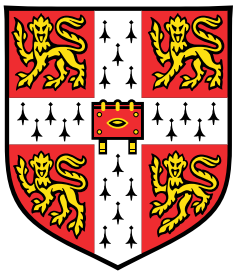
Contrastive Self-Supervised Learning for Tabular Data



Hugh Bishop

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Wolfson College

September 2021

I would like to dedicate this thesis to my parents, my brother Mark, and my girlfriend Jemima, who have all been extremely supportive throughout what has been a challenging year.

Declaration

I, Hugh Bishop of Wolfson College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for comparable purpose. This dissertation contains 12542 words.

The project is implemented in Python and PyTorch, with some code from the repository https://github.com/jsyoon0823/VIME being used for loading data and for benchmarking against their implementation in Chapter 3.

Hugh Bishop
September 2021

Acknowledgements

I would like to thank my supervisors, Professor Mihaela van der Schaar and Dr Fergus Imrie for introducing me to the project and for their help and guidance throughout which has been invaluable. I would also like to thank Dr Cheng Zhang for her advice and for regularly taking the time to meet with me to discuss the project. These meetings were extremely helpful.

Abstract

Much of the worlds data is stored in databases or spreadsheets and consists of columns of features which comprise of continuous, categorical or binary variables. A range of challenging and important machine learning tasks rely on data of this format such as applications in healthcare, business analytics and recommendation systems. However, large amounts of labelled data for a specific task can often be either expensive or impossible to obtain. Self-supervised learning is a technique in machine learning in which representations are learned from large amounts of unlabelled data and then used to assist other models trained on tasks for which there is only a small amount of labelled data. Recent research has shown that this approach can improve performance on tabular data when labelled data is scarce, and therefore finding better self-supervised learning methods for tabular data is an important area of research.

Contrastive learning is a framework for learning representations that has recently received a great deal of attention, particularly in computer vision, because of it's state-of-the-art performance in self-supervised learning. However, whilst it has been applied successfully to image data, the vast majority of current contrastive learning methods require the use of bespoke augmentations that are only applicable to images. Tabular data is a domain where data augmentations are not as widely used because there do not exist strong inductive biases about the latent information in the inputs for all datasets that would be equivalent to, for example, translation invariance for images. There has therefore been almost no research into contrastive learning for tabular data, in spite of the fact that contrastive learning does not demand the use of augmentations, and could potentially be used to achieve equivalent improvements in performance for tabular data if adapted correctly.

In this thesis, we explore the use of contrastive learning for tabular data and develop a novel approach called Masked Contrastive Learning (MCL) that does not require domain-specific augmentations and we show that it can be used to achieve state-of-the-art results for self-supervised learning on tabular data. We also extend our approach using a novel masking scheme we develop which allows the mask generator to learn to retain the structural informa-

tion in the input in a method we call Adaptively-Masked Contrastive Learning (AMCL).

We bencmark MCL against multiple other approaches to self-supervised learning for tabular data on three different datasets comprising of both regression and classification tasks and find that MCL outperforms all other methods on all of these datasets. Furthermore, we find that AMCL further improves the performance on two of the three datasets and has no impact on results for the third. This shows that it is possible to do contrastive learning without strong inductive biases about the structure of the inputs.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Whilst deep learning has seen great success when applied to modalities such as image data and natural language, it's application to tabular data has been relatively overlooked. Tabular data refers to any data in which instances are represented as vectors of ordinal, categorical, binary or scalar variables. This is the most general form of data as it does not assume any additional structure in the inputs. Other modalities, such as image data, can be viewed as a special case of tabular data. Given that there is such a large amount of tabular data in the world, and that many important applications of modern machine learning rely on tabular data, breakthroughs in deep learning in this domain could have far-reaching benefits.

Deep learning generally requires a lot of labelled data in order to outperform methods such as ensembles of decision trees (Chen and Guestrin, 2016) on tabular data, and often high-quality labelled data can be difficult or expensive to obtain. However, unlabelled data is generally much more readily available. An example of this is in the healthcare domain where a system may be tasked with diagnosing a patient with a given condition. The system may have access to a large amount of patient data, but not to specific diagnoses of the particular disease. An area of machine learning known as self-supervised learning focusses on learning representations from unlabelled data that can be used to improve performance when training other models on small amounts of labelled data, and advances in this area can be used to address the challenges presented by these scenarios.

Self-supervised learning has seen a great deal of research in the image and natural language domains, but these approaches generally rely on the development of bespoke architectures and algorithms that utilise inductive biases about the structure of the data in order to

train models suited to the required task. For example Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012) are designed to have outputs that are invariant to translations in the input, which encodes our belief about the structure of image data. These CNNs are then generally trained on unlabelled data by learning to perform tasks which are specific to images such as solving a jigsaw (Noroozi and Favaro, 2016). In the natural language domain, transformers (Vaswani et al., 2017) are used because they take into account long-term temporal dependencies, that we believe are necessary to understand language. These architectures work because every image dataset shares common structural similarities and likewise with natural language data. The challenge with designing self-supervised learning models for general tabular data, is that there are no such inductive biases or common structural dependencies that apply to all tabular datasets.

Contrastive learning is a new approach to self-supervised learning which has achieved state-of-the-art results in the image domain. The idea behind contrastive learning is to learn an encoding such that certain pairs, referred to as positive pairs, of images are close in the embedding space, and other pairs (negative pairs) of images are far apart. Positive pairs of images are generally created from the same image using image specific augmentations such as rotation, translation and scaling and therefore the same approach cannot be directly applied to tabular data. However, the general framework for contrastive learning does not require augmentation, and therefore adapting contrastive learning to the tabular domain could result in similarly impressive results.

The aim of this project is develop novel approaches for self-supervised learning on tabular data, in order to improve performance when unlabelled data is abundant. We focus on contrastive learning because of it's success in the image domain, and because it's application to tabular data represents a gap in the current literature. We develop a novel approach to contrastive learning called Masked Contrastive Learning (MCL), and show that it can be used to achieve state-of-the-art results for self-supervised learning in the tabular domain. We investigate different masking approaches and how the difficulty of the pretext task, meaning the task used for training a model on the unlabelled data, affects performance on the downstream task, i.e. the task that is ultimately of interest involving the labelled data. We then propose an extension to MCL called Adaptivel- Masked Contrastive Learning (AMCL) by designing a novel algorithm for learning to mask inputs in a way that retains information for a given dataset and show that this approach leads to improved results in most cases.

1.2 Project Outline

We begin by giving some general background and surveying the literature in Chapter 2, and provide an overview of different methods for contrastive learning categorised by their approach to generating positive and negative pairs. We also review works that address self-supervised learning for tabular data. In Chapter 3 we introduce a novel approach to contrastive learning which we call Masked Contrastive Learning, investigate different variations on our approach, and benchmark our results against other methods. In Chapter 4 we introduce a novel approach to adaptively masking inputs in a way that preserves the information in the input called Adaptively-Masked Contrastive Learning. We then show that this can be used to further improve performance on the benchmark datasets. Our method makes use of and extends recent research into backpropogation through discrete stochastic variables. Finally, we summarise our findings in Chapter 5, and highlight the limitations of the work as well as proposing directions for future research in this area.

The main contributions of this thesis are as follows:

1. A concise survey of the literature of contrastive learning and self-supervised learning in general, and a summary of the different approaches to generating positive pairs.

2. A novel method for contrastive learning called Masked Contrastive learning that is the first successful application of contrastive learning to tabular data that we are aware of, and that does not require strong inductive bias about the data, as other contrastive methods do.

3. A benchmark comparison of different approaches to self-supervised learning on 3 datasets, each with varying numbers of features, instances and comprising both classification and regression tasks. We show that MCL outperforms all previous methods.

4. A proposed extension to our method called Adaptively-Masked Contrastive Learning which learns to correlate the masking of certain inputs in a way that preserves the information in those inputs. This is done using a novel hierarchical application of the reparameterisation trick. We show that in most cases, this extension improves the performance of MCL.

Chapter 2

Background

In this chapter we will introduce the necessary background material and review the related work that has been done on self-supervised learning, with a particular focus on applications to tabular data, and on contrastive learning. We first look at some of the core methods for representation learning that do not make any assumptions about the structure of the data, and then look at some of the recent research into self-supervised learning for tabular data. We then outline the contrastive learning paradigm and review the literature on this topic. Finally, we summarise the literature, and evaluate which contrastive learning approaches could be applied to tabular data.

2.1 Self-supervised Representation Learning

The reason for deep learning's success over alternative methods is it's ability to learn successive representations of the data, which in principle encode progressively higher-level features of the input. In supervised learning these representations are generally trained in unison with the classification layer of the network. However, when the data is unlabelled, rich representations can still be learned.

Learning structure from unlabelled data is known in the field of machine learning as unsupervised learning. There is a range of tasks for which unsupervised learning can be useful, such as discovering human-interpretable structure in the data by, for example, grouping the datapoints into clusters (Saxena et al., 2017). Another way it can be used is to learn an encoding function which can be used for training machine learning models for a range of downstream tasks predominantly in cases where there is a lack of labelled data (Bengio et al., 2012). The usual approach to learning this encoding is to design a pretext task in which the

model is trained to optimise some objective function that is believed to be associated with useful representations. This is known as self-supervised representation learning and is an actively researched sub-field of unsupervised learning.

Formally, the aim of self-supervised representation learning is to learn an encoding function f, that maps a set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ of N unlabelled training examples to a set $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$ such that $f(\mathbf{x}_i) = \mathbf{z}_i$ captures the underlying semantic information of $\mathbf{x}_i$, and contains as little redundancy as possible. We assume that the values of each input $\mathbf{x}_i$ are i.i.d. and sampled from some distribution $p(\mathbf{x})$.

The quality of the learned encoding function f is usually assessed by training a linear model for a downstream task, such as classification or regression, on a small labelled dataset, after each input has been encoded. The performance of the linear model on the dataset can then be used as a proxy metric for the quality of the representations. Self-supervised learning has received a lot of attention in the machine learning community because labelled data can be scarce or expensive in many situations, but it is possible to train a model to learn a representation which can be as useful or in some cases more useful for the downstream task than one learned using labelled data (Grill et al., 2020).

In the natural language domain, self-supervision has been crucial in recent years to achieving state-of-the-art performance (Devlin et al., 2018, Radford and Narasimhan, 2018, Brown et al., 2020). This is largely thanks to the development of the transformer (Vaswani et al., 2017) which allowed for massive parallelization in the training of NLP models, meaning models can now be trained on huge amounts of raw text with no annotations. This is done by training the models to predict masked words given surrounding words. Once trained these large models can be fine-tuned for a multitude of different tasks, achieving the state-of-the-art in translation, summarization and sequence classification all utilising pre-trained self-supervised language models.

One of the earliest approaches to representation learning is the autoencoder. This is a neural network in which the architecture includes a bottleneck, i.e. at some point the representation at a certain layer must have a lower dimensionality than the input. The layers before the bottleneck are collectively referred to as the encoder and the layers after as the decoder. A typical autoencoder architecture is shown in Figure 2.1. The network is trained to minimize the reconstruction loss, which is simply the L2 norm between the input and output

of the network:

$$\mathscr{L}(\mathbf{x}, g(f(\mathbf{x}))) = \|\mathbf{x} - g(f(\mathbf{x}))\|^2 \tag{2.1}$$

where f is the encoder and g is the decoder. The bottleneck is what makes the task challenging, as without the presence of a bottleneck, the network could simply learn the identity function. In order for the decoder to be able to learn to reconstruct the input from the lower dimensional representation, the encoder must learn to map the input to a representation that preserves the information in the input. This representation can be learned on unlabelled data and then used for downstream tasks, as it may be easier to learn from this more succinct representation than from the original inputs.

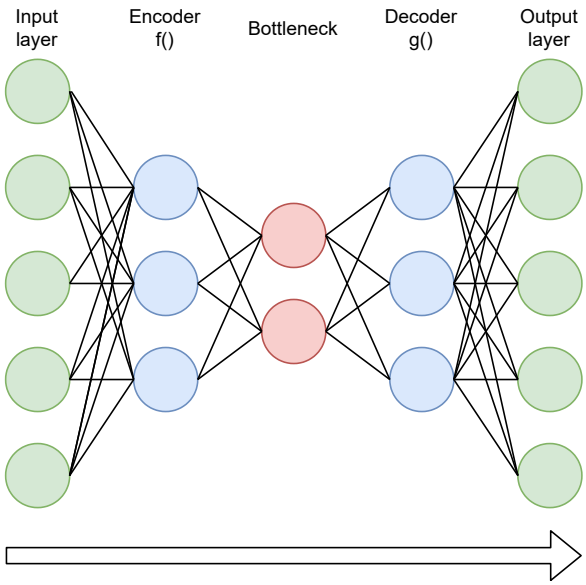


Fig. 2.1 Driagram showing the architecture of an autoencoder.

Another approach to training autoencoders is known as a Denoising AutoEncoder (DAE) which maps a corrupted version of the input back to it's original version using a loss function such as:

$$\mathscr{L}(\mathbf{x}, f(\tilde{\mathbf{x}})) = \|\mathbf{x} - f(\tilde{\mathbf{x}})\|^2 \tag{2.2}$$

where f is the DAE, and $\tilde{\mathbf{x}}$ is the corrupted input. This was inspired by the image domain where corrupting the input is normally done by setting some of the pixel values to 0. The challenge here is to predict the values of the corrupted inputs, and so dimensionality reduction is not necessary in order to learn a valuable encoding, hence the lack of clear distinction be-

tween encoder and decoder. The representation is instead taken from one of the intermediate layers of the network.

There has been comparatively little work on self-supervised learning for Tabular data, but the work that has been done tends to use approaches similar to that of a DAE, i.e. the pretext task consists of predicting either missing or corrupted input values. One of the most recent works is Value Imputation and Mask Estimation (VIME) (Yoon et al., 2020). In this paper, the authors mask each input $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ in a batch using a random mask $\mathbf{m} = [m_1, m_2, \ldots, m_d]$ which is generated such that each m_i is drawn independently from a Bernoulli distribution parametrerised by p_m which is a hyperparameter. The masked input $\tilde{\mathbf{x}}$ is generated by replacing each masked sample in the input with the corresponding element of $\hat{\mathbf{x}}$:

$$\tilde{\mathbf{x}} = \mathbf{m} \odot \bar{\mathbf{x}} + (1 - \mathbf{m}) \odot \mathbf{x} \tag{2.3}$$

where the elements of $\bar{\mathbf{x}}$ are samples of the corresponding features from other inputs in the training set and $\odot$ is element-wise multiplication.

They then use an encoder which maps each $\mathbf{x}$ to a corresponding $\mathbf{z}$ and which is optimised using two unsupervised pretext tasks, *mask vector estimation* and *feature vector estimation*, each of which has a corresponding network head on top of the encoder and it's own respective output. These both take as input an embedding of a corrupted row of data and each aims to predict *which* features have been corrupted, and *what* the true values of the corrupted features are respectively:

$$\text{Mask Vector Estimator: } f(\mathbf{z}) = \hat{\mathbf{m}} \tag{2.4}$$

$$\text{Feature Vector Estimator: } g(\mathbf{z}) = \hat{\mathbf{x}} \tag{2.5}$$

These are then jointly trained using a loss function which is the sum of two components:

$$\mathscr{L} = l_m(\mathbf{m}, \hat{\mathbf{m}}) + \alpha \cdot l_r(\mathbf{x}, \hat{\mathbf{x}}) \tag{2.6}$$

weighted by another hyperparameter α, where $l_m(\mathbf{m}, \hat{\mathbf{m}})$ is the sum of the binary cross-entropy losses for each dimension of the mask vector, and $l_r(\mathbf{x}, \hat{\mathbf{x}})$ is the reconstruction loss, which is the same as for that of a DAE, given by equation 2.2.

By jointly training the embedding function and the predictive models for the pretext tasks, the embedding function learns an embedding that is shown empirically to be useful for classification. This follows a common methodology in the image and natural language domains of obscuring part of the input and training a network to predict the missing data.

Another work that can be applied to self-supervised learning for tabular data is TabNet (Arik and Pfister, 2019), in which the authors use a specialised encoder, which uses a sequential attention mechanism to decide which features to reason from at each step. This can be trained in a self-supervised fashion using missing value imputation in a similar way to VIME. Whilst their encoder architecture addresses some of the issues of applying deep learning to tabular data, their scheme for self-supervision is not novel, and the encoder architecture could be used in conjunction with other methods.

In the image domain, the most successful recent approaches to self-supervised learning have been based around the framework of contrastive learning. Contrastive learning is closely linked to the unsupervised method for learning generative models known as Noise-Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010), which was the inspiration for some of the early contrastive learning works.

2.2 Noise-Contrastive Estimation

NCE (Gutmann and Hyvärinen, 2010) follows the principle of estimating the parameters of a statistical model by training the model to discriminate between the data $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ and some artificially generated noise $Y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$. Given a statistical model which is specified by a probability density function $p_m(\cdot;\theta)$, NCE gives an estimate θ_{NCE} for the optimal value $\hat{\theta}$ by optimising the following objective function with respect to θ:

$$\theta_{\text{NCE}} = \arg\max_{\theta} \left\{ \frac{1}{2T} \sum_{t} \ln[h(\mathbf{x}_t;\theta)] + \ln[1 - h(\mathbf{y}_t;\theta)] \right\} \tag{2.7}$$

$$h(\mathbf{u};\theta) = \frac{1}{1 + \exp\left(\ln p_m(\mathbf{u};\theta) - \ln p_n(\mathbf{u})\right)} \tag{2.8}$$

where the noise samples $\mathbf{y}_i$ are assumed to be drawn independently from $p_n(\cdot)$. The authors prove several desirable properties of their approach including that it is a consistent estimate of the parameters, meaning that $\lim_{T\to\infty} \theta_{NCE} = \hat{\theta}$.

This approach is very similar to a method used for training the popular word embedding framework Word2Vec (Mikolov et al., 2013) called negative sampling. The model was originally trained by predicting the next word given the current word, which involved generating a probability for each word in the whole vocabulary by applying a softmax function whose denominator required the calculation of the output for each word. However, negative sampling made this more efficient by only sampling a few negative examples and having the network predict which among these was the correct word, which improves the training time.

2.3 Contrastive Learning

Building on top of the ideas of NCE, recent works in the image domain have shown that learning an encoder which maps embeddings of different images to points far apart in latent space, but different augmentations of the same image to nearby points can achieve state-of-the-art results in self-supervised learning for image data. This is known as contrastive learning (van den Oord et al., 2018, Chen et al., 2020, He et al., 2019, Wu et al., 2018, Li et al., 2020). The crucial difference between contrastive learning and other approaches is that the objective function for a given input is only defined with respect to other inputs, as opposed to there existing a per-input label or correct output.

Most contrastive learning approaches learn an embedding function in which 'positive pairs' are relatively close in the embedding space, but 'negative pairs' are relatively distant. In the context of images, which is the domain in which most of the work on contrastive learning has been done, a positive pair might correspond two different augmentations, or *views*, of an image, where the original image is referred to as the *anchor*. Negative pairs consist of views of different anchors. This is shown in Figure 2.2. The learning is done using a loss function which directly rewards close proximity between positive pairs and penalises close proximity between negative pairs.

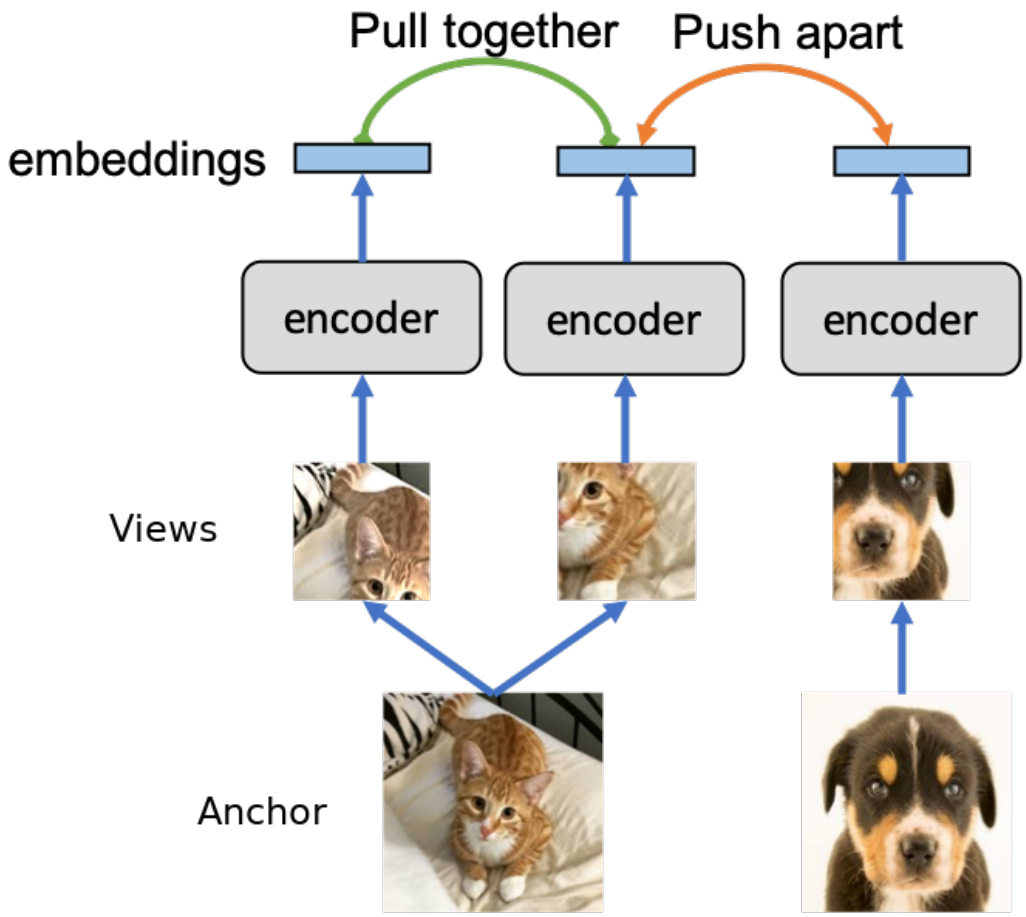


Fig. 2.2 Diagram showing the outline of contrastive learning. The positive pairs consist of different augmentations of the same image. The embeddings of different views of the same image are encouraged to be close together, whilst also being far apart from views of other images. Taken from (Li, 2020).

Most contrastive learning frameworks apply the loss function in a normalised space. This means that the task of optimising the contrastive loss can be thought of as finding a representation in which the embeddings of each image are both invariant to the method by which positive pairs are generated (usually augmentation) and uniformly scattered across a unit hyper-sphere.

The general setup for contrastive learning is to learn an embedding function $f : X \to Z$ where $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is the training data and $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T\}$ are the encodings, such that $\|\mathbf{z}_i\| = 1 \quad \forall i$. This encoding should maximize an objective function that specifies that positive pairs be close and negative pairs be far apart in latent space. In practice this has to be done in batches, and so for each batch a positive pair $(\tilde{\mathbf{x}}_\mathbf{i}, \tilde{\mathbf{x}}_\mathbf{j})$ and k instances which are

negative with respect to this pair $\{\tilde{\mathbf{x}}_\mathbf{1}, \tilde{\mathbf{x}}_\mathbf{2}, \ldots, \tilde{\mathbf{x}}_\mathbf{k}\}$ are generated from the training data and contrasted.

2.3.1 Contrastive Predictive Coding

Contrastive Predictive Coding (CPC) (van den Oord et al., 2018) is considered the first application of contrastive learning. The authors use a method in which deep learning models are trained via contrastive learning to map inputs to representations that are useful for various downstream tasks for various time-series data modalities. They use a combination of an encoder and an autoregressive model to generate context vectors $\mathbf{c}_t$ from all representations $\mathbf{z}_1, \ldots, \mathbf{z}_t$ of past inputs $\mathbf{x}_1, \ldots, \mathbf{x}_t$ and then use a specialised loss function InfoNCE, shown in equation 2.9 to enforce that the mutual information between future representations $\mathbf{z}_{t+k}$ and $\mathbf{c}_t$ is high compared with the mutual information between $\mathbf{c}_t$ and randomly sampled negative values. Either the representations $\mathbf{z}_t$ or $\mathbf{c}_t$ can then be used for downstream tasks.

$$\mathscr{L} = -\mathbb{E}_X \left[\log \frac{f_k(\mathbf{x}_{t+k}, \mathbf{c}_t)}{\sum_{\mathbf{x}_j \in X} f_k(\mathbf{x}_j, \mathbf{c}_t)} \right] \tag{2.9}$$

In equation 2.9, the function $f_k(\mathbf{x}_{t+k}, \mathbf{c}_t)$ is a function modelling the density ratio $\frac{p(\mathbf{x}_{t+k}|\mathbf{c}_t)}{p(\mathbf{x}_{t+x})}$, and in the paper the authors use $f_k(\mathbf{x}_{t+k}, \mathbf{c}_t) = \exp(\mathbf{z}_{t+k}^T W_k \mathbf{c}_t)$ where W_k defines a linear transformation specific to timestep k. Whilst the formulation implies temporal dependencies, this approach was applied to independent image data by using the top rows of the encoded image tensor as the representations $\mathbf{z}_1, \ldots, \mathbf{z}_t$ and the bottom rows as $\mathbf{z}_{t+1}, \ldots, \mathbf{z}_{t+k}$.

The success of CPC led to an increased level of research in contrastive learning. Most of these methods use an instantiation of InfoNCE loss given by:

$$\mathscr{L} = -\log \sum_{i=1}^{N} \frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_i' / \tau)}{\sum_{k=1}^{K} \exp(\mathbf{z}_i \cdot \mathbf{z}_k / \tau)} \tag{2.10}$$

where $(\mathbf{z}_i, \mathbf{z}_j)$ are encodings of the positive pair $(\mathbf{x_i}, \mathbf{x_j})$, and $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z_K}\}$ are the encodings of different inputs resulting in negative pairs. τ is a hyper-parameter referred to as the temperature. This can be viewed as an instantiation of the InfoNCE loss where the context vector $\mathbf{c}_t$ in this case is given by the $\mathbf{z}_i$, and $\mathbf{x}_{t+k}$ is given by $\mathbf{z}_j$. The choice of the function f_k (which can just be written f due to the lack of temporal dependence) is given by $f(\mathbf{x}_t, \mathbf{x}_t') = \exp(\mathbf{z}_t \cdot \mathbf{z}_t' / \tau)$.

This was introduced in the paper SimCLR (Chen et al., 2020) where the authors call it NT-Xent. However, most papers simply refer to this as InfoNCE without reference to the fact that it is an instantiation of the original InfoNCE loss function from CPC. In the rest of this thesis we will use InfoNCE to refer to this particular variant.

Whilst CPC uses temporal or spatial proximity to generate positive pairs, there is a range of different methods for generating positive pairs, and how the positive pairs are generated is generally the defining characteristic of a contrastive learning approach. We will now look at and compare the different approaches seen in the literature.

2.3.2 Methods for generating positive pairs

Augmentation

Most contrastive learning methods in the image domain rely on data augmentation to create positive pairs, with a positive pair consisting of two augmented versions of the same image. In SimCLR (Chen et al., 2020), a batch consists of N images, each augmented in two different ways to give $2N$ views. This augmentation is done using a combination of random cropping, random color distortion and random Gaussian blur. The batch loss is given by the sum of the InfoNCE loss function calculated with each pair of images as the positive pair. Using this approach as opposed to sampling new negative pairs for each positive pair can save dramatically on computational expense. One of their key findings was that large batch sizes, as high as $N = 8192$, were very important for obtaining good results for contrastive learning with images, which is likely due to the large number of negative examples in each batch. This is an example of how finding the right difficulty for the pretext task is crucial for learning a valuable embedding function. They also found that training a non-linear transformation $g(\cdot)$ to map the representations output by the encoder $f(\cdot)$ used for the downstream task to the outputs used for calculating the contrastive loss also improved performance, which is shown in Figure 2.3.

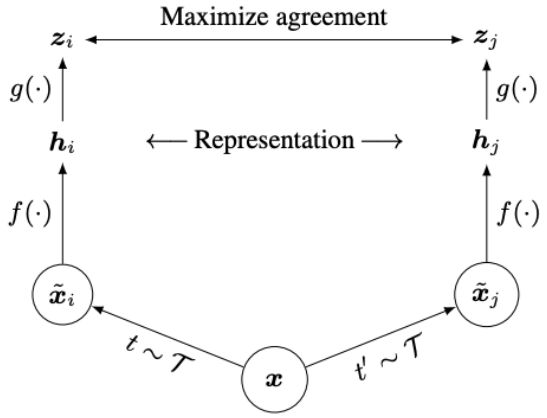


Fig. 2.3 Figure showing the SimCLR approach taken from the original paper. x is the original image, t and t' are the sampled transformations, $\tilde{x}_i$ and $\tilde{x}_j$ are the two views of x, $f(\cdot)$ is the encoder, $g(\cdot)$ is the projection head and z_i and z_j are the encodings of $\tilde{x}_i$ and $\tilde{x}_j$ on which the contrastive loss is calculated.

Computing gradient updates with each instance in a batch as the anchor is one approach to avoid computing representations for a lot of negatives for each positive. Another approach (Wu et al., 2018) is to store all of the encodings from each epoch in a memory bank and sample these when negatives are needed. This has the advantage that the number of negatives and the batch size can be decoupled completely and tuned separately. Momentum Contrast (MoCo) improves on this approach by using a dictionary which is stored as a queue of encoded samples. The elements of this dictionary are used as the negatives for each batch, and for each batch the encodings computed from the positive pairs are enqueued and the oldest encodings are dequeued. This is superior to a memory bank because the encodings are calculated earlier in the same epoch as opposed to the previous epoch leading to faster convergence. The augmentation methods used were very similar to SimCLR, namely random crop, random colour jitter and random horizontal flip.

One of the major drawbacks of augmentation-based approaches for generating positive pairs is that the augmentation methods and magnitudes have to be carefully selected and usually tuned for the downstream task. This is becasue each augmentation's suitability varies greatly depending on the dataset and downstream task, for example for a task in which the colour of an image is not important a model might benefit from invariance to colour and therefore benefit from an encoding trained using colour distortion to create positive pairs. However, this would presumably be detrimental to performance on a downstream task in which discerning between different colours can be very important. Tuning these augmentations for each specific downstream task can be costly and tedious. All of these augmentations

also only apply to images and therefore these approaches would not work for other modalities.

Labelled data

Contrastive learning has also been used in the supervised setting (Khosla et al., 2020). In this paper the authors restructure the contrastive learning framework so that positive pairs consist of the anchor and either an augmentation of the same instance, or a different instance of the same class, and negative pairs consist of images of different classes. They also introduce the novel approach of having many positives in the same batch as opposed to one positive and many negatives, and derive a suitable loss function, termed SupCon loss, for dealing with this generalisation. They show that learning representations using SupCon loss outperforms cross-entropy loss on 3 major image classifcation benchmark datasets. They also show empirically that using SupCon loss when training leads to less variance in results after small changes to hyper-parameters from their optimum values, meaning this approach is less sensitive to having finely-tuned hyper-parameters.

Temporal Information

Contrastive learning has also been applied in the speech domain, with COLA (COntrastive Learning for Audio) (Saeed et al., 2020) using a form of contrastive learning based on positive pairs taken from the same audio recording, and negative pairs taken from different audio recordings. They use this to learn a representation from a large audio database, and then use a linear classifier on top of this representation for 9 different classification tasks, acheiving state-of-the-art for most, in some cases by a large margin.

Clustering

Another way in which positive pairs have been generated is using clustering. In the paper Prototypical Contrastive Learning (PCL) (Li et al., 2020), the authors iteratively cluster the embeddings in the latent space and minimize a novel contrastive loss they call ProtoNCE. This loss function is a variant of InfoNCE which enforces that embeddings are close to their respective cluster centers and distant from other cluster centers. This approach has a lot in common with DeepCluster (Caron et al., 2018), with the key difference being the use of contrastive loss as opposed to treating the cluster assignments as pseudo-labels and optimising a classification objective. In practice they found that contrasting both at the cluster

level and at the instance level (similar to SimCLR) gave the best results.

Another approach which utilises clustering methods in conjunction with contrastive learning is Swapping Assignments between multiple Views of the same image (SwAV) (Caron et al., 2020). This aims to reduce the amount of computation needed for contrastive learning by removing the need for pairwise comparisons. It differs quite significantly from standard contrastive learning in that instead of enforcing proximity between positive pairs and distance between negative pairs, it enforces that different views of the same image have similar *codes*, where the code of an embedding is a set of distances from a set of prototypes and can be thought of as a cluster assignment. The prototypes themselves are jointly learned with the parameters of the encoder. This approach is outlined in Figure 2.4.

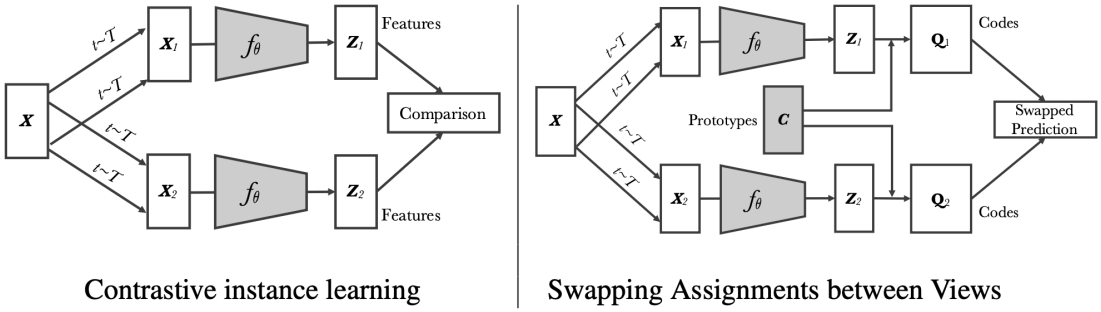


Fig. 2.4 Diagram showing the difference between SwAV and traditional contrastive learning.

2.3.3 BYOL

Bootstrap your own latent (BYOL) (Grill et al., 2020) is an approach that differs from contrastive learning in the sense that it doesn't make use of negative pairs. It does, however, draw inspiration from contrastive learning in that it is trying to enforce consistency in the outputs of views of the same image. They use two networks in a bootstrapping fashion, referred to as the online network and the target network. The target networks weights are updated using a slow-moving average of the online network. At training time each network is fed a different augmented view of the same image, and the online network predicts the encoding of the target network, with the loss function being simply the L2 difference between the two.

This approach is very interesting as negative examples were thought to be needed for diversity in outputs. If there are no negatives, it might seem as though it would be possible

that the network would just learn to output a constant, regardless of the input, and then trivially all views of the same image would have the same output and the loss would be minimized. The authors claim that the bootstrapping is the reason that the network does not learn these collapsed representations, although other researchers have found that the real reason might be that batch normalisation implicitly causes a form of contrastive learning and that without batch normalisation the network performs no better than random (Fetterman and Albrecht, 2020). Whether or not BYOL can perform well without batch normalisation is still an active area of research.

2.3.4 Application to Tabular Data

As far as we know there has been only one concurrent work which looks at applying contrastive learning to tabular data. SAINT (Somepalli et al., 2021) focusses on the use of contrastive learning in the fully supervised setting, meaning that they retrain the encoder on all of the labelled data using cross-entropy loss after first optimising an unsupervised contrastive loss. They also perform some experiments in the semi-supervised setting which means that they have a large amount of unlabelled data, but still fine-tune the encoding learnt from this unlabelled data on a small amount of labelled data as opposed to keeping the encoding fixed. Whilst they use contrastive learning, it is only a small part of the contribution of this paper, and they even report that without the pretraining, they see little or no reduction in the performance in the supervised setting and only very marginal losses in the semi-supervised setting.

SAINT performs contrastive learning by creating positive pairs using a mixture of two augmentation methods. The first is CutMix (Yun et al., 2019) which in the context of tabular data is similar to the masking procedure in VIME and works by randomly replacing features of an input vector with features from a different input vector. This is still slightly different to VIME in that the features all come from the same input vector in CutMix. The second is called mixup (Zhang et al., 2017), which operates in the embedding space, and creates a new instance from the original instance by moving in latent space a small amount towards another instance. The augmentation procedure is outlined for a point $\mathbf{x}_i$ as follows:

$$\textbf{CutMix: } \mathbf{x}'_i = \mathbf{x}_i \odot \mathbf{m} + \mathbf{x}_a \odot (\mathbf{1} - \mathbf{m}) \tag{2.11}$$

$$\textbf{mixup: } \mathbf{z}'_i = \alpha \cdot E(\mathbf{x}'_i) + (1 - \alpha) \cdot E(\mathbf{x}_b) \tag{2.12}$$

where $\mathbf{x}_a$ and $\mathbf{x}_b$ are samples from the current batch, E is the encoder (they also use a projection head similar to SimCLR so this is not where the contrastive loss is applied), $\mathbf{m}$ is a binary mask with each element drawn from a Bernoulli distribution and α is a hyperparameter. One problem with this augmentation strategy is that it is not clear that it will preserve the semantic information contained in the input. For example, it could be the case that one particular feature contains a large portion of the information in the input and is very indicative of the class of the input for a downstream classification task. If this feature in a given vector were then to be swapped with the same feature from a different vector via CutMix, but the two embeddings were still enforced to be close, this could clearly be detrimental. Mixup is also a curious choice of augmentation strategy because for it to make any sense, the value of α must be small otherwise the input will more closely represent a different input in the batch than itself. However, in this case, the pretext task consists of enforcing that two points that are close in an embedding space also be close in some projection space, which may make the problem trivial, or at least no more difficult than simple dimensionality reduction methods such as autoencoders.

2.4 Summary

This literature review shows that most of the work done on self-supervised learning for tabular data focusses on reconstruction based approaches, and there has been relatively little work done on contrastive learning for tabular data. The only work (Somepalli et al., 2021) that we are aware of studied it as a small part of a larger framework. We suggest that their implementation could distort the instances of a positive pair in a destructive way and this could be why the pretraining does not seem to make a significant contribution to their results in either the supervised or semi-supervised case. The methods for contrastive learning that have been applied to other domains such as images, natural language and speech are mostly modality dependent and cannot be applied to tabular data. Table 2.1 shows a summary of the various contrastive methods surveyed and shows which are suitable for application to tabular data.

Approach	Loss function	Positive pair selection	Domain specific augmentation	Suitable for tabular data
CPC (van den Oord et al., 2018)	InfoNCE	Temporal proximity	Not required	✗
SimCLR (Chen et al., 2020)	InfoNCE	Augmentation	✓	✗
MoCo (He et al., 2019)	InfoNCE	Augmentation	✓	✗
SwAV (Caron et al., 2020)	Adapted InfoNCE	Augmentation	✓	✗
BYOL (Grill et al., 2020)	L2	Augmentation	✓	✗
PCL (Li et al., 2020)	InfoNCE + ProtoNCE	Same cluster + Augmentation	Required for InfoNCE component	Without InfoNCE loss
COLA (Saeed et al., 2020)	InfoNCE	Same recording	✗	✗

Table 2.1 Review of different contrastive learning methods and their suitability for tabular data.

Many of the works that do investigate self-supervised learning for tabular data make use of bespoke encoder architectures either as their main contribution or as part of it. This means that benchmarks against other self-supervised learning approaches are obscured by different encoder architectures as opposed to different self-supervised methods. This is in contrast to the image domain where different approaches are generally compared using the same encoder.

In the next chapter, we will present our novel approach to contrastive learning for tabular data which we refer to as Masked Contrastive Learning. We focus on implementing an appropriate method for generating positive pairs for tabular data, and show that our approach is superior to other naive methods, as well as to value imputation based self-supervised learning methods, for 3 different benchmark datasets.

Chapter 3

Masked Contrastive Learning

3.1 Motivation

The main issue to address when applying contrastive learning to tabular data is how to create positive pairs such that enforcing members of this pair be close in the embedding space would lead to an embedding that is useful for downstream tasks. A naive approach might be to create a positive pair by simply augmenting each input with a small amount of Gaussian noise. However, this might fail for categorical variables, as it is not obvious how to add noise in this case, as changing the category of a variable might seriously alter the semantic information in the input. It would also be the case that this pair would be close in input space, so enforcing that they are close in the embedding space might be trivial and not lead to a useful embedding, or not have any advantages over a simple autoencoder.

Most of the methods previously discussed could not be applied to tabular data, as they rely on creating positive pairs using augmentations which are specific to image data, and for which there are no equivalents for general tabular data. Whilst Contrastive Predictive Coding does not rely on augmentations to create positive pairs, it does rely on temporal (or in the case of images, spatial) proximity in order to create positive pairs. Prototypical Contrastive Learning could in theory be trained by using only the ProtoNCE loss function and without any data augmentations as it relies on clustering for it's positive pair generation. However, in an ablation study they show that the results obtained this way were relatively much poorer. Moreover, in the ablation study they still use image-specific augmentations to distort the inputs before applying the ProtoNCE loss, and so results without any augmentation at all would presumably be poorer still.

In this chapter we propose a novel method for performing contrastive learning on general tabular data called Masked Contrastive Learning (MCL). We begin by presenting and motivating MCL, and introducing the three datasets which we use throughout the rest of the report. Then we present the experiments we conducted, starting with benchmarking the performance of MCL against value imputation based self-supervised approaches, and other possible approaches to contrastive learning for tabular data, on the three datasets. We then compare variations of MCL, and finally discuss our results and insights learned from our experiments.

3.2 Method

One approach to contrastive learning seen in the literature is to create pairs from different parts of the same input, as opposed to using augmentation (van den Oord et al., 2018, Saeed et al., 2020). This generally requires temporal information, or an encoder architecture that can be used on inputs of varying sizes, such as a convolutional neural netork. We use a similar approach, but one which can be applied to general tabular data. This consists of obscuring different features using a mask which replaces input values with 0, which is a method commonly used to handle missing data in the tabular setting. We then use InfoNCE as the loss function, which enforces that the two masked instances of the same input be close in the embedding space, but that masked versions of different inputs are far apart.

The approach we take for masking involves masking each input feature with some probability p independently for each instance in the positive pair:

$$\mathbf{m} = [m_1, m_2, \ldots, m_d] \in \{0,1\}^d \tag{3.1}$$

$$m_i \backsim B(p) \ \ \forall m_i \tag{3.2}$$

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} \tag{3.3}$$

where $\mathbf{m}$ is the mask, $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is the vector of input features, $\tilde{\mathbf{x}}$ is the masked feature vector, and B is the Bernoulli distribution parameterised by p. This masking method would be repeated for every feature vector independently.

In terms of calculating the loss for a batch we followed the SimCLR approach. This involves generating N pairs, and calculating the loss for a batch as the sum of the InfoNCE loss function calculated with each pair as the positive pair. This helps avoid calculating the

encodings several times for each input in order to sample negative pairs for each positive pair. It does however mean that the batch size is tied to the number of negatives used. We also found that using a projection head with a normalised output that gets discarded once the representation had been learned also gave a noticeable improvement in results. The point in the network at which the encoder becomes the projection head is arbitrary and generally tuned on the validation data. Generally the best results are achieved when the projection head has a relatively low capacity compared to the encoder so that not too much of the structure is learned by the projection head only to be discarded. We call our approach Masked Contrastive Learning (MCL) and the architecture is shown in Figure 3.1.

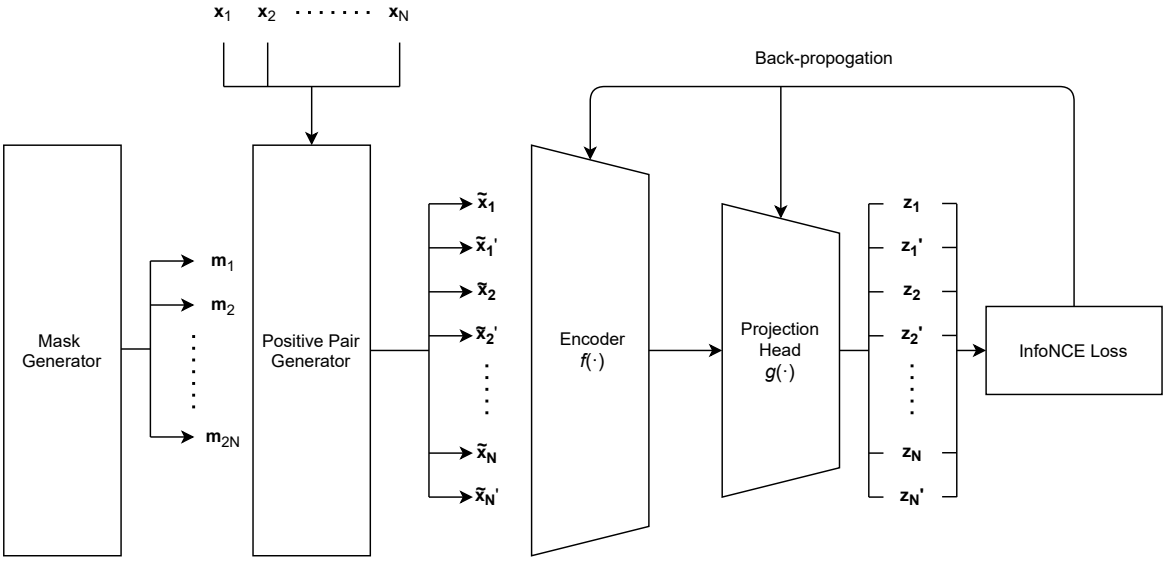


Fig. 3.1 Diagram of the architecture of MCL. The masks from the mask generator are used to create the positive pairs from the input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. These N pairs make up a batch of size $2N$ which is passed through the encoder and the projection head to give the encodings $\mathbf{z}_1, \mathbf{z}'_1, \mathbf{z}_2, \mathbf{z}'_2 \ldots, \mathbf{z}_N, \mathbf{z}'_N$. Then the loss is calculated using the InfoNCE loss function, and used to update the weights of the encoder $f(\cdot)$ and projection head $g(\cdot)$. For the downstream task, only the $f(\cdot)$ is kept.

The pseudocode for MCL is shown in Algorithm 1. The training and validation splits have been omitted for simplicity. Here we have shown InfoNCE as a function which takes as input two matrices, which correspond to embeddings of two copies of the batch matrix, each

with a different mask applied.

Algorithm 1: Masked Contrastive Learning

initialize $f(\cdot)$ and $g(\cdot)$;
for *epoch in range(epoch_num)* **do**
  for *X in Dataloader* **do**
    mask = Bernoulli(p=0.5, shape=(X.shape[0] * 2, X.shape[1]));
    X, X' = $X \odot$mask[:X.shape[0]], $X \odot$mask[X.shape[0]:];
    Z, Z' = $g(f(X))$, $g(f(X'))$;
    loss = InfoNCE(Z, Z');
    loss.backward();
  end
end
return f;

3.3 Datasets

For this project, we chose to evaluate different approaches by comparing performance on datasets covering a range of different sizes, downstream tasks and variable types. We focus on the case where there are 500 labelled examples, and the rest unlabelled. The labelled dataset is then split into training and test in a ratio of 4:1. We evaluate the quality of the representations learned on the unlabelled data by training linear models on the encoded training subset of the labelled data and testing on the encoded test subset of labelled data. For classification tasks this was a logistic regression model, and for regression tasks we used linear regression. In order to deal with categorical features, we used a one-hot encoding.

We look at three popular benchmark datasets, Tabular MNIST, UCI Income,and UCI Wine Quality:

- **Tabular MNIST** is a dataset derived from the famous image classification dataset MNIST, which consists of 60,000 images of handwritten digits and the goal is to classify each image as one of the 10 possible digits. It is transformed into tabular data by treating the data as a flattened vector of continuous variables as opposed to an image matrix. The images are 28×28 pixels, which when flattened gives a 784 dimensional feature vector. Whilst this is slightly contrived and does not represent a real world tabular dataset, it is often used for evaluating deep learning methods for tabular data (Arik and Pfister, 2019, Yoon et al., 2020, Somepalli et al., 2021) because

it offers some variety compared to traditional tabular datasets and can be used to see how models deal with a large number of inputs.

- **UCI Income** is a classification dataset where the task is to predict whether an individual has an annual salary above or below $50,000 based on a set of 14 different features such as age sex and race, which are a mixture of categorical and numerical variables. It has a total of 48,842 instances.

- **UCI Wine Quality** is the regression dataset we investigated, which is a relatively small regression dataset with a total of only 4,899 instances, which after the split gives 4,409 unlabelled instances and 490 labelled. The task is to predict the quality of the wine given a set of continuous valued features measuring levels of chemicals in the wine. We normalised the output ratings between 0 and 1 and used Mean Absolute Error (MAE) for evaluation.

3.4 Experiments

3.4.1 Benchmarks

In order to evaluate the effectiveness of MCL, we compare its performance on the benchmark datasets against other approaches. Specifically, we compare it to several supervised learning approaches that are trained only on the labelled portion of the data in order to see what the improvements of self-supervised learning approaches are in general, and how using large amounts of additional unlabelled data can improve performance. We also evaluate against two value-imputation based approaches to self-supervised learning for tabular data, namely VIME and a DAE. We chose these two because they represent methods of self-supervision, in contrast to, for instance, TabNet in which the novelty is in the encoder architecture, but the self-supervision scheme is very similar to that of a DAE.

Finally, we compare the results to two other naive approaches to contrastive learning for tabular data. The first is to generate positive pairs using the augmentation method of simply

adding Gaussian noise to each input:

$$\varepsilon \backsim \mathcal{N}(\mathbf{0},\mathbf{1}) \tag{3.4}$$

$$\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon \tag{3.5}$$

which we refer to as Noise Contrastive Learning (Noise CL). The second alternative approach was to use CutMix (Yun et al., 2019) to augment the inputs:

$$\mathbf{m} = [m_1, m_2, \ldots, m_d] \in \{0,1\}^d \tag{3.6}$$

$$m_i \backsim B(p) \;\; \forall m_i \tag{3.7}$$

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} + \bar{\mathbf{x}} \odot (\mathbf{1} - \mathbf{m}) \tag{3.8}$$

where $\bar{\mathbf{x}}$ is a different input from the batch. We refer to this approach as CutMix CL. We decided to investigate this approach because of it's use for contrastive learning on tabular data in SAINT, and because CutMix is one of the only data augmentation schemes that exist for tabular data.

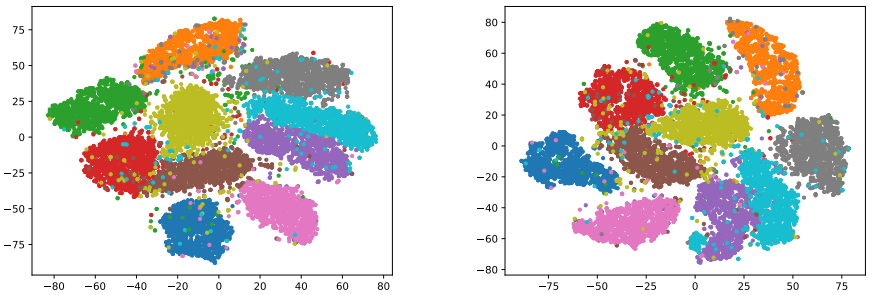
For each entry the results were averaged over 5 runs initialised with different random seeds. The results of these experiments are shown in Table 3.1.

Type	Model	MNIST % accuracy	UCI Income % accuracy	UCI Wine Quality MAE
Supervised	Logistic regression	0.840 ± 0	0.870 ± 0	0.0836 ± 0
	XGBoost	0.780 ± 0	0.840 ± 0	0.0888 ± 0
	MLP	0.834 ± 0.0102	0.866 ± 0.0162	0.0845 ± 0.0016
Self Supervised	DAE	0.864 ± 0.0102	0.876 ± 0	0.0809 ± 0.0021
	VIME	0.860 ± 0.0141	0.872 ± 0.0075	0.0816 ± 0.0018
	Noise CL	0.854 ± 0.0012	0.858 ± 0.0098	0.0817 ± 0.0021
	CutMix CL	0.860 ± 0.0090	0.856 ± 0.0049	0.0811 ± 0.0012
	MCL (ours)	$\mathbf{0.896 \pm 0.0049}$	$\mathbf{0.884 \pm 0.0102}$	$\mathbf{0.0782 \pm 0.0.0024}$

Table 3.1 Performance of different models on MNIST, UCI Bank and UCI Income datasets. The size of the labelled dataset is 500 instances split into 400 training and 100 test. The Supervised models were trained only on the 400 labelled instances, and the self-supervised models were trained on the rest of the dataset which was treated as unlabelled, and then evaluated using a linear model trained on the labelled training set.

We also decided to investigate the structure of the embeddings learned by the encoder during the training of MCL by visualising the embeddings of the tabular MNIST data using a t-SNE embedding (van der Maaten and Hinton, 2008). t-SNE, which stands for t-distributed

stochastic neighbor embedding is a method for visualising high-dimensional data on a two dimensional graph by applying non-linear dimensionality reduction. We plotted t-SNE embeddings for both the original inputs and the learned encodings, with each digit class plotted in a separate colour, shown in Figure 3.2. We trained the encoder on half of the total dataset and used the other half for visualisation so that the visualised examples would be unseen during training. These embeddings look relatively similar, but there appears to be slightly more separation between some of the classes in the MCL encodings, such as between the grey and pale blue classes. This increased separation between the classes means that the linear classifier is able to find decision boundaries in the higher dimensional space that better separate the classes and ultimately achieve better performance.



(a) Raw inputs (b) MCL encodings

Fig. 3.2 t-SNE embeddings of the tabular MNIST dataset for both the raw inputs and the representations learned by MCL. Each digit class is plotted using a distinct colour.

MCL involves using a hyperparameter p to control the probability of masking each individual variable, and so to investigate if varying p made a significant difference, we experimented with different values of p and their effect on the accuracy for the UCI Income dataset. The results are plotted in Figure 3.3, which shows that varying the value of p can have a large impact on the results. The two values that are of course very poor are $p = 0$ and $p = 1$, as these are the cases where there is no masking and fully masked inputs respectively. In both cases the encoding seems to have captured some information from the input because the ratio of dataset labels is about 0.76, so given a completely uninformative representation as input this would be the best a downstream model could perform.

In the case of no masking, the poor performance is due to the dot product of the normalised encodings of identical inputs (as they have not been masked) being equal to 1, and so the contrastive loss will simply be enforcing that encodings of different inputs be far

apart, which is not in itself a very useful task. In the case of fully masked inputs, the slight improvement over random may be due to the fact that even a random projection into a higher dimension can improve performance for linear regression.

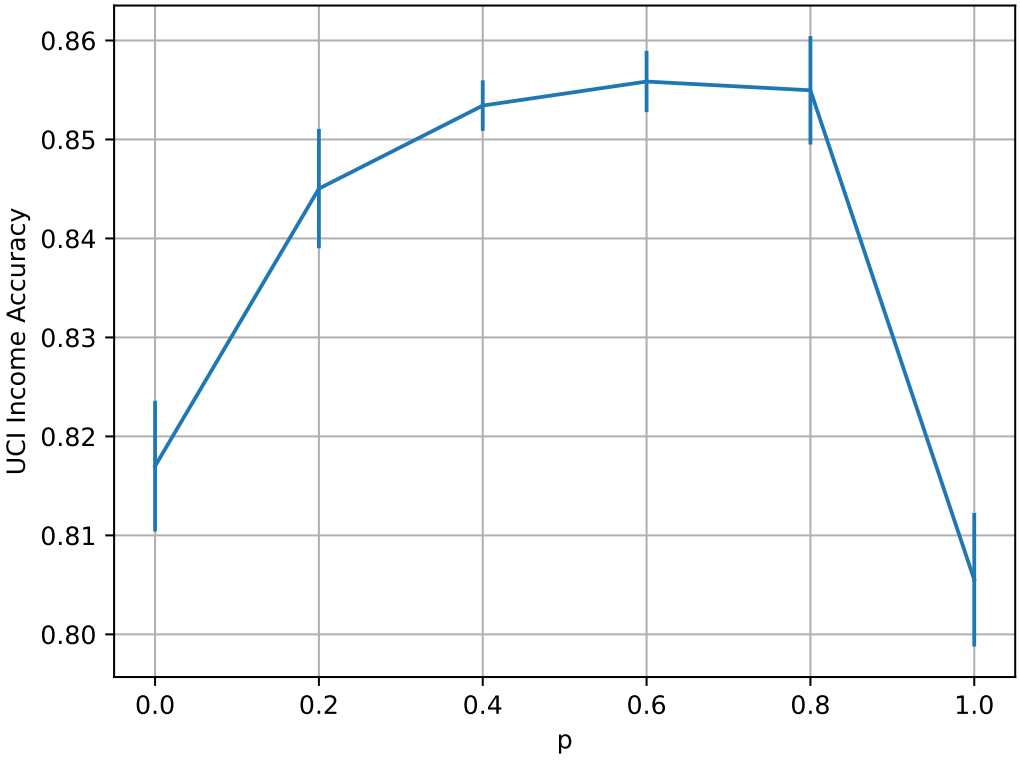


Fig. 3.3 Graph showing the effect of varying p, a parameter giving the probability of masking any given feature, on UCI Income classification accuracy for independent masking with MCL.

3.4.2 Comparison of Masking Approaches

Whilst the performance of MCL was superior to all previous approaches on all datasets, our method of masking the inputs independently with some probability p, still has some potential drawbacks. For instance, there will almost always be shared features between each of the two instances in a positive pair after having independent masks applied, and this may make the problem too easy for the encoder, which could overfit to a few features as opposed to learning deeper structure. In all previous contrastive learning methods in the literature, instances in a

positive pair do not share any unaugmented input features.

Another issue is the fact that the value of 0 may be a semantically meaningful input in some cases, such as for one-hot encoded categorical variables, and so using zero imputation for masking inputs may lead to problems such as learning to treat the value of 0 as uninformative for all inputs. To try and address these issues, we investigated two other approaches to masking.

Complementary Masking

To address the issue of shared input features between instances in a positive pair we decided to investigate complementary masks, so that the masks for each instance in a positive pair are complements of one another, meaning there would be no shared features. This can be thought of as the two instances in the positive pair being a half of the same input. The process for generating the positive pair would be as follows:

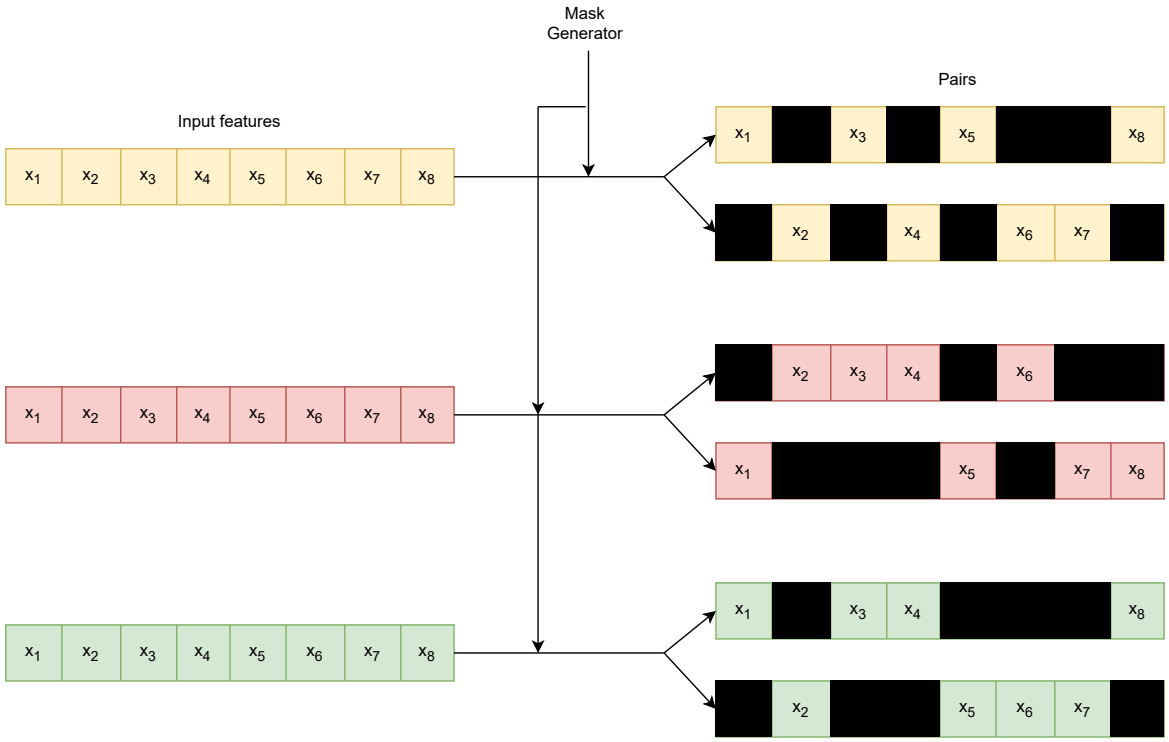
$$\mathbf{m} = [m_1, m_2, \ldots, m_d] \in \{0,1\}^d \tag{3.9}$$

$$m_i \backsim B(0.5) \;\; \forall m_i \tag{3.10}$$

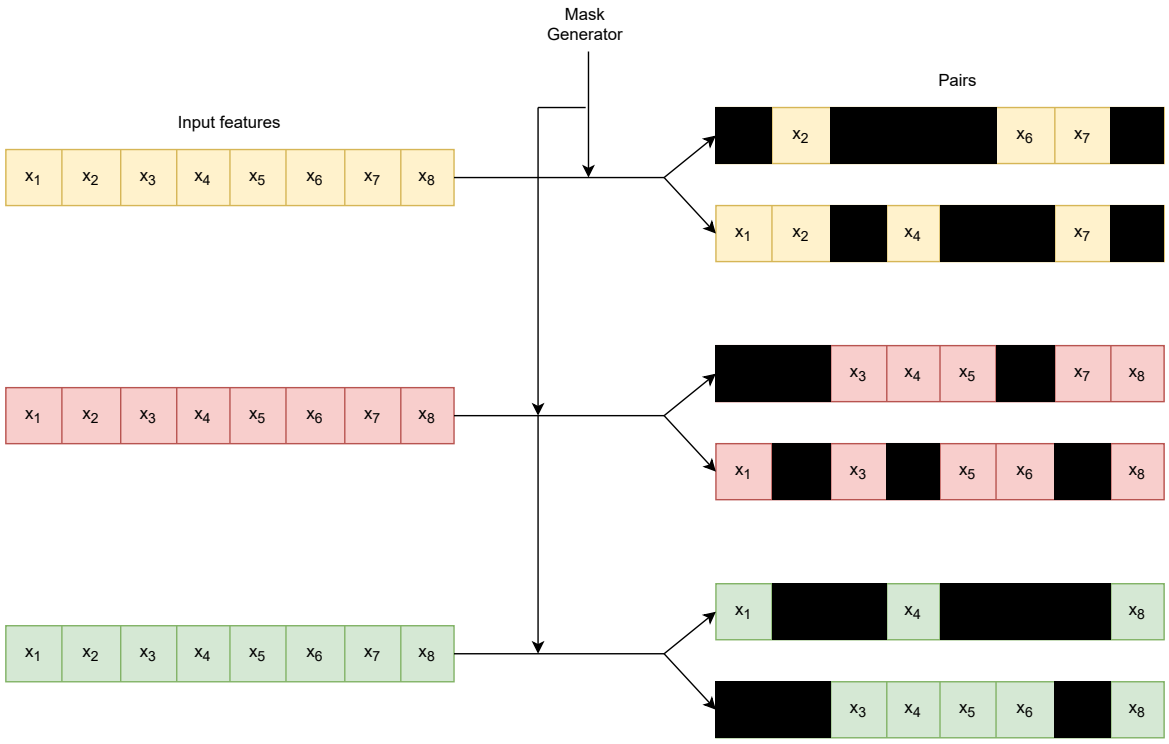
$$\tilde{\mathbf{x}} = \mathbf{x} \odot \mathbf{m} \tag{3.11}$$

$$\tilde{\mathbf{x}}' = \mathbf{x} \odot \neg\mathbf{m} \tag{3.12}$$

where $(\tilde{\mathbf{x}}', \mathbf{x})$ is the positive pair. A comparison of the batch generation processes for the two different masking procedures is shown in Figure 3.4.



(a) Complementary masking



(b) Independent masking

Fig. 3.4 Diagram showing the batch generation process for two different masking algorithms. In the complementary case, there are no shared features between the inputs, and in the independent case, each feature is masked independently in the two instances of a positive pair.

In order to investigate whether complementary masking leads to more informative representations than independent masking we compared each approach on each of the benchmark datasets. The architectures and hyper-parameters of the encoder and projection head were tuned for each dataset, with the optimal values being very similar for each masking procedure. The results were averaged over 5 trials, and the performance and standard error on each of the datasets are shown in table 4.1.

Masking method	**MNIST % accuracy**	**UCI Income % accuracy**	**UCI Wine Quality MAE**
Complementary	0.8920 ± 0.0117	$0.8000 \pm 0.0.0013$	0.0855 ± 0.0026
Independent	$\mathbf{0.8960 \pm 0.0049}$	$\mathbf{0.8840 \pm 0.0102}$	$\mathbf{0.0782 \pm 0.0025}$

Table 3.2 Performance of different masking procedures for MNIST, UCI Income and UCI Wine quality datasets. MNIST and UCI Income are classification tasks, and so performance on these datasets is measured using accuracy, meaning higher values are desirable. UCI Wine Quality is a regression dataset for which we measure performance using MAE, and therefore lower values are desirable.

This comparison shows us that independent masking outperforms complementary masking in all of the datasets. This could be because the task is made too difficult by the lack of shared features, and in some cases it is not possible to map different halves of the same inputs close together in embedding space. Therefore, we conclude that the shared features are crucial for making the pretext task possible in most cases, which allows the network to learn informative representations that are similar across different combinations of unmasked input features.

Masking The Latent Representation

We also briefly experimented with masking at one of the hidden layers as opposed to masking at the input. This would be similar to dropout, and we theorised that this could have benefits compared to masking the input, as all of the information from the input would be present in the representations, and therefore there would be no cases where the crucial features had been obscured. It also deals with the issue of using a masking value of 0 which may have a significant meaning for certain inputs that the encoder should otherwise learn. However, we found this method performed very poorly, and was outperformed by a linear model trained only on the labelled data for all of the benchmark datasets, meaning that the representation

learned was less informative for training models than simply using the input features.

One reason this could happen is because there could exist a trivial solution that allowed the encoder to learn a representation that minimizes the contrastive loss but didn't learn anything informative. We investigated this by plotting the contrastive validation loss curves for the two cases of masking at the input layer and masking at the output of the encoder before the projection head on the UCI Income dataset. These are shown in Figure 3.5.

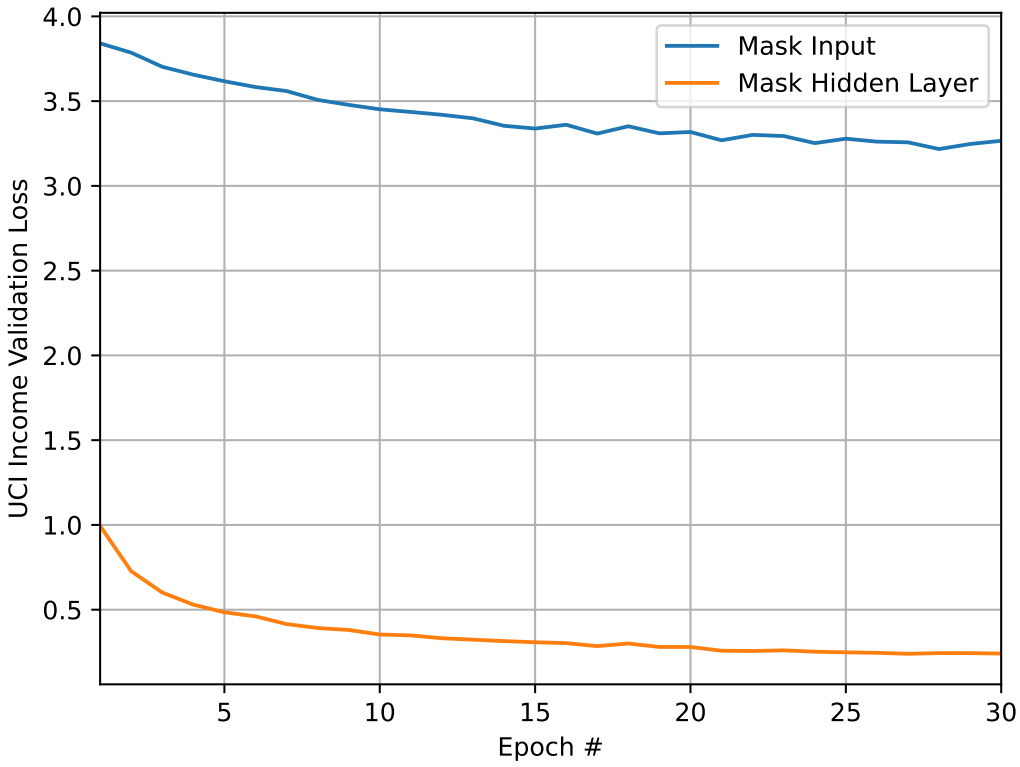


Fig. 3.5 Graph showing validation loss curves for MCL on the UCI Income dataset for both masking at the input layer and masking at a hidden layer.

This graph shows that the task becomes much easier when the masking is done at a hidden layer. Whilst it leads to a lower contrastive loss, this lower loss does not correspond to better downstream performance as the tasks are different in the two cases. The vast difference in the loss values indicates that there is some degenerate solution where the encoder does not have to learn useful information about the inputs. We suspect that one possible way the encoder could learn a representation which trivially allows the projection head to minimize

the contrastive loss is if it were to simply learn redundancy in the latent representation without actually learning anything informative. For example if every output were to be the same arbitrary function of the inputs, then trivially the projection head could learn to map this function output to a similar place in the representation space, even when the representation is masked, as this function value would be present in all of the features.

3.5 Discussion

The results presented in this chapter show that MCL gives significantly improved performance compared to supervised baselines, other self-supervised methods, and also augmentation-based contrastive approaches for tabular data. This shows that self-supervised learning is a powerful tool for machine learning on tabular data, and that the contrastive learning pretext task may lead to more informative representations than those based on reconstruction such as DAEs and VIME. This is likely to be because enforcing that representations of inputs with different features masked each contain enough information about the input in order for the projection head to map them close together in space, the encoder has to learn a representation that encodes some underlying properties of the input. This encoding is therefore informative for use in downstream tasks.

We have shown that contrastive learning can be successful even without strong assumptions being made about the structure of the data or invariances that the encoder should learn, which is something that all methods for contrastive self-supervised learning prior to this have made use of.

We also observed from our experiments that different masking procedures can have a big impact on the performance of the model. For instance, if there are no shared features between instances in a positive pair, then the problem becomes too difficult and there may not be enough shared information between the inputs to learn a meaningful representation. Therefore, in the next chapter we propose a novel approach to learning a mask generator which we call Adaptively-Masked Contrastive Learning (AMCL). This method learns to preserve the information in the input and ensure that the method for generating positive pairs allows for enough shared information between the inputs, whilst still keeping the pretext task challenging enough to force the encoder to have to learn an informative representation.

Chapter 4

Learning The Mask

4.1 Motivation

We observed from our previous experiments that the difficulty of the pretext task, which can be controlled by varying the parameter p or by changing the masking procedure, can have a dramatic impact on the performance of MCL. The approach taken in the previous chapter to masking was to mask each variable independently with an equal probability for each instance in a positive pair. However, it is not clear that this is the optimal approach, and it is possible that masking in this way could lead to positive pairs in which the crucial features in each instance are masked out, and there is very little common information shared between the inputs in each pair. This contradicts a lot of the approaches in the literature in which the methods for generating positive pairs are designed in such a way as to preserve the underlying semantic meaning in each input. Therefore, in this section we investigate an approach in which we use a mask generator with learnable parameters and optimise it in a collaborative way with the encoder, so that it can learn to preserve the information in the input.

Specifically, we investigate learning a mask generator which can correlate the masking of different inputs, but leave the marginal probability of each input being masked fixed. This mask generator will not be a function of each input but be global across all inputs. This will allow it to learn not to mask informative and correlated features at the same time, and to preserve some information in every input. By keeping the marginal probability of masking fixed for each variable, the encoder will still observe each input feature with equal probability which will prevent it from never seeing certain features. We call this approach Adaptively-Masked Contrastive Learning (AMCL).

4.2 Method

4.2.1 Concrete Distribution

In order to learn a distribution over the masks with learnable parameters, we need to be able to draw discrete samples from this distribution. However, it is impossible to differentiate a sample from a distribution with respect to the parameters of that distribution, and therefore it would seem to be impossible to calculate the gradient of the parameters of the mask generator with respect to the contrastive loss function using backpropagation. However, there do exist tricks to calculate gradients through continuous stochastic samples. The most common approach to backpropogating through stochastic samples is the reparameterization trick, which was initially invented for use with Variational Autoencoders (VAEs) (Kingma and Welling, 2014).

The key to the reparameterisation trick is to restructure the way the random variables are constructed. If a variable f depends on a stochastic variable z parameterized by ϕ which itself depends on x, so that z is sampled from the distribution $q_\phi(z|x)$, then we can restructure the model so that ϕ are viewed as deterministic variables we wish to optimize. We do this by introducing a variable ε which is a stochastic variable drawn from a distribution $p(\varepsilon)$ whose parameters remain fixed, and which accounts for all of the stochasticity. We can then view z as a deterministic variable depending on each of ϕ, x and ε calculated by a function $g(\phi, x, \varepsilon)$.

The reparameterisation trick is commonly used to differentiate samples of a Gaussian distribution with respect to the mean and covariance of the distribution. Say we have a variable $\mathbf{n}$ which is drawn from a multivariate Gaussian distribution:

$$\mathbf{n} \backsim \mathcal{N}(\mu, \Sigma) \tag{4.1}$$

and we wish to differentiate samples of $\mathbf{n}$ with respect to the parameters μ and Σ. This can be achieved by reparameterising the samples of $\mathbf{n}$ as a constant μ with the addition of a variable ε multiplied by Σ, where ε is drawn from a zero mean unit variance Gaussian:

$$\varepsilon \backsim \mathcal{N}(\mathbf{0}, \mathbf{1}) \tag{4.2}$$

$$\mathbf{n} = \mu + \varepsilon\Sigma \tag{4.3}$$

which gives samples of $\mathbf{n}$ which have the correct mean and covariance, but which are generated in a way that means they are deterministic given μ, ε and Σ.

The reparameterization trick has been adapted to deal with discrete variables in a method known as CONtinuous relaxation of disCRETE random variables (CONCRETE) (Maddison et al., 2016), which can be used to approximate discrete samples from a categorical distribution in a differentiable way. This approach was also concurrently researched in (Jang et al., 2017) who refer to it as the Gumbel-Softmax. The approach is based on of the Gumbel-Max trick (Maddison et al., 2015), which is a method for efficiently drawing samples $\mathbf{z}$ from a categorical distribution by calculating:

$$\mathbf{z} = \text{one hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right) \tag{4.4}$$

where π_i are the class probabilities and the g_i are samples from the Gumbel distribution Gumbel(0,1):

$$p(x) = e^x e^{-e^x} \tag{4.5}$$

which can be sampled from as follows:

$$u_i \backsim \text{Uniform}(0,1) \tag{4.6}$$

$$g_i = -\log(-\log(u_i)) \tag{4.7}$$

$$\tag{4.8}$$

However, the arg max in equation 4.4 is non-differentiable so we still cannot use the reparameterisation trick. The key insight of (Maddison et al., 2016) was to approximate the arg max function with a softmax function weighted by the parameter λ:

$$z_i = \frac{\exp((\log(\pi_i) + g_i)/\lambda)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\lambda)} \tag{4.9}$$

The important property of this distribution is that as λ becomes very small, the samples become very close to discrete samples from a categorical distribution, i.e. all but one z_i become very close to 0, and the other becomes very close to 1. This allows us to use the reparameterisation trick, as $\mathbf{z}$ is now a differentiable function of the random variables $\mathbf{g}$ and the parameters π of the categorical distribution.

4.2.2 Adaptively-Masked Contrastive Learning

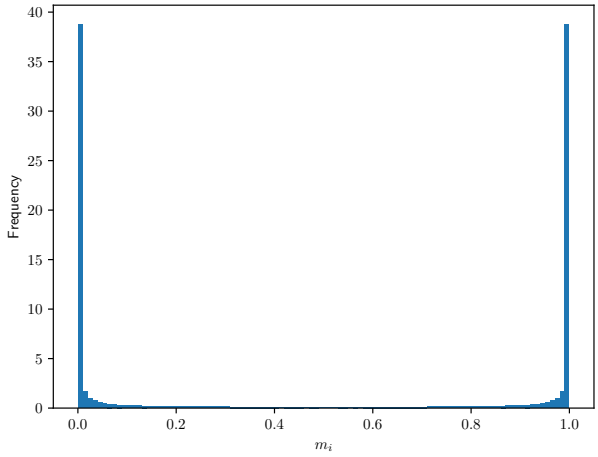
For the mask generator in AMCL, we make use of a variation of the Concrete distribution that has been adapted to deal with samples from a Bernoulli as opposed to a categorical distribution by using the sigmoid function. We shall refer to this as the Gumbel-Sigmoid distribution. The main difference is that we can avoid sampling twice from the Gumbel distribution (which would be required if the Gumbel-Softmax were used with two classes) by sampling from the distribution of the difference of two Gumbel samples. We use the following formula for generating a mask $\mathbf{m} = [m_1, m_2, \ldots, m_d]^T$ given a Bernoulli distribution parameterised by p, following the approach from Appendix B of (Maddison et al., 2016) :

$$u_i \backsim \text{Uniform}(1,0) \tag{4.10}$$

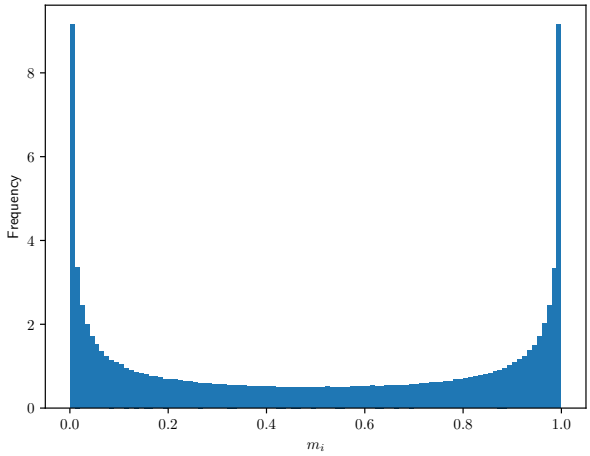
$$g_i \backsim \log(u_i) - \log(1-u_i) \tag{4.11}$$

$$m_i = \frac{1}{1+\exp((\log(\frac{p}{1-p}) + g_i)/\lambda)} \tag{4.12}$$

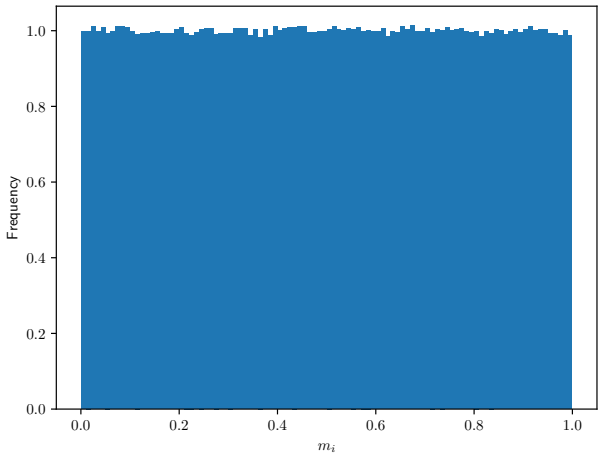
Graphs showing samples from this distribution for 3 different values of λ, for $p = 0.5$ are shown in Figure 4.1.



(a) $\lambda = 0.1$



(b) $\lambda = 0.5$



(c) $\lambda = 1$

Fig. 4.1 Graphs showing samples from the Gumbel-Sigmoid for different values of λ.

However, the samples m_i from each of the Gumbel-Sigmoid distributions would be independent without any correlations. The way we allow the mask generator to learn to correlate each m_i is by adding structure to the uniform samples u_i and therefore to the samples g_i. In order for the Gumbel-Sigmoid to still be valid, we require that the u_i still have a marginal distribution $p(u_i) = \text{Uniform}(0,1)$. Our approach therefore involves sampling from a correlated Gaussian, and transforming this sample such that it is a valid sample from the correct marginal distribution but it is correlated with the other variables of $\mathbf{u}$. To do this, we again make use of the reparameterisation trick.

We wish to generate correlated samples from a uniform distribution, and so the approach we take to do this is to generate samples from a zero mean Gaussian with a correlation matrix parameterised by a lower triangular matrix L, the parameters of which we wish to learn, and then transform the variables to uniform variables using the Gaussian Cumulative density function. We start by sampling ε from a zero mean unit Gaussian:

$$\varepsilon \backsim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4.13}$$

Then, we transform the samples using a lower triangular matrix L:

$$\tilde{\mathbf{n}} = L\varepsilon \tag{4.14}$$

which creates the samples $\tilde{\mathbf{n}}$ which are drawn from the distribution:

$$\tilde{\mathbf{n}} \backsim \mathcal{N}(\mathbf{0}, L^T L) \tag{4.15}$$

but are generated such that the parameters of L are differentiable with respect to the samples due to the use of the reparameterisation trick. The components $\tilde{n}_i$ of $\tilde{\mathbf{n}}$ will be correlated with each other but not necessarily have unit variance, and therefore we divide each $\tilde{n}_i$ by their standard deviation:

$$n_i = \frac{\tilde{n}_i}{\sqrt{(L^T L)_{ii}}} \tag{4.16}$$

which gives n_i which are correlated samples from a Gaussian but whose marginal distribution has zero mean and unit variance. Now in order to transform these into samples whose marginal distribution is Uniform(0,1) we simply use the Gaussian cumulative density

function:

$$u_i = C(n_i) \tag{4.17}$$

$$\text{where} \quad C(n_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{n_i} e^{-t^2/2} dt \tag{4.18}$$

which gives us u_i with the desired properties. A computation graph showing our approach to generating masks with learnable covariances governed by L is shown in Figure 4.2.

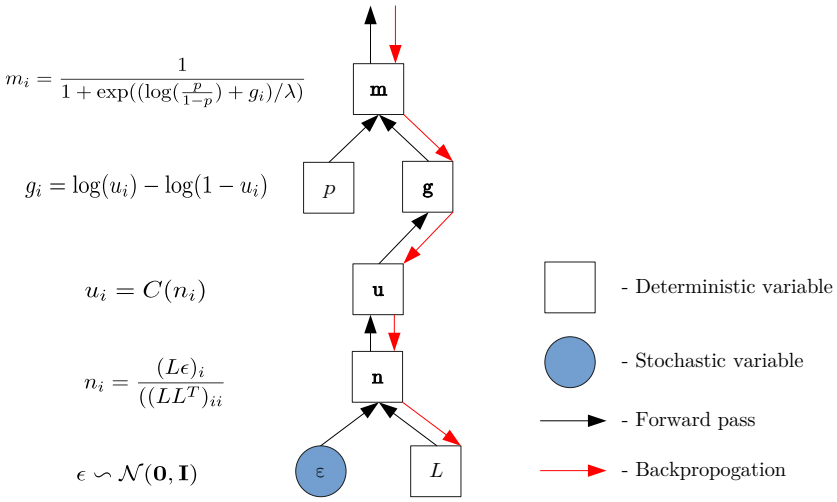


Fig. 4.2 Computation graph of the learned masking process.

The masks generated by this process are then applied to the input using element-wise multiplication. This means that the overall function that maps the noise ε to the outputs of the contrastive projection head is fully differentiable with respect to the parameters of L, which means we can optimise them with respect to the contrastive loss. We set the learning rate for these weights during training to be the same as for the encoder network and the projection head weights and optimise them all jointly.

4.3 Experiments

4.3.1 CUBE dataset

In order to investigate in which situations adaptive masking helps and which it doesn't, we conducted some experiments on synthetic data. Synthesizing datasets will help identify strengths and weaknesses of our approach as the data can be generated such that we can control aspects including number of classes, class imbalance, number of training and testing instances and correlation between variables. In our case, one of the dataset properties we are most interested in is the amount of correlation between variables and how the different models perform on datasets with varying amounts of correlation between input variables.

Our method for generating synthetic datasets was inspired in part by the dataset proposed in (Rückstieß et al., 2012). We will refer to the class of datasets that we generate as CUBE datasets, and they are generated by sampling points at random from the corners of a hypercube of side-length 1, with one corner at the origin and the opposite corner at $\{1\}^n$. Each hypercube corner will be given by a binary vector $\mathbf{z}_i$, and we create a binary classifcation problem by setting class label y_i for each sample as the parity of this vector. We choose parity as it will give interesting non-linear optimal decision boundaries.

The input vectors $\mathbf{x}_i$ are generated from $\mathbf{z}_i$ by repeating each element of $\mathbf{z}_i$ M times, and then adding noise from a Gaussian. This generative process is as follows:

$$\mathbf{z}_i = [z_{i1}, z_{i2}, \ldots, z_{iN}]^T \quad : \quad z_{ij} \backsim B(0.5) \tag{4.19}$$

$$y_i = z_{i1} \oplus z_{i2} \oplus \cdots \oplus z_{iN} \tag{4.20}$$

$$\mathbf{x}_i = [\underbrace{z_{i1}, z_{i1}, \cdots, z_{i1}}_{M \text{ times}}, \underbrace{z_{i2}, z_{i2}, \cdots, z_{i2}}_{M \text{ times}}, \cdots \underbrace{z_{iN}, z_{iN}, \cdots, z_{iN}}_{M \text{ times}}] + \varepsilon_i \tag{4.21}$$

$$\varepsilon_i \backsim \mathcal{N}(0, \sigma^2 \mathbf{I}) \tag{4.22}$$

where $\oplus$ is the XOR function. This dataset is parameterised by the variables M, N and σ^2, and by the size of the dataset. By varying σ^2 we can control how informative each variable is, and how easy the problem is. By varying M we can create datasets in which the information about each z_{ij} is spread across many variables. For example when M is high, each latent element of z_i is represented in $\mathbf{x}$ by many samples from a Gaussian with mean z_i which will therefore be correlated. This is desirable for our evaluation of self-supervised models, because in the extreme case where M is equal to one, every variable is independent and there is no structure to learn and therefore self-supervised learning will be not be helpful. However,

as M increases there is more correlation and structure between the variables and therefore more to learn via self-supervision.

Another property of this dataset which makes it useful for evaluating self-supervised models is that parity is the multi-variable generalisation of XOR. Linear models famously perform very poorly at learning the XOR function, as a linear decision boundary is never able to classify the points better than random, and the same applies for the parity function. This can be seen in Figure 4.3 which shows a plot of $\mathbf{x}$ space for the CUBE dataset when $M = 1$, $\sigma^2 = 0.2$ and $N = 3$. This means when we evaluate the encodings learned by using linear models, any improvement over random that is acheived must be because of structure learned in the representation.

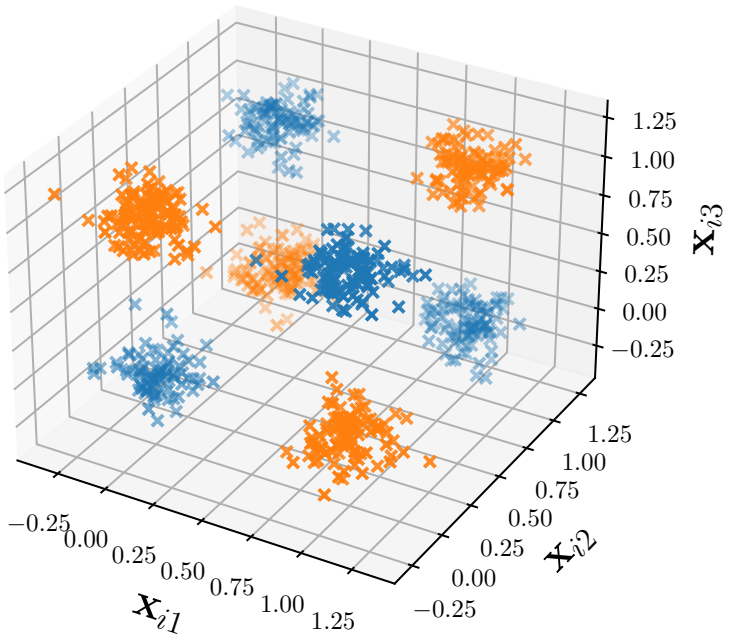


Fig. 4.3 Graph showing the $\mathbf{x}$ space of a CUBE dataset generated using $M = 1$, $\sigma^2 = 0.1$ and $N = 3$. Points in the blue clusters belong to class 0, and the orange clusters to class 1.

Our hypothesis is that learning the correlations in the mask will help when there is a small amount of correlation and structure to learn in the unlabelled inputs. In the fixed mask case,

there will be a high chance that a large amount of the structure in the input will be masked out at random, and therefore the pretext task will become less useful, as two instances in a positive pair may have largely different underlying information. However, if the mask can learn to preserve the structure in the input, then each time a positive pair is generated, the instances may still be informative. In the case where there is a lot of structure in the data, and a lot of input variables, such as for datasets similar to the tabular MNIST dataset, then it is unlikely that random masking will severely corrupt the information in the input and therefore we would see little if any improvement in this case.

We compared the performance of the original MCL with AMCL on different instantiations of the CUBE dataset in which we vary the amount of structure in the data by varying the M parameter between 1 and 4. We use a dataset of size 5000 with 4000 unlabelled and 1000 labelled, and we set $\sigma^2 = 0.2$ and $N = 4$. In order to account for the increased input size given by increasing the M parameter, the number of hidden units in the encoder network is increased proportionally with the input size. The results were averaged over 20 trials and are shown in Figure 4.4.

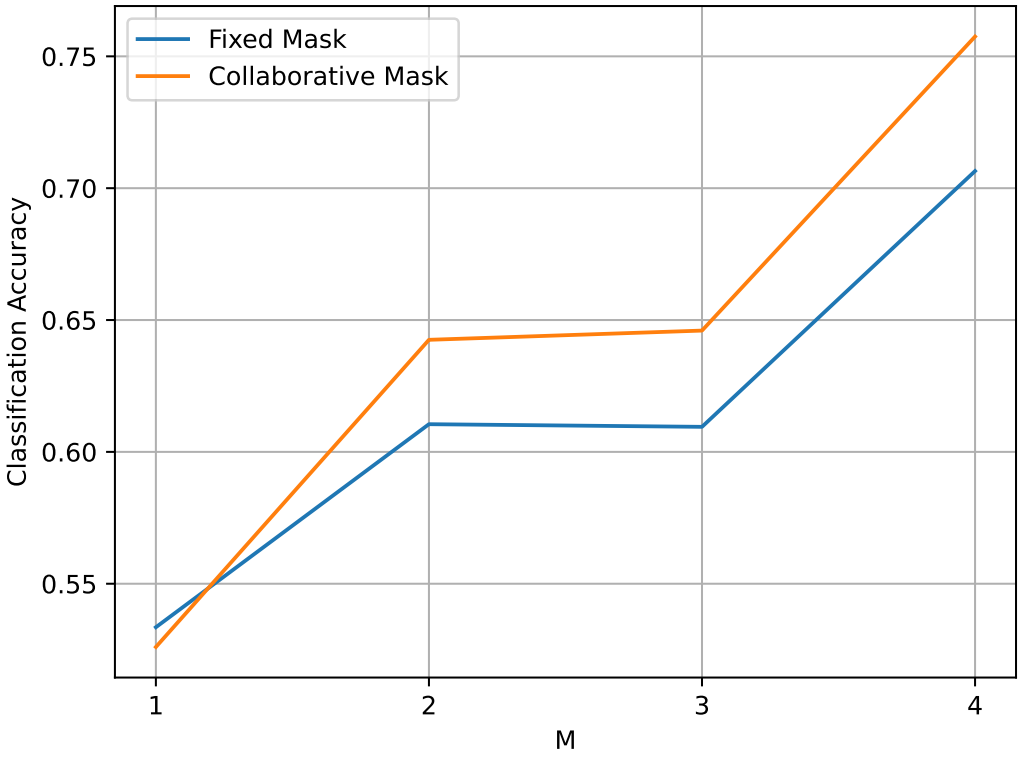


Fig. 4.4 Graph showing the performance of MCL (blue) and AMCL (orange) on variations of the CUBE dataset, created with different M values. As M increases, the amount of structure and correlation between the input variables increases.

We can see from Figure 4.4 that AMCL gives improvements in accuracy in all cases except in $M = 1$. However, in the case where $M = 1$ there are no correlations for the mask generator or for the encoder to learn, and so this is not a very interesting example. We expected that the two approaches would perform similarly as M increased, but even when there are 4 input features for every latent variable (the elements of $\mathbf{z}$), the adaptive masking helps. In order to understand how AMCL is learning to mask the variables we visualised the covariance matrix of the mask samples for the case where $M = 2$, shown in Figure 4.5. We can see from this that the mask generator has learned what we expected, i.e. that it does not mask out the input variables that are correlated at the same time, and this leads to positive pairs that allow the encoder to learn the correlations and dependencies in the inputs.

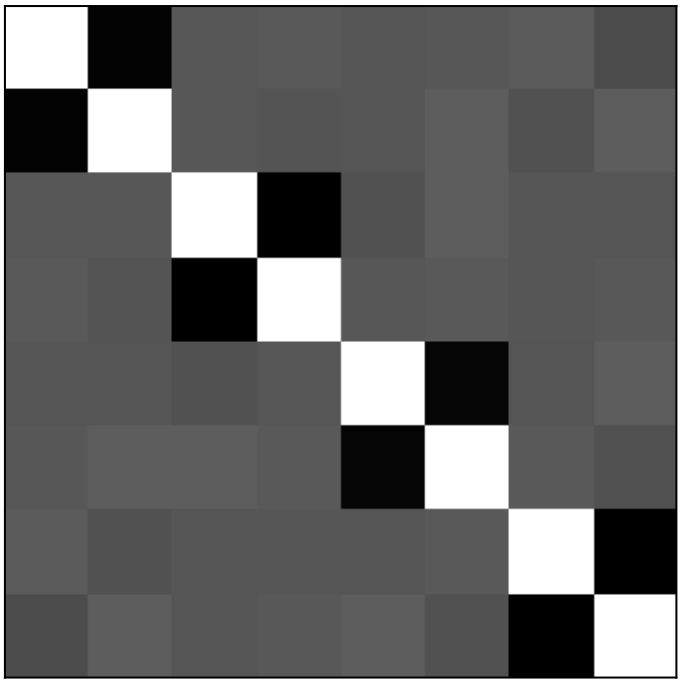


Fig. 4.5 Visualisation of the covariance matrix of the learned masks for the CUBE dataset when $M = 2$. White represents perfect correlation, and Black represents inverse correlation.

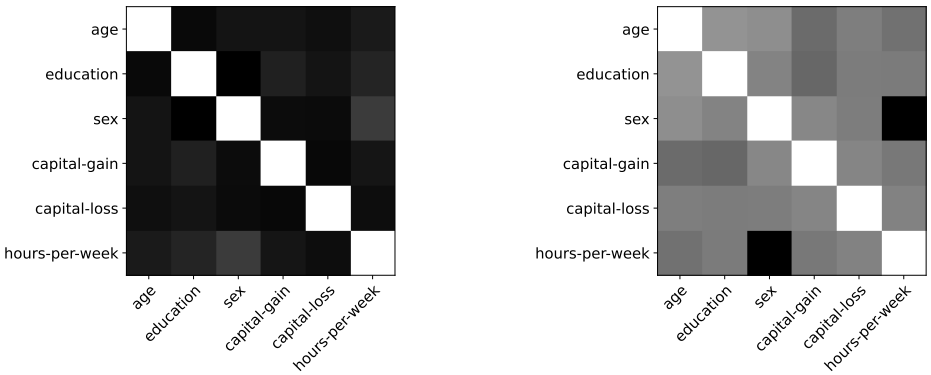
4.3.2 Benchmark Dataset Comparison

In order to evaluate whether or not learning the mask in an adaptive way improves performance, we compared the performance on the 3 benchmark datasets. The setup for this was the same as in Chapter 3, and the results were again averaged over 5 trials. For each dataset and for each masking method, the hyper-parameters were tuned separately as they had different optimum values for each approach. For instance, we found that in general the model with the learned masking function took more epochs to train and converge.

Model	**MNIST % accuracy**	**UCI Income % accuracy**	**UCI Wine Quality MAE**
MCL	**0.8960 ± 0.0049**	0.8840 ± 0.0102	0.0782 ± 0.0025
AMCL	**0.8960 ± 0.0109** -	**0.8920 ± 0.0117**	**0.0775 ± 0.0021**

Table 4.1 Coomparison of AMCL and MCL on MNIST, UCI Income and UCI Wine quality datasets. MNIST and UCI Income are classification tasks, and so performance on these datasets is measured using accuracy, meaning higher values are desirable. UCI Wine Quality is a regression dataset for which we measure performance using MAE, and therefore lower values are desirable.

In order to understand what the mask generator was learning to do, we decided to plot the covariance matrix for a subset of the features in the masks generated in the final epoch of training on the UCI Income dataset. We compare the covariance matrix of the features in the mask to the matrix of absolute values of the correlation coefficients in the raw features to see if it is learning to not mask out correlated features at the same time. We use absolute values because both negative and positive correlations represent shared information. These matrices are shown in Figure 4.6. We can see from the left plot that sex and hours worked per week have the highest correlation amongst the inputs and that the mask generator has learned not to mask these inputs out at the same time.



(a) Raw inputs (b) MCL encodings

Fig. 4.6 Visualisations of the correlations in both the inputs and the learned masks for the UCI Income dataset. on the left is the matrix of absolute values of correlation coefficients of the input variables, and on the right is the covariance matrix of the learned masks.

4.4 Discussion

The results on both the real and synthetic datasets show that AMCL does give improvements on certain datasets. The datasets which show a benefit from the learned masking are those in which there is less structure and correlation in the inputs. There is no improvement in MNIST, for example, because the structure of the data is such that random masking with a reasonable probability is very unlikely to obscure so much of the information as to make the contrastive task impossible. This is illustrated by Figure 4.7 which shows some examples of randomly masked MNIST images. Whilst there is no improvement in MNIST, this is a somewhat contrived dataset and real-world tabular datasets are more likely to resemble the UCI datasets in structure.

Fig. 4.7 Examples of MNIST images with a mask applied such that each feature is masked with a probability of 0.5. In these images a value of 0 is shown as black and 1 as white. Clearly much of the latent information in each image has been preserved.

Overall it appears that the improvements are significant as they were averaged over 5 trials. Given that UCI Income and UCI Wine Quality better fit the type of dataset that this approach is intended to be used on, as MNIST is not strictly a tabular dataset, these results show that the adaptive masking works as intended and could be a valuable addition to MCL.

In terms of training the mask generator there were some challenges that needed to be considered. For instance, the parameter λ that controls how close the continuous mask approximations will be to discrete values of 0 and 1 needs to be selected carefully. This is because whilst small values of λ give mask values that are close to discrete, they also give much smaller gradients. As λ becomes too small, the gradients of the mask values with respect to the elements of L go to 0, and so the mask generator would be unable to learn any correlations. The authors of (Jang et al., 2017) suggest starting with a higher value for λ and

gradually decaying it which was the approach we took. We start with a value of 0.5 and halve it every n epochs, where n varies with the dataset, until it reaches 0.0625. This is a somewhat ad-hoc approach and given more time we would have tried to find a more principled scheduler.

We also used the same learning rate for the mask generator as for the encoder, and trained them in unison. However, there is no reason to say that this would be optimal and so another set of experiments we could have done would be to investigate different learning rates for each, and possibly training them in alternating periods as is sometimes done with GANs. Another way in which we could have altered the training would have been to use the Straight-Through Gumbel-Softmax from (Jang et al., 2017). This is similar to the concrete distribution or Gumbel-Softmax, except that samples are rounded to their discrete values in the forward pass, but the continuous values are still used in the backward pass to allow for backpropogation. This could benefit performance as the encoder would receive as input fully masked versions of the input as opposed to there being a small amount of signal from masked variables, although it is not certain whether this change would actually have significant impact in practice.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis we develop a novel approach to self-supervised learning called Masked Contrastive Learning. We noted that existing methods for contrastive learning were not suitable for use on tabular data, and that for the most part they rely on bespoke augmentations which need to be tuned for each task. We showed that our method provides significant improvements over other self-supervised learning approaches.

We then extended our method to an approach we call Adaptively-Masked Contrastive Learning (AMCL) by masking the input variables in a way which kept the task challenging, therefore ensuring useful information was learned, but allowed the mask generator to learn not to mask out all the information in the input at once. This was implemented by making use of the reparameterisation trick and the Concrete distribution (Maddison et al., 2016), and we used them in a novel hierarchical way to learn the covariance matrix of a Gaussian which was used for sampling the masks. After visualising the learned covariance matrices, we could see that the mask generator does indeed learn to avoid masking out crucial information and the evidence suggests this is what lead to the improvement in results on most of the benchmark datasets.

This thesis has shown that it is possible to do contrastive learning without inductive bias about the structure of the dataset, and that this approach is the state-of-the-art method in terms of performance on three benchmark datasets for self-supervised learning with tabular data.

5.2 Future Work

There are several directions that future research in this area could explore. For instance, whilst we investigated a diverse range of benchmark datasets, the next step would be to apply our models to large-scale real world datasets on which improvements in performance could have a beneficial impact, such as healthcare data. In this setting we would be able to investigate datasets for which the self-supervised setting arises naturally in contrast to the slightly contrived approach of treating most of the datapoints as if they did not have labels.

Another interesting set of experiments would be to investigate the performance of MCL on other data modalities such as images and natural language and compare it's performance with other contrastive learning approaches that already exist for these data types. For images, it is clear that the inductive bias that the encoder should be invariant to certain transformations leads to representations that are very informative and concise. It is therefore likely that because MCL has no such inductive bias it would not perform as well as these methods. However, for natural language the approach could lead to strong results. It may also be possible to develop a different approach to learning the mask generator, such that the mask would depend on the input values such that all the informative words would not be masked out at once.

There are many variations of learned masking that could be investigated. These include learning an independent probability of masking each feature in such a way that doesn't allow it to not mask out any variables and still enforces variation in the inputs. Another example would be complementary masking in the case where the mask was learned. The learning of the correlations in this case could help deal with the potential issue with the fixed complementary masking of insufficient shared information between instances in a positive pair.

Another area believe could lead to interesting research is developing a semi-supervised extension of AMCL. In the case where we can make use of the labelled data for learning the representation, it may be interesting to use the labelled data in some way to learn a masking procedure. By using some labelled data, it would presumably be possible to devise a procedure which preserves the information in the input even better than our unsupervised approach. It may also help to train the encoder jointly with a combination of supervised signal and unsupervised signal, or to fine-tune the encoder on the supervised task. Training the encoder on the labelled data is something that has been shown to improve results in other works on self-supervised and semi-supervised learning (Arik and Pfister, 2019, Yoon

et al., 2020, Chen et al., 2020), but it is a slightly different paradigm, as with self-supervised learning the aim is to learn a general representation that is not specific to a given task.

In this project we have shown that contrastive learning can be applied to tabular data, and that it can be used to achieve state-of-the-art performance for self-supervised. Whilst we have given a few possible directions for future research, there are many more to be explored. The application of contrastive learning to tabular data could have far reaching benefits, and more research in this area will likely lead to even further improvements.

References

Arik, S. Ö. and Pfister, T. (2019). Tabnet: Attentive interpretable tabular learning. *CoRR*, abs/1908.07442.

Bengio, Y., Courville, A. C., and Vincent, P. (2012). Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. *CoRR*, abs/2006.09882.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Fetterman, A. and Albrecht, J. (2020). Understanding self-supervised and contrastive learning with "bootstrap your own latent" (byol).

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. (2019). Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *CoRR*, abs/2004.11362.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Li, J. (2020). Prototypical contrastive learning: Pushing the frontiers of unsupervised learning.

Li, J., Zhou, P., Xiong, C., Socher, R., and Hoi, S. C. H. (2020). Prototypical contrastive learning of unsupervised representations. *CoRR*, abs/2005.04966.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712.

Maddison, C. J., Tarlow, D., and Minka, T. (2015). A* sampling.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. *CoRR*, abs/1603.09246.

Radford, A. and Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Rückstieß, T., Osendorfer, C., and van der Smagt, P. (2012). Minimizing data consumption with sequential online feature selection. *International Journal of Machine Learning and Cybernetics*.

Saeed, A., Grangier, D., and Zeghidour, N. (2020). Contrastive learning of general-purpose audio representations. *CoRR*, abs/2010.10915.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W., and Lin, C.-T. (2017). A review of clustering techniques and developments. *Neurocomputing*, 267:664–681.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. (2021). SAINT: improved neural networks for tabular data via row attention and contrastive pre-training. *CoRR*, abs/2106.01342.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. (2018). Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. (2020). Vime: Extending the success of self- and semi-supervised learning to tabular domain. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11033–11043. Curran Associates, Inc.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412.