Self-Supervised Representation Learning for Relational Multimodal Data

Should we combine multiple pretext tasks?

Ilias Mc Auliffe[1]

Supervisor(s): Kubilay Atasu[1]**, Atahan Akyildiz**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Ilias Mc Auliffe
Final project course: CSE3000 Research Project
Thesis committee: Kubilay Atasu, Atahan Akyildiz, Burcu Özkan

Abstract

Deep Learning models can use pretext tasks to learn representations on unlabelled datasets. Although there have been several works on representation learning and pre-training, to the best of our knowledge combining pretext tasks in a multi-task setting for relational multimodal data has not been done before. In this work, we implemented 4 pretext tasks on top of a framework for handling relational multi-modal data and evaluated them based on two datasets. We first identified the best-performing masking strategy for pretext tasks that require masking. Then, we compared different combinations of the pretext tasks based on self-supervised metrics as a proxy for the quality of the representation learned. The results reveal that masking values by replacing from the column's empirical distribution yields 4.6% and 4% higher accuracy on each dataset respectively than replacing them with a fixed value. We also found that different combinations of pretext tasks, even with different numbers of tasks, converge to marginally different values and MoCo further reduces this difference. Our findings imply that the number of pretext tasks can scale efficiently allowing for a more diverse representation to be learned.

1 Introduction

Relational multimodal tabular data appears often in practice. Tree-based ensemble models, such as XGBoost, have consistently proven to outperform other Deep Learning (DL) models on tabular data [1]. The recent breakthroughs in Deep Learning on the Computer Vision (CV) and Natural Language Processing (NLP) domains have inspired research [2] into applying Deep Learning for tabular data, with the ambition of matching the performance of tree-based ensembles. Multiple DL models have been proposed for tabular data recently, such as ResNet [2] and the FT-Transformer [2]. Research in DL approaches is incentivised by the advantages DL offers over tree-based ensembles, such as accommodating multimodal data, minimizing manual data pre-processing, offering a theoretically higher performance ceiling [3], handling relational data by modelling them as graphs and the ability to pre-train to learn a representation of the data.

In this work, we investigate the potential of learning representations through pre-training a model capable of handling relational multimodal data. Pre-training has proven powerful in CV and NLP, as there are strong inductive biases in images and text. While there has been some success in pre-training DL models for tabular data [3], the success has not been as decisive as in the CV and NLP domains. Finding novel pre-training objectives for tabular data is an active area of research, with multiple recent works such as MET [4], SubTab [5], TransTab [6] and others [7; 8; 9; 10] introducing novel pre-training objectives.

Pre-training is the process of learning a representation from an unlabelled dataset, in a self-supervised or unsupervised setting, with the hypothesis that there are inductive biases that can be learned and used in a downstream task after pre-training. Pre-training involves pretext tasks each of which optimizes a pre-training objective, which may not be relevant for downstream tasks but are useful for learning a representation. Usually, pre-training objectives define a way to mask or produce different views of the data followed by a loss function which defines how well the model has performed in the pretext task. Pre-training plays a regularization role [11] which allows DL models to generalize well across multiple datasets. Pre-training is especially valuable in cases where the labelled dataset is expensive to obtain, for example classifying a rare disease. A large unlabelled dataset is used to learn generic patterns and after pre-training a DL model to discover those generic patterns, we can fine-tune it on a labelled dataset. Pre-training therefore allows for more sample-efficient and accurate learning when labelled data are scarce.

The framework we use for pre-training contains a fused DL architecture that handles both tabular data and graph data representing the relations in the data. This means two types of pre-training objectives can be combined: pre-training on tabular data and pre-training Graph Neural Networks (GNN) for graph data. There is extensive literature on pre-training tabular data [3] and pre-training GNNs [12; 13] but to the best of our knowledge, combining pre-training tabular and GNN pre-training objectives has not been explored before for relational multimodal data. There are also recent multi-task learning works which provide a strong foundation for combining pre-training objectives. One such work is MoCo, which treats the optimization of each pre-training objective as a multi-objective problem and finds a Pareto optimal solution for all objectives, which summing the losses does not guarantee.

This leads us to a natural question: *can a combination of pre-training objectives improve self-supervised metrics within the relational multimodal framework?* This is broken down into the following sub-questions.

- **(SQ1)** Which masking strategy is best for the pretext tasks that depend on a masking strategy?
- **(SQ2)** Can a combination of tabular pretext tasks and GNN pretext tasks improve self-supervised metrics?
- **(SQ3)** Can the multi-task algorithm MoCo [14] improve the performance of combinations of pretext tasks on self-supervised metrics?

The first sub-question investigates different strategies for masking the data since some pre-training objectives require masking the data. The second sub-question is a proxy to the main research question. The third sub-question, which is closely related to the second, delves into a design decision in multi-task pre-training which could improve the representation learned. Our main findings, which answer the three sub-questions, are summarized below:

- Using a masking strategy which makes it harder for the model to distinguish masked values from real values yields a better representation.
- Combining multiple pretext tasks marginally affects self-supervised metrics. This means that the encoder of

the fused architecture can learn multiple inductive biases of different pretext tasks without significantly harming the performance of individual tasks.

- Using MoCo to combine the tasks slightly improves self-supervised metrics but significantly decreases the variance in the self-supervised metrics they converge to. Therefore, MoCo allows more pretext tasks to be jointly trained, leading to a more diverse representation.

This paper is structured as follows. In Section 2 we cover the background to our work, which provides a brief explanation of the framework used in the experiments, related work and the formal problem description. Section 3 outlines our method to answer the sub-questions and the experiment setup is reported. Section 4 presents the results of our experiments, followed by Section 5 with a discussion of the results and how they relate to existing literature. Finally, we disclose ethical concerns in Section 6 and we conclude and discuss future work in Section 7.

2 Background

To introduce our work, we first explain the background leading up to our work. First, the multi-modal DL framework used for this work is briefly described. Secondly, related work in the fields of pre-training and multi-task learning is covered. Lastly, we formalize the problem this work aims to solve in order to answer the main research question.

2.1 Muti-modal DL framework

The DL framework this work builds upon is a bidirectionally fused model inspired by [15]. The framework was provided by the project supervisors. The details of the design decisions of the architecture are outside the scope of this paper but a high-level overview of the architecture is presented. The architecture, simplified and illustrated in Figure 1, takes a relational multimodal dataset as input and encodes each of its modalities into a vector format. Then, it transforms them into a table and a graph and applies a series of convolutions with a tabular transformer and a GNN. The output of this architecture is the embedding of the input table, the embeddings of the input nodes of the graph and the embeddings of the input edges of the graph. The framework then stacks a Mask Cell Modelling (MCM) decoder on top of the tabular embeddings which aims to reconstruct the input data, and a Link Prediction (LP) decoder is stacked on top of the edge embeddings to predict the existence of edges in the input graph. It is important to note that each pretext task requires a separate decoder and each decoder has its own individual loss function and metric function. The framework also provides an implementation of MoCo [14].

The model outputs tabular embeddings and node/edge embeddings, meaning tabular and GNN objectives can be used. Yasunga et al. [15] from which the architecture is inspired, chose to use Link Prediction and a variation of Mask Cell Modelling for hiding and predicting text tokens. However, Yasunga et al. did not explore different pre-training objectives or combinations of objectives. This motivates the research question this work aims to address. Related work into candidate pre-training objectives for this project follows.

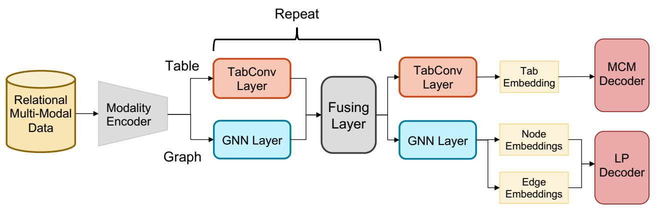

Figure 1: Our fused encoder architecture. It consists of a modality encoder, a backbone which consists of a series of tabular transformer and GNNs, and finally one or more decoders.

2.2 Related Work

Pre-training DL models is not new and its benefits have been proven. As Bengio et al. [16] have found, the success a DL model will have on a downstream task depends on how the data is represented since different representations can hide or amplify the explanatory factors of variation behind the data. Certain representations of data can help DL models perform better on downstream tasks. DL models can automatically learn a useful representation of an unlabelled dataset with pre-training objectives, which gives the model a ”head start” in performing supervised tasks. We focus on related work in tabular and GNN pre-training as they are required by the relational multimodal framework. We also focus on multi-task learning as we aim to learn a representation by concurrently optimizing multiple pretext tasks.

Tabular Pre-training Objectives

Several works have recently been proposed for tabular pre-training, each of which claims competitive results with XGBoost after fine-tuning the pre-trained models. We highlight three simple objectives described in [3].

Mask Cell Modelling (MCM). Used in [3], this objective involves corrupting parts of the input, reconstructing the input and finally evaluating the reconstruction with a loss function per modality.

Mask Vector Prediction (MV). Introduced in VIME [7], Mask Vector Prediction aims to generate a mask vector which indicates which columns will be masked for each row. The columns indicated by the mask vector are then masked by replacing them with a value from the column's empirical distribution. The reconstructive pretext task is to predict which columns were masked. The loss function is cross-entropy loss. Replacing from the column's distribution makes it harder for the model to distinguish masked values so it learns more complex intrinsic patterns, potentially leading to learning a good representation.

SCARF [17]. A contrastive learning objective, which generates positive samples by corrupting a subset of features of the input by replacing with a value from the feature's empirical distribution, and treating all other rows in the batch as negatives. InfoNCE loss is used to maximize the similarity between the input and its corrupted views.

Multiple other works introduce pre-training objectives in their pipeline. These include XTab [9], MSLR [8], TransTab [6], SAINT [10], MET [4] and TabNet [18]. Each of them has been evaluated in a different environment, making it difficult to objectively compare them. An objective comparison by Rubachev et al. [3] found that reconstruction objectives, specifically Mask Cell Modelling and Mask Vector Prediction, performed better than SCARF on a variety of datasets while also being much simpler to implement.

Graph Neural Network Pre-training Objectives

We use graphs to model relations in the relational tabular data, which requires GNNs. Xie et al. [13] have classified pre-training objectives into contrastive and predictive objectives. Contrastive objectives require a view generation strategy, which generates different variations of the graph and then uses a contrastive objective based on Mutual Information. On the other hand, predictive objectives like GraphSAGE [19] remove parts of the graph or hide attributes in the graph and then reconstruct them. We highlight link prediction, the graph pre-training objective used in the paper [15] that inspired the framework's architecture as well as the two objectives in [12].

Link Prediction (LP) [15]. This predictive objective holds out some edges and predicts them. The aim is to make the model predict the real held-out (positive) edges while correctly identifying that other non-existent (negative) edges do not exist. The balance between predicting which edges exist and which do not make the model learn representations of local neighbourhoods in the graph structure.

Context Prediction [12]. Subgraphs are used to predict their surrounding graph structures. The aim is to map nodes appearing in similar subgraphs close to the embedding space. This objective requires a definition of neighbourhood sampling to generate subgraphs, which can be computationally expensive.

Attribute Masking (AM) [12]. Node or edge attributes in the graph are masked and the GNN tries to reconstruct them. More specifically the attributes are masked, and then the GNN creates node or edge embeddings. A linear model is added as a decoder on top of the GNN embeddings to predict the original attributes.

Multi-task learning

The DL architecture can benefit from multiple pre-training objectives. The concurrent learning of multiple pre-training objectives can be considered a multi-objective optimization problem [20] and has been widely studied in the field of Machine Learning as multi-task learning. Multi-task learning can lead to more robust and universal representations [21] as knowledge is shared between the tasks being learned and inductive biases can be shared between the objectives. To ensure no objective dominates over others, algorithms such as the Multiple-Gradient Descent Algorithm (MGDA) [22] have been developed. These algorithms can provably converge to a Pareto stationary point, where no objective can be improved without harming another.

While multi-task learning can improve performance over single-objective training, it has been shown [23] that it can lead to negative transfer learning especially when the objectives are conflicting. Another issue with multi-task learning is that although the deterministic and computationally expensive MSGA algorithm can provably converge, its stochastic batch counterpart may have gradient bias which can lead it to not finding the optimal solution [14]. MoCo [14] solves this by providing a provably unbiased approach for finding the Pareto stationary point.

2.3 Formalized problem

Given are the candidate tabular pre-training objectives $O^t = \{O_1^t, O_2^t, ..., O_n^t\}$ and the candidate GNN pre-training objectives $O^g = \{O_1^g, O_2^g, ..., O_m^g\}$. Further, consider the mask types $M = \{m_1, m_2, ..., m_k\}$ each of which defines the strategy of how to mask the input data for the pretext tasks. Let c denote a function that takes a subset of pretext tasks and a mask type and combines the pretext tasks. In this work, we restrict c to be either $\sum_{o \in O^t \cup O^g} L_o$, where L_o denotes the loss function of each pretext task, or MoCo.

Find the subset of pre-training objectives $S \subset O^g \cup O^t$, the mask type m and the combination function c such that

$$\arg \max_{S \subset O^g \cup O^t, m, c} \{SSM_1(c(S,m)), ..., SSM_n(c(S,m))\} \tag{1}$$

where SSM_i denotes a self-supervised metric as a proxy to estimate the quality of the representation learned since they indicate how well the data distribution is learned.

This general problem formulation covers all 3 sub-questions. It covers **SQ1** by finding an optimal mask type and it covers **SQ2** by finding the subset of candidate pretext tasks that perform best on the selected metrics. It also answers **SQ3** as it considers the two options of combining objectives. The problem formulation is used and made more concrete in the method section.

3 Method and Experimental Setup

In this section, we present the method we followed to answer the sub-questions. We first present and explain which pre-training objectives we chose to consider in our experiments. We then describe the three experiments we conducted, each aimed at answering a corresponding sub-question. An illustration of the method is shown in Figure 2.

3.1 Pre-training objectives chosen.

We narrowed down the candidate objectives into 2 tabular and 2 GNN objectives. For the tabular objectives, we selected Mask Cell Modelling and Mask Vector Prediction for three reasons. First, Rubachev et al. [3] found that either one of these two objectives performs best over other contrastive objectives. Secondly, as Rubachev et al. also concluded, they are easier to implement. Thirdly, Mask Cell Modelling was already implemented in the provided framework, meaning only Mask Vector Prediction would have to be implemented. We chose our candidate tabular pre-training objectives $O^t = \{MCM, MV\}$ for these reasons.

As for GNN objectives, Link Prediction was chosen because it was used in a similar architecture in [15] and because it was already implemented in the provided framework. We also chose Attribute Masking as described in [12], as it was the simplest GNN pretext task to integrate with the rest of the framework. Therefore, the candidate GNN pre-training objectives were $O^g = \{LP, AM\}$. We followed the implementation of each objective as described in the paper that introduced them but we summarize them here for completeness.

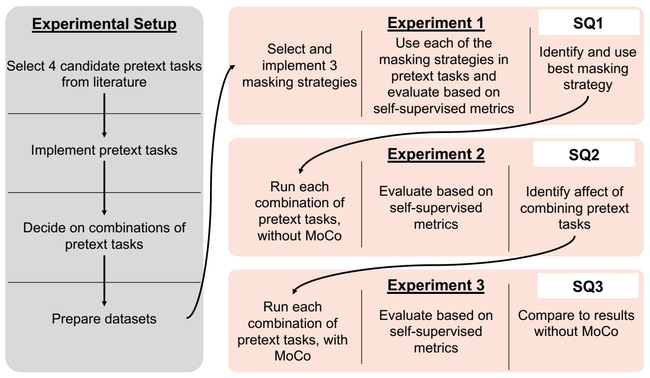
Experimental Setup
Select 4 candidate pretext tasks from literature
Implement pretext tasks
Decide on combinations of pretext tasks
Prepare datasets
Experiment 1
Select and implement 3 masking strategies
Use each of the masking strategies in pretext tasks and evaluate based on self-supervised metrics
SQ1
Identify and use best masking strategy
Experiment 2
Run each combination of pretext tasks, without MoCo
Evaluate based on self-supervised metrics
SQ2
Identify affect of combining pretext tasks
Experiment 3
Run each combination of pretext tasks, with MoCo
Evaluate based on self-supervised metrics
SQ3
Compare to results without MoCo

Figure 2: High-level overview of our method. Firstly, we choose and implement 4 pre-training objectives, then answer **SQ1** with experiment 1. Secondly, we use the best masking strategy found in experiment 1 and use it in experiment 2 to answer **SQ2**. Lastly, for experiment 3, we run the same subsets of pretext tasks as experiment 2 but with MoCo and observe the difference in results to answer **SQ3**.

Mask Cell Modelling was implemented by masking one feature per row. The masking strategy was decided after conducting experiment 1 and answering the first sub-question. After masking, a decoder is stacked on top of the framework's architecture which reconstructs the original values. Only numerical and categorical modalities are supported for masking, but the framework can be extended for more modalities. The loss function for numerical features was Root Mean Squared Error (RMSE) while for categorical features it was Cross Entropy Loss (CE).

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

$$\text{CE}(y, \hat{y}) = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{3}$$

where for numerical features y_i denotes the ground truth value for the i-th data point, $\hat{y}_i$ denotes the predicted value and for categorical features $y_{i,c}$ is 0 if the i-th data point does not belong to the c-th class, and 1 if it belongs. $\hat{y}_{i,c}$ is the output of our decoder, which is the predicted probability distribution of the categorical feature. The metric functions associated with this pretext task are RMSE for numerical columns and accuracy (ACC) for categorical columns.

$$ACC(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}(y_i = \hat{y}_i) \tag{4}$$

where y_i represents the class the i-th data point belongs to and $\hat{y}_i$ is the class with the highest predicted probability from the decoder. An RMSE of 0 is the best, with larger values being worse. For ACC, 0 is the worst possible value and 1 is the best.

Mask Vector Prediction was implemented by generating a mask vector $\boldsymbol{mv} = \{0, 1\}^n$ where n is the dimensionality of the data. If $\boldsymbol{mv}_i = 0$, then the i-th feature will not be masked but if $\boldsymbol{mv}_i = 1$ the value is masked with the masking strategy found in sub-question 1. We only mask one feature per row, so $\sum_{i=1}^{n} \mathbf{mv}_i = 1$. A decoder is consequently stacked on the framework's model that predicts $\boldsymbol{mv}$ and uses Cross Entropy Loss for the loss function between the model's prediction and the real mask vector. The metric function is ACC, where the accuracy of predicting the correct masked column is measured.

Link Prediction works as follows. First, all edges in the current training batch are considered positive edges and are removed from the graph. Secondly, the neighbourhood is sampled to obtain a larger subgraph, which acts as the context from which we can predict the positive edges. Lastly, we create artificial negative edges such that the model learns that negative edges do not exist, while positive edges do exist. After sampling positive and negative edges, we stack a linear decoder on top of the GNN's predictions for positive and negative edges and compute the loss as

$$LP_{pos}(\boldsymbol{pred}^{+}) = -\frac{1}{n}\sum_{i=1}^{n}\log(\boldsymbol{pred}_{\boldsymbol{i}}^{+} + \epsilon) \tag{5}$$

$$LP_{neg}(\boldsymbol{pred}^{-}) = -\frac{1}{m}\sum_{j=1}^{m}\log(1 - \boldsymbol{pred}_{\boldsymbol{j}}^{-} + \epsilon) \tag{6}$$

$$LP(\boldsymbol{pred}^{+}, \boldsymbol{pred}^{-}) = LP_{pos}(\boldsymbol{pred}^{+}) + LP_{neg}(\boldsymbol{pred}^{-}) \tag{7}$$

where $\boldsymbol{pred}^{+}$ of size N and $\boldsymbol{pred}^{-}$ of size M are the prediction vectors after going through our fused architecture and then through the MLP. Intuitively, each $\boldsymbol{pred}_i^{+}$ is the predicted probability of positive edge i existing and $\boldsymbol{pred}_j^{+}$ represents the predicted probability of negative edge j existing. The loss function wants to maximize the predicted probability of positive edges while minimizing the predicted probability of negative edges existing. We add a small fixed value $\epsilon = 10^{-12}$ to avoid passing 0 to the log function. The metric function for LP is the Mean Reciprocal Rank (MRR). It works by ordering the LP positive edge predictions of the LP decoder from highest to lowest and averaging the reciprocal rank r_i of each batch, where r_i is the position of the first true positive edge in the ordered list of predictions. The worst MRR value is 0, the best is 1.

$$MRR = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\mathrm{r}_i} \tag{8}$$

Attribute Masking followed a similar implementation to Mask Cell Modelling. We mask one value in the attribute vector of each edge and then let the GNN reconstruct the masked value. The masking strategy is the same as in Mask Cell Modelling and Mask Vector prediction, found in sub-question 1. The loss and metric functions are again the same as MCM, with the only difference being that instead of stacking the decoder on top of the tabular embeddings of the fused model, it is stacked on top of the edge embeddings.

3.2 Metric and pretext subsets selection

We chose three metrics as the self-supervised metrics from the problem formulation. The first is accuracy (ACC), primarily for the MCM decoder but we also investigate the accuracy of the AM decoder as an extra metric. The second metric is RMSE of the MCM decoder. The third metric is MRR. These three metrics were chosen since they were already provided in the given framework.

The number of runs required to evaluate all combinations of the candidate pretext tasks grows exponentially with the number of candidate tasks. Despite recent approaches to minimize the number of runs required to identify the best combination of pretext tasks, such as by analyzing the Conditional Independence of each pretext task [24], most studies adopt an empirical search. This work also adopts an empirical search, but the method used for this work can be extended to use Conditional Independence.

Selecting a feasible amount of subsets required filtering certain subsets as there are $2^4 - 1 = 15$ subsets of the 4 pre-training objectives when excluding the empty set. We reduced the number of combinations to 9 since some self-supervised metrics would not apply to certain subsets. For example, MV with AM would not have any of the three self-supervised metrics chosen. A more detailed explanation of why each subset was included or not is in Appendix A.

3.3 Experiments

We conducted three experiments to answer the sub-questions. We used two datasets in our experiments, namely IBM Transactions for Anti Money Laundering [25] (IBM AML) and Amazon Review Fashion [26]. The IBM AML dataset was used as it was provided, with minimal pre-processing. The Amazon Fashion dataset was sampled to reduce its size from 883,636 rows to 400,000 rows because of the limited resources available for this work. A summary of the datasets is provided in Table 1. Finding compatible datasets for the framework's model is not trivial, as the dataset needs to contain a single well-structured intrinsic graph, and be stored as a single table.

Experiment 1: Masking Strategy.
To find which masking strategy is best with regard to self-supervised metrics (**SQ1**), we investigated the effect of three different masking strategies. The first strategy for masking was to **remove** values by masking categorical values with a [MASK] token and numerical values with the mean of their column's distribution.

The second strategy was to **replace** the values to mask by sampling from their column's empirical distribution, similar to VIME [7]. For categorical columns, the replace strategy replaces the value to be masked with another value, that is not the same, with probability proportional to the frequency of the alternative value in the column. For numerical columns, we sample from a Gaussian distribution, with the mean and standard deviation equal to the mean and standard deviation of the column.

The third strategy, inspired by **BERT** [27], combines the previous two while sometimes not changing the value to be masked. We remove with a probability of 80%, we replace with a probability of 10% and do nothing with a probability of 10%.

Of the 4 selected pre-training objectives, only Mask Cell Modelling, Mask Vector Prediction and Attribute Masking are affected by the masking strategy. To quantify the effect of the masking strategy in this experiment, we compared self-supervised metrics on the model pre-trained with MCM, MV and AM with each of the three masking strategies. We ran each of the three pre-training configurations 2 times on the two datasets to accommodate for randomness in parameter initialization, for 10 epochs each.

The dataset is split into a train, validation and test set. The train set is used for training the model, while the test set is used to obtain the unbiased estimate of the representation learned through self-supervised metrics on the test set. For the IBM AML dataset, we split the dataset 60/20/20 for train, validation and test sets respectively while for Amazon Fashion we split it 70/10/20. Hyperparameter tuning on the validation set was part of the design decisions for the framework's architecture, which was done before this work.

Experiment 2: Combining pretext tasks.
This experiment aimed to compare the representation learned with different objectives with regard to self-supervised metrics to answer **SQ2**. The pretext tasks that depend on a masking strategy are set to use the masking strategy found in experiment 1. The experiment is carried out by running each of the 9 different chosen subsets of the pretext tasks on the two datasets. The loss functions in this experiment are combined by unweighted summing. We run each subset twice. The same train, validation and test split is used as experiment 1.

Experiment 3: Combining pretext tasks with MoCo.
The third experiment, aimed at answering **SQ3**, follows a similar setup to experiment 2. We only consider the 6 subsets of pretext tasks with more than one pretext task, since this experiment aims to investigate the effect of combining multiple tasks. The loss functions of each of the 6 subsets are combined with MoCo. Each run is run twice. We do not include Amazon Fashion in this experiment as it was shown in experiment 2 that it would not give meaningful results for this experiment. The same train/validation/test set split is used as in experiment 2.

Experimental setup. Our architecture and pre-training objectives were implemented in PyTorch, PyTorch Frame and PyTorch Geometric. All runs on the IBM AML dataset were run on a cluster of A100 GPUs, while all runs on Amazon Fashion were run on a GTX1660ti and a GTX1060 GPU.

4 Results

Experiment 1: Masking Strategy
The results of experiment 1 are summarized in Table 2. The "replace" mask type achieves a 4.6% higher MCM accuracy than the "remove" mask type. Despite "replace" showing better performance on the categorical columns through increased accuracy, it does not show any significant improvement for numerical columns in terms of RMSE for both MCM and AM. The accuracy of the MV decoder in predicting which

Table 1: Summary statistics of the datasets used for the experiments.

Dataset	#Columns	#Categorical	#Numerical	#Text	#Timestamp	#Rows
IBM AML	5	3	1	0	1	5,078,345
Amazon Fashion	5	1	1	2	1	400,000

column has been masked is almost 1 for ”remove” on both datasets since it is trivial for the model to learn masked values are marked as a fixed value: [MASK] for categorical columns and the mean for numerical columns. When the mask type is ”replace” it is harder for the model to predict which model has been masked, so the MV accuracy drops.

The learning curve of the metrics over the 10 epochs for the IBM AML dataset is visualized in Figure 3. The visualization includes an estimate of the uncertainty in the results by displaying the difference between each of the two runs for each mask type. Both runs for each mask type appear to converge to a similar value for each metric.

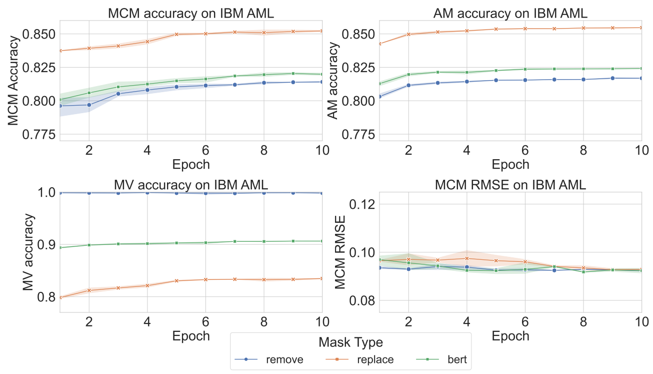

Figure 3: Comparison of the three mask types with the {MCM,MV,AM} subset on the IBM AML dataset. The shaded area for each mask type represents the area between the two runs.

Experiment 1 summary: The ”replace” mask type performs best for categorical columns, but not for numerical columns.

Experiment 2: Combining pretext tasks

The results of self-supervised learning on the two datasets with the different subsets of pretext tasks are summarized in Table 3. The difference in results is marginal for all subsets of pretext tasks. Despite the marginal difference in results for each metric across subsets, smaller subsets tend to perform slightly better than larger subsets. On the IBM AML dataset, the subset achieving the highest MCM accuracy is {MCM}, while the subset achieving the highest MRR is {LP}. The subset reaching the lowest RMSE is {MCM,LP}, however the difference is less than 0.2% compared to the next lowest RMSE result from {MCM,AM} which could be an indication that this is attributed to randomness.

The learning curve for each of the subsets for accuracy, RMSE and MRR are plotted in Figure 4. The visualization shows that the accuracy on all subsets might still be increasing, even at epoch 10. This is an indication that the model might have not converged yet with some of the subsets. A similar trend appears with MRR. Nevertheless, the visualization shows subsets with more pretext tasks such as {MCM,MV,LP} and {MCM,MV,AM,LP} converge slower for accuracy and MRR.

The results on Amazon Fashion do not vary much. All of the subsets are able to reach an accuracy of 1, or at least very close to 1, and all of the subsets perform poorly on MRR. As for RMSE, there is a slight difference, with {MCM,MV,AM} performing the best. Considering accuracy was perfect on Amazon Fashion and could not be increased with MoCo and that all subsets performed very poorly on MRR, indicating that the graph structure of the dataset was poor, the Amazon Fashion dataset is not used in the final experiment.

Summary experiment 2: The difference in all metrics across subsets of pretext tasks is marginal. However, smaller subsets of pretext tasks perform better on individual metrics than larger subsets.

Experiment 3: Combining pretext tasks with MoCo

The comparison between using Sum and using MoCo to combine subsets with multiple pretext tasks is shown in Table 4. MoCo improves for accuracy on all subsets of pretext tasks. It also improves MRR for two of the three subsets with an LP decoder. However, MoCo worsens RMSE for all except two subsets. A possible explanation is that the loss for MCM combines the individual losses of the numerical and categorical columns by summing. This means that despite all losses in the subset being Pareto optimized and not harming each other, no such guarantee applies between numerical the categorical columns. A solution would be to decouple the loss of each modality and optimize each of them independently, rather than summing the loss of each modality and then optimising the summed loss. This becomes especially important when more modalities are included.

A key advantage of using MoCo is that all subsets appear to converge to a more similar value for each metric compared to using sum to combine the tasks. The standard deviation for ACC decreases by 30.4%, indicating all subsets converge to a closer value within each other. The difference is even more profound for MRR, where a 75.4% reduction in standard deviation is reported when using MoCo. This increased stability in results across subsets implies that using MoCo allows the subset size of pretext tasks to scale better than using sum, allowing the model to learn a more diverse representation using more pretext tasks.

The learning curve in Figure 5 highlights this stability in converging for all metrics across subsets. Despite converging at slower speeds, with smaller subsets converging faster than larger subsets, all subsets converge at similar values in Epoch 10. When compared to Figure 4 which visualizes the learning

Table 2: Comparison of the effect of each of the three masking strategies. The best results for each metric on each dataset are highlighted in bold iff the result dominates all other masking strategies by at least 1%.

Mask Type	IBM AML					Amazon Fashion				
	ACC↑		RMSE↓		MV ACC↑	ACC↑		RMSE↓		MV ACC↑
	MCM	AM	MCM	AM		MCM	AM	MCM	AM	
replace	**0.8517**	**0.8545**	0.0925	0.0908	0.8341	**1**	**0.9999**	1.5046	1.5042	0.9671
bert	0.8207	0.8240	0.0933	0.0892	0.9065	0.9620	0.9616	**1.5687**	1.5179	0.9295
remove	0.8136	0.8170	0.0927	0.0908	**0.9992**	0.9608	0.9607	1.5203	1.5019	**0.9976**

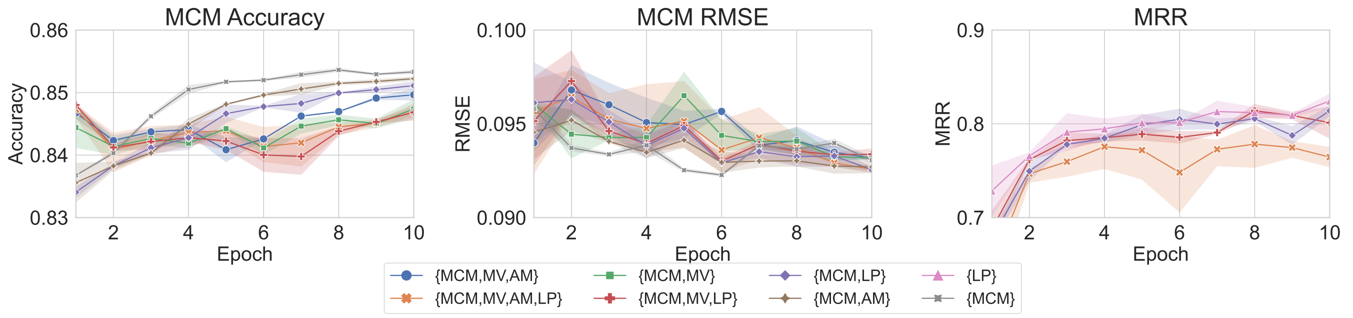

Figure 4: Results of experiment 2 on IBM AML. The learning curve of the different subsets of pretext tasks. The shaded area for each run represents the area between the two runs.

Table 3: Comparison of the performance of different combinations of pretext tasks without MoCo at epoch 10. The best result for each metric is highlighted in bold iff the result dominates all other subsets by at least 1%. The result for {AM} for Amazon Fashion is left out due to a technical error.

Pretext tasks	IBM AML				Amazon Fashion			
	ACC↑		RMSE↓	MRR↑	ACC↑		RMSE↓	MRR↑
	MCM	AM			MCM	AM		
{AM}	n/a	0.8532	n/a	n/a	n/a	-	n/a	n/a
{LP}	n/a	n/a	n/a	**0.8244**	n/a	n/a	n/a	0.0769
{MCM}	0.8533	n/a	0.0931	n/a	1	n/a	1.5916	n/a
{MCM,AM}	0.8522	0.8533	0.0927	n/a	1	1	1.6523	n/a
{MCM,MV}	0.8474	n/a	0.0931	n/a	0.9993	n/a	1.5853	n/a
{MCM,LP}	0.8511	n/a	0.0926	0.8139	0.9998	n/a	1.6426	0.0749
{MCM,MV,AM}	0.8496	0.8536	0.0932	n/a	1	0.9999	**1.5046**	n/a
{MCM,MV,LP}	0.8468	n/a	0.0934	0.8012	0.9994	n/a	1.5837	0.0768
{MCM,MV,AM,LP}	0.8469	0.8529	0.0926	0.7645	0.9996	1	1.6130	**0.0379**

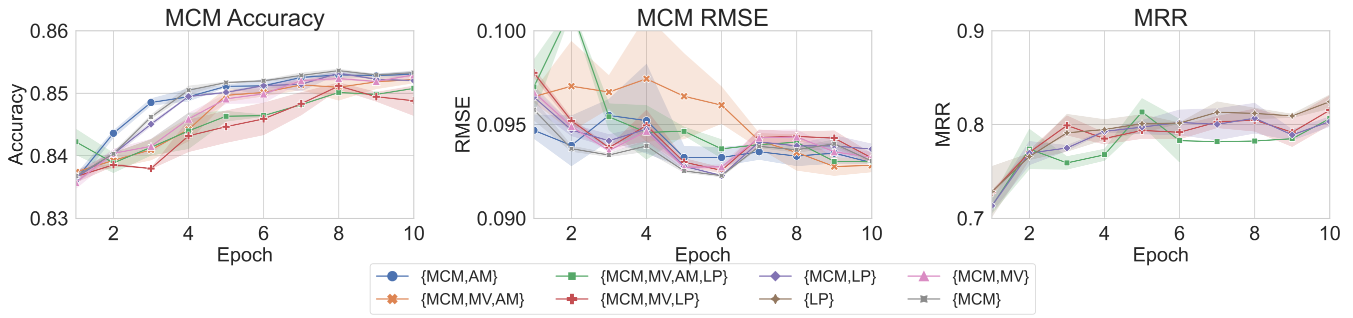

Figure 5: Experiment 3 results on IBM AML. The learning curve of the different subsets of pretext tasks. The shaded area for each run represents the area between the two runs. The second run for {MCM,MV,AM,LP} was stopped after 6 epochs due to a technical limitation.

Table 4: Comparison on the effect of using MoCo instead of Sum for multi-task learning at epoch 10. The bold value represents whether MoCo or Sum yields the best value in the given metric and subset.

Pretext tasks	**IBM AML**					
	ACC↑		**RMSE↓**		**MRR↑**	
	Sum	**MoCo**	**Sum**	**MoCo**	**Sum**	**MoCo**
{MCM,AM}	0.8522	**0.8531**	**0.0927**	0.0929	n/a	n/a
{MCM,MV}	0.8474	**0.8530**	**0.0931**	0.0932	n/a	n/a
{MCM,LP}	0.8511	**0.8520**	**0.0926**	0.0937	**0.8139**	0.8032
{MCM,MV,AM}	0.8496	**0.8522**	0.0932	**0.0928**	n/a	n/a
{MCM,MV,LP}	0.8468	**0.8488**	0.0934	**0.0932**	0.8012	**0.8153**
{MCM,MV,AM,LP}	0.8469	**0.8508**	**0.0926**	0.0930	0.7645	**0.8063**
σ	0.0023	0.0016	0.0004	0.0003	0.0256	0.0063
% Change in σ	-30.4%		-25%		-75.4%	

curve for the pretext tasks with sum instead of MoCo, we notice more stability in converging.

Summary experiment 3: MoCo provides a significant improvement in the stability of convergence across subsets. It also helps larger subsets converge to better values.

5 Discussion

Mask Type. The first experiment showed that masking values in a way that makes it difficult for the model to distinguish real values from masked values leads to better self-supervised metrics. This finding replicates what the authors in VIME [7] found, from which this masking strategy was inspired. The VIME authors hypothesized this might be attributed to the fact that the model learns more complex non-linear correlations between the features of the data. For example, masking the column containing the currency of the transaction from "USD" to "EURO" when the transaction is between two US banks it might allow the model to learn that transactions between US Banks are usually in "USD" and not in "EURO". In this example, the mask type allows the model to learn that (1) transactions between US Banks are usually in "USD" and (2) transactions between US Banks are not usually in "EURO". If the remove mask strategy was used, the model would not learn the second pattern, so this could be a potential explanation as to why replace performs significantly better. It can learn negative and positive patterns at the same time, while the remove mask type only learns positive patterns.

The superiority of the "replace" type is only evident in categorical columns, for numerical columns there is almost no impact. A possible explanation for this was the choice of distribution used to fit the numerical columns. The Gaussian distribution was used, however, numerical columns might have different distributions such as exponential or logarithmic, so the distribution of the numerical columns needs to be better approximated.

Combining pretext tasks. Subsets with different pretext tasks reach a similar value in all metrics. Despite reaching similar values, {MCM} obtains the best accuracy and {LP} reaches the highest MRR. This could be a sign that the addition of other pretext tasks harms the learning of the previous pretext task. Therefore, if there are multiple pretext tasks there might be conflicting inductive biases leading. This answers **SQ2**, as a combination of pretext tasks does not improve self-supervised metrics.

Effect of using MoCo. Using MoCo to combine the losses of the pretext tasks provides two benefits over summing the loss functions. First, MoCo reaches slightly higher values in the metrics for most subsets. Secondly, MoCo reduces the gap between the value of the metrics all subsets converge to. The second benefit is an impactful finding, as it allows the fused architecture to be trained with a larger amount of pretext tasks with minimal impact on individual pretext tasks. This finding can be exploited to pre-train the fused architecture with a more diverse set of pretext tasks. This is especially useful for pre-training with a diverse set of pretext tasks and then used for fine-tuning downstream tasks. It has been shown a larger set of pretext tasks for pre-training can lead to better downstream task performance [28].

Datasets. The datasets used for the experiments could be improved. The IBM AML dataset [25] was carefully constructed to have a well-structured transaction graph. On the other hand, the graph structure of Amazon Fashion was not explored before the experiments. This is a possible explanation for the poor MRR results on Amazon Fashion. Another issue is the low dimensionality of both datasets. There are not many features in both datasets, so it could be that the results can differ with higher dimensional datasets.

6 Responsible Research

Threats to validity

Our research has three threats to validity. Firstly, since our DL model is stochastic with random parameter initialization, it is prone to converge to different optima in different runs. Our results partially account for this uncertainty by running each combination of pre-training objectives twice. Secondly, the performance of any DL model depends on the data itself, so results on one dataset are not necessarily representative of another. For this reason, we used two datasets to arrive at more general conclusions. Nevertheless, our approach should be applied to more and of higher quality datasets to mitigate this threat further. Lastly, we have tested our implementation of the DL model and the pre-training objectives to the best of our ability but there exists the risk of having used an inefficient implementation for the provided results. We aim to make our code accessible and public in the future so that our results can be replicated and different implementations can be investigated.

Environmental impact

Obtaining the results presented in this paper required approximately 750 GPU hours of training time, which is a cost to the environment. As with any ML model with many free parameters to be learned, the time it takes for the model to be pre-trained on large datasets is significantly more than it takes to be directly trained with supervised labels. Longer training times lead to more GPU resources used, which require a significant amount of energy. We considered the trade-off between the energy required to get results and the value of results. We believe the environmental cost for this project is justified, as researching more efficient and effective pre-training can lead to more efficient downstream task training. Pre-training is only done once, and the pre-trained model can be used with a small amount of fine-tuning for a diverse range of tasks. This offsets the computational resources and environmental cost of pre-training.

7 Conclusions and Future Work

This experimental work investigated the effect of combining pretext tasks for multi-task self-supervised representation learning on relational multimodal data. Three experiments were conducted and each of them answered a specific sub-question related to combining pretext tasks. The experiments were built on top of a framework for relational multimodal data provided by the project supervisors. Two datasets were used for the experiments.

Main findings. For pretext masks which require masking the data, using a masking strategy that replaces the value from the empirical distribution of the column leads to better accuracy compared to replacing a fixed token. As for combining pretext tasks, summing the losses of each pretext task leads to similar results concerning self-supervised metrics. Nevertheless, smaller subsets of pretext tasks perform slightly better than larger ones. This means combining pretext tasks does not improve self-supervised metrics. However, by combining the losses with MoCo, this gap becomes smaller implying that the number of pretext tasks can scale better with MoCo rather than summing losses. This can lead to a more robust and diverse representation learned.

Future work. The self-supervised metrics only provide one dimension of the quality of the representation learned. Another important aspect of the representation learned with the pretext tasks is the performance of the pre-trained model on downstream tasks after fine-tuning. A more robust evaluation of the impact of combining pretext tasks would consider the performance of fine-tuning on a downstream task. Additional work should also go into preparing higher-quality datasets to improve the reliability of the results.

References

[1] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" *Advances in Neural Information Processing Systems*, vol. 35, p. 507–520, Dec. 2022.

[2] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, p. 18932–18943. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html

[3] I. Rubachev, A. Alekberov, Y. Gorishniy, and A. Babenko, "Revisiting pretraining objectives for tabular deep learning," *arXiv preprint arXiv:2207.03208*, 2022.

[4] K. Majmundar, S. Goyal, P. Netrapalli, and P. Jain, "Met: Masked encoding for tabular data," *arXiv preprint arXiv:2206.08564*, 2022.

[5] T. Ucar, E. Hajiramezanali, and L. Edwards, "Subtab: Subsetting features of tabular data for self-supervised representation learning," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, p. 18853–18865. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/9c8661befae6dbcd08304dbf4dcaf0db-Abstract.html

[6] Z. Wang and J. Sun, "Transtab: Learning transferable tabular transformers across tables," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2902–2915, 2022.

[7] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "Vime: Extending the success of self- and semi-supervised learning to tabular domain," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, p. 11033–11043. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/hash/7d97667a3e056acab9aaf653807b4a03-Abstract.html

[8] X. Weng, H. Song, Y. Lin, X. Zhang, B. Liu, Y. Wu, and J. Yang, "Mslr: A self-supervised representation learning method for tabular data based on multi-scale ladder reconstruction," *Information Sciences*, vol. 660, p. 120108, Mar. 2024.

[9] B. Zhu, X. Shi, N. Erickson, M. Li, G. Karypis, and M. Shoaran, "Xtab: Cross-table pretraining for tabular transformers," *arXiv preprint arXiv:2305.06090*, 2023.

[10] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein, "Saint: Improved neural networks for tabular data via row attention and contrastive pre-training," *arXiv preprint arXiv:2106.01342*, 2021.

[11] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, Mar. 2010, p. 201–208. [Online]. Available: https://proceedings.mlr.press/v9/erhan10a.html

[12] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[13] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: A unified review," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 2, pp. 2412–2429, 2022.

[14] H. Fernando, H. Shen, M. Liu, S. Chaudhury, K. Murugesan, and T. Chen, "Mitigating gradient bias in multi-objective learning: A provably convergent approach," in *International Conference on Learning Representations*. International Conference on Learning Representations, 2023.

[15] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. S. Liang, and J. Leskovec, "Deep bidirectional language-knowledge graph pretraining," *Advances in Neural Information Processing Systems*, vol. 35, p. 37309–37323, Dec. 2022.

[16] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, p. 1798–1828, Aug. 2013.

[17] D. Bahri, H. Jiang, Y. Tay, and D. Metzler, "Scarf: Self-supervised contrastive learning using random feature corruption," *arXiv preprint arXiv:2106.15147*, 2021.

[18] S. Ö. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 6679–6687.

[19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[20] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html

[21] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, p. 5586–5609, Dec. 2022.

[22] J.-A. Désidéri, "Multiple-gradient descent algorithm (mgda) for multiobjective optimization," *Comptes Rendus Mathematique*, vol. 350, no. 5, p. 313–318, Mar. 2012.

[23] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?" in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, p. 9120–9132. [Online]. Available: https://proceedings.mlr.press/v119/standley20a.html

[24] S. Zaiem, T. Parcollet, S. Essid, and A. Heba, "Pretext tasks selection for multitask self-supervised audio representation learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1439–1453, 2022.

[25] E. Altman, J. Blanuša, L. Von Niederhäusern, B. Egressy, A. Anghel, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[26] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[28] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

A Pre-training objective subset selection

Subset of pre-training objectives	Is Included	Reason for exclusion
Single objective		
Mask Cell Modelling (MCM)	Yes	n/a
Link Prediction (LP)	Yes	n/a
Mask Vector prediction (MV)	No	None of the metrics chosen for this paper apply to this objective.
Attribute Masking (AM)	Yes	n/a
Two objectives		
MCM+LP	Yes	n/a
MCM+MV	Yes	n/a
MCM+AM	Yes	n/a
LP+MV	No	As introduced in VINE [7], MV should only be used in combination with MCM.
LP+AM	No	None of the metrics chosen for this paper apply to this objective.
MV+AM	No	None of the metrics chosen for this paper apply to this objective.
Three objectives		
MCM+LP+MV	Yes	n/a
MCM+LP+AM	No	n/a
MCM+MV+AM	Yes	n/a
LP+MV+AM	No	As introduced in VINE [7], MV should only be used in combination with MCM.
Four objectives		
MCM+MV+AM+LP	Yes	n/a

Table 5: Explanation on the reason for exclusion for each of the 15 possible combinations of the 4 pre-training objectives.

B Experiment 2 additional plots

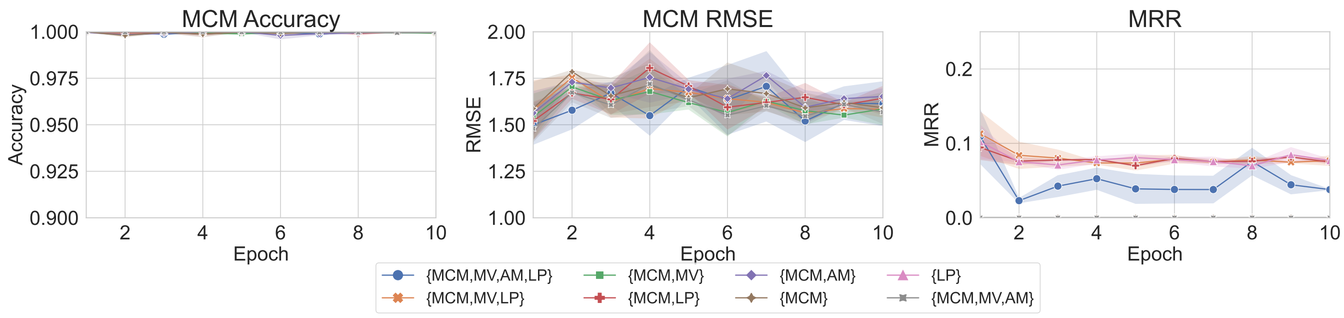

Figure 6: Results of experiment 2 on the Amazon Fashion dataset.