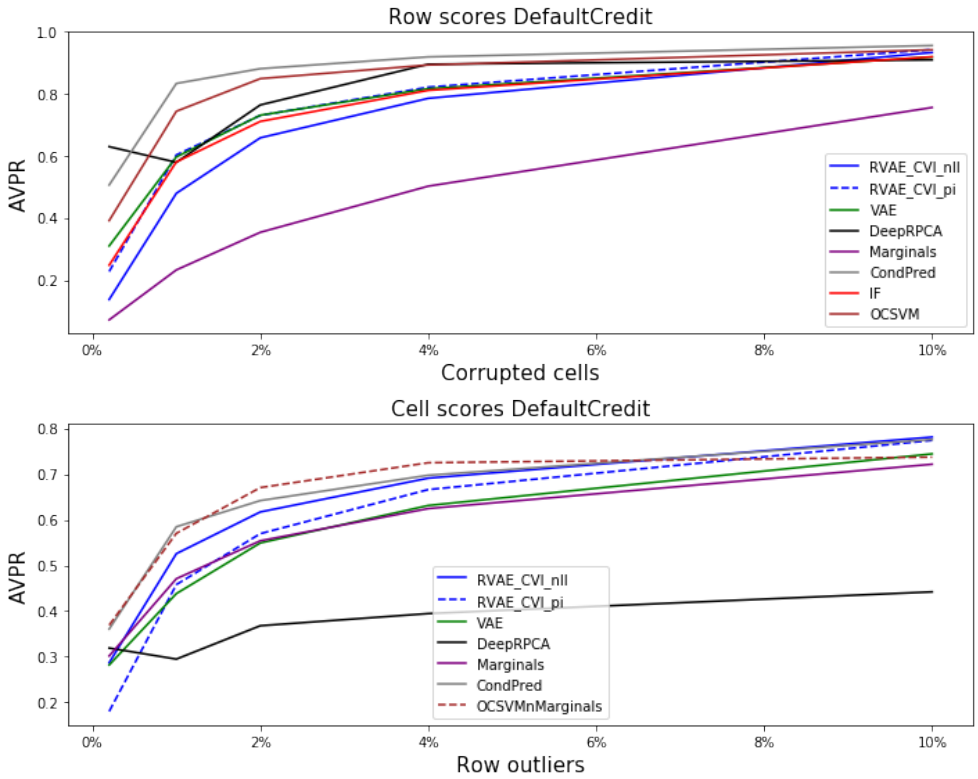


Figure 3.10: Row and cell outlier detection scores on Credit default dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

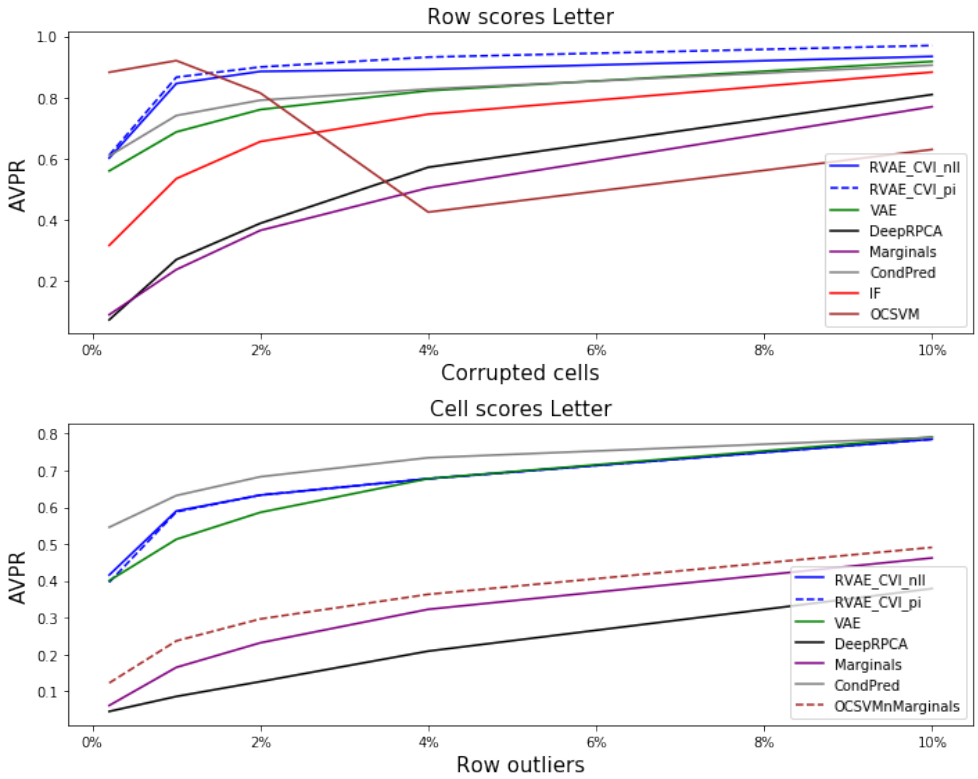


Figure 3.11: Row and cell outlier detection scores on Letter dataset in 5 different cells corruption levels. Upper figure shows the AVPR at row level. Lower figure shows the AVPR at cell level.

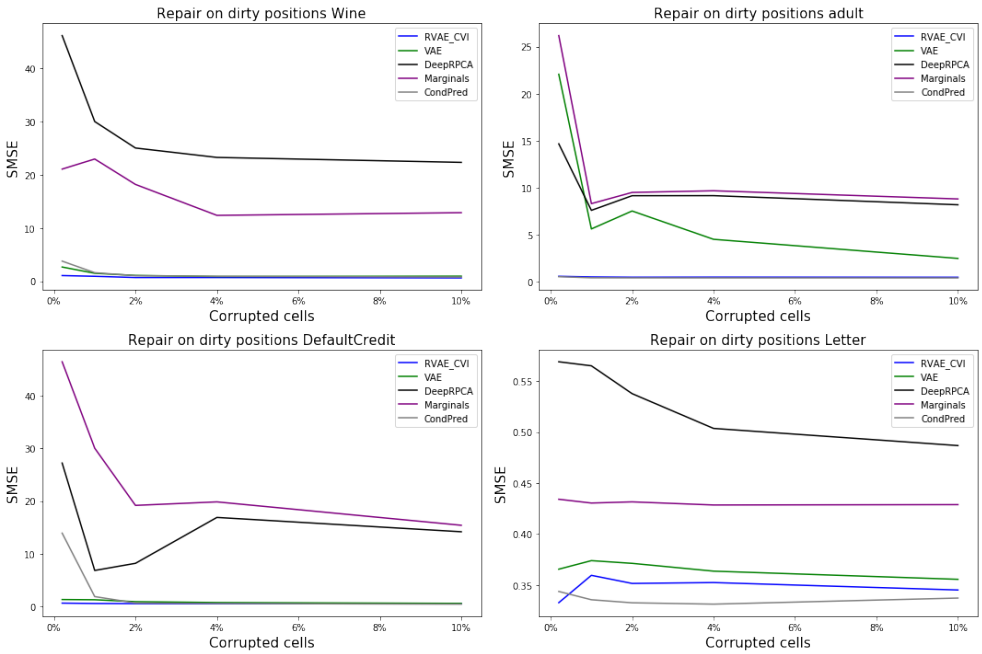


Figure 3.12: Repair performance on the dirty cells of all models for each datasets

while other models suffer in some specific datasets for either row or cell outlier detection.

3.8.2 Repair additional details

In this section, we present the full disclosure of all the models in while repairing dirty cells in each of the datasets of the experiments, in Figure 3.12. RVAE-CVI performs better than the other methods for low level corruption, except for the adult dataset where RVAE-CVI and the conditional predictor are equivalent and the Letter dataset, where the conditional predictor does slightly better.

3.8.3 RVAE-CVI vs RVAE-AVI

We present here the AVPR evolution of RVAE-CVI and RVAE-AVI for each dataset and all noise corruption levels. RVAE-CVI outperforms RVAE-AVI in all datasets in both cell and row outlier detection, obtaining a similar performance only for the Letter dataset.

Additionally, in Figure 3.14 we show the difference in repair performance of the dirty cells for both models. We can observe that RVAE-CVI performs better than RVAE-AVI for all datasets and noise corruption levels.

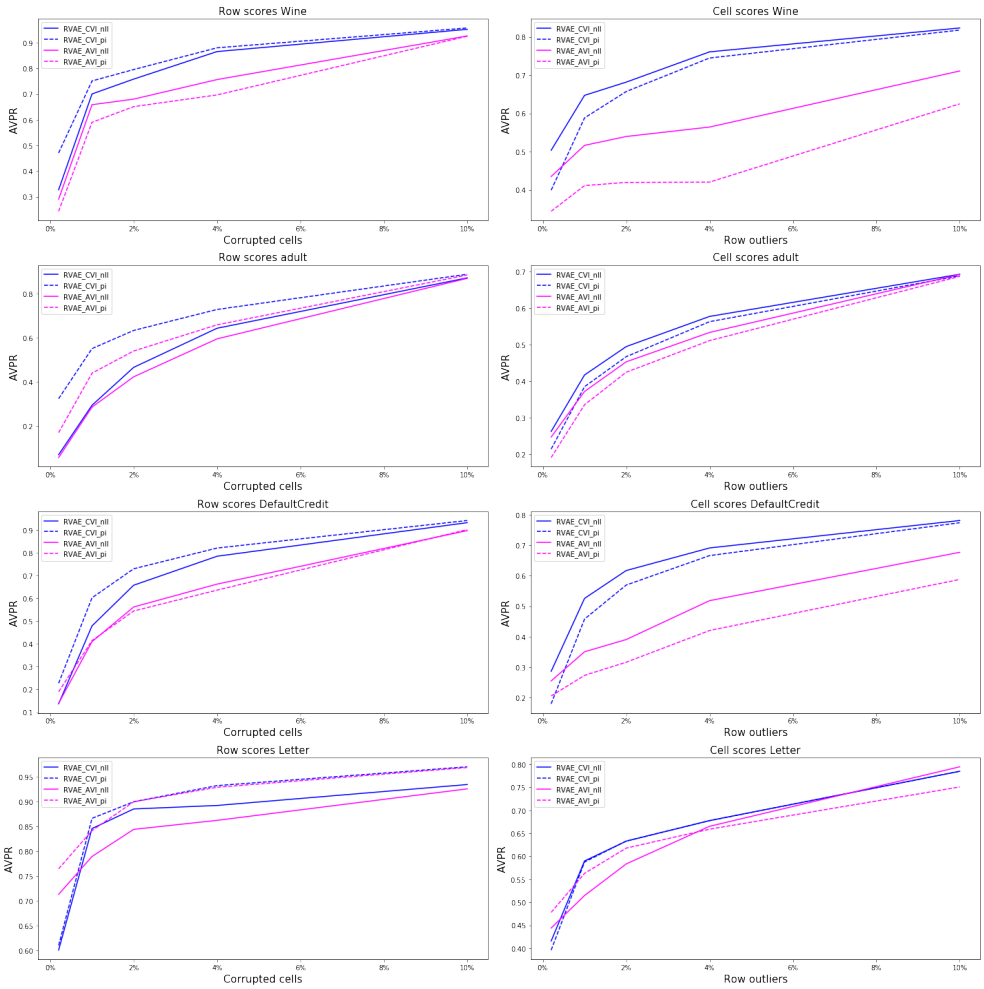


Figure 3.13: Comparison between RVAE-CVI and RVAE-AVI for each dataset in row outlier detection (left figures) and cell outlier detection (right figures)

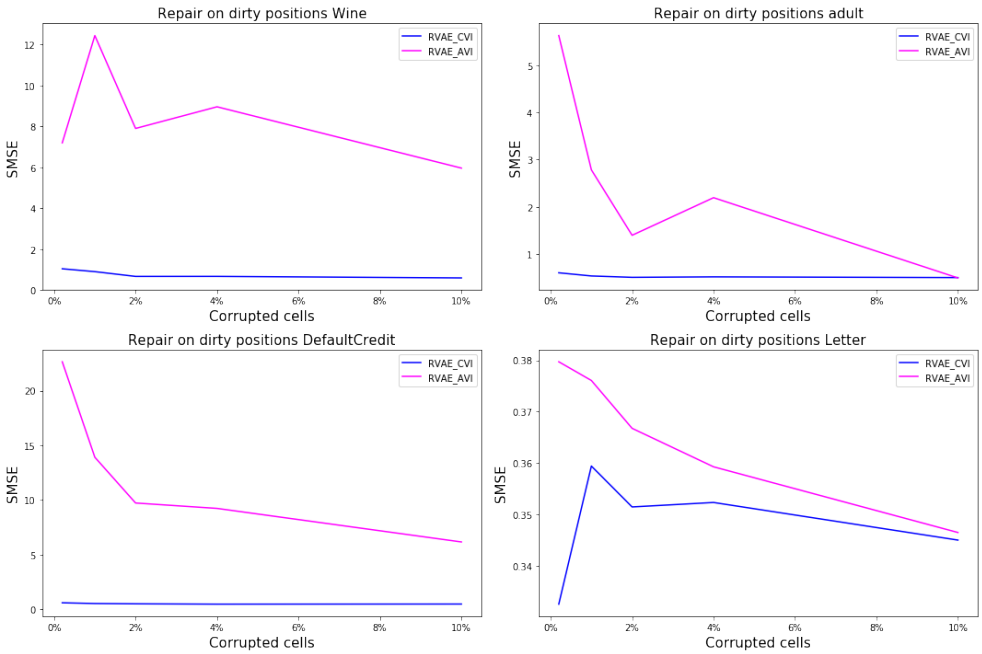


Figure 3.14: Comparison between RVAE-CVI and RVAE-AVI for each dataset in repair of dirty dells. The lower SMSE the better.

3.8.4 Different noise processes additional details

In this section we present all the results in row and cell outlier detection and repair for all six combinations of noise processes, which are:

- Gaussian noise ($\mu = 0$, $\eta = 5\hat{\sigma}_d$), Tempered Categorical ($\beta = 0$)
- Laplace noise ($\mu = 0$, $\eta = 4\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.5$)
- Laplace noise ($\mu = 0$, $\eta = 4\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.8$)
- Laplace noise ($\mu = 0$, $\eta = 8\hat{\sigma}_d$), Tempered Categorical ($\beta = 0.8$)
- Log normal noise ($\mu = 0$, $\eta = 0.75\hat{\sigma}_d$), Tempered Categorical ($\beta = 0$)
- Mixture of two Gaussian noise components ($\mu_1 = -0.5, \eta_1 = 3\hat{\sigma}_d$, with probability 0.6 and $\mu_2 = 0.5, \eta_2 = 3\hat{\sigma}_d$ with probability 0.4), Tempered Categorical ($\beta = 0$)

Figures 3.15-3.17 show a disclosure of the full results on all noise processes across the different models for both row and cell outlier detection and repair.

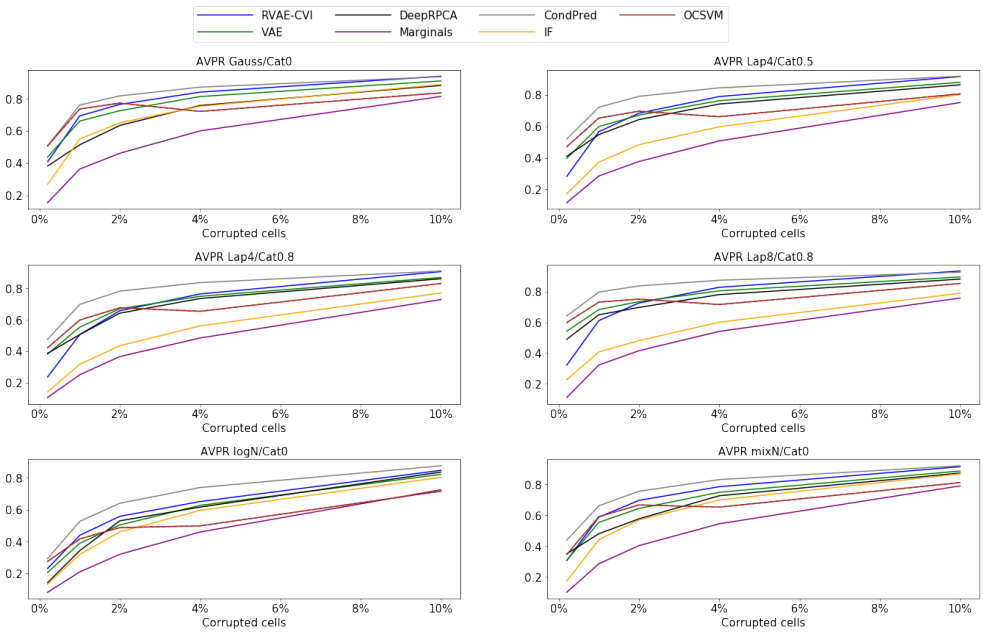


Figure 3.15: Row outlier detection across all models and noise processes, averaging all datasets

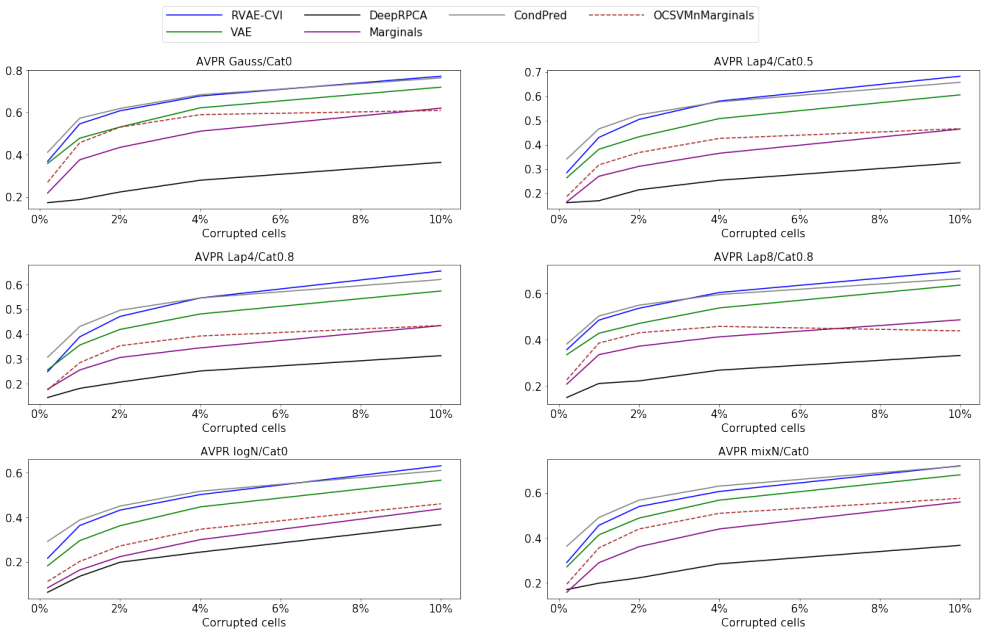


Figure 3.16: Cell outlier detection across all models and noise processes, averaging all datasets

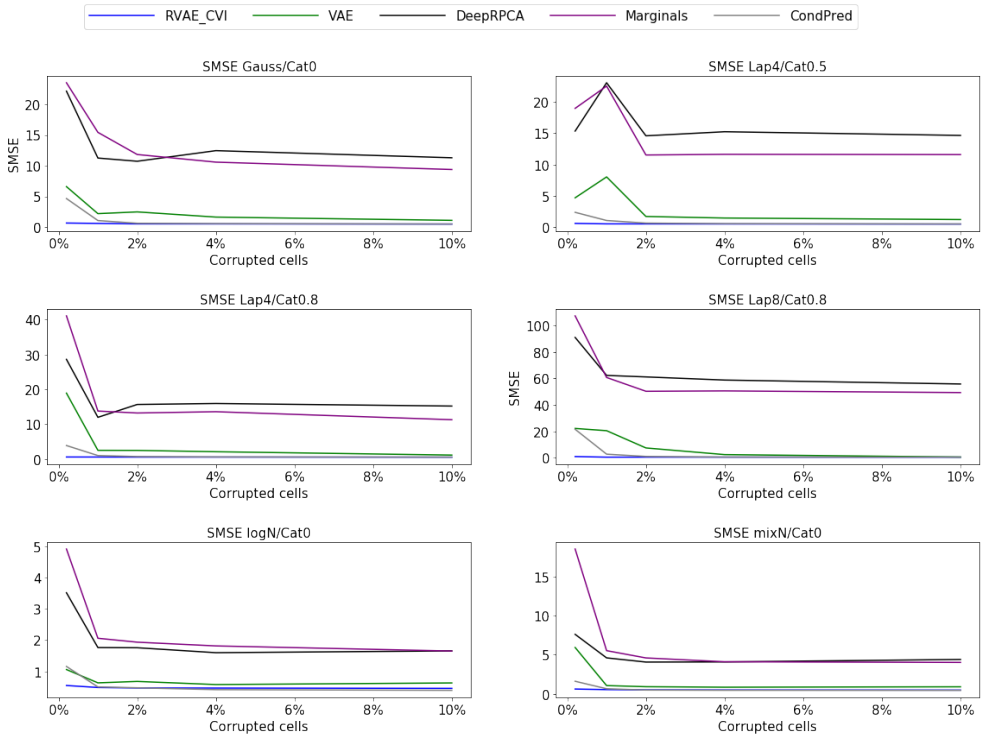


Figure 3.17: Repair of dirty cells across all models and noise processes, averaging all datasets

3.8.5 Error Bars per Noise Level

Here, we show results for VAE, RVAE and CondPred with error bars provided for each noise level. The error bars were obtained by generating five independent instances of corruption – randomly corrupting different cells in the dataset each time. The corruption process is the same as in Section 3.6.5. The inference mechanism for repair is MAP like in Section 3.6.6. We report the results both for outlier detection and repair (Figure 3.18). In lower noise levels, the standard deviation tends to be higher, more significantly in repair (last row, Figure 3.18). Since fewer cells are affected at lower noise levels, this leads to more diverse behaviours in repair and outlier detection, and thus to larger error bar.

We can see that the main conclusions about the "ranking" of our method against baselines still holds in either outlier detection or repair. Further, in repair, in the two lowest noise levels RVAE (MAP) seems to less dependent on the corrupted cells (see Adult and Credit Default figures, in Figure 3.18).

To further complete this analysis, we provide in Table 3.2 the p-values computed from an independent t-test between RVAE, VAE and CondPred. These were

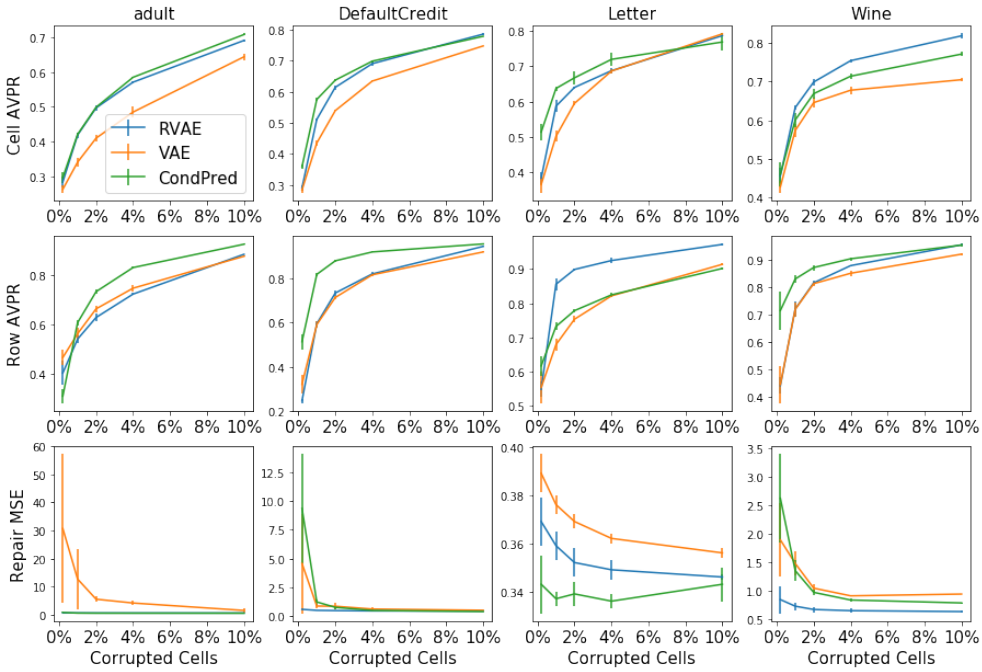


Figure 3.18: Plots with error-bars for each dataset (column), using 5 different instances of corruption, at each corruption level (x-axis). We show cell outlier detection (upper row), row outlier detection (middle row) and repair (lower row).

averaged across datasets and noise levels.

	avg. p-values RVAE vs CondPred	avg. p-values RVAE vs VAE
Cell AVPR	0.121	0.070
Row AVPR	0.040	0.227
Repair SMSE	0.025	0.013

Table 3.2: Independent t-test between RVAE, VAE and CondPred. If p-values in range 0.05-0.10 assume that models have different performance.

3.8.6 Different Outlier Detection Task: RVAE vs ABDA

In this section we compare RVAE to ABDA (Vergari et al., 2019), a recent algorithm employed both in outlier detection and missing data imputation. We followed the details in the outlier detection section of the ABDA paper and compare RVAE with ABDA in terms of row AUC ROC as used therein (we use the results reported by the ABDA authors). Table 3.3 shows that we perform

Dataset	AUC RVAE	AUC ABDA
Letter	**0.8359**	0.7036
Breast	0.9815	**0.9836**
Pen Global	**0.9316**	0.8987
Pen Local	0.9053	**0.9086**
Satellite	**0.9460**	0.9455
Thyroid	0.8211	**0.8488**
Shuttle	**0.9985**	0.7861
Aloi	**0.5515**	0.4720
Speech	**0.5584**	0.4696
KDD	**0.9993**	0.9979
Average	**0.8529**	0.8014

Table 3.3: Comparison between RVAE and ABDA in row AUC ROC for 10 different datasets.

better in average than ABDA, with 7 out 10 cases being better in outlier detection. Notice that, the noising scenarios for these datasets (described in (Goldstein & Uchida, 2016)) are based of standard row outlier detection, where one or some classes are considered normal while another class or classes are considered outliers. This scenario is completely different to the scenarios described in this chapter. In our work, we assume that some cells in the data corrupt several rows in a tabular dataset, and we need to detect and correct them. These experiments showcase the robustness of RVAE to a different outlier detection process.

3.8.7 Different Inference Method

In this section, we compare the MAP inference (reconstruction, eq. 3.14) for VAEs employed throughout the chapter with more powerful inference methods (Figure 3.19). In particular, we provide results for pseudo-Gibbs sampling, (see Rezende et al., 2014, section F), applying it on a trained RVAE at evaluation time. The final repair estimate was provided after running the MCMC procedure for $T = 5$ iterations (samples), since larger values of T provided marginal improvements. We used the same scenarios of Sections 3.6.5 and 3.6.6.

A mask removing anomalous entries needs to be either defined, or inferred. We

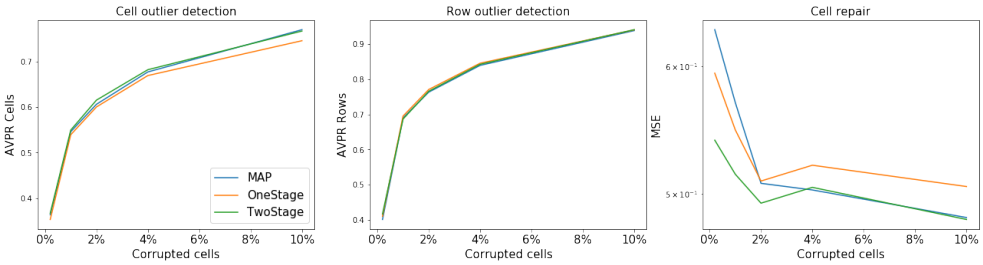


Figure 3.19: Comparison between MAP, OneStage, TwoStage inference methods in terms of both row / cell outlier detection, and repair.

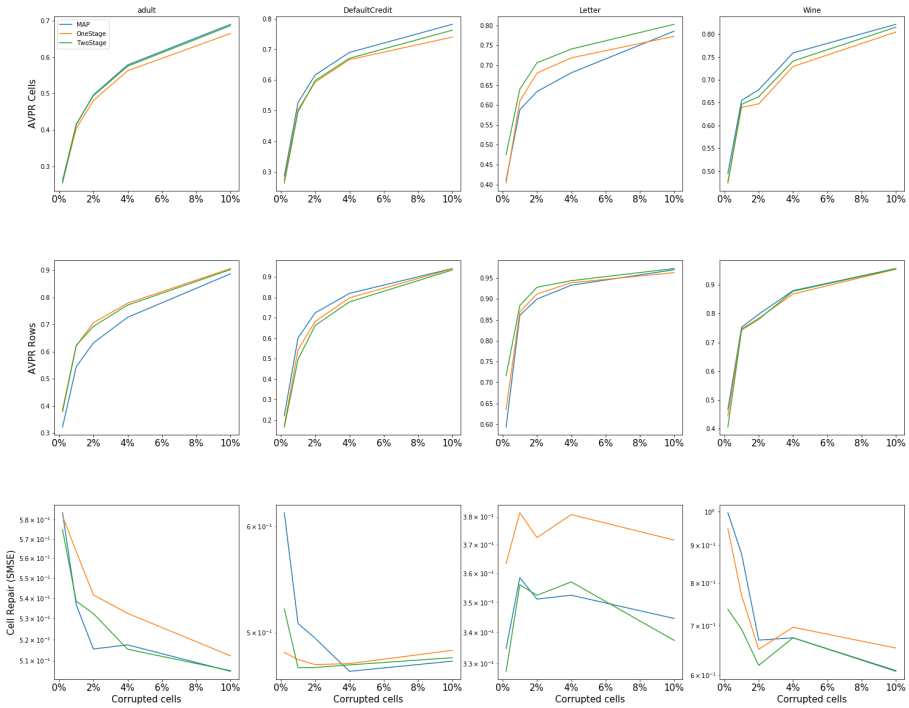


Figure 3.20: Comparison between MAP, OneStage, TwoStage, CondPred inference methods in terms of both row / cell outlier detection, and repair. Results for each dataset.

provide two options to do this automatically:

- **OneStage** (Algorithm 2): Treat all cells in a row as anomalous and perform pseudo-Gibbs, for T iterations. Final repair value is line 6 of Algorithm 2, and outlier detection is line 8.
- **TwoStage** (Algorithm 3): Use OneStage, obtaining a more stable estimate of π_d, then sample mask w_d using it to perform pseudo-Gibbs (as described in (Rezende et al., 2014)). The assumed clean cells (i.e. $w_d = 1$) have their value x_d fixed throughout the MCMC chain (of T iterations). Meanwhile cells that are dirty are initialized with mean behaviour imputation, i.e. $\bar{x}_d$ (Algorithm 3, line 4). For continuous features since our data is standardized ($\hat{\mu}_d = 0$), so we use 0. For categorical features, given our VAE models use normalized (word) embeddings, we use vectors of the same dimension with zeros – such strategy has been applied for imputation when using embeddings. Final repair value is line 8 of Algorithm 3, and outlier detection is in line 2 (i.e. $\hat{\pi}$ from *OneStage*).

Note that in the *OneStage* method the mask $\boldsymbol{w}$ is not inferred, while in *TwoStage* it is. In addition, we remind the reader that $\boldsymbol{x}$ is the observed row, which can be clean or dirty.

Figure 3.19 shows that there were gains on average in outlier detection and repair using *TwoStage*, particularly for repair at low noise levels. These are still close to MAP, specifically in the case of higher noising levels.

For completion, we disclose in Figure 3.20 the comparison across the inference methods per dataset. In general, we can see that TwoStage has better repair performance (last row of Figure 3.20), particularly in low level noise.

Lastly, other methods like (Mattei & Frellsen, 2019) could also have been used to improve repair. However, more powerful inference schemes can sometimes lead to overfitting to noise. On the other hand, inference schemes like MCMC (vs MAP) can provide more stable solutions (lower error bars), particularly in lower noise levels or in smaller datasets (number of rows).

Algorithm 2 OneStage: pseudo-Gibbs sampling

procedure ONESTAGE(T, fixed $\{\phi, \theta\}$)
  $\boldsymbol{x}^{(1)} = \boldsymbol{x}$
  for $1, ..., T$ **do**
    $\boldsymbol{z}^{(t+1)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(t)})$
    $\tilde{\boldsymbol{x}}^{(t+1)} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(t+1)})$
  $\hat{\boldsymbol{x}} = \tilde{\boldsymbol{x}}^{(T+1)}$ ▷ for repair
  $\hat{\boldsymbol{z}} = \boldsymbol{z}^{(T+1)}$
  $\hat{\pi}_d = g\left(r(\boldsymbol{x}_d, \hat{\boldsymbol{z}}) + \log \frac{\alpha}{1-\alpha}\right)$ ▷ eq. (3.10), outlier detection
  return $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}, \hat{\pi})$

Algorithm 3 TwoStage: pseudo-Gibbs sampling

procedure TWOSTAGE(T, fixed $\{\phi, \theta\}$)
  $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}, \hat{\pi}) \leftarrow$ OneStage(T,$\{\phi, \theta\}$)
  $\hat{w}_d \sim q_{\hat{\pi}_d}(w_d)$
  $x_d^{(1)} = \hat{w}_d \times x_d + (1 - \hat{w}_d) \times \bar{x}_d$
  for $1, ..., T$ **do**
    $\boldsymbol{z}^{(t+1)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(t)})$
    $\tilde{\boldsymbol{x}}^{(t+1)} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(t+1)})$
  $\hat{x}_d = \hat{w}_d \times x_d + (1 - \hat{w}_d) \times \tilde{x}_d^{(T+1)}$
  $\hat{\boldsymbol{z}} = \boldsymbol{z}^{(T+1)}$
  return $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}, \hat{\pi})$

3.9 Concluding Remarks

In this chapter RVAE has been presented, a deep unsupervised model for cell outlier detection and repair in mixed-type tabular data. RVAE allows robust identification of outliers during training, reducing their contribution to parameter learning. Furthermore, a novel row outlier score for mixed-type features was introduced. RVAE outperforms or matches competing models for outlier detection and dirty cell repair, even though they heavily rely on fine-tuning of hyperparameters by using a labelled validation set.

3.9.1 Advantages and Disadvantages

In summary, a list of ***advantages*** related to this chapter and the proposed model (RVAE) is given below:

- ***Simple and tractable.*** RVAE uses gradient-based training in mini-batches, which is tractable for large datasets. The model can be easily implemented in any differentiable programming framework used in deep learning, only decoder structure is different compared to standard VAEs. In fact, different neural network architectures (e.g. convolutional NNs, self-attention, capsules) can be combined with RVAE. It can be used off-the-shelf in initial data exploration without substantial user intervention.
- ***Robust to hyperparameter fine-tuning, for moderate corruption.*** If the model is faced with corruption up to a moderate amount then hyperparameter tunning on α is not needed, unlike competitor methods. Though in higher levels of corruption, or when systematic errors are present, the hyperparameter α can be tuned.
- ***Interpretable outlier detection*** The RVAE allows the user to use the outlier scores for each cell to perform detection, but also to interpret why that instance is an outlier. This is particularly useful in higher dimension dataset. Few models provide a way to highlight the cells that are to blame for outlierness.
- ***End-to-end data cleaning*** RVAE provides a practical solution to the problem of cleaning a dataset that has been corrupted. It provides both outlier detection and data repair in one model, and thus the user does not have to resort or train additional methods. It is an end-to-end solution to data cleaning.
- ***Representation learning for clean data*** RVAE like any other VAE during training learns a smaller latent representation that encodes the characteristics of the entire dataset. Looking at the RVAE formulation (see Section 3.5) only the clean component is modelled by an autoencoder neural network. Indeed, the AE model parameters θ (decoder) and ϕ (encoder) are only part of the clean component; being that the outlier component is a static distribution. If RVAE is successful then the AE neural network learns to map an outlier instance to a repaired (or underlying inlier) instance at

reconstruction – i.e. the repair process in eq. 3.14. Moreover, as discussed in Section 3.5 (near equation 3.7) the gradients of an outlier will be down-weighted in the dirty cells in order to minimize its influence during AE training. In fact, as stated before, due to the gradient backpropagation rule if the gradients of θ (decoder) are down-weighted, so will be the gradients of ϕ (encoder) with respect to the ELBO loss. Even though the encoder sees both inlier and outlier instances, its parameters are mostly affected by clean cells if RVAE is successfully trained. Therefore, much like a Denoising AE (DAE) (Vincent et al., 2010) the RVAE model will map outlier instances to repaired (inlier) instances. However, a DAE will need paired inlier and outlier samples for training, whilst RVAE does not. Typically, a DAE will map both outlier and underlying inlier instances to the same neighborhood in the latent space (Shen et al., 2019). Thus learning an idealized latent representation of the underlying clean data. If a trained RVAE can repair the dataset properly, then it is possible that it may learn this idealized latent representation as well. This latent representation could then be used in downstream tasks for the dataset in question, which we leave for future work.

Some relevant ***disadvantages*** are listed below:

- ***Threshold setting for row and cell scores.*** In this chapter, by using a metric like AVPR for outlier detection evaluation, it was proven that RVAE anomaly scores have great performance. However, like the vast majority of outlier detection models thresholds have to be set in order to detect corrupt cells or instances in practice. This can be accomplished by using a small labelled validation set, or much more commonly by the user directly setting it.

- ***Traditional outlier detection only.*** If the task is just traditional outlier detection, and a labelled validation set is given, then other methods might be more relevant than RVAE as a first try. Other methods may have faster training times, and are easier to deploy (e.g. standard software packages). Though, one of the advantages of RVAE is that validation sets are not needed for most scenarios. Most outlier detection methods lack cell level interpretability like RVAE, but that may not always be necessary if the user is subject-matter expert and data is not high-dimensional. Moreover, it is

possible other recent SOTA methods for traditional outlier detection may have better performance.

- ***Complexity of deep architectures.*** Deep learning models unlike classic outlier detection methods need to search for the most appropriate neural architecture for the task. In this case, deep learning may add another layer of complexity, which the user may need to consider. This also applies to RVAE, and other deep VAE models.

- ***Corruption by systematic errors.*** The RVAE model was mainly designed to handle random errors, and may handle a low amount of systematic errors. In fact, the noise process given by a mixture of two Gaussian components (see Section 3.8.4) does introduce a systematic error corruption, and RVAE still has good performance. However, early exploration into the issue indicated that systematic errors in higher amounts lead to overfitting to corruption much more easily. Tackling high corruption by systematic errors is virtually impossible without supervision, or some other prior knowledge embedded into the model. The only option here is to increase the strength of data reweighting, or regularization, hence diminishing the contribution of potential outliers to the modelling of inliers. In RVAE this corresponds to decreasing the value of hyperparameter α, meaning that we now assume the dataset is more corrupt. This will increase outlier detection performance to a point, but at substantial cost for data repair performance with loss of detail for repairs. Indeed, the model starts to collapse to mean behavior, e.g. blurred image reconstructions.

- ***Outlier component model for other types of categorical features.*** In this work the outlier component p_0 of the RVAE for categorical features was defined as a uniform distribution. Although this seems to work quite well in our experiments, for ordinal features this is probably not optimal. Note that ordinal features are a sub-type of categorical features, where a clear ordering of the categories exists. Count data features is a good example, where a Poisson likelihood could have been used. Other likelihoods could also be specified for the case where a hierarchical dependency exists between the categories of the feature.

- ***Exploring RVAE in Image Data.*** One thing unexplored in this work

was image data, which is left as future work. It is possible the outlier component p_0 may need some slight change or improvement, but overall the p_0 in eq. (3.9) should be a good starting point.

3.9.2 Comparing to a Recent Model: Picket

After RVAE was released to the public, the recent work in Liu et al. (2020) (Picket) has used our model as a baseline in random, systematic, and adversarial error settings. The task was that of traditional outlier detection, i.e. detecting corrupt rows.

The RVAE did not do well with adversarial errors generally. It did decently well as far as random errors are concerned. For moderate to high corruption with systematic errors, the RVAE clearly registered worse performance than the proposed model therein. However, for small to moderate corruption with systematic errors RVAE actually did surprisingly good (seen in the appendix). However, note that in Liu et al. (2020) the authors did not actually tune α, and just used the proposed value of 0.95 from this chapter. This was meant for random errors and up to moderate corruption levels. As hinted above when dealing with systematic errors, and very high corruption, there is a benefit to decreasing the value α. This probably would have improved RVAE performance for systematic error detection.

Interestingly in Picket (Liu et al., 2020) some real-world examples of corruption in data are used in the experiments where RVAE is a baseline. In one case datasets (i.e. *Titanic*, *Restaurant*) are injected with real-world common examples of error corruption. On the other hand, *Food* is a real-world example of a dataset that already presents error corruption, and where the ground-truth (outliers) is known. Hence, no corruption injection is needed. In Figure 8 of Liu et al. (2020) the performance of all models is measured on the task of traditional outlier detection. In this particular case, RVAE performs well in *Restaurant* dataset and close to the Picket model, which is better. In the *Titanic* dataset RVAE performance is erratic, presenting large error bounds. Lastly, all models have mediocre performance in the *Food* dataset. In Table 22 of Liu et al. (2020) the performance of all models including RVAE is measured on the ability to filter out outliers, with the purpose of improving the training of a downstream task. It is shown in the experiments that removing the outliers from the training dataset has indeed a positive impact

on the downstream task accuracy. In Table 22 the RVAE model registers good overall performance and close to Picket. In fact, in the *Titanic* dataset it is the best performing model.

Chapter 4

Repairing Systematic Outliers via Clean Subspace VAEs

4.1 Motivation: How does it fit into the thesis?

The main motivation behind this model proposal was dealing with systematic errors in data cleaning. These are corruption-based outliers, and exist within the scope of point outliers (see Section 2.2.1). These errors are notoriously difficult to deal with because they can easily be learnt as clean data. Unlike random errors they present a clear repeatable structure throughout the dataset, since they result from nearly deterministic corruption transformations (plus potentially some noise). Consequently, models with enough capacity easily overfit to these errors, making both outlier detection and data repair difficult. Having said that, our goal is to define a generative model that performs outlier detection and data repair with very little user intervention (automating as much as possible).

Since the model needs to provide high quality data repair, the choice was to develop a novel generative model (a VAE) for the purpose. Since it is difficult to isolate these types of outliers using unsupervised models, we propose some user interaction in the form of semi-supervision. Given we want to limit this we focus on the user only labelling a few examples, particularly those types of outliers (systematic errors) that needs to be repaired. We feel this is far more practical than using rule-based or pattern-based approaches, since these often require expert knowledge or some external source.

Seeing as a systematic outlier is a combination of patterns of a clean instance and systematic error patterns, our main insight is that inliers can be modelled by a smaller representation (subspace) in a model than outliers. By exploiting this, we propose the Clean Subspace Variational Autoencoder (CLSVAE), a novel semi-supervised model for detection and automated repair of systematic errors. The main idea is to partition the latent space and model inlier and outlier patterns separately.

CLSVAE is effective with much less labelled data compared to previous related models, often with less than 2% of the data. We provide experiments using three image datasets in scenarios with different levels of corruption and labelled set sizes, comparing to relevant baselines. CLSVAE provides superior repairs without human intervention, e.g. with just 0.25% of labelled data we see a relative error decrease of 58% compared to the closest baseline.

4.2 Introduction

Often practitioners have to deal with dirty datasets before they can start applying machine learning (ML) models. We focus on datasets where some instances have been corrupted by noise, producing outliers. The corresponding clean instances prior to corruption are called inliers, as well as any other instances that have not been corrupted. Because the presence of outliers can degrade the performance of ML methods (Krishnan et al., 2016; Liu et al., 2020), a standard option is to resort to a data cleaning pipeline before applying any model. This pipeline includes two key tasks: i) *outlier detection* (Ruff et al., 2021), detecting all outliers; ii) *repair* those outlier instances (Neutatz et al., 2021; Wan et al., 2020), recovering the underlying inlier instance. Ultimately, the goal is to propose a method that performs these steps automatically, i.e. *automatic detection and repair*.

Generally, we can consider two types of errors present in an outlier: *random* or *systematic* (Taylor, 1997; Liu et al., 2020). Both in industry and in practice these type of outliers are most commonly within the scope of point outliers (see Section 2.2.1). This means we can evaluate whether an instance is an outlier on its own, without context from other instances (e.g. space proximity, time-series, connections in a graph). Random errors corrupt each instance independently, and feature value changes are sampled from an unknown distribution. This noising

cannot be replicated in a repeatable manner. For continuous features, a common example of this type of error are those well-modelled by additive noise with zero-mean. Systematic errors result from deterministic, or nearly deterministic (potentially with some noise), transformations that occur repeatedly in the data. Examples of systematic errors include watermarks or deterministic pixel corruption (e.g. artifacts) in images; additive offsets or replacement by default values (e.g. NaN) in sensor data; deterministic change of categories (mislabelling) or of name formats in categorical features in tabular data. Usually, the same features are affected, but not always. In most cases, this noising can be replicated.

These two types of errors have a different impact when it comes to models performing detection or repair. Random errors do not show a distinct pattern across outliers and thus are not predictable. As a result, unsupervised models using regularization or data reweighting (Zhou & Paffenroth, 2017; Akrami et al., 2019a; Eduardo et al., 2020) can avoid overfitting to random errors. On the contrary, a systematic error shows a pattern across outliers, as a result of the (nearly) deterministic transformation, making them predictable (Liu et al., 2020). This property makes higher capacity models (e.g. deep learning) prone to overfitting to these errors, even in the presence of regularization. As a result, models for outlier detection and repair in the presence of systematic errors more easily conflate outliers with inliers. In this work we study how to develop a model for data cleaning robust to the effect of systematic errors.

One solution is to provide some supervision, so the model can distinguish between inliers and outliers with systematic errors. This supervision can be provided in different forms, such as logic rules (Rekatsinas et al., 2017) or programs (Lew et al., 2021) describing the underlying clean data. However, these may require expert knowledge or substantial effort to formalize. Hence, it is easier and less time consuming for the practitioner to simply provide a *trusted set* as form of supervision. A trusted set is a small labelled subset of the data, which can be used to train a method for detection and repair. The user labels the instances either inlier or outlier, providing a few examples (e.g. 10 instances) per type of systematic error to repair. Overall, this might correspond to less than 2% of the entire dataset (*sparse semi-supervision*). Equally important, labelling does not require the user to manually repair the instances in the trusted set.

Deep generative models (DGM) have high capacity (flexible) and thus can easily

overfit to systematic errors. This motivates us to propose a method for detection and repair of such errors based on a semi-supervised DGM, which to the best of our knowledge has not been explored. We postulate that inlier data needs a *smaller representation* relative to outlier data when being represented by DGMs, since an outlier has more information to model (i.e. underlying inlier and systematic error). In addition, we claim that outliers are a *combination* of a *representation describing the inlier portion* of an instance, and a *representation describing the type of systematic error*.

Given these insights, we propose Clean Subspace Variational Autoencoder (CLSVAE), a novel semi-supervised model for detection and automated repair of systematic errors. This model deviates from a standard VAE (Kingma & Welling, 2014), in two ways. First, the latent representation is partitioned into two subspaces: one that describes the data if it were an inlier (clean subspace), and the other that describes what systematic error (if any) has been applied (dirty subspace). The model is encouraged to learn a disentangled representation using a simple yet effective approach: for outliers the decoder will use a clean subspace concatenated with the dirty subspace; in turn, for inliers the decoder will reuse the same clean subspace but concatenated with random noise. Then, at repair time the decoder will only need the clean subspace to reconstruct the underlying inlier. Secondly, we introduce semi-supervision through a trusted set, in so helping the model distinguish between inliers and outliers with systematic errors. Additionally, to encourage the clean subspace to represent inlier data, and the dirty subspace to represent systematic errors, we introduce a novel penalty term minimizing their mutual information (MI). This penalty is based on the *distance correlation* (DC) (Székely et al., 2007), and it improves model performance (stability) and repair quality. Compared to baselines we provide superior repairs, particularly, we show significant advantage in smaller trusted sets or when more of the dataset is corrupted.

4.3 Related Work

For random errors, several works have explored detection (Akrami et al., 2019a; Ruff et al., 2019; Liu et al., 2020; Lai et al., 2019). Some have proposed methods for detection and automated repair of random errors (Eduardo et al., 2020; Zhou

& Paffenroth, 2017; Krishnan et al., 2016). A few works have explored outlier detection for systematic outliers, both semi-supervised (Ruff et al., 2019) and unsupervised (Liu et al., 2020; Lai et al., 2020), but these works do not consider automated repair. In all these works, the main focus has been *point outliers* (Section 2.2.1) as this is the most common type in practice.

For tabular data, methods have been proposed that can do repair after detection has been performed; for an overview, see (Ilyas & Chu, 2019; Chu et al., 2016). These use probabilistic models melded with logic rules, i.e. probabilistic relational models (Rekatsinas et al., 2017), or user-written programmatic descriptions of the data, e.g. in a probabilistic programming language (Lew et al., 2021). They have the potential to capture systematic errors. However, these require the user to provide rules or programs that characterize clean data, which requires manual effort and expertise. In contrast, labelling a few inlier and outlier instances to build a trusted set may be more user-friendly.

The idea of using unsupervised models for outlier detection is not new (Schölkopf et al., 1999; Liu et al., 2008). Unsupervised removal from a dataset of a small fraction of instances suspected of being outliers has been explored in Koh et al. (2018); Diakonikolas et al. (2018); Liu et al. (2020), when used against adversarial errors it is called *data sanitization* (Koh et al., 2018). However, if one wants to repair existing dirty data, a common unsupervised approach for moderate corruption is to apply enough regularization or data reweighting to an autoencoder (Eduardo et al., 2020; Zhou & Paffenroth, 2017; Akrami et al., 2019a), hopefully repairing the errors at reconstruction. This has proven successful for random errors, though as we show, this is less effective for systematic errors. Often regularization is too strong leading to bad repair quality, since reconstruction is collapsing to mean behaviour – e.g. blurry image samples, missing details.

Attribute manipulation models (Klys et al., 2018; Choi et al., 2020) usually rely on extensive labelled data, usually fully supervised. Data cleaning can be seen as an attribute manipulation problem with one attribute: either the instance is an inlier or outlier. Some models like CVAE (Sohn et al., 2015) may use discrete latent variables, instead of continuous, to model attributes. These may lack the capacity to capture diversity in the same attribute (Joy et al., 2020), e.g. distinct types of systematic errors in the data, and thus offer a poor fit to the data. Later, this may result in poor repair.

Disentanglement models, unsupervised (Locatello et al., 2019b; Tonolini et al., 2019) and semi-supervised (Ilse et al., 2020; Locatello et al., 2019b; Joy et al., 2020), encourage individual (or set of) continuous latent variables to capture different attributes of the data instances. More in-depth discussion about these models can be found in Section 2.5.4. In theory, a disentanglement model could isolate the inlier / outlier attribute into a latent variable, then after training one could find a value for this variable that repairs the outlier. In practice, this is much simpler with semi-supervised models, since we know which variable (or set of) corresponds the attribute, but these tend to use more labelled data than we consider. In either case, these models usually need additional processing, often requiring human in the loop to explore the latent variable and find the best repair. Conversely, once our model is trained, further human intervention is not needed, so that repair is automated.

More specifically, there have been (semi-)supervised disentanglement models (Klys et al., 2018; Ilse et al., 2020; Joy et al., 2020; Locatello et al., 2019b) that segregate the latent space in order to model particular attributes. Once more, further discussion on these is found in Section 2.5.4, where a SOTA model of this type (CCVAE, (Joy et al., 2020)) is discussed in detail (see Section 2.5.4.1). Generally, these models use the same latent subspace to model some attribute y. In this case both inliers ($y = 1$) and the systematic error patterns of outliers ($y = 0$). The remainder of the latent space is left to model unlabelled attributes of the data. This type of modelling is usually geared toward general disentanglement of data attributes, and not data instance repair.

The proposed CLSVAE model also segregates the latent space. However, unlike the other models, it expresses an *inductive bias* related to the problem of corrupted data, which improves the disentanglement in latent space of (underlying) inlier and error patterns. The CLSVAE splits the latent space in two: one subspace models inlier patterns ($y = 1$); and the other subspace models systematic error patterns ($y = 0$). Furthermore, CLSVAE defines a specific neural network encoder for each subspace, such that inlier and error patterns do not share network parameters. We found this to lead to better repair performance. On the other hand, other disentanglement models typically share the same neural encoder.

CLSVAE exploits the fact that inliers are numerous and appear in the dataset uncorrupted; and thus these can be directly modelled by one subspace during

training. This architectural bias can improve disentanglement, since for the less common outliers an additional subspace is used for error information. Seeing this is not always stable with very small trusted sets, a penalty loss based on *distance correlation* is used to minimize mutual information between subspaces. Conversely, the other models always assume every instance needs to be disentangled (inliers from errors); and thus do not exploit the fact inliers are frequent and appear without errors. We believe this issue is further compounded in other models by the use of the same subspace for inliers and errors, and the sharing of encoder neural network. Although with a large enough trusted set some of these models can have good performance, in this chapter we focus on very small trusted sets. Therefore such inductive biases as provided by CLSVAE for this problem are quite important for performance.

4.4 Problem Definition

We assume the user knows that systematic errors have corrupted dataset $\mathcal{X}$, and thus outliers exist therein, but the inliers are still the majority. The user also has an idea of what patterns constitute systematic errors, and is able to recognize them. Intuitively, we think of inlier data being characterized by a set of patterns, which we call *clean patterns*, and those that constitute (systematic) errors, called *dirty patterns*. *An outlier is an instance that has been corrupted by systematic errors, where prior to that would be constituted by clean patterns only.* Formally, for $\tilde{\boldsymbol{x}}$ an underlying inlier instance, an outlier is defined by $\boldsymbol{x} = f_{cr}(\tilde{\boldsymbol{x}})$. We define f_{cr} as a *general transformation that corrupts the inlier instance with systematic errors*, and not necessarily invertible. Each type of systematic error usually affects specific features of $\tilde{\boldsymbol{x}}$ in the same predictable way, repeatedly in the dataset. This is unlike random errors, which both the feature and changed values are at random throughout instances. The type of outliers described here are *point outliers* (see Section 2.2.1), and that is the focus of this thesis.

The user builds a small *trusted set* by labelling a few of the inliers and outliers in the train set $\mathcal{X}$, forming the labelled subset $\mathcal{X}_l = \{\boldsymbol{x}_n\}_{n=1}^{N_l}$. So we have the dirty dataset $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_u$, where $\mathcal{X}_u = \{\boldsymbol{x}_n\}_{n=1}^{N_u}$ is the unlabelled part. The overall size of the train set is $N = N_l + N_u$. Each $\boldsymbol{x}_n \in \mathcal{X}_l$ is associated with a label $y_n \in \{0, 1\}$, which indicates whether x_n is an inlier ($y_n = 1$) or an outlier

($y_n = 0$). We write $\mathcal{Y}_l = \{y_n\}_{n=1}^{N_l}$ and thus the trusted set is formally defined by $(\mathcal{X}_l, \mathcal{Y}_l)$. The trusted set should be representative of the inliers and outliers in the data. Note that there is a set of different corrupting transformations, i.e. systematic error types, each applied to several instances. Hence, the trusted set should provide at least a few labelled samples per type of systematic error. This is important so as to help the model distinguish between inliers and outliers. In our problem, the labelled portion of the dataset $\mathcal{X}_l$ (trusted set) is significantly smaller than the unlabelled portion $\mathcal{X}_u$, e.g. 0.5% of N_u. Given how small the trusted set is, we refer to this as *sparse semi-supervision*.

In this work, the main task is to first perform *outlier detection* on $\mathcal{X}$, in order to discover y for each instance $\boldsymbol{x}$. The second task is *repair* those instances that are considered to be outliers ($y = 0$). Specifically, the model needs to provide a repair transformation g_r such that $\hat{\boldsymbol{x}} = g_r(\boldsymbol{x})$ where $\hat{\boldsymbol{x}} \approx \tilde{\boldsymbol{x}}$ (i.e. repair is close enough to the underlying inlier instance). Thus, we want a model that performs these two steps automatically, i.e. without a human in the loop.

4.5 Proposal: Clean Subspace VAE (CLSVAE)

In this section, we introduce the generative and variational models for our proposal. Our Variational Autoencoder (VAE) model is motivated by three observations:

- ***Outliers are systematic.*** Outlier instances can be described by a number of predictable recurring patterns, some of which are considered dirty (e.g. black patch on an image). Our assumption is that the latter patterns can be well represented by a DGM.
- ***Subspace hypothesis.*** Outliers are composed of the underlying inlier and the transformation (error) that made it an outlier. Since we are trying to capture more information in our representation for outliers, a larger latent space in a DGM is needed. Conversely, the inlier data can thus be modelled using a subspace of the overall latent space.
- ***Outliers are a combination of clean and dirty patterns.*** Because clean and dirty patterns produce different visible effects in the instance, the effects of clean and dirty patterns can be modelled separately. Hence, this DGM only requires the parameters (or variables) of the clean patterns to

generate a repair. Note that this assumption is valid for corruption-based outliers, in particular for systematic errors.

We call our model the *sparse semi-supervised Clean Subspace VAE (CLSVAE)*. This reflects the fact that our model has a partitioned latent space: a subspace for clean patterns, and a subspace for dirty patterns. Thus at test time only the subspace for clean patterns is used to generate a repair. A standard VAE (Kingma & Welling, 2014), with weight decay, is provided in section 2.5.2.1 for readers unfamiliar with this type of model.

Additional Details

We note to our reader that the above assumption on outliers being a *combination of clean and dirty patterns* only applies to *corruption-based outliers* (see Section 2.2.1, some examples are presented). Specifically, we consider systematic error corruption where the anomalous pattern can be learnt or predicted (Liu et al., 2020; Krishnan et al., 2016; Lew et al., 2021). This means that in a DGM potentially the errors can be modelled separately from the patterns that constitute the underlying inlier of that instance. However, there are many outliers that present a structured or systematic behavior, and are not a combination of clean and dirty patterns. For instance, all outliers that are structured but are not originated by error corruption affecting part of the data features. Some examples of these outliers can be found in out-of-distribution outliers Yang et al. (2021), and in anomalous data classes that sometimes populate datasets (Ruff et al., 2019; Diakonikolas et al., 2018). Further discussion on outlier types can be found in Section 2.2.1.

4.5.1 Generative Model

The main idea is that inlier samples ($y = 1$) will have a smaller representation relative to outliers ($y = 0$), since outliers need an extra representation to model errors. This follows from the observations made earlier in Section 4.5. In our model, $\boldsymbol{z}_c$ will represent inliers, $\boldsymbol{z}_d$ will represent the error pattern for outliers, and $\boldsymbol{z}_\epsilon$ is random noise. The overall latent code is $\boldsymbol{z} = [\boldsymbol{z}_c; y\ \boldsymbol{z}_\epsilon + (1-y)\ \boldsymbol{z}_d]$, where $[\ ;\ ]$ defines vector concatenation, leading to a latent space that is partitioned into two subspaces. If generating an inlier ($y = 1$) then $\boldsymbol{z}_c \in \mathbb{R}^q$ is by itself responsible for modelling all the clean patterns present in an instance, encouraging a lower dimensional manifold for inlier data. This is due to $\boldsymbol{z}_\epsilon$ being a random noise vector

of the same dimension as $\boldsymbol{z}_d \in \mathbb{R}^p$, devoid of additional information. If generating an outlier ($y = 0$) both $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ are used, where $\boldsymbol{z}_d$ models the different types of dirty patterns that can be detected in an instance. In fact, using $\boldsymbol{z}_\epsilon$ during inlier generation encourages $\boldsymbol{z}_d$ to model dirty patterns only, encouraging a higher dimensional manifold for outlier data.

Our generative model is therefore defined by the joint distribution between y and latent subspaces $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$. The main idea expressed above can be written as a two component mixture model for the decoder, see eq. (4.4), where y is the gating variable. Hence we have

$$p_\theta(\boldsymbol{x}, \boldsymbol{z}_c, \boldsymbol{z}_d, \boldsymbol{z}_\epsilon, y) = p_\theta(\boldsymbol{x}|\boldsymbol{z}_c, \boldsymbol{z}_d, \boldsymbol{z}_\epsilon, y) p_{\sigma_c}(\boldsymbol{z}_c) p_{\sigma_d}(\boldsymbol{z}_d) p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon) p_\alpha(y), \tag{4.1}$$

where

$$p_\alpha(y) = \text{Bernoulli}(y|\alpha), \tag{4.2}$$

$$p_{\sigma_c}(\boldsymbol{z}_c) = \mathcal{N}(\boldsymbol{z}_c|\boldsymbol{0}, \sigma_c^2 \boldsymbol{I}) \;\; p_{\sigma_d}(\boldsymbol{z}_d) = \mathcal{N}(\boldsymbol{z}_d|\boldsymbol{0}, \sigma_d^2 \boldsymbol{I}) \;\; p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon) = \mathcal{N}(\boldsymbol{z}_\epsilon|\boldsymbol{0}, \sigma_\epsilon^2 \boldsymbol{I}), \tag{4.3}$$

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}_c, \boldsymbol{z}_d, \boldsymbol{z}_\epsilon, y) = p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\; \boldsymbol{z}_\epsilon])^y p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\; \boldsymbol{z}_d])^{(1-y)}, \tag{4.4}$$

where the density $p_\theta(\boldsymbol{x}\,|\,[\cdot;\;\cdot])$ is parameterized by a neural network. We assume inlier data has a smaller variance as whole than outlier data, so we use a σ_c that is *smaller* than σ_d, e.g. smaller by an order of magnitude. Further, $\boldsymbol{z}_\epsilon$ is just low level Gaussian random noise with an order of magnitude similar to σ_c. Hence, after training, the region around $\boldsymbol{0}$ (zero mean) for this latent subspace encourages $p_\theta(\boldsymbol{x}\,|\,[\cdot;\;\cdot])$ to only use $\boldsymbol{z}_c$ for reconstruction, obtaining a repaired instance. The parameter α reflects the prior belief on the fraction of clean data. Smaller values for α means more data points are rejected when modelling $\boldsymbol{z}_c$, which offers more robustness. So we have σ_ϵ, σ_c, σ_d and α as hyper-parameters.

Additional Details

One final note should be added about the definition of z_ϵ. We tried setting $\boldsymbol{z}_\epsilon = \boldsymbol{0}$ at training, however this did not work as well as setting $\boldsymbol{z}_\epsilon$ to random noise. One hypothesis is that using random noise centered at $\boldsymbol{0}$ (zero mean) allows to create a margin (separation) between the neighborhood around $\boldsymbol{z}_d = 0$, and the $\boldsymbol{z}_d$ space that encodes different types of noise present in an instance. As a result,

this lead to a better distinction by the decoder (generator) between inlier and outlier representations in $\boldsymbol{z}_d = 0$. This in turn allows for better automatic repair performance by the model.

Lastly, further discussion on why changing prior variance values (i.e. σ_c and σ_d) has an impact on the fully trained model is presented in section 4.6.3.1. We theorize that stochastic gradient optimization makes degenerate solutions unlikely, i.e. where σ_c and σ_d value changes might be *scaled out* (made irrelevant) by the encoder or decoder network weights. Secondly, in section 4.6.5.3, we provide empirical evidence using a standard VAE that corrupted data has larger variance than clean data. This means that there is more information to model with corrupted data (with outliers) than clean data (inliers only). Indeed, we see that clean data can be better modelled by a standard VAE using a smaller latent space than corrupted data.

4.5.2 Variational Model

We consider separate encoders for $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$, and make y depend on $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$. The idea is that parameters of each encoder focus on different aspects, i.e. clean or dirty patterns respectively. The model factorizes as

$$q(\boldsymbol{z}_c, \boldsymbol{z}_d, \boldsymbol{z}_\epsilon, y|\boldsymbol{x}) = q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d) q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x}) q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x}) q_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon), \tag{4.5}$$

and

$$q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_c|\boldsymbol{\mu}_{\phi_c}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\phi_c}(\boldsymbol{x})), \quad q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_d|\boldsymbol{\mu}_{\phi_d}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\phi_d}(\boldsymbol{x})), \tag{4.6}$$

$$q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d) = \text{Bernoulli}(y|\pi_{\phi_y}([\boldsymbol{z}_c; \boldsymbol{z}_d]), \quad q_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon) = p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon), \tag{4.7}$$

where for distributions in eq. (4.6): $\{\boldsymbol{\mu}_{\phi_c}(.), \boldsymbol{\sigma}_{\phi_c}(.)\}$ is a neural network with parameters ϕ_c; similarly for $\{\boldsymbol{\mu}_{\phi_d}(.), \boldsymbol{\sigma}_{\phi_d}(.)\}$ with ϕ_d. We have $\boldsymbol{\sigma}_{\phi_c}(\boldsymbol{x})$ and $\boldsymbol{\sigma}_{\phi_d}(\boldsymbol{x})$ being diagonal covariance matrices. The distribution for random noise $\boldsymbol{z}_\epsilon$ in eq. (4.7) is the same as in the generative model, see eq. (4.3). Further, in eq. (4.7), the $\pi_{\phi_y}(.)$ parametrizes the Bernoulli distribution $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d)$, and is a neural network with parameters ϕ_y.

Additional Details

We found that using $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d)$ – rather than $q_{\phi_y}(y|\boldsymbol{z}_d)$ or $q_{\phi_y}(y|\boldsymbol{x})$ – yielded better performance in more challenging scenarios, e.g. smaller trusted set or when

higher data corruption is present. One could have used $q_{\phi_y}(y|\boldsymbol{z}_d)$, which works, but it is still worse than our proposal. This is because $\boldsymbol{z}_c$ provides important context on the clean patterns present in the instance, which in turn allows for $\boldsymbol{z}_d$ to contain less information about clean patterns. Hence, $\boldsymbol{z}_d$ is able to focus better on modelling the dirty patterns, and so the *mutual information* (MI) between $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ will be lower. Moreover, $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d)$ allows for the cross-entropy loss used for the trusted set to directly bias the latent space. Unlike when using $q_{\phi_y}(y|\boldsymbol{x})$ directly as a neural network. Other works have tried biasing the latent space in the same fashion as this work, e.g. (Locatello et al., 2019b; Ilse et al., 2020; Joy et al., 2020).

In practice, in order to stabilize the optimization procedure in some cases, we used $\pi_{\phi_y}([\text{sg}(\boldsymbol{z}_c); \boldsymbol{z}_d])$ in eq. (4.7), i.e. $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d) = \text{Bernoulli}(y|\pi_{\phi_y}([\text{sg}(\boldsymbol{z}_c); \boldsymbol{z}_d]))$. Note that $\text{sg}(\boldsymbol{z}_c)$ stands for *stop gradient* operator applied to $\boldsymbol{z}_c$; and this prevents $\boldsymbol{z}_c$ from being updated with dirty pattern information early on in the training. By using $\text{sg}(\boldsymbol{z}_c)$, when executing back-propagation, we prevent the gradients from $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d)$ to influence the learning of parameters ϕ_c. As such, ϕ_c will only be affect by gradients stemming from the reconstruction of inlier instances, and not the decision on y. In the implementation code of CLSVAE *we always used the stop-gradient operator on* $\boldsymbol{z}_c$ as it proved to be able to stabilize the convergence of the training procedure in harsh conditions. These harsh conditions are usually a combination of *i) small trusted sets* and *ii) high corruption*. In all other cases this was not seen as too necessary, as the benefits to training convergence were marginal. Hence, it was decided to always use the stop-gradient operator on $\boldsymbol{z}_c$ for all experiments of CLSVAE presented in this chapter.

4.5.3 Training Loss

Our model is trained to maximize an objective function with three terms, which accounts for our semi-supervised setting. The first term $\mathcal{L}(\boldsymbol{x})$ is the evidence lower bound (ELBO) for the unlabelled part of the data. The second term $\mathcal{L}(\boldsymbol{x}, y)$ is the ELBO for the trusted set. The third term is based on the negative $\mathcal{C}_{\text{WCE}}(\boldsymbol{x}, y)$, where $\mathcal{C}_{\text{WCE}}(\boldsymbol{x}, y)$ is the weighted cross-entropy loss which ensures that $q_{\phi_y}(y|\boldsymbol{z}_c, \boldsymbol{z}_d)$ correctly predicts the trusted set labels y.

The ELBO for the unlabelled (unsupervised) part is

$$\mathcal{L}(\boldsymbol{x}) = \mathbb{E}_{q_{\phi_y}(y|\boldsymbol{z}_c,\boldsymbol{z}_d)q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon)}\left[\log \frac{p_\theta(\boldsymbol{x}|\boldsymbol{z}_c,\boldsymbol{z}_d,\boldsymbol{z}_\epsilon,y)p_{\sigma_c}(\boldsymbol{z}_c)p_{\sigma_d}(\boldsymbol{z}_d)p_\alpha(y)}{q_{\phi_y}(y|\boldsymbol{z}_c,\boldsymbol{z}_d)q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})}\right],$$

which can be rewritten as

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}) =& \mathbb{E}_{q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon)}\Big[\pi_{\phi_y}([\boldsymbol{z}_c;\boldsymbol{z}_d])\log p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\boldsymbol{z}_\epsilon]) \\ &+ (1-\pi_{\phi_y}([\boldsymbol{z}_c;\boldsymbol{z}_d]))\log p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\boldsymbol{z}_d]) - D_{KL}\left(q_{\phi_y}(y|\boldsymbol{z}_c,\boldsymbol{z}_d)||p_\alpha(y)\right)\Big] \\ &- D_{KL}\left(q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})||p_{\sigma_c}(\boldsymbol{z}_c)\right) - D_{KL}\left(q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})||p_{\sigma_d}(\boldsymbol{z}_d)\right),\end{aligned} \tag{4.8}$$

where $q_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon) = p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon)$ and so they cancel each other. The expectations are obtained via Monte-Carlo (MC) estimation via reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014).

For the trusted set (supervised) part the ELBO is

$$\begin{aligned}\mathcal{L}(\boldsymbol{x},y) =& \mathbb{E}_{q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})p_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon)}\Big[y\log p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\boldsymbol{z}_\epsilon]) + (1-y)\log p_\theta(\boldsymbol{x}|\,[\boldsymbol{z}_c;\boldsymbol{z}_d])\Big] \\ &+ \log p_\alpha(y) - D_{KL}\left(q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})||p_{\sigma_c}(\boldsymbol{z}_c)\right) - D_{KL}\left(q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})||p_{\sigma_d}(\boldsymbol{z}_d)\right).\end{aligned} \tag{4.9}$$

Lastly, we need to define the *weighted cross-entropy* $\mathcal{C}_{\text{WCE}}(\boldsymbol{x},y)$, which is

$$\mathcal{C}_{\text{WCE}}(\boldsymbol{x},y) = -y.\log q(y=1|\boldsymbol{x}) - \omega_{imb}.(1-y).\log\left(1-q(y=1|\boldsymbol{x})\right) \tag{4.10}$$

where

$$\omega_{imb} = \max\left\{1, \frac{N_{l_1}}{N_{l_0}}\right\}, \quad N_{l_1} = \sum_{i=1}^{N_l} y_i, \quad N_{l_0} = N_l - N_{l_1}, \tag{4.11}$$

and ω_{imb} compensates for trusted set class imbalance.[1] However, we do not have access to $q(y=1|\boldsymbol{x})$ and thus we cannot estimate $\mathcal{C}_{\text{WCE}}(\boldsymbol{x},y)$ directly. Still, we can minimize an upper-bound

$$\begin{aligned}\mathcal{C}_{\text{WCE}}(\boldsymbol{x},y) \leq \tilde{\mathcal{C}}_{\text{WCE}}(\boldsymbol{x},y) =& \mathbb{E}_{q_{\phi_c}(\boldsymbol{z}_c|\boldsymbol{x})q_{\phi_d}(\boldsymbol{z}_d|\boldsymbol{x})}\Big[-y\log q(y=1|\boldsymbol{z}_c,\boldsymbol{z}_d) \\ &- \omega_{imb}.(1-y).\log\left(1-q(y=1|\boldsymbol{z}_c,\boldsymbol{z}_d)\right)\Big],\end{aligned} \tag{4.12}$$

which is obtained by applying Jensen's inequality.

Combining the three terms defined above, we ***minimize*** the overall loss

$$\mathcal{I} = -\frac{1}{N}\left[\sum_{\boldsymbol{x}\in\mathcal{X}_u}\mathcal{L}(\boldsymbol{x}) \;+ \sum_{(\boldsymbol{x},y)\in\mathcal{X}_l\times\mathcal{Y}_l}\mathcal{L}(\boldsymbol{x},y)\right] + \beta\frac{1}{N_l}\sum_{(\boldsymbol{x},y)\in\mathcal{X}_l\times\mathcal{Y}_l}\tilde{\mathcal{C}}_{\text{WCE}}(\boldsymbol{x},y) \tag{4.13}$$

[1] Useful when the number of labelled outliers outnumbers the inliers. Such a case does not reflect the common dataset composition, i.e. the number of inliers is larger than outliers.

with respect to the generative and variational parameters. We note that the above loss minimization will maximize the ELBO of the data, and minimize the weighted cross-entropy loss. The hyperparameter β value controls the amount of up-sampling and importance relative to the other terms, which tends to be moderately high due to how small the trusted set is.

4.5.4 Distance Correlation Penalty

In this chapter, we assume that inlier and (systematic) error patterns have generative processes that are statistically independent. This is true for a significant portion of systematic errors, which also fit the description in our problem setting (see Section 4.4). Some examples and related literature can be found in the Background chapter (see Section 2.2.1) and in the Introduction of this chapter (see Section 4.2). As such, we can make the assumption that the latent variables $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ should be close to independent, and their marginal distributions $q(\boldsymbol{z}_c)$ and $q(\boldsymbol{z}_d)$ should reflect that. One way to enforce independence between $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ is to have zero (or low) marginal *mutual information* (MI).

Still, in some cases systematic errors may depend on the data classes $t \in \mathcal{T}$ of $\mathcal{X}$. In this case, the inlier and error generative processes should be conditionally independent given the data label t – i.e. $q(\boldsymbol{z}_c|t)$ and $q(\boldsymbol{z}_d|t)$. Thus $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ should have zero (or low) conditional mutual information given t. We leave this type of systematic errors to be explored as future work, as they are less common than the former.

In CLSVAE, ideally $\boldsymbol{z}_c$ captures clean patterns only, whilst $\boldsymbol{z}_d$ captures dirty patterns. However, in more challenging scenarios, e.g. small trusted set or higher dataset corruption, obtaining this solution may not be guaranteed. Enforcing a constraint encouraging low marginal MI between $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ will lead to better model performance and stability in challenging scenarios, improving repair quality.

We would like to introduce a constraint that minimizes marginal MI between $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$. However, approximating MI properly can be complex. Instead, we use *distance correlation* (DC) as a surrogate for MI (Székely et al., 2007), which is easier to compute and can measure non-linear dependencies between vector variables. Other works have used DC as a surrogate for MI, e.g. (Chen et al., 2021). Further, DC can also measure dependence between vector variables of

different dimensions, which is often the case with $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$. For the data batch $(\boldsymbol{z}_c, \boldsymbol{z}_d) \in (\boldsymbol{Z}_c, \boldsymbol{Z}_d)$ where $\boldsymbol{Z}_c \in \mathbb{R}^{N\times q}$ and $\boldsymbol{Z}_d \in \mathbb{R}^{N\times p}$, we can define the empirical estimate of DC as $dCorr_N(\boldsymbol{Z}_c, \boldsymbol{Z}_d)$. Essentially, DC is the standard correlation between the elements of the *double centered* pairwise distance matrices of each data batch $\boldsymbol{Z}_c$ and $\boldsymbol{Z}_d$. The range is $0 \leq dCorr_N(\boldsymbol{Z}_c, \boldsymbol{Z}_d) \leq 1$, where 0 means variables are independent, and 1 implies that $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ are strongly correlated. The definition of estimator $dCorr_N(\boldsymbol{Z}_c, \boldsymbol{Z}_d)$ is provided below. Then, we show how to modify the training loss in order to integrate this constraint.

Empirical Distance Correlation

For a random data batch of n samples, such as $(\boldsymbol{Z}, \boldsymbol{Z}') = \{(\boldsymbol{z}_k, \boldsymbol{z}'_k) : k = 1, ..., n\}$, one can define the empirical estimate for the *distance correlation* between multivariate random variables $\boldsymbol{z} \in \mathbb{R}^q$ and $\boldsymbol{z}' \in \mathbb{R}^p$. Note that q and p can be of different dimensions, i.e. $q \neq p$. Given this data batch, using the definition from (Székely et al., 2007), we first define A_{kl} and B_{kl} as

$$a_{kl} = ||\boldsymbol{z}_k - \boldsymbol{z}_l||_2 \ ,$$

$$\bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl} \ , \qquad \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl} \ , \qquad \bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl} \ ,$$

$$A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot} \ ,$$

and similarly, using $b_{kl} = ||\boldsymbol{z}'_k - \boldsymbol{z}'_l||_2$ we define

$$B_{kl} = b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot} \ .$$

Now we are ready to define the empirical *distance covariance* as

$$dCov_n(\boldsymbol{Z}, \boldsymbol{Z}') = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}B_{kl} \ ,$$

and the following empirical *distance variances* for each random variable as

$$dVar_n(\boldsymbol{Z}) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}^2 \ , \qquad dVar_n(\boldsymbol{Z}') = \frac{1}{n^2}\sum_{k,l=1}^{n} B_{kl}^2 \ .$$

Finally, combining the above measures we can write the *distance correlation* as

$$dCorr_n(\boldsymbol{Z}, \boldsymbol{Z}') = \frac{dCov_n(\boldsymbol{Z}, \boldsymbol{Z}')}{\sqrt{dVar_n(\boldsymbol{Z})\ dVar_n(\boldsymbol{Z}')}} \ ,$$

where $0 \leq dCorr_n(\boldsymbol{Z}, \boldsymbol{Z}') \leq 1$. Note that if $dCorr_n(\boldsymbol{Z}, \boldsymbol{Z}') = 0$ then $\boldsymbol{z}$ and $\boldsymbol{z}'$ are statistically independent random variables. Otherwise, if $dCorr_n(\boldsymbol{Z}, \boldsymbol{Z}') = 1$ then they are strongly correlated. The *distance correlation* measure can capture non-linear dependencies, whilst the more common Pearson correlation can only capture linear dependencies. Further, a value of 0 for Pearson correlation does not imply independence, unlike in *distance correlation* which it does.

Constraining the Optimization Procedure

Enforcing this constraint based on the distance correlation means adding a penalty to the training loss. Therefore, reusing the loss defined in eq. (4.13) we now have

$$\min_{\phi_c, \phi_d, \phi_y, \theta} \mathcal{I} + \lambda_t \; dCorr_N(\boldsymbol{Z}_c, \boldsymbol{Z}_d), \tag{4.14}$$

where λ_t increases every epoch from 0 until it reaches a maximum value λ_T, which is then maintained. The rate of increase of λ_t and λ_T are hyper-parameters. This strategy is a type of penalty method as used in constrained optimization.

4.5.5 Outlier Detection and Repair Process

After training, we proceed with *outlier detection* and *automated repair*, as in section 4.4. The detection task is to discover the ground-truth labels y for each $\boldsymbol{x} \in \mathcal{X}$, using inferred labels $\hat{y}$. A score $\mathcal{A}(\boldsymbol{x})$ and threshold $\gamma \geq 0$ are used to get the set of outliers $\mathcal{O} = \{\boldsymbol{x} \in \mathcal{X} |\; \mathcal{A}(\boldsymbol{x}) \geq \gamma\}$, where a higher $\mathcal{A}(\boldsymbol{x})$ means the more likely $\boldsymbol{x}$ is an outlier. The inferred label $\hat{y}$ is obtained as: *(inlier)* $\hat{y} = 1$ if $\boldsymbol{x} \notin \mathcal{O}$; *(outlier)* $\hat{y} = 0$ if $\boldsymbol{x} \in \mathcal{O}$. For our model, we use a score based on the negative log probability of inlier given the latent subspaces, which is

$$\mathcal{A}(\boldsymbol{x}) = -\log \pi_{\phi_y}([\boldsymbol{\mu}_{\phi_c}(\boldsymbol{x}); \boldsymbol{\mu}_{\phi_d}(\boldsymbol{x})]), \quad \boldsymbol{x} \in \mathcal{X}. \tag{4.15}$$

The threshold γ can be chosen as $\gamma \approx -\log(0.5)$ assuming $q_{\phi_y}(y = 1|.) = \pi_{\phi_y}(.)$ is near calibrated, or it can be user-defined. The repair task is to obtain an inferred reconstruction $\hat{\boldsymbol{x}}$ from the outlier $\boldsymbol{x} \in \mathcal{O}$ such that it is close to the inlier ground-truth $\tilde{\boldsymbol{x}}$. The repair is generated using the most likely reconstruction under our model for $y = 1$ (inliers), which means only the clean subspace $\boldsymbol{z}_c$ is used. This is the *maximum a posteriori* estimate for a VAE, where one approximates $p_\theta(\boldsymbol{z}_c|\boldsymbol{x})$ by $q_\phi(\boldsymbol{z}_c|\boldsymbol{x})$, and then uses the means of $q_\phi(\boldsymbol{z}_c|\boldsymbol{x})$, $q_\epsilon(\boldsymbol{z}_\epsilon)$ and $p_\theta(\boldsymbol{x} \,|\, [\boldsymbol{z}_c;\; \boldsymbol{z}_\epsilon])$ in the estimate. Hence, we have

$$\hat{\boldsymbol{x}} = \boldsymbol{\mu}_\theta([\boldsymbol{\mu}_{\phi_c}(\boldsymbol{x}); \mathbf{0}]), \quad \boldsymbol{x} \in \mathcal{O}. \tag{4.16}$$

Additional Details

One should note that the formulation in eq. (4.16) is similar to the *maximum a posteriori* (MAP) estimate derived for the standard VAE in Section 2.5.1.1. Moreover, this type of MAP repair (or attribute manipulation) is commonly used for conditional VAEs, as seen in Klys et al. (2018) (see Section 2.5.3.1). Such a MAP repair strategy can be applied to CLSVAE and other types VAE models (see Section 2.5). The reader should also note that the $\mathbf{0}$ in eq. (4.16) corresponds to the mean of noise distribution $q_{\sigma_\epsilon}(\boldsymbol{z}_\epsilon)$ used during training. Just like $\boldsymbol{\mu}_{\phi_c}(\boldsymbol{x})$ corresponds to $q_\phi(\boldsymbol{z}_c|\boldsymbol{x})$, and $\boldsymbol{\mu}_\theta(\,.\,)$ to the decoder $p_\theta(\boldsymbol{x}\,|\,.\,)$.

As stated in Section 4.5.1 (in Additional Details), we initially tried the dirac distribution $q(\boldsymbol{z}_\epsilon) = \delta\{\boldsymbol{z}_\epsilon = \mathbf{0}\}$, which sometimes still had performance issues at repair time. However, better repair performance was obtained when using $q(\boldsymbol{z}_\epsilon)$ defined as zero-mean Gaussian noise, and at repair time still using $\boldsymbol{\mu}_\epsilon = \mathbf{0}$ as input to the decoder. During hyper-parameter search σ_ϵ was kept as small as possible, as performance was evaluated using the trusted set (like other models). Empirically, we did not see an improvement when sampling from $q(\boldsymbol{z}_\epsilon)$ to obtain a repaired instance, for the same $\boldsymbol{\mu}_{\phi_c}(\boldsymbol{x})$. In fact, in some few cases we noticed a drop in performance. Therefore we opted to maintain the MAP solution which uses $\mathbf{0}$. Further, as stated in Section 4.5.1 (Additional Details), we believe that using $\boldsymbol{z}_\epsilon$ at train time allows to create a margin between the neighborhood around $\boldsymbol{z}_d = 0$, and the $\boldsymbol{z}_d$ space that encodes different types of errors. Indeed, the authors in Ghosh et al. (2019) state that noise injection is a standard trick in VAEs to smooth out the latent space around a particular neighborhood – e.g. around $\mathbf{0}$.

4.6 Experiments

We evaluate two tasks: *outlier detection*, and *automated repair*. Our experiments use three image datasets: *Frey-Faces*[2], *Fashion-MNIST* (Xiao et al., 2017), *Synthetic-Shapes*. *Synthetic-Shapes* is a synthetic dataset built around four different shapes (classes): a circle, a rectangle, an ellipse and a triangle. These are colored white and set in a black background. We corrupt datasets with synthetic systematic errors, since *public real-world* datasets with ground-truth repairs and respective labels are difficult to find, as seen in Eduardo et al. (2020); Krishnan

[2] `http://www.cs.nyu.edu/~roweis/data/frey_rawface.mat`

Dataset	Data Type	No. Data Classes	No. Error Classes	Error Types
Synthetic-Shapes	28×28 binary (black / white)	4	4	4 lines
Fashion-MNIST	28×28 continuous (grey-scale)	10	8	4 lines & 4 squares
Frey-Faces	28×20 continuous (grey-scale)	1	4	4 squares

Table 4.1: Description of dataset and its corruption.

et al. (2016); Liu et al. (2020). We compare our model (CLSVAE) with baselines ranging from completely supervised to unsupervised.

4.6.1 Evaluation

For outlier detection we use *AVPR (Average Precision)* (Everingham et al., 2015) to measure detection quality, which is a surrogate for the area under the precision-recall curve (Hendrycks & Gimpel, 2016). This metric is preferred since it is insensitive to label imbalance, typical in outlier detection. AVPR score is between $[0, 1]$ and higher means better. For *automated repair* we want to quantify the quality of the repair, for outlier instances. We report the standardized mean squared error (SMSE) between pixels of the ground-truth (inlier) instance and that of the proposed repair. We report SMSE separately for the dirty pixels (those affected by the systematic error) and for the clean pixels (those unaffected). The first measures repair performance, while the latter measures *distortion* that the repair process causes to clean pixels. In both cases, a lower SMSE means better. Note in the case of binary pixels, the SMSE is just the Brier score, and thus is in $[0, 1]$.

4.6.2 Datasets and Corruption Process

In our experiments we take an uncorrupted dataset and inject it with systematic errors. These systematic errors are synthetic, designed to seem like reasonable image corruptions, e.g. occlusion or failing of a camera sensor. The types of systematic errors used across datasets are either *lines* or *squares*. Lines (two diagonal, one vertical, and one horizontal) cross the image from side to side, and may have their color set at random (black / white). These lines always affect the same pixels, and have thickness of one pixel. Squares are randomly uniformly

placed, so is their fill-in color, with size 6×6 pixels. We use different noise levels so we can study their impact, we use $[15\%, 25\%, 35\%, 45\%]$ of dataset. The systematic error corruption process is done by picking uniformly at random an instance, and then applying that systematic error. We use a range of trusted set sizes by defining the number of instances labelled per systematic error class (type) and per data class. The latter is the underlying classes in the dataset, e.g. item labels in *Fashion-MNIST*. For each class, either systematic error or data class, we provide label y for a few instances at random obtaining a trusted set. We use the range $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ labelled samples per class, which results in different trusted set sizes depending on number of classes. Particularly, we have the trusted set ranges: *Synthetic-Shapes* with $[40, 80, 200, 400]$ total samples; *Frey-Faces* with $[25, 50, 125, 250]$ total samples; *Fashion-MNIST* with $[90, 180, 450, 900]$ total samples. We create five examples (different random seeds) per noise level and per trusted set size, and train the models on them. The results are then averaged. Table 4.1 gives a summary of the datasets and their corruption (error types), number of systematic errors and data classes. Regarding this experimental setup, a more detailed description is provided below.

Detailed Description

Here we have a detailed description of the datasets and their corruption. For each noise level corruption in $\text{NL}_{\text{size}} = [15\%, 25\%, 35\%, 45\%]$, we instantiate five different examples of the same dataset using different random seeds. For each of those examples we build several trusted sets using the sizes in $\text{TS}_{\text{size}} = [5, 10, 25, 50]$, where labelled instances of smaller trusted sets are reused in bigger ones. The models are run on each example, for each trusted set, and results are then averaged.

Synthetic-Shapes This is a synthetic image dataset. It is meant to test the models in a simpler setting as it relates to the clean dataset. We treat the pixel values as a Bernoulli variable. The underlying clean dataset is composed of four different shapes (classes): a circle, a rectangle, an ellipse and a triangle. The shapes are filled by white pixels and the background is black. For each instance, the shape is placed uniformly at random inside the 28×28 black background. The systematic errors have four types, all are white lines that cross the square image from one side to another side. We have 4 fixed-in-place lines (two diagonal, one vertical, and one horizontal), affecting the same pixels, and sometimes intersecting

with the shapes. Hence, we have a total of eight underlying classes for trusted set constitution, i.e. four data and four systematic error classes.

The size of the images is 28×28, and the pixel values are binary $\{0, 1\}$, i.e. black and white. Overall, we have a dataset size of $N = 5000$, with the following split: train (80%), validation (10%), test (10%). Given the eight underlying classes, the size dataset N, and TS_{size}, we have the trusted set range: $[40, 80, 200, 400]$ total number of labelled instances, which corresponds to $[0.8\%, 1.6\%, 4\%, 8\%]$ of the entire dataset

Frey-Faces This is a gray-scale image dataset consisting of the same person with different facial expressions. We treat pixel values as continuous. In terms of data classes, we only have one (monolithic), since no labels for the expressions are provided. We have four systematic error classes, which consist of four randomly uniformly placed squares of 6×6 pixels. The place and color of these squares is defined by the random seed, when the corruption example is created. After that, always the same features are affected. Hence, in total we have five underlying classes for trusted set constitution, i.e. one data and four systematic error classes.

The size of images is 28×20, and the pixel values range from $[0, 256]$, i.e. gray-scale. The size of the entire dataset is $N = 1965$, with the following split: train (80%), validation (10%), test (10%). Given the above and TS_{size}, the trusted set range is: $[25, 50, 125, 250]$ total number of labelled instances, which corresponds to $[1.3\%, 2.5\%, 6.4\%, 12.7\%]$ of the entire dataset.

Fashion-MNIST This is a gray-scale image dataset, which consists of images of different types of clothing and accessories from an online merchant. There are 10 existing data classes, provided with the dataset. We have 4 fixed-in-place lines (two diagonal, one vertical, and one horizontal) where the color (black or white) for each depends on the random seed. Then we have 4 squares of size 6×6 placed randomly uniformly with random color, dependent on random seed. Hence, we have a total of 18 underlying classes for trusted set constitution, i.e. 10 data and 8 systematic error classes.

The size of the images is 28×28, and the pixel values are continuous with range $[0, 1]$, i.e. gray-scale. The original train set is of size 60000 instances, which we split for our actual train set of 54000 (90%) and validation set of 6000 (10%). We use the same test set of 10000 instances, so $N = 70000$. Given TS_{size} and the train

set size N, the trusted set range is: $[90, 180, 450, 900]$ total number of labelled instances, which corresponds to $[0.12\%, 0.25\%, 0.64\%, 1.28\%]$ of the entire dataset.

4.6.3 Comparative Models

Our baselines are VAEs since most of the relevant work in *sparse semi-supervision* is of this type (Ilse et al., 2020; Joy et al., 2020), and the task of repair is related with that of manipulating the reconstruction. We have four baselines: VAE-L2, CVAE, VAEGMM, CCVAE. For details on model architecture, hyperparameter selection and training see below in section 4.6.3.2. The VAE-L2 model is an unsupervised method tackling the issue of corruption by applying strong regularization (ℓ_2 regularization on weights). VAE-L2 uses the reconstruction likelihood for detection, more details in section 2.5.2.1. CVAE is the supervised version of the semi-supervised M2 model (Kingma et al., 2014), and should have better repair quality than M2. CVAE uses the reconstruction likelihood as an detection score. We found a smaller variance for $p(\boldsymbol{z})$ to be beneficial, for details see section 2.5.3.1. VAEGMM, based on (Willetts et al., 2020), is an improved version of M2 for the sparse semi-supervision setting. In this setting the M2 model tends to have posterior collapse issues with $q(y|\boldsymbol{x})$, picking one class over others. VAEGMM overcomes this issue, improving clustering and classification performance. Hence, we expect competitive detection performance from VAEGMM, for details see section 2.5.3.3. The CCVAE (Joy et al., 2020) is a state-of-the-art (SotA) semi-supervised VAE disentanglement model, allowing attribute manipulation in semi-supervised settings. For repair, we follow the automatic attribute manipulation procedure proposed by (Joy et al., 2020), for details see section 2.5.4.1. We adapted their code to our pipeline. Contrary to Joy et al. (2020), we found performance was superior when using a large up-sampling coefficient for the classifier, i.e. like β in CLSVAE. More details on the choice of β value for CCVAE are given in section 4.6.3.2 (see Hyperparameter Selection). We provide two versions of our model, with and without distance correlation penalty in section 4.5.4. So for CLSVAE-NODC use eq. (4.13) as training loss, whilst for CLSVAE use eq. (4.14).

4.6.3.1 Effects on Training: Scaling VAE latent prior variances whilst using SGD methods

In this section, we discuss the impact of scaling the variances of the $\boldsymbol{z}$ prior in the generative model, i.e. the hyperparameters σ_c and σ_d (see section 4.5.1). *This discussion also applies to other VAE models where changing variance values can also positively impact detection and repair performance*, e.g. VAEGMM (section 2.5.3.3) or CVAE (section 2.5.3.1). Note that all these VAE models use some type of stochastic gradient descent (SGD) optimization for training, e.g. Adam (Kingma & Ba, 2014) or AdaGrad (Duchi et al., 2011).

Theoretically, for the generative model in eq. (4.3) changing the variance of the Gaussian priors for $\boldsymbol{z}$ should not change the mathematical optimum of the ELBO loss – eq. (4.13). This is because the weights of both the encoder and decoder neural networks could be scaled together, thus nullifying the variance scaling effect – i.e. *scaled out*. However, empirically we see that when the model is trained using an optimization algorithm like SGD the change of variance does impact model performance. Unlike what is theoretically suggested above. This is likely due to a regularization effect of SGD (Sekhari et al., 2021) that conditions network weights, making it harder to arbitrarily scale them. Note that SGD uses noisy gradients during training, which likely justifies this effect. Hence, *if the weights are conditioned*, then varying the *variance of the prior of* $\boldsymbol{z}$ *will have an impact on the trained model*.

It should be noted that other papers on autoencoders (Rivera et al., 2020; Ruff et al., 2019; Pol et al., 2019) have also assumed that latent representations of outliers tend to lay at the tail end of the (data) distribution. This can mean assuming larger variances when modelling outliers in latent space, e.g. (Rivera et al., 2020; Ruff et al., 2019) and our CLSVAE. Or instead, like in Pol et al. (2019), assume that outliers will cause a larger KL divergence between posterior and prior distributions of $\boldsymbol{z}$. In this case, this basically means outlier instances will have more information to encode, which relates to a larger variance for outliers. Once more, all these AE models have been trained using SGD type optimization like CLSVAE.

Lastly, we point the reader to section 4.6.5.3, where an additional experiment is provided showing that data corrupted with outliers has larger variance than

inlier data alone. This was carried out using a standard VAE (Kingma & Welling, 2014), so the discussion is broadly applicable.

4.6.3.2 Model Architectures, Hyperparameters and Training

In this section, we provide a detailed description on how the models introduced above were setup for the experiments. Specifically, we describe hyperparameter value selection, neural network architectures, and training procedure (e.g. optimization algorithm).

Hyperparameter Selection

In our experiments, we tuned model hyperparameters according to outlier detection performance, which means highest AVPR (average precision). This was evaluated on the trusted set, the only labelled part of the dataset. Often, we would look at the repairs (reconstructions) offered by each model for the trusted set. This way we confirm that the repair process is reasonable enough, and no adjustment is needed on the hyperparameters side. In the case of VAE-L2 we had to not just account for AVPR in the trusted set, but also check repair performance via SMSE (standardized mean squared error). This is because strong regularization, higher ℓ_2 coefficient, often leads to better outlier detection performance, but that comes at the cost of repair quality due to the VAE collapsing to mean behaviour. For each model, the hyperparameter search was carried out for each dataset at a noise level of 35%, and with 25 samples per class.

- **VAE-L2** In *Synthetic-Shapes*: 200 epochs; KL divergence annealing used; ℓ_2 coefficient is 35.0. In *Frey-Faces*: 300 epochs; KL divergence annealing used; ℓ_2 coefficient is 100.0. In *Fashion-MNIST*: 100 epochs; KL divergence annealing used; ℓ_2 coefficient is 100.0.
- **VAEGMM** In *Synthetic-Shapes*: 200 epochs; KL divergence annealing used; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 1000$; $\sigma_{y=1} = 0.9$ and $\sigma_{y=0} = 5.0$. In *Frey-Faces*: 300 epochs; KL divergence annealing used; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 1000$; $\sigma_{y=1} = 0.6$ and $\sigma_{y=0} = 5.0$. In *Fashion-MNIST*: 100 epochs; KL divergence annealing used; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 100$; $\sigma_{y=1} = 0.5$ and $\sigma_{y=0} = 5.0$.

- **CVAE** In *Synthetic-Shapes*: 200 epochs; KL divergence annealing used; $\sigma = 0.5$. In *Frey-Faces*: 300 epochs; KL divergence annealing used; $\sigma = 0.2$. In *Fashion-MNIST*: 100 epochs; KL divergence annealing used; $\sigma = 0.5$.
- **CCVAE** In *Synthetic-Shapes*: 200 epochs; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 50000.0$. In *Frey-Faces*: 300 epochs; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 10000.0$. In *Fashion-MNIST*: 100 epochs; fraction of clean data $\alpha = 0.6$; (trusted set) up-sampling coefficient $\beta = 250000.0$.
- **CLSVAE** In *Synthetic-Shapes*: 200 epochs; fraction of clean data $\alpha = 0.6$; KL divergence annealing used; (trusted set) up-sampling coefficient $\beta = 1000.0$. $\sigma_\epsilon = 0.5$; $\sigma_c = 0.5$; $\sigma_d = 5.0$; distance correlation (DC) penalty used; λ_t annealing ratio of 0.5 (DC penalty); λ_T maximum value of 100.0 (DC penalty). In *Frey-Faces*: 300 epochs; fraction of clean data $\alpha = 0.6$; KL divergence annealing used; (trusted set) up-sampling coefficient $\beta = 1000.0$. $\sigma_\epsilon = 0.6$; $\sigma_c = 0.2$; $\sigma_d = 5.0$; distance correlation (DC) penalty used; λ_t annealing ratio of 0.5 (DC penalty); λ_T maximum value of 1000.0 (DC penalty). In *Fashion-MNIST*: 100 epochs; fraction of clean data $\alpha = 0.6$; KL divergence annealing used; (trusted set) up-sampling coefficient $\beta = 100.0$. $\sigma_\epsilon = 0.1$; $\sigma_c = 0.2$; $\sigma_d = 5.0$; distance correlation (DC) penalty used; λ_t annealing ratio of 0.5 (DC penalty); λ_T maximum value of 1000.0 (DC penalty).
- **CLSVAE-NODC** Same hyperparameter options as **CLSVAE** but without the distance correlation penalty.

Note the annealing ratio above for the KL divergence and for the DC penalty is inspired by Fu et al. (2019). We use only one cycle (monotonic), and the R (or *ratio*) is the proportion used to increase the penalty coefficient (or KL term coefficient), for instance a value of 0.5.

Now we discuss in more detail how the β value was chosen for the CCVAE model. In our experiments, the β in CCVAE was set to higher values so it could compensate for the very small trusted set sizes used herein. Empirically, at first we tried setting β to small values, or even $\beta = 1$ as suggested in Joy et al. (2020). However, CCVAE when using a small β was not able to disentangle properly the clean and dirty patterns, specifically in smaller trusted sets. This was noticeable

when we used CCVAE for repairing the data instance, where it was common to find part of the error pattern still present. As a result, the CCVAE model had poor detection and repair performance. On the other hand, empirically we found that higher β values resulted in properly disentangled clean and dirty error patterns when repairing data instances. In this case, CCVAE had good outlier detection and data repair performance.

We hypothesize that the original application of CCVAE assumed larger labelled sets, and probably better balanced as well (labels of data attributes). Note that outliers are often smaller portions of the entire dataset, unlike the labelled attributes of the applications in Joy et al. (2020). This means that there is a class imbalance problem in typical outlier detection tasks. We believe that a higher β allows to virtually up-sample the importance of the small trusted set when training the CCVAE and other models, particularly outlier examples. It is probably the case that smaller values of β are not enough to bias the latent space $\boldsymbol{z}_c$ to properly disentangle clean and dirty patterns. As a result, the data repair performance of CCVAE and other models suffers.

Optimization

We used the PyTorch framework to code all our models, and trained on a GeForce TITAN X GPU. All models were trained using the Adam optimizer (Kingma & Ba, 2014), with an initial learning rate of 0.001.

Model Architectures

For continuous type data, i.e. *Fashion-MNIST* and *Frey-Faces*, we used the Gaussian distribution as the likelihood of each pixel in the reconstruction loss. The variance of the Gaussian distribution is shared amongst all the pixels in the image, and it is learnt as a parameter of the model. This was done for all models. For binary type data, i.e. *Synthetic-Shapes*, we treated each pixel as a Bernoulli variable, and used the log-likelihood of this distribution for each pixel in the reconstruction loss. This was done for all models.

We used very similar encoder and decoder architectures for all models, so as to be fair and the results comparable. In the case of CLSVAE we used two encoders, one for the clean subspace $\boldsymbol{z}_c$, and the other for the dirty subspace $\boldsymbol{z}_d$. This architecture yielded better results for us in terms of repair in the trusted set.

Encoder	Decoder
(img_size, 200) →	→ (15, 50) →
→ ReLU →	→ ReLU →
→ (200, 100) →	→ (50, 100) →
→ ReLU →	→ ReLU →
→ (100, 50) →	→ (100, 200) →
→ ReLU →	→ ReLU →
→ 2 × (50, 15)	→ (200, img_size)

Table 4.2: Architecture of encoder and decoder for VAE and VAEGMM. Further, for binary pixels the decoder will use a Sigmoid non-linearity at the end.

Encoder	Decoder
(img_size, 200) →	→ (16, 50) →
→ ReLU →	→ ReLU →
→ (200, 100) →	→ (50, 100) →
→ ReLU →	→ ReLU →
→ (100, 50) →	→ (100, 200) →
→ ReLU →	→ ReLU →
→ 2 × (50, 15)	→ (200, img_size)

Table 4.3: Architecture of encoder and decoder for CVAE. Further, for binary pixels the decoder will use a Sigmoid non-linearity at the end.

CLSVAE architecture can be seen in Table 4.5 for the encoders and decoder, and the classifier can be seen in Table 4.7. In Table 4.2, we see the neural architecture for the encoder and decoder of VAE and VAEGMM. The classifier architecture for the VAEGMM can be seen in Table 4.6. In Table 4.3, we find the architecture for the encoder and decoder of CVAE. In Table 4.4, we find the architecture for the encoder and decoder of CCVAE. The classifier architecture of CCVAE is just $\boldsymbol{z}_c$ multiplied by a weight parameter plus a bias parameter, and then a *sigmoid* non-linearity is applied – like in Joy et al. (2020).

Encoder	Decoder
(img_size, 200) $\rightarrow$	$\rightarrow$ (16, 50) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow$ (200, 100) $\rightarrow$	$\rightarrow$ (50, 100) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow$ (100, 50) $\rightarrow$	$\rightarrow$ (100, 200) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow 2 \times$ (50, 16)	$\rightarrow$ (200, img_size)

Table 4.4: Architecture of encoder and decoder for CCVAE. Further, for binary pixels the decoder will use a Sigmoid non-linearity at the end.

Encoder $\boldsymbol{z}_c$	Encoder $\boldsymbol{z}_d$	Decoder
(img_size, 200) $\rightarrow$	(img_size, 200) $\rightarrow$	$\rightarrow$ (15, 50) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow$ (200, 100) $\rightarrow$	$\rightarrow$ (200, 100) $\rightarrow$	$\rightarrow$ (50, 100) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow$ (100, 50) $\rightarrow$	$\rightarrow$ (100, 50) $\rightarrow$	$\rightarrow$ (100, 200) $\rightarrow$
$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$	$\rightarrow$ ReLU $\rightarrow$
$\rightarrow 2 \times$ (50, 10)	$\rightarrow 2 \times$ (50, 5)	$\rightarrow$ (200, img_size)

Table 4.5: Architecture of encoder and decoder for CLSVAE. Note for CLSVAE latent space of size 15 is split: 10 for $\boldsymbol{z}_c$ and 5 for $\boldsymbol{z}_d$. Further, for binary pixels the decoder will use a Sigmoid non-linearity at the end.

Classifier
(img_size, 200) →
ReLU
→ (200, 100) →
ReLU
→ (100, 50) →
ReLU
→ (50, 1)
Sigmoid

Table 4.6: Architecture of classifier VAEGMM.

Classifier
(15, 7) →
ReLU
→ (7, 5) →
ReLU
→ (5, 1)
Sigmoid

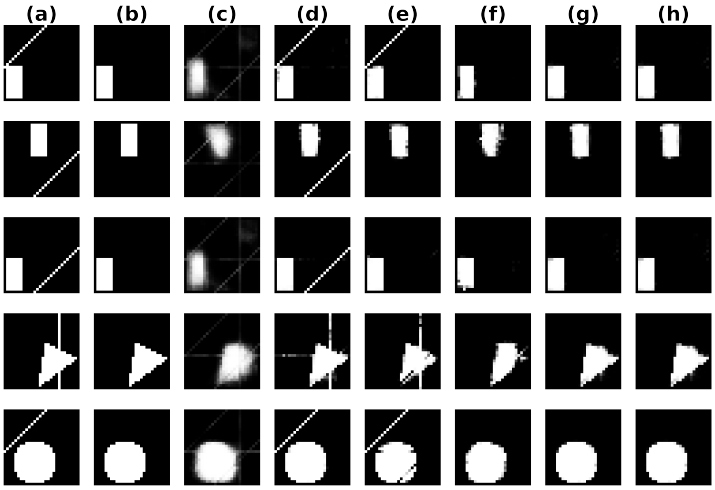
Table 4.7: Architecture of classifier CLSVAE, input is $\boldsymbol{z}$.

4.6.4 Discussion of Results

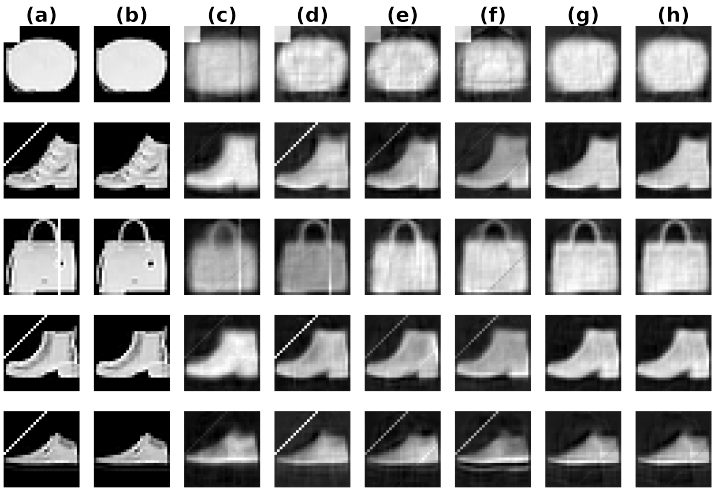
In Figure 4.2(a) we show performance as a function of the size of the trusted set, i.e. *sweep* of trusted set sizes, for a 35% noise level. Similar performance is seen for other datasets (see section 4.6.5.1). Table 4.2(b) shows the results for all datasets for a 35% noise level and trusted set size of 10 labelled samples per class. Results for all trusted set sizes and noise levels are found in section 4.6.5.1, with similar analysis on performance. Additional examples of repairs are seen in section 4.6.5.2, including inlier instances, 45% noise level, and 5 samples per class (trusted set size) for *Synthetic-Shapes* since its an easier dataset.

Outlier Detection Looking at Figure 4.2, we see that on average both CLSVAE (our model) and CLSVAE-NODC have the highest AVPR, registering the best detection performance, with similar scores. We see that CCVAE, the previous SotA, has similar detection performance as CLSVAE. VAEGMM, also semi-supervised, lags behind both likely because its designed for slightly larger trusted sets. All semi-supervised models (CLSVAE, CLSVAE-NODC, CCVAE, VAEGMM) improve their detection performance as the trusted set grows larger. CVAE and VAE-L2 do not use a trusted set, and thus have the same performance throughout all the trusted set range in Figure 4.2(a). These two observations about the trusted set range are also seen, for all datasets and noise levels, in section 4.6.5.1. CVAE is supervised, still it shows poor performance, this may be due to: issues linked to (decoder) likelihood-based scores (Eduardo et al., 2020; Lan & Dinh, 2020); poor fitting to the data, thus impacting negatively the score. VAE-L2 uses a likelihood score, and is unsupervised, so poorer detection performance is understandable. This highlights semi-supervision as being important in systematic error detection. VAE-L2 registering good performance in *Synthetic-Shapes* (see Figure 4.2(b)) is likely due to this dataset being easier. Lastly, we note that CLSVAE tends to have better detection performance in higher noise levels relative to other methods (see section 4.6.5.1). Therein, CCVAE has close to or similar detection performance as CLSVAE.

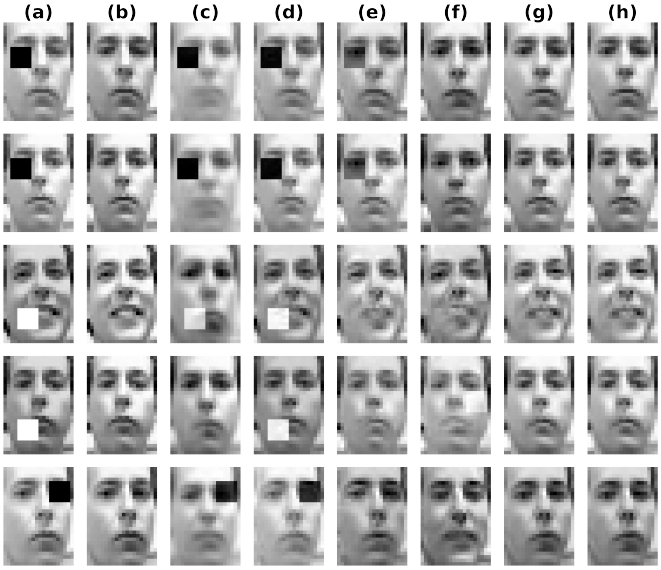
Automated Repair In Figure 4.2, we see that on average CLSVAE (our model) is best at automated repair (lowest SMSE on dirty pixels). We also see that distortion (repair clean pixels, SMSE) is relatively low, but not the lowest. This results in CLSVAE overall being the best repair method, not only replacing pixel values of the systematic error, but also inferring correctly the structure of the



(a) *Synthetic-Shapes*: 35% noise, 10 labels per class (1.6% of dataset).

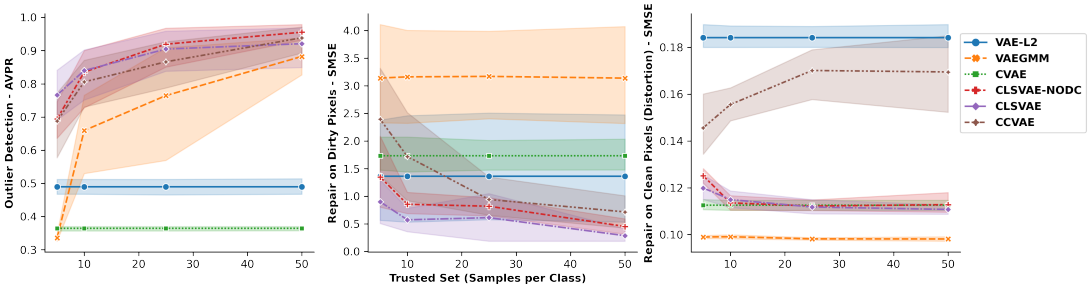


(b) *Fashion-MNIST*: 35% noise, 10 labels per class (0.25% of dataset).



(c) *Frey-Faces*: 35% noise, 10 labels per class (2.5% of dataset)

Figure 4.1: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE.



(a)

Dataset	Model	Outlier Detection (AVPR ↑)	Repair on Dirty Pixels (SMSE ↓)	Repair on Clean Pixels (SMSE ↓)
Synthetic-Shapes	VAE-L2 (Kingma & Welling, 2014)	0.93 (0.03)	0.049 (0.008)	0.015 (0.002)
	VAEGMM (Willetts et al., 2020)	0.62 (0.10)	0.974 (0.009)	0.003 (3e-4)
	CVAE (Kingma et al., 2014)	0.47 (0.03)	0.429 (0.114)	0.003 (3e-4)
	CCVAE (Joy et al., 2020)	**0.98** (0.03)	**0.031** (0.023)	0.008 (0.001)
	CLSVAE-NODC **(Ours)**	**0.99** (3e-4)	**0.018** (0.024)	**0.002** (3e-4)
	CLSVAE **(Ours)**	**0.99** (0.02)	**0.014** (0.008)	**0.005** (0.004)
Fashion-MNIST	VAE-L2 (Kingma & Welling, 2014)	0.49 (0.03)	**1.362** (1.140)	0.175 (0.004)
	VAEGMM (Willetts et al., 2020)	**0.66** (0.13)	3.161 (1.032)	**0.095** (0.001)
	CVAE (Kingma et al., 2014)	0.36 (0.01)	1.732 (0.335)	**0.099** (0.001)
	CCVAE (Joy et al., 2020)	**0.81** (0.09)	1.719 (0.956)	0.136 (0.003)
	CLSVAE-NODC **(Ours)**	**0.84** (0.10)	**0.854** (0.214)	0.107 (0.002)
	CLSVAE **(Ours)**	**0.84** (0.08)	**0.572** (0.238)	0.108 (0.002)
Frey-Faces	VAE-L2 (Kingma & Welling, 2014)	0.73 (0.14)	10.32 (6.118)	0.420 (0.033)
	VAEGMM (Willetts et al., 2020)	**0.96** (0.06)	22.32 (4.112)	**0.070** (0.003)
	CVAE (Kingma et al., 2014)	0.42 (0.02)	3.190 (0.675)	0.111 (0.024)
	CCVAE (Joy et al., 2020)	**0.99** (0.01)	0.947 (0.123)	0.270 (0.059)
	CLSVAE-NODC **(Ours)**	**0.85** (0.13)	**0.269** (0.078)	0.172 (0.048)
	CLSVAE **(Ours)**	**0.99** (0.02)	**0.321** (0.168)	0.177 (0.033)

(b)

Figure 4.2: Outlier detection uses AVPR score where **highest is best**. Repair for dirty pixels, and for clean pixels (distortion), uses SMSE where **lowest is best**. (a) Trusted set range sweep for *Fashion-MNIST* where [0.12, 0.25, 0.64, 1.28] % of the dataset, at 35 % noise level. (b) Table for results at 35% noise level, and 10 labelled samples per class for the trusted set. Boldface corresponds to the best performances within a standard error, and green color to best mean performance overall. Standard error in brackets.

ground-truth repair (both clean and dirty pixels). This is confirmed in Figure 4.1, where reconstructions (repairs) by CLSVAE show the best quality: replacing the error values of affected pixels and recovering the underlying ground-truth, whilst preserving the uncorrupted image portion (i.e. low distortion). CLSVAE has slightly better repair than CLSVAE-NODC on average, but most importantly, it has better performance stability than CLSVAE-NODC, which can be seen in section 4.6.5.1 for repair (dirty pixels). Further, in section 4.6.5.1, repair with CLSVAE is more advantageous relative to other models at higher noise levels. As expected, semi-supervised models (CLSVAE, CLSVAE-NODC, CCVAE) improve their repair of dirty pixels as the trusted set increases (see Figure 4.2(a)). Both VAE-L2 and CVAE do not use a trusted set, so performance is static. CCVAE has the ability to perform good repair, registering the second best repair for dirty pixels after CLSVAE, but often with higher distortion. CCVAE suffers from two issues that account for its worse performance relative to CLSVAE. For one, looking at Figure 4.1, CCVAE can sometimes fail to replace the pixel values from systematic errors. Secondly, and more often, it can fail to recover the underlying ground-truth even when replacing erroneous pixel values. Similarly, it has difficulty preserving the uncorrupted image portion (higher distortion). So some information about inlier appearance is being lost. This is explained by the fact that CCVAE latent space is not disentangled regarding the clean and dirty patterns. CVAE can repair some outliers well, but it fails to deal with other systematic errors, which leads to an overall poor repair performance. This is maybe due to the binary latent variable used for y making it harder to model multiple systematic errors, for more discussion see Joy et al. (2020). VAE-L2 is able to repair some errors, but overall has worse repair than CLSVAE. Its strong regularization, optimized mostly for detection, leads to higher distortion and loss of detail (see Figure 4.1). Its higher standard error (erratic repairs) is due to not being able to distinguish between clean and dirty patterns. VAEGMM does not do well in repair. This is likely due to it being better suited for classification or clustering tasks.

4.6.4.1 Timing Information on Models

All the models explored in this chapter are VAEs with similar architectures in terms of their neural networks, however some differences exist in terms of inference time. For all models a GPU (GeForce TITAN X) was used for training, and generally models use the same order of magnitude in terms of computation time.

The models that take less time are VAE-L2 and CVAE since they have simple architectures and inference procedures. Taking a bit more time we have VAEGMM, since an additional classifier neural network has to be trained. The models CCVAE and CLSVAE have similar computation times, and are the most expensive to train.

As an example, for *Fashion-MNIST* the training time for VAE-L2 and CVAE is about 35 minutes, whilst for VAEGMM is an additional 5 minutes totalling 40 minutes. Meanwhile for CLSVAE we register about 40 minutes of compute time, and for CCVAE we register about 50 minutes overall. Note that the *Fashion-MNIST* dataset is the largest dataset in our experiments, and thus it is the most expensive in terms of computation time.

4.6.5 Additional Results

The purpose of this section is to provide a **complement of results to the main results section** (Section 4.6.4), and show the reader the complete set of experiments. As such, we present additional figures for the experiments carried out in this chapter.

These figures show the entirety of the experimental scenarios as defined in section 4.6.2. In section 4.6.5.1, results are shown for outlier detection (AVPR) and data repair (SMSE) for all noise levels and trust sets sizes. The main conclusions made in the discussion section 4.6.4 also apply to these results.

In addition, in section 4.6.5.2, supplementary examples of outlier instances being repaired by the different models are shown, where different trusted set sizes and noise levels are explored. Reconstructions of inlier instances are also shown so as to check if clean data is being correctly modelled.

Lastly, in section 4.6.5.3, we present a simple experiment providing additional evidence to the claim that corrupted datasets (with outliers) have larger variance (entropy) when modelled than clean datasets. This means that there is more information to be modelled by the VAE in corrupted datasets (with outliers), compared to less information to be modelled in clean datasets (inliers only). We also show that clean data (inliers) can be modelled by smaller latent space in VAEs when compared to dirty data. Both these assumptions are tied to the fact that corrupted datasets have more diversity due to added errors, and hence more

information to model. Note that these assumptions were made in the generative model of CLSVAE, in section 4.5.1, which helped discern inliers from outliers and thus improve model performance.

4.6.5.1 Results for all Noise Levels and Trusted Set Sizes (Sweep)

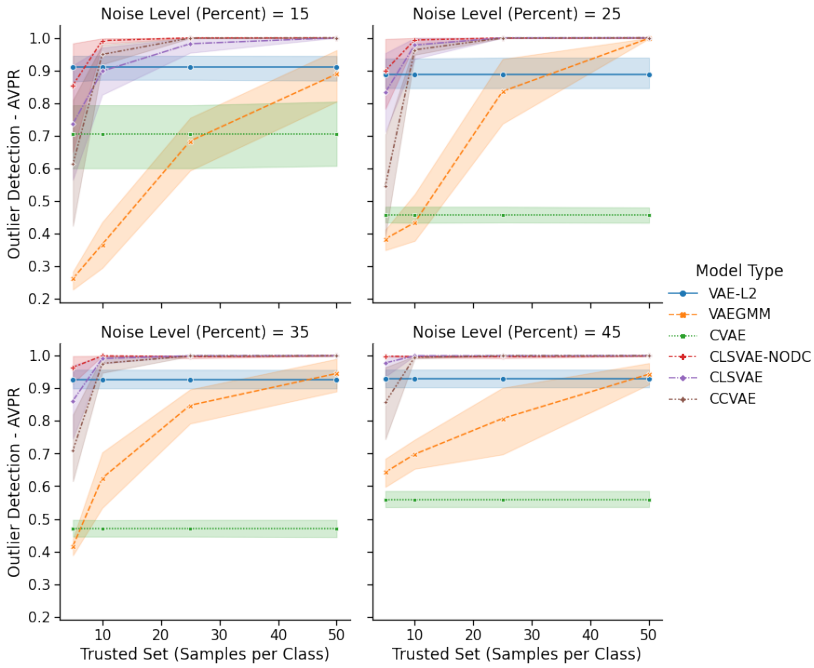


Figure 4.3: **Synthetic-Shapes**. Outlier detection (AVPR) where *higher is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e.$[0.8\%, 1.6\%, 4\%, 8\%]$ of the entire dataset.

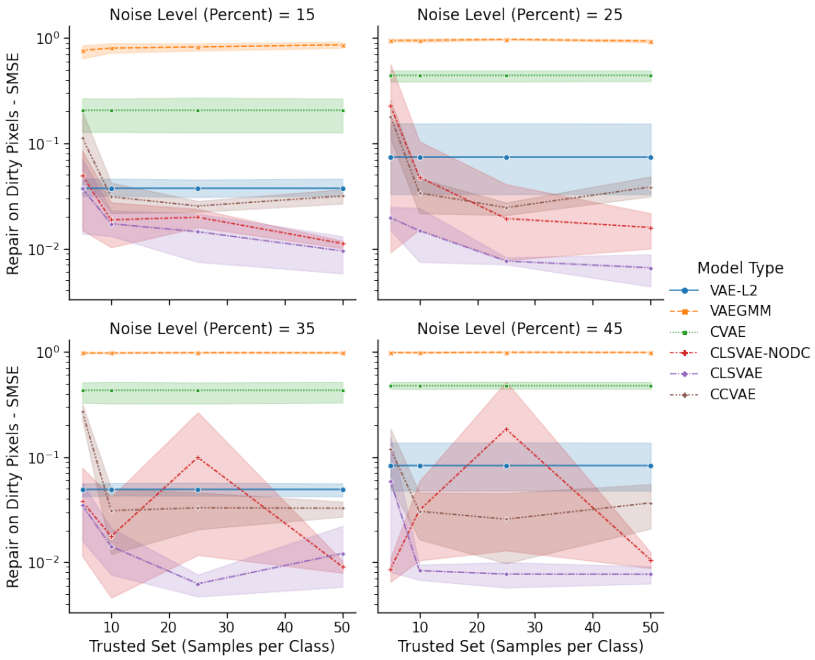


Figure 4.4: **Synthetic-Shapes**. Repair of dirty pixels in outliers (SMSE), where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[0.8\%, 1.6\%, 4\%, 8\%]$ of the entire dataset.

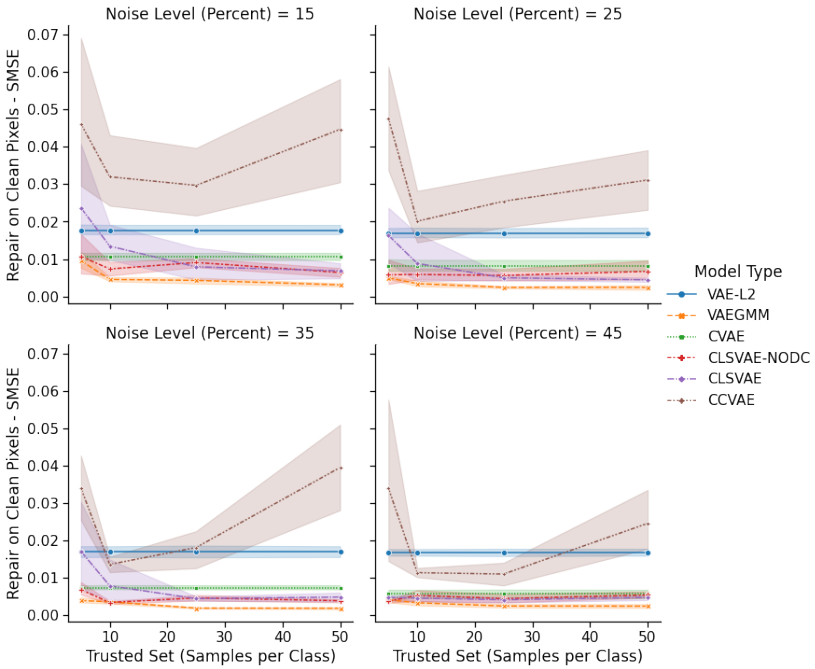


Figure 4.5: **Synthetic-Shapes**. Repair of clean pixels in outliers (SMSE), i.e. distortion, where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[0.8\%, 1.6\%, 4\%, 8\%]$ of the entire dataset.

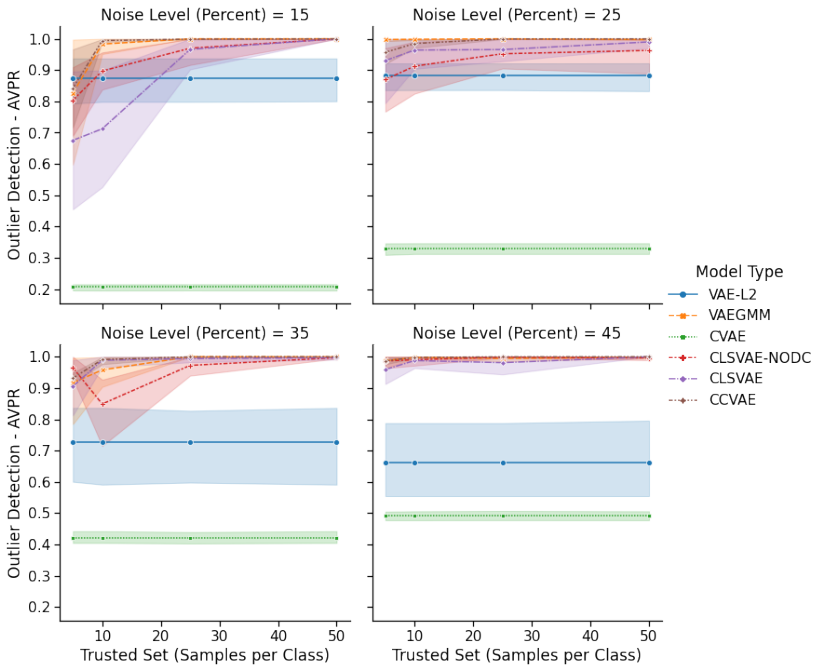


Figure 4.6: **Frey-Faces**. Outlier detection (AVPR) where *higher is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[1.3\%, 2.5\%, 6.4\%, 12.7\%]$ of the entire dataset.

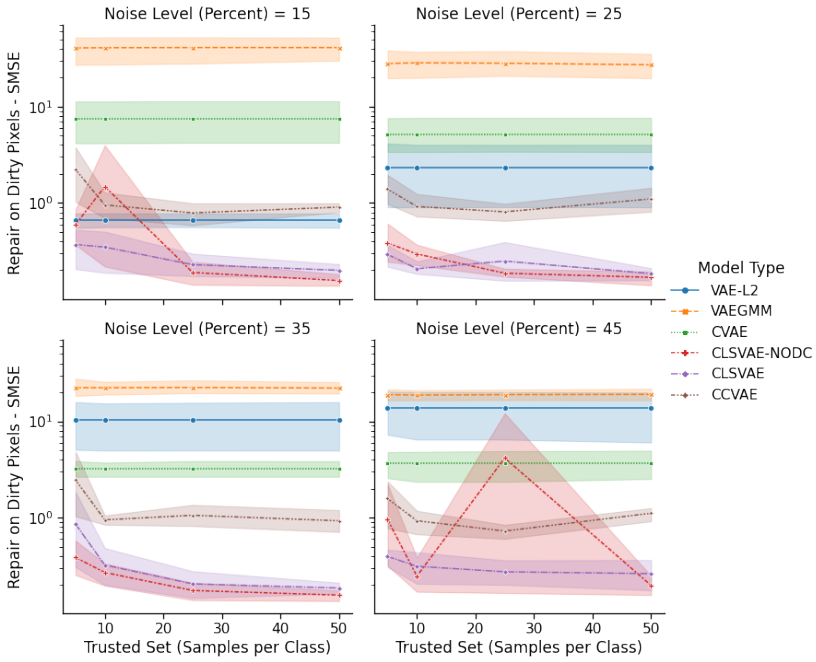


Figure 4.7: **Frey-Faces**. Repair of dirty pixels in outliers (SMSE), where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[1.3\%, 2.5\%, 6.4\%, 12.7\%]$ of the entire dataset.

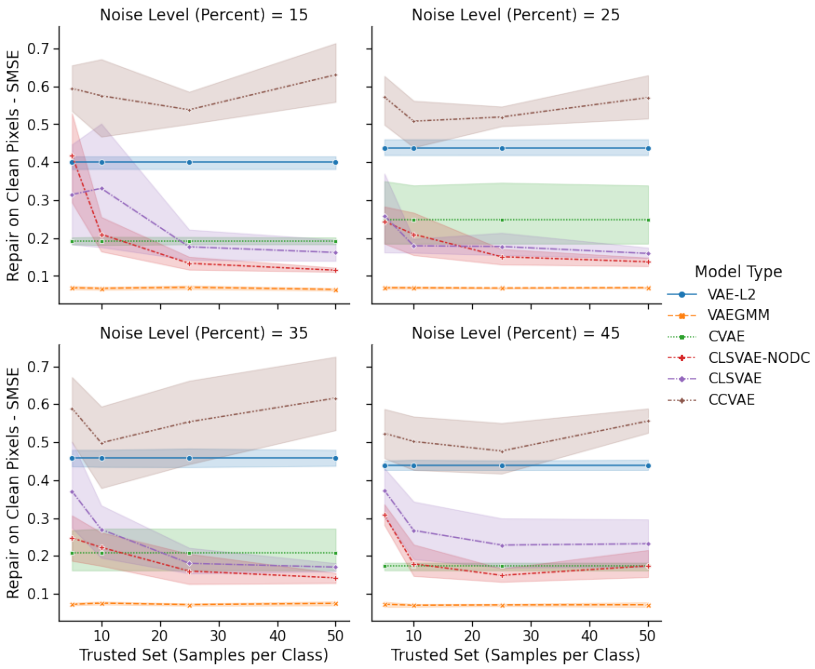


Figure 4.8: **Frey-Faces**. Repair of clean pixels in outliers (SMSE), i.e. distortion, where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[1.3\%, 2.5\%, 6.4\%, 12.7\%]$ of the entire dataset.

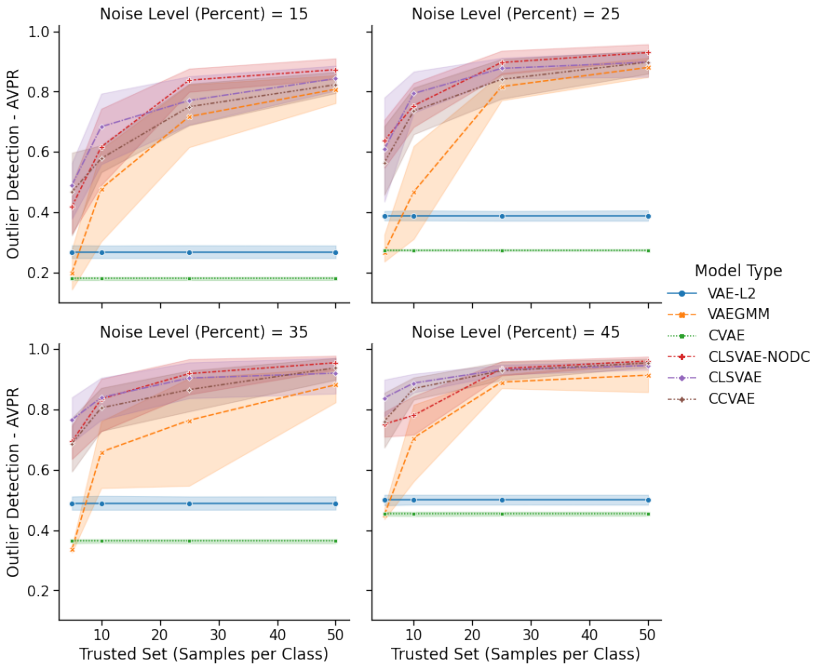


Figure 4.9: **Fashion-MNIST**. Outlier detection (AVPR) where *higher is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[0.12\%, 0.25\%, 0.64\%, 1.28\%]$ of the entire dataset.

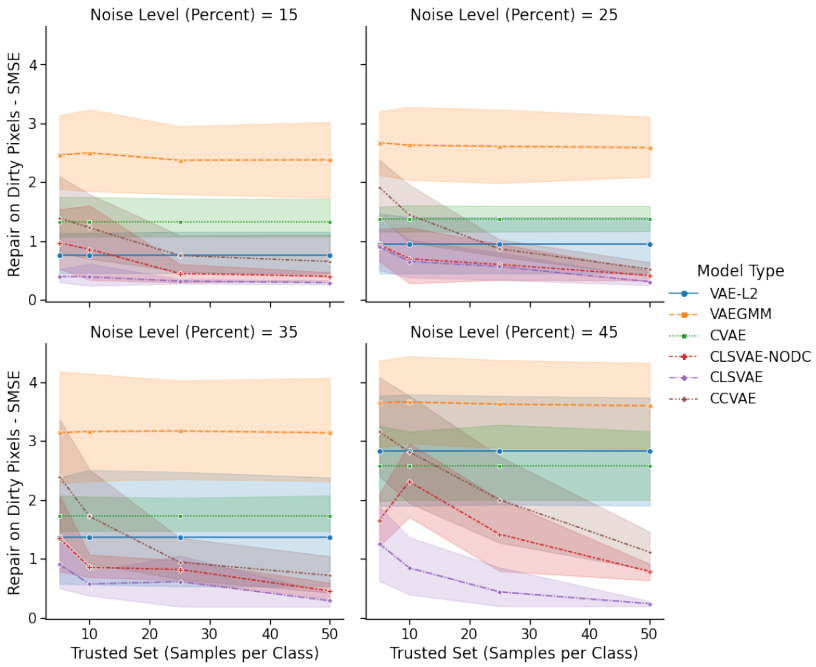


Figure 4.10: **Fashion-MNIST**. Repair of dirty pixels in outliers (SMSE), where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[0.12\%, 0.25\%, 0.64\%, 1.28\%]$ of the entire dataset.

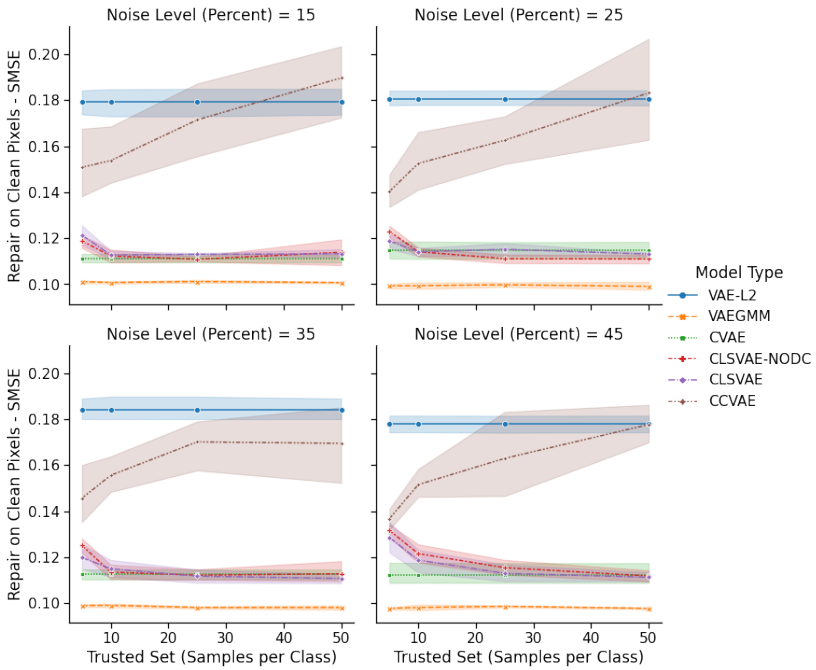


Figure 4.11: **Fashion-MNIST**. Repair of clean pixels in outliers (SMSE), i.e. distortion, where *lower is better*. Trusted set range sweep where $\text{TS}_{\text{size}} = [5, 10, 25, 50]$ samples per class, i.e. $[0.12\%, 0.25\%, 0.64\%, 1.28\%]$ of the entire dataset.

4.6.5.2 Additional Reconstructions (Repairs) for all Datasets

Synthetic-Shapes

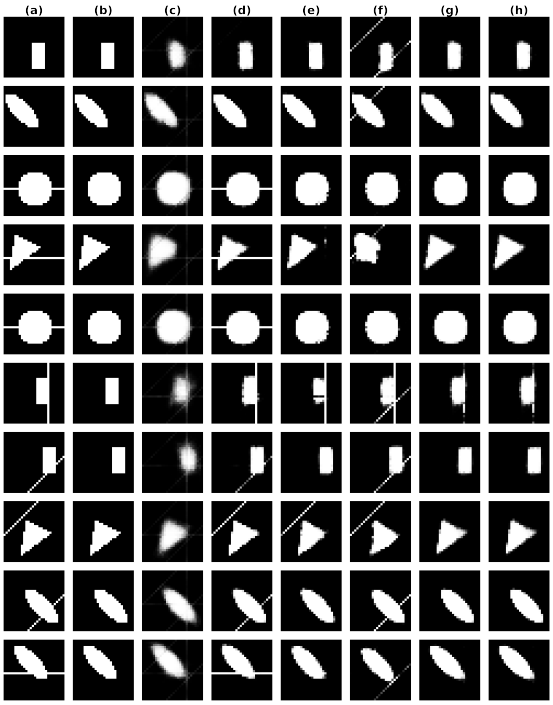


Figure 4.12: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Synthetic-Shapes*: **35% noise**, **5 labels per class** (0.8% of dataset).

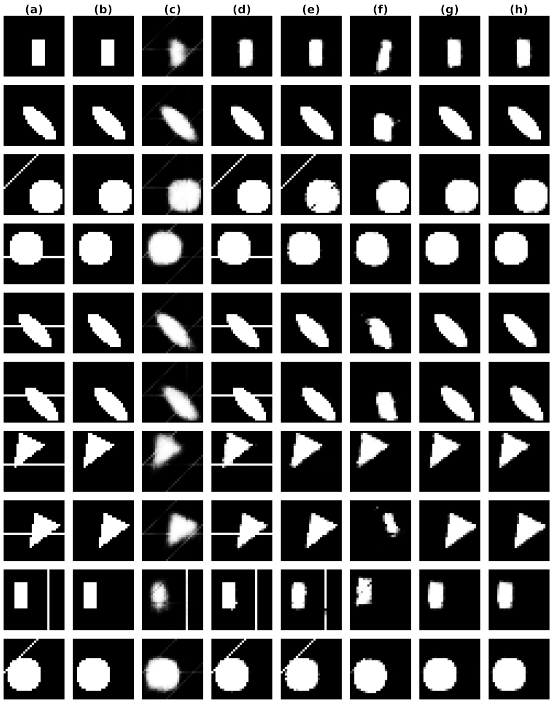


Figure 4.13: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Synthetic-Shapes*: **35% noise**, **50 labels per class** (8% of dataset).

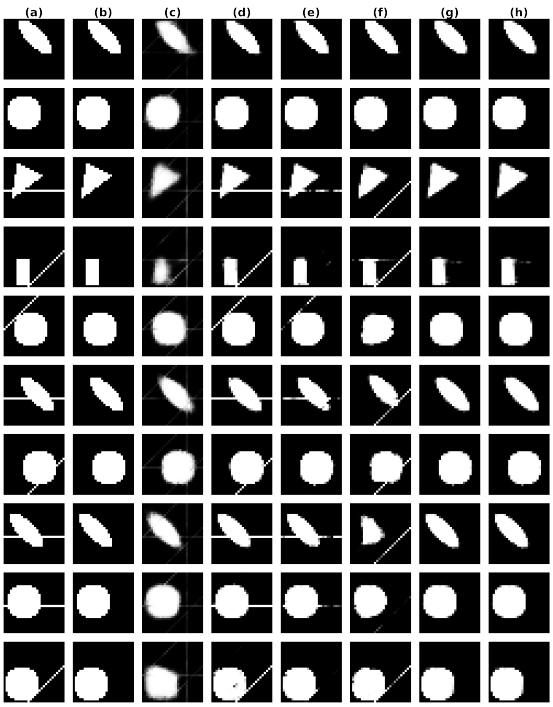


Figure 4.14: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Synthetic-Shapes*: **45% noise**, **5 labels per class** (0.8% of dataset).

Frey-Faces

Figure 4.15: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Frey-Faces*: **35% noise**, **10 labels per class** (2.5% of dataset).

Figure 4.16: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Frey-Faces*: **35% noise**, **50 labels per class** (12.7% of dataset).

Figure 4.17: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Frey-Faces*: **45% noise**, **10 labels per class** (2.5% of dataset).

Fashion-MNIST

Figure 4.18: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Fashion-MNIST*: **35% noise**, **10 labels per class** (0.25% of dataset).

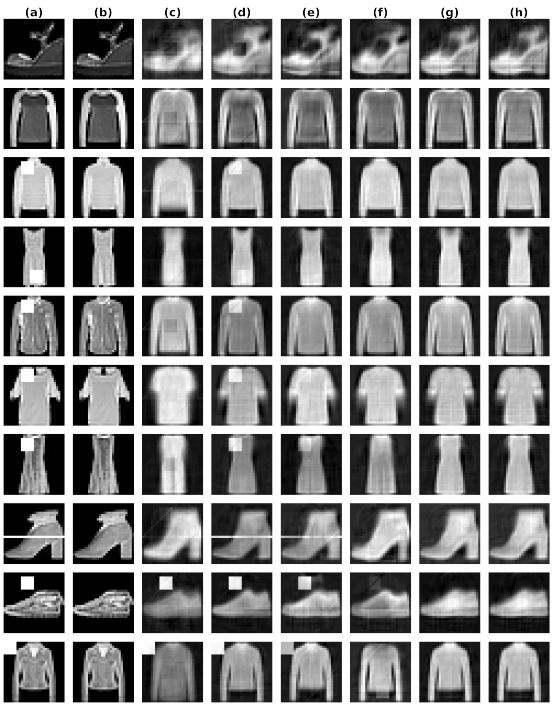


Figure 4.19: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Fashion-MNIST*: **35% noise**, **50 labels per class** (1.28% of dataset).

Figure 4.20: Images for model repair (reconstruction), outlier (corrupted) and inlier (uncorrupted): (a) Original (Outlier); (b) Ground-Truth (Inlier); (c) VAE-L2; (d) VAEGMM; (e) CVAE; (f) CCVAE; (g) CLSVAE-NODC; (h) CLSVAE. **The first two rows are inlier examples**, the others being outliers. *Fashion-MNIST*: **45% noise**, **10 labels per class** (0.25% of dataset).

4.6.5.3 Testing Standard VAE: Entropy of Clean vs. Corrupted Data

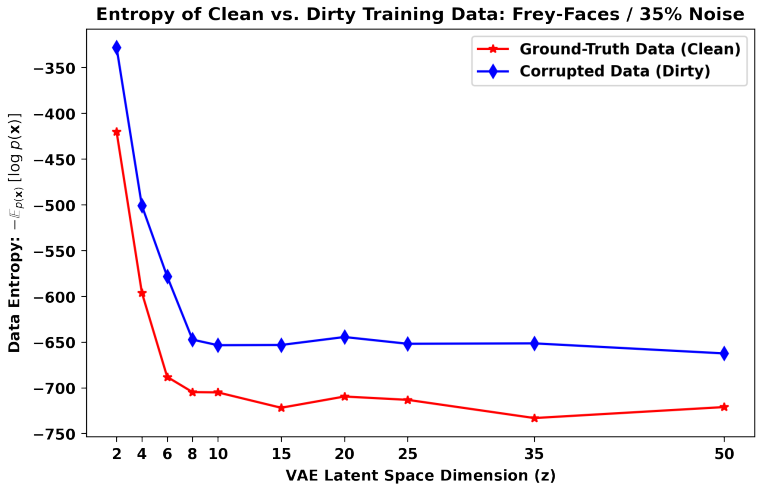


Figure 4.21: Entropy of ground-truth training data (clean: without corruption) vs the entropy of corrupted training data (as in Table 4.1). Entropy estimation via IWAE (Burda et al., 2016), using a standard VAE (not regularized). VAE uses same architecture of section 4.6.3.2, except dimension of $\mathbf{z}$ (latent space) now has the range [2, 4, 6, 8, 10, 15, 20, 25, 35, 50] (x-axis). *Frey-Faces* training data, with 35% noise level for corrupted dataset.

In this section, we experimentally compare the entropy of clean (without corruption) training data, and the entropy of corrupted training data. A larger entropy means that a dataset has larger variance overall. Note that in the setup of our problem, i.e. repairing systematic errors (see section 4.4), only corrupted data is used for training. In Figure 4.21, for *Frey-Faces*, we compare the estimated entropy of clean training data against one that has been corrupted (35 % noise, corruption as in Table 4.1). We estimate the entropy by first training a standard VAE model on the dataset (clean or corrupted), and then after training, we compute a tight bound on the marginal log-likelihood of that dataset. We compute this tight bound via IWAE estimator (Importance Weighted Autoencoders, (Burda et al., 2016)), where we use $K = 250$ samples. Note that entropy is $\mathcal{H}(x) = -\mathbb{E}_{p_\theta(x)}[\log p_\theta(x)]$, and hence marginal log-likelihood is just $-\mathcal{H}(x)$. We vary the dimension of VAE latent space ($\mathbf{z}$) in order to see how well the model can learn the training data. For this experiment, the VAE is not regularized.

The main idea is to see whether corrupted data (with outliers) has larger variance than clean data (inliers only). Once more, larger entropy equates to larger data

variance. A dataset with larger variance is a dataset with more diversity in terms of the patterns it contains, and hence it has more information to be modelled. Therefore corrupted data has more information to be modelled by the latent space of a VAE compared to clean data.

Our claim, supported in literature (Eduardo et al., 2020; Ruff et al., 2019), is that a dataset that has been corrupted has larger variance (entropy) because the added outliers (e.g. systematic errors) increase data pattern diversity. Looking at Figure 4.21, we see that overall the entropy of corrupted data is larger than clean data, for all sizes of $\mathbf{z}$. Hence, corrupted data has larger variance than clean data. For the smaller dimensions of $\mathbf{z}$, in range $[2, 10]$ units, we see that the VAE has less trouble learning the clean data compared to the corrupted data. This is also evidenced by how more quickly the entropy decreases for clean data relative to corrupt data, as we increase the latent space size in $[2, 10]$. We conclude that a VAE only needs a smaller latent space (subspace) to model clean data (inliers), and that corrupted data needs a larger latent space to be modelled properly.

4.7 Concluding Remarks

We have proposed a novel semi-supervised VAE (CLSVAE) for outlier detection and automated repair, in the presence of systematic errors. Our model exploits the fact that systematic errors are predictable by high capacity models, unlike random errors. Thus, CLSVAE partitions the latent space into two subspaces: one for clean patterns, and another for dirty ones. Inliers are only modelled by the clean pattern subspace, whilst outliers use both subspaces. We encourage low mutual information between these subspaces through a penalty, improving performance stability. Empirically this encourages higher fidelity repairs by the model, without human in the loop or other post-processing. We show CLSVAE only needs a small trusted set, requiring the user to label less data. We show that unsupervised models may not be able to distinguish between clean patterns and systematic errors, and strong regularization leads to a lower quality repair. Experimentally, CLSVAE showed superior repair quality and performance compared to other supervised and semi-supervised models, including a state-of-the-art VAE latent disentanglement model. Note that some of these baseline models were specially designed to handle small labelled sets. Further, we notice that CLSVAE is able to

repair more instances affected by systematic errors compared to baseline models. Experiments were carried out on image data, and in the future, we would like to explore other types of systematic errors and data types (tabular, sensor, natural language).

4.7.1 Advantages and Disadvantages

In summary, a list of ***advantages*** related to this chapter and the proposed model (CLSVAE) is given below:

- ***Labels are easier to obtain and more broadly applicable.*** In terms of effort by the user encoding prior knowledge into the model using labels is far easier than using scripts or logic rules. Scripts require the user to know how to computer program. Similar with logic rules as these are often defined using first-order logic, thus require skill. Moreover, labelling inliers and outliers should be applicable to all kinds of data – e.g. image, tabular, or even time-series data. In CLSVAE the user only needs to be able to label all type of errors it wants removed, and a few inliers.
- ***Less labelled data is needed for good data repair.*** It was shown that CLSVAE compared to other semi-supervised generative models needed less labelled inliers and outliers for similar data repair performance. This is important because user intervention, even in the form of labelling data instances, still takes time and effort.
- ***Semi-supervision allows for precise detection and repair.*** Models using semi-supervision like CLSVAE allow for the detection and repair of only those types of outliers that the user wants fixed. Since unsupervised models cannot easily encode this information given by the user, there is no guarantee those types of outliers will be fixed. Furthermore, increasing the strength of regularization in order to detect these outliers may lead to poor repair performance later on. This is a drawback that semi-supervised models can avoid far more easily.
- ***Deep learning provides flexibility.*** CLSVAE is a deep autoencoder, and thus the neural architectures of the encoders and decoders can be adapted to tackle different data types, e.g. image or tabular data. Further, different architectures like convolutional neural networks, or Transformer layers could

be explored.

- ***Representation learning for clean data.*** The CLSVAE model learns a latent subspace that only encodes information about the clean patterns present in data instances. This latent representation could be used later on for downstream tasks (e.g. classification, clustering). The main advantage is that this latent representation will not be corrupted by errors. Therefore, it will not taint the downstream task model training.
- ***Thresholds for traditional outlier detection easier to set.*** An advantage of the semi-supervised formulation of CLSVAE, like other semi-supervised VAEs, is that outlier detection is done using a classifier – e.g. $q_\phi(y|\boldsymbol{x})$. Generally, deep classifiers are more or less calibrated and thus threshold setting by the user should be easier. Hence a good starting point is about $\gamma \approx -\log(0.5)$.

Some relevant ***disadvantages*** are listed below:

- ***Complexity of deep learning architectures.*** Deep learning models offer flexibility and increased performance in many tasks. However, one issue about deep learning models like CLSVAE is that one needs to find the correct neural architecture and tune several hyperparameters. This means time and effort on the part of the user.
- ***Exploring tabular datasets.*** Given time constrains on this project only image datasets were explored. For future work tabular data is left as a goal. We believe that CLSVAE should be able to be applied for tabular data. Given the experience with RVAE (Chapter 3), we believe that similar datasets with synthetic systematic errors are reasonable experiments. The RVAE neural architecture for the encoder and decoder is also a good start; and we believe it should work with the current CLSVAE formulation and training loss. If that is not good enough, then we can start looking into self-attention layers for tabular data, where only a partition of the cells are selected. This would work like masks such that for the clean subspace $\boldsymbol{z}_c$ the encoder looks at the clean cells. Concurrently, for the dirty pattern subspace $\boldsymbol{z}_d$ attends to the dirty cells. Other neural architectures are also possible, and we can borrow from other deep generative models for tabular data (see Section 2.4.1).

- ***Downsides of repair using just the latent space.*** In this chapter, the repair process for CLSVAE produced a repair for the entire repaired image (all pixels). This is seen in the qualitative examples presented. It should be noted that for the repair results using the SMSE metric, we partitioned the results into clean pixels and dirty pixels. The repair process in CLSVAE relies on a latent space representation, i.e. clean subspace $\boldsymbol{z}_c$. There is the possibility that small changes to the latent space may affect all pixels (features), even if only a few are dirty. Therefore, in practice, we can use CLSVAE to predict a mask of pixels that are considered dirty (outliers). One can use the anomaly score defined in eq. (2.10) for standard VAEs to obtain a mask. Then only the dirty pixels need to be repaired by CLSVAE, whilst the rest remain unchanged.

- ***Testing on more complex errors and real-world data.*** As pointed out in Section 4.6, often prior research on data repair, or machine learning robust to corruption, has been carried out using synthetic error injection (Liu et al., 2020; Krishnan et al., 2016). For instance, in (Krishnan et al., 2016; Wang et al., 2017b) use similar corruptions (masking out pixels), but not for systematic error repair. The main reason for this is precisely the lack of easily accessible public datasets for the data repair task, where for proper evaluation one needs the following targets: (a) labelling of all errors, either anomalous pixels or instances; (b) the ground-truth repairs, i.e. underlying inliers. This is especially frustrating given that data cleaning (outlier detection and repair) is quite common in machine learning pipelines. Though these errors are synthetic, we believe them to be as hard to repair as several real examples. Particularly given that baseline models seem to have struggled in these scenarios, and some of them are SOTA models.

  Similarly, we can look at the related task of blind-inpainting in images (Elharrouss et al., 2020; Jam et al., 2021). The goal of blind-inpaiting is to estimate which pixels are corrupted, or missing, and then infer the value of those pixels. However, unlike our setup of systematic errors, almost all of the models for blind-inpainting are trained using clean (uncorrupted) datasets (Elharrouss et al., 2020; Jam et al., 2021; Dehaene et al., 2019), a very important difference. It is very common to have experiments mostly consisting of synthetic errors (Elharrouss et al., 2020; Jam et al., 2021),

likely due to the lack of curated datasets for research. Specifically, we see that corruption related to: 1) image coding or transmission often involves errors as square blocks; 2) image restoration often involves inserting lines to emulate scratches, camera sensors failing, or watermarks; 3) other mask shapes can also be used to emulate object removal or occlusion, where often lines and geometric shapes are used.

Having said that, a *valid criticism* is that more complex systematic errors could have been tried, or that real-world examples could have been found. Unfortunately there was not enough time to pursue this in this chapter, hence we leave it as future work. Some examples of more complex systematic errors are: inserting patches of blurred pixels in images in the same location; scramble patches of pixels in the same exact way for several instances; add text on top of the images (e.g. watermarks); take a patch of an image and insert it in the same position for several images in the dataset. In terms of real-world examples, we can try and curate data from medical imaging where systematic errors seem to be a problem; or image datasets with real-world watermarks. In section 4.7.2 we briefly discuss some application scenarios as well.

- ***Not all systematic errors may be repaired by CLSVAE.*** In order to repair dirty pixels CLSVAE relies on the context given by clean pixels in a corrupted image. This means a portion of the pixels needs to be clean or recognizable enough such that CLSVAE can infer the underlying inlier structure of the image. However, some systematic errors may affect the entire image and distort all pixels in an image. In this case it is quite possible that CLSVAE may struggle to repair the data instance. If this happens, then one solution might to modify CLSVAE to use a different type of supervision. For instance, one can use paired data instances of the underlying inlier and corresponding outlier image. It is also possible to modify the generative prior distributions to cater to specifically difficult systematic errors. Generally, it will be hard to find one model that without tailoring it will repair all kinds of systematic errors, for all types of data (e.g. image, tabular, time-series, natural language).

4.7.2 Potential Real-World Applications

Medical imaging distortions (e.g. MRI, RX), restoration of images, or resolving issues related to bad image coding and transmission are possible applications. For instance, in medical imaging it is not uncommon to find corruptions that are: 1) similar to occlusion (by objects) as a result of dental or metal implants; 2) imaging sensor corruption (e.g. lines or objects) due to artifacts or sensor failure. We also believe that our model can be extended to tackle tabular data, where systematic errors often appear as data entry issues; data format or unit of measurement issues from data source merging; or problems with data transmission.

Chapter 5

Conclusion and Future Work

In this thesis, the problem of data cleaning was tackled from the point of view of combining the steps of outlier detection and data repair. The main reason being that in a lot of applications the practitioner that is using outlier detection methods also would want to repair the data later on. In this case, outliers are a result of corruption that may have its origin in a variety of different processes. Further, it is often the case that the practitioner needs help identifying the dirty pixels or cells that make the instance an outlier – i.e. interpretability.

Deep generative models were chosen for their ability to reconstruct and sample data, which is necessary for data repair. Deep generative models make use of neural networks as function approximators making them extremely flexible in terms of capturing complex data distributions. Moreover, they can be quickly adapted to several data types, such as image data or tabular data. Recently deep learning models have started to make their way into outlier detection, often registering SOTA performance.

VAE models were preferred for their simplicity in terms of implementation and ease of training. VAEs and GANs are very popular both in academia and industry. However, GANs can be quite unstable in terms of their training and performance. Therefore, VAEs were picked for this work as a first incursion into deep generative models that are *robust* to corruption in the data. Furthermore, the VAE is a reconstruction-based model and thus it allows for the granularity of cell anomaly scores. So VAEs can be used to find which dirty cells are to blame for the outlier.

We have tackled the problem of combined outlier detection and data repair for

two important types of corruption found in real-world data: *random* errors and *systematic* errors. The impact these errors have on generative models is also different, and thus different strategies of error robustness were devised.

Unsupervised models using data reweighting can handle random error corruption up to moderate amounts without overfitting to these. This thesis has proposed a novel data reweighting VAE for mixed-type tabular data, the *Robust Variational Autoencoder* (RVAE). The model outperformed or matched baseline model performance in outlier detection (cell and row) and in data repair.

On the other hand, this unsupervised paradigm is not enough to guarantee good performance in the case of systematic errors. Systematic errors are a result of nearly deterministic transformations (plus potentially some noise) that repeatedly corrupt data instances. The result is a type of corruption that in enough amounts is very easy to overfit to even for unsupervised robust models. This thesis therefore presents a novel semi-supervised VAE that learns separate latent codes for clean patterns and the systematic errors. Only the clean patterns are needed when reconstructing the underlying inlier (repair). This model is the *Clean Subspace Variational Autoencoder* (CLSVAE). The model showed superior performance compared to baselines, particularly when the labelled set provided is very small.

5.1 Using RVAE and CLSVAE in Practice

In practice, when using either RVAE or CLSVAE, the user needs to have the needed computational resources at hand. These are deep generative models and thus they are more computationally demanding compared to classic methods. This probably includes GPUs, or even servers of these when searching for architectures or hyperparameters. The user also needs to have some intuition or know already that the dataset is corrupted and needs to be repaired. This is something that can be discovered through initial data exploration, using frameworks for that purpose like OpenRefine, Trifacta or simple Jupyter notebooks with data visualization packages.

At this point the user already has an idea on what type of corruption is found therein. If it is mostly random errors than RVAE should be selected. Otherwise, if the user only wants to repair a few systematic errors found during data exploration, then CLSVAE should be used. In this case, the user needs to have collected a small

labelled set (trusted set) of inliers and more importantly examples of systematic errors. This can be done during the data visualization and exploration phase. Alternatively, the user can choose to always apply RVAE first to repair general random error corruption. Then by inspection, or using a labelled set for validation, conclude whether the repair process was successful. If it failed, then adjust the model hyperparameters. If it was successful, then one can now focus on repairing the remaining systematic errors using CLSVAE, and then rechecking the quality of the repair.

Throughout the process of cleaning the data thresholds have to be set for each anomaly score used. This is important for the outlier detection process, as it determines which instances are considered outliers. In the case of CLSVAE, if using the classifier therein, it is always good to start with a threshold around $\gamma \approx -\log(0.5)$ and then make necessary adjustments. This is because the neural classifier tends to be more or less calibrated. For RVAE when setting thresholds for cell outlier detection a similar approach should be taken, and tunning can be done independently for each feature. Threshold setting is always made easier if a labelled set of inliers and outliers is available. These can be smaller than typical validation sets, and can be obtained during data exploration.

Sometimes the goal of data cleaning is to use the repaired data for downstream tasks. Since RVAE and CLSVAE learn latent representations of clean data, then downstream task methods could use these for training. This may lead to better performance than using the repaired data directly. Alternatively, it is also possible to jointly train the RVAE or CLSVAE and the downstream task model. This could lead to improved performance not only on the downstream task itself, but also for the data repair task as well. Since the downstream task can function as an *inductive bias* for the data repair process provided by RVAE or CLSVAE.

More specifically, the most typical case is a downstream task that is defined by a supervised machine learning problem. This means that there is a predictor model that needs to be learnt on the corrupted dataset to predict some label t. Most applications are classification or regression problems. Typically, joint training of a predictor model and a generative model (e.g. RVAE or CLSVAE) is done by: *i)* adding to the current generative model training loss an extra training loss for the downstream predictor model; *ii)* chain together the generative model and the predictor model, either by reusing the latent representation $\boldsymbol{z}$ or sampling

repaired $\boldsymbol{x}$ from the generative model as input to the predictor to estimate t. Since these models are trained using automatic differentiation frameworks the gradients of the predictor model will impact the parameters of generative model. This assumes that one can differentiate through $\boldsymbol{x}$ or $\boldsymbol{z}$, as defined in *ii)*, and thus *backpropagation* obtains the gradients of the predictor loss w.r.t generative model parameters. In literature, this type of end-to-end training of the models is commonly designated as an *hybrid model* (Nalisnick et al., 2019). It is thus evident that the downstream model will influence how CLSVAE or RVAE does data repair.

Still, some improvements can be made to the *hybrid model* above. For instance, in some cases the label t is also corrupted in the dataset. For classification tasks, one option is to apply *label smoothing* (Müller et al., 2019) to the predictor loss. This introduces noise or uncertainty to the labels used for training, which will downweight the contribution of dirty labels. Another option is to estimate the probability of t being clean, and downweight its contribution if dirty. This can be done in a similar fashion to how RVAE downweighs dirty cell contributions, e.g. use a π_t and an outlier model for t. On the other hand, including the label t in the encoder and/or decoder model may also help in capturing outlier information. For instance, one could concatenate label t to the latent code of CLSVAE at the decoder input, since the labels could present systematic errors as well. In terms of reusing the latent space of CLSVAE, typically if the predictor model requires a clean dataset for training, then $\boldsymbol{z}_c$ should be used. Although, using the entire latent space ($\boldsymbol{z}_c$ and $\boldsymbol{z}_d$) can provide the predictor with more information in case label t is corrupted. Additionally, it may allow for the predictor model to better capture uncertainty about the instance being predicted. Note that in CLSVAE the code $\boldsymbol{z}_c$ captures inlier information, whilst $\boldsymbol{z}_d$ captures error information.

Moreover, the RVAE decoder defined by the generative model in eq. (3.2) should be able to be used in more complex VAEs. It should not matter if these architectures are based on Transformers (Vaswani et al., 2017) or ResNets (He et al., 2016), or have more complex inference schemes like VQVAEs (Razavi et al., 2019b), since the ELBO of a VAE should always have a reconstruction loss. Further, the coordinate ascent optimization scheme of RVAE should be applicable as well. It is possible that the outlier model would need to be adapted in some applications, or for some data types like counts – e.g. Poisson distribution.

Lastly, the CLSVAE approach should be able to be applied to more complex VAE architectures as well. This should be viable as long as it is possible to segregate the latent space. However, use of more complex inference schemes should always be aware of the *distance correlation* penalty as an optimization constrain (see eq. 4.14). For instance, one could potentially apply the latent segregation of CLSVAE – i.e. clean and dirty subspaces, see eq. (4.4) – to each latent layer of an Hierarchical VAE (Zhao et al., 2017b). This would allow more control on the level of feature abstraction in terms of error patterns one would want to remove. Particularly, we would find simpler error patterns at lower layers, and more complex errors at higher layers. Alternatively, we can just apply at higher abstraction layers near the VAE bottleneck, if computation complexity is an issue.

5.2 Data Benchmarks and Frameworks

In the data science pipeline the data cleaning process is quite important. It is often a necessary step so that machine learning models can be trained and deployed correctly, but also so that the data can be stored for further applications. Two steps often applied together in the data cleaning process are outlier detection, followed by data repair. The problem combined outlier detection and data repair is thus an important one.

One issue we have had during the work in this thesis is the lack of frameworks and dataset benchmarks. Most outlier detection benchmark datasets only focus on labelled instances, and never cells or pixels – e.g. (McHugh, 2000, KDD 99), (Keller et al., 2012, Thyroid), (Liu et al., 2008, Satellite), (Ruff et al., 2021, see Table III), benchmark framework (Emmott et al., 2015), and benchmark repository (Rayana, 2016). This is important since quite a few outliers are due to corruption, and thus outliers are caused by specific features being affected. On the other hand, from a data repair perspective, very few datasets provide the actual ground-truth inlier corresponding to an outlier. Most often published works rely on synthetic corruption processes, often specific for that paper in question. This makes model benchmarking more difficult across literature.

The machine learning and database communities should therefore come together and define a set of agreed upon dataset benchmarks that can be used for model development. Alternatively, the communities could promote and jointly develop

a software framework for synthetic dataset corruption. Frameworks for data synthesis in tabular or image data already exist, so that could be a starting point for one focusing on corrupt data. This will help researchers in the field greatly, and in the end benefit the practitioner with new and improved models.

5.3 Going Forward on Robust Generative Models

There are a few potential paths forward for novel robust deep generative models. Here a few of these options are discussed more abstractly.

One way is leveraging multi-modal data in novel generative models capable of outlier detection and repair. The idea is that one modality of the data can help in repairing the other, and vice-versa. As an example, many real-world datasets include instances that have a part being a row from a table and then an image of an object. Two use cases seem relevant: electronic retailer (e.g. Amazon, eBay) database with product descriptions in a table, and then images of said products; secondly, in healthcare datasets often include medical imaging and other medical tests in tables for each patient.

In this thesis we focused on using labelled data when tackling systematic errors. This is because it is easier during data exploration to obtain a few labelled instances, than to obtain logic rules or data cleaning scripts. Having said that, sometimes rules or some other templates could be sourced together with the data, or perhaps by the use of paid subject-matter experts. In this scenario, it makes sense to have a deep generative model that can encode this information into the model as prior knowledge. This inductive bias would then be used to make the model robust to outliers during training. An interesting starting point could be to combine either RVAE or CLSVAE with differentiable relaxations of first-order logic, which would express the data quality constraints.

We mainly focused on MAP estimation of repairs using VAEs. Although in Chapter 3 we provided an alternative using pseudo-Gibbs sampling (MCMC) for RVAE, where we only saw very marginal improvements. It is quite possible that other types of inference methods or generative models (e.g. GANs, normalizing flows) could prove superior. For instance semi-amortized VAEs (Kim et al., 2018) could potentially enhance the quality of the inferred repair for VAEs. Energy-based models tend to use MCMC sampling during training and inference, and are

capable of high-fidelity image inpaiting (Du & Mordatch, 2019; Dehaene et al., 2019).

Nowadays the success of diffusion models is apparent in the field of deep generative models – see Section 2.4 for more details. Although most works have focused on image data, more recently tabular data has also been explored (Kotelnikov et al., 2022). There might be a more clear extension of diffusion models for outlier detection and data repair in the case of systematic error corruption. Since the impact of random errors is yet to be properly evaluated in diffusion models.

For instance, one could define a conditional diffusion model that uses a trusted set for systematic error repair – like CLSVAE in Chapter 4. One option is to use a *classifier guided diffusion model* (Dhariwal & Nichol, 2021), where a classifier is first trained on the trusted set (noised with the forward diffusion process) to learn to distinguish inliers from outliers. We also assume one has access to a diffusion model trained on the corrupted dataset. The *diffusion sampling process* of the model will explicitly include gradient information from the classifier in order to condition on y. Therefore, at repair time, the outlier instances can be repaired by setting $y = 1$ (inlier) in the sampling process, after the forward diffusion process has been applied to the instance. Alternatively, and perhaps a better performing solution, is to use *classifier-free guidance* (GLIDE) (Nichol et al., 2021) as the conditional diffusion model. In this case labels y are used during training of the diffusion model. Though GLIDE is a supervised model, perhaps modifications can be made to make it semi-supervised for use with a trusted set.

One could use a contrastive learning approach (Chen et al., 2020; Zbontar et al., 2021) for outlier detection and data repair. Contrastive learning has gained popularity recently due to being quite effective in representation learning. For instance, it can be used if there is a dataset of paired inliers and outliers already; or perhaps if the corruption process can be accessed (e.g. simulated, data augmentation) to get these pairs. One use case could be datasets corrupted with systematic errors. In this case, then one can use contrastive learning to learn a representation in an embedding space where underlying inliers and their corrupted versions (outliers) are close together. If the task is data repair, then these learned representations can be reused to train a generative model like a VAE. In this case, the pre-trained encoder from the contrastive learning model would be used to train an additional generator (decoder) network that repairs the data. In terms of outlier detection,

one could potentially use the aforementioned generator to get an anomaly score. On the other hand, these learned representations can also be directly reused to train a downstream task model that requires repaired data (or embeddings). Inspired by Ren et al., one could also use contrastive learning directly to learn a generative model that disentangles in the latent space inliers from error patterns (outliers). Like CLSVAE, this model could then be used for both outlier detection and repair.

5.4 An Outlook of the Problem in 2022

In this section a short discussion is provided about tackling the thesis problem in 2022. Once more, in this thesis we have focused on specific data cleaning tasks. In particular, outlier detection and subsequent data repair. Still, we will also try to give a broader outlook on machine learning (ML) for data cleaning.

If one were to start in 2022, given existing work, then one should first focus on promoting with other researchers (or industry) a common benchmark or framework for evaluation. As previously discussed, it is often difficult to compare models for outlier detection or data repair since they use different synthetic corruption processes, or even datasets. Standardizing evaluation metrics would also be important. Additionally, interpretability for outlier detection and cell outlier detection are still quite relevant today. Further, much work is still needed on improving generative models for outlier detection and subsequent data repair. Interesting directions are perhaps other model types like diffusion models, or new inference schemes in VAE models for better repair quality.

Now we discuss some specific ongoing opportunities or research directions in machine learning for data cleaning.

- ***Robust Generative Modelling for Data Cleaning*** Recently *diffusion models* have been registering state-of-the-art performance in image datasets (see Section 2.4). Moreover, there are ongoing efforts to extend these to tabular data (Dhariwal & Nichol, 2021). Thus interesting directions would be diffusion models that are robust to outliers. Alternatively, there are still opportunities to improve inference schemes in robust VAEs for tabular data. Particularly, new inference schemes going beyond MAP estimation of data repair. For instance, extending or modifying MCMC methods for

VAEs that are robust to outliers. A recent example in the related field of data imputation (Peis et al.) is using Hamiltonian Monte Carlo (HMC) in Hierarchical VAEs for tabular data. This strategy could be adapted for robust VAEs. Transformer (Vaswani et al., 2017) based architectures for tabular data have become more relevant, even in outlier detection (Liu et al., 2020). There is space for improvement still. Moreover, generative models that take advantage of concepts like data instance memorization (in neural networks) (Arpit et al., 2017; Feldman & Zhang, 2020), or data instance influence on model parameters (Koh & Liang, 2017; Hara et al., 2019) are also good research directions. Particularly, it can prove useful in outlier detection, and in downweighting said outliers during training, thus providing robustness. More discussion about these concepts (memorization and influence) can be seen in Section 2.2.

- ***ML for Multi-modal Data Cleaning*** Quite a few datasets in the real-world are a combination of structured data (e.g. tabular data) with unstructured data (e.g. images, text). These types of datasets are often called multi-modal, since they combine different modes of data. The example used in Section 5.3 was that of an online retailer with a product catalogue. But there are many other examples in industry. Therefore, a trend that will continue to grow is deep learning models (e.g. Transformers (Vaswani et al., 2017)) that perform data cleaning in multi-modal datasets. For instance, extending current robust generative models to handle both tabular data and text.

- ***Data Cleaning for Downstream Tasks*** There should be continued interest in ML models that adapt outlier detection or data repair processes to downstream tasks of interest. For instance, generative models for data repair that are jointly trained with the downstream task model – i.e. *hybrid models* (Nalisnick et al., 2019). Often data cleaning can be quite specific depending on the machine learning model being trained down the line. Further, the downstream task can bias the data cleaning procedure in order to improve its performance. This makes sense as practitioners often work on entire machine learning pipelines, and usually there is some dependency between the different pipeline steps. Going a step further, it is possible that *automated machine learning* (AutoML) (Karmaker et al., 2021, Table 1.) for

data cleaning that takes into consideration downstream task performance is a better solution. The AutoML model would combine typical data cleaning procedures and user defined ones, e.g. (Krishnan et al., 2016). Therefore, this joint perspective should continue to grow in importance.

- ***Weak-Supervision or User Interaction for Data Cleaning*** In machine learning, often labelled datasets are expensive or impractical to obtain, even if tools like crowdsourcing are available. This is particularly true for outlier detection, but also other steps in data cleaning. The data cleaning step is often a very bespoke process that takes into consideration particular datasets or applications. As such, user input or interaction is often a must. Semi-supervised models that handle small trusted sets are one option, however in some cases this might not be enough to get good performance. Another option is to estimate which instances are most relevant and ask user feedback, i.e. *active learning* (Ren et al., 2021). For instance, this can be a way to make robust generative models more adaptable to user needs. On the other hand, one can use predefined programs or patterns to label several instances before training an outlier detection / data repair model, i.e. *weak-supervision* (Ratner et al., 2017; Zhang et al., 2022; Shin et al., 2021). Further, one could embed such prior knowledge directly in the model (Rekatsinas et al., 2017; Rühling Cachay et al., 2021; Lew et al., 2021). Examples of weak-supervision are heuristics, logic rules, scripts, ontologies and even other simpler models. There should be continued interest in novel machine learning models for data cleaning that take direct user feedback.

- ***Frameworks for Evaluating Data Cleaning*** Once more, as mentioned in Section 5.2, developing new software frameworks for benchmarking data cleaning algorithms should be a focus of the research community. This will prove particularly useful in outlier detection and in data repair tasks. But perhaps can be extended to other data cleaning tasks, e.g. data imputation or de-duplication. However, it is not clear whether the community will actually converge on this, despite a clear opportunity and need. For instance, similar efforts have been carried out for tabular data synthesis, e.g. *SDGym*[1]. SDGym framework defines datasets, synthesizers (i.e. generative model baselines), and evaluation (e.g. data quality metrics, privacy metrics,

[1] `https://github.com/sdv-dev/SDGym`

computation time). Another example is *Cleanlab*[2] a software framework for cleaning noisy labels, where some support is given to benchmarking and standardization of synthetic noise injection for labels. The framework also provides a strong baseline, and curates a database of known noisy labels in machine learning datasets – *Label Errors in ML Test Sets*[3]. Alternatively, one can use existing online frameworks like *OpenML*[4] to share benchmarks, datasets, models and metrics for data cleaning tasks. Perhaps the aforementioned frameworks could serve as inspiration for the ML for data cleaning community.

- ***Data Cleaning under Privacy Constraints*** In the last few years data privacy has become quite important in machine learning, both in terms of model training and model deployment. For example, the field of *differential privacy* (Dwork et al., 2014) in machine learning has grown spectacularly. However, the same privacy constraints still exist in the data cleaning step of the machine learning pipeline. One option is to generate synthetic data that is considered safe, i.e. complies with privacy constraints, and then data cleaning tasks are applied. For instance, synthetic data can be obtained by sampling a generative model (Zhang et al., 2017; Jordon et al., 2019) that complies with the definition of differential privacy. But this might not be an ideal arrangement, and may result in loss of data quality for downstream tasks. Another option is to develop a generative model that adheres to privacy constraints, and performs the much needed data repair. So there should be ample opportunities for researchers in this space.

- ***Data Cleaning in Graph Datasets*** Until recently, the most popular type of structured dataset in industry was tabular data. However, graph structured datasets have become common in many companies like online retailers, social networks, pharmaceuticals or even in financial data. Moreover, there has been considerable research in the last few years in machine learning models for graph datasets (Xia et al., 2021). Particularly in deep learning (Gilmer et al., 2017; Waleffe et al., 2022). Still, very few works have explored ML for data cleaning in the presence of graph datasets (Heidari et al., 2020). Therefore, there should be ample interest and opportunities in

[2] `https://github.com/cleanlab/cleanlab`
[3] `https://labelerrors.com/`
[4] `https://www.openml.org/`

this line of research.

- ***Fairness in Data Cleaning*** Machine learning *fairness* (Caton & Haas, 2020) is concerned with correcting or mitigating algorithmic biases (e.g. nationality, gender, disability) in machine learning models. It is quite possible that the current data repair models may produce repairs that are *unfair*, since they might be based on data features considered *sensitive* by society. This analysis can be further extended to both outlier detection, and the related field of data imputation. Hence, there are clear opportunities to study *fairness* in the field of data cleaning. Firstly, researchers should assess if current models produce biased repairs, and which biases need to be corrected. Secondly, researchers should propose novel models that produce unbiased data repairs, where user input can be accounted for.

Bibliography

Abubakar Abid, Muhammad Fatih Balin, and James Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346*, 2019.

Charu C Aggarwal. Outlier analysis second edition, 2016.

S Aigrain, H Parviainen, S Roberts, S Reece, and T Evans. Robust, open-source removal of systematics in kepler data. *Monthly Notices of the Royal Astronomical Society*, 471(1):759–769, 2017.

Yariv Aizenbud, Ofir Lindenbaum, and Yuval Kluger. Probabilistic robust autoencoders for anomaly detection. *arXiv preprint arXiv:2110.00494*, 2021.

Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29 (3):626–688, 2015.

H. Akrami, Anand A. Joshi, J. Li, and R. Leahy. Robust variational autoencoder. *ArXiv*, abs/1905.09961, 2019a.

Haleh Akrami, Anand A. Joshi, Jian Li, and Richard M. Leahy. Robust variational autoencoder, 2019b.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168. PMLR, 2018.

Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2:1–18, 2015.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville,

Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002.

Chenglong Bao, Jian-Feng Cai, and Hui Ji. Fast sparsity-based orthogonal dictionary learning for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3384–3391, 2013.

Jonathan T Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4331–4339, 2019.

Stephen D Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 29–38, 2003.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction

method of multipliers. *Foundations and Trends® in Machine learning*, 3(1): 1–122, 2011.

Gustav Bredell, Kyriakos Flouris, Krishna Chaitanya, Ertunc Erdil, and Ender Konukoglu. Explicitly minimizing the blur error of variational autoencoders. In *Submission ICLR 2023*, 2022.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Coleman Broaddus, Alexander Krull, Martin Weigert, Uwe Schmidt, and Gene Myers. Removing structured noise with self-supervised blind-spot networks. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pp. 159–163. IEEE, 2020.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Yuri Burda, Roger B Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR (Poster)*, 2016.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

Miguel A Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pp. 33–40. PMLR, 2005.

Simon Caton and Christian Haas. Fairness in machine learning: A survey. *arXiv preprint arXiv:2010.04053*, 2020.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018a.

Raghavendra Chalapathy, Edward Toth, and Sanjay Chawla. Group anomaly detection using deep generative models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 173–189. Springer, 2018b.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. *arXiv preprint arXiv:2202.04200*, 2022.

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Sanjay Chawla and Aristides Gionis. k-means–: A unified approach to clustering and outlier detection. In *Proceedings of the 2013 SIAM international conference on data mining*, pp. 189–197. SIAM, 2013.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In *NIPS*, 2006.

Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Yanzhi Chen, Dinghuai Zhang, Michael U. Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In *Ninth International Conference on Learning Representations (ICLR 2021)*, May 2021. URL `https://iclr.cc/Conferences/2021/Dates`. Ninth International Conference on Learning Representations 2021, ICLR 2021 ; Conference date: 04-05-2021 Through 07-05-2021.

Rewon Child. Very deep vaes generalize autoregressive models and can outperform

them on images. In *International Conference on Learning Representations*, 2020.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8185–8194, 2020.

Xu Chu, Ihab F Ilyas, and Paolo Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013a.

Xu Chu, Ihab F Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 458–469. IEEE, 2013b.

Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1247–1261, 2015.

Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 international conference on management of data*, pp. 2201–2206, 2016.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4): 495–508, 1980.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Bin Dai, Ziyu Wang, and David Wipf. The usual suspects? reassessing blame for vae posterior collapse. In *International Conference on Machine Learning*, pp. 2313–2322. PMLR, 2020.

Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. Nadeef: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 541–552, 2013.

Robert B Dean and William J Dixon. Simplified statistics for small numbers of observations. *Analytical chemistry*, 23(4):636–638, 1951.

David Dehaene, Oriel Frigo, Sébastien Combrexelle, and Pierre Eline. Iterative energy-based projection on a normal data manifold for anomaly localization. In *International Conference on Learning Representations*, 2019.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Ilias Diakonikolas, Gautam Kamath, Daniel M Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. *arXiv preprint arXiv:1803.02815*, 2018.

Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407, 2014.

Simao Eduardo, Alfredo Nazábal, Christopher KI Williams, and Charles Sutton. Robust variational autoencoders for outlier detection and repair of mixed-type

data. In *International Conference on Artificial Intelligence and Statistics*, pp. 4056–4066. PMLR, 2020.

Omar Elharrouss, Noor Almaadeed, Somaya Al-Maadeed, and Younes Akbari. Image inpainting: A review. *Neural Processing Letters*, 51(2):2007–2028, 2020.

Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158*, 2015.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2525–2534. PMLR, 2019.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111: 98–136, 2014.

Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Wenfei Fan. Data quality: From theory to practice. *Acm Sigmod Record*, 44(3): 7–18, 2015.

Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Transactions on Database Systems (TODS)*, 33(2):1–48, 2008.

Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering*, 23(5):683–698, 2010.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and

why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Diego Fernández-Francos, Óscar Fontenla-Romero, and Amparo Alonso-Betanzos. One-class convex hull-based algorithm for classification in distributed environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(2): 386–396, 2017.

Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, volume 99, pp. 1300–1309, 1999.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Çelikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *NAACL*, 2019.

Futoshi Futami, Issei Sato, and Masashi Sugiyama. Variational inference based on robust divergences. In *AISTATS*, 2018.

M. J. F. Gales and Peder A. Olsen. Tail distribution modelling using the richter and power exponential distributions. In *Sixth European Conference on Speech Communication and Technology, EUROSPEECH 1999, Budapest, Hungary, September 5-9, 1999*. ISCA, 1999a. URL `http://www.isca-speech.org/archive/eurospeech_1999/e99_1507.html`.

Mark John Francis Gales and Peder A. Olsen. Tail distribution modelling using the richter and power exponential distributions. In *EUROSPEECH*, 1999b.

Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. The llunatic data-cleaning framework. *Proceedings of the VLDB Endowment*, 6(9): 625–636, 2013.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889. PMLR, 2015.

Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard

Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

Amol Ghoting, Srinivasan Parthasarathy, and Matthew Eric Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16(3):349–364, 2008.

Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. Tikhonov regularization and total least squares. *SIAM journal on matrix analysis and applications*, 21(1):185–194, 1999.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013.

Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, 26(9):2250–2267, 2013.

Hermanni Hälvä, Sylvain Le Corff, Luc Lehéricy, Jonathan So, Yongjie Zhu, Elisabeth Gassiat, and Aapo Hyvarinen. Disentangling identifiable features from noisy data with structured nonlinear ica. *Advances in Neural Information Processing Systems*, 34, 2021.

Hannes Hapke and Catherine Nelson. *Building machine learning pipelines*. O'Reilly Media, 2020.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. Data cleansing for models trained with sgd. *Advances in Neural Information Processing Systems*, 32, 2019.

Douglas M Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

Louay Hazami, Rayhane Mama, and Ragavan Thurairatnam. Efficient-vdvae: Less is more. *arXiv preprint arXiv:2203.13751*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1(Oct): 49–75, 2000.

Alireza Heidari, Ihab F Ilyas, and Theodoros Rekatsinas. Approximate inference in structured instances with noisy categorical observations. In *Uncertainty in Artificial Intelligence*, pp. 412–421. PMLR, 2020.

Joseph M Hellerstein. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)*, 25:1–42, 2008.

Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, abs/1807.01697, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, abs/1610.02136, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. *International Conference on Learning Representations (ICLR)*, abs/1812.04606, 2019.

Shohei Hido, Yuta Tsuboi, Hisashi Kashima, Masashi Sugiyama, and Takafumi Kanamori. Statistical outlier detection using direct density ratio estimation. *Knowledge and Information Systems*, 26(2):309–336, 2011.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pp. 2722–2730. PMLR, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. *Advances in neural information processing systems*, 30, 2017.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pp. 492–518. Springer, 1992.

Peter J Huber. *Robust statistics*, volume 523. John Wiley & Sons, 2004.

Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in Neural Information Processing Systems*, 29, 2016.

Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pp. 322–348. PMLR, 2020.

Ihab F Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

Ihab F Ilyas and Xu Chu. *Data cleaning*. Morgan & Claypool, 2019.

Ihab F Ilyas and Theodoros Rekatsinas. Machine learning and data cleaning: Which serves the other? *ACM Journal of Data and Information Quality (JDIQ)*, 2020.

Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. not-miwae: Deep generative modelling with missing not at random data. In *International Conference on Learning Representations*, 2020.

Abdul Jabbar, Xi Li, and Bourahla Omar. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)*, 54:1 – 49, 2022.

Jireh Jam, Connah Kendrick, Kevin Walker, Vincent Drouard, Jison Gee-Sern Hsu, and Moi Hoon Yap. A comprehensive review of past and present image inpainting methods. *Computer vision and image understanding*, 203:103147, 2021.

Bowen Jing, Gabriele Corso, Renato Berlinghieri, and Tommi Jaakkola. Subspace diffusion generative models. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pp. 274–289. Springer, 2022.

Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2019.

Tom Joy, Sebastian Schmon, Philip Torr, N Siddharth, and Tom Rainforth.

Capturing label characteristics in vaes. In *International Conference on Learning Representations*, 2020.

Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. Distribution augmentation for generative modeling. In *International Conference on Machine Learning*, pp. 5006–5019. PMLR, 2020.

Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *ACM Human Factors in Computing Systems (CHI)*, 2011. URL `http://vis.stanford.edu/papers/wrangler`.

Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)*, 54(8):1–36, 2021.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in neural information processing systems*, 33:12104–12114, 2020.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

Gregor Kasieczka, Benjamin Nachman, and David Shih. New methods and datasets for group anomaly detection from fundamental physics. *arXiv preprint arXiv:2107.02821*, 2021.

Fabian Keller, Emmanuel Muller, and Klemens Bohm. Hics: High contrast subspaces for density-based outlier ranking. In *2012 IEEE 28th international conference on data engineering*, pp. 1037–1048. IEEE, 2012.

Wesley Khademi, Sonia Rao, Clare Minnerath, Guy Hagen, and Jonathan Ventura. Self-supervised poisson-gaussian denoising. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2131–2139, 2021.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pp. 2207–2217. PMLR, 2020.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pp. 2649–2658. PMLR, 2018.

JooSeuk Kim and Clayton D Scott. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1):2529–2565, 2012.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pp. 2678–2687. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Guenter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, 2017.

Jack Klys, J. Snell, and R. Zemel. Learning latent subspaces in variational autoencoders. In *NeurIPS*, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Pang Wei Koh, Jacob Steinhardt, and Percy Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2018.

Solmaz Kolahi and Laks VS Lakshmanan. On approximating optimum repairs

for functional dependency violations. In *Proceedings of the 12th International Conference on Database Theory*, pp. 53–62, 2009.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. *arXiv preprint arXiv:2209.15421*, 2022.

Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J Franklin, and Ken Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, 9(12):948–959, 2016.

Sanjay Krishnan, Michael J Franklin, Ken Goldberg, and Eugene Wu. Boostclean: Automated error detection and repair for machine learning. *arXiv preprint arXiv:1711.01299*, 2017.

Jonathan Kropko, Ben Goodrich, Andrew Gelman, and Jennifer Hill. Multiple imputation for continuous and categorical data: comparing joint multivariate normal and conditional approaches. *Political Analysis*, 22(4), 2014.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

Aditya Kunar, Robert Birke, Zilong Zhao, and Lydia Chen. Dtgan: Differential private training for tabular gans. *arXiv preprint arXiv:2107.02521*, 2021.

Mikael Kuusela, Tommi Vatanen, Eric Malmi, Tapani Raiko, Timo Aaltonen, and Yoshikazu Nagai. Semi-supervised anomaly detection–towards model-independent searches of new physics. In *Journal of Physics: Conference Series*, volume 368, pp. 012032. IOP Publishing, 2012.

Ivy Kwok and Raymond Ng. Fast computation of 2-dimensional depth contours. 1998.

Chieh-Hsin Lai, Dongmian Zou, and Gilad Lerman. Robust subspace recovery layer for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2019.

Chieh-Hsin Lai, Dongmian Zou, and Gilad Lerman. Novelty detection via robust variational autoencoding. *arXiv preprint arXiv:2006.05534*, 2020.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic

Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. *Advances in neural information processing systems*, 30, 2017.

Charline Le Lan and Laurent Dinh. Perfect density models cannot guarantee anomaly detection. *arXiv preprint arXiv:2012.03808*, 2020.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *International Conference on Learning Representations (ICLR)*, abs/1711.09325, 2018.

Alexander Lew, Monica Agrawal, David Sontag, and Vikash Mansinghka. Pclean: Bayesian data cleaning at scale with domain-specific probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pp. 1927–1935. PMLR, 2021.

Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Chieh Hubert Lin, Hsin-Ying Lee, Yen-Chi Cheng, Sergey Tulyakov, and Ming-Hsuan Yang. Infinitygan: Towards infinite-pixel image synthesis. *arXiv preprint arXiv:2104.03963*, 2021.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pp. 413–422. IEEE, 2008.

Yi Liu and Yuan F Zheng. Minimum enclosing and maximum excluding machine for pattern description and discrimination. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pp. 129–132. IEEE, 2006.

Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. Picket: Guarding against

corrupted data in tabular data during learning and inference. *arXiv preprint arXiv:2006.04730*, 2020.

Zifan Liu, Jong Ho Park, Theodoros Rekatsinas, and Christos Tzamos. On robust mean estimation under coordinate-level corruption. In *International Conference on Machine Learning*, pp. 6914–6924. PMLR, 2021.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019a.

Francesco Locatello, Michael Tschannen, Stefan Bauer, Gunnar Rätsch, Bernhard Schölkopf, and Olivier Bachem. Disentangling factors of variations using few labels. In *International Conference on Learning Representations*, 2019b.

Romain Lopez, Jeffrey Regier, Michael I Jordan, and Nir Yosef. Information constraints on auto-encoding variational bayes. *Advances in Neural Information Processing Systems*, 31, 2018.

Chao Ma and Cheng Zhang. Identifiable generative models for missing not at random data imputation. *Advances in Neural Information Processing Systems*, 34, 2021.

Shweta Mahajan, Apratim Bhattacharyya, Mario Fritz, Bernt Schiele, and Stefan Roth. Normalizing flows with multi-scale autoregressive priors. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8412–8421, 2020.

Zelda Mariet, Rachael Harding, Sam Madden, et al. Outlier detection in heterogeneous datasets using automatic tuple expansion. 2016.

Emile Mathieu, Tom Rainforth, Nana Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In *International Conference on Machine Learning*, pp. 4402–4412. PMLR, 2019.

Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International conference on machine learning*, pp. 4413–4423. PMLR, 2019.

Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. Eracer: a database approach

for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pp. 75–86, 2010.

John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022.

Sebastian Mika, Bernhard Schölkopf, Alex Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. *Advances in neural information processing systems*, 11, 1998.

Gemma Elyse Moran, Dhanya Sridhar, Yixin Wang, and David Blei. Identifiable deep generative models via sparse decoding. *Transactions on Machine Learning Research*, 2022.

Karl Mosler. Depth statistics. In *Robustness and complex data structures*, pp. 17–34. Springer, 2013.

Mary M Moya and Don R Hush. Network constraints and multi-objective optimization for one-class classification. *Neural networks*, 9(3):463–474, 1996.

Krikamol Muandet and Bernhard Schölkopf. One-class support measure machines for group anomaly detection. *arXiv preprint arXiv:1303.0309*, 2013.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pp. 4723–4732. PMLR, 2019.

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*, 2013.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107: 107501, 2020.

Felix Neutatz, Binger Chen, Ziawasch Abedjan, and Eugene Wu. From cleaning before ml to cleaning for ml. *Data Engineering*, pp. 24, 2021.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Minh-Nghia Nguyen and Ngo Anh Vien. Scalable and interpretable one-class SVMs with deep learning and random fourier features. In *ECML/PKDD*, 2018a.

Minh-Nghia Nguyen and Ngo Anh Vien. Scalable and interpretable one-class svms with deep learning and random fourier features. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 157–172. Springer, 2018b.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Laurel Orr, Atindriyo Sanyal, Xiao Ling, Karan Goel, and Megan Leszczynski. Managing ml pipelines: feature stores and the coming wave of embedding ecosystems. *arXiv preprint arXiv:2108.05053*, 2021.

Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, Philip Torr, et al. Learning disentangled representations with semi-supervised deep generative models. *Advances in neural information processing systems*, 30, 2017.

Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep

learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (Cat. No. 03CH37405)*, pp. 315–326. IEEE, 2003.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10), 2018.

Ignacio Peis, Chao Ma, and José Miguel Hernández-Lobato. Missing data imputation and acquisition with deep hierarchical models and hamiltonian monte carlo. In *Advances in Neural Information Processing Systems*.

Eduardo HM Pena, Eduardo C de Almeida, and Felix Naumann. Discovery of approximate (and exact) denial constraints. *Proceedings of the VLDB Endowment*, 13(3):266–278, 2019.

Tomáš Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

Adrian Alan Pol, Victor Berger, Cecile Germain, Gianluca Cerminara, and Maurizio Pierini. Anomaly detection with conditional variational autoencoders. In *2019 18th IEEE international conference on machine learning and applications (ICMLA)*, pp. 1651–1657. IEEE, 2019.

Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. *arXiv preprint arXiv:1406.1831*, 2014.

Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey

on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

John A Quinn, Christopher KI Williams, and Neil McIntosh. Factorial switching linear dynamical systems applied to physiological condition monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1537–1551, 2008.

John A. Quinn, Christopher K. I. Williams, and Neil McIntosh. Factorial switching linear dynamical systems applied to physiological condition monitoring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1537–1551, 2009.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, pp. 269. NIH Public Access, 2017.

Shebuti Rayana. Odds library, 2016. URL `http://odds.cs.stonybrook.edu`.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019a.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019b.

Sergey Redyuk, Sebastian Schelter, Tammo Rukat, Volker Markl, and Felix Biessmann. Learning to validate the predictions of black box machine learning models on unseen data. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pp. 1–4, 2019.

Jean-Baptiste Regli and Ricardo Silva. Alpha-beta divergence for variational inference. *CoRR*, abs/1805.01045, 2018a.

Jean-Baptiste Regli and Ricardo Silva. Alpha-beta divergence for variational inference. *arXiv preprint arXiv:1805.01045*, 2018b.

Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proceedings of the VLDB Endowment*, 10(11), 2017.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo,

Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Xuanchi Ren, Tao Yang, Yuwang Wang, and Wenjun Zeng. Learning disentangled representation by exploiting pretrained generative models: A contrastive learning view. In *International Conference on Learning Representations*.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1):107–136, 2006.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*, 2011.

Adín Ramírez Rivera, Adil Khan, Imad Eddine Ibrahim Bekkouch, and Taimoor Shakeel Sheikh. Anomaly detection based on zero-shot outlier synthesis and hierarchical feature distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Bernard Rosner. Percentage points for a generalized esd many-outlier procedure. *Technometrics*, 25(2):165–172, 1983.

Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed

Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*, pp. 4393–4402. PMLR, 2018.

Lukas Ruff, Robert A Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep semi-supervised anomaly detection. *arXiv preprint arXiv:1906.02694*, 2019.

Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. End-to-end weak supervision. *Advances in Neural Information Processing Systems*, 34: 1845–1857, 2021.

Adrià Ruiz, Oriol Martinez, Xavier Binefa, and Jakob Verbeek. Learning disentangled representations with reference-based variational autoencoders. *arXiv preprint arXiv:1901.08534*, 2019.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann, and Andreas Grafberger. Automating large-scale data quality verification. *Proceedings of the VLDB Endowment*, 11(12):1781–1794, 2018.

Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pp. 146–157. Springer, 2017.

Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al. Support vector method for novelty detection. In *NIPS*, volume 12, pp. 582–588. Citeseer, 1999.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

Ayush Sekhari, Karthik Sridharan, and Satyen Kale. Sgd: The role of implicit regularization, batch-size and multiple-epochs. *Advances in Neural Information Processing Systems*, 34, 2021.

Tianxiao Shen, Jonas Mueller, Regina Barzilay, and T. Jaakkola. Educating text autoencoders: Latent representation guidance via denoising. In *International Conference on Machine Learning*, 2019.

Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. Universalizing weak supervision. *arXiv preprint arXiv:2112.03865*, 2021.

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems*, 31, 2018.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Guillaume Staerman, Pavlo Mozharovskyi, Stéphan Clémen, et al. The area of the convex hull of sampled curves: a robust functional statistical depth measure. In *International Conference on Artificial Intelligence and Statistics*, pp. 570–579. PMLR, 2020.

Yu-Sung Su, Andrew Gelman, Jennifer Hill, and Masanao Yajima. Multiple imputation with diagnostics (mi) in r: Opening windows into the black box. *Journal of Statistical Software*, 45:1–31, 2011.

Gábor J Székely, Maria L Rizzo, and Nail K Bakirov. Measuring and testing

dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

John Taylor. *Introduction to error analysis, the study of uncertainties in physical measurements*. 1997.

Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep learning on image denoising: An overview. *Neural Networks*, 131:251–275, 2020.

Gary L Tietjen and Roger H Moore. Some grubbs-type statistics for the detection of several outliers. *Technometrics*, 14(3):583–597, 1972.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

F. Tonolini, B. S. Jensen, and R. Murray-Smith. Variational sparse coding. In *UAI*, 2019.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ruben Verborgh and Max De Wilde. *Using OpenRefine*. Packt Publishing Ltd, 2013.

Antonio Vergari, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera. Automatic bayesian density analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5207–5215, 2019.

Miryam Elizabeth Villa-Pérez, Miguel Á Álvarez-Carmona, Octavio Loyola-González, Miguel Angel Medina-Pérez, Juan Carlos Velazco-Rossell, and Kim-Kwang Raymond Choo. Semi-supervised anomaly detection algorithms: A comparative summary and future research directions. *Knowledge-Based Systems*, 218:106878, 2021.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Roger Waleffe, Jason Mohoney, Theodoros Rekatsinas, and Shivaram Venkataraman. Marius++: Large-scale training of graph neural networks on a single machine. *arXiv preprint arXiv:2202.02365*, 2022.

Ziyu Wan, Bo Zhang, Dongdong Chen, Pan Zhang, Dong Chen, Jing Liao, and Fang Wen. Bringing old photos back to life. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2747–2757, 2020.

Yixin Wang, Alp Kucukelbir, and David M. Blei. Robust probabilistic modeling with bayesian data reweighting. In *ICML*, 2017a.

Yu Wang, Bin Dai, Gang Hua, John Aston, and David P Wipf. Green generative modeling: Recycling dirty data using recurrent variational autoencoders. In *UAI*, 2017b.

Matthew Willetts, Stephen Roberts, and Chris Holmes. Semi-unsupervised learn-

ing: Clustering and classifying using ultra-sparse labels. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 5286–5295. IEEE, 2020.

Christopher KI Williams and Michalis K Titsias. Learning about multiple objects in images: Factorial learning without factorial search. In *Advances in Neural Information Processing Systems*, pp. 1415–1422, 2003.

Feng Xia, Ke Sun, Shuo Yu, Abdul Aziz, Liangtian Wan, Shirui Pan, and Huan Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2 (2):109–127, 2021.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. *arXiv preprint arXiv:2010.00654*, 2020.

Doris Xin, Hui Miao, Aditya Parameswaran, and Neoklis Polyzotis. Production machine learning pipelines: Empirical analysis and optimization opportunities. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2639–2652, 2021.

Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *WWW*, 2018.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32, 2019.

Mohamed Yakout, Laure Berti-Équille, and Ahmed K Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 553–564, 2013.

Makoto Yamada, Song Liu, and Samuel Kaski. Interpreting outliers: Localized logistic regression for density ratio estimation. *CoRR*, abs/1702.06354, 2017.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pp. 5689–5698. PMLR, 2018.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*, 2022.

Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

Qian Zhao, Deyu Meng, Zongben Xu, Wangmeng Zuo, and Lei Zhang. Robust principal component analysis with complex noise. In *International conference on machine learning*, pp. 55–63. PMLR, 2014.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017a.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from generative models. *arXiv preprint arXiv:1702.08396*, 2017b.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, pp. 97–112. PMLR, 2021.

Guanjie Zheng, Susan L Brantley, Thomas Lauvaux, and Zhenhui Li. Contextual spatial outlier detection with metric learning. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2161–2170, 2017.

Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674. ACM, 2017.

Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5):363–387, 2012.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018a.

Bo Zong, Qiankun Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Dae ki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations (ICLR)*, 2018b.