Relation Guided Message Passing for Multi-label Classification

Anonymous authors
Paper under double-blind review

Abstract

A well known-challenge in multi-label classification is modelling the dependencies between the labels. Most of the attempts in the literature focus on label dependencies that exhibit themselves through co-occurrences. Co-occurrences represent a *pulling* type of relationship between labels, meaning that labels that are observed together in training samples are more likely to co-occur. But other label relationships are common, such as a group of labels that never occur together. We call this a *pushing* relation. Successfully modeling such relations and the dependencies they induce can also lead to improved prediction performance. In this work, we develop a graph-based dependency module that models multiple types of relations between labels and thus captures richer dependencies. The module is designed to be flexible so that it can be integrated into most embedding-based multi-label classification approaches. We propose a generic method to extract pulling and pushing relations between labels for any multi-label data. We then present Relation Guided Message Passing (RGMP), a Transformer based classifier for multi-label classification that uses the proposed label dependency module. Experiments on benchmark datasets show that RGMP yields similar or superior performance compared to state-of-the-art methods and the approach imposes only minor additional computational and memory overheads.

1 Introduction

Multi-label classification (MLC) involves selecting the correct subset of tags for each instance from the available label set. There are numerous real world applications ranging from image annotation to document categorization (Madjarov et al., 2012). A naive approach consists of converting the problem into multiple binary classification problems. This is the *binary relevance (BR)* method (Luaces et al., 2012). A critical difference between multi-label and multi-class classification is that the class values are not mutually exclusive in multi-label learning. Usually, we anticipate that there are dependencies between the labels, and there are often multiple different types of relationships. Incorporating and learning the dependency relationships between labels during the learning process has an appealing potential of boosting predictive performance.

The problem of modelling dependencies between labels is a well-known challenge in MLC. Most attempts at modelling label dependencies focus on co-occurrences, ignoring the valuable information that can be extracted by detecting label subsets that rarely occur together. For example, consider customer product reviews; a product probably would not simultaneously be tagged by both “recommended” (i.e., reviewer is happy and recommends the product) and “urgent” (i.e., the review suggests immediate action to remedy an unsatisfactory experience). In addition to the consideration of positive and negative dependencies, the direction of a relationship should also be considered. For a multi-label image classification problem, the “ship” and “sea” labels have an obvious dependency, but the presence of the former implies the latter much more strongly than the other way around. These examples motivate the modelling of multiple types of bi-directional relationships between labels.

The simplest approach that considers label dependencies is the *label powerset (LP)* method (Tsoumakas & Katakis, 2007). This involves converting the problem to multi-class classification by treating each possible combination of labels as a separate class, i.e., a multi-label problem with L possible labels would be converted to a multi-class problem with 2^L classes. While the LP method enables us to make use of powerful multi-class classification methods, it struggles to scale to problems with a large number of labels. Aside from the LP method, the existing methods that account for label dependencies construct the relationships between labels by assessing co-occurrences in the training data (Chen et al., 2019b; Lanchantin et al., 2020; Bai et al., 2021). Co-occurrences represent a *pulling* type of relationship between labels (labels that often appear together should be strongly encouraged to appear together in predictions). However, focusing solely on frequent co-occurrences ignores other valuable information. In particular, there should be a distinction between labels that occasionally appear together and those that *never* appear together. Aside from statistical label dependencies determined by co-occurrence (or the lack thereof), there may be other valuable relations that should be considered when designing a classifier. These include semantic relations (e.g., synonyms, plural forms) or pre-defined structural relations (e.g., hierarchical category relations).

In this paper, we propose a method, entitled *Relation Guided Message Passing (RGMP)*, which employs a *Compositional Graph Convolutional Network (CompGCN)* (Vashishth et al., 2020) based approach to model multiple and bi-directional relationships between labels. We make the following contributions:

1. We design a multi-relation label embedding module that can be integrated into most embedding based methods to account for the dependencies between labels and propose a simple and efficient method to extract *pulling* and *pushing* relations between labels.

2. We design a Transformer (Vaswani et al., 2017) based classifier for the multi-label classification problem, considering two types of statistical relations between labels.

3. We perform experiments on benchmark multi-label classification datasets, comparing with classical and state-of-the-art baselines, to demonstrate the efficacy of our method.

2 Related Work

There have been multiple attempts in the literature to capture label dependencies for multi-label classification problems. *Probabilistic classifier chains (PCC)* stack a sequence of binary classifiers and predict one label at a time conditioned on previously predicted labels (Read et al., 2011; Dembczynski et al., 2010; Senge et al., 2019). PCC based methods decompose the joint probability of observing label subsets into a product of conditional probabilities. The computational complexity therefore increases exponentially with the number of possible labels. To reduce the length of the classifier chain and the corresponding model complexity, Nam et al. (2017) suggest a sequence-to-sequence architecture that focuses on predicting positive labels only. PCC models can suffer from a sub-optimal pre-defined label ordering. More recently, Gerych et al. (2021) propose learning a Bayesian network of class dependencies and leverage this network instead of following a pre-defined label ordering. However, training and inference time is an issue due to the inability to harness parallel computation.

In latent embedding learning methods the inputs and outputs are projected into a shared latent space (Bhatia et al., 2015b; Chen et al., 2019a; Ma et al., 2020; Yeh et al., 2017; Bai et al., 2021; Zhao et al., 2021; Bai et al., 2022). Yeh et al. (2017) propose C2AE, an autoencoder which learns a feature-aware latent subspace for label embeddings by imposing a canonical correlation analysis constraint on the latent space. An effective latent embedding learning method is MPVAE (Bai et al., 2021). This maps features and labels to probabilistic subspaces and uses the Multivariate Probit (MP) model to make predictions by sampling from these subspaces. Building on MPVAE, C-GMVAE (Bai et al., 2022) learns a Gaussian mixture structure on the latent space and employs a contrastive loss to account for label correlations implicitly in a data-driven manner. The methods that learn a shared latent space for labels and features do not attempt to learn label dependencies explicitly and fail to incorporate prior knowledge about label space structures. For a given sample, the contrastive loss of C-GMVAE brings the input feature embeddings closer to the embeddings of the present labels and pushes the absent ones away. Therefore, the embeddings of the labels that co-occur frequently become similar while the embeddings become dissimilar if the labels never co-occur. This approach implicitly enables incorporating a pushing type of relation in addition to a pulling type. However, for each label all events of co-occurrence and absence of other labels are treated equally, without any consideration of the statistical significance of these events. Given that the average label subset cardinality is very low compared to the total number of labels in most multi-label classification datasets, treating all absent labels equally is problematic. Ben-Baruch et al. (2021) propose ASL which uses an asymmetric loss function that operates differently on positive and negative samples and down-weights easy negative labels by hard-thresholding. This enables the gradients to focus on the labels that are present. However, this method ignores the information that the absence of labels can carry.

In another line of research that formulates the label-specific feature learning as a feature selection problem, recent studies exploit the label correlations via a label specific feature generation process (Hang & Zhang, 2021; Yu & Zhang, 2021; Li et al., 2022). This enables the resultant methods to account for co-occurrence based dependencies. However, since they perform the label-specific feature generation and classification independently, these methods cannot learn label dependencies in the context of the actual task of classification, in an end-to-end manner.

Graph-based methods for capturing label dependencies have shown promising performance (Chen et al., 2019b; Wang et al., 2019). Chen et al. (2019b) use *Graph Convolutional Networks (GCNs)* (Kipf & Welling, 2017) to map label representations to interdependent object classifiers for the multi-label image classification task. In these methods, a label graph is typically built based on label co-occurrence, with nodes corresponding to labels and edges corresponding to how two labels interact. The label graph can be combined with graph neural networks to enable the interactive learning of features and label embeddings. One of the most effective graph-based methods is LaMP (Lanchantin et al., 2020), which employs an attention mechanism to pass messages among label and feature embeddings and between label embeddings to enable learning of higher order label correlations. LaMP starts with a simple fully-connected graph or a co-occurrence label graph. A *Message Passing Neural Network (MPNN)* then passes messages among label embeddings and features to enable learning of label correlation structure that is conditioned on the features. Due to the greater flexibility offered by MPNNs compared to GCNs, MPNN based approaches achieve state-of-the-art performance on many benchmark datasets. However, all of the above graph methods use simple undirected graphs, neglecting the richness of information present in the joint empirical probability distribution of the labels, such as whether the occurrence of label A encourages or suppresses the occurrence of label B, relative to the marginal distribution of the latter.

Employing the framework introduced by Ozmen et al. (2022) for multi-label-text classification, in this paper, we use multi-relational label graph to more comprehensively capture the information present in the joint empirical probability distribution of the labels, expand the experiments from text classification to include other feature domains, perform additional ablation studies, and examine the effectiveness of the proposed module by inspecting sample-specific examples.

3 Problem Statement

Given an N-dimensional feature space and number of labels L, associated with each sample is its feature vector $\mathbf{x} \in \mathbb{R}^N$ and label set represented as a binary vector $\mathbf{y} = (y_1, \ldots, y_L)$, where $y_i \in \{0, 1\}$ indicates whether label i appears. In the multi-label classification problem, the overall objective is to design a classifier $f(\cdot)$ that maps features to labels, $\mathbf{x} \xrightarrow{f} \mathbf{y}$. Let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots \hat{y}_L)$ denote the final predicted label likelihoods by the classifier, i.e., $f(\mathbf{x}) = \hat{\mathbf{y}}$. Traditionally, the quality of estimation is evaluated by the binary cross entropy loss $\mathbb{L}_{\text{bce}}(\mathbf{y}, \hat{\mathbf{y}})$ during training:

$$\mathbb{L}_{\text{bce}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{L} -y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i). \tag{1}$$

We use a training set $(\mathbf{x}_j, \mathbf{y}_j)_{j \in \mathcal{D}_{\text{Train}}}$ to learn the parameters of the predictive models, a validation set $(\mathbf{x}_j, \mathbf{y}_j)_{j \in \mathcal{D}_{\text{Validation}}}$ to tune the hyperparameters, and a test set $(\mathbf{x}_j, \mathbf{y}_j)_{j \in \mathcal{D}_{\text{Test}}}$ to evaluate the performance.

Most of the embedding based multi-label classification algorithms represent each instance by a set of label embeddings $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_L)$ such that $\mathbf{z}_i \in \mathbb{R}^{d_{\text{emb}}}$, where d_{emb} represents the embedding dimension. We call the module that generates the embeddings the *feature to label (F2L)* transformation module. The calculated label embeddings are then used to predict label likelihoods via a final *readout* function.

The goal of this paper is to design an efficient *label dependency modelling (LDM)* module situated between the F2L transformation module and the readout layer. This module learns the label dependencies and updates the label embeddings accordingly. The proposed module is (1) capable of incorporating information regarding multiple types of bi-directional relationships between labels while determining the dependencies between them; and (2) sufficiently flexible to adapt to any embedding based method without greatly increasing the model size. The overall architecture can then be expressed as:

$$\mathbf{x} \xrightarrow{\text{F2L}} \mathbf{z} = (\mathbf{z}_1, \dots, \mathbf{z}_L) \xrightarrow{\text{LDM}} \mathbf{z}' = (\mathbf{z}'_1, \dots, \mathbf{z}'_L) \xrightarrow{\text{Readout}} \hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_L). \tag{2}$$

4 Methodology

For the multi-label classification problem, the advantage of using Message Passing Neural Networks (MPNNs) arises due to the structural power of graphs for representing relationships between the labels. However, the use of RNN based architectures has two often overlooked advantages. The ordering in the prediction path allows the architecture to focus on the next most probable label and ignore irrelevant labels. Moreover, this ordering effectively models dependencies in a directed manner, albeit in only one direction. Our proposed method RGMP strives to combine the advantages of the RNN and the MPNN approaches. In this section, we provide the methodological details of the modules that combine to form the proposed model. An overview of the proposed model is provided in Figure 1.

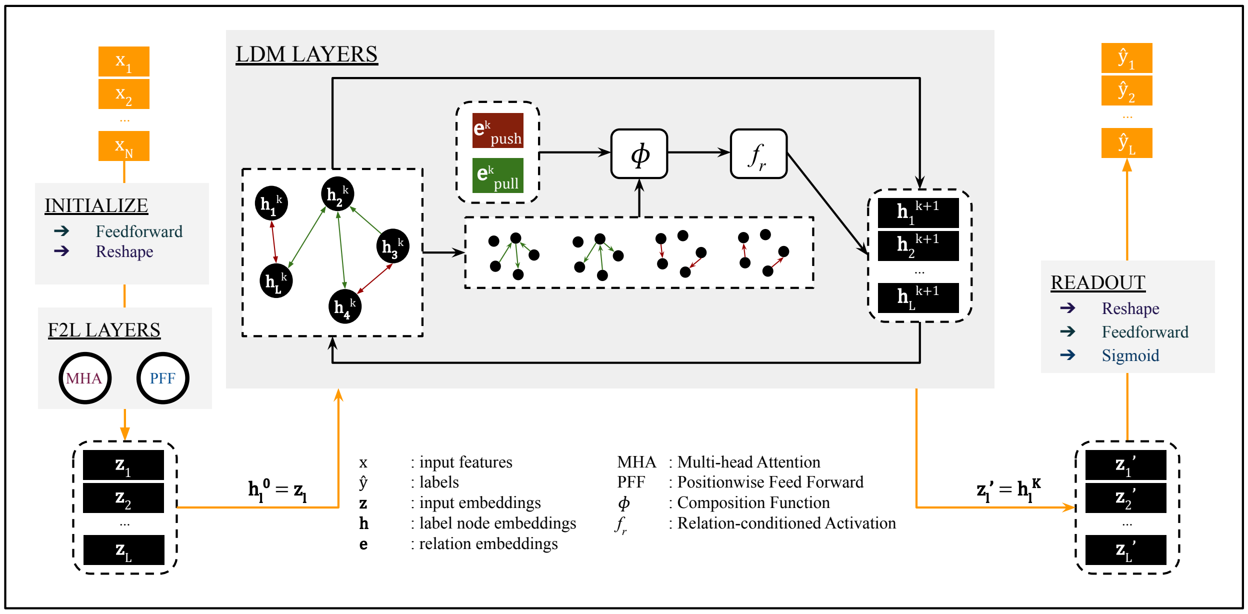


Figure 1: **Overview of Relation Guided Message Passing.** The input features $\mathbf{x}$ pass through embedding initialization and self attention layers of MHA (Multi-Head Attention) and a PFF (Position-wise Feed Forward network) to obtain the input embeddings $\mathbf{z}$. These are used as input label node embeddings $\mathbf{h}$. The Label Dependency Module (LDM) layers update the embeddings by relation-based composition, direction-specific neighbourhood aggregation and relation-conditioned activation by **pulling** and **pushing** relation sets. The label predictions $\hat{\mathbf{y}}$ are obtained by reshaping the embeddings, and then passing them through a final feedforward layer and a sigmoid to normalize.

4.1 Feature to Label Transformation by Transformer

Although the proposed label dependency module is suitable to use with most embedding based methods, we employ a Transformer (Vaswani et al., 2017) based F2L transformation for our experiments. It initializes input specific label embeddings $\mathbf{z} \in \mathbb{R}^{L \times d_{\text{emb}}}$ by input features $\mathbf{x} \in \mathbb{R}^N$ as follows:

$$\mathbf{z} = \text{Reshape}\left(\text{ReLU}\left(\mathbf{x}\mathbf{W}_{\text{init}} + \mathbf{b}_{\text{init}}\right)\right), \tag{3}$$

where $\mathbf{W}_{\text{init}} \in \mathbb{R}^{N \times d_{\text{init}}}$ and $\mathbf{b}_{\text{init}} \in \mathbb{R}^{d_{\text{init}}}$ are learnable parameters and $d_{\text{init}} = L \times d_{\text{emb}}$. After initializing the label embeddings, F2L stacks several layers of self attention to extract the most relevant features for each label. The main building blocks of the layers are Multi-head Attention (MHA) and a Position-wise Feed Forward (PFF) network.

Multi-head Attention (MHA) MHA is performed by concatenating parallel heads of scaled dot-product attention. The main motivation is to prevent destabilization during training by bad initialization of learnable weights. Let n_{head} be the number of attention heads, and define the head dimension as $d_{\text{head}} = d_{\text{emb}}/n_{\text{head}}$. We first obtain disjoint representations $\mathbf{q}^k \in \mathbb{R}^{L \times d_{\text{head}}}$ of the input for each head k by splitting the input embedding $\mathbf{z} \in \mathbb{R}^{L \times d_{\text{emb}}}$ in the embedding dimension. Afterwards, scaled-dot product attention is applied at each head to evaluate:

$$\text{Softmax}\left(\frac{\left(\mathbf{q}^k \mathbf{W}^k_{\text{queries}}\right)\left(\mathbf{q}^k \mathbf{W}^k_{\text{keys}}\right)^{\mathrm{T}}}{\sqrt{d_{\text{head}}}}\right)\left(\mathbf{q}^k \mathbf{W}^k_{\text{values}}\right), \tag{4}$$

where $\mathbf{W}^k_{\text{queries}} \in \mathbb{R}^{d_{\text{head}} \times L}$, $\mathbf{W}^k_{\text{keys}} \in \mathbb{R}^{d_{\text{head}} \times L}$, $\mathbf{W}^k_{\text{values}} \in \mathbb{R}^{d_{\text{heat}} \times L}$ are learnable parameters (the subscripts follow those in Vaswani et al. (2017)). Finally, the output is reconstructed by aggregating multiple head representations back to the original dimension: $\mathbf{z}' = \text{Concat}\left(\mathbf{q}^1, \ldots, \mathbf{q}^{n_{\text{head}}}\right)$.

Positionwise Feed Forward (PFF) Network PFF networks are fully connected feed-forward network layers which consist of two linear transformations, with a ReLU activation between them. Let d_{hid} represent the hidden dimension. For the input embeddings $\mathbf{z} \in \mathbb{R}^{L \times d_{\text{emb}}}$, PFF corresponds to the following operation:

$$\text{ReLU}\left(\mathbf{z}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{hid}}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{\text{hid}}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{emb}}}$ and $\mathbf{b}_2 \in \mathbb{R}^{d_{\text{emb}}}$ are learnable parameters.

4.2 Label Dependency Modelling

The idea of considering multiple label relations to model label dependencies for multi-label classification problem was introduced relatively recently (Ozmen et al., 2022). Modeling more complicated dependencies between the labels, beyond simple assessments of correlation, has the potential to boost prediction performative. This motivates us to design a label dependency module that can handle multiple types of relations. We aim to ensure that the module can be easily adapted to any embedding based method. We now describe the core components of the Label Dependency Module.

Label Relation Graph Let $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ be an undirected relation graph where $\mathcal{V}$ denotes the set of nodes, $\mathcal{R}$ denotes the set of relationships and the edges are defined as tuples $(u, v, r) \in \mathcal{E}$ indicating the presence of relation r between nodes u and v. Each relation r can be represented by an individual adjacency matrix defined as follows:

$$\mathbf{A}^r_{uv} = \begin{cases} 1, & \text{if } (u, v, r) \in \mathcal{E}\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{6}$$

We consider a label relation graph $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ where the labels are represented as nodes, i.e., $\mathcal{V} = \{1, \ldots, L\}$. Although our module can process multiple types of relations, in this paper, for concreteness, we focus on the case where there are two types of statistical relations between labels: *pull* and *push*, i.e., $\mathcal{R} = \{\text{pull}, \text{push}\}$.

Building Graphs Th phi-coefficient (Matthews, 1975) is a measure of the association between two binary variables. In order to identify a set of pulling and pushing edges, we (1) calculate the phi-coefficient ρ_{uv} for each label pair $u, v \in \mathcal{V}$ based on the training data; (2) set positive and negative edge thresholds τ_{pull} and τ_{push} based on the distribution of calculated statistics over all label pairs; and (3) obtain the adjacency matrices $\mathbf{A}^{\text{pull}}$ and $\mathbf{A}^{\text{push}}$ by masking pairwise correlations with corresponding edge thresholds. The details of this procedure are provided in Appendix A.

Layer Updates Note that the estimated adjacency matrices are symmetric as the test statistic is symmetric. Inspired by the direction-specific weights concept of CompGCN (Vashishth et al., 2020), we introduce into our model the flexibility to learn directional relationships by decomposing the undirected graph into two directed graphs defined by the lower and upper triangular portions of the original adjacency matrix:

$$\mathbf{A}^{r,\text{lower}} = [\mathbf{A}^r_{uv} \text{ for } u > v, 0 \text{ for } u \leq v], \tag{7}$$
$$\mathbf{A}^{r,\text{upper}} = [\mathbf{A}^r_{uv} \text{ for } u < v, 0 \text{ for } u \geq v]. \tag{8}$$

In this construction, the ordering of the nodes is arbitrary but assumed fixed. Directional neighbourhoods give the model the flexibility to learn different strengths of relationship in different directions.

Denote the hidden states of the label embeddings at layer l by $\mathbf{H}^{(l)} \in \mathbb{R}^{L \times d_{\text{emb}}}$, with $\mathbf{H}^{(0)} = \mathbf{z}$. RGMP learns a d_{emb}-dimensional representation $\mathbf{e}_r^{(l)} \in \mathbb{R}^{d_{\text{emb}}}$ for each relation $r \in \mathcal{R}$ while updating the label embeddings as follows:

$$\mathbf{H}^{(l+1)} = \mathbf{H}^{(l)}\mathbf{W}^{(l)}_{\text{self}} + \sum_{r \in \mathcal{R}} \sum_{\lambda} f_r \left(\mathbf{A}^{r,\lambda} \phi \left(\mathbf{H}^{(l)}, \mathbf{e}_r^{(l)} \right) \mathbf{W}^{(l)}_{\lambda} \right), \tag{9}$$

where $\mathbf{W}^{(l)}_{\text{self}} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{emb}}}$ are self-loop trainable weights, and $\mathbf{W}^{(l)}_{\lambda} \in \mathbb{R}^{d_{\text{emb}} \times d_{\text{emb}}}$ are direction specific trainable weights at layer l for $\lambda \in \{\text{lower}, \text{upper}\}$. The composition function $\phi(\cdot, \cdot)$ combines the label hidden state and the relation hidden state. In our case, since we also have an intuitive understanding of the nature of the two relations, we use a relation-conditioned activation function $f_r(\cdot)$ such that $f_{\text{pull}}(x) = \max(0, x)$ and $f_{\text{push}}(x) = \min(0, x)$. The relation embeddings are initialized randomly from a uniform distribution and updated at each layer by relation specific trainable parameters, $\mathbf{e}_r^{(l+1)} = \mathbf{W}^{(l)}_{\text{rel}} \mathbf{e}_r^{(l)}$.

4.3 Readout

Finally, in order to calculate the predicted label likelihoods, the label embeddings are flattened back to a vector of dimension $d_{\text{init}} = d_L \times d_{\text{emb}}$ and then scaled by a sigmoid after application of a linear projection:

$$\hat{\mathbf{y}} = \text{Sigmoid} \left(\text{Reshape} \left(\mathbf{z} \right) \mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}} \right), \tag{10}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d_{\text{init}} \times L}$ and $\mathbf{b}_{\text{out}} \in \mathbb{R}^{L}$ are learnable parameters.

5 Experiments

In multi-label classification, there have been numerous approaches that aim to improve feature to label mapping (F2L) in different feature domains and many other endeavours that attempt to model label dependencies (LDM) in order to improve prediction performance. However, these two lines of research have generally advanced independently of one another. Therefore, it is very important for the label dependency modelling to be adaptable to accommodate different network flows without disrupting the learning of the feature to label mapping (e.g., by introducing complicated loss function formulations) and without significantly increasing the model size. In order to understand if RGMP achieves these objectives we perform a series of analyses:

1. *Principle Comparison.* We first test if RGMP achieves competitive performance compared to the reported performance of the state-of-the-art models in terms of most commonly reported multi-label classification performance metrics. For these experiments, we use the original data splits for the datasets we study.

2. *Reproduction Study.* We reproduce the results of the most successful baseline algorithm and evaluate it using additional performance metrics. These extend beyond the conventional metrics and allow us to assess other important aspects of prediction performance. We repeat these experiments for multiple random data splits (i.e., train-validation-test sets) to obtain a statistical comparison of experimental results between the most competitive baseline and our proposed method.

Table 1: **Dataset Statistics.** $|\mathcal{D}|$ stands for total number of samples, $|\mathcal{D}_{\text{Train}}|$ for number of training samples, $|\mathcal{D}_{\text{Validation}}|$ for number of validation samples, $|\mathcal{D}_{\text{Test}}|$ for number of test samples, N for number of features, L for number of labels, $\bar{L}$ for average number of labels per sample, $\bar{D}$ for average number of samples per label.

Dataset	Domain	$\|\mathcal{D}\|$	$\|\mathcal{D}_{\text{Train}}\|$	$\|\mathcal{D}_{\text{Validation}}\|$	$\|\mathcal{D}_{\text{Test}}\|$	N	L	$\bar{L}$	$\bar{D}$
Bibtex	Text	7,379	4,377	487	2,515	1,836	159	2.38	66
Bookmarks	Text	87,856	48,000	12,000	27,856	2,150	208	2.03	467
eBird	Ecology	50,384	40,298	5,049	5,037	15	100	20.65	8,323
Fish	Ecology	61,996	44,637	11,159	6,200	46	12	3.05	11,368
Scene	Image	2,405	969	241	1,195	294	6	1.07	171
Sider	Biology	1,427	1,141	143	143	38	27	15.30	649
Yeast	Biology	2,417	1,200	300	917	103	14	4.24	363

3. *Ablation on LDM.* We perform an ablation study with and without the label dependency module to clarify the benefits of label dependency modelling. This allows us to show that the Transformer based F2L transformation we use in our experiments is competitive with most baselines when used alone. The study establishes that the performance improvement is not caused by introducing additional learnable parameters but rather by the successful modelling of label dependencies.

5.1 Datasets

We have experimented on 7 benchmark multi-label classification datasets from different instance domains. *Bibtex* and *Bookmarks* (Katakis et al., 2008) involve automated tag suggestion for entries from the BibSonomy social publication and the Bookmark sharing system, respectively. *eBird* (Munson et al., 2010) is a bird presence-absence dataset formed by observations recorded by volunteers all over the world. *Fish* (Morley et al., 2018) is a fish distribution dataset collected by trawlers in the North Atlantic. *Scene* (Boutell et al., 2004) is an image domain dataset composed of tracked camera frames and tagged with the main landscape features. *Sider* (Kuhn et al., 2016) is a biology database that measures the side effects of drug molecules. *Yeast* (Nakai & Kanehisa, 1992) contains genes annotated with a subset of functional categories. Our experiments cover a range of dataset sizes between 1,427 and 87,856. The average subset cardinality ranges from 1.07 to 20.65. The number of labels ranges between 6 and 208; and the feature space dimension ranges between 15 and 2,150. The datasets are divided into training (80%), validation (10%) and testing (10%) if there is no original split. If the original split contains only train and test sets, we use 10% of the training data for validation. The dataset details are provided in Table 1.

5.2 Metrics

For the primary comparison, we use the most commonly employed MLC evaluation metrics. The instance-based performance metrics are example-based F1 score (ebF1), and Hamming Accuracy (HA). The label-based performance metrics are micro-averaged F1 score (miF1), and macro-averaged F1 score (maF1). The example-based F1 score is aggregated over samples and the macro-averaged F1 score over labels. The micro-averaged F1 score takes the average of the F1 score weighted by the contribution of each label, and thus takes class imbalance into account. In the case of imbalanced classes, maF1 focuses more on rare labels than miF1, as the former weights each of the classes equally.

In addition to these conventional metrics, we compare the most competitive method to RGMP in terms of subset accuracy (ACC), also known as *Exact Match Ratio*. This is a stricter performance measurement which measures the fraction of times that an algorithm identifies the correct subset of labels for an instance. In some settings, it is valuable if an algorithm can correctly capture the entire set of labels. For example, consider a website that allows users to shortlist products by filtering on page tags. Suppose that a user is searching for an apartment with a 'garden' but does not have a car so a 'garage' is neither required nor desired. The best outcome is to provide the user with a short list with each option including 'garden' but excluding 'garage'. In a multi-label classification framework this requires exact matching.

We also evaluate ranking-based precision at k (P@k) metrics. P@k measures the number of relevant labels among the top k labels with the highest estimated likelihoods. Ranking-based metrics eliminate the need to convert estimated likelihoods to hard label-based decisions. While the most commonly used metric is P@1, precision at higher ranks provides insight into the performance for less obvious labels. The mathematical expressions for the evaluation metrics are provided in Appendix B.

Table 2: **Principle Comparison Results.** For each dataset and evaluation metric, the best performing algorithm is in bold and the second best is underlined.

	ebF1							miF1						
	Bibtex	*Bookmarks*	*eBird*	*Fish*	*Scene*	*Sider*	*Yeast*	*Bibtex*	*Bookmarks*	*eBird*	*Fish*	*Scene*	*Sider*	*Yeast*
MLKNN	0.183	0.213	0.510	0.764	0.691	0.738	0.618	0.178	0.181	0.557	0.735	0.667	0.772	0.625
HARAM	0.353	0.216	0.510	0.507	0.717	0.722	0.629	0.365	0.230	0.573	0.518	0.693	0.754	0.635
SLEEC	0.449	0.363	0.258	0.779	0.718	0.581	0.643	0.407	0.300	0.412	0.756	0.699	0.697	0.653
C2AE	0.335	0.309	0.501	0.765	0.698	0.768	0.614	0.388	0.316	0.546	0.739	0.713	0.798	0.626
LaMP	0.447	0.389	0.477	0.784	0.728	0.766	0.624	0.473	0.373	0.517	0.760	0.716	0.797	0.641
MPVAE	<u>0.453</u>	0.382	0.551	<u>0.788</u>	0.751	<u>0.769</u>	0.648	<u>0.480</u>	0.375	0.593	<u>0.765</u>	0.742	<u>0.800</u>	0.655
ASL	-	0.373	0.528	-	<u>0.770</u>	0.752	0.613	-	0.354	0.580	-	<u>0.753</u>	0.795	0.637
C-GMVAE	0.428	<u>0.392</u>	**0.576**	0.784	**0.777**	**0.771**	**0.656**	0.458	<u>0.377</u>	**0.633**	0.762	**0.762**	**0.803**	**0.665**
RGMP	**0.455**	**0.397**	<u>0.548</u>	**0.796**	0.751	0.768	<u>0.655</u>	**0.492**	**0.387**	<u>0.605</u>	**0.772**	0.743	0.795	<u>0.664</u>
std	0.003	0.005	0.004	0.003	0.006	0.001	0.003	0.002	0.006	0.005	0.002	0.006	0.001	0.002
	HA							maF1						
	Bibtex	*Bookmarks*	*eBird*	*Fish*	*Scene*	*Sider*	*Yeast*	*Bibtex*	*Bookmarks*	*eBird*	*Fish*	*Scene*	*Sider*	*Yeast*
MLKNN	0.985	<u>0.991</u>	0.827	0.883	0.863	0.715	0.784	0.073	0.041	0.338	0.638	0.693	0.667	0.472
HARAM	0.986	0.990	0.819	0.671	0.902	0.650	0.744	0.227	0.140	0.474	0.427	0.713	0.649	0.448
SLEEC	0.982	0.989	0.816	0.891	0.894	0.675	0.782	0.294	0.195	0.363	0.657	0.699	0.592	0.425
C2AE	<u>0.987</u>	<u>0.991</u>	0.771	0.884	0.893	0.749	0.764	0.268	0.232	0.426	0.647	0.728	0.667	0.427
LaMP	**0.988**	**0.992**	0.811	0.888	0.903	0.751	0.786	0.376	0.286	0.381	0.687	0.745	0.668	0.480
MPVAE	**0.988**	<u>0.991</u>	0.829	<u>0.891</u>	0.909	0.755	0.792	<u>0.386</u>	0.285	<u>0.494</u>	<u>0.693</u>	0.750	<u>0.690</u>	0.482
ASL	-	<u>0.991</u>	0.831	-	<u>0.912</u>	<u>0.759</u>	<u>0.796</u>	-	0.264	0.467	-	<u>0.765</u>	0.668	0.484
C-GMVAE	<u>0.987</u>	**0.992**	**0.847**	0.889	**0.915**	**0.767**	<u>0.796</u>	0.350	<u>0.291</u>	**0.538**	0.687	**0.769**	**0.691**	<u>0.487</u>
RGMP	**0.988**	**0.992**	<u>0.838</u>	**0.893**	<u>0.912</u>	0.747	**0.801**	**0.391**	**0.307**	0.503	**0.695**	0.753	0.664	**0.488**
std	0.000	0.000	0.001	0.001	0.002	0.001	0.002	0.007	0.005	0.006	0.001	0.006	0.000	0.005

The reported results for our algorithm are averages and standard deviations calculated over 5 runs with different random initialization after the hyperparameter tuning

5.3 Baselines

We compare to the following methods:

- C2AE (Yeh et al., 2017) uses deep canonical correlation analysis and an autoencoder to learn a feature-aware latent subspace for label embeddings.
- LaMP (Lanchantin et al., 2020) performs attention-based neural message passing in order to encode the correlations among labels.
- ASL (Ben-Baruch et al., 2021) uses an asymmetric loss for multi-label classification which is derived as a combination of binary cross entropy and focal loss.
- MPVAE (Bai et al., 2021) is an autoencoder that use multivariate probit model for learning label correlations and strives to align the probabilistic feature and label subspaces.
- C-GMVAE (Bai et al., 2022) employs contrastive loss along with a Gaussian mixture variational autoencoder to learn a multimodal prior.
- HARAM (Benites & Sapozhnikova, 2015) use Adaptive Resonance Theory (ART) based clustering and Bayesian inference to calculate label probabilities.
- ML-KNN (Zhang & Zhou, 2007) finds the nearest examples to a test class using the k-Nearest Neighbors algorithm and then selects assigned labels using Bayesian inference.

5.4 Implementation

Model For all datasets, the number of self attention layers in the feature to label transformation is set to 2, and the number of label dependency update layers is set to 3. The embedding dimension d_{emb} is selected from $[64, 128]$, the latent model dimensionality d_{hid} is set to $2d_{\text{emb}}$, and the number of attention heads n_{head} is set to 4.

The distribution of phi-coefficients over all possible label pairs is highly skewed. In order to select the label pairs that have statistically significant relationships based on the training data, we apply positive and negative edge thresholds on the phi-coefficients τ_{pull} and τ_{push}. These thresholds are set according to the percentiles of the empirical distribution. For each dataset, it is selected from the following list of percentile pairs $[(5\%, 95\%), (10\%, 90\%), (30\%, 70\%), (30\%, 99\%), (1\%, 70\%)]$.

Training For all datasets, the maximum number of training epochs is set to 50 and early stopping is applied if none of the performance metrics, evaluated on the validation data, improves by 0.01% in 5 consecutive epochs. The model is trained with a batch size of 128, and the Adam (Kingma & Ba, 2015) optimizer is used to compute gradients and update parameters. The initial learning rate is tuned using grid search within the range $[1 \times 10^{-4}, 3 \times 10^{-3}]$. The learning rate is updated using a cosine annealing schedule (Loshchilov & Hutter, 2017) that performs stochastic gradient descent with warm restarts on each parameter group. The performance metrics ebF1, miF1, maF1, HA and ACC require conversion of the predicted label likelihoods $\hat{\mathbf{y}} \in [0, 1]^L$ to class decisions $\tilde{\mathbf{y}} \in \{0, 1\}^L$. The value of the threshold used to convert soft prediction to predicted class is determined using the validation sets and optimized individually for all performance metrics. The ranking metrics do not require this procedure.

Regularization We perform dropout with probability 0.2 and layer normalization after each MHA, PFF, and LDM update. We set the L2 penalty term coefficient in the loss function to 10^{-5}.

Hardware Experiments were run on a device with a GPU of NVIDIA V100 Volta (32G HBM2 memory) and CPU of Intel Silver 4216 Cascade Lake @ 2.1GHz.

5.5 Results

We first report the results for the principle comparison, examining the performance of 9 algorithms across 7 datasets using 4 performance metrics. Table 2 compares the proposed method, RGMP, with the baselines in terms of the most commonly reported performance metrics: example-based F1 (ebF1), micro-averaged F1 (miF1), macro-averaged F1 (maF1) scores and Hamming Accuracy (HA) [1].

- RGMP yields better results than MLKNN and HARAM for all metrics since these two algorithms rely only on clustering of input word embeddings for mapping features to labels, while RGMP employs a message passing neural network where input word embeddings are mapped to label embeddings via multi-head attention. The superiority of RGMP to early embedding based methods SLEEC and C2AE reveal that RGMP performs better in terms of learning both feature and label embeddings.
- Comparison to LaMP reveals the true impact of the multi-relation approach in terms of capturing label dependencies. Out of all of the baselines, our model is most similar to LaMP in terms of the main building blocks of mapping features to labels. A key distinction is that RGMP employs multi-relation GCNs to model label dependencies instead of the decoder self attention in LaMP.
- The comparison with MPVAE shows that multi-head attention based mapping of input to output and multi-relation based label dependency modelling are more effective than using a multivariate probit model for the mapping and a global covariance matrix for dependency modelling.

[1] The results for MLKNN and HARAM are obtained using functions provided in the scikit-learn library (Pedregosa et al., 2011). The results for SLEEC (Bhatia et al., 2015a), C2AE (Yeh et al., 2017), and ASL (Ben-Baruch et al., 2021) are those reported in Bai et al. (2022). The results for LaMP (Lanchantin et al., 2020), MPVAE (Bai et al., 2021) and C-GMVAE (Bai et al., 2022) are taken from the respective original papers. For these three algorithms, in the cases where the metrics are not reported in the original paper, we provide the results obtained using our own implementations.

- The asymmetric loss function of ASL emphasizes positive labels during training, which does not enable the classifier to learn from the absence of labels for a given instance. While the assymmetric loss provides efficiency in terms of running time and model size, comparing the results of ASL to RGMP shows that the performance can be improved by considering the information contained in the absence of labels.
- While RGMP clearly outperforms most baselines, C-GMVAE achieves very similar performance for the F1 metrics and Hamming accuracy. C-GMVAE implicitly considers pushing-type relations via its contrastive loss. We investigate the performance of these two algorithms in more detail via a reproduction study.

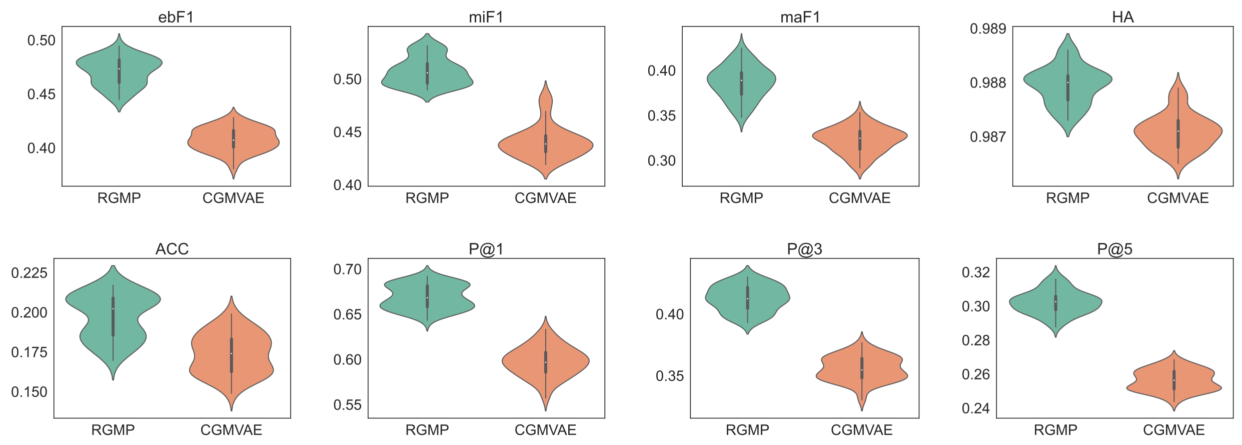


Figure 2: Comparison of RGMP and C-GMVAE by the reproduction study on **Bibtex** dataset.

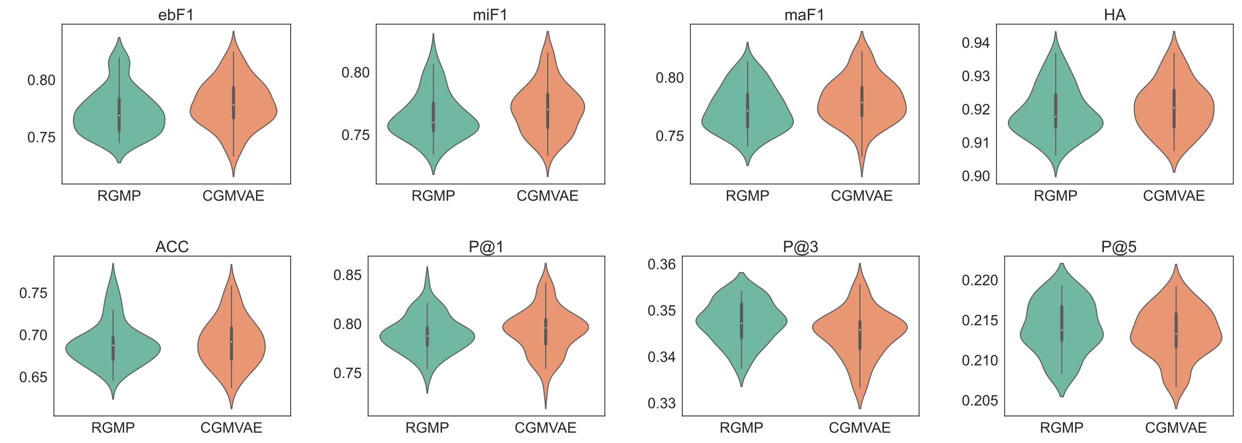


Figure 3: Comparison of RGMP and C-GMVAE by the reproduction study on **Scene** dataset.

Reproduction Study To provide a more fine-grained comparison between RGMP and C-GMVAE, we performed another set of experiments. We create 10 random splits of the training, validation, and test sets for the *Bibtex*, *Scene* and *Yeast* datasets and measure the performance of C-GMVAE and RGMP with 5 random initializations for each data partitioning. Figures 2 - 4 show violin plots of the performance metrics for the two classifiers. RGMP performs as well or better than C-GMVAE for all performance metrics on the Bibtex dataset. In fact, based on 95% confidence intervals computed using the student t-distribution, the upper critical value computed for C-GMVAE is smaller than the lower critical value computed for RGMP for most of the performance metrics (See Appendix E). We do not observe any dominance between the two classifiers on the Scene dataset. The average number of labels per sample on Scene is almost 1, and therefore

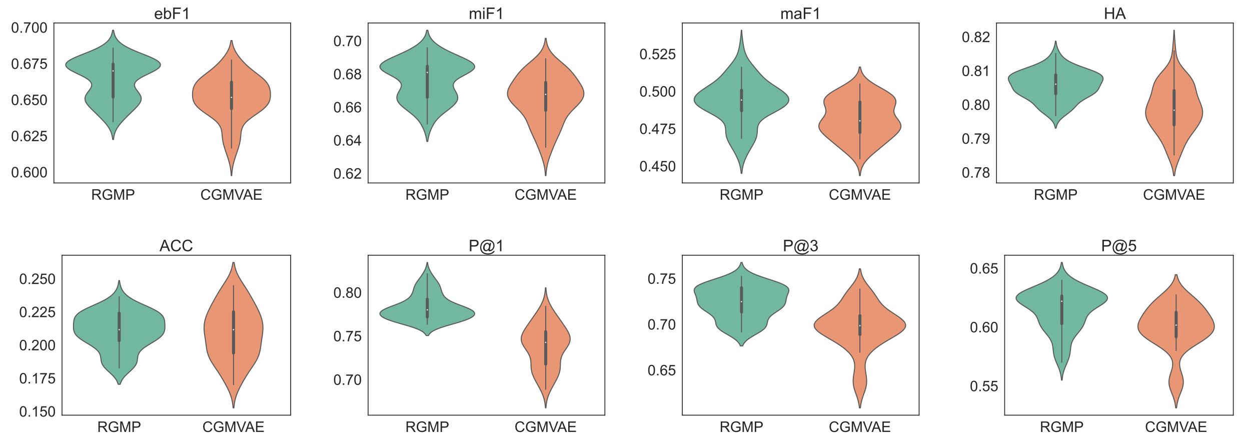


Figure 4: Comparison of RGMP and C-GMVAE by the reproduction study on **Yeast** dataset.

the problem is very similar to the multi-class classification setting and modelling label dependencies become less useful. RGMP outperforms C-GMVAE in terms of ranking metrics on the Yeast dataset. Even in the cases where its mean performance is only slightly better (ebF1, miF1, HA), RGMP generally exhibits less variability over trials.

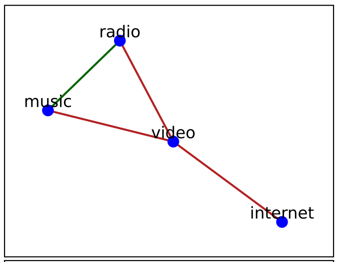
We also compare our algorithm to C-GMVAE for additional metrics. We report subset accuracy (ACC) and ranking metrics (p@k) for all datasets with the original data splits in Appendix C. Subset accuracy indicates the capability of capturing the exact label subset, and precision at higher ranks measures a model's ability to discover less obvious labels. RGMP performs considerably better than C-GMVAE on average. RGMP achieves better ACC for all datasets with an average improvement of 12.6%, better P@1 for 4 of the datasets with an average improvement of 4.8%, better P@3 for all datasets with an average improvement of 4.7%, and better P@5 for 5 of the datasets with an average improvement of 3.0%.

Ablation on LDM In order to examine the effectiveness of label dependency module, we perform an ablation study by assessing the performance of the proposed architecture with and without the LDM module. In Appendix D, the percent improvements obtained by LDM module are provided for each performance metric. The inclusion of the LDM module leads to improved performance for all metrics for every dataset. The usage of LDM improves HA by 0.5%, ebF1 by 2.4%, miF1 by 2.3%, maF1 by 3.2%, ACC by 76.7%, P@1 by 2.1%, P@3 by 1.5% and P@5 by 1.8%.

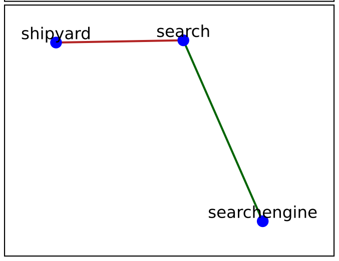
In Figure 5, we provide examples from the *Bookmarks* dataset to illustrate the impact of the proposed module. The first sample has ground truth labels 'radio' and 'music'. The induced label sub-graph shows that the true labels pull each other, but push the label 'video' at the first hop and push 'internet' at the second hop. Based on the predicted label likelihoods, it can be observed that the integration of the LDM module reinforces the belief concerning the relevance of 'music' and the irrelevance of 'video' and 'internet'. Indeed, after inclusion of LDM a threshold can be imposed to correctly predict that 'radio' is a label and 'video' is not. The second example shown in Figure 5 is tagged only with the label 'shipyard', which pushes 'search' at the first hop and 'searchengine' at the second hop. Accordingly, the LDM module amplifies the difference between the likelihoods of relevant and irrelevant labels. When determining the final labels, we need to decide upon a threshold, and therefore the changes in the predicted likelihoods are critical in expanding the range of thresholds that can correctly capture the full label set.

6 Conclusion

In this paper, we propose a Relation Guided Message Passing (RGMP) for the multi-label classification problem. The proposed model can be adapted to any number of relations as long as the adjacency matrix



Label	Without LDM	With LDM	Target
radio	0.209	0.201	1
music	0.664	0.864	1
internet	0.16	0.08	0
video	0.29	0.109	0



Label	Without LDM	With LDM	Target
search	0.182	0.022	0
searchengine	0.106	0.013	0
shipyard	0.651	0.785	1

Figure 5: **Ablation Study Samples. On left:** induced sub-graph graph of labels relevant to the sample with **pulling** and **pushing** edges. **On right:** predicted likelihood for each label with and without label dependency modelling and observed target values.

is known. We consider *pulling* and *pushing* statistical relations between labels and use the Composition-based GCN to learn label embeddings that reflect these statistical relations. The experiments show that the integration of the multi-relation module with pulling and pushing relations allows the architecture to emulate the advantage of making predictions sequentially in RNN based architectures, while enjoying the flexibility and computational efficiency of an MPNN.

References

Junwen Bai, Shufeng Kong, and Carla Gomes. Disentangled variational autoencoder based multi-label classification with covariance-aware multivariate probit model. In *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, pp. 4313–4321, 2021.

Junwen Bai, Shufeng Kong, and Carla P Gomes. Gaussian mixture variational autoencoder with contrastive learning for multi-label classification. In *Proc. Int. Conf. Machine Learning (ICML)*, pp. 1383–1398, 2022.

Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, pp. 82–91, 2021.

F. Benites and E. Sapozhnikova. Haram: A hierarchical aram neural network for large-scale text classification. In *Proc. IEEE Int. Conf. Data Mining Workshop*, pp. 847–854, 2015.

K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *Proc. Advances in neural information processing systems*, pp. 730–738, 2015a.

K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *Proc. Advances Neural Information Processing Systems (NeurIPS)*, pp. 730–738, 2015b.

Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

C. Chen, H. Wang, W. Liu, X. Zhao, T. Hu, and G. Chen. Two-stage label embedding via neural factorization machine for multi-label classification. In *Proc. AAAI Conf. Artificial Intelligence*, pp. 3304–3311, 2019a.

Z. Chen, X. Wei, P. Wang, and Y. Guo. Multi-label image recognition with graph convolutional networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 5172–5181, 2019b.

K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multi-label classification via probabilistic classifier chains. In *Proc. Int. Conf. Machine Learning (ICML)*, pp. 279–286, 2010.

Walter Gerych, Tom Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke A. Rundensteiner. Recurrent bayesian classifier chains for exact multi-label classification. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 15981–15992, 2021.

Jun-Yi Hang and Min-Ling Zhang. Collaborative learning of label semantics and deep label-specific features for multi-label classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2021.

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *Proc. ECML-PKDD Discovery Challenge*, 2008.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic Acids Research*, 44(D1):D1075–D1079, 2016.

J. Lanchantin, A. Sekhon, and Y. Qi. Neural message passing for multi-label classification. In *Proc. ECML-PKDD*, volume 11907, pp. 138–163, 2020.

Junlong Li, Peipei Li, Xuegang Hu, and Kui Yu. Learning common and label-specific features for multi-label classification with correlation information. *Pattern Recognition*, 121, 2022.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2017.

O. Luaces, J. Díez, J. Barranquero, J. del Coz, and A. Bahamonde. Binary relevance efficacy for multi-label classification. *Prog. Artificial Intelligence*, pp. 303–313, 2012.

J. Ma, B. Chi Y. Chiu, and T. W. S. Chow. Multi-label classification with group-based mapping: A framework with local feature selection and local label correlation. *IEEE Trans. Cybernetics*, pp. 1–15, 2020.

Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition*, 45(9):3084–3104, 2012.

Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

James W. Morley, Rebecca L. Selden, Robert J. Latour, Thomas L. Frolicher, Richard J. Sea-graves, and Malin L. Pinsky. Projecting shifts in thermal habitat for 686 species on the north american continental shelf. *PloS one*, 13(5):1–28, 2018.

M. Arthur Munson, Kevin Webb, Daniel Sheldon, Daniel Fink, Wesley M. Hochachka, Marshall Iliff, Mirek Riedewald, Daria Sorokina, Brian Sullivan, Christopher Wood, and Steve Kelling. The ebird reference dataset, version 2.0. *Cornell Lab of Ornithology and National Audubon Society, Ithaca, NY*, 2010.

Kenta Nakai and Minoru Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14(4):897–911, 1992.

J. Nam, E. Loza Mencía, H. J. Kim, and J. Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5413–5423, 2017.

M. Ozmen, H. Zhang, P. Wang, and M. Coates. Multi-relation message passing for multi-label text classification. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in python. *J. Machine Learning Research*, 12:2825–2830, 2011.

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *J. Machine Learning*, 85(3):333–359, 2011.

R. Senge, J. José del Coz, and E. Hüllermeier. Rectifying classifier chains for multi-label classification. *arXiv preprint arXiv:1906.02915*, 2019.

Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int. J. Data Warehousing and Mining*, 3:1–13, 2007.

S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. Composition-based multi-relational graph convolutional networks. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2020.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. Advances Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

Y. Wang, D. He, F. Li, X. Long, Z. Zhou, J. Ma, and S. Wen. Multi-label classification with label graph superimposing. *arXiv preprint arXiv:1911.09243*, 2019.

Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent spaces for multi-label classification. In *Proc. AAAI Conf. Artificial Intelligence*, pp. 2838–2844, 2017.

Ze-Bang Yu and Min-Ling Zhang. Multi-label classification with label-specific feature generation: A wrapped approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2021.

M. Zhang and Z. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40 (7):2038–2048, 2007.

Wenting Zhao, Shufeng Kong, Junwen Bai, Daniel Fink, and Carla P. Gomes. Hot-vae: Learning high-order label correlation for multi-label classification via attention-based variational autoencoders. In *Proc. AAAI Conf. Artificial Intelligence*, pp. 15016–15024, 2021.

Appendices

A Building Label Graphs

In this appendix, we present the procedure for building a label graph. The procedure involves two steps. We first calculate pairwise contingency tables from the training data and use them to evaluate the phi-coefficients. Subsequently, we derive the adjacency matrices of the graphs by applying thresholds to the derived phi-coefficients.

1. For each label pair $u, v \in= \{1, \ldots, L\}$;
   (a) Calculate the pairwise contingency table entries;
   $$c_{11} = \sum_{i \in \mathcal{D}_{\text{Train}}} y_{iu} y_{iv} \qquad c_{01} = \sum_{i \in \mathcal{D}_{\text{Train}}} (1 - y_{iu}) y_{iv}$$
   $$c_{10} = \sum_{i \in \mathcal{D}_{\text{Train}}} y_{iu} (1 - y_{iv}) \qquad c_{00} = \sum_{i \in \mathcal{D}_{\text{Train}}} (1 - y_{iu})(1 - y_{iv})$$
   (b) Calculate phi-coefficient;
   $$\rho_{uv} = \frac{c_{11}c_{00} - c_{01}c_{10}}{\sqrt{(c_{11} + c_{01})(c_{10} + c_{00})(c_{11} + c_{10})(c_{01} + c_{00})}}$$
2. Let τ_{pull} and τ_{push} be the positive and negative edge thresholds, the adjacency matrices are obtained by masking phi-coefficients with thresholds as follows;
   $$\mathbf{A}_{uv}^{\text{pull}} = \begin{cases} 1, & \text{if } \rho_{uv} \geq \tau_{\text{pull}}, \\ 0, & \text{otherwise}, \end{cases} \qquad \mathbf{A}_{uv}^{\text{push}} = \begin{cases} 1, & \text{if } \rho_{uv} \leq \tau_{\text{push}}, \\ 0, & \text{otherwise}. \end{cases}$$

B Evaluation Metrics

In this appendix, we present the formal definitions of the performance metrics used in our experiments.

Instance based metrics Given L labels and M samples, let $\mathbf{y}_i = (y_{i1}, \ldots y_{iL})$ and $\tilde{\mathbf{y}}_i = (\tilde{y}_{i1}, \ldots, \tilde{y}_{iL})$ denote binary vectors of ground-truth and predicted labels on sample i respectively. Subset accuracy is defined as:

$$\text{ACC} = \frac{1}{M} \sum_{i=1}^{M} I[\mathbf{y}_i = \tilde{\mathbf{y}}_i].$$

Hamming accuracy is defined as:

$$\text{HA} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{L} I[y_{ij} = \tilde{y}_{ij}].$$

Example-based F1 score is defined as:

$$\text{ebF1} = \frac{1}{M} \sum_{i=1}^{M} \frac{2 \sum_{j=1}^{L} y_{ij} \tilde{y}_{ij}}{\sum_{j=1}^{L} y_{ij} + \sum_{j=1}^{L} \tilde{y}_{ij}}.$$

Label based metrics Each label y_j is treated as a separate binary classification problem (each label j has its own confusion matrix of true-positives (tp_j), false-positives (fp_j), true-negatives (tn_j), false-negatives (fn_j).) Micro-averaged F1 score is defined as:

$$\text{miF1} = \frac{\sum_{j=1}^{L} 2tp_j}{\sum_{j=1}^{L} 2tp_j + fp_j + fn_j}.$$

Macro-averaged F1 score is defined as:

$$\text{maF1} = \frac{1}{L} \sum_{j=1}^{L} \frac{2tp_j}{2tp_j + fp_j + fn_j}.$$

Ranking based metrics Precision at k metrics are defined by predicted likelihood vector $\hat{\mathbf{y}} \in [0,1]^L$ and ground truth label vector $\mathbf{y} \in \{0,1\}^L$ as:

$$\text{P@}k = \frac{1}{k} \sum_{l \in \text{rank}_k(\hat{\mathbf{y}})} \mathbf{y}_l \,,$$

where $\text{rank}_k(\hat{\mathbf{y}})$ returns the k largest indices of $\hat{\mathbf{y}}$ ranked in descending order. For instance, P@k measures the number of relevant labels among the top k labels with highest estimated likelihoods.

C Reproduction Comparison on Complete Set of Datasets with Beyond Reported Metrics

In this appendix, subset accuracy (ACC) and ranking metrics (p@k) for all datasets with the original data splits obtained through reproduction of C-GMVAE and experiments of RGMP are provided.

Table 3: **Reproduction Study Results.** The results are averaged over 5 random initialization.

	ACC			*p@1*			*p@3*			*p@5*		
	C-GMVAE	*RGMP*	*std*	*C-GMVAE*	*RGMP*	*std*	*C-GMVAE*	*RGMP*	*std*	*C-GMVAE*	*RGMP*	*std*
Bibtex	0.180	**0.190**	0.000	0.600	**0.651**	0.004	0.368	**0.397**	0.002	0.268	**0.292**	0.002
Bookmarks	0.222	**0.253**	0.001	0.465	**0.487**	0.003	0.242	**0.271**	0.002	**0.197**	**0.197**	0.002
eBird	0.000	**0.001**	0.000	**0.825**	0.804	0.005	0.756	**0.760**	0.004	0.721	**0.726**	0.005
Fish	0.382	**0.398**	0.004	0.897	**0.924**	0.001	0.614	**0.622**	0.001	0.461	**0.465**	0.000
Scene	0.668	**0.669**	0.008	**0.788**	0.771	0.007	0.340	**0.347**	0.001	**0.217**	**0.217**	0.000
Sider	0.014	**0.021**	0.006	**0.962**	0.944	0.004	0.893	**0.921**	0.003	0.880	**0.891**	0.001
Yeast	0.204	**0.207**	0.006	0.751	**0.773**	0.006	0.677	**0.718**	0.004	0.586	**0.606**	0.002

D Ablation Study Results on Complete Set of Datasets

In this appendix, the percentage improvements obtained by the LDM module are provided for each performance metric.

Table 4: **Ablation Study Results.** The percentage changes are calculated as the averages of the differences between the evaluations with and without the LDM module over 5 random initializations.

	% HA	*% ebF1*	*% miF1*	*% maF1*	*% ACC*	*% p@1*	*% p@3*	*% p@5*
Bibtex	0.02	3.55	1.69	5.25	0.64	2.11	2.06	2.50
Bookmarks	0.01	4.20	5.02	6.30	1.69	1.93	2.65	4.51
eBird	0.92	4.16	3.95	6.50	300.00	2.39	3.19	3.44
Fish	0.43	1.09	1.13	0.36	4.05	0.76	0.50	0.26
Scene	0.46	1.32	1.80	1.90	5.28	1.54	0.09	0.09
Sider	0.55	0.52	0.20	0.18	200.00	3.85	0.25	0.16
Yeast	1.28	2.03	2.03	1.84	24.97	2.32	1.98	1.73
avg.	0.52	2.41	2.26	3.19	76.66	2.13	1.53	1.81

E Estimated Confidence Intervals of Performance Measurements

In this appendix, 95% confidence intervals computed using the student t-distribution over the performances for RGMP and C-GMVAE on Bibtex, Scene and Yeast datasets are provided.

Table 5: **Reproduction Study Results.** Estimated confidence intervals are calculated using t distribution critical values at 95% significance level over 50 samples for each dataset on each metric.

	Bibtex		*Scene*		*Yeast*	
	C-GMVAE	RGMP	C-GMVAE	RGMP	C-GMVAE	RGMP
HA	[0.986, 0.988]	[0.987, 0.989]	[0.906, 0.936]	[0.905, 0.934]	[0.785, 0.813]	[0.797, 0.814]
ebF1	[0.386, 0.429]	[0.445, 0.497]	[0.738, 0.820]	[0.732, 0.812]	[0.621, 0.682]	[0.636, 0.692]
miF1	[0.410, 0.471]	[0.481, 0.533]	[0.732, 0.810]	[0.728, 0.802]	[0.637, 0.693]	[0.652, 0.700]
maF1	[0.295, 0.351]	[0.350, 0.423]	[0.741, 0.820]	[0.736, 0.811]	[0.456, 0.507]	[0.463, 0.522]
ACC	[0.149, 0.199]	[0.169, 0.225]	[0.634, 0.749]	[0.639, 0.741]	[0.169, 0.249]	[0.183, 0.238]
P@1	[0.565, 0.630]	[0.642, 0.696]	[0.747, 0.838]	[0.752, 0.827]	[0.690, 0.787]	[0.753, 0.812]
P@3	[0.336, 0.376]	[0.392, 0.434]	[0.335, 0.355]	[0.338, 0.356]	[0.646, 0.746]	[0.689, 0.758]
P@5	[0.244, 0.269]	[0.289, 0.316]	[0.207, 0.220]	[0.208, 0.221]	[0.560, 0.639]	[0.580, 0.649]

F Computational Complexity

Let N represent the number of features, L represent the number of labels, R represent number of relations, and d_{emb} and d_{hid} represent the hidden dimension and inner model dimension, respectively. (We set $d_{\text{hid}} = 2 \times d_{\text{emb}}$ in our experiments). For each sample and training iteration the feature to label transformation by the transformer module has a complexity of $\mathcal{O}(N \cdot L \cdot d_{\text{emb}})$ for the embedding initialization, $\mathcal{O}(d_{\text{emb}}^2)$ per layer of multi-head self attention, and $\mathcal{O}(d_{\text{emb}} \cdot d_{\text{hid}})$ per layer of positionwise feed forward network. Denote the number of self attention layers by n_1. The feature to label (F2L) transformation module has a complexity of $\mathcal{O}(N \cdot L \cdot d_{\text{emb}} + n_1 \cdot (d_{\text{emb}}^2 + d_{\text{emb}} \cdot d_{\text{hid}}))$. The label dependency modelling module has a complexity of $\mathcal{O}(n_2 \cdot (d_{\text{emb}}^2 + r \cdot d_{\text{emb}} + R^2))$ when there are n_2 layers in the module.