universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Variational Autoencoders with Structured Missingness“

verfasst von / submitted by

Michael Raffelsberger BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 645
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Data Science
Betreut von / Supervisor:	Ass.-Prof. Dipl.-Ing. Dr.techn. Sebastian Tschiatschek, BSc

Acknowledgements

Ich möchte mich ganz besonders bei meinen Eltern sowie meinem Bruder bedanken, die mich im Laufe meiner Studienzeit und darüber hinaus immer unterstützt und motiviert haben. Außerdem danke ich meinem Betreuer Ass.-Prof. Sebastian Tschiatschek, ohne den diese Arbeit nicht möglich geworden wäre.

Abstract

We investigate and improve variational autoencoders (VAE) in the presence of missing at random data. In our assumed setting, certain variables of a sample are unobserved because of the values of variables recorded earlier that already carry some information about the unobserved values. Imagine a doctor deciding to skip further diagnostic tests based on already available results of other tests. In this setting, we have both knowledge about the missingness process as well as a presumption about the values of the unobserved variables. We show that such missingness leads to unsatisfactory results, especially for small datasets and that knowledge about the missingness process is barely helpful. Therefore, our core idea is to incorporate knowledge about the actual unobserved values via imputation or additional loss terms to regularize the model. This idea allows to find a sweet spot between data and knowledge, improves results and makes the model more robust and reliable.

Kurzfassung

Wir untersuchen und verbessern Variational Autoencoders (VAE), wenn Variablen eines Datensatzes missing at random sind. Im untersuchten Setting fehlen Werte einer oder mehrerer Variablen aufgrund der bereits beobachteten Werte anderer Variablen, die Information über die fehlenden Werte enthalten. Bei einer medizinischen Untersuchung beispielsweise könnte ein Doktor entscheiden, Folgeuntersuchungen auszulassen, da bereits vorliegende Befunde Rückschlüsse auf die Resultate der Folgeuntersuchungen ermöglichen. In diesem Fall liegt sowohl Vorwissen zum Missingness Prozess als auch zu den fehlenden Werten selbst vor. Wir zeigen, dass dieser Fall zu unerwünschten Ergebnissen führen kann, insbesondere bei kleinen Datensätzen, und dass Vorwissen zum Missingness Prozess nur eingeschränkt hilfreich ist. Unsere Kernidee besteht darin, das Vorwissen zu den fehlenden Werten direkt zu verwenden, um das Modell zu regularisieren. Diese Kombination aus Daten und Vorwissen verbessert die Ergebnisse und macht das Modell robuster und zuverlässiger.

Contents

List of Tables

List of Figures

1 Introduction

Missing data is a frequent challenge in real-life data analysis and many models naturally require fully observed data samples without any missing features. Often, we have prior knowledge about the underlying missingness process or even some information about the value range of the missing features for a partially observed data set. In a medical setting, a doctor might conduct some diagnostic tests for a patient and given the results, decide to skip further tests. Consequently, the values of those further tests are missing. However, they are missing because the earlier diagnosis results or variables concerning the medical history already indicated that they would not be relevant any more. Being able to incorporate such additional knowledge can help a machine learning model to learn a more suitable representation of data samples leading to improvements in various downstream tasks.

1.1 Background

Our work extends deep latent variable models (DLVMs) [KW13, RMW14, KW19] to such settings. The most prominent DLVM is the variational autoencoder (VAE). It combines the network architecture of autoencoders with the probabilistic framework of directed graphical models (bayesian networks) [BN06]. In the simplest case, the probabilistic model is given by the joint probability distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$ of the observed data sample $\mathbf{x} \in \mathbb{R}^D$ and the corresponding latent variable $\mathbf{z} \in \mathbb{R}^J$. Figure 1.1 illustrates the corresponding model graph. Maximizing the marginal log-likelihood $\log p_\theta(\mathbf{x})$ is not possible since $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z}$ is intractable. For this reason, VAEs make use of variational inference, or more precisely amortized inference. We introduce the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$, which is parametrized by an encoder neural network with parameters ϕ. The marginal log-likelihood or evidence can then be rewritten as

$$\log p_\theta(\mathbf{x}) = \mathcal{L}_{\theta,\phi}(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$$

with

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})),$$

which is called the evidence lower bound (ELBO). Note that the distribution $p_\theta(\mathbf{x}|\mathbf{z})$ is parametrized by the decoder neural network with parameters θ and the first term in the ELBO resembles a reconstruction error similar to a standard autoencoder. The approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is typically a diagonal Gaussian distribution, i.e. the encoder or inference network outputs parameters $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_J]$ and $\log \boldsymbol{\sigma}^2 = [\log \sigma_1^2, \ldots, \log \sigma_J^2]$. The prior distribution $p_\theta(\mathbf{z})$ is typically an isotropic Gaussian distribution $N(\mathbf{0}, \mathbf{I}_J)$. The

Kullback-Leibler divergence between the approximate and the true posterior distribution functions as a regularization constraining the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$. Maximizing the ELBO on a given dataset approximately maximizes the intractable marginal log-likehood $\log p_\theta(\mathbf{x})$ and at the same time decreases the Kullback-Leibler divergence term. VAEs are a powerful self-supervised model class capable of representation learning, data generation, missing value imputation and many other probabilistic reasoning tasks.

We consider a more complex setting where a data sample $\mathbf{x} \in \mathbb{R}^D$ consists of an observed part $\mathbf{x}_O$ and an unobserved or missing part $\mathbf{x}_U$. If $\mathcal{I} = 1, \ldots, D$ is the observable index set, $\mathcal{O}$ is the index set of the observed part and $\mathcal{U}$ is the index set of the unobserved part, we have $\mathcal{I} = \mathcal{O} \cup \mathcal{U}$ and $\mathcal{O} \cap \mathcal{U} = \emptyset$. The missingness mask $\mathbf{m} \in \mathbb{R}^D$ indicates which parts of $\mathbf{x}$ are missing, $m_d = 0$ means that $d \in \mathcal{O}$ and $m_d = 1$ means that $d \in \mathcal{U}$. We actually observe the corrupted $\tilde{\mathbf{x}} = (1 - \mathbf{m}) \odot \mathbf{x}$ as well as the missingness mask $\mathbf{m}$. We can factorize the joint probability distribution of our graphical model in Figure 1.1 as $p(\mathbf{z}, \mathbf{x}, \mathbf{m}, \tilde{\mathbf{x}}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{m}|\mathbf{x})p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})$, where $p(\mathbf{z})$ is the prior distribution, $p(\mathbf{x}|\mathbf{z})$ is the conditional distribution of $\mathbf{x}$ given $\mathbf{z}$, $p(\mathbf{m}|\mathbf{x})$ is the missingness process and $p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})$ is deterministic.

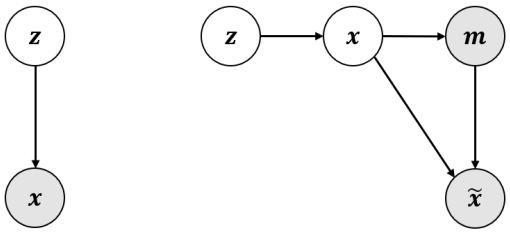


Figure 1.1: The directed graphical model of a simple VAE (left) and our model (right). Variables in white circles are unobserved, variables in grey circles are observed.

Following the convention of [LR19], we can distinguish multiple types of missingness as elaborated on in [IMF20]:

- Missing completely at random (MCAR): $p(\mathbf{m}|\mathbf{x}) = p(\mathbf{m})$. The missingness is independent of all variable values. Maximum likelihood estimation is sound in this setting.

- Missing at random (MAR): $p(\mathbf{m}|\mathbf{x}) = p(\mathbf{m}|\mathbf{x}_O)$. The missingness just depends on observed variables. In the MAR case, we can still make use of maximum likelihood estimation.

- Missing not at random (MNAR): $p(\mathbf{m}|\mathbf{x}) = p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U)$. The missingness can depend on observed variables like in the MAR setting, but most importantly also depends on the missing values themselves. A well-known case is self-censoring, where a variable is missing because of its own value. Maximum likelihood estimation ignoring MNAR missingness leads to biased results.

1.2 Related Work

There has been work on extending VAEs to missing data: Early attempts derive an ELBO objective just for the observed part $\mathbf{x}_O$, which works for MCAR and MAR but not for MNAR. [NOGV20] just zero-impute the missing values, [MF19] present a similar approach, but use the importance-weighted autoencoder (IWAE) [BGS15] instead of the VAE, which uses a slightly different lower bound on the marginal log-likelihood from the ELBO. In a more general solution, [MTP+18] use a permutation invariant encoder to deal with the missingness. Subsequent work also deals with MNAR missingness: [CNW20] outline a VAE for MCAR, MAR and MNAR settings maximizing the marginal log-likelihood of the observed part $\mathbf{x}_O$ given the missingness mask, i.e. $\log p(\mathbf{x}_O|\mathbf{m})$. Both the encoder and the decoder have $\mathbf{m}$ as input, which helps to distinguish zero-imputed variables from actual zeros. In contrast, [IMF20] maximize the joint log-likelihood $\log p(\mathbf{x}_O, \mathbf{m})$ of the observed data and the missingness mask. Their directed graphical model is similar to ours. In particular, they incorporate prior information about the missingness type in their model, e.g. self-censoring of some variable. However, they exclusively deal with the MNAR setting. In contrast to them, we primarily analyze a MAR setting where the missingness of a feature depends on the value of another previously recorded feature, which is correlated with the missing feature. Our goal is to leverage knowledge about that missingness process on the one hand and knowledge about that correlation on the other hand.

1.3 Contribution

Apart from the observed $\tilde{\mathbf{x}}$ and $\mathbf{m}$, we assume to have some knowledge about the missingness process. In medicine, it is common to depict the information acquisition process used for diagnostic workup in the form of a decision tree [ZBFA08] or similarly as a flow diagram. [THHM15] use classification and regression trees (CARTs) [BFOS17] to explore the structure of missing data in datasets. CARTs make use of recursive binary splits and are easily interpretable by a human. Inspired by this format, we assume to have prior knowledge about the missingness process specified in the form of CARTs. Note that we do not however use those trees as machine learning models trained on data. They are just a means of being able to flexibly specify prior knowledge about the missingness process in a unified format. We assume to know $\mathbf{t} = [t_1, \ldots, t_D]$, where each t_d is a binary decision tree specifying information about the missingness structure of variable d. Each of these trees has to be manually designed based on domain knowledge beforehand and a tree will be empty, i.e. $t_d = \perp$, if no such knowledge is available. Note that by using multiple splits, a binary decision tree can model a wide range of possible missingness structures. However, we will limit our analysis to very simple decision stumps. Unlike MNAR, MAR missingness does not directly reveal any information about the missing values. However, we argue that by considering the missing data problem as imbalanced problem, sample weighting can alleviate the issues caused by the missingness.

In addition to knowledge about the missingness process itself, we investigate the

setting where we have access to expert knowledge or weak scientific knowledge about a missing feature given the feature causing the missingness. Say we have a sample with missingness in variable x_u with $u \in \mathcal{U}$. We assume to have prior knowledge in the form of a simple linear regression with x_o with $o \in \mathcal{O}$ as explanatory variable, i.e. $x_u = a + b \cdot x_o + \varepsilon$ and $\varepsilon \sim N(0, \sigma^2)$. Hence, for a given value of x_o, we approximately know that $\mathbb{E}[x_u|x_o] = a + b \cdot x_o$. Alternatively, we could construct a prior knowledge interval $[a + b \cdot x_o - \sigma, a + b \cdot x_o + \sigma]$ for x_u. This specification resembles the expert knowledge a doctor might have regarding the relationship between two blood levels for instance. To incorporate such knowledge into a VAE, we take inspiration from the work on physics-guided neural networks (PGNN) [DKW$^+$17] and domain-adapted neural networks (DANN) [MIM$^+$18]. [DKW$^+$17] combine data-centric models (e.g. multi-layer perceptrons) with existing physical knowledge for better generalization and scientific consistency in supervised learning, particularly in small data settings. They leverage two ideas, first to use the output of traditional physics-based model as additional input to a data-centric model and second to add physics-based loss terms to the data-centric model penalizing the output towards valid values. [MIM$^+$18] elaborate more extensively on the idea of knowledge-based loss terms to find the sweet spot between data and prior knowledge in situations where data is limited, expensive or of poor quality. We analyze similar approaches of incorporating prior knowledge into a VAE with missing data:

- Imputing missing values in x_u based on $\mathbb{E}[x_u|x_o]$ is similar to the idea of using the output of a traditional model as input to a data-centric model.
- Using loss terms to constrain the reconstructed values of a VAE to lie in the interval $[a + b \cdot x_o - \sigma, a + b \cdot x_o + \sigma]$ is an application of the idea of knowledge-based loss terms.

We will conduct experiments dealing with the following bullet points:

- We confirm the findings and solutions of previous related work on the MNAR setting, especially of [IMF20].
- We analyze problems arising with MAR specifications where the missing feature is correlated with the feature causing the missingness.
- We show that prior knowledge about the missingness process is not as helpful in our MAR setting as in an MNAR setting.
- We propose and investigate ideas from domain-adapted and physics-guided neural networks for VAEs with missing at random data when we have access to prior knowledge about the missing values.

We evaluate our models and ideas on simulated data where we control both the data generating and the missingness process. Later, we apply our model to a medical dataset, where we still control the missingness process. We show that our approaches help to reduce the imputation error but struggle with learning better latent state representations.

2 Background and Related Work

2.1 Missing Data Theory

[LR19] define missing data as unobserved values that would be meaningful for analysis if they were observed. Recall the factorization of our graphical model $p(\mathbf{z}, \mathbf{x}, \mathbf{m}, \tilde{\mathbf{x}}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{m}|\mathbf{x})p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})$. We call $p(\mathbf{m}|\mathbf{x})$ the missingness process and denote by $\check{\mathbf{m}}$ the realized missingness indicators and by $\check{\mathbf{x}}$ the realized values of $\mathbf{x}$. As outlined in [LR19] and defined in [MR15], the missingness process can follow different mechanisms:

- Missing always completely at random (MACAR): $p(\mathbf{m}|\mathbf{x}) = p(\mathbf{m})$ for all $\mathbf{m}, \mathbf{x}$.
- Missing completely at random (MCAR): $p(\check{\mathbf{m}}|\mathbf{x}) = p(\check{\mathbf{m}})$ for all $\mathbf{x}$.
- Missing always at random (MAAR): $p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) = p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U^*)$ for all $\mathbf{m}, \mathbf{x}_O, \mathbf{x}_U, \mathbf{x}_U^*$.
- Missing at random (MAR): $p(\check{\mathbf{m}}|\check{\mathbf{x}}_O, \mathbf{x}_U) = p(\check{\mathbf{m}}|\check{\mathbf{x}}_O, \mathbf{x}_U^*)$ for all $\mathbf{x}_U, \mathbf{x}_U^*$.
- Missing not always at random (MNAAR): $p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) \neq p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U^*)$ for some $\mathbf{m}, \mathbf{x}_O$ and some $\mathbf{x}_U \neq \mathbf{x}_U^*$.
- Missing not at random (MNAR): $p(\check{\mathbf{m}}|\check{\mathbf{x}}_O, \mathbf{x}_U) \neq p(\check{\mathbf{m}}|\check{\mathbf{x}}_O, \mathbf{x}_U^*)$ for some $\mathbf{x}_U \neq \mathbf{x}_U^*$.

The usage of the terms MCAR, MAR and MNAR is not consistent in literature [SGJC13, MR15, BPR20]. For instance, it makes a difference whether the term MAR refers just to the realized data, which is sometimes also called realized MAR, or to the missingness mechanism itself, which we call MAAR and is sometimes also called everywhere MAR. MAAR always produces MAR data, i.e. MAAR implies MAR but not vice versa. In the upcoming sections, we will stick to the terms MCAR, MAR and MNAR meaning both the realized data being MCAR, MAR and MNAR as well as the underlying missingness mechanism being MACAR, MAAR and MNAAR.

The missingness mechanism has decisive implications on the analysis of the data. MAR suffices for pure-likelihood inference without explicitely modeling the missingness mechanism. As pointed out by [Mar08], inference in a MAR setting can still be biased if the data model neglects dependencies between variables. MNAR on the other hand generally poses a greater challenge. In the face of MNAR missingness, ignoring the missingness mechanism leads to biased results, since the unobserved values themselves induce the missingness. A simple example of MNAR is self-censoring, where the missingness in dimension d exactly depends on the value in that dimension, i.e. $p(m_d|\mathbf{x}) = p(m_d|x_d)$.

[LR19] presents a taxonomy for methods dealing with missing data with four categories:

- Complete-case analysis (CC): In complete-case analysis, we restrict the analysis to fully observed samples. This solution is simple but can lead to heavily biased results. In our experiments, we briefly investigate the consequences of complete-case analysis in our setting.

- Weighting: If we know the selection or missingness mechanism, we can use the selection or missingness probabilities to conduct a weighted analysis. A well-known example is the Horvitz-Thompson estimator [HT52] using sample weights inversely proportional to the selection probability when estimating the mean of a finite population from a sample without replacement.

- Imputation: Various imputation methods try to replace the missing values in a meaningful way to afterwards conduct the analysis on a "pseudo-complete" dataset. In our work, we try to leverage prior knowledge to impute the missing values with a conditional expectation.

- Model-based approches.

2.2 Variational Autoencoders

2.2.1 Fundamentals

Autoencoders

Autoencoders (AE) are an unsupervised model class trying to reconstruct their input $\mathbf{x} \in \mathbb{R}^D$ [GBC16]. An encoder $g(\mathbf{x})$, often a feedforward neural network, maps $\mathbf{x}$ to a latent representation $\mathbf{z} \in \mathbb{R}^J$ and a decoder $h(\mathbf{z})$, often symmetric to the encoder, maps $\mathbf{z}$ to $\hat{\mathbf{x}} \in \mathbb{R}^D$, i.e. $\hat{\mathbf{x}} = h(g(\mathbf{x})) = h(\mathbf{z})$. The minimization objective is a reconstruction error between $\mathbf{x}$ and $\hat{\mathbf{x}}$, e.g. the mean squared error for continuous $\mathbf{x}$. To prevent autoencoders from just copying the input to the latent space and finally the output, and thereby achieving perfect reconstruction accuracy, the latent representation is typically a bottleneck with a lower dimensionality than the input, i.e. $J < D$. The reconstruction criterion is primarily an auxiliary task to obtain useful latent representations of $\mathbf{x}$ for e.g. dimensionality reduction and visualization.

Several extensions introduce regularization techniques to make the latent representations of autoencoders more robust and eventually more meaningful. Denoising autoencoders (DAE) [VLBM08] are one such attempt. Instead of directly using $\mathbf{x}$ as input to the encoder, DAEs try to reconstruct $\mathbf{x}$ from a corrupted version $\tilde{\mathbf{x}}$. It turns out the model not only learns to denoise a corrupted sample $\tilde{\mathbf{x}}$ but also to yield more robust and better generalizing latent representations.

Variational autoencoders can similarly be interpreted as autoencoders with a specific regularization technique penalizing the latent space to approximately follow the shape of a predefined prior distribution. They are based on a thorough probabilistic framework allowing many applications that require some sort of probabilistic reasoning.

Bayesian Networks and Latent Variables

Bayesian networks or directed graphical models (DGM) are probabilistic models representing the relationships between random variables in the form of a directed acyclic graph (DAG) [BN06]. The nodes of this graph represent the random variables while the directed edges represent the relationships between those variables. In particular, a bayesian network depicts how we can factorize the joint probability distributions of those variables to account for their causal relationships. For instance, the joint distribution of the variables in the DAG in Figure 2.1 can be factorized as $p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_2)$. More generally, the factorization of $p(\mathbf{x})$ according to some DGM is given by $p(\mathbf{x}) = \prod_{d=1}^{D} p(x_d|\text{pa}(x_d))$ where $\text{pa}(x_d)$ denotes the parents of variable x_d in the graph. If we know the dynamics of the conditional distributions in the factorization, we can use ancestral sampling to generate samples from $p(\mathbf{x})$. In ancestral sampling, we first sample values for the root nodes, i.e. nodes without a parent and then sequentially sample realizations of the other variables x_d from $p(x_d|\text{pa}(x_d))$ where we set the values of the parents to their already sampled realizations.

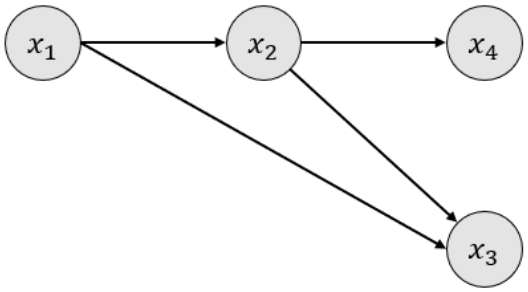


Figure 2.1: Directed acyclic graph example.

Variational autoencoders are based on bayesian networks with latent variables as explained in [KW19]. In a latent variable model with joint distribution $p(\mathbf{z}, \mathbf{x})$, the latent variables $\mathbf{z}$ unlike $\mathbf{x}$ are unobserved. They are typically parents of the observed variables $\mathbf{x}$. Via latent variables, we can specify very flexible marginal distributions for $\mathbf{x}$ even if all the conditional distributions in the factorization follow simple distributions. [KW19] provide an example of this capability: If $p(\mathbf{z})$ is discrete and $p(\mathbf{x}|\mathbf{z})$ is Gaussian, then the marginal distribution $p(\mathbf{x})$ is a mixture of Gaussians. If $p(\mathbf{z})$ is continuous, $p(\mathbf{x})$ is even an infinite mixture of Gaussians.

Variational autoencoders are an example of a deep latent variable model (DLVM). A DLVM is a latent variable model $p(\mathbf{z}, \mathbf{x})$ that is parametrized by a neural network. For a diagonal Gaussian conditional distribution $p(\mathbf{x}|\mathbf{z})$ for instance, a neural network might predict the parameters $\boldsymbol{\mu}$ and $\log \boldsymbol{\sigma}^2$ to sample a realization $\mathbf{x}^{(i)} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.

Variational Inference

In a latent variable model, the posterior distribution $p(\mathbf{z}|\mathbf{x})$ and the marginal distribution or evidence $p(\mathbf{x})$ are often intractable and it is therefore not possible to directly maximize the likelihood of the observed variables $\mathbf{x}$. Variational inference solves this problem by approximating the true posterior with a variational distribution $q(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$ [BN06, ZBKM18, KW13]. It can be shown that the log-likelihood of the observed data becomes $\log p(\mathbf{x}) = \mathcal{L}(q) + D_{KL}(q||p)$, where $\mathcal{L}(q) = \int q(\mathbf{z}) \log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} d\mathbf{z} = \mathbb{E}_{q(\mathbf{z})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z})}\right]$ and $D_{KL}(q||p)$ is the Kullback-Leibler divergence between the variational distribution $q(\mathbf{z})$ and the true posterior $p(\mathbf{z}|\mathbf{x})$. Since $D_{KL}(q||p) \geq 0$, we have $\log p(\mathbf{x}) \geq \mathcal{L}(q)$ which is why $\mathcal{L}(q)$ is called evidence lower bound (ELBO). The basic idea of variational inference is to maximize the ELBO with respect to $q(\mathbf{z})$ and thereby approximately maximize the evidence and minimize the divergence between the variational distribution and the true posterior distribution. By constraining the search for such a $q(\mathbf{z})$ to a family of distributions parametrized with ϕ, maximizing the ELBO with respect to ϕ becomes tractable.

2.2.2 Model

Architecture

Variational autoencoders (VAE) [KW13, RMW14, KW19] combine directed graphical models with continuous latent variables and the architecture of autoencoders. In the vanilla case, the joint probability distribution of the underlying graphical model is factorized as $p_\theta(\mathbf{z}, \mathbf{x}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$ where we assume the prior $p_\theta(\mathbf{z})$ to follow an isotropic Gaussian distribution $N(\mathbf{0}, \mathbf{I}_J)$ and the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ to follow a Gaussian distribution in the case of continuous $\mathbf{x}$ and a Bernoulli distribution in the case of binary $\mathbf{x}$. The encoder or recoginition/inference model with parameters ϕ parametrizes the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ of the unobserved latent variables. Hence, unlike a standard autoencoder, the encoder of a variational autoencoder outputs the parameters of the variational distribution from which we can sample the latent code $\mathbf{z}$ instead of the latent code directly. Usually, the approximate posterior is modeled as a diagonal Gaussian distribution. Therefore, the encoder outputs $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_J]$ and $\log \boldsymbol{\sigma}^2 = [\log \sigma_1^2, \ldots, \log \sigma_J^2]$ and $\mathbf{z}$ is consequently sampled from $N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. The decoder or generative model with parameters θ takes the sampled $\mathbf{z}$ as input and parametrizes the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. Note that in more complex architectures, there can be multiple (hierarchical) layers of latent variables [RMW14].

Objective

Similar to Section 2.2.1, the log-likelihood for an observed sample $\mathbf{x}^{(i)}$ is intractable, but we can rewrite it as

$$\log p_\theta(\mathbf{x}^{(i)}) = \mathcal{L}_{\theta,\phi}(\mathbf{x}^{(i)}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})), \tag{2.1}$$

where

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) \tag{2.2}$$

is the ELBO [KW13]. Recall that maximizing the ELBO on the one hand approximately maximizes the marginal log-likelihood $\log p_\theta(\mathbf{x}^{(i)})$ and on the other hand reduces the divergence between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ and the true posterior $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$ [KW19]. The first term in the ELBO objective is basically a reconstruction error and the second term is a regularization term penalizing the approximate posterior to be close to the prior distribution $p_\theta(\mathbf{z})$. [HMP+16] extend the ELBO objective with a hyperparameter β yielding

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})] - \beta \cdot D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})). \tag{2.3}$$

Setting $\beta = 1$ recovers the standard VAE objective while setting $\beta > 1$ enforces a disentangled latent space, where different dimensions control independent data generating factors. In general, the hyperparameter β symbolizes the tradeoff between a nicely-behaved latent space close to the specified prior distribution and the reconstruction performance of the model. In the case of image data, a large value of β leads to blurrier images.

Optimization

[KW13] and [RMW14] propose a reparametrization trick, also called stochastic backpropagation, to compute the gradients of the ELBO with respect to ϕ. This solution allows us to jointly learn the model parameters ϕ and θ via (stochastic) gradient descent. To compute gradients of the reconstruction term $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})]$ in the ELBO, we can take a Monte Carlo sample

$$\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(l)}) \quad \text{where} \quad \mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}). \tag{2.4}$$

Unfortunately, we cannot backpropagate gradients through the sampling operation of $\mathbf{z}^{(l)}$, which we need to compute gradients with respect to the encoder parameters ϕ. However, using the reparametrization trick, we can reformulate the sampling operation as $\mathbf{z}^{(l)} = g_\phi(\mathbf{x}^{(i)}, \boldsymbol{\varepsilon}^{(l)})$ with $\boldsymbol{\varepsilon}^{(l)} \sim p(\boldsymbol{\varepsilon})$ and $g(\cdot)$ a deterministic function. For instance, we could compute $\mathbf{z}^{(l)} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}^{(l)}$ with $\boldsymbol{\varepsilon}^{(l)} \sim N(\mathbf{0}, \boldsymbol{I}_J)$ instead of directly sampling $\mathbf{z}^{(l)} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. After the reparametrization, any automatic differentiation package such as PyTorch [PGM+19] can backpropagate gradients to the encoder. Note that the gradient computation with respect to θ is more straightforward and there is often a differentiable closed-form solution of the Kullback-Leiber divergence term in the ELBO.

Importance Weighted Autoencoders

[RMW14] presents a way to approximately evaluate the marginal log-likelihood of a sample $\mathbf{x}^{(i)}$ using the trained recognition model for importance sampling weights

$$\log p(\mathbf{x}^{(i)}) = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \right] \tag{2.5}$$

$$\approx \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(k)}) p_\theta(\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x}^{(i)})} \quad \text{where} \quad \mathbf{z}^{(k)} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}). \tag{2.6}$$

[BGS15] use the right-hand side of the second line as objective function instead of the ELBO and call their approach the importance weighed autoencoder (IWAE). In fact, the objective is equivalent to the ELBO for $K = 1$ but becomes a tighter lower bound of the evidence the larger K gets. [BGS15] show that their approach leads to richer representations in the latent space with more active dimensions.

2.2.3 Applications (Selection)

As demonstrated in [RMW14], VAEs offer a wide range of possible usages, including tasks relying on the generative model to e.g. generate novel samples or impute missing values as well as tasks using the learned latent representation for e.g. 2D visualization. [KMJRW14] use VAEs for semi-supervised learning. They feed the latent space representation into a classifier or alternatively model the class label itself as latent variable. In a review paper, [TBL18] demonstrate that VAE approaches for various representation learning meta-priors like disentanglement or clustering structure exist. Conditional variational autoencoders (CVAE) are used for image segmentation [SLY15] or generating images with certain attributes like a smiling old man without eyewear. [BVV+15] propose a VAE with a recurrent encoder and decoder to obtain continuous representations of sentences. The model is particularly successful in interpolating sentences by averaging their latent space representations. A similar approach is taken by [GBWD+18] to generate molecules encoded as SMILES strings. Further, VAEs have demonstrated to be successful in dealing with multiple modalities, e.g. combining textual and visual information. [SNM16] present a VAE with a latent joint representation of multiple modalities, where we can even generate a missing modality given the others. [KGGV19] use a bimodal VAE for text and images to detect fake news on social media. Thereby, a binary classifier trained end-to-end with the VAE architecture takes the learned latent representation as input. Finally, [MTP+18] present a framework to use a VAE for active feature acquisition at test time by approximating an information reward.

2.2.4 VAEs with Missing Data

Several previous works deal with extensions of variational autoencoders when entries are missing.

[NOGV20] present a solution for the MCAR case. They assume that the conditional distribution of $\mathbf{x}$ factorizes as $p(\mathbf{x}|\mathbf{z}) = \prod_d p(x_d|\mathbf{z})$. Under this assumption, the observed and unobserved dimensions of $\mathbf{x}$ are conditionally independent. Using a variational

distribution $q_\phi(\mathbf{z}|\mathbf{x}_O)$ that just relies on the observed parts of $\mathbf{x}$, they derive an adjusted version of the ELBO:

$$\log p(\mathbf{x}_O^{(i)}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})}\left[\sum_{d\in\mathcal{O}} \log p(x_d^{(i)}|\mathbf{z})\right] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})||p_\theta(\mathbf{z})), \tag{2.7}$$

and argue that by using the zero-imputed $\tilde{\mathbf{x}}$ as encoder input, the missing attributes do neither contribute to the encoder output nor the derivatives during backpropagation.

[MF19] present a similar zero imputation based solution but replace the VAE with an importance weighted autoencoder. Therefore, their lower bound becomes:

$$\log p(\mathbf{x}_O^{(i)}) \geq \mathbb{E}_{\mathbf{z}^{(1)},\dots,\mathbf{z}^{(K)}\sim q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})}\left[\log \frac{1}{K}\sum_{k=1}^{K} \frac{p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z}^{(k)})p_\theta(\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x}_O^{(i)})}\right]. \tag{2.8}$$

The alternative importance weighted objective is supposed to alleviate the weakness of simply imputing missing values with zeros.

The partial variational autoencoder [MTP$^+$18] is another solution for the MCAR/MAR setting. The authors also integrate out the missing values but equip the encoder with a permutation invariant function to become invariant to missing dimensions in the input samples. All available dimensions x_d with $d \in \mathcal{O}$ are combined with some identity information e_d, e.g. a one-hot encoding or an embedding, and then mapped to dimension J via $h(x_d, e_d)$. A permutation invariant function $g((h(x_d, e_d)_{d\in\mathcal{O}})$ such as max-pooling is applied leading to representation $c(\mathbf{x}_O) \in \mathbb{R}^J$. The following encoder layers finally map this representation to the parameters of the variational distribution. [MTP$^+$18] primarily use the partial VAE in a framework for active variable selection at test time, i.e. the question what variable to record next to maximize the information reward. Our setting is somewhat different. We assume to have a recorded dataset with missing values at random and know why those values are missing and optionally what their values should approximately be like.

[CNW20] point out that simple zero imputation implies the disadvantage that the network cannot distinguish between missing values and observed values with value zero. In their approach, which is also valid under MNAR, they feed the missingness indicator mask $\mathbf{m}$ as additional input to the encoder and optionally also the decoder. Their lower bound is given by:

$$\log p(\mathbf{x}_O^{(i)}|\mathbf{m}^{(i)}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\tilde{\mathbf{x}}^{(i)},\mathbf{m}^{(i)})}[\log p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z},\mathbf{m}^{(i)})] - D_{KL}(q_\phi(\mathbf{z}|\tilde{\mathbf{x}}^{(i)},\mathbf{m}^{(i)})||p_\theta(\mathbf{z})). \tag{2.9}$$

The not-MIWAE [IMF20] is a primary inspiration for our work. The underlying factorization of the joint probability distribution $p(\mathbf{z},\mathbf{x},\mathbf{m}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{m}|\mathbf{x})$ resembles our graphical model. However, the not-MIWAE is particularly designed for MNAR missingness, ideally with prior knowledge about the missingness process. In the presence of MNAR missingness, it is necessary to optimize the parameters of the generative model and the missingness model jointly, which is accomplished via the following importance sampling based lower bound:

$$\log p(\mathbf{x}_O^{(i)}, \mathbf{m}^{(i)}) \geq \mathbb{E}\left[\log \frac{1}{K}\sum_{k=1}^{K} w_k^{(i)}\right], \tag{2.10}$$

where

$$w_k^{(i)} = \frac{p_\gamma(\mathbf{m}^{(i)}|\mathbf{x}_O^{(i)}, \mathbf{x}_U^{(k)}) p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z}^{(k)}) p_\theta(\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x}_O^{(i)})}, \tag{2.11}$$

and $(\mathbf{z}^{(k)}, \mathbf{x}_U^{(k)})$ are sampled from $q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})p_\theta(\mathbf{x}_U|\mathbf{z})$. The reparametrization trick is applied twice, first when sampling $\mathbf{z}$ and second when sampling $\mathbf{x}_U$. The objective approximately maximizes the likelihood of both the observed part $\mathbf{x}_O$ of $\mathbf{x}$ and the missingness mask $\mathbf{m}$. Ideally, we have some prior knowledge about the missingness process $p_\gamma(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U)$ to improve the performance over an agnostic model. [IMF20] present self-censoring as straightforward example that could easily be incorporated into the model.

The work that comes closest to our setting in terms of applications is probably [LROI21]. It introduces an importance weighted autoencoder for non-ignorable missingness in electronic health records. Similar to [IMF20], the reparametrization trick is used twice. The approach is more general than ours in the sense that it also works for MNAR missingness. However, it does not explicitly attempt to incorporate prior knowledge into the model, an idea that is central in our approach.

Other work in the area of deep generative modeling in the presence of missing data includes [GHH+21], [MZ21] and [GCHK21].

2.3 Differentiable Decision Trees

Recall that we formulate the missingness process as a binary decision tree, which can be easily constructed by a human domain expert. To be able to incorporate such knowledge into the core model similar to the not-MIWAE [IMF20], we propose to convert the human-designed decision tree into a differentiable decision tree.

The most famous implementation of decision trees as machine learning models are classification (and regression) trees [BFOS17]. They are a supervised method that applies recursive binary splits to the feature space. In particular, they greedily construct a tree in a top-down manner. All internal nodes $n \in \mathcal{N}$ of the tree are decision or split nodes, which split the feature space into two parts with an axis-aligned, deterministic binary split along one dimension of the feature space. A sample that is sent through the decision tree ends up at exactly one terminal or leaf node $l \in \mathcal{L}$, which typically holds a probability distribution π_l over the classes in the classification case or a real number in the regression case to serve as prediction for the given sample.

Multiple publications deal with the topic of differentiable decision trees, often from different perspectives.

[KFCB15] introduce a differentiable version of a binary decision tree. In particular, they aim at using the successful architecture of decision trees trained via gradient descent

and inheriting the advantages from representation learning in neural networks. The internal splits in this differentiable decision tree are stochastic rather than deterministic and follow the form $d_n(\mathbf{x}) = \sigma(f_n(\mathbf{x}))$, where $d_n(\cdot)$ is the split function at decision node n, $\sigma(\cdot)$ is the sigmoid function and $f_n(\cdot)$ is a (fully-connected) neural network with learnable parameters. These splits are probabilistic since the sigmoid function outputs values in $(0, 1)$, which determine what share is passed to the left subtree of split node n ($d_n(\mathbf{x})$) and what share is passed to its right subtree ($1 - d_n(\mathbf{x})$). Since all those splits are soft, we call the routing function $\mu_l(\mathbf{x}) = \prod_{n \in \mathcal{N}} d_n(\mathbf{x})^{\mathbb{1}_{l \in L(n)}} (1 - d_n(\mathbf{x}))^{\mathbb{1}_{l \in R(n)}}$ of such a tree a soft routing function. Here, $L(n)$ is the set of nodes in the left subtree of node n while $R(n)$ is the set of nodes in the right subtree of node n and $\mathbb{1}$ is the indicator function. The soft routing function is evaluated by traversing the tree and eventually some non-zero share of $\mathbf{x}$ is routed to every leaf in the tree. The predicted probability of y from tree t is then given by $p_t(y|\mathbf{x}) = \sum_{l \in \mathcal{L}} \pi_{l_y} \mu_l(\mathbf{x})$ where π_l holds the probability distribution over the output classes for leaf node l and is also a learnable parameter. [KFCB15] evaluate their differentiable decision tree both as a stand-alone model and as a classification head instead of fully-connected layers in a convolutional neural network.

[FH17] leverage a similar idea for a different purpose. Their main interest is to distill an accurate but hard to interpret deep neural network into a soft decision tree, which is easier to interpret than the neural network and has a better performance than a decision tree that is directly trained on the data. Their split function $d_n(\mathbf{x}) = \sigma\left(\frac{\mathbf{w}_n^T \mathbf{x} + b_n}{\tau}\right)$ is just the sigmoid function applied to a linear mapping of the input sample $\mathbf{x}$, where $\mathbf{w}_n$ and b_n are learnable parameters. Importantly, they use a fixed temperature hyperparameter τ controlling the steepness of the sigmoid function and thereby the softness of the whole decision tree. They further point out a key difference between traditional and differentiable decision trees: While the former are built greedily during training, the later have a fixed architecture where only the parameters used in split and leaf nodes are adapted during training.

Unlike traditional decision trees, the abovementioned formulations of differentiable decision trees make use of all dimensions of the input sample $\mathbf{x}$ at each split node. Note that a decision tree is much easier to interpret if just a single feature is used at each split though. Using just one feature per node is also the approach taken in [YMH18]. Their decision function for dimension d looks like $d_d(x_d) = \text{softmax}(\frac{w_d \cdot x_d + b_d}{\tau})$, where the softmax function allows for splits with more than just two child nodes. There is exactly one such decision function for each dimension $d \in D$ and the tree structure is built up via a Kronecker product of all their outcomes.

Follow-up work improving and extending the framework of differentiable decision trees exists. For instance, [HPM+20] present a tree ensemble layer with split functions relying on a novel smooth-step activation function instead of the sigmoid function. This function is similar to the sigmoid function around zero, but exactly takes on the values 0 and 1 outside of some symmetric interval around zero. Therefore, it allows for hard routing in a differentiable tree.

2.4 Prior Knowledge in Neural Networks

There is a rich amount of literature on the question of how to incorporate prior knowledge independent of the actual learning task into neural networks.

[VRMB+19] deliver a survey and taxonomy of what they call informed machine learning. The key idea of most related attempts is to combine data and knowledge to improve models, especially when data is scarce or the problem demands certain physical, regulatory or security constraints. According to [VRMB+19], there are three major sources of such prior knowledge:

- Scientific knowledge like known physical laws.
- World knowledge including facts that basically every human is aware of.
- Expert knowledge, which is often rather informal or intuitive and often based on personal experience like that of a doctor.

They further list eight different ways in which prior knowledge is usually formulated:

- The most important one for our work is algebraic equations, i.e. equalities or inequalities defining a feasible set for certain values.
- The other formats are differential equations, simulation results, spatial invariances, logic rules, knowledge graphs, probabilistic relations, and human feedback.

Finally, they list four distinct approaches how to get the prior knowledge into the model:

- By augmenting the training data.
- By constraining the hypothesis set like in convolutional neural networks in the case of image processing.
- By adapting the learning algorithm, e.g. by adding additional loss terms penalizing the output towards the prior knowledge.
- By manipulating the final hypothesis, e.g. by discarding predictions that are inconsistent with the prior knowledge.

We primarily assume to have expert knowledge in the format of algebraic equations and are particularly interested in using further loss terms to incorporate such knowledge. Two papers on physics-guided neural networks (PGNN) [DKW+17] and domain-adapted neural networks (DANN) [MIM+18] provide the main inspiration for our approach.

A PGNN combines a data-centric neural network model with existing physical knowledge in order to achieve better generalization and scientific consistency, particularly in small data settings. The following two ideas come into play:

- First, the PGNN uses the output of a traditional physics-based model as additional input, i.e. the PGNN takes $[\mathbf{x}, \hat{y}_{PHY}]$ as input to predict the target y.

- Second, the PGNN, which we denote by f_{PGNN}, uses additional physics-based loss terms. It minimizes $L(\hat{y}, y) + \lambda R(f_{PGNN}) + \lambda_{PHY} L_{PHY}(\hat{y})$, where $L(\hat{y}, y)$ is the task-specific loss function, $\lambda R(f_{PGNN})$ is some generic regularization term like an L2-penalty and $\lambda_{PHY} L_{PHY}(\hat{y})$ is the physics-based loss term. For known physical relationships in the form of $g(y, z) = 0$ and $h(y, z) \leq 0$, we can define $L_{PHY}(\hat{y}) = ||g(\hat{y}, z)||^2 + \text{ReLU}(h(\hat{y}, z))$.

[DKW+17] apply a PGNN to lake temperature modeling incorporating the known physical relationship between water density and temperature.

In contrast, the DANN is a more general idea. It leverages a similar idea trying to find the sweet spot between data and prior knowledge in situations where data is limited, expensive, or of poor quality. Again the optimization objective consists of a general loss term $L(\hat{y}, y)$, a general regularization term $\lambda R(f_{DANN})$ and a domain loss $\lambda_D L_D(\hat{y})$ similar to above. [MIM+18] distinguish between two different types of constraints:

- Approximation constraints of the form $g(\hat{y}) = \begin{cases} 0 & \hat{y} \in [y_l, y_u] \\ |y_l - \hat{y}| & \hat{y} < y_l \\ |y_u - \hat{y}| & \hat{y} > y_u \end{cases}$ can be incorporated as $L_D(\hat{y}) = \text{ReLU}(y_l - \hat{y}) + \text{ReLU}(\hat{y} - y_u)$. Approximation constraints are helpful when some expert knowledge demands certain predicted values to lie within some predefined range.

- Monotonicity constraints of the form $x_1 > x_2 \Rightarrow y_1 = h(x_1) > y_2 = h(x_2)$ can be incorporated as $L_D(\hat{y}_1, \hat{y}_2) = \mathbb{1}_{(x_1 < x_2) \wedge (\hat{y}_1 > \hat{y}_2)} \text{ReLU}(\hat{y}_1 - \hat{y}_2)$. The water temperature modeling approach in [DKW+17] is an instance of this approach.

3 Setting and Methods

This chapter introduces the ideas and methods investigated in our work. Section 3.1 describes from which real-world setting and which earlier work we draw our inspiration and presents the big picture of our work. Section 3.2 outlines in detail our framework of using decision trees to specify missingness processes in a unified format. In Section 3.3, we elaborate on the idea to use differentiable decision trees to incorporate prior knowledge as specified in Section 3.2. We derive the idea as a conretization of the work in [IMF20]. For the sake of completeness, Section 3.4 mentions reweighting as a trivial baseline approach to put more emphasis on observations with a low probability of being observed according to the known missingness process. Eventually, we introduce our prior knowledge formulation, which is a proxy for various possible occurences of expert knowledge in the real world, and provide two ideas, namely penalization and imputation, to incorporate such prior knowledge (Section 3.5).

3.1 Inspiration

As outlined in the last chapter, VAEs are a popular model class for unsupervised learning in various application domains with a wide range of purposes. [LROI21] develop an IWAE for electronic health records (EHR), however their focus is on the MNAR setting. In addition, [IMF20] improve the IWAE for MNAR tasks from a more general perspective, namely by jointly modeling the data and the missingness process in a recent work. They particularly stress the value of incorporating prior knowledge into the missingness model in order to improve the performance of the so-called not-MIWAE. Our work combines ideas from both papers, but in a somewhat different setting. We take the domain inspiration from the modeling of electronic health records or generally medical data similar to [LROI21] and aim at incorporating prior knowledge about the missingness process and/or the missing values into the model similar to [IMF20].

Unlike these two papers however, we specifically deal with a setting where data is missing at random (MAR). Imagine a doctor who conducts diagnostic tests, and depending on the results decides whether it is necessary to conduct further tests. If the doctor comes to the conclusion that no further tests are necessary, the unknown results for those further tests are missing at random since their missingness depends on the known values of the already conducted tests. In practice, we might even have access to further (expert) knowledge about the missing values, e.g. the doctor might know out of experience that if blood level A is above or below some threshold, the value of blood level B will usually lie within some interval $[a, b]$. Note that this example is just illustrative and the idea not restricted to the medical domain. In principle, our work applies to various settings where

a similar decision process is in place.

Since MAR mechanisms are ignorable in pure likelihood based methods [LR19], the emphasis in literature is often on the more challenging MNAR mechanism. We analyse and show that MAR missingness can still lead to unsatisfactory results, investigate what happens if we directly apply ideas from the not-MIWAE to MAR data, and to what extent we can improve a VAE with the help of prior knowledge in the presence of a MAR mechanism. Note that we do not develop a new model, but leverage existing ideas and methods in a very specific setting and propose and evaluate ideas to cope with this setting.

In the following, we present our missingness and our prior knowledge specification and how we try to incorporate those.

3.2 Missingness Formulation

Recall our graphical model in Figure 1.1 depicting the factorization

$$p(\mathbf{z}, \mathbf{x}, \mathbf{m}, \tilde{\mathbf{x}}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{m}|\mathbf{x})p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m}), \tag{3.1}$$

where $p(\mathbf{m}|\mathbf{x})$ is the missingness process. Unlike other work [Mar08], we do not additionally condition the distribution of the missingness mask $\mathbf{m}$ on the latent variables $\mathbf{z}$, i.e. there is no edge going directly from $\mathbf{z}$ to $\mathbf{m}$. The reason is that we assume the missingness process to be completely determined by $\mathbf{x}$ and therefore $p(\mathbf{m}|\mathbf{x}, \mathbf{z}) = p(\mathbf{m}|\mathbf{x})$.

We consider missingness processes in the form of a binary decision tree. Decision trees are a common format to depict the information acquisition process in diagnostic workup [ZBFA08] and easy to interpret but also to design for a human (expert). This idea is inspired by [THHM15], where the authors use decision trees as machine learning models to investigate the missingness structure in a dataset. Unlike them, we only rely on decision trees as a way of formalizing the missingness process and do not use decision trees as machine learning models at this stage. For the variables of interest $\mathbf{x} = [x_1, \ldots, x_D]$ we have a collection of binary decision trees $\mathbf{t} = [t_1, \ldots, t_D]$ of corresponding dimension. Each t_d with $d \in \{1, \ldots, D\}$ is a binary decision tree specifying the missingness mechanism for x_d, or equal to $\perp$ if no such knowledge is available. Each leaf of tree t_d contains the missingness probability $p(m_d = 1)$ for a sample routed to that leaf.

Note that binary decision trees in principle provide a highly flexible framework for specifying missingness processes. Figure 3.1 demonstrates possible missingness formulations of varying complexity. The leftmost tree shows a simple example, where the missingness of x_7 depends on variable x_2 with $p(m_7 = 1|x_2 \leq 3) = 0.95$ and $p(m_7 = 1|x_2 > 3) = 0.1$. In the second example, the missingness of x_7 still just depends on x_2, however in a more complicated fashion with $p(m_7 = 1|x_2 \leq 3) = 0.95$, $p(m_7 = 1|x_2 \in (3, 10)) = 0.1$ and $p(m_7 = 1|x_2 \geq 10) = 0.2$. The third example, which we do not explain at length, shows a process where the missingness is determined by an interaction between x_2 and x_4, where x_4 is categorical.

Our formulation extends the simple example provided in [IMF20], where a self-censoring missingness process is presented as simple MNAR example. Via decision trees, we can

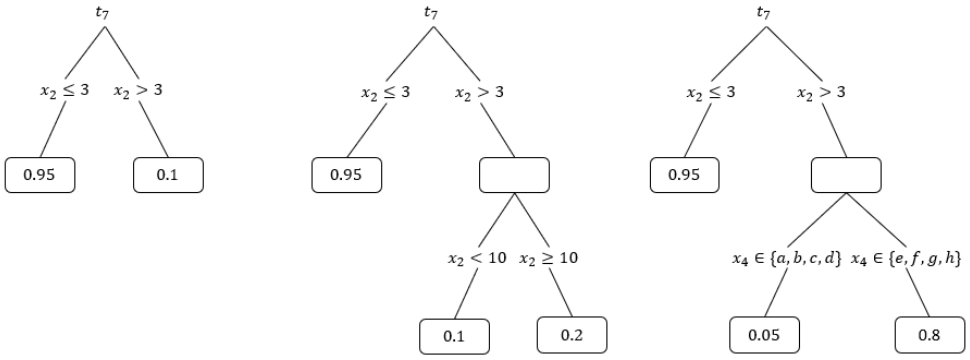


Figure 3.1: Examples of missingness mechanisms for variable x_7 in the format of a binary decision tree.

easily specify MCAR, MAR and MNAR processes. The MNAR process in Figure 3.2 for example is an instance of a self-censoring MNAR mechanism.

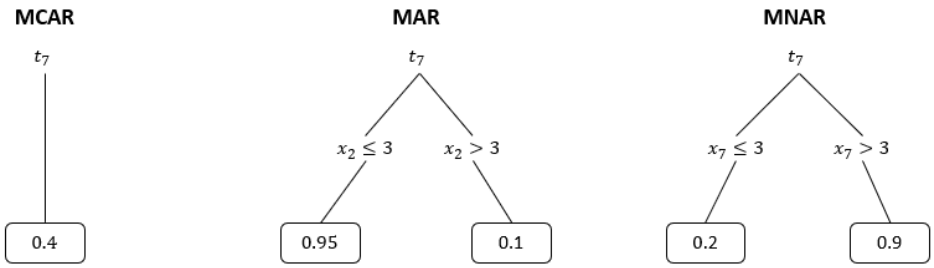


Figure 3.2: Possible MCAR, MAR and MNAR missingness mechanisms for x_7 in the format of a binary decision tree.

In our experiments, we only consider very simple missingness processes, where the underlying tree features a single split.

3.3 Differentiable Decision Trees

To incorporate (knowledge about) the missingness process into an IWAE, [IMF20] optimize a lower bound of the joint likelihood $p(\mathbf{x}_O, \mathbf{m})$ via the following objective:

$$\mathcal{L}^K_{\theta,\phi,\gamma}(\mathbf{x}_O, \mathbf{m}) = \mathbb{E}\left[\log \frac{1}{K}\sum_{k=1}^{K} w^{(k)}\right], \tag{3.2}$$

where

$$w^{(k)} = \frac{p_\gamma(\mathbf{m}|\mathbf{x}_O, \hat{\mathbf{x}}_U^{(k)}) p_\theta(\mathbf{x}_O|\mathbf{z}^{(k)}) p_\theta(\mathbf{z}^{(k)})}{q_\phi(\mathbf{z}^{(k)}|\mathbf{x}_O)}, \tag{3.3}$$

and $\mathbf{z}^{(k)} \sim q_\phi(\mathbf{z}^{(k)}|\mathbf{x}_O)$ and $\hat{\mathbf{x}}_U^{(k)} \sim p_\theta(\mathbf{x}_U|\mathbf{z}^{(k)})$. We can derive a similar formulation for a VAE instead of an IWAE:

$$p(\mathbf{x}_O, \mathbf{m}) = \int p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) \cdot p(\mathbf{x}_O|\mathbf{z}) \cdot p(\mathbf{x}_U|\mathbf{z}) \cdot p(\mathbf{z}) \; d\mathbf{x}_U \; d\mathbf{z} \tag{3.4}$$

$$= \int p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) \cdot p(\mathbf{x}_O|\mathbf{z}) \cdot p(\mathbf{x}_U|\mathbf{z}) \cdot p(\mathbf{z}) \cdot \frac{q(\mathbf{z}|\mathbf{x}_O)}{q(\mathbf{z}|\mathbf{x}_O)} \; d\mathbf{x}_U \; d\mathbf{z} \tag{3.5}$$

$$= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_O), p(\mathbf{x}_U|\mathbf{z})} \left[p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) \cdot p(\mathbf{x}_O|\mathbf{z}) \cdot \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_O)} \right] \tag{3.6}$$

$$\log p(\mathbf{x}_O, \mathbf{m}) = \log \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_O), p(\mathbf{x}_U|\mathbf{z})} \left[p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) \cdot p(\mathbf{x}_O|\mathbf{z}) \cdot \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}_O)} \right] \tag{3.7}$$

$$\geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_O), p(\mathbf{x}_U|\mathbf{z})} [\log p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U) + \log p(\mathbf{x}_O|\mathbf{z})] - D_{KL}[q(\mathbf{z}|\mathbf{x}_O)||p(\mathbf{z})] \tag{3.8}$$

$$= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_O)} [\log p(\mathbf{x}_O|\mathbf{z})] - D_{KL}[q(\mathbf{z}|\mathbf{x}_O)||p(\mathbf{z})] + \mathbb{E}_{q(\mathbf{z}|\mathbf{x}_O), p(\mathbf{x}_U|\mathbf{z})} [\log p(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U)], \tag{3.9}$$

where the inequality is because of Jensen's inequality and we can use the factorization $p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x}_O|\mathbf{z}) \cdot p(\mathbf{x}_U|\mathbf{z})$ if we assume $p(\mathbf{x}|\mathbf{z}) = \prod_{d=1}^{D} p(x_d|\mathbf{z})$ which is a common assumption in the VAE literature. Considering a single-sample approximation with a given sample with observed values $\mathbf{x}_O^{(i)}$, the objective function becomes

$$\mathcal{L}_{\theta,\phi,\gamma}(\mathbf{x}_O^{(i)}, \mathbf{m}^{(i)}) = \log p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z}^{(i)}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})||p_\theta(\mathbf{z})) + \log p(\mathbf{m}^{(i)}|\mathbf{x}_O^{(i)}, \hat{\mathbf{x}}_U^{(i)}), \tag{3.10}$$

where $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})$ and $\hat{\mathbf{x}}_U^{(i)} \sim p_\theta(\mathbf{x}_U|\mathbf{z}^{(i)})$. This means that $\mathbf{x}_O^{(i)}$ is encoded and a sample $\mathbf{z}^{(i)}$ is drawn from the approximate posterior distribution. This sample from the latent space in turn is decoded to reconstruct $\mathbf{x}_O^{(i)}$ and generate $\mathbf{x}_U^{(i)}$. $\mathbf{x}_O^{(i)}$ and the generated $\hat{\mathbf{x}}_U^{(i)}$ are then input to the missingness model $p_\gamma(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_U)$ to reconstruct the missingness mask. If the true missingness model is (partially) known, the errors propagating back from $\log p(\mathbf{m}^{(i)}|\mathbf{x}_O^{(i)}, \hat{\mathbf{x}}_U^{(i)})$ can penalize the generation of $\hat{\mathbf{x}}_U$ to become consistent with the missingness pattern and thereby improve the generation quality.

Note that as a complementary approach, we also try to optimize a somewhat adapted objective function:

$$\mathcal{L}_{\theta,\phi,\gamma}(\mathbf{x}_O^{(i)}, \mathbf{m}^{(i)}) = \log p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z}^{(i)}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})||p_\theta(\mathbf{z})) + \log p(\mathbf{m}^{(i)}|\hat{\mathbf{x}}_O^{(i)}, \hat{\mathbf{x}}_U^{(i)}), \tag{3.11}$$

where $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})$, $\hat{\mathbf{x}}_U^{(i)} \sim p_\theta(\mathbf{x}_U|\mathbf{z}^{(i)})$ and $\hat{\mathbf{x}}_O^{(i)} \sim p_\theta(\mathbf{x}_O|\mathbf{z}^{(i)})$. The difference is that this time the reconstructions $\hat{\mathbf{x}}_O^{(i)}$ are fed into the missingness model instead of the true $\mathbf{x}_O^{(i)}$. Consequently, a stronger gradient flow is possible judging the consistency of the whole decoder result. More precisely, gradients are additionally allowed to propagate back via $\hat{\mathbf{x}}_O^{(i)}$ which is not possible if we only feed $\mathbf{x}_O^{(i)}$ into the missingness model. The inspiration comes from domain-adapted and physics-guided neural networks, which also improve the model by assessing the consistency of the model output.

Since we formulate our missingness mechanisms as decision trees, a natural choice for the implementation of a missingness model as part of a VAE is a differentiable decision tree as described in Section 2.3. For instance, we would implement the MAR missingness process in Figure 3.2 as

$$p(\mathbf{m}|\mathbf{x}) = 0.1 \cdot \sigma\left(\frac{x_2 - 3}{\tau}\right) + 0.95 \cdot \left(1 - \sigma\left(\frac{x_2 - 3}{\tau}\right)\right). \tag{3.12}$$

Each split node (in this example there is only one) takes exactly one feature as input, the whole missingness model is frozen and a temperature parameter τ controls the steepness of the sigmoid function performing the soft routing.

Note that the whole idea of making use of the missingness process in a similar manner as the not-MIWAE is problematic in the MAR setting. Unlike MNAR, MAR missingness does not reveal any immediate information about the missing variable. For instance, for the process $p(m_7 = 1|x_2 \leq 3) = 0.95$ and $p(m_7 = 1|x_2 > 3) = 0.1$, the missingness mask of x_7 only depends on x_2 and therefore only reveals information about x_2. Since x_2 has to be observed when x_7 is missing in MAR, we get $p(x_7|\mathbf{x}_O, \mathbf{m}) = p(x_7|\mathbf{x}_O)$ for $7 \in \mathcal{U}$. This shortcoming motivates other approaches than just incorporating knowledge about the missingness process, particularly incorporating prior knowledge on the dependency between x_7 and x_2, as elaborated on in Section 3.5.

3.4 Reweighting

Even though m_7 does not reveal any immediate information about x_7 in our above MAR example, the knowledge about the missingness mechanism can still help to improve the performance via sample weights. [LR19] mention weighting as one approach to deal with missing data. Using tree t_7, we can easily evaluate $p(m_7|x_2)$ as long as x_2 is observed. Assuming both x_2 and x_7 are observed, we can compute a weight based on the probability of x_7 being unobserved for the recorded x_2 to put more weight on the reconstruction error of x_7 if x_2 lies in a region where x_7 is likely to be missing.

Example. Imagine we have two samples $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ with x_2 and x_7 observed under the MAR process in Figure 3.2. Assume $x_2^{(i)} = 2$ and $x_2^{(j)} = 4$, then $p(m_7^{(i)}|x_2^{(i)}) = 0.95$ and $p(m_7^{(j)}|x_2^{(j)}) = 0.1$. We can use the weights $w_7^{(i)} = \frac{1}{1-0.95} = 20$ and $w_7^{(j)} = \frac{1}{1-0.1} \approx 1.11$ to put more emphasis on sample i, which lies in a region where x_7 is unobserved in 95% of all cases. If missingness only occurs in a single feature, which is unrealistic in practice, we could use sample weights. Otherwise, if missingness occurs in multiple features, we can use dimension wise weights to increase the reconstruction error for single dimensions of a sample.

Note that there is no guarantee that x_2 is observed when x_7 is observed though. Our MAR setting demands x_2 to be observed whenever x_7 is unobserved since the missingness of x_7 is decided on based on the recorded value of x_2. However, x_2 might itself be unobserved due to some other reason. Therefore, one of the downsides of this method is that for it to work we need to have some samples with both x_2 and x_7 observed with

the value of x_2 lying in a region where x_7 often remains unobserved. We will investigate reweighting as a baseline idea next to other approaches.

3.5 Prior Knowledge

3.5.1 Formulation

Recall that in the medical setting providing our inspiration, we do not only assume knowledge about the missingness mechanism, but also a rough estimation for the missing values themselves. We aim at using such knowledge to alleviate problems with missing at random data in a VAE. According to the taxonomy in [VRMB+19] the source could be expert knowledge based on professional experience but also (weak) scientific knowledge about the dependency between two (or more) variables. Different formalizations of such knowledge are possible, but we restrict ourselves to a formulation based on algebraic equations, or more precisely a simple linear regression.

Example. In our MAR example from Figure 3.2 where the missingess of x_7 depends on x_2, imagine that we have prior knowledge regarding x_7 in the form of a simple linear regression with x_2 as explanatory variable, i.e. $x_7 = a + b \cdot x_2 + \varepsilon$ with $\varepsilon \sim N(0, \sigma^2)$. Assume that this dependency is at least reasonably accurate in the area where x_7 is typically unobserved. Then, we have access to the conditional expectation $\mathbb{E}[x_7|x_2]$ or in a more realistic scenario to a prior knowledge interval $[a + b \cdot x_2 - \sigma, a + b \cdot x_2 + \sigma]$ for x_7. This interval symbolizes a rough expert assessment of what the missing values are like. It is similar to the intervals presented in [MIM+18] to penalize neural networks via approximation constraints.

This specification resembles two things:

1. It is similar to the expert knowledge a doctor might have based on his or her experience regarding the relationship between two blood levels. Actually, this exact knowledge might be the reason not to record x_7 given x_2 since the value of x_7 is approximately determined given x_2.

2. Furthermore, it is basically an oversimplified model for x_7 similar to traditional scientific models that neglect unknown dynamics and are therefore less flexible than modern data-driven models.

Note that our linear regression formulation of prior knowledge comes with a couple of problems. In practice, it might not be straightforward to formalize expert knowledge. In addition, a model as simple as a linear regression with a single explanatory variable can fall victim to a heavy omitted variable bias. Nevertheless, the idea can be easily extended to a more complicated format alleviating problems under the current specification.

3.5.2 Penalty

We are interested in comparing different ways of incorporating prior knowledge as specified in the previous section into a VAE. Our first idea is inspired by the approximation

constraints in domain-adapted neural networks (DANN) [MIM+18]. Therefore, we add an extra penalty term to the loss function to penalize generations of the unobserved variables[1] outside their respective prior knowledge interval. Denote the prior knowledge interval for variable x_d by $\text{PI}_d = [x_{(d,l)}, x_{(d,u)}]$, e.g. the interval for x_7 in the example above becomes $\text{PI}_7 = [x_{(7,l)}, x_{(7,u)}] = [a + b \cdot x_2 - \sigma, a + b \cdot x_2 + \sigma]$. Then, the objective function for sample $\mathbf{x}^{(i)}$ becomes

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}_O^{(i)}) = \log p_\theta(\mathbf{x}_O^{(i)}|\mathbf{z}^{(i)}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})||p_\theta(\mathbf{z})) - \lambda \cdot \sum_{d \in \mathcal{U}^{(i)}} \Omega_d(\hat{x}_d), \tag{3.13}$$

where $\mathbf{z}^{(i)} \sim q_\phi(\mathbf{z}|\mathbf{x}_O^{(i)})$, $\hat{x}_d \sim p_\theta(x_d|\mathbf{z}^{(i)})$, λ is a hyperparameter controling the strength of the penalty and

$$\Omega_d(\hat{x}_d) = \text{ReLU}(\hat{x}_d - x_{(d,u)}) + \text{ReLU}(x_{(d,l)} - \hat{x}_d). \tag{3.14}$$

Obviously, we might not have access to prior knowledge in this form for every $d \in \mathcal{U}^{(i)}$, but the idea is also appicable to any subset of $\mathcal{U}^{(i)}$. Instead of using the prior knowledge about the missingness process for regularization like in the not-MIWAE [IMF20], the prior knowledge penalty regularizes a VAE via the prior knowledge with respect to the actual values of the unobserved variables. It therefore combines ideas from DANN and not-MIWAE.

3.5.3 Imputation

Another approach to carry out informed machine learning is by augmenting the training data [VRMB+19]. While we do not augment our training data in the sense that we generate new modified samples, we consider imputation of missing values based on our prior knowledge in order to make the unobserved values in $\tilde{\mathbf{x}}$ more realistic. To this end, let $\tilde{\tilde{\mathbf{x}}}$ denote an imputed version of $\tilde{\mathbf{x}}$ where missing values have been replaced by their conditional expectation. In our earlier example, we would set $\tilde{\tilde{x}}_7$ to $\mathbb{E}[x_7|x_2]$ if $7 \in \mathcal{U}$. We investigate three different strategies to utilize $\tilde{\tilde{\mathbf{x}}}$:

1. Just using $\tilde{\tilde{\mathbf{x}}}$ as encoder input, i.e. modeling the approximate posterior as $q_\phi(\mathbf{z}|\tilde{\tilde{\mathbf{x}}})$. Our hypothesis is that the more realistic conditional expectations in $\tilde{\tilde{\mathbf{x}}}$ lead to less distortion than the zeros in $\tilde{\mathbf{x}}$.

2. Using $\tilde{\tilde{\mathbf{x}}}$ as noisy target changing the reconstruction term in the ELBO objective to $\log p(\tilde{\tilde{\mathbf{x}}}|\mathbf{z})$ instead of $\log p(\mathbf{x}_O|\mathbf{z})$ and thereby guiding the generation of the VAE with approximately correct targets during training.

3. Combining these two approaches.

[1] only when actually unobserved

4 Experimental Setup

4.1 Datasets

4.1.1 Simulated Data

Data Generation

We developed our ideas on simulated data and evaluate them both on simulated data and a real-life electronic health record (EHR) dataset. We generate the simulated data from a latent variable model with $\mathbf{z} \in \mathbb{R}^2 \sim N(\mathbf{0}, \mathbf{I})$ and $\mathbf{x} = 10 \cdot \sigma(\mathbf{W}\mathbf{z} + \mathbf{b}) + \boldsymbol{\varepsilon}$ with $\mathbf{W} \in \mathbb{R}^{10\times 2}$, $\mathbf{b} \in \mathbb{R}^{10}$ and $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{I})$, i.e. $\mathbf{x} \in \mathbb{R}^{10}$. The matrix $\mathbf{W}$ and the vector $\mathbf{b}$ are fixed throughout data generation and initialized by sampling all their elements from a $U(-\sqrt{\frac{1}{2}}, \sqrt{\frac{1}{2}})$ uniform distribution. This data-generating process is deliberately simple and the distribution of the data should be easy to learn even with moderately flexible models. Note that all synthetic results are generated for a single fixed data generating process. This setup limits the reliability of our results.

Data Splitting

For the final evaluation, we rely on four datasets generated from the described process:

1. A prior knowledge set of size $n = 1000000$. We explain its main purpose in Section 4.1.1 below. In addition, we use the statistics computed from this massive dataset to standardize itself and the other three datasets to mean 0 and variance / standard deviation 1. This procedure is a straightforward alternative to directly generating data with mean 0 and variance 1 in every coordinate.
2. A test set of size $n = 10000$. We derive all our final results on this test set. $n = 10000$ on the one hand guarantees representative results and on the other hand leads to clear scatterplots without heavy overplotting.
3. A training set of size $n = 1000$.
4. A second training set of size $n = 10000$ to demonstrate differences between a small and a medium sized training set.

We generated extra data for prototyping our ideas and figuring out reasonable hyper-parameters from the same data generating process in order not to misuse the test set for validation purposes. Table 4.1 presents the correlation matrix computed on the test set. Note that from now on we use a zero-based index for variables in accordance with the Python programming language.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_0	1.00	-0.34	-0.04	0.32	0.35	-0.55	0.24	-0.01	0.52	-0.63
x_1	-0.34	1.00	-0.42	0.20	-0.18	0.41	0.24	0.33	-0.05	0.35
x_2	-0.04	-0.42	1.00	-0.60	-0.02	-0.16	-0.59	**-0.51**	-0.37	0.01
x_3	0.32	0.20	-0.60	1.00	0.18	-0.11	0.59	0.42	0.55	-0.29
x_4	0.35	-0.18	-0.02	0.18	1.00	-0.30	0.13	0.00	0.29	-0.35
x_5	-0.55	0.41	-0.16	-0.11	-0.30	1.00	-0.05	0.15	-0.34	0.55
x_6	0.24	0.24	-0.59	0.59	0.13	-0.05	1.00	0.42	0.48	-0.21
x_7	-0.01	0.33	**-0.51**	0.42	0.00	0.15	0.42	1.00	0.25	0.03
x_8	0.52	-0.05	-0.37	0.55	0.29	-0.34	0.48	0.25	1.00	-0.49
x_9	-0.63	0.35	0.01	-0.29	-0.35	0.55	-0.21	0.03	-0.49	1.00

Table 4.1: Correlation matrix for the simulated test set with $n = 10000$ observations.

Missingness Specification

We define both a simple MAR and a simple MNAR mechanism for a single variable depicted in Figure 4.1.

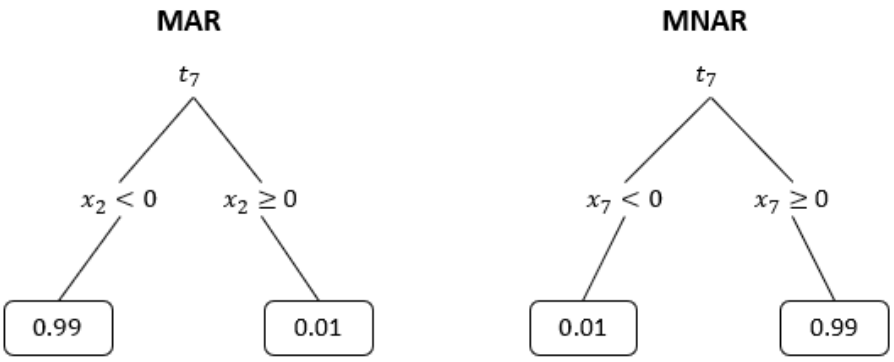
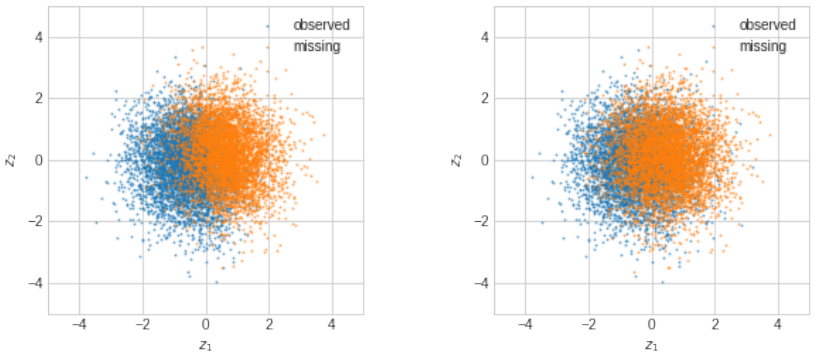


Figure 4.1: Assumed MAR and MNAR mechanisms for experiments with simulated data. x_7 is the only corrupted variable, i.e. the only variable affected by missingness.

The MNAR mechanism is a simple self-censoring process for variable x_7. We aim at reproducing some of the results in the not-MIWAE paper [IMF20] for this process. However, our primary interest and the focus of our work is the MAR mechanism. In this process, we purposely define the missingness of x_7 to depend on x_2 since these two variables have a relatively strong (negative) correlation making them well-suited for our assumed setting. Figure 4.2 visualizes the latent space of the test set with all samples featuring missingness in variable x_7 colored in orange.
Likewise, the scatterplots in Figure 4.3 visualize x_7 versus x_2 clarifying the difference between the MAR and the MNAR mechanism.
Note that we used 0 as splitting point for our missingness specifications to ensure that



(a) MAR (b) MNAR

Figure 4.2: Latent space from which the test set was generated. Orange samples have missingness in variable x_7, blue samples are fully observed after applying the missingness process.

approximately half of the values in x_7 are unobserved for every specification. In fact, there are 4780 unobserved values in the test set in the MAR case and 5012 in the MNAR case.

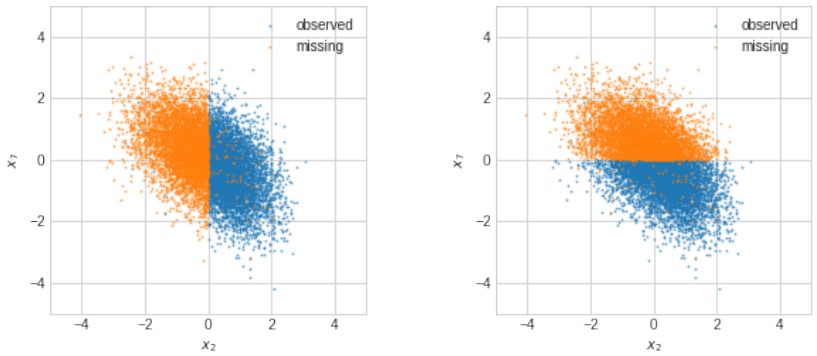
Prior Knowledge Specification

Since we are interested in settings where we have access to some prior knowledge about the missing values in a MAR setting, we make use of what we call a prior knowledge set to imitate independent prior knowledge. On this set, we fit a simple linear regression with x_7 as target variable and x_2 as explanatory variable together with an intercept term. Figure 4.4 shows the corresponding prior knowledge interval for x_7 on the left-hand side and an imputed version of the test set on the right-hand side. Refer to Section 3.5 for detailed information on the prior knowledge interval. Note that we use an estimate $\hat{\sigma}$ for σ to compute it.
Our assumption is that such prior knowledge is useful to guide the model when only a small dataset is available for training.

4.1.2 EHR Dataset

Dataset Description

Our real-world dataset [Sad20] is an EHR dataset originally intended for patient treatment classification with laboratory results from a hospital in Indonesia. It contains $n = 4412$ rows (one row per treatment) and 11 columns without any missing values. The columns comprise eight different blood levels (haematocrit, haemoglobins, erythrocyte, leucocyte, thrombocyte, mch, mchc, mcv), the age and the sex of the patient and the binary target variable, which is either “in” (in care treatment) or “out” (out care treatment). Since we



(a) MAR (b) MNAR

Figure 4.3: Scatterplots of x_7 versus x_2 for the test set. Orange samples have missingness in variable x_7, blue samples are fully observed after applying the missingness process. Under MAR the missingness of x_7 is determined by x_2, in MNAR it is determined by x_7 itself.

are not directly interested in classification, we only use the first nine columns, i.e. the blood levels and the age of a patient.

Data Splitting

Similar to before, we split the dataset into different subsets (after shuffling):

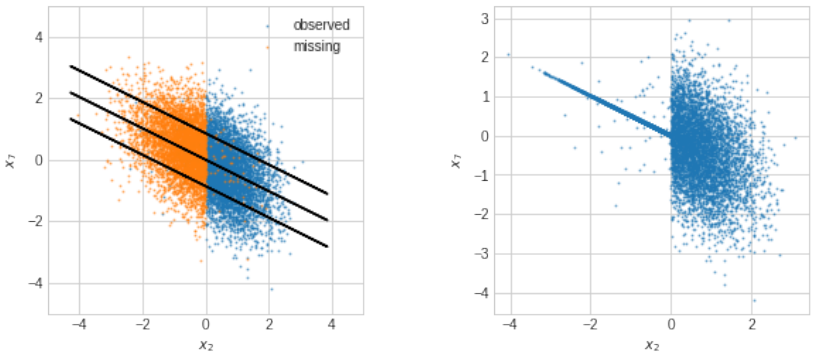
1. A prior knowledge set of size $n = 2412$. We use it for the same purpose as before, i.e. for standardizing all subsets to mean 0 and variance 1 and particularly for imitating prior knowledge.
2. A test set of size $n = 1000$ to evaluate all models.
3. 10 training sets of size $n = 100$ each. Since the dataset is a very simple one, we need to keep the size of the training set low. This splitting scheme also allows us to report the average of 10 results for every approach we investigate.

Table 4.2 presents the correlation matrix computed on the test set.

Missingness Specification

We define a MAR process corrupting four variables (x_1, x_2, x_6, x_7). The missingness of all of these four variables is determined by two variables (x_0, x_5) as depicted in Figure 4.5.

The variables x_0, x_1 and x_2 have strong correlations as have the variables x_5, x_6 and x_7. Figure 4.6 shows the test set $(n = 1000)$ after application of this missingness process. There are in total 486 missing values in x_1, 515 in x_2, 394 in x_6 and 407 in x_7. 221 samples in the test set are fully observed, 182 have one unobserved value, 329 have two, 110 have three and in 158 samples all of $\{x_1, x_2, x_6, x_7\}$ are unobserved.



(a) Prior knowledge interval (b) Prior knowledge imputation

Figure 4.4: Scatterplots of x_7 versus x_2 for the test set. (a) shows the prior knowledge interval bounded by the outer black lines and the conditional expectation of x_7 given by the center black line. (b) shows the test set where corrupted values in x_7 are replaced by the conditional expectation.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_0	1.00	**0.98**	**0.87**	-0.21	-0.01	0.16	0.13	0.13	-0.22
x_1	**0.98**	1.00	0.82	-0.20	-0.06	0.27	0.33	0.17	-0.20
x_2	**0.87**	0.82	1.00	-0.18	0.02	-0.31	-0.02	-0.36	-0.40
x_3	-0.21	-0.20	-0.18	1.00	0.30	-0.02	0.02	-0.03	0.15
x_4	-0.01	-0.06	0.02	0.30	1.00	-0.14	-0.20	-0.08	0.03
x_5	0.16	0.27	-0.31	-0.02	-0.14	1.00	**0.58**	**0.93**	0.37
x_6	0.13	0.33	-0.02	0.02	-0.20	**0.58**	1.00	0.25	0.04
x_7	0.13	0.17	-0.36	-0.03	-0.08	**0.93**	0.25	1.00	0.42
x_8	-0.22	-0.20	-0.40	0.15	0.03	0.37	0.04	0.42	1.00

Table 4.2: Correlation matrix for the treatment test set with $n = 1000$ observations.

Prior Knowledge Specification

Once again, we generate prior knowledge by means of a simple linear regression fit on the knowledge set ($n = 2412$). The resulting conditional expected value and prior knowledge intervals are visualized in Figure 4.7. Note that the prior knowledge for x_1 and x_7 is almost too good to be realistic due to their strong correlation with their explanatory variables x_0 and x_5.

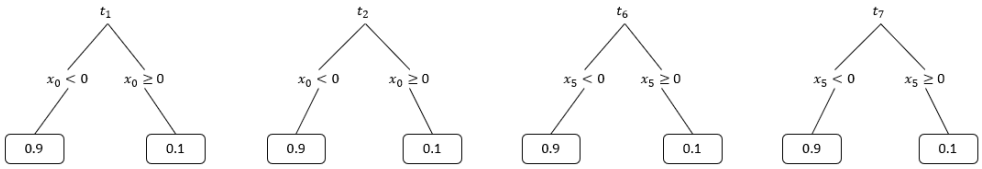


Figure 4.5: Assumed MAR mechanisms for experiments with treatment data. Up to four variables can be unobserved.

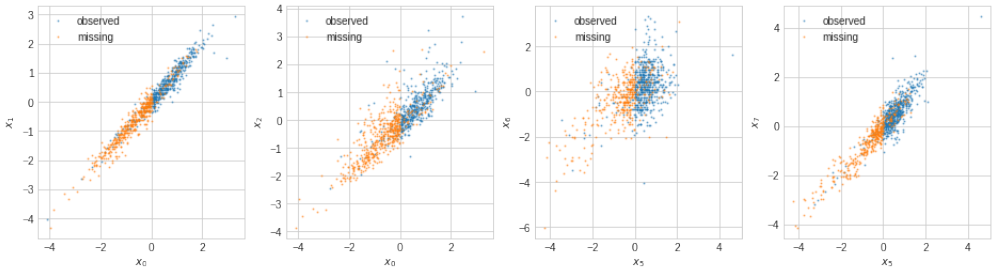


Figure 4.6: Test set scatterplots of the variables with missingness versus their missingness determining variables. Orange samples have missingness in the variable on the y-axis, blue samples are fully observed after applying the missingness process.

4.2 Models

4.2.1 Benchmarks and Baselines

1. Vanilla VAE (VAE): We fit a standard variational autoencoder to all training samples before applying the missingness process, i.e. the model has access to the true values $\mathbf{x}$ instead of the corrupted $\tilde{\mathbf{x}}$. This is an unrealistic benchmark for the corrupted case since this model has access to information that we assume to be unknown in our setting.

2. Complete-case analysis (CC): We fit a standard variational autoencoder to all samples that are still fully observed after applying the missingness process to find out what implications the missingness specification has. In addition, we repeat this experiment with sample weights (CC (W)) to compensate for the data scarcity in some regions, and with a subsampled dataset to directly train on a balanced dataset (CC (S)).

3. Masked VAE (MASKED): We disregard attributes in the encoder that are not always observed, e.g. x_7 for the simulated datasets. Therefore, the encoder becomes $q_\phi(\mathbf{z}|\mathbf{x}_{O_{all}})$ and $\mathcal{O}_{all}$ defines the index set for which no missingness mechanism is specified. The masked VAE still generates all dimensions of the data but only computes a reconstruction loss for the observed parts $\mathbf{x}_O$. On simulated data, this model is supposed to learn to generate x_7 from the other variables and serves as a

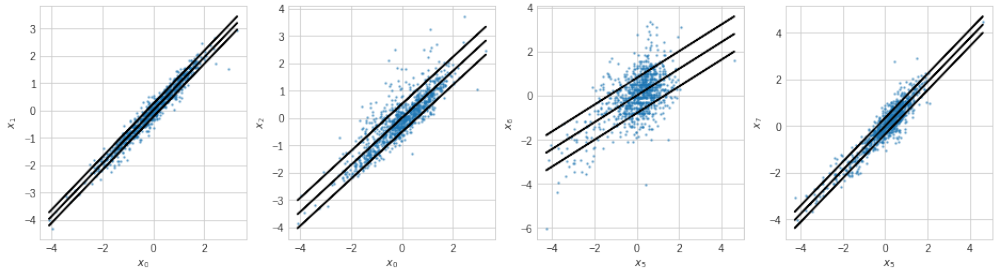


Figure 4.7: Test set scatterplots with prior knowledge intervals for the treatment data.

benchmark in this regard. Note that this model is not practical since it does not use x_7 as encoder input when actually observed and might end up without any inputs in practice.

4.2.2 Models with Corrupted Input

1. Zero-VAE (ZERO): A variational autoencoder that directly uses the corrupted (i.e. zero imputed) $\tilde{\mathbf{x}}$ as input and only computes a reconstruction loss for the observed parts $\mathbf{x}_O$. This specification is basically equivalent to the model proposed in [NOGV20]. Note that in our case with standardized features, zero imputation is equivalent to mean imputation.

2. M-VAE (M): Is identical to the Zero-VAE but additionally takes the missingess mask $\mathbf{m}$ as input leading to $q_\phi(\mathbf{z}|\tilde{\mathbf{x}}, \mathbf{m})$ to be able to distinguish between observed zeros and imputed zeros. It resembles the EO specification in [CNW20].

3. Partial VAE (PVAE): Optimizes the same objective as the Zero-VAE and the M-VAE but uses a permutation invariant encoder inspired by the partial VAE in [MTP$^+$18]. To encode a sample $(\tilde{\mathbf{x}}, \mathbf{m})$, we feed each dimension $\tilde{x}_d$ into a dimension-specific network $h_d(\tilde{x}_d)$ with two fully connected layers leading to D representations $[h_1(\tilde{x}_1), \ldots, h_D(\tilde{x}_D)]$ of equal size. We apply a permutation invariant function $g(\cdot)$ (either max-pooling or average-pooling) to this collection to obtain $\mathbf{c}(\mathbf{x}_O) = g([h_1(\tilde{x}_1), \ldots, h_D(\tilde{x}_D)])$, where dimensions d with $m_d = 1$ are neglected in $g(\cdot)$. $\mathbf{c}(\mathbf{x}_O)$ is then mapped to a hidden layer of equal size and finally to the statistics of the approximate posterior distribution of the latent space. We try both using individual subnetworks $h_d(\tilde{x}_d)$ (PVAE) for each attribute and a shared subnetwork $h([\tilde{x}_d, \mathbf{e}_d])$ (SPVAE) with a one-hot encoding e_d of d as additional input. Applying maximum and average pooling in both specifications leads to a total of four models: PVAE (MAX), PVAE (AVG), SPVAE (MAX), SPVAE (AVG).

4.2.3 Missingness Models

We equip both the Zero-VAE and the M-VAE with a missingness model similar to the not-MIWAE [IMF20]. We try two different flavors, one where the input to the missingness

model $p(\mathbf{m}|\mathbf{x}_O, \hat{\mathbf{x}}_U)$ is the observed parts $\mathbf{x}_O$ and decoder generations $\hat{\mathbf{x}}_U$ (U) for the unobserved parts and another one where the input to the model $p(\mathbf{m}|\hat{\mathbf{x}}_O, \hat{\mathbf{x}}_U)$ (OU) consists of the full decoder output. On top, we try both a generic missingness model (G) comprising two fully-connected layers and the true missingness model (T) implemented as frozen differentiable decision tree. These variations lead to eight different specifications: ZERO (GU), ZERO (GOU), M (GU), M (GOU), ZERO (TU), ZERO (TOU), M (TU), M(TOU).

4.2.4 Weighted Models

In our MAR settings, if $\mathbf{p} = [p(m_1 = 1|\mathbf{x}_O), \ldots, p(m_D = 1|\mathbf{x}_O)]$, we can use $\frac{1}{1-\mathbf{p}}$ as dimension wise weights for the losses in the reconstruction part of the ELBO defining the models ZERO (DW) and M (DW). In our concrete experimental specifications, we can always evaluate $\mathbf{p}$ since the variables deterimining the missingness mechanisms are always observed. Note that this is not generally the case in our assumed setting though.

4.2.5 Penalized Models

In the models ZERO (P) and M (P), we penalize generations of $\hat{\mathbf{x}}_U$ outside of the bounds of their prior knowledge interval, which we compute based on $\hat{\mathbf{x}}_O$. The strength of this regularization is controlled by the hyperparameter λ.

4.2.6 Models with Imputation

Finally, we try different types of imputation based on our prior knowledge:

- Using imputation in the encoder only (EO) leading to $q_\phi(\mathbf{z}|\tilde{\tilde{\mathbf{x}}}, \mathbf{m})$ (M (EO)) and $q_\phi(\mathbf{z}|\tilde{\tilde{\mathbf{x}}})$ (ZERO (EO)).
- Using imputation for the targets only replacing the $\log p(\mathbf{x}_O|\mathbf{z})$ term in the ELBO by $\log p(\tilde{\tilde{\mathbf{x}}}|\mathbf{z})$ in the specification M (TO).
- Using a standard variational autoencoder with $\tilde{\tilde{\mathbf{x}}}$ as encoder input and decoder target (VAE (ET)).

Note that we do not train all specified models for all datasets and missingness specifications.

4.3 Hyperparameters

To manually figure out reasonable hyperparameters (as well as for prototyping our ideas), we used separately generated data for the simulated datasets from the same data generating process as the datasets used in our final evaluation. For the patient treatment dataset, we used the prior knowledge set for validation. We always checked learning curves to spot severe over- and underfitting and made sure that all latent space

dimensions approximately followed a standard normal distribution as demanded by the prior $p_\theta(\mathbf{z}) = N(\mathbf{0}, \boldsymbol{I})$. Note that we did not direclty optimize the ELBO, but instead minimized the mean squared error (MSE) instead of the negative Gaussian log-likelihood and similar to a beta-VAE [HMP+16] multiplied the Kullback-Leibler divergence with a hyperparameter β to ensure a reasonable regularization strength. Except for the partial VAE, we used symmetric encoders and decoders in terms of their architecture. You can find a detailed overview over the chosen hyperparameters in Table 4.3.

Hyperparameter	Value
hidden layers	1 (except (S)PVAE)
hidden neurons	20, (S)PVAE: 40, missingness model: 10
latent space dimension	4
activation function	sigmoid (simulated), ReLU (treatment, (S)PVAE)
epochs	100 ($n = 1000$), 50 ($n = 10000$), 200 (treatment)
learning rate	0.01 (multiplied by 0.1 after half of the epochs)
batch size	8 ($n = 1000$), 32 ($n = 10000$), 10 (treatment)
D_{KL} weight β	0.01
mask reconstruction weight	1
prior knowledge penalty weight λ	0.1 (simulated), 1 (treatment)
temperature τ	0.1

Table 4.3: Hyperparameters used in experiments. PVAE and SPVAE models are trained for twice as many epochs on simulated data, i.e. 200 ($n = 1000$) and 100 ($n = 10000$) epochs. We also train for more epochs in the case of complete-case models, where the size of the training set is smaller.

4.4 Evaluation Metrics

We evaluate the various approaches in a quantitative and a qualitative way:

- We compute squared reconstruction errors defined as $(\hat{x}_d - x_d)^2$ for $d \in \mathcal{O}$ as well as squared imputation errors defined as $(\hat{x}_d - x_d)^2$ for $d \in \mathcal{U}$.
- We further compute reconstruction residuals defined as $\hat{x}_d - x_d$ for $d \in \mathcal{O}$ as well as imputation residuals defined as $\hat{x}_d - x_d$ for $d \in \mathcal{U}$ to find out whether reconstruction or imputation results are biased in any direction.
- We look at scatterplots of $\hat{x}_d$ versus x_d for dimensions d that are corrupted by a missingness process.
- We further look at scatterplots of $\hat{x}_d$ versus $\hat{x}_t$, where d is a dimension that is corrupted by a missingness process and t is the dimension responsible for the missingness in dimension d with respect to the specified MAR process. These scatterplots visually capture the resulting joint distribution of $\hat{x}_d$ and $\hat{x}_t$ and

therefore the uncertainty in the data, which is not captured by the quantitative metrics above.

For quantitative results we generally report the mean (standard deviation) of the ten runs for the ten training sets in the case of the treatment data and report the mean (standard deviation) of the results gathered by ten random restarts in the case of the simulated data.

5 Experiments

This chapter presents, illustrates and discusses the results of our experiments. Section 5.1 deals with simulated data in the MNAR setting. It shows the problem, confirms findings of other work, especially [IMF20], but also points out possible pitfalls. Section 5.2 ist the most extensive part covering all our MAR experiments on simulated data. It demonstrates the challenge in the presence of small data, how knowledge about the missingness process itself does not help and finally how knowledge about the missing values themselves makes a difference. Section 5.3 tries to substantiate some findings from Section 5.2 by conducting a subset of the experiments on the treatment data (MAR only).

5.1 Simulated Data (MNAR)

5.1.1 Baselines and Benchmarks

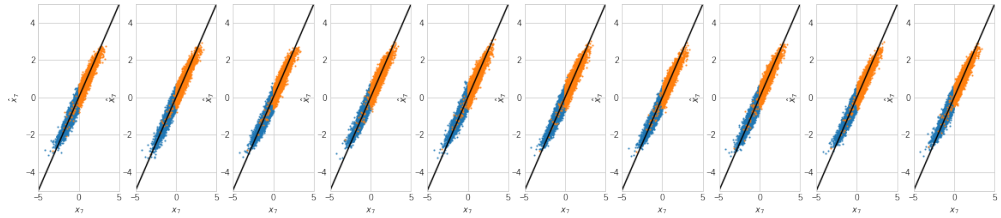
	MSE (O)	MSE (U)	MSE-7 (O)	**MSE-7 (U)**	Residuals-7 (O)	**Residuals-7 (U)**
$n = 1000$						
VAE	0.30 (0.00)	0.29 (0.00)	0.07 (0.00)	**0.07 (0.00)**	0.05 (0.03)	**-0.02 (0.03)**
CC	0.25 (0.00)	0.26 (0.00)	0.29 (0.01)	**1.83 (0.03)**	0.02 (0.01)	**-1.23 (0.01)**
CC (W)	0.29 (0.01)	0.43 (0.03)	0.31 (0.01)	**0.64 (0.10)**	0.05 (0.04)	**-0.62 (0.09)**
CC (S)	0.60 (0.04)	0.66 (0.07)	0.52 (0.07)	**0.57 (0.13)**	0.36 (0.06)	**-0.42 (0.12)**
MASKED	0.25 (0.00)	0.25 (0.00)	0.34 (0.00)	**2.08 (0.04)**	0.02 (0.02)	**-1.32 (0.02)**
$n = 10000$						
VAE	0.29 (0.00)	0.29 (0.00)	0.08 (0.00)	**0.08 (0.00)**	0.04 (0.02)	**-0.07 (0.02)**
CC	0.24 (0.00)	0.25 (0.00)	0.27 (0.00)	**1.97 (0.05)**	0.01 (0.02)	**-1.28 (0.02)**
CC (W)	0.27 (0.00)	0.28 (0.02)	0.30 (0.07)	**0.41 (0.14)**	0.22 (0.06)	**-0.38 (0.10)**
CC (S)	0.30 (0.01)	0.30 (0.01)	0.38 (0.08)	**0.35 (0.08)**	0.30 (0.07)	**-0.27 (0.07)**
MASKED	0.24 (0.00)	0.24 (0.00)	0.33 (0.00)	**2.16 (0.07)**	0.00 (0.03)	**-1.35 (0.03)**

Table 5.1: Baseline and benchmark results on simulated data under MNAR.

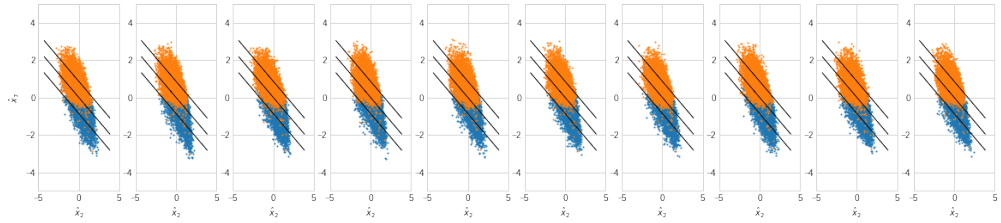
Table 5.1 shows results for some baseline and benchmark models on simulated data for both training set sizes ($n = 1000$ and $n = 10000$) under the specified MNAR process. As elaborated on in Section 4.1.1, this MNAR process is a simple self-censoring mechanism for variable x_7. MSE (O) denotes the mean squared error over all variables except for x_7 evaluated on all test samples where x_7 is observed after corruption. MSE (U) is the corresponding metric for all samples where x_7 is unobserved. MSE-7 (O) and MSE-7 (U) are the mean squared errors of the potentially missing variable x_7 only. Finally, Residuals-7 (O) and Residuals-7 (U) are the average residuals $\hat{x}_7 - x_7$ to determine whether x_7 is under- or overestimated in some region. Note that the test set ($n = 10000$) is the exact same one for all experiments on simulated data and we average the results of

ten runs with random initializations with the standard deviation of the ten runs shown in parantheses.

The first model is the vanilla VAE. It is an unrealistic benchmark since it uses the train and test sets before corruption, i.e. without any missingness in x_7, and just has to reconstruct the encoder input. The residuals are close to zero and Figure 5.1 shows that the scatterplots of $\hat{x}_7$ versus $\hat{x}_2$ resemble the scatterplot of the true x_7 versus x_2 (cf. Figure 4.4). The ten columns in Figure 5.1 and subsequent figures represent the results of the ten random restarts in our experiments. If the visualizations look similar for $n = 1000$ and $n = 10000$ training samples, which is the case here, we only show the plots for $n = 1000$ training samples. All in all, the results for the VAE model demonstrate that the chosen hyperparameters make it possible to generate realistic samples. However, note that this VAE does not (have to) learn how to handle corrupted inputs.



(a) $\hat{x}_7$ versus x_7

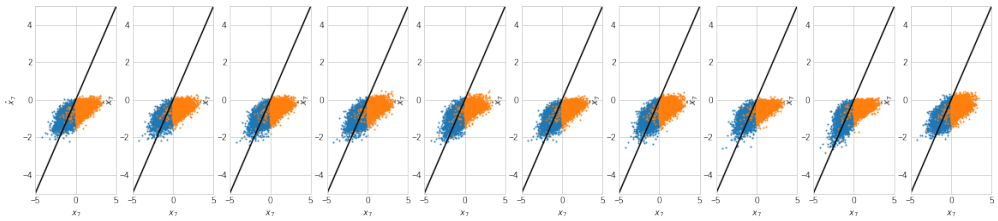


(b) $\hat{x}_7$ versus $\hat{x}_2$

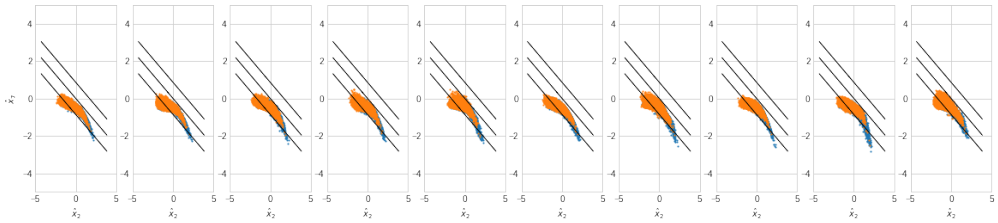
Figure 5.1: VAE results for $n = 1000$ training samples under MNAR. As expected, the results are very good.

Like the vanilla VAE, the CC model is trained on and applied to fully observed samples. Unlike the vanilla VAE, the CC model considers datasets after corruption though. Therefore, it ignores samples with missingness in x_7 during training and rarely sees samples with large values in x_7 as a consequence. This lack leads to severe underestimation of the value of large instances of x_7 (cf. Residuals-7 (U)). Indeed, hardly any values of the decoder output $\hat{x}_7$ are greater than zero as shown in Figure 5.2. This analysis demonstrates the principal problem with the assumed MNAR mechanism. With large values remaining unobserved due to self-censoring, the model sees almost no large values during training and tends to underestimate them. The issue is even slightly more pronounced with $n = 10000$ training samples indicating that just increasing the training set size does not resolve the problem.

CC (W) and CC (S) reduce the underestimation of large x_7 compared to plain CC



(a) $\hat{x}_7$ versus x_7

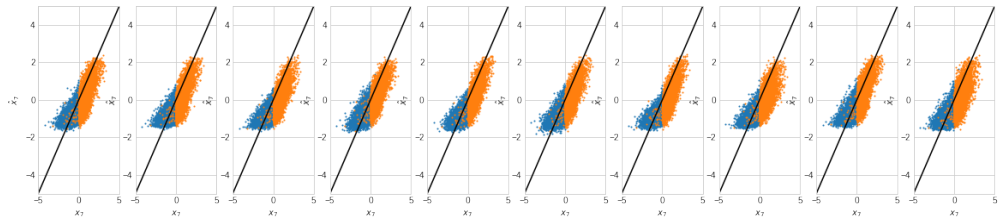


(b) $\hat{x}_7$ versus $\hat{x}_2$

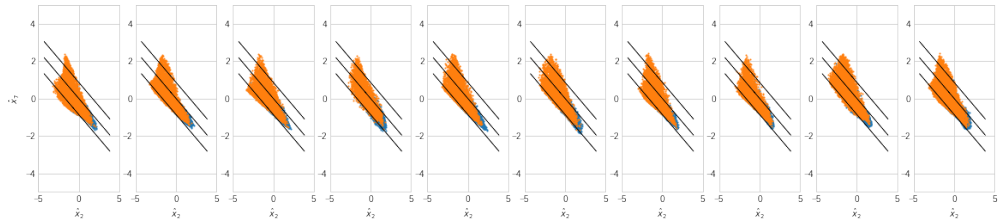
Figure 5.2: CC results for $n = 1000$ training samples under MNAR. Since most samples with large x_7 are omitted, the model clearly underestimates x_7 when unobserved.

(cf. MSE-7 (U) and Residuals-7 (U)), but yield worse results regarding the other metrics. CC (W) uses the same training sets as CC, but puts more weight on the reconstruction loss of the few samples with both large and observed x_7. CC (S) subsamples the corrupted training sets to balance the amount of samples with large x_7 and small x_7 and is left with approximately 10 and 100 training samples instead of 1000 and 10000. As Table 5.1 shows, CC (W) gives better results than CC (S) in our experiments while still producing somewhat unnatural plots for a lower number of training samples (cf. Figure 5.3). Still, we argue that CC (S) should asymptotically work as well as a vanilla VAE because by subsampling we obtain a balanced dataset from the original distribution. However, all CC approaches do not learn how to handle corrupted inputs and are therefore impractical. Note that for CC (W) and CC (S) increasing the dataset size does make a difference (cf. Figure 5.3 versus Figure 5.4). The high sample weights in CC (W) are distributed among ten times more samples and CC (S) is simply a ten times larger dataset for $n = 10000$ compared to $n = 1000$.

The results for the MASKED model convey the same difficulties as the plain CC approach, i.e. a high MSE-7 (U) as well as large negative Residuals-7 (U) for both training set sizes. Recall that the masked model trains on all $n = 1000$ / $n = 10000$ training samples, but only computes reconstruction loss for observed parts $\mathbf{x}_O$ and more importantly does not use x_7 as input, i.e. there is no input node for x_7 at all.

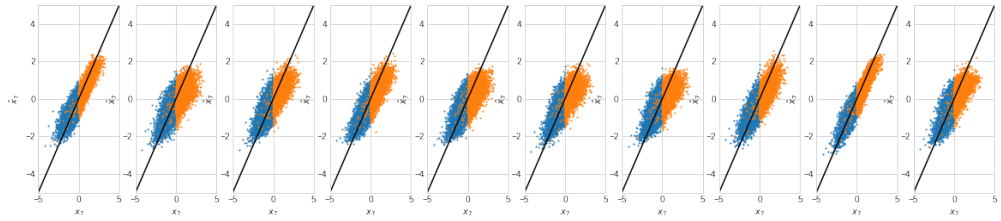


(a) $\hat{x}_7$ versus x_7

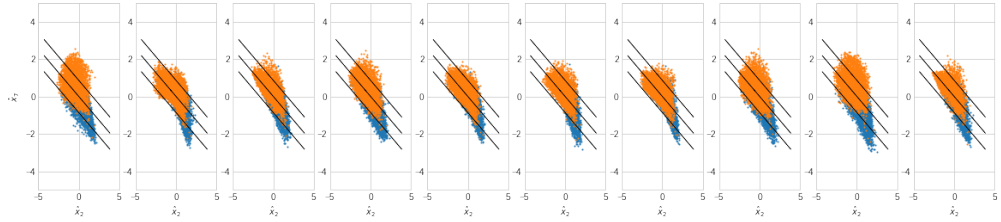


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.3: CC (W) results for $n = 1000$ training samples under MNAR. CC (W) accomplishes to generate larger values of $\hat{x}_7$.



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.4: CC (W) results for $n = 10000$ training samples under MNAR. The result is better and looks more natural than for the smaller training set size.

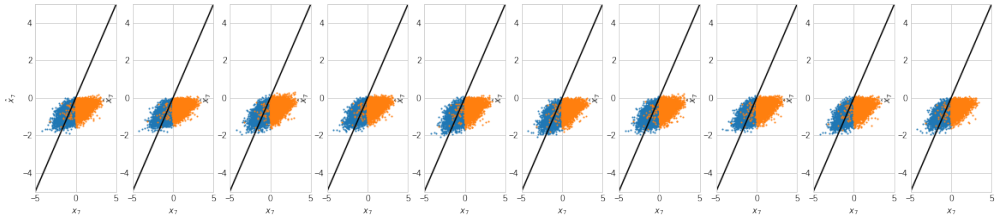
5.1.2 Models with Corrupted Input

The results for the ZERO and the M model in Table 5.2 are very similar to the results for CC and MASKED. All four approaches yield similar visualizations as in Figure 5.5 showing the plots for the M model with $n = 1000$ training samples. We are surprised that

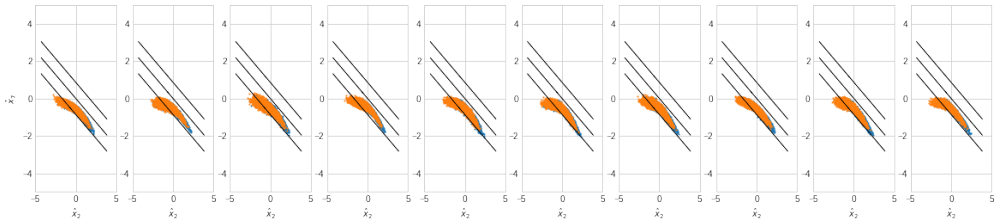
	MSE (O)	MSE (U)	MSE-7 (O)	**MSE-7 (U)**	Residuals-7 (O)	**Residuals-7 (U)**
$n = 1000$						
ZERO	0.25 (0.00)	0.25 (0.00)	0.32 (0.00)	**1.99 (0.03)**	0.03 (0.01)	**-1.28 (0.01)**
M	0.25 (0.00)	0.25 (0.00)	0.30 (0.00)	**2.00 (0.07)**	0.02 (0.03)	**-1.29 (0.03)**
$n = 10000$						
ZERO	0.24 (0.00)	0.24 (0.00)	0.30 (0.00)	**2.08 (0.08)**	0.00 (0.03)	**-1.32 (0.03)**
M	0.24 (0.00)	0.24 (0.00)	0.29 (0.00)	**2.07 (0.06)**	0.01 (0.02)	**-1.32 (0.02)**

Table 5.2: Results for models with corrupted input on simulated data under MNAR.

the MSE-7 (O) of ZERO and M are only slightly better than the MSE-7 (O) of MASKED given they have access to the true x_7 in the encoder whenever $7 \in \mathcal{O}$.



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.5: M results for $n = 1000$ training samples under MNAR. Since most large values of x_7 are unobserved, the model clearly underestimates x_7 when unobserved.

5.1.3 Missingness Models

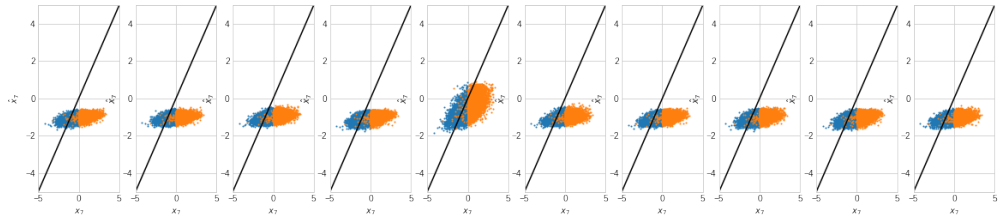
We can see the quantitative results for all eight variations of VAEs with a missingness model in Table 5.3. The GU models use a generic missingness model taking $\mathbf{x}_O$ as input if $7 \in \mathcal{O}$, and $\mathbf{x}_O$ together with $\hat{x}_7$ if $7 \in \mathcal{U}$ whereas the GOU models use a generic missingness model with the full generation $\hat{\mathbf{x}}$ as input.

The results of M (GU) and ZERO (GU) are even worse than the results of M and ZERO. Only one random restart of M (GU) with $n = 1000$ training samples leads to a somewhat reasonable result (cf. column five in Figure 5.6).

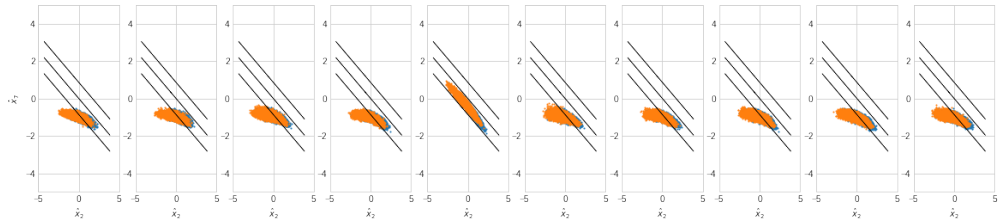
However, the M (GOU) model improves results for $n = 1000$ training samples and even more for $n = 10000$ training samples (cf. MSE-7 (U) and Residuals-7 (U)). As Figure 5.7 ($n = 1000$) and Figure 5.8 ($n = 10000$) show, this specification particularly yields large values of $\hat{x}_7$ without seeing many such values during training. In principle, the model

	MSE (O)	MSE (U)	MSE-7 (O)	**MSE-7 (U)**	Residuals-7 (O)	**Residuals-7 (U)**
$n = 1000$						
M (GU)	0.25 (0.00)	0.25 (0.00)	0.41 (0.03)	**3.23 (0.59)**	-0.23 (0.12)	**-1.68 (0.21)**
ZERO (GU)	0.25 (0.00)	0.25 (0.00)	0.42 (0.01)	**3.31 (0.10)**	-0.23 (0.04)	**-1.72 (0.03)**
M (GOU)	0.28 (0.02)	0.27 (0.01)	0.23 (0.09)	**1.48 (0.39)**	0.01 (0.02)	**-0.55 (0.87)**
ZERO (GOU)	0.26 (0.00)	0.25 (0.00)	0.28 (0.00)	**1.83 (0.05)**	0.01 (0.02)	**-1.22 (0.02)**
M (TU)	0.29 (0.01)	0.26 (0.00)	0.13 (0.03)	**0.50 (0.09)**	0.05 (0.04)	**0.10 (0.19)**
ZERO (TU)	0.29 (0.01)	0.26 (0.00)	0.25 (0.12)	**0.52 (0.06)**	0.26 (0.13)	**-0.39 (0.09)**
M (TOU)	0.29 (0.01)	0.28 (0.01)	0.11 (0.04)	**0.58 (0.14)**	-0.06 (0.03)	**0.21 (0.15)**
ZERO (TOU)	0.28 (0.01)	0.26 (0.01)	0.18 (0.06)	**0.47 (0.02)**	-0.01 (0.03)	**-0.23 (0.05)**
$n = 10000$						
M (GU)	0.24 (0.00)	0.24 (0.00)	0.35 (0.00)	**2.66 (0.02)**	-0.08 (0.01)	**-1.52 (0.01)**
ZERO (GU)	0.24 (0.00)	0.24 (0.00)	0.35 (0.00)	**2.66 (0.02)**	-0.08 (0.01)	**-1.52 (0.01)**
M (GOU)	0.25 (0.00)	0.24 (0.00)	0.29 (0.00)	**0.66 (0.25)**	0.01 (0.02)	**0.18 (0.20)**
ZERO (GOU)	0.25 (0.00)	0.24 (0.00)	0.26 (0.01)	**1.81 (0.06)**	0.00 (0.02)	**-1.21 (0.03)**
M (TU)	0.26 (0.01)	0.25 (0.00)	0.24 (0.05)	**1.50 (1.51)**	0.05 (0.05)	**0.56 (0.70)**
ZERO (TU)	0.27 (0.01)	0.25 (0.00)	0.19 (0.03)	**0.44 (0.02)**	0.10 (0.02)	**-0.27 (0.08)**
M (TOU)	0.25 (0.00)	0.24 (0.00)	0.29 (0.00)	**4.05 (0.99)**	-0.01 (0.02)	**1.62 (0.24)**
ZERO (TOU)	0.25 (0.00)	0.24 (0.00)	0.28 (0.01)	**0.39 (0.01)**	-0.01 (0.03)	**-0.09 (0.08)**

Table 5.3: Results for models with a missingness model on simulated data under MNAR.



(a) $\hat{x}_7$ versus x_7



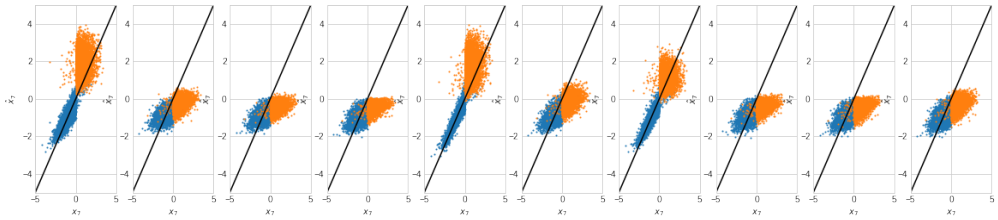
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.6: M (GU) results for $n = 1000$ training samples under MNAR. Only one restart (column five) gives a somewhat reasonable result.

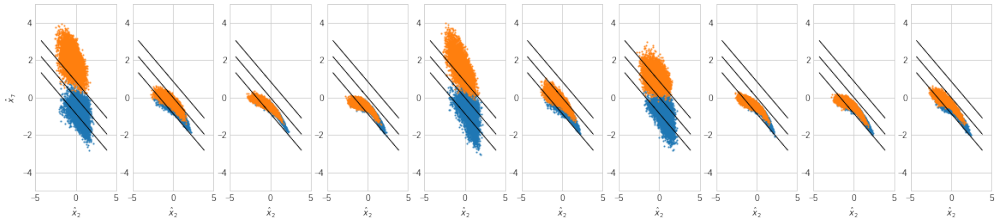
generates reconstructions that are useful to discriminate between $m_7 = 0$ and $m_7 = 1$. Still, the results are not fully satisfactory and rather volatile, especially for $n = 1000$ training samples. Moreover, we want to point out that large values of $\hat{x}_7$ happen to be useful to reconstruct the missingness mask in this specific task, but it might not be the case for a different dataset that the desired values are similarly useful, which can potentially lead to undesired results. Note that the model without access to $\mathbf{m}$ in the input (ZERO (GOU)) does not achieve any relevant improvement over the plain ZERO

model.

Summing up, generic missingness models can help but often struggle in the MNAR setting in our experiments. The hope is that they generate values of $\hat{x}_7$ that help to discriminate between $m_7 = 0$ and $m_7 = 1$ and at the same time happen to be close to the true values of x_7. Even for our very simple dataset this idea is not reliable and robust.



(a) $\hat{x}_7$ versus x_7



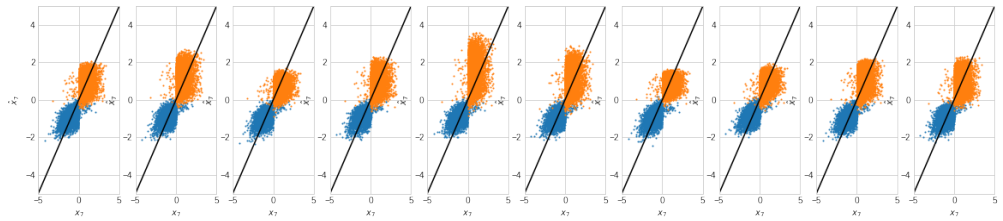
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.7: M (GOU) results for $n = 1000$ training samples under MNAR. The model sometimes succeeds in generating large values of $\hat{x}_7$.

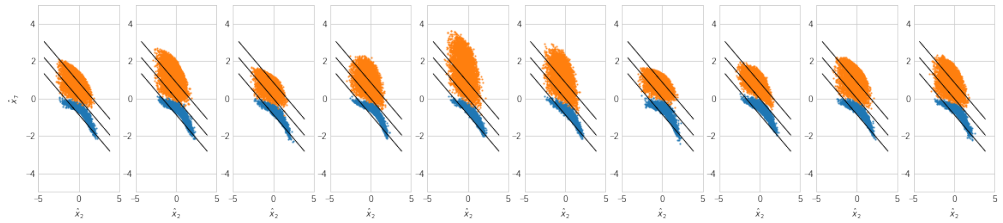
Examining the results of the models with the true missingness model (TU and TOU) for $n = 1000$, we can see clear improvements confirming the findings in [IMF20]. Recall that knowing the true missingness process leads to the conclusion that if x_7 is missing it will typically be greater than zero motivating larger generated values. The MSE-7 (U) is significantly lower and the Residuals-7 (U) are closer to zero indicating that the earlier (extreme) underestimation of x_7 due to missingness of large values of x_7 does not happen. The true (frozen) missingness model guides the variational autoencoder to generate values of $\hat{x}_7$ greater than zero when $m_7 = 1$.

Figure 5.9 shows the visualizations for M (TU), which look very reasonable. In contrast, the ZERO (TU) model tends to generate values of $\hat{x}_7$ that are just slightly greater than zero (cf. Figure 5.10). We suspect that without the missingness mask $\mathbf{m}$ in the encoder, the model is less confident to do the right thing when generating large values of $\hat{x}_7$. The M (TOU) also uses $\hat{x}_7$ as input in the missingness model when $7 \in \mathcal{O}$ and thereby induces a slight gap between samples with $m_7 = 0$ and samples with $m_7 = 1$ (cf. Figure 5.11).

For $n = 10000$ training samples, the ZERO (TU) and ZERO (TOU) models have a comparable performance and similar results to the $n = 1000$ case. However, the M (TU) and especially the M (TOU) have a very high MSE-7 (U). Obviously, with $\mathbf{m}$ in the encoder, the decoder sometimes produces extremely high values of $\hat{x}_7$ as shown in Figure 5.12. This picture does not arise with a doubled batch size reducing the number

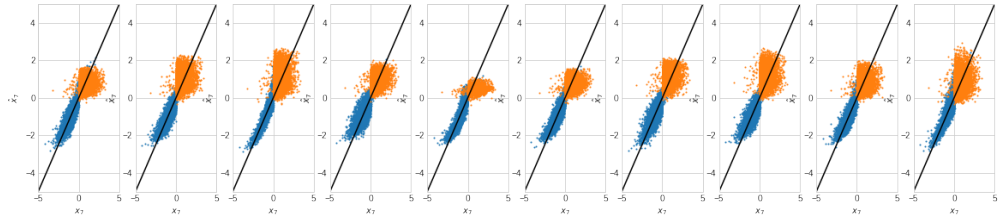


(a) $\hat{x}_7$ versus x_7

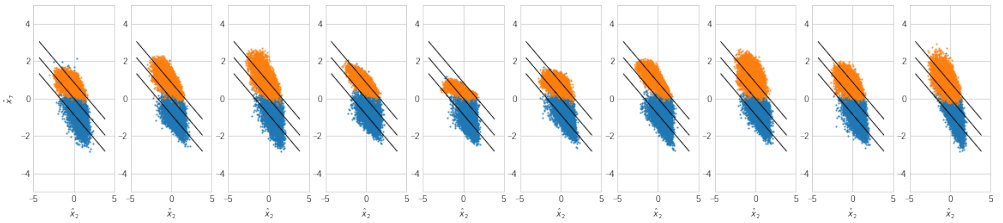


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.8: M (GOU) results for $n = 10000$ training samples under MNAR. The model succeeds in generating large values of $\hat{x}_7$.



(a) $\hat{x}_7$ versus x_7

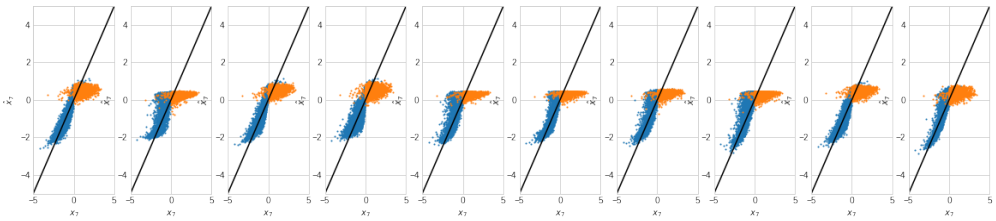


(b) $\hat{x}_7$ versus $\hat{x}_2$

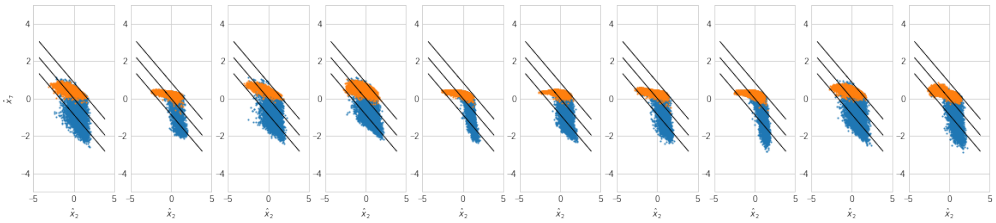
Figure 5.9: M (TU) results for $n = 1000$ training samples under MNAR. The model succeeds in generating reasonable values of $\hat{x}_7$.

of update steps (cf. Figure 5.13).

All in all, incorporating the true missingness mechanism potentially helps. It can guide a variational autoencoder to generate values of $\hat{x}_7$ larger than zero even though such values are barely represented in the corrupted training set. Still the missingness process

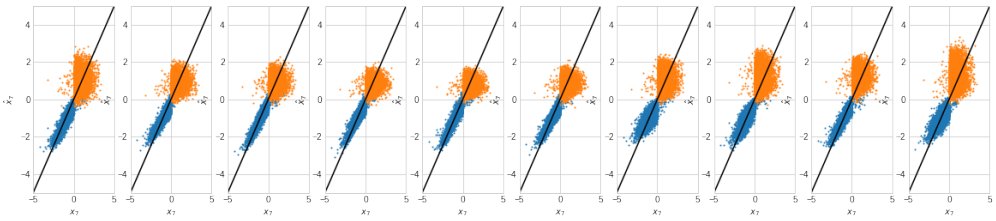


(a) $\hat{x}_7$ versus x_7

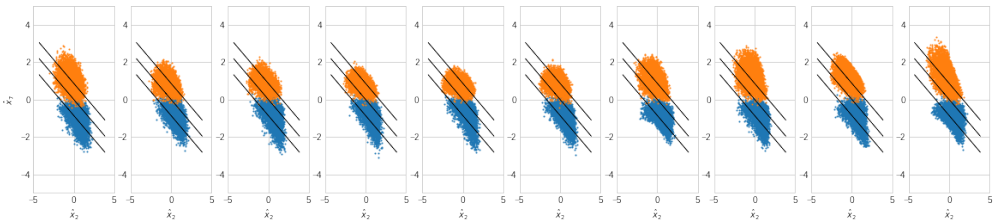


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.10: ZERO (TU) results for $n = 1000$ training samples under MNAR. The model generates values of $\hat{x}_7$ greater than zero but all such values are only slightly above zero.



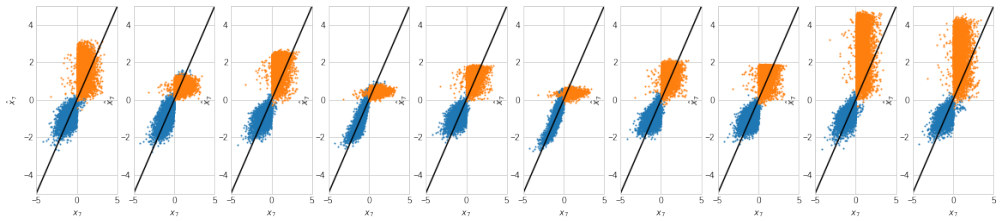
(a) $\hat{x}_7$ versus x_7



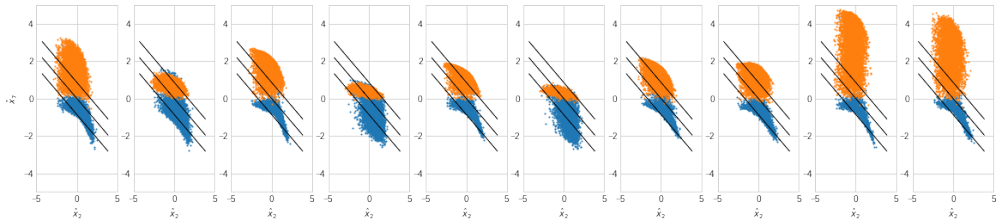
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.11: M (TOU) results for $n = 1000$ training samples under MNAR. Not only pushes the orange point cloud above zero, but also pushes the blue point cloud below zero inducing a slight gap.

(in our case) does not reveal information regarding the distribution of x_7 in this region. As a consequence, decoded values can be consistent with the missingness model but at

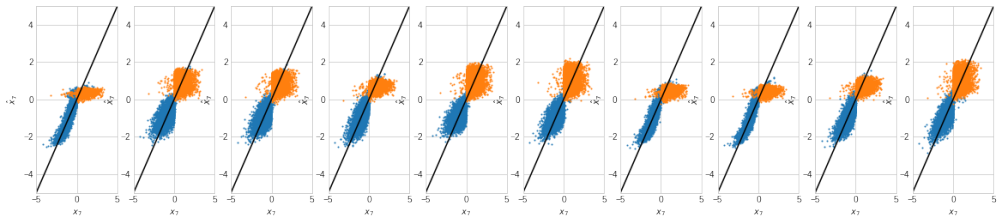


(a) $\hat{x}_7$ versus x_7

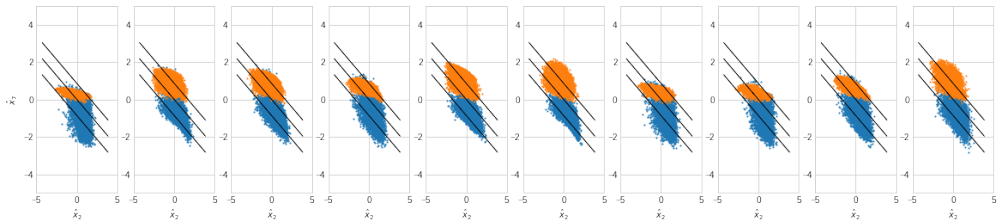


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.12: M (TU) results for $n = 10000$ training samples under MNAR. The model sometimes produces extremely high values of $\hat{x}_7$, which are undesired.



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.13: M (TU) results for $n = 10000$ training samples under MNAR with half the number of update steps. The model generates more reasonable values of $\hat{x}_7$ than in Figure 5.12.

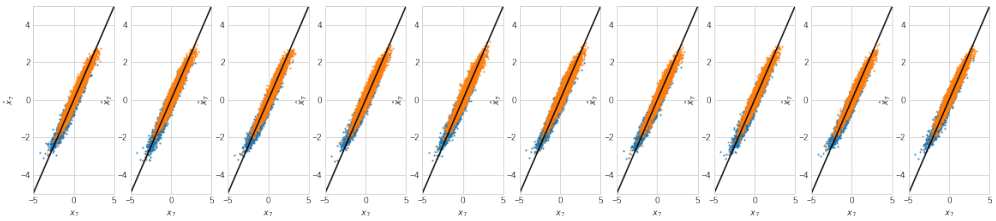
the same time inconsistent with the actual data distribution.

5.2 Simulated Data (MAR)

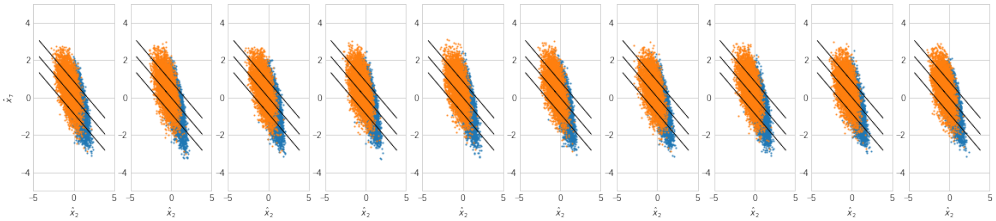
5.2.1 Baselines and Benchmarks

Table 5.4 shows the quantitative results of baselines, benchmarks and models with corrupted input for simulated data under the MAR process. This time, x_2 determines the missingness in x_7. When x_2 is small (lower than 0), x_7 is typically unobserved. Therefore, we also report results for x_2 separately in the tables of this section.

The performance of the vanilla VAE is basically the same as under MNAR. It is trained on the exact same data as before. The only difference is that due to the different missingness process, different samples are colored orange in the plots (e.g. Figure 5.14) and the abbreviations (U) and (O) define different underlying sets of test samples.



(a) $\hat{x}_7$ versus x_7

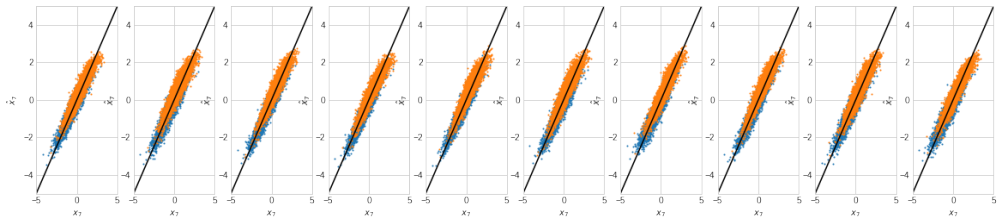


(b) $\hat{x}_7$ versus $\hat{x}_2$

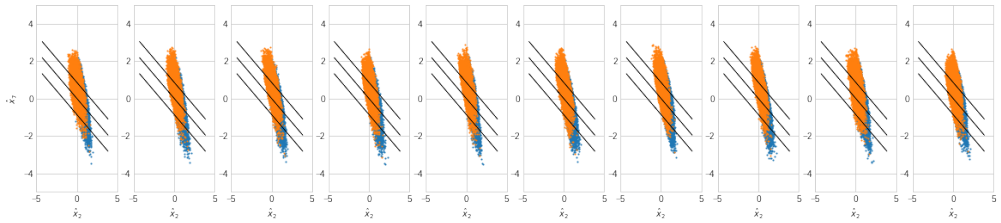
Figure 5.14: VAE results for $n = 1000$ training samples under MAR.

The CC analysis mainly affects the MSE-2 (U) and the Residuals-2 (U) value. CC analysis under MAR relates to x_2 as CC analysis under MNAR relates to x_7. Samples with low values in x_2 and consequently missingness in x_7 are omitted before training so that the model does not learn to generate low values for x_2. As a result the dependency between $\hat{x}_7$ and $\hat{x}_2$ in Figure 5.15 ($n = 1000$) is steeper than that between the original x_7 and x_2. Similar to the MNAR setting, weighting (CC (W)) and subsampling (CC (S)) alleviate the problems with CC, but among others deteriorate the MSE-7 (O) and MSE-7 (U) results.

The MASKED approach provides a realistic benchmark for MSE-7 (U). Since it does not use x_7 as input, it learns (among other things) to generate x_7 from the other nine variables. Therefore, the results for MSE-7 (O) (0.73 and 0.71) and MSE-7 (U) (0.68 and 0.68) are very close. Obviously, this model is otherwise impractical because it leaves available information unused, namely x_7 whenever it is observed. Figure 5.16 shows that



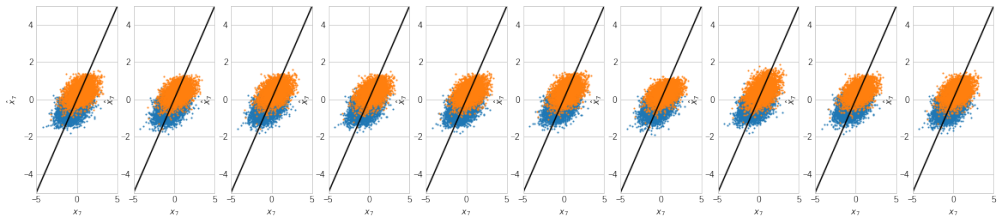
(a) $\hat{x}_7$ versus x_7



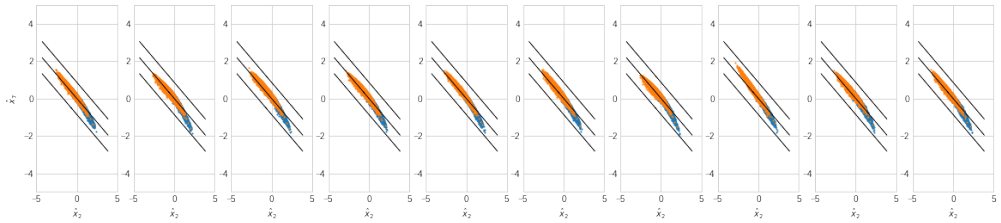
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.15: CC results for $n = 1000$ training samples under MAR. Since low values of x_2 are overestimated, the dependency between $\hat{x}_7$ and $\hat{x}_2$ is steeper.

the generations of x_7 from MASKED with $n = 1000$ training samples are very similar to $\mathbb{E}[x_7|x_2]$ with respect to our simple linear regression providing prior knowledge.



(a) $\hat{x}_7$ versus x_7



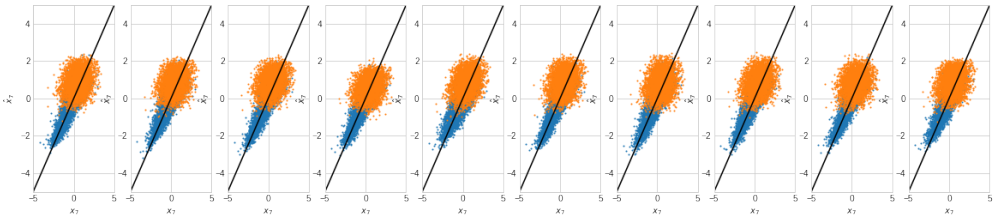
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.16: MASKED results for $n = 1000$ training samples under MAR. Generated values for unobserved x_7 are very close to $\mathbb{E}[x_7|x_2]$ in terms of the prior knowledge.

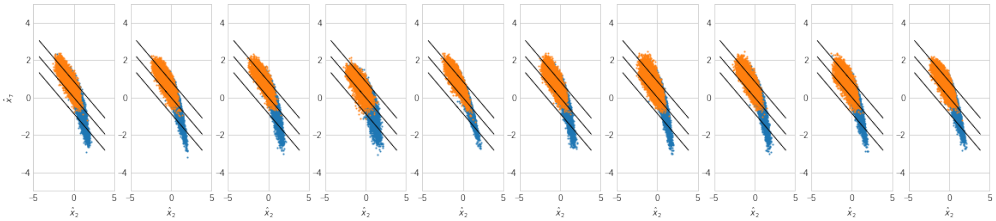
5.2.2 Models with Corrupted Input

As Table 5.4 shows, all models with corrupted input overestimate unobserved values of x_7 (cf. Residuals-7 (U)). However, almost all models clearly improve with $n = 10000$ instead of $n = 1000$ training samples indicating that increasing the training set helps under our MAR specification. This trend is in opposition to the MNAR setting, where we could not spot such an improvement.

The ZERO model (Figure 5.17 for $n = 1000$) reaches the best MSE-7 (U) among all models with corrupted input for both training set sizes. Recall that zero imputation is less problematic in our setting compared to a setting where zero is outside the typical range of the imputed variable. With 10000 training observations, the MSE-7 (U) of 0.80 is only slightly worse than the MSE-7 (U) of the MASKED model (0.68).



(a) $\hat{x}_7$ versus x_7

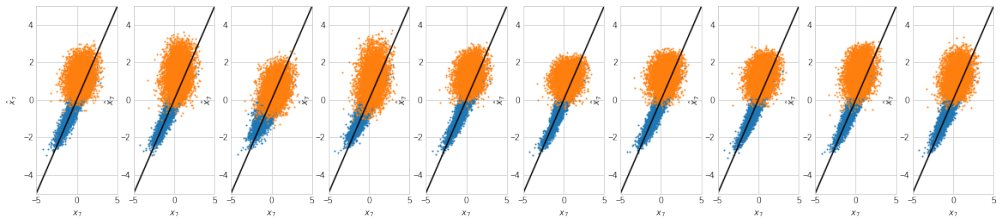


(b) $\hat{x}_7$ versus $\hat{x}_2$

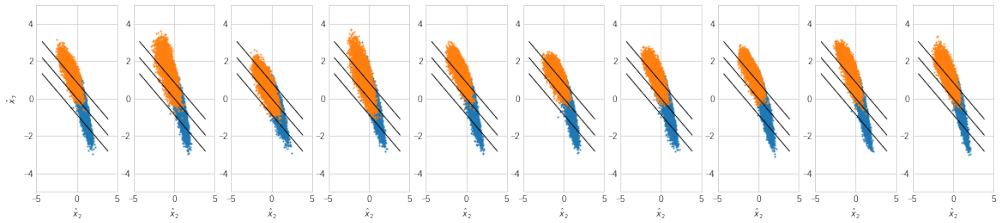
Figure 5.17: ZERO results for $n = 1000$ training samples under MAR. The model reaches the best MSE-7 (U) among all models with corrupted input.

While achieving a better MSE-7 (O), the M model leads to a worse MSE-7 (U) and worse Residuals-7 (U) compared to ZERO. The M specification can distinguish between samples with imputed zero and observed zero and therefore learn specialized mappings for the $m_7 = 0$ and $m_7 = 1$ case. However, it has no guidance to generate reasonable values of $\hat{x}_7$ for the $m_7 = 1$ case during training. Figure 5.18 shows M results for the small training set illustrating the clear overestimation of unobserved x_7, which is less severe for the larger training set as demonstrated in Figure 5.19.

Our specifications of a partial VAE [MTP+18] (PVAE (MAX), PVAE (AVG), SPVAE (MAX), SPVAE (AVG)) have a more flexible encoder and often achieve lower MSE-7 (O) results compared to ZERO and M. The results for MSE-7 (U) and Residuals-7 (U) also clearly improve from 1000 to 10000 training samples. Nevertheless, note that the outcome of a partial VAE is often volatile as indicated by the standard deviations in parantheses

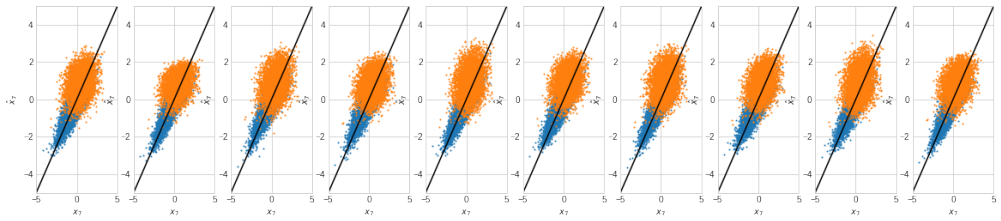


(a) $\hat{x}_7$ versus x_7

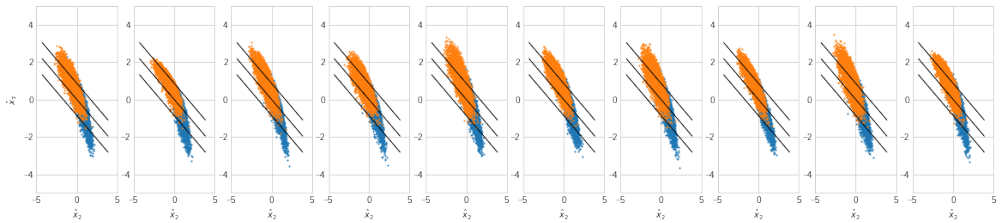


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.18: M results for $n = 1000$ training samples under MAR. The model clearly overestimates many unobserved values of x_7.



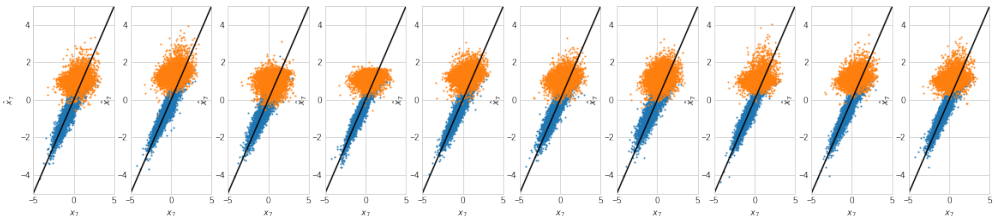
(a) $\hat{x}_7$ versus x_7



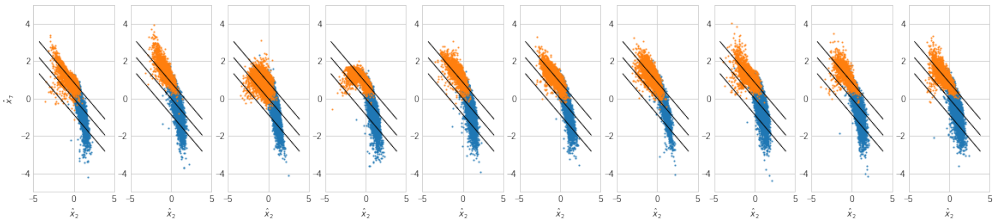
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.19: M results for $n = 10000$ training samples under MAR. The overestimation of unobserved values of x_7 is less severe.

of MSE-7 (U) and Residuals-7 (U) and observable in Figure 5.20 and Figure 5.21 where we can see visualizations for the PVAE (MAX) model.

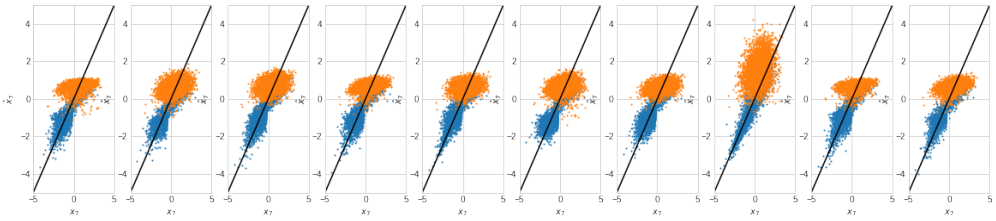


(a) $\hat{x}_7$ versus x_7

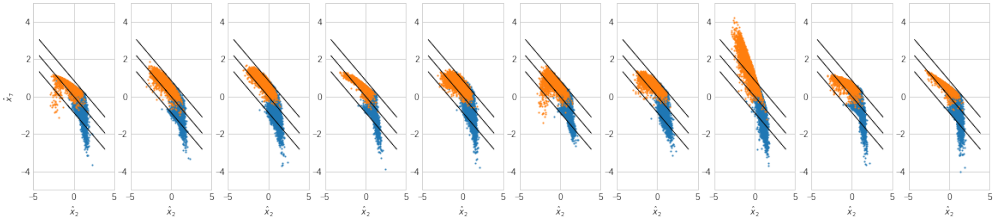


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.20: PVAE (MAX) results for $n = 1000$ training samples under MAR. Overall, the results are reasonable but somewhat volatile.



(a) $\hat{x}_7$ versus x_7



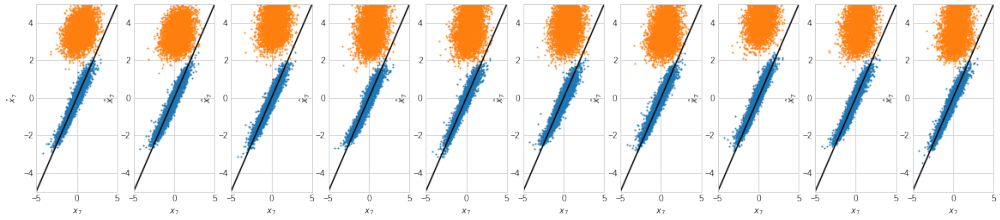
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.21: PVAE (MAX) results for $n = 10000$ training samples under MAR. The volatility of the outcome becomes clear.

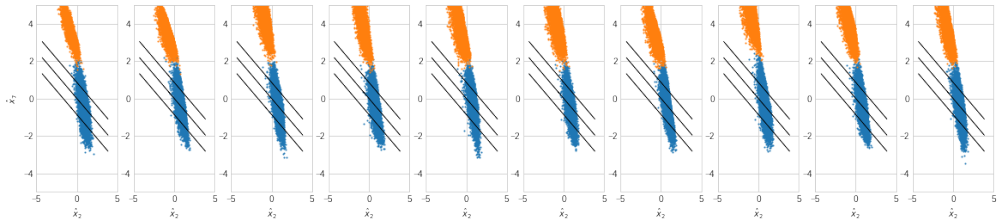
5.2.3 Missingness Models

Table 5.5 contains the MAR results for all missingness models, weighted models, penalized models as well as models with imputation. We can see that variational autoencoders with a generic missingness model either do not improve (GU) over their counterparts

without a missingness model or make things drastically worse (GOU). Espcially, the M (GOU) approach is very hurtful (cf. Figure 5.22). It seems to generate very large values of $\hat{x}_7$ whenever $m_7 = 1$, which certainly help to reconstruct the missingness mask but are far from the (unobserved) original values. Recall, that under MNAR a similar outcome happened to slightly improve results.



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

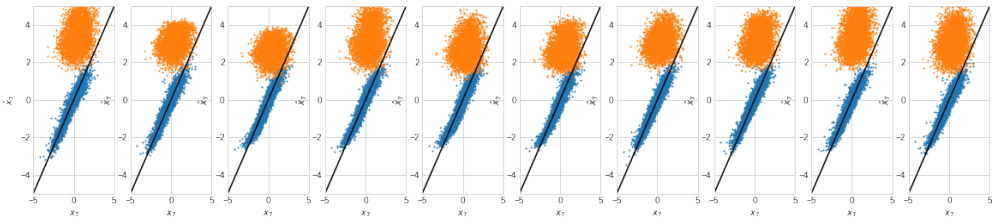
Figure 5.22: M (GOU) results for $n = 1000$ training samples under MAR. The model generates absurdly high values of $\hat{x}_7$.

In particular, incorporating knowledge regarding the true missingness process was helpful under MNAR. However, the TOU[1] models do not yield improvements, especially when the model has access to $\mathbf{m}$ (M (TOU)). Figure 5.23 and Figure 5.24 show visualizations for M (TOU) and ZERO (TOU) with $n = 1000$ training samples. In essence, knowing the MNAR missingness process leads the model to generate higher values of $\hat{x}_7$, which was beneficial in that setting. On the contrary, knowing the MAR process reveals that x_2 is likely to be lower than zero. However, since x_2 is observed anyway, the prior knowledge about the missingness mechanism does not provide a comparable advantage.

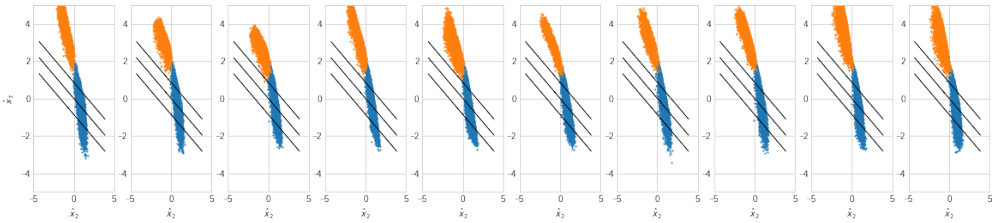
5.2.4 Weighted Models

Putting more weight on the reconstruction of observed values of x_7 with a high probability of missingness yields considerable improvements in the MSE-7 (U) metric while hurting MSE-7 (O) a little for the smaller training set. Obviously putting more weight on rare samples diminishes the focus on the remaining samples during training. We are surprised to see that this improvement does not translate to the larger training set. As shown in Figure 5.25, the approach sometimes produces unnatural results substantiating the

[1]Note that the TU approach is not meaningful in the MAR case.

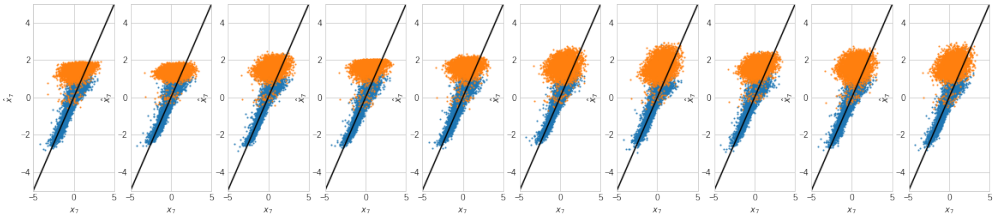


(a) $\hat{x}_7$ versus x_7

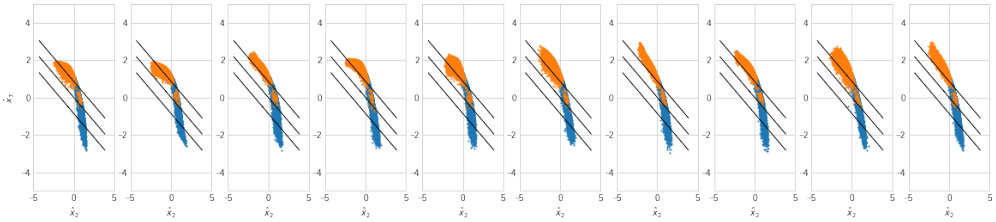


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.23: M (TOU) results for $n = 1000$ training samples under MAR. The model is worse than M.



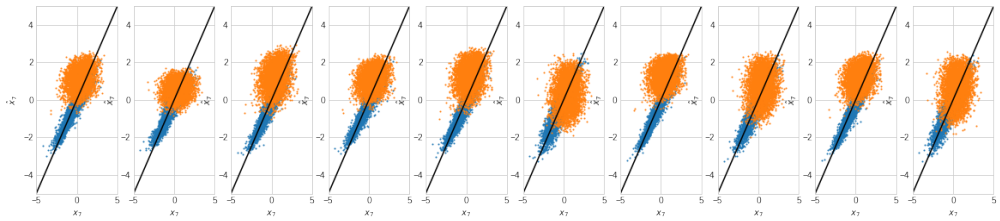
(a) $\hat{x}_7$ versus x_7



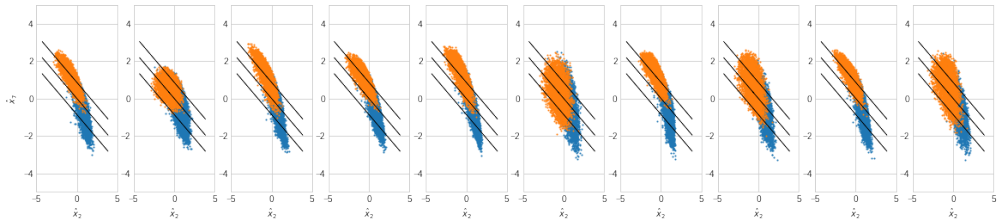
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.24: ZERO (TOU) results for $n = 1000$ training samples under MAR. The missingness model is not helpful.

moderate results for $n = 10000$ training samples in Table 5.5.



(a) $\hat{x}_7$ versus x_7



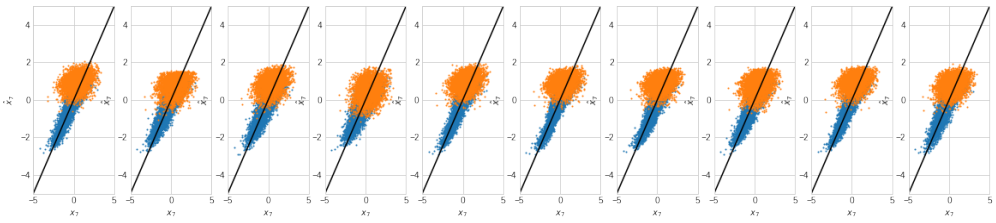
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.25: M (DW) results for $n = 10000$ training samples under MAR. The approach sometimes produces undesired results (column two, six, eight and ten).

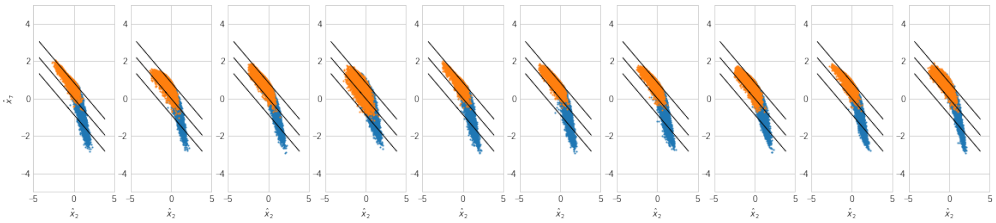
5.2.5 Penalized Models

Adding a penalty term based on the prior knowledge to the loss function clearly improves both the MSE-7 (U) and the Residuals-7 (U) metrics for $n = 1000$ training samples. Comparing Figure 5.26 (M (P)) with Figure 5.18 (M), we can detect that generations outside of the prior knowledge interval are shrunk due to the penalty. Some overestimation still happens (cf. Residuals-7 (U)) since the penalty is not active within the prior knowledge interval. The improvement is less pronounced for $n = 10000$ training samples. Therefore, while the effect of our penalty idea to incorporate prior knowledge in MAR settings vanishes with larger amounts of training data, we argue that in small data settings guiding the model not just with data but also with (a) prior knowledge (interval) can have a beneficial regularization effect and make results better and more robust. Be aware that especially for larger datasets a penalty based on such a prior knowledge interval can be quite restrictive and eventually harmful.

We further conduct an analysis to find out about how strong the influence of the penalization is on the encoder and the decoder. We train the M (P) model for 150 epochs instead of 100 for $n = 1000$ training samples. During the first 100 epochs we train it like the plain M model without a penalty and after 100 epochs we turn on the penalty and in the first experiment freeze the encoder parameters and in the second experiment the decoder parameters. It turns out that with a frozen encoder, the model still reaches an MSE-7 (U) of 1.05 (0.04) and Residuals-7 (U) of 0.55 (0.07). When freezing the decoder instead, the model reaches an MSE-7 (U) of 1.37 (0.12) and Residuals-7 (U) of 0.72 (0.11), which is still an improvement over M but a less pronounced one. Comparing Figure 5.27 and Figure 5.28, we can clearly see the difference in results between a model with a frozen



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.26: M (P) results for $n = 1000$ training samples under MAR. The model penalizes orange dots outside of the prior knowledge interval and thereby regularizes the VAE.

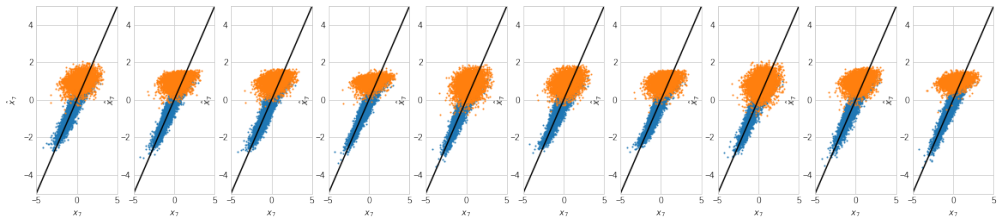
encoder and a model with a frozen decoder. The weak improvement when the decoder is frozen indicates that the impact of the penalty on the encoder and therefore the latent space is rather low. Hence, the benefit of our method is likely to be greater for tasks that rely on the decoder than for tasks that directly use the learned latent space representation. Further investigations and solutions in this matter are necessary.

5.2.6 Models with Imputation

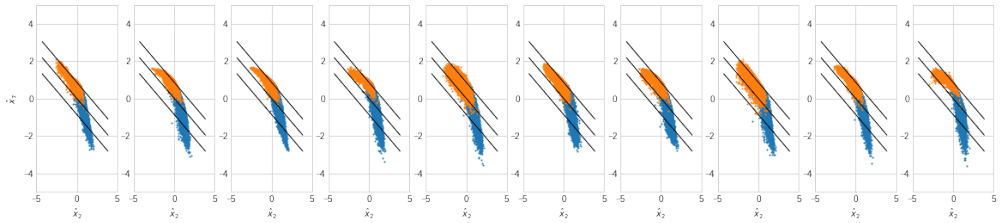
Finally, using the imputed $\tilde{\tilde{\mathbf{x}}}$ leads to the best results for MSE-7 (U) with slightly affected performance regarding MSE-7 (O) among all practical models using corrupted inputs. The generated values of $\hat{x}_7$ are very close to $\mathbb{E}[x_7|x_2]$ for $m_7 = 1$ and therefore similiar to the results of the MASKED specification for $m_7 = 1$ (cf. Figure 5.29). Note that there is no significant difference between the model relying on $\tilde{\tilde{\mathbf{x}}}$ as both input and target (VAE (ET)) and the model relying on $\tilde{\tilde{\mathbf{x}}}$ as target only (M (TO)). However, using $\tilde{\tilde{\mathbf{x}}}$ instead of $\tilde{\mathbf{x}}$ as input in the M and the ZERO model does not help in this experiment. Still, we would expect it to be a reasonable approach if features were not standardized and zero outside of the typical range of x_7 because $\mathbb{E}[x_7|x_2]$ would be a more realistic imputation than zero.

5.3 Treatment Data

For the treatment dataset, we analyze and compare results of a subset of the considered models that is particularly interesting with respect to our motivation to use prior knowledge

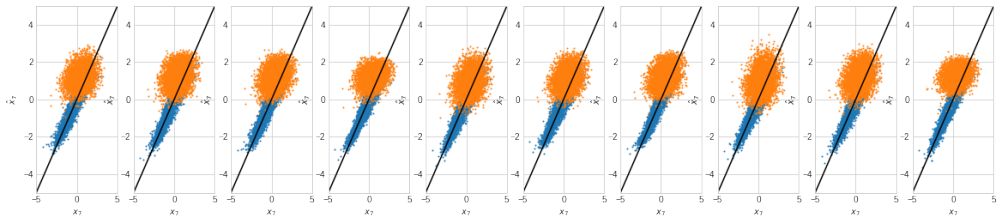


(a) $\hat{x}_7$ versus x_7

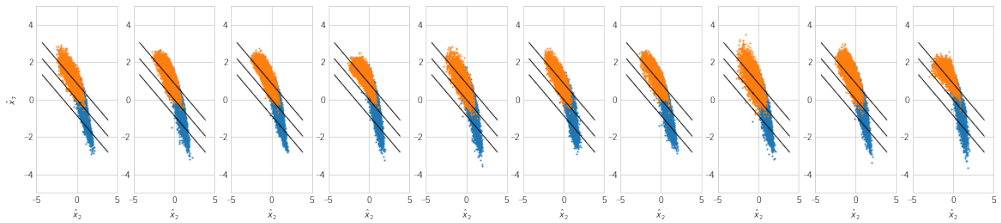


(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.27: M (P) results for $n = 1000$ training samples under MAR with a frozen encoder. When freezing the encoder before turning on the penalty, the results are still similar to the plain M (P) model.



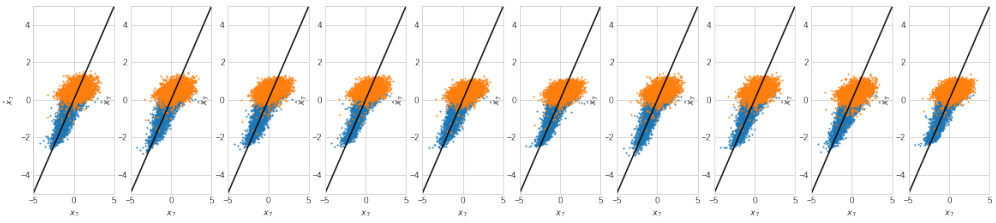
(a) $\hat{x}_7$ versus x_7



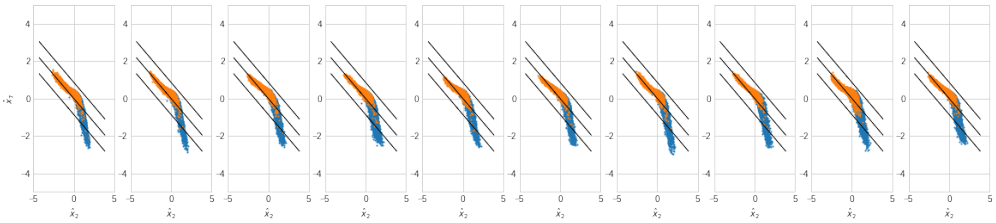
(b) $\hat{x}_7$ versus $\hat{x}_2$

Figure 5.28: M (P) results for $n = 1000$ training samples under MAR with a frozen decoder. When freezing the decoder before turning on the penalty, the penalty's effect is much weaker compared to the plain M (P) model.

in MAR settings. In particular, we ignore all models with a missingness model and only consider one specification of partial VAEs, namely the PVAE (MAX) specification.



(a) $\hat{x}_7$ versus x_7



(b) $\hat{x}_7$ versus $\hat{x}_2$

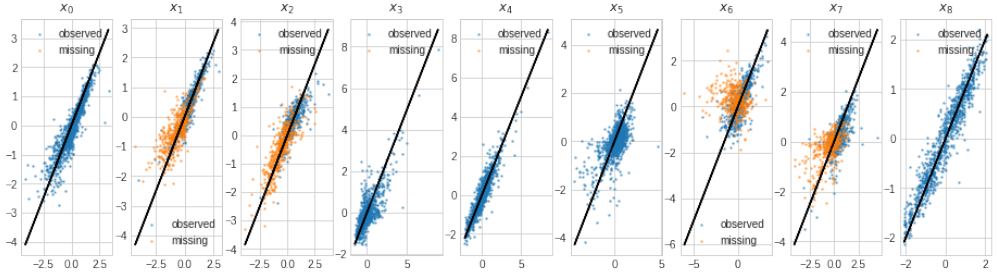
Figure 5.29: VAE (ET) results for $n = 1000$ training samples under MAR. The generated orange dots are very close to $\mathbb{E}[x_7|x_2]$ based on our prior knowledge.

Furthermore, we do not include various imputation strategies, but solely rely on a VAE with $\tilde{\tilde{\mathbf{x}}}$ as both encoder input and target instead. Our treatment missingness specification is more realistic compared to the process defined for simulated datasets and allows up to four variables (x_1, x_2, x_6, x_7) to be missing simultaneously. Note that three of them are highly correlated with their respective missingness determining variables and comparatively easy to correctly reconstruct and generate. This is not the case for variable x_6 though, which is why we will mainly concentrate on what works and fails in terms of x_6.

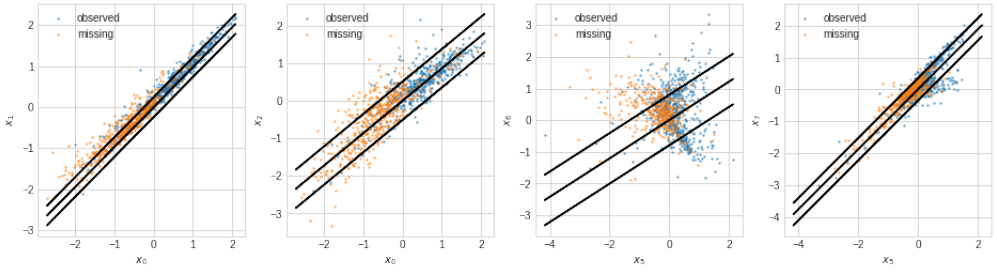
Table 5.6 presents the results for the treatment data. "MSE" columns contain reconstruction errors (O) and imputation errors (U) while "Residuals" columns contain the corresponding residuals indicating whether over- or underestimation takes place. The subsets denoted by (O) and (U) are variable specific, e.g. MSE-1 (O) is the mean squared reconstruction error of variable x_1 for all samples where variable x_1 is still observed after applying the missingness mechanism. Consequently, MSE-1 (U) is the mean squared reconstruction error of variable x_1 for all samples where x_1 is corrupted after applying the missingness mechanism.

As Table 5.6 shows, the MSE-6 (U) is very high in case of the models MASKED (1.57 (0.87)), ZERO (1.52 (0.66)), M (1.21 (0.51)) and PVAE (MAX) (1.38 (0.55)) when compared to the simple VAE (0.29 (0.13)) with uncorrupted observations. In particular, beware of the high standard deviations in parantheses indicating that results for different runs can be quite volatile. As outlined in the experimental setup section, all treatment results are averaged over ten training runs for ten different training sets, each with $n = 100$ samples. In general, small datasets tend to pose a higher danger of variance problems and overfitting. Nevertheless, the visualizations show plots for one run only to save space.

Making use of the prior knowledge penalty decreases the MSE-6 (U) and particularly reduces variance in the results. By comparing the ZERO results in Figure 5.30 with the ZERO (P) results in Figure 5.31, we can detect that the shape of $\hat{x}_6$ versus $\hat{x}_5$ clearly changes under the prior knowledge penalty. In the third plot in the bottom row of Figure 5.30, multiple orange dots lie outside of the prior knowledge interval defined by the outer two black lines, while in Figure 5.31, almost all orange points either lie within the prior knowledge interval or close to it. Obviously, such prior knowledge penalization regularizes the model to yield more robust results and is therefore especially capable of preventing it from very bad results in small data settings like the present.



(a)

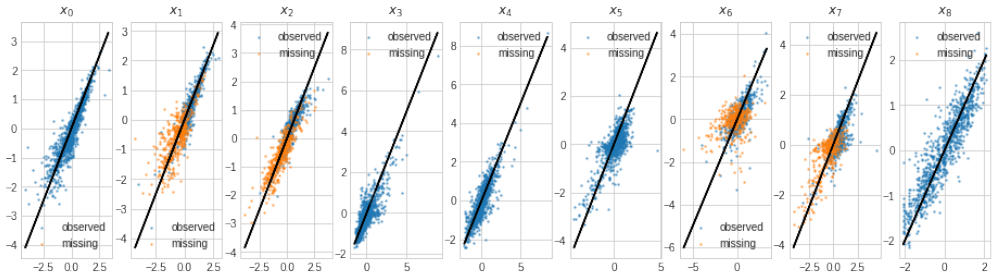


(b)

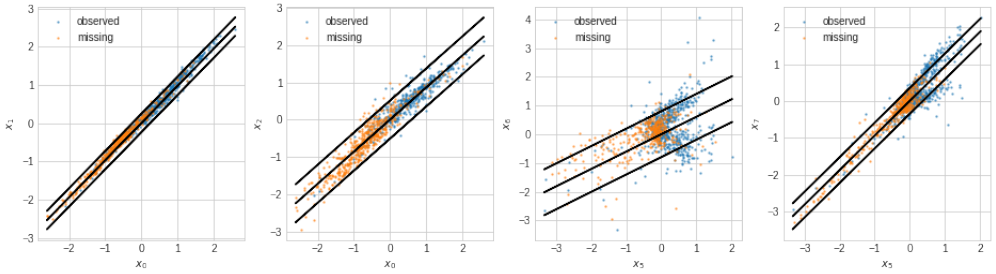
Figure 5.30: ZERO results for treatment data. The quantitative results are bad and many values of $\hat{x}_6$ are far off the true x_6.

On the contrary, reweighting (cf. ZERO (DW) and M (DW)) does not have such a regularizing effect. Instead, our weighting scheme puts even more weight on some rare instances in an already small dataset. The potential danger becomes clear when comparing M and M (DW). The average MSE-6 (U) increases from 1.21 to 1.64 and more importantly the standard deviation across the ten training runs increases from 0.51 to 1.05.

Using a vanilla VAE with corrupted values replaced by their conditional expectation based on the prior knowledge leads to the best MSE-6 (U) result, or more generally the best MSE (U) results among all models with corrupted data. Instead of extending the objective by a penalty, we directly alter the data to guide the model to a more robust and conservative solution. As we can see in Figure 5.32, replacing missing values in x_1, x_2, x_6



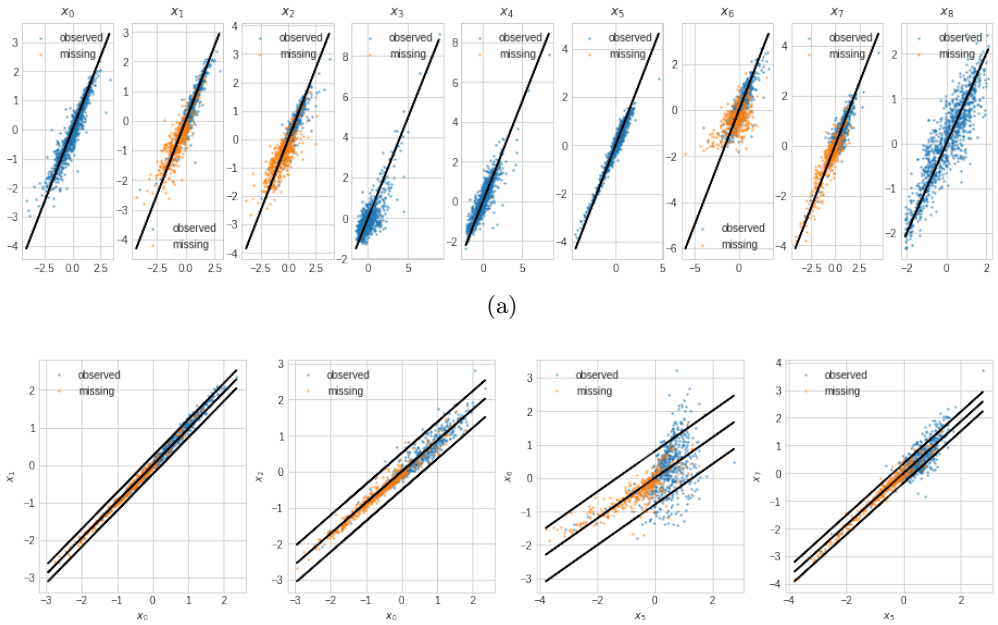
(a)



(b)

Figure 5.31: ZERO (P) results for treatment data. The prior knowledge penalty pushes orange dots closer to or inside the prior knowledge interval regularizing the model.

and x_7 by the conditional expected values with respect to a simple linear regression leads to decoder generations along the expected value line in the center of the prior knowledge interval. This result is in agreement with the result seen on simulated data, where the training sets were 10 and 100 times larger though.



(a)

(b)

Figure 5.32: VAE (ET) results for treatment data. Imputing missing values based on the prior knowledge generates orange dots close to their expected value in terms of our simple linear regressions.

	MSE (O)	MSE (U)	MSE-2 (O)	MSE-2 (U)	MSE-7 (O)	**MSE-7 (U)**	Residuals-2 (O)	Residuals-2 (U)	Residuals-7 (O)	**Residuals-7 (U)**
$n = 1000$										
VAE	0.29 (0.00)	0.30 (0.00)	0.26 (0.01)	0.29 (0.01)	0.07 (0.00)	**0.07 (0.01)**	-0.19 (0.02)	0.23 (0.02)	-0.02 (0.03)	**0.05 (0.03)**
CC	0.29 (0.00)	0.31 (0.00)	0.22 (0.00)	1.04 (0.05)	0.09 (0.01)	**0.13 (0.01)**	0.02 (0.02)	0.89 (0.03)	0.00 (0.02)	**0.14 (0.03)**
CC (W)	0.29 (0.01)	0.36 (0.02)	0.22 (0.02)	0.36 (0.06)	0.33 (0.08)	**0.48 (0.06)**	0.00 (0.02)	0.28 (0.09)	0.03 (0.02)	**0.26 (0.05)**
CC (S)	0.58 (0.03)	0.62 (0.03)	0.40 (0.07)	0.75 (0.16)	0.65 (0.06)	**0.55 (0.07)**	-0.19 (0.04)	-0.17 (0.15)	0.03 (0.03)	**0.04 (0.11)**
MASKED	0.25 (0.00)	0.26 (0.00)	0.22 (0.02)	0.26 (0.01)	0.73 (0.00)	**0.68 (0.00)**	-0.15 (0.03)	0.23 (0.02)	0.05 (0.02)	**-0.04 (0.03)**
ZERO	0.28 (0.00)	0.26 (0.00)	0.25 (0.01)	0.22 (0.01)	0.25 (0.03)	**1.00 (0.05)**	-0.15 (0.02)	0.20 (0.01)	0.04 (0.02)	**0.38 (0.07)**
M	0.29 (0.01)	0.26 (0.00)	0.24 (0.01)	0.22 (0.02)	0.16 (0.06)	**1.64 (0.29)**	-0.09 (0.04)	0.17 (0.02)	0.01 (0.02)	**0.79 (0.21)**
PVAE (MAX)	0.29 (0.00)	0.25 (0.00)	0.25 (0.01)	0.22 (0.01)	0.13 (0.03)	**1.35 (0.16)**	-0.07 (0.02)	0.13 (0.03)	0.01 (0.04)	**0.75 (0.11)**
PVAE (AVG)	0.30 (0.01)	0.27 (0.02)	0.25 (0.01)	0.22 (0.03)	0.20 (0.11)	**1.38 (0.37)**	-0.09 (0.04)	0.17 (0.03)	0.01 (0.02)	**0.70 (0.22)**
SPVAE (MAX)	0.29 (0.00)	0.26 (0.00)	0.26 (0.02)	0.24 (0.04)	0.11 (0.02)	**1.57 (0.39)**	-0.07 (0.02)	0.11 (0.03)	0.02 (0.03)	**0.87 (0.19)**
SPVAE (AVG)	0.31 (0.01)	0.29 (0.02)	0.25 (0.01)	0.23 (0.04)	0.35 (0.13)	**1.15 (0.43)**	-0.12 (0.05)	0.18 (0.03)	0.02 (0.02)	**0.55 (0.27)**
$n = 10000$										
VAE	0.29 (0.00)	0.29 (0.00)	0.26 (0.00)	0.28 (0.01)	0.08 (0.00)	**0.08 (0.00)**	-0.20 (0.01)	0.20 (0.02)	-0.05 (0.02)	**0.02 (0.02)**
CC	0.28 (0.00)	0.29 (0.00)	0.21 (0.00)	1.36 (0.07)	0.09 (0.00)	**0.13 (0.01)**	0.00 (0.03)	1.04 (0.03)	0.00 (0.02)	**0.04 (0.02)**
CC (W)	0.29 (0.01)	0.30 (0.01)	0.21 (0.01)	0.28 (0.03)	0.24 (0.07)	**0.38 (0.15)**	-0.09 (0.02)	0.23 (0.03)	0.00 (0.02)	**-0.02 (0.06)**
CC (S)	0.31 (0.01)	0.30 (0.01)	0.34 (0.03)	0.25 (0.01)	0.37 (0.10)	**0.40 (0.12)**	-0.32 (0.03)	0.08 (0.03)	-0.05 (0.05)	**-0.03 (0.02)**
MASKED	0.24 (0.00)	0.24 (0.00)	0.24 (0.01)	0.26 (0.01)	0.71 (0.00)	**0.68 (0.01)**	-0.19 (0.02)	0.20 (0.02)	-0.01 (0.03)	**-0.11 (0.04)**
ZERO	0.26 (0.00)	0.25 (0.00)	0.26 (0.01)	0.24 (0.01)	0.32 (0.01)	**0.80 (0.02)**	-0.18 (0.01)	0.19 (0.02)	-0.01 (0.02)	**0.22 (0.03)**
M	0.27 (0.00)	0.24 (0.00)	0.25 (0.01)	0.26 (0.01)	0.25 (0.02)	**1.13 (0.10)**	-0.16 (0.01)	0.20 (0.02)	0.00 (0.02)	**0.47 (0.04)**
PVAE (MAX)	0.24 (0.00)	0.24 (0.00)	0.26 (0.01)	0.24 (0.02)	0.23 (0.04)	**0.93 (0.40)**	-0.14 (0.02)	0.17 (0.03)	-0.01 (0.02)	**0.34 (0.25)**
PVAE (AVG)	0.25 (0.01)	0.24 (0.00)	0.26 (0.01)	0.25 (0.01)	0.22 (0.04)	**0.91 (0.19)**	-0.14 (0.01)	0.17 (0.01)	0.00 (0.02)	**0.35 (0.08)**
SPVAE (MAX)	0.25 (0.01)	0.24 (0.00)	0.27 (0.01)	0.24 (0.02)	0.23 (0.02)	**0.82 (0.06)**	-0.15 (0.03)	0.17 (0.02)	-0.02 (0.02)	**0.27 (0.06)**
SPVAE (AVG)	0.25 (0.01)	0.24 (0.00)	0.25 (0.02)	0.23 (0.01)	0.20 (0.05)	**1.27 (0.64)**	-0.11 (0.05)	0.13 (0.06)	0.00 (0.02)	**0.33 (0.63)**

Table 5.4: Results for benchmarks, baselines and models with corrupted input on simulated data under MAR.

	MSE (O)	MSE (U)	MSE-2 (O)	MSE-2 (U)	MSE-7 (O)	**MSE-7 (U)**	Residuals-2 (O)	Residuals-2 (U)	Residuals-7 (O)	**Residuals-7 (U)**
$n = 1000$										
M (GU)	0.29 (0.00)	0.26 (0.00)	0.23 (0.00)	0.19 (0.01)	0.12 (0.01)	**2.62 (0.50)**	-0.07 (0.02)	0.11 (0.02)	0.04 (0.04)	**1.27 (0.20)**
ZERO (GU)	0.28 (0.00)	0.26 (0.00)	0.24 (0.00)	0.21 (0.01)	0.24 (0.02)	**1.06 (0.06)**	-0.13 (0.02)	0.20 (0.02)	0.05 (0.04)	**0.46 (0.05)**
M (GOU)	0.29 (0.00)	0.25 (0.00)	0.22 (0.00)	0.20 (0.01)	0.08 (0.01)	**12.82 (1.95)**	0.00 (0.01)	0.03 (0.02)	0.02 (0.05)	**3.41 (0.27)**
ZERO (GOU)	0.29 (0.00)	0.27 (0.00)	0.20 (0.00)	0.13 (0.02)	0.18 (0.02)	**2.46 (0.21)**	0.00 (0.02)	0.04 (0.04)	0.03 (0.05)	**1.29 (0.07)**
M (TOU)	0.29 (0.00)	0.26 (0.01)	0.23 (0.00)	0.18 (0.02)	0.09 (0.00)	**7.77 (1.33)**	0.04 (0.02)	-0.02 (0.02)	0.00 (0.03)	**2.60 (0.25)**
ZERO (TOU)	0.29 (0.00)	0.27 (0.00)	0.21 (0.00)	0.12 (0.02)	0.17 (0.01)	**2.18 (0.22)**	0.01 (0.02)	-0.01 (0.02)	0.03 (0.03)	**1.18 (0.09)**
ZERO (DW)	0.28 (0.00)	0.26 (0.00)	0.26 (0.01)	0.26 (0.01)	0.32 (0.05)	**0.85 (0.03)**	-0.17 (0.02)	0.23 (0.02)	0.02 (0.02)	**0.16 (0.03)**
M (DW)	0.28 (0.00)	0.26 (0.00)	0.25 (0.01)	0.25 (0.01)	0.26 (0.04)	**0.97 (0.05)**	-0.13 (0.02)	0.20 (0.03)	0.00 (0.03)	**0.36 (0.04)**
ZERO (P)	0.28 (0.00)	0.26 (0.00)	0.25 (0.01)	0.22 (0.01)	0.27 (0.02)	**0.85 (0.03)**	-0.15 (0.02)	0.20 (0.01)	0.04 (0.02)	**0.28 (0.05)**
M (P)	0.28 (0.00)	0.26 (0.00)	0.24 (0.01)	0.23 (0.01)	0.20 (0.04)	**0.96 (0.06)**	-0.11 (0.03)	0.19 (0.03)	-0.01 (0.02)	**0.43 (0.12)**
VAE (ET)	0.28 (0.00)	0.26 (0.00)	0.25 (0.01)	0.20 (0.01)	0.31 (0.01)	**0.70 (0.00)**	-0.14 (0.02)	0.18 (0.01)	0.00 (0.02)	**0.05 (0.03)**
ZERO (EO)	0.28 (0.00)	0.26 (0.00)	0.25 (0.00)	0.20 (0.01)	0.24 (0.01)	**1.11 (0.06)**	-0.14 (0.01)	0.18 (0.01)	0.04 (0.02)	**0.47 (0.04)**
M (EO)	0.29 (0.00)	0.26 (0.00)	0.24 (0.00)	0.20 (0.01)	0.15 (0.02)	**1.95 (0.27)**	-0.09 (0.02)	0.16 (0.03)	0.01 (0.02)	**0.95 (0.09)**
M (TO)	0.28 (0.01)	0.26 (0.00)	0.24 (0.01)	0.22 (0.01)	0.31 (0.05)	**0.71 (0.01)**	-0.12 (0.02)	0.20 (0.02)	-0.02 (0.03)	**0.04 (0.02)**
$n = 10000$										
M (GU)	0.27 (0.00)	0.25 (0.00)	0.25 (0.01)	0.24 (0.01)	0.23 (0.03)	**1.41 (0.17)**	-0.16 (0.02)	0.17 (0.02)	0.00 (0.03)	**0.68 (0.07)**
ZERO (GU)	0.26 (0.00)	0.25 (0.00)	0.26 (0.01)	0.24 (0.01)	0.32 (0.02)	**0.83 (0.05)**	-0.18 (0.03)	0.20 (0.03)	0.00 (0.03)	**0.30 (0.06)**
M (GOU)	0.28 (0.00)	0.24 (0.00)	0.21 (0.00)	0.22 (0.00)	0.07 (0.01)	**19.27 (1.02)**	0.00 (0.02)	0.00 (0.02)	0.00 (0.03)	**4.25 (0.11)**
ZERO (GOU)	0.28 (0.00)	0.26 (0.01)	0.20 (0.00)	0.16 (0.02)	0.14 (0.01)	**1.69 (0.18)**	0.00 (0.02)	0.02 (0.03)	-0.01 (0.03)	**0.96 (0.10)**
M (TOU)	0.28 (0.00)	0.24 (0.00)	0.22 (0.00)	0.23 (0.01)	0.07 (0.01)	**18.87 (3.07)**	0.03 (0.02)	-0.03 (0.02)	0.00 (0.02)	**4.19 (0.33)**
ZERO (TOU)	0.28 (0.00)	0.26 (0.01)	0.21 (0.00)	0.16 (0.03)	0.15 (0.02)	**1.52 (0.21)**	-0.01 (0.02)	-0.03 (0.03)	0.01 (0.02)	**0.84 (0.14)**
ZERO (DW)	0.29 (0.00)	0.25 (0.00)	0.25 (0.01)	0.25 (0.01)	0.20 (0.04)	**0.99 (0.03)**	-0.17 (0.01)	0.19 (0.02)	-0.02 (0.04)	**-0.28 (0.04)**
M (DW)	0.29 (0.00)	0.25 (0.00)	0.25 (0.01)	0.23 (0.02)	0.12 (0.02)	**1.35 (0.22)**	-0.15 (0.03)	0.16 (0.02)	0.00 (0.02)	**0.52 (0.33)**
ZERO (P)	0.26 (0.00)	0.25 (0.00)	0.26 (0.01)	0.24 (0.00)	0.32 (0.01)	**0.77 (0.01)**	-0.18 (0.01)	0.19 (0.02)	-0.01 (0.02)	**0.19 (0.02)**
M (P)	0.27 (0.00)	0.24 (0.00)	0.25 (0.01)	0.26 (0.01)	0.27 (0.02)	**0.86 (0.02)**	-0.17 (0.02)	0.20 (0.02)	0.00 (0.02)	**0.31 (0.03)**
VAE (ET)	0.26 (0.00)	0.25 (0.00)	0.26 (0.00)	0.23 (0.00)	0.36 (0.01)	**0.69 (0.00)**	-0.17 (0.01)	0.18 (0.01)	-0.03 (0.02)	**0.01 (0.02)**
ZERO (EO)	0.26 (0.00)	0.25 (0.00)	0.26 (0.01)	0.24 (0.01)	0.32 (0.01)	**0.82 (0.03)**	-0.18 (0.01)	0.19 (0.02)	-0.01(0.02)	**0.26 (0.03)**
M (EO)	0.27 (0.00)	0.25 (0.00)	0.25 (0.01)	0.25 (0.01)	0.24 (0.02)	**1.21 (0.11)**	-0.16 (0.02)	0.20 (0.02)	0.00 (0.03)	**0.52 (0.05)**
M (TO)	0.26 (0.00)	0.25 (0.00)	0.25 (0.01)	0.24 (0.01)	0.35 (0.02)	**0.69 (0.00)**	-0.16 (0.02)	0.19 (0.02)	-0.02 (0.03)	**0.02 (0.02)**

Table 5.5: Results for missingness models, weighted models, penalized models and models with imputation under MAR.

	MSE-1 (O)	MSE-1 (U)	MSE-2 (O)	MSE-2 (U)	**MSE-6 (O)**	**MSE-6 (U)**	MSE-7 (O)	MSE-7 (U)
VAE	0.05 (0.01)	0.07 (0.03)	0.05 (0.02)	0.09 (0.03)	**0.19 (0.13)**	**0.29 (0.13)**	0.19 (0.04)	0.19 (0.08)
MASKED	0.11 (0.04)	0.24 (0.16)	0.13 (0.03)	0.27 (0.09)	**1.08 (0.27)**	**1.57 (0.87)**	0.28 (0.06)	0.45 (0.31)
ZERO	0.10 (0.04)	0.29 (0.13)	0.11 (0.05)	0.28 (0.08)	**0.40 (0.16)**	**1.52 (0.66)**	0.21 (0.07)	0.42 (0.22)
M	0.09 (0.03)	0.32 (0.22)	0.10 (0.03)	0.32 (0.12)	**0.32 (0.16)**	**1.21 (0.51)**	0.21 (0.06)	0.39 (0.14)
PVAE (MAX)	0.09 (0.02)	0.35 (0.11)	0.10 (0.04)	0.39 (0.16)	**0.56 (0.15)**	**1.38 (0.55)**	0.20 (0.06)	0.46 (0.20)
ZERO (P)	0.08 (0.03)	0.18 (0.07)	0.12 (0.06)	0.26 (0.09)	**0.41 (0.16)**	**0.88 (0.12)**	0.21 (0.06)	0.24 (0.08)
M (P)	0.09 (0.03)	0.20 (0.04)	0.11 (0.04)	0.25 (0.06)	**0.35 (0.16)**	**0.84 (0.06)**	0.22 (0.05)	0.27 (0.06)
ZERO (W)	0.08 (0.03)	0.26 (0.11)	0.09 (0.03)	0.25 (0.09)	**0.37 (0.18)**	**1.16 (0.65)**	0.19 (0.05)	0.52 (0.37)
M (W)	0.09 (0.04)	0.27 (0.12)	0.08 (0.02)	0.24 (0.05)	**0.34 (0.16)**	**1.64 (1.05)**	0.20 (0.06)	0.41 (0.16)
VAE (ET)	0.06 (0.01)	0.10 (0.03)	0.07 (0.03)	0.25 (0.03)	**0.27 (0.12)**	**0.75 (0.04)**	0.21 (0.05)	0.21 (0.04)

	Residuals-1 (O)	Residuals-1 (U)	Residuals-2 (O)	Residuals-2 (U)	**Residuals-6 (O)**	**Residuals-6 (U)**	Residuals-7 (O)	Residuals-7 (U)
VAE	-0.05 (0.03)	0.04 (0.02)	-0.03 (0.03)	0.04 (0.03)	**0.01 (0.03)**	**0.02 (0.08)**	-0.10 (0.05)	0.06 (0.08)
MASKED	0.04 (0.05)	0.21 (0.13)	0.03 (0.04)	0.21 (0.11)	**0.04 (0.06)**	**0.17 (0.48)**	-0.02 (0.04)	0.23 (0.19)
ZERO	-0.01 (0.03)	0.23 (0.12)	-0.02 (0.04)	0.20 (0.07)	**0.02 (0.06)**	**0.38 (0.33)**	-0.06 (0.08)	0.26 (0.15)
M	-0.03 (0.04)	0.15 (0.22)	-0.03 (0.05)	0.19 (0.15)	**0.05 (0.05)**	**0.10 (0.39)**	-0.04 (0.05)	0.27 (0.11)
PVAE (MAX)	0.00 (0.05)	0.23 (0.16)	-0.01 (0.04)	0.23 (0.14)	**0.04 (0.05)**	**0.28 (0.42)**	-0.02 (0.06)	0.25 (0.12)
ZERO (P)	-0.04 (0.03)	0.12 (0.05)	-0.04 (0.05)	0.05 (0.06)	**0.00 (0.07)**	**0.05 (0.14)**	-0.07 (0.07)	0.11 (0.06)
M (P)	-0.05 (0.04)	0.13 (0.07)	-0.05 (0.04)	0.05 (0.07)	**0.04 (0.06)**	**-0.02 (0.15)**	-0.04 (0.06)	0.12 (0.07)
ZERO (W)	-0.03 (0.03)	0.21 (0.07)	-0.03 (0.03)	0.14 (0.08)	**0.03 (0.05)**	**0.25 (0.31)**	-0.05 (0.04)	0.30 (0.19)
M (W)	-0.05 (0.05)	0.20 (0.14)	-0.03 (0.04)	0.18 (0.07)	**0.06 (0.05)**	**0.41 (0.56)**	-0.05 (0.05)	0.26 (0.13)
VAE (ET)	-0.05 (0.04)	0.05 (0.03)	-0.02 (0.04)	0.00 (0.04)	**0.02 (0.04)**	**0.00 (0.02)**	-0.08 (0.06)	0.03 (0.03)

Table 5.6: Results for all experiments on the treatment dataset.

6 Discussion

To recap, we are interested in settings with missing at random data where we have prior knowledge about both the underlying missingness mechanism as well as some expectation about the unobserved values. We are mainly inspired by a medical setting where a doctor decides not to conduct further diagnostic tests because the recorded results already contain sufficient information about their results. Our work tries to incorporate such knowledge into variational autoencoders. We formulate missingness mechanisms in terms of binary decision trees, which are easy to construct and interpret by a human (domain expert). As a byproduct, we concretize the idea in [IMF20] by using an explicit missingness model in the form of a differentiable decision tree. Furthermore, we formalize prior knowledge about the actual missing values as a simple linear regression.

As preparatory work, we carry out multiple experiments under a self-censoring MNAR process. We face biased results when training models on corrupted data, which do not even improve when increasing the training set size. While generic missingness models can get lucky, it is the usage of the true missingness model that can guide the model to generate more suitable values, in our case larger values of $\hat{x}_7$ knowing that if $m_7 = 1$, x_7 is typically greater than 0. We also figure out that it makes a difference whether the encoder has access to the missingness mask $\mathbf{m}$ (M) or not (ZERO). We think that with access to $\mathbf{m}$ the model is more confident in producing larger values of $\hat{x}_7$. Unfortunately, the approach also has weaknesses. In particular, while the true missingness model reveals that x_7 is typically greater than zero when unobserved, not knowing the distribution of x_7 when greater than zero can still lead to undesired results.

Experiments under MAR settings constitute our primary investigation. In this setting, we see a clear improvement when increasing the training set size. However, for small data, the results in regions with only few observed values for a corrupted dimension are not satisfactory. Most importantly, making use of the true missingness model similar to the MNAR setting does not alleviate the problem. Unfortunately, under MAR missingness, knowing the true missingness process does not directly reveal new information about the missing values. While we can use the missingness process to adjust sample weights, we find this approach to be unreliable.

Our core idea is therefore to combine the architecture in [IMF20] with the idea of domain-adapted neural networks [MIM+18] and directly use prior knowledge about the missing values instead of just the missingness process. The idea to use a further penalty based on a prior knowledge interval helps guiding the model and allows to find the sweet spot between data and knowledge. We demonstrate that the approach is reasonable, especially in small data settings while it loses its efficacy the larger the training set gets. However, we also point out that the main impact is on the decoder.

Alternatively, we propose to directly alter the corrupted values with prior knowledge

based imputation. While it works very well in our experiments due to the simplicity of the data, the approach is rather intrusive compared to the penalty idea. The penalty approach lets the data speak within the prior knowledge interval and does not penalize unless a certain amount of deviation from the expected occurs. This distinction particularly becomes relevant when the prior knowledge is not perfectly accurate.

Overall, our experiments with real-life EHR data confirm our findings on simulated data.

Notice that our work has some limitations:

- The assumed setting is unrealistic. We investigate a setting where all missingness is assumed to be MAR. In real-life, at least some variables with missingness might be corrupted by an MNAR process though. Moreover, real-life prior knowledge might be available in a different format than that of a simple linear regression.
- The data used in our experiments is very simple. Some methods that are successful in our experiments might fail in practice due to more complicated dependencies. Also, deep learning methods are often applied to unstructured data. It is not straightforward to extend our ideas to e.g. image datasets.
- In addition, the specified missingness mechanisms are very simplistic. While it is easily possible to specify a more complicated process as a binary decision tree, it is not clear what implications that would bring about.
- Due to the simplicity of our data, our models are very simple too. For more complicated real-life settings, some questions remain unanswered: Can a more flexible latent space distribution help to deal with missing data? Is a more sophisticated optimization schedule where the weight of the prior knowledge penalty is increased during training beneficial? In the end, all our experiments make use of rather shallow models, mostly with sigmoid activation functions, which does not conform with the state of the art in deep learning.

Despite these and maybe further limitations, we hope to inspire subsequent work to incorporate prior domain knowledge into the learning of deep latent variable models to make them succeed whenever good data is not available in abundance.

Bibliography

[BFOS17] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[BGS15] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[BPR20] Iavor I Bojinov, Natesh S Pillai, and Donald B Rubin. Diagnosing missing always at random in multivariate data. *Biometrika*, 107(1):246–253, 2020.

[BVV+15] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[CNW20] Mark Collier, Alfredo Nazabal, and Christopher KI Williams. Vaes in the presence of missing data. *arXiv preprint arXiv:2006.05301*, 2020.

[DKW+17] Arka Daw, Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[FH17] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GBWD+18] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[GCHK21] Sahra Ghalebikesabi, Rob Cornish, Chris Holmes, and Luke Kelly. Deep generative missingness pattern-set mixture models. In *International Conference on Artificial Intelligence and Statistics*, pages 3727–3735. PMLR, 2021.

[GHH+21] Yu Gong, Hossein Hajimirsadeghi, Jiawei He, Thibaut Durand, and Greg Mori. Variational selective autoencoder: Learning from partially-observed heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 2377–2385. PMLR, 2021.

[HMP+16] 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.

[HPM+20] Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

[HT52] Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

[IMF20] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. not-miwae: Deep generative modelling with missing not at random data. *arXiv preprint arXiv:2006.12871*, 2020.

[KFCB15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[KGGV19] Dhruv Khattar, Jaipal Singh Goud, Manish Gupta, and Vasudeva Varma. Mvae: Multimodal variational autoencoder for fake news detection. In *The world wide web conference*, pages 2915–2921, 2019.

[KMJRW14] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[KW19] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019.

[LR19] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[LROI21] David K Lim, Naim U Rashid, Junier B Oliva, and Joseph G Ibrahim. Handling non-ignorably missing features in electronic health records data using importance-weighted autoencoders. *arXiv preprint arXiv:2101.07357*, 2021.

[Mar08] Benjamin Marlin. *Missing data problems in machine learning*. PhD thesis, 2008.

[MF19] Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International conference on machine learning*, pages 4413–4423. PMLR, 2019.

[MIM+18] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pages 36–45. IEEE, 2018.

[MR15] Fabrizia Mealli and Donald B Rubin. Clarifying missing at random and related definitions, and implications when coupled with exchangeability. *Biometrika*, 102(4):995–1000, 2015.

[MTP+18] Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin, and Cheng Zhang. Eddi: Efficient dynamic discovery of high-value information with partial vae. *arXiv preprint arXiv:1809.11142*, 2018.

[MZ21] Chao Ma and Cheng Zhang. Identifiable generative models for missing not at random data imputation. *Advances in Neural Information Processing Systems*, 34:27645–27658, 2021.

[NOGV20] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107:107501, 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[Sad20] Mujiono Sadikin. Ehr dataset for patient treatment classification, 2020.

[SGJC13] Shaun Seaman, John Galati, Dan Jackson, and John Carlin. What is meant by "missing at random"? *Statistical Science*, 28(2):257–268, 2013.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[SNM16] Masahiro Suzuki, Kotaro Nakayama, and Yutaka Matsuo. Joint multimodal learning with deep generative models. *arXiv preprint arXiv:1611.01891*, 2016.

[TBL18] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[THHM15] Nicholas J Tierney, Fiona A Harden, Maurice J Harden, and Kerrie L Mengersen. Using decision trees to understand structure in missing data. *BMJ open*, 5(6):e007450, 2015.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[VRMB+19] Laura Von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems. *arXiv preprint arXiv:1903.12394*, 2019.

[YMH18] Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

[ZBFA08] David Zakim, Niko Braun, Peter Fritz, and Mark Dominik Alscher. Underutilization of information and knowledge in everyday medical practice: Evaluation of a computer-based solution. *BMC Medical Informatics and Decision Making*, 8(1):1–12, 2008.

[ZBKM18] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.