Self-Supervised Learning for Tabular Data: Analysing VIME and introducing Mix Encoder

Max Svensson

LUND
UNIVERSITY

Thesis for the degree of Bachelor of Science in Physics
Thesis advisor: Mattias Ohlsson

Made at the Division of Computational Biology and Biological Physics in the autumn semester 2023. Will be presented at the Department of Physics at Lund University in February 2024.

Abstract

We introduce Mix Encoder, a novel self-supervised learning framework for deep tabular data models based on *Mixup* [1]. Mix Encoder uses linear interpolations of samples with associated pretext tasks to form useful pre-trained representations. We further analyze the viability of tabular self-supervised learning by introducing VIME [2], an established representation learning framework for tabular data structures, to scarce healthcare datasets. We demonstrate that Mix Encoder outperforms VIME and a normal MLP in classifying breast cancer tabular data as well as show that both self-supervised learning frameworks can grant deep tabular models increased performance. Finally, we demonstrate that the combination of both representations, VIME and Mix, can yield even higher performance on certain datasets, such as early classification of diabetes.

Populärvetenskaplig beskrivning

Having a meaningful conversation with a machine can be an incredible, yet slightly eerie, experience. Being able to tell it to write a short poem can make you crack a smile, but at the same time, it can make you cringe. With the recent breakthrough of self-supervision, a framework that largely removes humans as a necessary part of a machine's training, humans are now getting used to the prospect of machines aiding us in various writing tasks. But we have yet to feel the profound impact of self-supervision across other, possibly more urgent, domains.

It is easy to think that the solution to crossing intellectual domains is to just keep training our machine learning models on more and more text. After all, this is what made them successful, right? Perhaps surprisingly, this is no longer the belief among experts who now understand that text is limited, not only in content but as a medium. Painfully obvious statements, that are not obvious to machines, are rarely written down, and such a thing as common sense is often not discovered by reading, but by experiencing. Forcefully pursuing this path when we enter even more foreign concepts than creative writing might be utterly pointless. Important domains such as economy, science, and healthcare are mostly based on tabular presentations of data that have so far been inaccessible to the self-supervised approach. New studies, applying to tabular data the ideas that have made language models so successful, hint at great improvements in speed and performance over previous models, both of which are of utmost importance in healthcare. The studies take the main idea of self-supervision, namely removing the need for manually labeled data, and are instead letting the machines understand the data by augmenting what they are given. The type of augmentations is most clearly explained in the context of vision. A model can be made to rotate an image of a dog and later compare it with the original, non-rotated image. It is different, yes, but not as different as an image of a cat would have been, much like turning our heads sideways does not fundamentally change the world around us. If the model understands this, it has found some understanding of how our world functions and can, hopefully, apply it effectively. With this type of training, the size of the data at hand becomes less important, such that possibly life-saving predictions on both very small and very large data sets can gain a significant performance boost.

Following the studies, it can now be said that some sort of nuanced understanding can be found in tabular data. The deep implications this has can resonate across all large tabular-dominated domains, sparking new fascinating discoveries in important societal issues, such as cancer research. In this project, we apply this new approach to tabular cancer treatment data, confident that taking this journey further, by introducing novel augmentations such as mixing, will prove once more that we can teach machines to not only process information, but dig deeper and reveal its secrets.

Contents

List of acronyms

SSL (Self-Supervised Learning)

VIME (Value Imputation and Mask Estimation)

MLP (Multilayer perceptron)

ReLU (Rectified Linear Unit)

t-SNE (t-Distributed Stochastic Neighbor Embedding)

List of Figures

List of Tables

1 Introduction

Supervised learning with deep neural networks has shown great success on a variety of tasks given labeled data. However, acquiring large amounts of carefully labeled data is often very costly, or even impossible in the cases where it does not exist. This bottleneck prevents models solely trained with supervised learning from utilizing the entirety of the existing data; especially in the domains of text and images where an abundance of potentially useful unlabeled data is known to exist. These large domains have thus provided opportunities for alternate forms of training to emerge that can extract useful information from the data, even if it is unlabeled. For this purpose, self-supervised learning supplies deep neural networks with a pre-training process in many areas where normal, supervised learning has reached a plateau [3]. Recently, self-supervised learning has shown great progress in doing so and has escalated the usage of artificial intelligence on text and images to the mainstream. Still, many data structures lend themselves poorly to this and have yet to be subject to this new approach.

Even with the success of self-supervised learning in text and image domains, approaches to tailor its application to tabular data are still in their infancy. Based on the self-supervised concepts of contrastive learning and pretext tasks [4], several papers are taking steps to accommodate this early gap in applicable domains. The gap mainly stems from the tasks that are commonplace among images and text not applying to tabular data due to its differences in spatial and semantic relationships, both of which are usually non-existent in tabular data. Furthermore, the tight combination of continuous and categorical data is often unique to tables and provides further complications. Examples of tasks, such as rotation of images and next-word prediction, have no clear equivalent for tables and must be adapted. Despite early works of pretext tasks and contrastive approaches that are relevant for tabular data showing promising results, self-supervised learning in this domain remains largely unexplored.

Contribution: In this thesis, we introduce a previously established self-supervised learning framework for tabular data, namely VIME, to authentic real-world data in the form of scarce healthcare datasets. We further propose a novel tabular self-supervised learning framework by introducing *Mixup* [1], a well-known framework for creating out-of-distribution data points, as a pretext task by adding a mixing and restorative task. To solve the pretext task, an encoder will learn to minimize their combined loss and thereby construct a representation space containing useful information about the raw featured data. Finally, we combine the representation outputs of VIME and *Mixup* in an attempt to gain a more nuanced and wider view of the scarce data at hand.

1.1 Related Works

Self-supervised learning (SSL) is an unlabeled representation learning framework that has garnered a great deal of attention as large language models (LLMs) have entered the

mainstream. To achieve training on unlabeled data, self-supervised learning utilizes two approaches: contrastive learning and pretext task(s).

Contrastive learning is a framework where positive and negative samples are generated from some anchor sample. Relative to this anchor, their representation vectors are then pushed closer or further away in representation space with the help of a contrastive loss [4]. This approach has not been used in this thesis although it has played a major part in advancing the field of self-supervised learning.

Pretext tasks are predefined tasks that create auxiliary targets for the model. A common scheme is to apply augmentations to the data such that the corrupted parts can become the generated labels for the model to predict. The current and most well-known pretext tasks are applicable only to text and images. Some of these include: image colorization, missing patch prediction, rotation prediction, solving jigsaw puzzles [3], next-sentence prediction, and masked-language modeling [5].

Exploration of pretext tasks that are applicable to tabular data has recently also been made. In TabNet, Arik et al. (2020) use the popular transformer architecture to extract relevant features [6]. Transformers introduce the concept of only using attention where the model can capture the relationships of a data sequence and output a single representation of their correlations by maintaining global dependencies between input and output [7]. In text, this helps models understand the context of words in a long sentence, which other networks might struggle to grasp. Furthermore, TabNet utilizes feature selection as a corruption scheme during the learning process. In doing so, it is not wasting learning capacity on irrelevant features, which ultimately makes it more efficient. Feature selection in TabNet is a trainable step, mainly done by using a transformer-based masking process where the mask is trained to selectively find the relevant features. By also utilizing attentive transformers, a representation of the input is then created. This is later decoded and reconstructed into tabular data using a transformer-decoder model. TabNet has shown promising results on complex tabular data sets.

More recently, Hajiramezanali et al. (2022) introduced STab [8]; an attempt to move away from the corruptive approach and introduce an augmentation-free framework that aims to capture non-structured correlations as commonly found in tabular data. The paper argues that the existing contrastive methods are domain-specific and are therefore inapplicable to tabular structures. They also note that their performance heavily relies on access to large computational resources. To solve this issue, STab employs a Siamese model with two different encoders, both of which are subject to stochastic regularization at each layer. To provide different views of the data, the regularizations differ between the encoders. A contrastive, cosine distance loss then measures the similarity between the outputs and is optimized. However, models like these commonly converge to trivial solutions. This is overcome in STab by utilizing a stop-gradient operation technique [8].

Similarly, other works have applied *Mixup* as a pretext task, but this was done in the context of images [9]. To our knowledge, this thesis is the first paper that introduces

Mixup as a self-supervised learning pretext task for tabular data.

2 Background theory

This section aims to introduce the general formulation of self-supervised learning and how it can be used to stretch small data sets and help us extract information from very large ones, as well as detailing the specific approaches relating to this thesis.

2.1 Supervised learning

In supervised learning, assume that we are given some labeled data $D_l = \{x_n, d_n\}_{n=1...N_l}$, and in some cases also some unlabeled data $D_u = \{x_n\}_{n=N_l+1...N_u}$ where each $x_n \in \mathcal{X} \subseteq \mathbb{R}^d$ is sampled independently from an unknown feature distribution P_X, and each labeled data pair $D_l = (x_n, d_n)$ is sampled independently from a joint distribution $P_{X,D}$ [10]. Given a labeled sample, a neural network outputs an estimation $y(\mathbf{x}, \omega)$ of the conditional average $\langle d|x\rangle$, where ω are the weights of the network. In the learning procedure, the model then minimizes the empirical supervised loss $l(y(\mathbf{x}, w), d)$. Here, l is some loss function, e.g. mean-squared error for continuous values.

Supervised learning runs into issues when the labeled data set is limited, i.e. when N_l is small. In this case, the model is very likely to overfit the training data as the empirical supervised loss deviates from the expected supervised loss, $\mathbb{E}(l(y(\mathbf{x}, w), \mathbf{x}))$, of the true data distribution. Furthermore, apart from using regularization techniques, creating a large labeled data set to circumvent this issue while also increasing model performance is often very costly and time-consuming.

In self-supervised learning, the final step of training is often supervised, commonly referred to as the "downstream task". The pre-trained model, trained using self-supervision, is then adapted, or "fine-tuned", to perform well on the original task of predicting d_n. This proceeds using normal supervised learning where the loss then backpropagates through the joint predictive and pre-trained models.

2.2 Self-supervised learning

Self-supervised learning is a representation learning framework that utilizes the entirety of $D = D_l + D_u$ by forming auxiliary labels $d_s \in D_s$ from augmentations of $\mathbf{x}$. Although, if D_u does not exist, the entirety of D will consist of D_l momentarily stripped of its proper labels. This process is often called pretext task or pretext learning where the tasks of predicting d_s require finding relationships that are challenging enough and are appropriate for the downstream task, i.e., when we fine-tune the model afterward using only D_l. In this

way, the model can extract relevant information and construct informative representations of D as dictated by the pretext task.

To this purpose, self-supervised learning employs an encoder $e(\mathbf{x})$ which takes an augmented sample $\mathbf{x_s} \in \mathcal{X}$ and outputs a representation $\mathbf{z} \in \mathcal{Z}$ as $e : \mathcal{X} \to \mathcal{Z}$. The representation can then be optimized with e.g. classification or restoration of $\mathbf{x}$ by having a predictive model estimate d_s, where the constructed labels here have retained some original information about $\mathbf{x}$. In computer vision, the rotation of an image is often used where the labels d_s hold the angle of the rotation. An encoder and a predictive model are then employed to estimate it. The predictive model can be written as $h(\mathbf{z})$ and performs the transformation $h : \mathcal{Z} \to D_s$. It is jointly trained along with the encoder during the pretext learning. Learning proceeds by minimizing the expected loss function,

$$\min_{e,h} \mathbb{E}_{(\mathbf{x}_s, d_s) \sim P_{X_s, D_s}} [l_s(d_s, (h \circ e)(\mathbf{x_s}))],$$

where l_s is some standard loss function and P_{X_s,D_s} is our pretext distribution as generated by the pretext task. As we can create a labeled pair $(\mathbf{x_s}, d_s)$ for every sample in D, we are estimating the expected objective function and not the empirical function, and therefore, albeit depending on the augmentations, it is usually very difficult for the model to overfit. The representation space found by the encoder is commonly called an embedding [11] where it has embedded a, usually, high-dimensional vector $\mathbf{x_s}$, in a lower-dimensional space. In this case, placing correlated inputs in some spatial manner.

Following self-supervised training, the encoder is then decoupled from the rest of the system and used along with a new predictive model $h_l(\mathbf{z})$ for the downstream task, which is the same task as for supervised learning. Ideally, the pre-trained representations $\mathbf{z} = e(\mathbf{x})$ that are output by the encoder when training h_l to predict the actual labels $d_n \in D_l$ now contain better correlative information about $\mathbf{x}$, which might improve performance. The degree of improvement is highly dependent on the pretext task which is therefore the focus of this thesis.

2.3 VIME

VIME (Value Imputation and Mask Estimation) is an autoencoder-based model proposed by Yoon et al. (2020) that adapts the self-supervised pretext tasks to tabular data [2]. Tabular data has up until recently been inaccessible to the existing pretext tasks as they are only relevant for images and text where spatial or semantic relationships are utilized.

VIME proposes a novel augmentation scheme by masking parts of the table, imputing the masked values by other existing values of the same column, and then introducing two pretext tasks: estimating the mask vector and restoring the original sample. The augmentation proceeds as follows: A mask generator outputs a binary mask vector $\mathbf{m} = [m_1, ..., m_d]^\top \in \{0, 1\}^d$ where each m_i is randomly chosen from $p_\mathbf{m} = \Pi_{j=1}^{d} \text{Bern}(m_i | p_m)$, i.e. a Bernoulli distribution with probability p_m. Taking this mask vector and a sample from

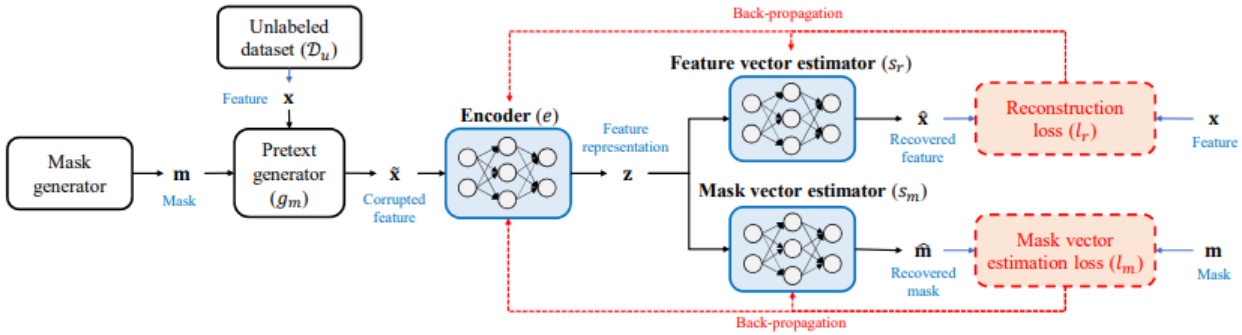


Figure 1: A diagram of the VIME framework for tabular data as given in the paper [2]. A pretext generator with input from a mask generator and an unlabeled dataset D_u generates a corrupted sample $\tilde{\mathbf{x}}$. An encoder takes this as input and generates a representation $\mathbf{z}$. Using $\mathbf{z}$, two estimators, s_r and s_m, predict two different pretext tasks: feature vector estimation, and mask vector estimation to optimize the encoder representations. Their joint loss backpropagates through all models.

D_u as input, a pretext generator $g_m : \mathcal{X} \times \{0,1\}^d \to \mathcal{X}$ overlays the mask vector on top of the sample vector and then imputes the masked positions with other random values of the same column. This process can be written as

$$\tilde{\mathbf{x}} = g_m(\mathbf{x}, \mathbf{m}) = \mathbf{m} \odot \bar{\mathbf{x}} + (\mathbf{1} - \mathbf{m}) \odot \mathbf{x},$$

where $\bar{\mathbf{x}}$ is a column-wise shuffled version of the sample $\mathbf{x}$. Column-wise shuffling of the augmented sample maintains the categorical and continuous separation of the columns, as opposed to mixing them and possibly leading the model astray. To this end, if several categorical columns are related and mutually exclusive, as will be the case for our study, we have taken the liberty of masking all of them if one is randomly chosen.

The amount of randomness, partially controlled by the hyperparameter p_m, sets the difficulty for the model and must be fine-tuned. If too many values are removed the model will not find any correlations, and if too few are removed, it will not challenge the model enough, possibly leading to dimensional collapse [12].

Given the corrupted sample $\tilde{\mathbf{x}}$, the encoder e transforms the sample to a representation $\mathbf{z}$. This is given as input to two predictive models with separate pretext tasks: mask vector estimation and feature vector estimation. The masked vector estimator $s_m : \mathcal{Z} \to [0,1]^d$, outputs a vector $\hat{\mathbf{m}} = (s_m \circ e)(\tilde{\mathbf{x}})$ that predicts which features have been replaced in $\tilde{\mathbf{x}}$. The feature estimator $s_r : \mathcal{Z} \to \mathcal{X}$ returns $\hat{\mathbf{x}}$ which is an estimate of $\mathbf{x}$.

The entire architecture consisting of e, s_m, and s_r is jointly trained by minimizing the expected loss

$$\min_{e,s_m,s_r} \mathbb{E}_{\mathbf{x}\sim p_X, \mathbf{m}\sim p_{\mathbf{m}}, \tilde{\mathbf{x}}\sim g_m(\mathbf{x},\mathbf{m})}[l_m(\mathbf{m}, \hat{\mathbf{m}}) + \alpha \cdot l_r(\mathbf{x}, \hat{\mathbf{x}})],$$

where l_m is the binary-cross entropy loss function, l_r is the mean squared error, and α is a hyperparameter that weighs the two losses against each other, giving priority to either. The encoder can then be decoupled and used for supervised downstream training on the labeled data set D_l. Figure 1 depicts an overview of VIME.

The model as it has been presented here will be used on our specific scarce data sets to analyze its usefulness and act as a benchmark. The semi-supervised part that was included in the original paper will not be included here as it is not the focus of this thesis.

2.4 Proposed model: Mix Encoder

This section introduces *Mixup* as a novel augmentation scheme for pre-training tabular encoders. It follows the same principles and ideas as VIME, albeit with different corruptions of the feature data.

The main concept of *Mixup* is to generate in-between samples of a labeled data set D_l by mixing two different input feature vectors and their target labels. In doing so, the labeled data set is extended and can obtain a more generalized performance on out-of-distribution data points. The process operates on the premise that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. A problem, as pointed out by Yoon et al. [2], is that this approach fails when the original data manifold is non-convex, i.e., when we incorporate categorical as well as continuous values, as is very common in tabular data.

Our proposed model, Mix Encoder, instead introduces *Mixup* as a representation learning framework where mixing can help the encoder separate differences between samples and find their correlations by introducing two pretext tasks: mix estimation and feature vector estimation. The pretext mixing introduces a mixing parameter $\lambda \in [0.5, 1]$ which is randomly drawn from a beta distribution as $\lambda \sim Beta(\alpha, \beta)$. This mixing parameter is shifted towards 1 and controls the amount of mixing between a leading sample $x_i \in \mathcal{X}$ and another randomly drawn sample $x_j \in \mathcal{X}$. See Appendix A for a detailed explanation of how λ is generated. The resulting mixed vector $\tilde{\mathbf{x}}$ is thus given by

$$\tilde{\mathbf{x}} = \lambda \mathbf{x_i} + (1 - \lambda)\mathbf{x_j}.$$

As before, an encoder $e(\mathbf{x})$ takes $\tilde{\mathbf{x}}$ as input and generates a representation $\mathbf{z}$. Two different predictive models, here named h_m, and h_r, take this as input and collaboratively estimate λ and $\mathbf{x_i}$. In detail, the two models are,

- **Mix parameter estimator:** $h_m : \mathcal{Z} \to [0, 1]$ takes $\mathbf{z}$ as input and outputs an estimation $\hat{\lambda} = (h_m \circ e)(\tilde{\mathbf{x}})$ of the mixing parameter λ which is shifted towards 1 such that $\mathbf{x_i}$ obtains the largest fraction of its values in $\tilde{\mathbf{x}}$.
- **Feature vector estimator:** $h_r : \mathcal{Z} \to \mathcal{X}$ takes $\mathbf{z}$ as input and outputs $\hat{\mathbf{x}} = (h_r \circ e)(\tilde{\mathbf{x}})$, an estimation of $\mathbf{x_i}$, the leading mixed feature vector.

The encoder and the predictive models together minimize the expected loss

$$\min_{e,h_m,h_r} \mathbb{E}_{\mathbf{x_i}\sim p_X,\mathbf{x_j}\sim p_X,\lambda\sim Beta(\alpha,\beta)}[l_m(\lambda,\tilde{\lambda}) + \theta \cdot l_r(\mathbf{x},\hat{\mathbf{x}})],$$

where both l_m and l_r are the mean-squared error function. Following training, the same type of supervised fine-tuning as before can be done using the labeled samples of D_l. An overview of the Mix architecture can be seen in Figure 2.

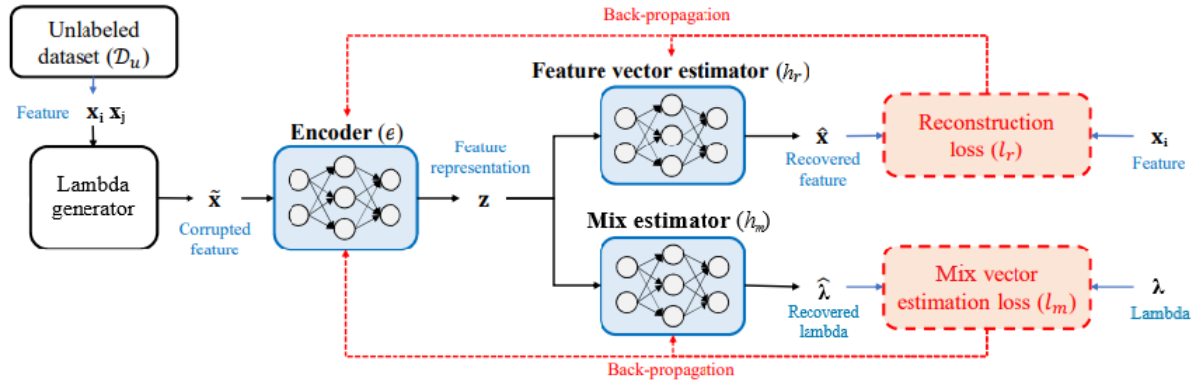


Figure 2: A diagram of the Mix framework for tabular data. A lambda generator with input from an unlabeled dataset D_u generates a corrupted sample $\tilde{\mathbf{x}}$. An encoder takes this as input and generates a representation $\mathbf{z}$. Using $\mathbf{z}$, two estimators, h_r and h_m, predict two different pretext tasks: feature vector estimation, and mix estimation to optimize the encoder representations. Their joint loss backpropagates through all models.

2.5 Combined Encoder

As an additional study, we analyze the practicality of combining the outputs of two different encoders when fine-tuning for our downstream task. The two pre-trained encoders, trained using VIME and Mix, will have their outputs concatenated to a single representation $\mathbf{z_c}$ which is then fed into a predictive model.

In more detail, given a VIME representation $\mathbf{z_v} = (z_{v1}, z_{v2}, ..., z_{vN})$ and a Mix representation $\mathbf{z_m} = (z_{m1}, z_{m2}, ..., z_{mN})$, the combined representation can be written as

$$\mathbf{z_c} = \mathbf{z_v} \frown \mathbf{z_m} = (z_{v1}, z_{v2}, ..., z_{vN}, z_{m1}, z_{m2}, ..., z_{mN}).$$

A predictive model $h_c(\mathbf{z_c})$ takes this as input and performs the same supervised training as before by predicting the labels d_n of our downstream task.

3 Method

3.1 Breast cancer dataset

As our main study, analysis of the models has been carried out on the so-called "N0 dataset"; an internal breast cancer dataset used here at Lunds University [13]. It contains 800 data points, 27 features, and 1 target class. The target, N0, denotes whether the cancer has spread to the lymph nodes (N0=1 no lymph node metastasis, N0=0 lymph node metastasis detected). The other features are data taken from a real healthcare setting. Some of the features include: Age, weight, height, tumor size, tumor localization, histological type, and estrogen receptor (positive or negative). Some features are continuous and some are categorical. Among the categorical, several features, namely diagnosis and position, are separated into several columns although they are mutually exclusive. To prevent the model from inferring these values from its related columns, should they be corrupted, instead of inferring from other features, we will mask all of the related columns should one be randomly chosen. This is only an issue when using VIME.

For self-supervised learning, we compare Mix Encoder and Combined Encoders against VIME. As a benchmark, both these pre-training models are also compared against an MLP (Multilayer Perceptron).

3.2 Early classification of diabetes dataset

As an additional study, an early classification of diabetes dataset is used [14]. The purpose of this study is to further analyze the viability of our pre-training, as well as compare the models on even smaller training data sizes where pre-trained models are expected to have an advantage. The same evaluations as for the breast cancer dataset are applied here.

The dataset contains 520 data points, 16 features, and 1 target class. Some of its features include: Age, gender, polyuria, sudden weight loss, delayed healing, polyphagia, and alopecia. Non-neural approaches to this dataset, such as XGBoost [15], have shown near-perfect accuracy and precision scores (99%) on this dataset [16]. As our study concerns the improvements self-supervised learning can grant neural networks, our benchmark is yet an MLP.

3.3 Model validation

Model validation is based on predicting the proper labels of the downstream task. To assess downstream model performance, pre-training and fine-tuning have been carried out in the following way:

For pre-training, the dataset is stripped of its proper labels and the self-supervised models

are pre-trained on 90% of the unlabeled data points. The remaining 10% is used as validation data to monitor pre-training performance. Monitoring is important to find appropriate hyperparameters and avoid overtraining, should it occur.

Given pre-trained encoders, downstream fine-tuning on the dataset with the proper labels can be done. To gain a broader understanding of the viability of the models, the models are fine-tuned on varying sizes of the training data. The dataset is therefore split randomly where the randomization is controlled using a seed. Between seeds, the training and validation data are thus different. For a given training size, we take the mean of a model's validation performance across several seeds to acquire its estimated performance. Note that as the size of the training data varies, the validation size remained at a constant 25% of the training data across all seeds. K-fold cross-validation is used to find appropriate hyperparameters across the different sizes. For replicability, all models are initialized the same, using Xavier initialization [17].

Also, when fine-tuning, loss only backpropagates through the pre-trained encoders after 300 epochs, such that the prediction heads can adjust before optimizing the embeddings.

Before any training, we use z-score normalization [18] to normalize both datasets. Furthermore, as we are analyzing small training splits, regularization during supervised fine-tuning is necessary to avoid overfitting. We have used L2 regularization on all downstream models. Lastly, if no validation loss improvement is seen within 70 epochs, the learning rate is reduced. This is known as "reduce learning rate on plateau" in PyTorch; a type of early stopping.

Performance for all models will be evaluated with AUC [19].

4 Results

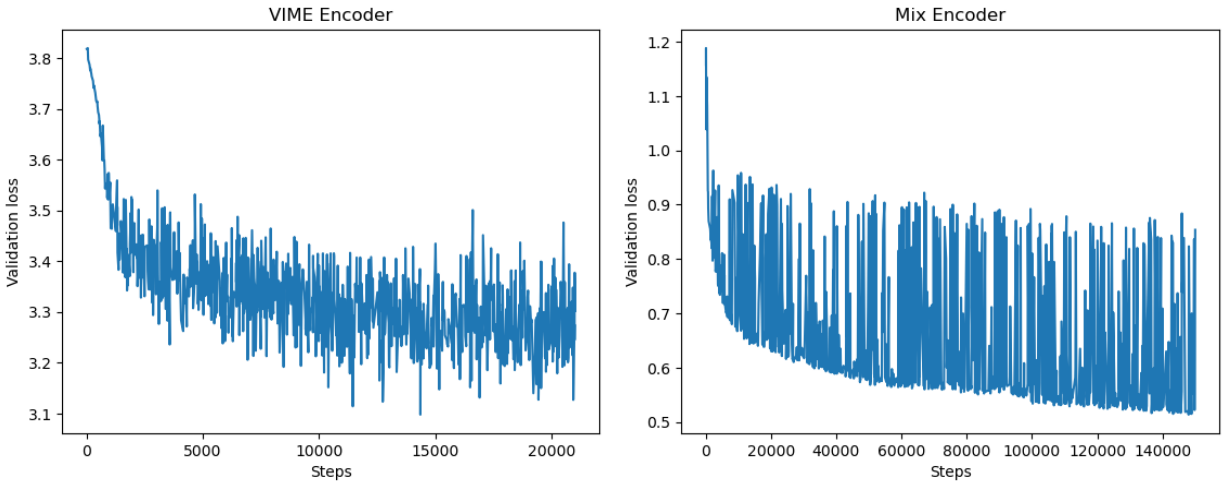
In this section, we evaluate the methods and present the results of pre-training and fine-tuning our models on both datasets. Once an appropriate size for the MLP benchmark had been found, this became the prediction heads for all other models. Throughout this analysis, the prediction heads across all models therefore consisted of 3 layers. The hidden size for each layer is equal to the number of input features. Naturally, this doubles the nodes of the Combined Encoder compared to the other models. Other hyperparameters can be found in Appendix A. Implementation of all models has been made using PyTorch and can be found on GitHub[1].

[1]https://github.com/msvenssons/Mix-Encoder

4.1 Pre-training results

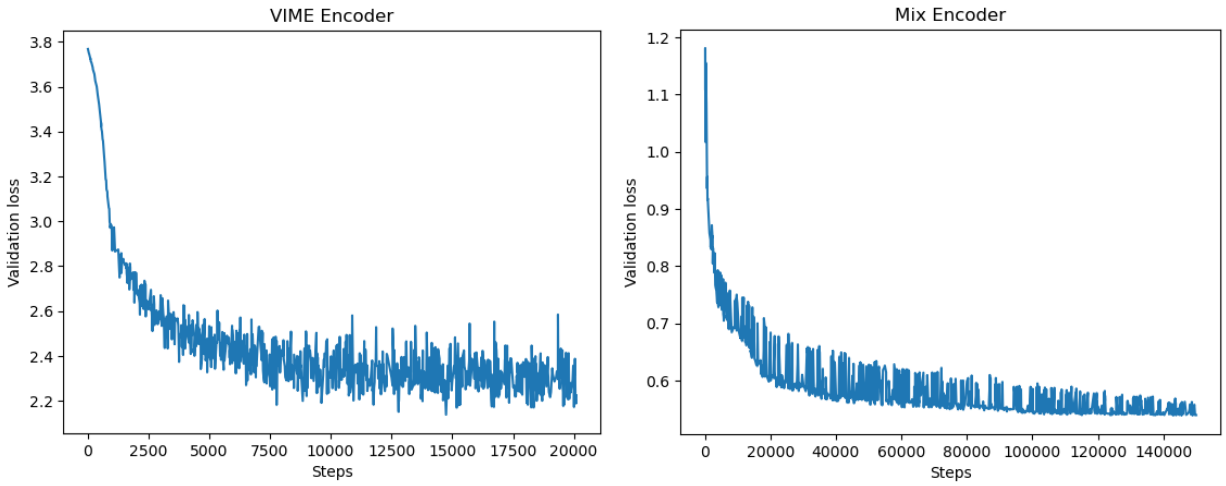
For pre-training, all of VIME's and Mix's models consist of 2 and 3 layers respectively, and for each dataset, the encoders are trained on 90% of the available data while the remaining is used for validation. Pretext learning and downstream performance depend heavily on hyperparameters, especially p_m and λ, for VIME and Mix respectively.

Example validation curves showing the VIME and Mix encoders optimizing their embeddings can be seen in Figures 3 and 4. The hyperparameters have mostly been kept the same between the figures. However, in Figure 3, the pretext task for each model is made more difficult to show an example curve of when the models struggle to find relationships. There, $p_m = 0.6$, and for Mix, $\lambda \sim 0.5 \leq Beta(2, 1) \leq 1$. Similarly, for Figure 4, $p_m = 0.2$ and $\lambda \sim 0.7 \leq Beta(5, 1) \leq 0.9$ (if $\lambda \geq 0.9$ then $\lambda = 1$; see Appendix A). Note that Mix reaches roughly the same loss, albeit with much more noise, for both runs.



(a) VIME (b) Mix

Figure 3: Validation loss when pre-training VIME and Mix encoders with bad p_m and λ on the early classification of diabetes dataset. Each encoder is trained on 90% of the dataset and validated on the remaining 10%. For VIME, $p_m = 0.6$, and for Mix, $\lambda \sim 0.5 \leq Beta(2, 1) \leq 1$.



(a) VIME (b) Mix

Figure 4: Validation loss when optimizing VIME and Mix encoders on the early classification of diabetes dataset. Each encoder is trained on 90% of the dataset and validated on the remaining 10%. For VIME, $p_m = 0.2$ and for Mix, $\lambda \sim 0.7 \leq Beta(5, 1) \leq 0.9$. Note that, if $\lambda \geq 0.9$ then $\lambda = 1$.

4.2 Downstream results

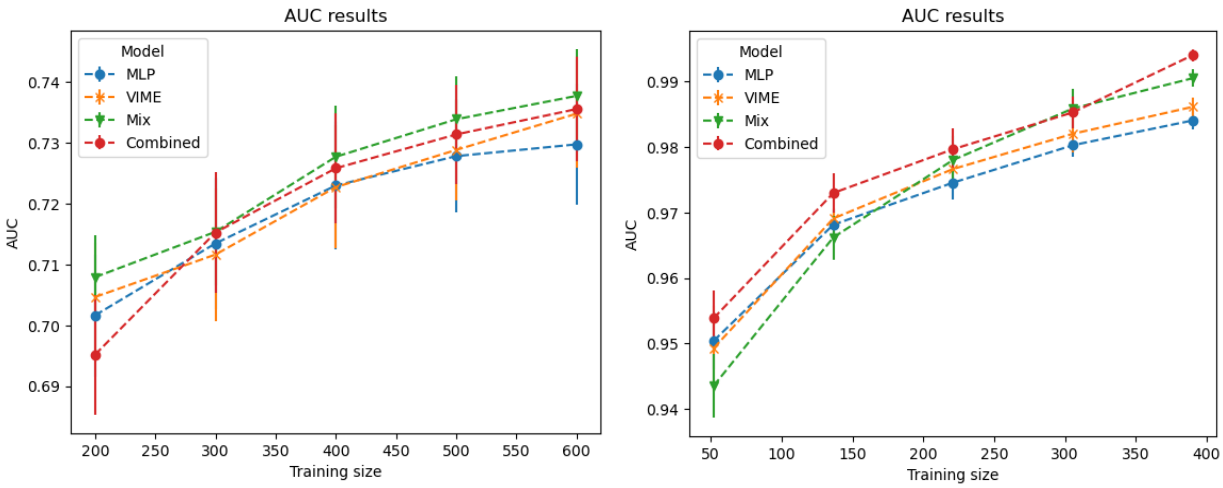
To acquire the desired downstream performances, all models were tasked with predicting the proper labels d_n on varying sizes of the labeled training data: 25%, 37.5%, 50%, 62.5%, 75% for breast cancer and 10%, 26.25%, 42.5% 58.75%, 75% for diabetes. In the process, the encoders became fine-tuned to perform well on this task. The experiments were then run over 10 different seeds for the 5 different escalating training sizes. The results can be found in Figure 5.

Figure 5a shows the AUC performance of all models on the breast cancer dataset for each training size. Our results show that Mix outperformed the other models on all training sizes. Moreover, for training sizes 37.5%, 50%, 62.5%, and 75% of the entire dataset, Combined also outperformed VIME and the benchmark MLP. On the smallest training size, 25%, the uncombined pre-trained models both exhibited higher performance than the benchmark MLP, as expected.

Figure 5b shows the AUC performance of all models on the diabetes dataset. Here, the Combined Encoder outperformed the other models on all training sizes. The results further reveal that on sizes 42.5%, 58.75%, and 75%, the pre-trained models showed a clear improvement over the benchmark. Meanwhile, Mix exhibited the lowest performance on both 10% and 26.25%.

Note that, for both datasets, all pre-trained models exhibited a higher AUC performance than the benchmark MLP for sizes >50%.

Tables detailing the AUC scores of the important limits, 10%/25%, and 75% of the available data, can be found in Tables 1 and 2 respectively.



(a) Breast cancer dataset

(b) Diabetes dataset

Figure 5: AUC scores for MLP, VIME, Mix, and Combined encoders on the breast cancer and diabetes datasets. Results are given as the mean of 10 runs with different randomized data splits and evaluated on different sizes of the training data.

Table 1: AUC scores when using 25% (breast cancer) and 10% (diabetes) of the available data for downstream training on all models (the higher the better). Each mean and standard deviation is computed over 10 different seeds.

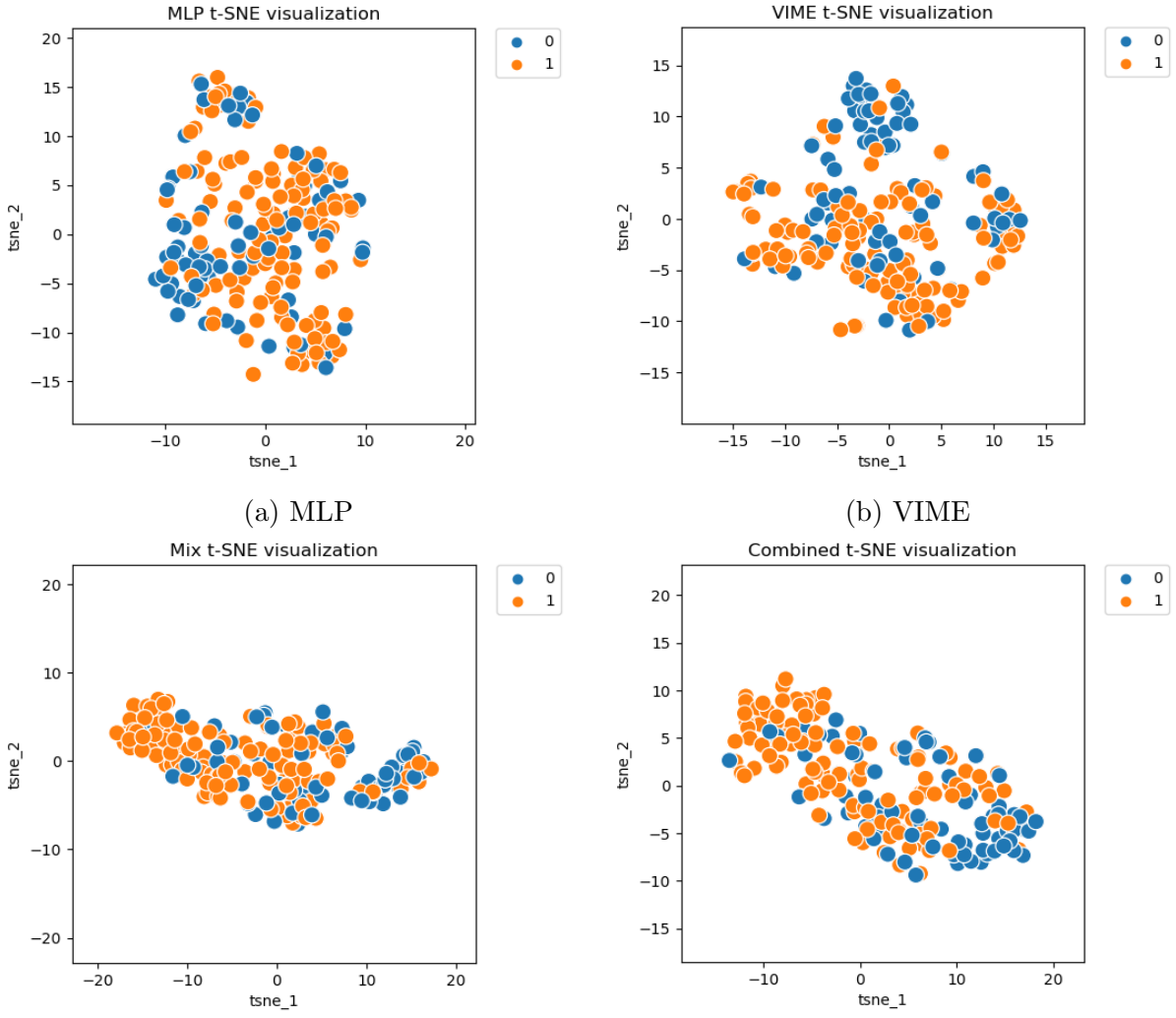
Model	Breast cancer	Diabetes
MLP	0.7017 ± 0.0100	0.9504 ± 0.0044
VIME	0.7047 ± 0.0088	0.9493 ± 0.0065
Mix	**0.7079 ± 0.0069**	0.9435 ± 0.0048
Combined	0.6952 ± 0.0099	**0.9539 ± 0.0043**

Table 2: AUC scores when using 75% of the available data for downstream training on all models (the higher the better). Each mean and standard deviation is computed over 10 different seeds.

Model	Breast cancer	Diabetes
MLP	0.7298 ± 0.0099	0.9840 ± 0.0014
VIME	0.7348 ± 0.0088	0.9861 ± 0.0014
Mix	**0.7378 ± 0.0077**	0.9905 ± 0.0013
Combined	0.7355 ± 0.0085	**0.9940 ± 0.0009**

4.3 t-SNE analysis

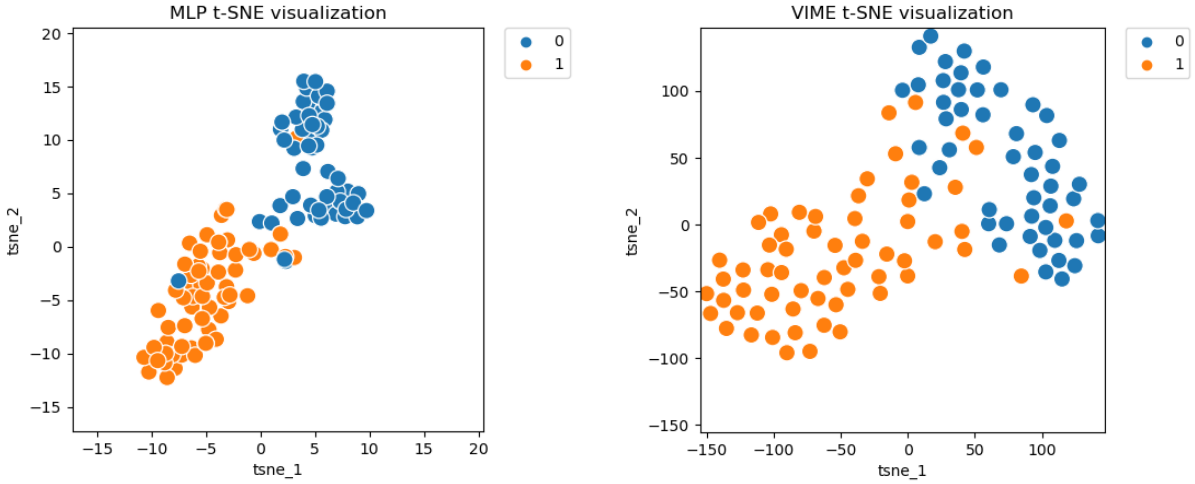
In this section, we visualize the latent representations of each model using t-SNE (t-Distributed Stochastic Neighbour Embedding) [20]. An overview of this method can be found in Appendix B. To obtain these results, t-SNE was implemented using *sklearn*'s library and applied to the latent representations of the MLP, VIME, Mix, and Combined Encoder for both the breast cancer and diabetes datasets. The visualized data points are representations of the validation data for seed 2127 with fine-tuning being done on 75% of the training data. Note that these are therefore representations that contributed to the downstream results (same hyperparameters).



(a) MLP (b) VIME

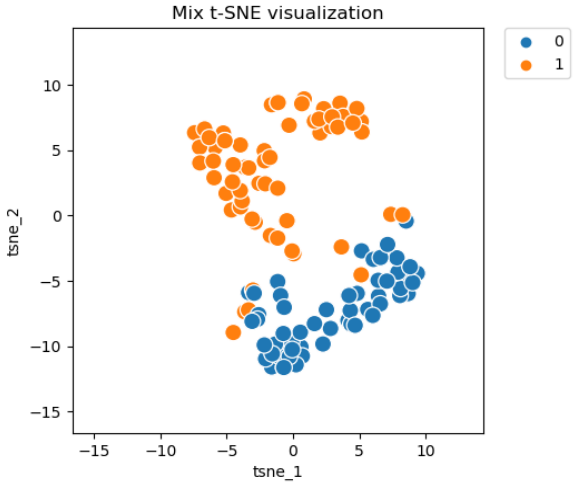
(c) Mix (d) Combined

Figure 6: t-SNE analyses of the breast cancer validation data for seed 2127 with 75% of the remaining training data used for fine-tuning.

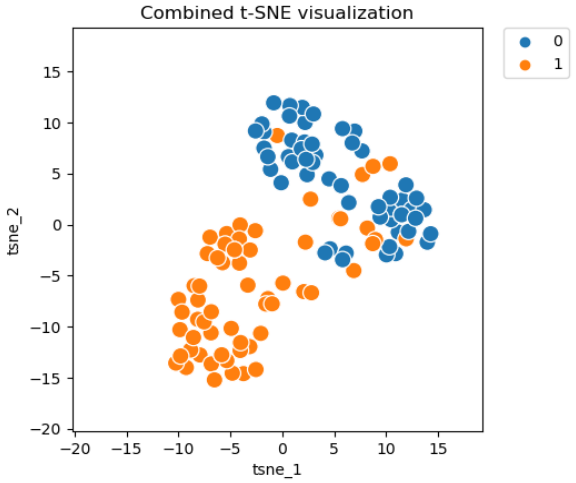


(a) MLP

(b) VIME



(c) Mix



(d) Combined

Figure 7: t-SNE analyses of the diabetes validation data for seed 2127 with 75% of the remaining training data used for fine-tuning.

As can be seen in Figure 6, the pre-trained models demonstrate a clearer attempt at separating the classes of breast cancer data compared with the MLP. Comparing Mix with VIME, Mix arguably finds a clearer separation. This also becomes evident in the Combined representation as it utilizes most of Mix's representation. Figure 7 shows that the high performance on the diabetes dataset corresponded to a clear separation of the representations for all models. The distinct difference between Mix and VIME is the respective dense and sparse separations. In this case, it can be seen that Combined Encoder optimized its representations by inheriting the dense clusters of Mix, but with the shape and divisions of VIME.

5 Discussion

The results gathered in this research show that our proposed model, Mix Encoder, constitutes a viable pre-training model by exhibiting the highest AUC performance across all sizes on the breast cancer dataset. For most training sizes, our Combined Encoder also outperformed VIME and the benchmark MLP. Still, given the large variation for all models, a larger set of seeds would be preferable for further analysis. This would come to the detriment of computation times.

A clearer result is shown on the diabetes dataset where the Combined Encoder exhibits the highest AUC performance across all sizes. And, for larger training sizes, Mix outperformed both VIME and the MLP and remained the highest performer of its size. Surprisingly, however, VIME and Mix performed the worst on both the 10% and 26.25% sizes.

These downstream results of VIME, Mix, and Combined depend on the pre-training process. For both approaches this process is stochastic. Due to this, large jumps during training can be seen in Figure 4. As a consequence, the fully trained representation space might have been subject to a difficult final pretext step and therefore had a poorer representation space than that of a previous epoch. Variance of this kind originates from the design of the pretext tasks and is a limitation of the model. Here, Mix shows a more stable training procedure than VIME.

Why does this work? As interpreting and understanding deep neural networks is notoriously difficult, in this study we only gather an intuitive view as to why Mix Encoder achieves good performance. In comparison to VIME, Mix does not attempt to find correlations by discretely looking at surrounding features, instead, it continuously compares all features of a patient to all features of another patient and has to reconstruct them entirely. For certain datasets, this could be a more efficient approach, especially if only a few features are correlated which might not be captured by VIME. Whether or not the datasets are suitable for representation learning therefore greatly affects the applicability of pre-trained models. If the dataset is simple, other methods, such as XGBoost, could provide better results. Given that the dataset is somewhat suitable, demonstrated using t-SNE, we have shown that Mix Encoder might provide better performance than state-of-the-art models such as VIME. Combined Encoder on the other hand is an attempt to minimize the shortcomings of both pre-training methods by supplying two views of the same object. The latent vector does, however, become doubled. It is also possible that the performance gained by doing this is limited by the inferior representation such that the combined representation simply confuses the model.

Analysis of proper hyperparameters for pre-training models is another important aspect that requires further study to draw more certain conclusions about its viability on scarce datasets. The hyperparameters used here are approximated to obtain good results but could be better given proper study. Yoon et al. [2] carried out a detailed hyperparameter analysis of pre-training VIME in the context of the MNIST dataset. This has been taken into consideration here.

On the topic of pre-training, this study is limited by information leaks potentially occurring during the pre-training stage as most of the dataset is used for this purpose. The impact of this is difficult to estimate but could lead to some of the overfitting issues. In the same vein, a dedicated test set is also important to avoid overfitting on validation data. To obtain a large range of different data sizes it was omitted in this study and we instead opted for a randomized approach.

Important to note is that the diabetes dataset was analyzed in a shorter amount of time and has less optimized hyperparameters.

6 Conclusion

In this thesis, we have analyzed the viability of pre-trained models on tabular data. By using VIME and introducing a novel pre-training scheme, Mix Encoder, we have shown that a signal can be found in tabular data by utilizing self-supervised learning. Our proposed model shows higher AUC performance on breast cancer predictions compared to state-of-the-art pre-training, namely VIME, and a benchmark MLP. Furthermore, by combining the representations of two different pre-trained models, a strong signal was possibly found when classifying early onsets of diabetes, as compared with an MLP and the respective pre-trained models of their own. However, further studies should be made.

Acknowledgments

I would like to thank Mattias Ohlsson for his great supervision and input throughout this entire project. I would also like to thank family and friends for their support during this time. A special thanks to Linus, for the endless curiosity you had.

A Appendix

This appendix contains all hyperparameters that were used in this thesis. Table 3 shows hyperparameters used in pre-training VIME and Mix while Tables 4, 5, 6, and 7 show hyperparameters used when training/fine-tuning the models for the downstream task. All models used the ReLU (Rectified Linear Unit) activation function and the Adam optimizer.

The seeds used for the 10 different runs were: 2127, 10291, 61691, 912811, 44444, 7562, 5678910, 192927, 58517, and 5607.

A.1 Mix parameter generation

The mix parameter λ is generated using a Beta distribution as $\lambda \sim Beta(\alpha, \beta)$. The values of α and β can be adjusted to shift the probability distribution towards 1 such that we are more likely to generate larger values. While doing so, we also set thresholds for which values λ can take. In general, $\lambda \sim$ *lower bound* $\leq Beta(\alpha, \beta) \leq$ *upper bound*. If the generated value is below the *lower bound*, we set $\lambda =$ *lower bound*. Likewise, if the generated value is above the *upper bound*, we set $\lambda = 1$. The shifting and thresholds are implemented in order to adjust the pretask difficulty.

A.2 Pre-training

Table 3: A table of the hyperparameters used to pre-train VIME and Mix Encoder for the breast cancer/diabetes dataset as seen in Figure 3. Note that if $\lambda = Beta(5, 1) \geq 0.9$ then $\lambda = 1$.

	VIME	Mix
Epochs	2500/7000	15000/50000
Batch size	400/200	400/200
Learning rate	0.0001	0.0001
α/θ	3.0	1.0
Training size	0.9 (90%)	0.9
p_m	0.2	None
λ	None	$0.7 \leq Beta(5, 1) \leq 0.9$

A.3 Downstream training - Breast cancer

Table 4: A table of the hyperparameters used for supervised training of all models on the breast cancer dataset. The number of epochs varies due to varying sizes, but this is handled in the code given a target number of steps.

	MLP	VIME	Mix	Combined
Steps	4000	3000	3000	3000
Batch size	200	200	200	200
Learning rate	0.0001	0.0001	0.0001	0.0001

Table 5: A table of the training sizes and the respective L2 for each downstream model. The L2 for VIME's and Mix's encoders was 0.01 for all sizes. The sizes are given as fractions of the entire dataset.

Training sizes:	0.25	0.375	0.5	0.625	0.75
MLP	0.06	0.058	0.056	0.054	0.04
VIME	0.038	0.028	0.028	0.026	0.02
Mix	0.06	0.059	0.058	0.057	0.055
Combined	0.05	0.049	0.048	0.047	0.045
Combined (encoder)	0.05	0.045	0.035	0.025	0.015

A.4 Downstream training - Diabetes

Table 6: A table of the hyperparameters used for supervised training of all models on the diabetes dataset. The number of epochs varies due to varying sizes, but this is handled in the code given a target number of steps.

	MLP	VIME	Mix	Combined
Steps	4000	4000	4000	4000
Batch size	200	200	200	200
Learning rate	0.0001	0.0001	0.0001	0.0001

Table 7: A table of the training sizes and the respective L2 for each downstream model. The L2 for VIME's and Mix's encoders was 0.01 and 0 for Combined's encoder, for all sizes. The sizes are given as fractions of the entire dataset.

Training sizes:	0.1	0.2625	0.425	0.5875	0.75
MLP	0.063	0.062	0.061	0.03	0
VIME	0.05	0.03	0.02	0.005	0.005
Mix	0.08	0.06	0.01	0.005	0.005
Combined	0.005	0.005	0.005	0.005	0

B Appendix

This appendix aims to give a short explanation of SNE and t-SNE.

Created by Geoffrey Hinton and Laurens van der Maaten (2008), t-SNE (t-Distributed Stochastic Neighbour Embedding) is a method of visualizing high-dimensional data on a two- or three-dimensional map [20]. The method originates from normal SNE and is adapted to be easier to optimize and produce better visualization.

The main idea of SNE is to convert the high-dimensional Euclidean distances between data points into conditional probabilities. These conditional probabilities can then be interpreted as similarities. Thus, $p_{j|i}$ is the conditional probability that x_i would pick x_j as its neighbor from a Gaussian distribution. In this way, it is very unlikely for two dissimilar points to be picked as neighbors. We can write the conditional probability as

$$p_{j|i} = \frac{exp(\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|x_i - x_k\|^2/2\sigma_i^2)},$$

where σ_i is the variance of our Gaussian, which can be estimated. A similar conditional probability can be computed for the low-dimensional counterparts of x_i and x_j. Denote these as y_i and y_j and the probability can be written as

$$q_{j|i} = \frac{exp(\|y_i - y_j\|^2)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)}.$$

The goal of SNE is to find a pair of data points such that the discrepancy between the conditional probabilities $p_{j|i}$ and $q_{j|i}$ is minimized. A good measure for this discrepancy is the Kullback-Leibler divergence which is used as a cost function,

$$\sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}.$$

Here, P_i is the conditional probability for all other high-dimensional data points given an x_i, and Q_i is the conditional probability for all other low-dimensional data points given an y_i. Normal SNE, which follows this approach, can be subject to a 'crowding problem'. To solve this issue, t-SNE changes the cost function by using a Student t-distribution instead of a Gaussian and modifying the SNE cost function such that it is symmetrized.

References

[1] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. URL: http://arxiv.org/abs/1710.09412.

[2] Jinsung Yoon et al. "VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11033–11043. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/7d97667a3e056acab9aaf653807b4a03-Paper.pdf.

[3] Veenu Rani et al. "Self-supervised Learning: A Succinct Review." In: *Archives of Computational Methods in Engineering: State of the Art Reviews* 30.4 (2023), pp. 2761–2775. ISSN: 1134-3060. URL: https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edssjs&AN=edssjs.9DAA2316&site=eds-live&scope=site.

[4] Xiao Liu et al. "Self-supervised Learning: Generative or Contrastive". In: *CoRR* abs/2006.08218 (2020). arXiv: 2006.08218. URL: https://arxiv.org/abs/2006.08218.

[5] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[6] Sercan Ömer Arik and Tomas Pfister. "TabNet: Attentive Interpretable Tabular Learning". In: *CoRR* abs/1908.07442 (2019). arXiv: 1908.07442. URL: http://arxiv.org/abs/1908.07442.

[7] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[8] Ehsan Hajiramezanali et al. "STab: Self-supervised Learning for Tabular Data". In: *NeurIPS 2022 First Table Representation Workshop*. 2022. URL: https://openreview.net/forum?id=EfR55bFcrcI.

[9] Yichen Zhang et al. *Mix-up Self-Supervised Learning for Contrast-agnostic Applications*. 2022. arXiv: 2204.00901 [cs.CV].

[10] Patrik Edén Mattias Ohlsson. "Introduction to Artificial Neural Networks and Deep Learning". In: *Computational Biology and Biological Physics - Department of Astronomy and Theoretical Physics - Lund University* (Fall 2022).

[11] Google. "Embeddings". In: (2022). URL: https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture.

[12] Li Jing et al. "Understanding Dimensional Collapse in Contrastive Self-supervised Learning". In: *CoRR* abs/2110.09348 (2021). arXiv: 2110.09348. URL: https://arxiv.org/abs/2110.09348.

[13] Dihge L et al. "Artificial neural network models to predict nodal status in clinically node-negative breast cancer". In: *Bmc Cancer, 19, Article 610* (2019). URL: https://doi.org/10.1186/s12885-019-5827-6.

[14] M. M. Faniqul Islam et al. "Likelihood Prediction of Diabetes at Early Stage Using Data Mining Techniques". In: *Computer Vision and Machine Intelligence in Medical Image Analysis*. Ed. by Mousumi Gupta et al. Singapore: Springer Singapore, 2020, pp. 113–125. ISBN: 978-981-13-8798-2. URL: https://doi.org/10.1007/978-981-13-8798-2_12.

[15] Wikipedia. "XGBoost". In: (2024). URL: https://en.wikipedia.org/wiki/XGBoost.

[16] Pratham Thakral. "Diabetes — EDA Prediction — Bangladesh". In: (2023). URL: https://www.kaggle.com/code/pthakral1998/diabetes-eda-prediction-bangladesh.

[17] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics*. 2010. URL: https://api.semanticscholar.org/CorpusID:5575601.

[18] J Behboodian and Akbar Asgharzadeh. "On the distribution of Z-scores". In: *Iranian Journal of Science Technology, Transaction A* 32 (Dec. 2008).

[19] Google. "Classification: ROC Curve and AUC". In: (2022). URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[20] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.