TUM School of Computation, Information and Technology
Technische Universität München

Leveraging Representation Learning for Multi-Modal and Graph-Structured Data

Muhammad Umer Anwaar

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. habil. Gerhard Rigoll

Prüfende der Dissertation:

1. Priv.-Doz. Dr. rer. nat. Martin Kleinsteuber
2. Prof. Dr.-Ing. Eckehard Steinbach

Die Dissertation wurde am 20.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 23.01.2023 angenommen.

Muhammad Umer Anwaar. *Leveraging Representation Learning for Multi-Modal and Graph-Structured Data.* Dissertation, Technische Universität München, Munich, Germany, 2022.

To Sana: my beloved wife and Asiya: my beautiful mother. *Without their prayers and continuous support, I would not be able to achieve this.*

To Musa, Ali and Momin: my amazing and wondrous kids*. Without their continuous interruption, this thesis would have been completed much earlier.*

Acknowledgements

I would like to start by thanking my wife, Sana, for her continuous support and unconditional love. The encouraging conversations with her have always comforted me to keep pursuing my dreams. She kept me motivated whenever I was feeling down. The prayers and uplifting words of my mother, Asiya, have always been a blessing at every stage of my life. The joy and adventures with my siblings and friends have always been a source of motivation for me. Those experiences kept me on track and reassured that I can overcome the challenges faced on this path.

Next, my heartfelt thanks to my supervisor, Martin, for providing me with the excellent opportunity to pursue the doctoral degree. He granted me the freedom to explore different research topics. This enabled me to learn so much, not only in terms of how to conduct research but also in terms of many other professional traits. Our discussions have equipped me to approach a research problem more systematically while keeping both theoretical and practical aspects in mind. Without his relentless trust, I would not have been able to find my own path.

I would also like to thank my key collaborator, Rayyan Ahmed Khan, with whom I worked on several research ideas. Conducting research in close collaboration with him was not only very productive but also an absolute joy. In addition to that, I really loved and had fun working with my students, especially Dmytro Rybalko, Egor Labintcev and Zhihui Pan.

Finally, I would like to thank my colleagues and friends at Unite Services GmbH. Especially, Rudolf Sailer, Till Brychcy, Alan Schelten and Dominik Meyer for the fun and insightful conversations at lunch or during coffee breaks (pre-COVID times). Their company has always been welcoming and cheering.

München, May 2022 Muhammad Umer Anwaar

Abstract

This thesis leverages the tools in representation learning and applies them to specific problems involving graph-structured and multi-modal data.

In the first part, we deal with these graph-structured data problems: link prediction, community detection and node representation learning. Specifically, we focus on variational graph autoencoder (VGAE) and collaborative subgraphs (CSGs) for addressing these problems. Before delving into the applications, we first investigate an important theoretical problem in variational graph autoencoder (VGAE) model, i.e. *over-pruning*. We propose a solution to mitigate this issue. We illustrate the problem and effectiveness of our solution via experiments on three benchmark datasets. Our approach yields better performance than simple VGAE on the link prediction task. After that, we turn towards jointly learning the node representation and community detection. We argue that a single node representation is sufficient for learning both the representation of the node itself and its context. We propose a VGAE based model, J-ENC, for learning community-aware node embeddings in such a way that connected nodes are not only "closer" to each other but also share similar community assignments. Our experiments demonstrate that J-ENC not only outperforms the competitive approaches but is also computationally efficient than them. At the end, we employ a graph based approach to tackle the recommender system problem. We employ CSGs and metapaths to form metapath-aware subgraphs, for explicitly capturing sequential semantics in graph structures. We compare the performance of our approach with several baselines on different real-world datasets.

The second part of this thesis deals with composition of multi-modal data by leveraging deep metric learning approach. We consider only two modalities of data, i.e., image and text. First, we investigate the problem of retrieving images from a database based on a multi-modal (image-text) query. We propose ComposeAE, an autoencoder based approach to learn the composition of image and text query for retrieving images. We devise a rotational symmetry constraint on the optimization problem that enables learning better composed representations. Second, we focus on smart information retrieval systems which should not only be able to retrieve images of never seen objects based on multi-modal (image-text) query but also be able to learn the underlying primitive concepts (state-object) present in the images. Building on ComposeAE, we propose ContraNet, which leverages the rich semantics of the state-object to learn multimodal representation in a contrastive manner.

At the end, we combine the two parts of this thesis and employ VGAE for learning the composition of primitive concepts, i.e. objects and states and retrieving images based on a multi-modal (image-text) query. Our proposed method Compositional Variational Graph Autoencoder (CVGAE), learns a similarity metric in common embedding space via bi-directional contrastive loss between projected graph and image embeddings.

Contents

List of Figures

List of Tables

1 Introduction

Machine Learning (ML) methods rely on the choice and quality of data representation for performing well on a given task. In traditional ML algorithms, feature engineering has been employed to compensate for the inability of the algorithms to extract informative representation from the raw data. But designing hand-crafted features is labor-intensive and requires domain-specific knowledge. Thus, it greatly reduces the applicability of such ML algorithms in a variety of real-world problems.

Representation learning promises to get rid of such hand-crafted features [2]. It aims to automatically learn the relevant representation (features) from raw data and has been an area of active research in the past ten years. This has led to the development of enormous number of algorithms for learning good representation of the data. Some areas which have been revolutionised by this research are: Speech Recognition and Signal Processing [3, 4], Object Recognition [5], Recommendation Systems [6] and Natural Language Processing [7, 8].

Historically, autoencoder is an architecture which despite being conceptually simple has proven itself quite effective architecture for feature learning or dimensionality reduction. Autoencoder (AE) attempts to non-trivially copy its input to its output with an aim that it learns some useful properties of the data. The architecture consists of two parts, namely: *encoder* and *decoder*. The encoder passes the input data through a feedforward neural network. The encoded input information after the last hidden layer of encoder is commonly referred as "code" or latent representation and the output units in this hidden layer are called latent units. The task of the decoder is to try to reconstruct the input from the latent representation. Variational autoencoders (VAE) [9] treat the encoder and decoder as stochastic mappings rather than deterministic functions. That is, VAE assumes that input data is being generated from some prior distribution, e.g., gaussian distribution. It employs encoder to learn the parameters of the prior distribution. In VAE, the decoder reconstructs the input based on samples from the *learnt latent* distribution. Despite being elegantly simple approach, VAE obtains excellent results and is among the state-of-the-art approaches to generative modeling. There are following two competing objectives VAE is trying to achieve:

1. Learning a "good" latent representation of the input data. This ensures that the model is useful for tasks like dimensionality reduction, information retrieval, classification etc.

2. The latent representation of input data should follow the prior distribution. This is typically ensured via minimizing the Kullback-Leibler (KL) divergence between latent and prior distribution. This is extremely important for ensuring good generative ability of the model.

In recent times, the field of deep learning has seen a noticeable growth in the interest in graph-related problems. Major factors fueling this growth are: (1) increase in computational power, (2) the power of graphs to model complex relations between objects and (3) algorithmic developments in learning feature representations from "raw" data. Graphs are flexible data structures that model complex relationships among entities, i.e. data points as nodes and the relations between nodes via edges. A large number of real world problems can be represented in terms of graphs. Some prominent examples are protein-protein interactions [10], social and traffic network s[11, 12] and knowledge graphs [13] and economic activity flows [14]. In the first part of this thesis, we focus on the graph-structured data.

Taking advantage of this trend, Kipf and Welling have proposed Variational Graph Autoencoder (VGAE) [1], which is an extension of VAE for graph-structured data. VGAE is a generative model for learning latent representations of nodes and edges of a graph. Before delving into the applications of representation learning on graphs, we first study VGAE model in Chapter 2. We particularly investigate the *over-pruning* problem, which adversely affects learning diverse and interpretable latent representations (see Sec. 2.3). We explain the *over-pruning* phenomenon briefly. The generative model of VGAE has a set of independent stochastic latent variables. These variables are responsible for capturing various factors of variation in the data such that new samples can be generated based on these factors. These factors are learnt by encoder and encoded in the stochastic latent units. In practice, it has been widely observed that VGAE (and also VAE) converge to a solution in which a significant number of latent variables fail to capture any information about the input data and the corresponding stochastic latent units become inactive. This leads VAE (VGAE) model to learn a suboptimal generative model, whose capacity is limited due to small number of active latent units. This phenomenon is termed as *over-pruning* [15, 16, 17].

Several solutions have been proposed to tackle this problem. For instance, adding dropout can be a simple solution to achieve more active latent units. However, this solution adds redundancy rather than encoding more useful information with latent variables [18]. [19] proposes division of the latent units into subsets and forcing each subset to contribute to the Kullback-Leibler (KL) divergence between latent and prior distribution. [16] uses KL cost annealing to activate more latent units. [18] uses a model based approach where latent units are divided into subsets with only one subset penalized for a certain data point. These subsets also share some latent variables which helps in reducing the redundancy between different subsets.

To suppress this issue in VGAE, Kipf and Welling[1] adopted a solution which inadvertently sacrificed the generative ability of the model. That is, they harshly penalize the KL-divergence term (which is responsible for ensuring good generative ability) by dividing it with the number of nodes in the graph. We emphasize here that the KL-divergence term actually ensures that the latent variables follow the prior distribution. Reducing the weight of KL-divergence term adversely affects the generative ability of VGAE and effectively reduces it to non-variational graph autoencoder. We propose our solution and discuss it in detail in Chapter 2. Briefly, we attempt to improve the generative capability of VGAE via *epitomes*. *Epitomes* are groups of latent variables sharing the latent space, see Section 2.5 for details. Our model based approach, called epitomic VGAE (EVGAE), is a generative variational framework for graph datasets consist of multiple sparse VGAE models. This epitomic approach aids in increasing *active* units as epitomes compete to learn better representation of the graph data. This also enables other latent variables to be more free in encoding useful information for the node. We verify the effectiveness of our claims via experiments on three benchmark datasets, namely: Cora, Citeseer and Pubmed. Our experiments show that EVGAE has a better generative ability than VGAE, in terms of better matching of latent distribution with the prior distribution. Moreover, we also demonstrate that the representations learnt by EVGAE yield better performance than VGAE on the link prediction task. This chapter is based on the peer-reviewed publication [20].

Afterwards, in Chapter 3, we tend towards two very important tasks in graph analysis, namely: community detection and node representation learning. The objective in community detection is to cluster nodes of a graph into multiple groups (communities). Thus, each community is a set of densely connected nodes. The communities can be overlapping or non-overlapping, depending on whether they share some nodes or not. The second task, i.e., learning good node representation is quite useful for downstream tasks like graph visualization, recommender systems and classification. Traditionally, these tasks are usually treated separately. Early community detection algorithms are inspired from clustering algorithms [21]. For instance, spectral clustering [22] is applied to the graph Laplacian matrix for extracting the communities. Similarly, several matrix factorization based methods have been proposed to tackle the community detection problem. For example, Bigclam [23] treats the problem as a non-negative matrix factorization (NMF) task. Another method CESNA [24] extends Bigclam by modelling the interaction between the network structure and the node attributes. Some generative models, like vGraph [25], Circles [26] etc, have also been proposed to detect communities in a graph. Many successful algorithms which learn node representation in an unsupervised way are based on random walk objectives [27, 28, 29]. Some known issues with random-walk based methods (e.g. DeepWalk, node2vec etc) are: (1) They sacrifice the structural information of the graph by putting over-emphasis on the proximity information [30] and (2) great dependence of the performance on hyperparameters (walk-length, number of hops etc) [27, 28]. Some interesting GCN based approaches include graph autoencoders e.g. GAE and VGAE[1] and DGI[31].

In the literature, several attempts have been made to tackle both these tasks in a single framework. Most of these methods propose an alternate optimization process, i.e. learn node embeddings and improve community assignments with them and vice versa [32, 33]. Some approaches (CNRL [33], ComE [32]) are inspired from random walk, thus they inherit the issues discussed above. Others, like GEMSEC [34], are limited to the detection of non-overlapping communities. Some generative models like CommunityGAN [35] and vGraph [25] also jointly learn community assignments and node embeddings.

CNRL, ComE and vGraph require learning two embeddings for each node for simultaneously tackling the two tasks. Unlike them, our insight is to employ the VGAE model, introduced in chapter 2, for learning a single community-aware node representation which can be directly used for both tasks. We propose a joint generative model called

J-ENC for learning **J**oint **E**mbedding for **N**ode representation and **C**ommunity detection. **J-ENC** learns a community-aware node representation, i.e., learning of the node embeddings are constrained in such a way that connected nodes are not only "closer" to each other but also share similar community assignments. This joint learning framework leverages community-aware node embeddings for better performance on these tasks: node classification, overlapping community detection and non-overlapping community detection. In order to test the effectiveness of **J-ENC**, we have selected 18 different graph datasets ranging from 270 to 126,842 edges and 61 to 19793 nodes. Our experiments show that **J-ENC** effectively outperforms the competitive baselines on these tasks. We also investigate the robustness of our proposed approach with varying hyperparameters. Afterwards, we demonstrate that, due to our novel formulation of the problem, **J-ENC** is computationally efficient than its direct competitors. This chapter is based on the peer-reviewed publication [36].

After that, in chapter 4, we adopt graph based approach to investigate the problem of recommender systems. In graph neural networks, nodes' information from their direct neighbours is iteratively aggregated via message passing while neglecting the sequential nature of multi-hop node connections. Such sequential node connections e.g., metapaths, capture critical insights for tasks like recommender systems. Concretely, disregarding a larger neighbourhood and focusing only on the immediate neighbours leads to inadequate distillation of the collaborative signals for the recommendation. To tackle this, we employ *collaborative subgraphs (CSGs)*, i.e. a CSG is formed by focusing on a single user-item pair and is aimed to suppress the influence of feature nodes from other user-item interactions. Such local subgraphs contain rich semantic and collaborative information of user-item interactions. We employ collaborative subgraphs (CSGs) and metapaths to form metapath-aware subgraphs, which explicitly capture sequential semantics in graph structures. We propose meta**P**ath and **E**ntity-**A**ware **G**raph **N**eural **N**etwork (PEAGNN), which trains the graph neural network to perform metapath-aware information aggregation on such subgraphs. The information from different metapaths is then fused using attention mechanism. To leverage the local structure of CSGs, we introduce entity-awareness that acts as a contrastive regularizer on node embedding. Moreover, our proposed approach can be combined with prominent layers such as GAT [37], GCN [12] and GraphSage[38]. Our empirical evaluation shows that the proposed approach outperforms competitive baselines on three public datasets. We also conduct further analysis on PEAGNN to investigate whether it learns meaningful metapath combinations from a given set of metapaths. This chapter is based on a manuscript which is currently under-review in a peer-reviewed conference [39].

Next, we turn our attention towards the second part of our thesis, i.e., multi-modal data. Concretely, we only deal with two modalities, i.e., image and text. We focus on addressing the following two problems:

- Image Retrieval from a database based on a multi-modal query, i.e., the query is specified in the form of an image and natural language expressions describing the desired modifications in the query image. This task has applications in the domain of E-Commerce search, surveillance systems and internet search.

- Compositional Zero-Shot learning (CZSL) problem. The task here is to learn composition of primitive concepts, i.e. objects and states, from the images in such a way that even their novel compositions can be zero-shot classified. In contrast to image classification, the goal of CZSL classification is to simultaneously identify the class of the object and the state in which the object appears. Sometimes the visual differences of the same object in two states can be huge and that is where traditional classification methods fail. For example, "sliced cheese" and "molten cheese" can be visually very different from each other.

We address the first problem, i.e., image retrieval problem in chapter 5. We consider an advanced image retrieval system, which enables the users in expressing the concept in their mind by allowing a multi-modal (image-text) query. Such a retrieval system offers a natural and effective interface [40]. For instance, a user on an E-commerce platform can simply upload the image of a dress and request the system to find similar dresses but with some desired modifications like V-neck, sleeveless etc. In such setting the problem is to properly combine these two modalities, which despite being generated by naturally different processes, may still capture the same entities. While we can apply different metric learning techniques to compare and learn similarity between representations, it is still unclear how to get most benefit from both sources of information. We propose ComposeAE, an autoencoder based approach for composing the modalities in the multi-modal query. We adopt a novel approach and map these features to a complex space. We propose that the target image representation is an element-wise rotation of the representation of the source image in this complex space. The information about the degree of rotation is specified by the text features. We learn the composition of these complex vectors and their mapping to the target image space by adopting a deep metric learning (DML) approach. In this formulation, text features take a central role in defining the relationship between query image and target image. This also implies that the search space for learning the composition features is restricted. From a DML point of view, this restriction proves to be quite vital in learning a good similarity metric. We also propose an explicit rotational symmetry constraint on the optimization problem based on our novel formulation of composing the image and text features. We validate the effectiveness of our approach on three datasets: MIT-States, Fashion200k and Fashion IQ.

In chapter 6, we build upon ComposeAE model and address both the image retrieval and CZSL tasks jointly. Specifically, (1) we aim to learn such model which understand different states of an object and can recognise even unseen combinations of them. (2) The model should be able to retrieve images based on multi-modal (image-text) query, where the text describes the changes sought by the user in the query image. We propose a unified approach, ContraNet, which bridges the existing gap in these two tasks. ContraNet, an autoencoder based model, aims to predict a composition of multiple semantic concepts in images. We adopt a contrastive learning approach to learn embeddings which are visually grounded and semantically meaningful. We conduct several experiments to evaluate the performance of ContraNet on three benchmark datasets. We also conduct several ablation studies to quantify the contribution of different losses in the performance of our approach.

In the end, in chapter 7, we combine the two parts of this thesis quite nicely. That is, we employ our insights in VGAE and compositional learning of the concepts to tackle the CZSL task. We argue that objects and states are being generated from a prior distribution. They are treated as nodes of a graph and an edge between them indicates the existence of a compositional pair. This formulation enables us to learn the latent representation of our primitive concepts in a space of reduced dimensionality along with the feasibility of their compositions (edges). The embeddings of the compositional pairs are obtained by simply concatenating the respective state and object node embeddings. Such problem formulation enables us to not assume the knowledge of novel composition of concepts (objects and states) in the model output space at test time. We refer to this case, where set of test compositions is not known, as Open-World (OW) CZSL task. The other case when set of test compositions is known, is referred to as Close-World (CW) CZSL task. This chapter is based on a manuscript which is currently under-review in a peer-reviewed conference [41].

1.1 List of Publications

This thesis is built from the following publications:

1. **Anwaar, Muhammad Umer[*]**; Labintcev, Egor [*] and Kleinsteuber, Martin. Compositional Learning of Image-Text Query for Image Retrieval. In IEEE Winter Conference on Applications of Computer Vision (WACV '21). 2021.

2. **Anwaar, Muhammad Umer[*]**; Khan, Rayyan Ahmad [*]; Pan, Zhihui and Kleinsteuber, Martin. A Contrastive Learning Approach for Compositional Zero-Shot Learning. In ICMI '21: Proceedings of the 2021 International Conference on Multimodal Interaction. 2021.

3. **Anwaar, Muhammad Umer**; Pan, Zhihui and Kleinsteuber, Martin. On Leveraging Variational Graph Embeddings for Open World Compositional Zero-Shot Learning. In Proceedings of the 30th ACM International Conference on Multimedia (MM '22). Association for Computing Machinery, New York, NY, USA, 4645–4654.

4. Khan, Rayyan Ahmad [*]; **Anwaar, Muhammad Umer [*]** and Kleinsteuber, Martin. Epitomic Variational Graph Autoencoder (EVGAE). In 25th International Conference on Pattern Recognition (ICPR). 2020.

5. Khan, Rayyan Ahmad [*]; **Anwaar, Muhammad Umer [*]**; Kaddah, Omran, Zhiwei Han and Kleinsteuber, Martin. Unsupervised Learning of Joint Embeddings for Node Representation and Community Detection. In the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2021.

6. **Anwaar, Muhammad Umer[*]**; Han, Zhiwei[*]; Arumugaswamy, Shyam [*]; Khan, Rayyan Ahmad [*]; Weber, Thomas; Qiu, Tianming; Shen, Hao; Liu,

Yuanting and Kleinsteuber, Martin. On Leveraging the Metapath and Entity Aware Subgraphs for Recommendation. In Proceedings of the 1st Workshop on Multimedia Computing towards Fashion Recommendation (MCFR '22). 2022.

[*] indicates that the authors contributed equally to this work.

The following publications were published during the PhD, but are not made part of this thesis.

1. **Anwaar, Muhammad Umer**; Rybalko, Dmytro and Kleinsteuber, Martin. Mend The Learning Approach, Not the Data: Insights for Ranking E-Commerce Products . In the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2020.
2. S. Nawaz, A. Calefati, M. K. Janjua, **Muhammad Umer Anwaar** and I. Gallo, "Learning Fused Representations for Large-Scale Multimodal Classification," in IEEE Sensors Letters, vol. 3, no. 1, pp. 1-4, Jan. 2019.

Part I

Graph-Structured Data

2 Variational Graph Autoencoder and Over-pruning Problem

This chapter is based on the following peer-reviewed publication:

*Khan, Rayyan Ahmad [*]; **Anwaar, Muhammad Umer [*]** and Kleinsteuber, Martin. Epitomic Variational Graph Autoencoder (EVGAE). In 25th International Conference on Pattern Recognition (ICPR). 2020.*
[*] indicates that the authors contributed equally to this work.

2.1 Introduction

Graphs are data structures that model data points via nodes and the relations between nodes via edges. A large number of real world problems can be represented in terms of graphs. Some prominent examples are protein-protein interactions[10], social and traffic networks[11, 12] and knowledge graphs[13]. Deep learning applications related to graphs include but are not limited to link prediction, node classification, clustering [42, 43] and recommender systems[44, 45, 46].

Kipf and Welling [1] introduced variational graph autoencoder (VGAE) by extending the variational autoencoder (VAE) model [9]. Like VAE, VGAE tends to achieve the following two competing objectives:

1. An approximation of input data should be possible.
2. The latent representation of input data should follow standard gaussian distribution.

There is, however, a well-known issue with VAE in general: The latent units, which fail to capture enough information about the input data, are harshly suppressed during training. As a result the corresponding latent variables collapse to the prior distribution and end up simply generating standard gaussian noise. Consequently, in practice, the number of latent units, referred to as *active units*, actually contributing to reconstruction of the input data are quite low compared to the total available latent units. This phenomenon is referred to as *over-pruning* ([15, 16, 17]). Several solutions have been proposed to tackle this problem for VAEs . For instance, adding dropout, division of the latent units into subsets and forcing each subset to contribute to the KL divergence [19], KL cost annealing to activate more latent units [16] and adopting a model based approach and dividing the latent units into overlapping subsets [18].

VGAE, being an extension of VAE for graph datasets, is also susceptible to the over-pruning problem. This greatly reduces the modeling power of pure VGAE and undermines its ability to learn diverse and meaningful latent representations As demonstrated in detail in Sec. 2.3. To suppress this issue, the authors of [1] simply reduce the weight of the second objective by the number of nodes in training data. For instance, PubMed dataset[1] has ∼20k nodes, so the second objective is given 20,000 times less weight than the first objective. Surprisingly, this factor is not mentioned in their paper, although it is present in their code [48]. Since the second objective is the one enforcing standard gaussian distribution for the latent variables, reducing its weight adversely affects the generative ability of VGAE and effectively reduces it to non-variational graph autoencoder. We discuss this further in Sec. 2.4.

In this work, we refer to VGAE without any weighted objective as *pure VGAE* to distinguish it from VGAE[1]. In order to attain good generative ability and mitigate over-pruning, we adopt a model based approach called epitomic VGAE (EVGAE). Our approach is motivated by a solution proposed for tackling over-pruning problem in VAE [18]. We consider our model to consist of multiple sparse VGAE models, called *epitomes*, that share the latent space such that for every graph node only one epitome is forced to follow prior distribution. This results in a higher number of active units as epitomes compete to learn better representation of the graph data. Our main contributions are summarized below:

- We identify that VGAE[1] has poor generative ability due to the incorporation of weights in training objectives.
- We show that pure VGAE (without any weighted objectives) suffers from the over-pruning problem.
- We propose a true variational model EVGAE that not only achieves better generative ability than VGAE but also mitigates the over-pruning issue.

2.2 Pure Variational Graph Autoencoder

Given an undirected and unweighted graph $\mathcal{G}$ consisting of N nodes $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_N}\}$ with each node having F features. We assume that the information in nodes and edges can be jointly encoded in a D dimensional real vector space that we call latent space. We further assume that the respective latent variables $\{\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_N}\}$ follow standard gaussian distribution. These latent variables are stacked into a matrix $\boldsymbol{Z} \in \mathbb{R}^{N \times D}$. For reconstructing the input data, this matrix is then fed to the decoder network $p_\theta(\mathcal{G}|\boldsymbol{Z})$ parameterized by θ. The assumption on latent representation allows the trained model to generate new data, similar to the training data, by sampling from the prior distribution. Following VAE, the joint distribution can be written as

$$p(\mathcal{G}, \boldsymbol{Z}) = p(\boldsymbol{Z}) p_\theta(\mathcal{G}|\boldsymbol{Z}), \tag{2.1}$$

[1]PubMed is a citation dataset[47], widely used in deep learning for graph analysis. Details of the dataset are given in experiments Sec. 2.6.1

where

$$p(\boldsymbol{Z}) = \prod_{i=0}^{N} p(\boldsymbol{z}_i) \tag{2.2}$$

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{0}, \text{diag}(\boldsymbol{1})) \ \forall i. \tag{2.3}$$

For an unweighted and undirected graph $\mathcal{G}$, we follow [1] and restrict the decoder to reconstruct only edge information from the latent space. The edge information can be represented by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ where $\boldsymbol{A}[i,j]$ refers to the element in i^{th} row and j^{th} column. If an edge exists between node i and j, we have $\boldsymbol{A}[i,j] = 1$. Thus, the decoder is given by

$$p_\theta(\boldsymbol{A}|\boldsymbol{Z}) = \prod_{(i,j)=(1,1)}^{(N,N)} p_\theta(\boldsymbol{A}[i,j] = 1|\boldsymbol{z_i}, \boldsymbol{z_j}), \tag{2.4}$$

with

$$p_\theta(\boldsymbol{A}[i,j] = 1|\boldsymbol{z_i}, \boldsymbol{z_j}) = \sigma(< \boldsymbol{z_i}, \boldsymbol{z_j} >), \tag{2.5}$$

where $< . \ , \ . >$ denotes dot product and $\sigma(.)$ is the logistic sigmoid function.

The training objective should be such that the model is able to generate new data and recover graph information from the embeddings simultaneously. For this, we aim to learn the free parameters of our model such that the log probability of $\mathcal{G}$ is maximized i.e.

$$\begin{aligned} log\Big(p(\mathcal{G})\Big) &= log\Big(\int p(\boldsymbol{Z})p_\theta(\mathcal{G}|\boldsymbol{Z}) \ d\boldsymbol{Z}\Big) \\ &= log\Big(\int \frac{q_\phi(\boldsymbol{Z}|\mathcal{G})}{q_\phi(\boldsymbol{Z}|\mathcal{G})} p(\boldsymbol{Z})p_\theta(\mathcal{G}|\boldsymbol{Z}) \ d\boldsymbol{Z}\Big) \\ &= log\Big(\mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{\frac{p(\boldsymbol{Z})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\}\Big), \end{aligned} \tag{2.6}$$

where $q_\phi(\boldsymbol{Z}|\mathcal{G})$, parameterized by ϕ, models the recognition network for approximate posterior inference. It is given by

$$q_\phi(\boldsymbol{Z}|\mathcal{G}) = \prod_{i}^{N} q_\phi(\boldsymbol{z_i}|\mathcal{G}) \tag{2.7}$$

$$q_\phi(\boldsymbol{z_i}|\mathcal{G}) = \mathcal{N}\Big(\boldsymbol{\mu_i}(\mathcal{G}), \text{diag}(\boldsymbol{\sigma_i^2}(\mathcal{G}))\Big) \tag{2.8}$$

where $\boldsymbol{\mu}_i(.)$ and $\boldsymbol{\sigma}_i^2(.)$ are learnt using graph convolution networks (GCN) [12] and samples of $q_\phi(\boldsymbol{Z}|\mathcal{G})$ are obtained from mean and variance using the reparameterization trick [9].

In order to ensure computational tractability, we use Jensen's Inequality [49] to get ELBO bound of Eq. equation 7.8. i.e.

$$log\Big(p(\mathcal{G})\Big) \geq \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \tag{2.9}$$

$$= \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(p_\theta(\mathcal{G}|\boldsymbol{Z})\Big)\Big\} + \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \tag{2.10}$$

$$= -\text{BCE} - D_{KL}\Big(q_\phi(\boldsymbol{Z}|\mathcal{G})||p(\boldsymbol{Z})\Big) \tag{2.11}$$

where BCE denotes binary cross-entropy loss between input edges and the reconstructed edges. D_{KL} denotes the Kullback-Leibler (KL) divergence. By using equation 2.2, equation 2.3, equation 2.7 and equation 2.8, the loss function of pure VGAE can be formulated as negative of equation 7.11 i.e.

$$L = \text{BCE} + \sum_{i=1}^{N} D_{KL}\Big(\mathcal{N}\Big(\boldsymbol{\mu}_i(\mathcal{G}), \boldsymbol{\sigma}_i^2(\mathcal{G})\Big) \;||\; \mathcal{N}(\mathbf{0}, \text{diag}(\mathbf{1}))\Big) \tag{2.12}$$

2.3 Over-pruning in pure VGAE

Burda et al. [15] showed that the learning capacity of VAE is limited by *over-pruning*. Several other studies [16, 17, 19, 18] confirm this and propose different remedies for the over-pruning problem. They hold the KL-divergence term in the loss function of VAE responsible for over-pruning. This term forces the latent variables to follow standard gaussian distribution. Consequently, those variables which fail to encode *enough* information about input data are harshly penalized. In other words, if a latent variable is contributing little to the reconstruction, the variational loss is minimized easily by "turning off" the corresponding hidden unit. Subsequently, such variables simply collapse to the prior, i.e. generate standard gaussian noise. We refer to the hidden units contributing to the reconstruction as *active units* and the turned-off units as *inactive units*. The activity of a hidden unit u was quantified by Burda et al. [15] via the statistic

$$A_u = Cov_x(\mathbb{E}_{u\sim q(u|x)}\{u\}). \tag{2.13}$$

A hidden unit u is said to be *active* if $A_u \geq 10^{-2}$.

VGAE is an extension of VAE for graph data and loss functions of both models contain the KL-divergence term. Consequently, pure VGAE inherits the over-pruning issue. We verify this by training VGAE with Eq. equation 2.12 on Cora dataset[2]. We employ the same graph architecture as Kipf and Welling [1]. The mean and log-variance of 16-dimensional latent space are learnt via Graph Convolutional Networks[12]. From

[2]Details of Cora dataset are given in experiments Sec. 2.6.1

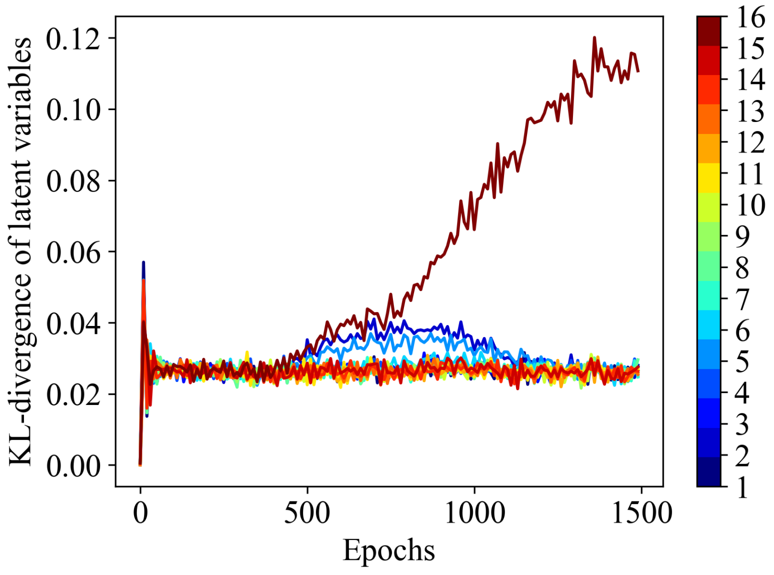

Figure 2.1: KL-divergence of latent variables in pure VGAE

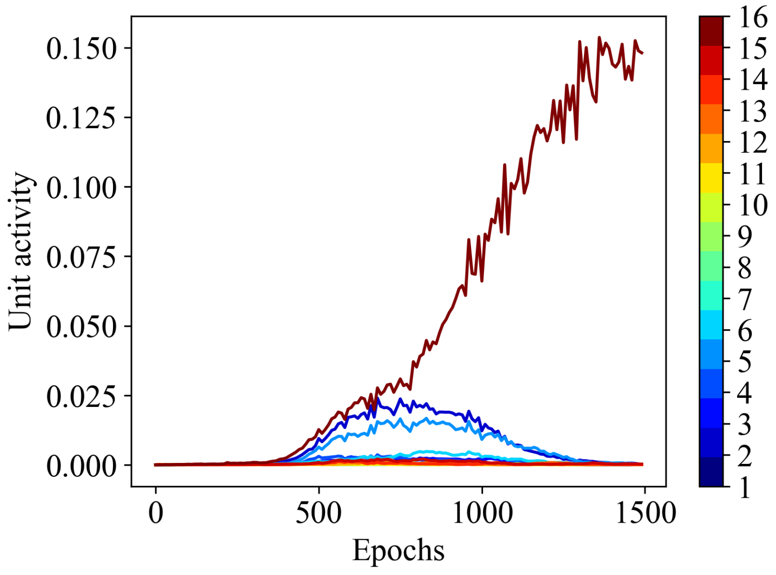

Figure 2.2: Unit activity of 16 hidden units in pure VGAE

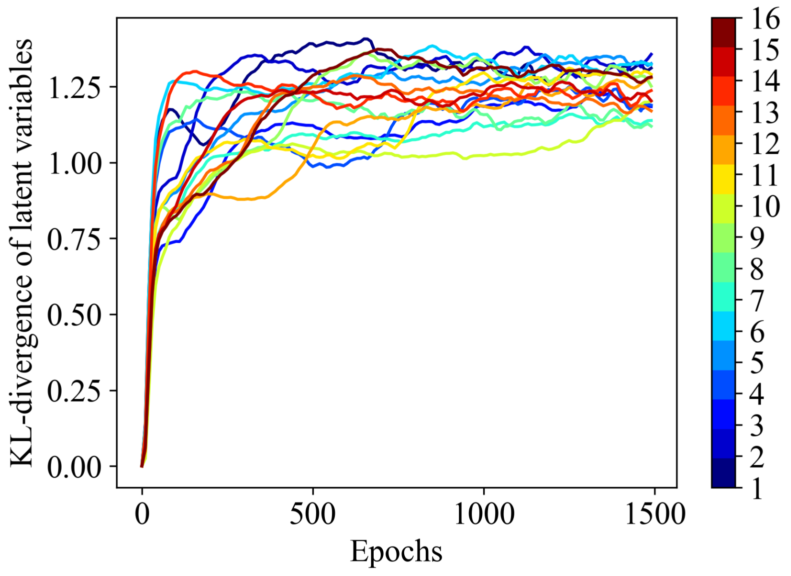

Figure 2.3: KL-divergence of latent variables: VGAE ($\beta \approx 0.0003$[1])

Fig. 2.1, we observe that 15 out of 16 latent variables have KL-divergence around 0.03, indicating that they are very closely matched with standard gaussian distribution. Only one latent variable has managed to diverge in order to encode the information required by the decoder for reconstruction of the input.

In other words pure VGAE model is using only one variable for encoding the input information while the rest 15 latent variables are not learning anything about the input. These 15 latent variables collapse to the prior distribution and are simply generating standard gaussian noise. Fig. 2.2 shows the activity of hidden units as defined in Eq. 2.13. It is clear that only one unit is *active*, which corresponds to the latent variable with highest KL-divergence in the Fig. 2.1. All other units have become *inactive* and are not contributing in learning the reconstruction of the input. This verifies the existence of *over-pruning* in pure VGAE model.

2.4 VGAE[1]: Sacrificing Generative Ability for Handling Over-pruning

Kipf and Welling's VGAE[1] employed a simple way to get around the over-pruning problem by adding a penalty factor to the KL-divergence in Eq. equation 2.12. That is

$$\mathcal{L} = BCE + \beta\ D_{KL}\Big(q(\boldsymbol{Z}|\mathcal{G})||p(\boldsymbol{Z})\Big). \tag{2.14}$$

But a consequence of using the penalty factor β is poor generative ability of VGAE. We verify this by training VGAE on Cora dataset with varying β in Eq. equation 2.14. We call the penalty factor β, as the loss of βVAE ([50, 51]) has the same factor multiplied with its KL-divergence term. Specifically, in βVAE, $\beta > 1$ is chosen to enforce better

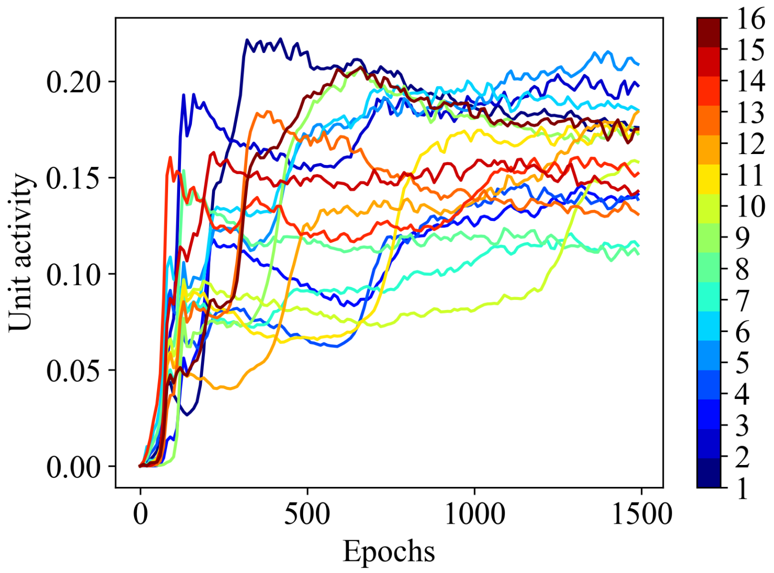

Figure 2.4: Unit activity of 16 hidden units: VGAE ($\beta \approx 0.0003$[1])

distribution matching. Conversely, a smaller β is selected for relaxing the distribution matching, i.e. the latent distribution is allowed to be more different than the prior distribution. This enables latent variables to learn better reconstruction at the expense of the generative ability. In the degenerate case, when $\beta = 0$, VGAE model is reduced to non-variational graph autoencoder (GAE). VGAE as proposed by Kipf and Welling[1] has the loss function similar to βVAE with β chosen as reciprocal of number of nodes in the graph. As a result β is quite small i.e. $\sim$ 0.0001-0.00001.

Fig. 2.3 shows the KL-divergence and hidden unit activity for original VGAE[1] model. We observe that all the hidden units are active, i.e. $A_u \geq 10^{-2}$. However, the value of KL-divergence is quite high for all latent variables, indicating poor matching of $q_\phi(\boldsymbol{Z}|\mathcal{G})$ with the prior distribution. This adversely affects the generative ability of the model. Concretely, the variational model is supposed to learn such a latent representation which follows standard gaussian (prior) distribution. Such high values of KL-divergence implies that the learnt distribution is not standard gaussian. The reason is that the KL-divergence term in equation 2.14 was responsible for ensuring that the posterior distribution being learned follows standard gaussian distribution. VGAE[1] model assigns too small weight ($\beta = 0.0003$) to the KL-divergence term. Consequently, when new samples are generated from standard gaussian distribution $p(\boldsymbol{Z})$ and then passed through the decoder $p_\theta(\boldsymbol{A}|\boldsymbol{Z})$, we get quite different output than the graph data used for training.

Fig. 2.4 shows that Kipf and Welling's[1] approach to deal with over-pruning makes VAGE similar to its non-variational counter-part i.e. graph autoencoder (GAE). As β is decreased, VGAE model learns to give up on the generative ability and behaves similar to GAE. This can be seen in Fig. 2.4 (a), where the average KL-divergence per active hidden unit increases drastically as β becomes smaller. On the other hand, we observe

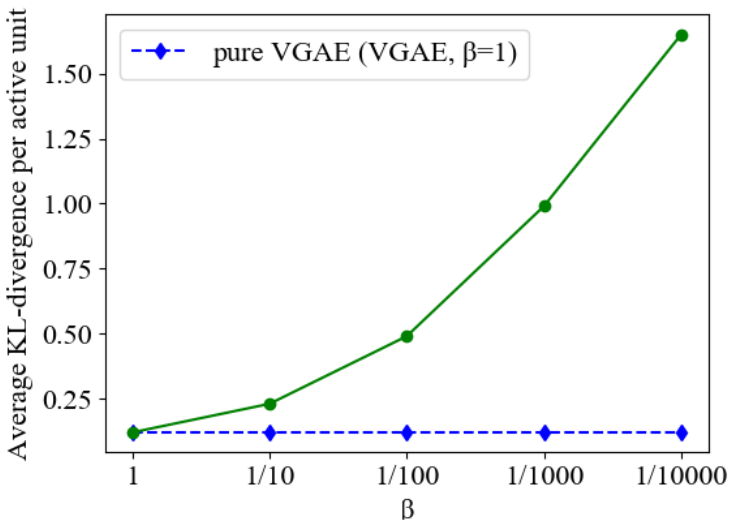

Figure 2.5: Effect of varying β on original VGAE[1]: Change in the active units of original VGAE[1]

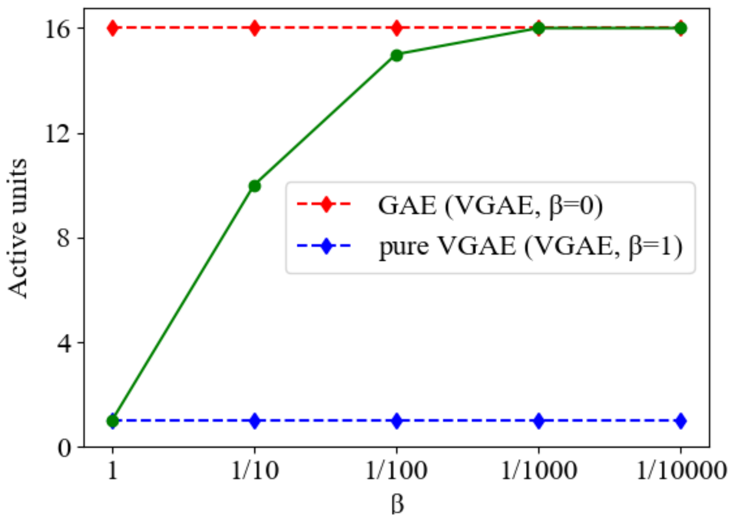

Figure 2.6: Effect of varying β on original VGAE[1]: Change in the Average KL-divergence per active unit

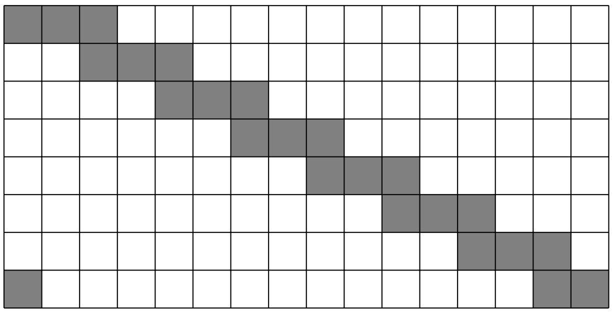

Figure 2.7: Example of eight epitomes in a 16-dimensional latent space.

from Fig. 2.4 (b) that decreasing β results in higher number of active hidden units till it achieves the same number as GAE.

We conclude that as the contribution of KL-divergence is penalized in the loss function (Eq. 2.14), VGAE model learns to sacrifice the generative ability for avoiding over-pruning. Conversely, VGAE handles the over-pruning problem by behaving like a non-variational model GAE [52].

2.5 Epitomic Variational Graph Autoencoder

We propose epitomic variational graph autoencoder (EVGAE) which generalizes and improves the VGAE model. EVGAE not only successfully mitigates the over-pruning issue of pure VGAE but also attains better generative ability than VGAE[1]. The motivation comes from the observation that for a certain graph node, a subset of the latent variables suffices to yield good reconstruction of edges. Yeung et al. [18] proposed a similar solution for tackling over-pruning problem in VAE. We assume M subsets of the latent variables called *epitomes*. They are denoted by $\{\mathcal{D}_1, \cdots, \mathcal{D}_M\}$. Furthermore, it is ensured that every subset shares some latent variables with at least one other subset. We penalize only one epitome for an input node. This encourages other epitomes to be active. Let y_i denote a discrete random variable that decides which epitome is penalized for a node i. For a given node, the prior distribution of y_i is assumed to be uniform over all the epitomes. $\boldsymbol{y}$ represents the stacked random vector for all N nodes of the graph. So:

$$p(\boldsymbol{y}) = \prod_{i=0}^{N} p(y_i); \quad p(y_i) = \mathcal{U}(1, M), \tag{2.15}$$

where $\mathcal{U}(\cdot)$ denotes uniform distribution.

Let $\boldsymbol{E} \in \mathbb{R}^{M \times D}$ denote a binary matrix, where each row represents an epitome and each column represents a latent variable. Fig. 2.7 shows $\boldsymbol{E}$ with $M = 8$ and $D = 16$ in a D-dimensional latent space. The grayed squares of r^{th} row show the latent variables which constitute the epitome $\mathcal{D}_r$. We denote r^{th} row of $\boldsymbol{E}$ by $\boldsymbol{E}[r, :]$.

2.5.1 Generative Model

of EVGAE is given by:

$$p(\mathcal{G}, \boldsymbol{Z}, \boldsymbol{y}) = p(\boldsymbol{y})p(\boldsymbol{Z}|\boldsymbol{y})p_\theta(\mathcal{G}|\boldsymbol{Z}), \tag{2.16}$$

where

$$p(\boldsymbol{Z}|\boldsymbol{y}) = \prod_{i=0}^{N} p(\boldsymbol{z}_i|y_i) \tag{2.17}$$

$$p(\boldsymbol{z}_i|y_i) = \prod_{j=1}^{D} \Big(E[y_i, j]\ \mathcal{N}(0,1) + (1 - E[y_i, j])\delta(0)\Big), \tag{2.18}$$

where $\boldsymbol{E}[y_i, j]$ refers to j^{th} component of epitome y_i. Eq. equation 2.18 shows that $\boldsymbol{z}_i|y_i$ follows standard gaussian distribution for the latent variables j where $E[y_i, j] = 1$ and for the rest it follows degenerate distribution $\delta(0)$ located at 0.

2.5.2 Inference Model

uses the following approximate posterior:

$$q_\phi(\boldsymbol{Z}, \boldsymbol{y}|\mathcal{G}) = q_\phi(\boldsymbol{y}|\mathcal{G})q_\phi(\boldsymbol{Z}|\mathcal{G}), \tag{2.19}$$

with

$$q_\phi(\boldsymbol{y}|\mathcal{G}) = \prod_{i=1}^{N} q_\phi(y_i|\mathcal{G}) \tag{2.20}$$

$$q_\phi(y_i|\mathcal{G}) = \mathrm{Cat}(\boldsymbol{\pi}_i(\mathcal{G})) \tag{2.21}$$

$$q_\phi(\boldsymbol{Z}|\mathcal{G}) = \prod_{i}^{N} q_\phi(\boldsymbol{z}_i|\mathcal{G}) \tag{2.22}$$

$$q_\phi(\boldsymbol{z}_i|\mathcal{G}) = \mathcal{N}\Big(\boldsymbol{\mu}_i(\mathcal{G}), \mathrm{diag}(\boldsymbol{\sigma}_i^2(\mathcal{G}))\Big), \tag{2.23}$$

where Cat(.) refers to the categorical distribution. $\boldsymbol{\pi}_i(.)$, $\boldsymbol{\mu}_i(.)$ and $\boldsymbol{\sigma}_i^2(.)$ are learnt using two-layer GCN networks. Specifically, $\boldsymbol{\pi}_i(.)$ is obtained by learning a real vector which is then passed through softmax layer to give probabilities. Under the assumption that $\boldsymbol{y}$ and $\mathcal{G}$ are independent, given $\boldsymbol{Z}$; the objective function is given by

$$log\Big(p(\mathcal{G})\Big) = log\Big(\int \sum_{\boldsymbol{y}} p(\boldsymbol{y})p(\boldsymbol{Z}|\boldsymbol{y})p_\theta(\mathcal{G}|\boldsymbol{Z})\ d\boldsymbol{Z}\Big) \tag{2.24}$$

$$= log\Big(\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{y})\sim q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\{\frac{p(\boldsymbol{y})p(\boldsymbol{Z}|\boldsymbol{y})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\}\Big) \tag{2.25}$$

$$= log\Big(\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{y})\sim q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\{\frac{p(\boldsymbol{y})p(\boldsymbol{Z}|\boldsymbol{y})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})q_\phi(\boldsymbol{y}|\mathcal{G})}\Big\}\Big). \tag{2.26}$$

By using Jensen's inequality [49], the ELBO bound for log probability becomes

$$log\Big(p(\mathcal{G})\Big) \geq \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{y})\sim q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{y})p(\boldsymbol{Z}|\boldsymbol{y})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})q_\phi(\boldsymbol{y}|\mathcal{G})}\Big)\Big\} \tag{2.27}$$

$$\begin{aligned} &= \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(p_\theta(\mathcal{G}|\boldsymbol{Z})\Big)\Big\} \\ &+ \mathbb{E}_{\boldsymbol{y}\sim q_\phi(\boldsymbol{y}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{y})}{q_\phi(\boldsymbol{y}|\mathcal{G})}\Big)\Big\} \\ &+ \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{y})\sim q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z}|\boldsymbol{y})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\}. \end{aligned} \tag{2.28}$$

Following VGAE[1], we restrict the decoder to recover only edge information from the latent space. Hence, the decoder is the same as in Eq. equation 2.4. Thus, the first term in Eq. equation 2.28 simplifies in a similar way as in VGAE i.e. binary cross-entropy between input and reconstructed edges.

The second term in Eq. equation 2.28 is computed as:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{y}\sim q_\phi(\boldsymbol{y}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{y})}{q_\phi(\boldsymbol{y}|\mathcal{G})}\Big)\Big\} &= \mathbb{E}_{\boldsymbol{y}\sim q_\phi(\boldsymbol{y}|\mathcal{G})}\Big\{\sum_{i=1}^{N} log\Big(\frac{p(y_i)}{q_\phi(y_i|\mathcal{G})}\Big)\Big\} \\ &= \sum_{i=1}^{N} \mathbb{E}_{y_i\sim q_\phi(y_i|\mathcal{G})}\Big\{log\Big(\frac{p(y_i)}{q_\phi(y_i|\mathcal{G})}\Big)\Big\} \\ &= -\sum_{i=1}^{N} D_{KL}\Big(q_\phi(y_i|\mathcal{G})||p(y_i)\Big) \\ &= -\sum_{i=1}^{N} D_{KL}\Big(\text{Cat}(\boldsymbol{\pi}_i(\mathcal{G}))||\,\mathcal{U}(1,M)\Big). \end{aligned} \tag{2.29}$$

The third term in Eq. equation 2.28 is computed as follows:

$$\begin{aligned} &\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{y})\sim q_\phi(\boldsymbol{Z},\boldsymbol{y}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z}|\boldsymbol{y})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \\ =&\mathbb{E}_{\boldsymbol{y}\sim q_\phi(\boldsymbol{y}|\mathcal{G})}\Big\{\mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z}|\boldsymbol{y})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\}\Big\} \\ =&\sum_{\boldsymbol{y}} q_\phi(\boldsymbol{y}|\mathcal{G})\mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z}|\boldsymbol{y})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \\ =&\sum_{i=1}^{N}\sum_{\boldsymbol{y}} q_\phi(\boldsymbol{y}|\mathcal{G})\mathbb{E}_{\boldsymbol{z}_i\sim q_\phi(\boldsymbol{z}_i|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{z}_i|y_i)}{q_\phi(\boldsymbol{z}_i|\mathcal{G})}\Big)\Big\} \\ =&\sum_{i=1}^{N}\sum_{y_i} q_\phi(y_i|\mathcal{G})\mathbb{E}_{\boldsymbol{z}_i\sim q_\phi(\boldsymbol{z}_i|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{z}_i|y_i)}{q_\phi(\boldsymbol{z}_i|\mathcal{G})}\Big)\Big\} \\ =&-\sum_{i=1}^{N}\sum_{y_i} q_\phi(y_i|\mathcal{G})D_{KL}\Big(q_\phi(\boldsymbol{z}_i|\mathcal{G})||p(\boldsymbol{z}_i|y_i)\Big) \end{aligned} \tag{2.30}$$

We take motivation from [18] to compute Eq. equation 2.30 as:

$$
\begin{aligned}
&-\sum_{i=1}^{N}\sum_{y_i} q_\phi(y_i|\mathcal{G}) D_{KL}\Big(q_\phi(\boldsymbol{z}_i|\mathcal{G})||p(\boldsymbol{z}_i|y_i)\Big) \\
&= -\sum_{i=1}^{N}\sum_{y_i} q_\phi(y_i|\mathcal{G}) \sum_{j=1}^{D} \boldsymbol{E}[y_i,j] D_{KL}\Big(q_\phi(z_i^j|\mathcal{G})||p(z_i^j)\Big) \qquad (2.31)\\
&= -\sum_{i=1}^{N}\sum_{y_i} \boldsymbol{\pi}_i(\mathcal{G}) \sum_{j=1}^{D} \boldsymbol{E}[y_i,j] \\
&\qquad D_{KL}\Big(\mathcal{N}\Big(\mu_i^j(\mathcal{G}),(\sigma_i^2)^j(\mathcal{G})\Big)||\mathcal{N}(0,1)\Big), \qquad (2.32)
\end{aligned}
$$

where z_i^j denotes j^{th} component of vector $\boldsymbol{z}_i$. In Eq. equation 2.32, for each node, we sum over all the epitomes. For a given epitome, we only consider the effect of those latent variables which are selected by $\boldsymbol{E}$ for that epitome. This also implies that the remaining latent variables have the freedom to better learn the reconstruction. Consequently, EVGAE encourages more hidden units to be active without penalizing the hidden units which are contributing little to the reconstruction. The final loss function is given by:

$$
\begin{aligned}
\mathcal{L} = \text{BCE} &+ \sum_{i=1}^{N} D_{KL}\Big(\text{Cat}(\boldsymbol{\pi}_i(\mathcal{G}))||\ \mathcal{U}(1,M)\Big) \\
&+ \sum_{i=1}^{N}\sum_{y_i} \boldsymbol{\pi}_i(\mathcal{G}) \sum_{j=1}^{D} \boldsymbol{E}[y_i,j] \\
&\qquad D_{KL}\Big(\mathcal{N}\Big(\mu_i^j(\mathcal{G}),(\sigma_i^2)^j(\mathcal{G})\Big)||\mathcal{N}(0,1)\Big). \qquad (2.33)
\end{aligned}
$$

VGAE model can be recovered from EVGAE model, if we have only one epitome consisting of all latent variables. Hence the model generalizes VGAE. The algorithm for training EVGAE is given in Algo. 1.

2.6 Experiments

2.6.1 Datasets

We compare the performance of EVGAE with several baseline methods on the link prediction task. We conduct the experiments on three benchmark citation datasets[47].

Cora dataset has 2,708 nodes with 5,297 undirected and unweighted links. The nodes are defined by 1433 dimensional binary feature vectors, divided in 7 classes.

Citeseer dataset has 3,312 nodes defined by 3703 dimensional feature vectors. The nodes are divided in 6 distinct classes. There are 4,732 links between the nodes.

PubMed consists of 19,717 nodes defined by 500 dimensional feature vectors linked by 44,338 unweighted and undirected edges. These nodes are divided in 3 classes.

```
Algorithm 1 EVGAE Algorithm
Input:
  • $\mathcal{G}$
  • Epochs
  • The matrix $\boldsymbol{E}$ to select latent variables
    for each epitome.
Initialize model weights; $i = 1$
$While \quad e \leq Epochs$
compute $\boldsymbol{\pi_i}(.)$, $\boldsymbol{\mu_i}(.)$ and $\boldsymbol{\sigma_i^2}(.)$ $\forall i$
compute $\boldsymbol{z_i}$ $\forall i$ by reparameterization trick
compute loss using Eq. equation 2.33
update model weights using back propagation
```

2.6.2 Implementation Details and Performance Comparison

In order to ensure fair comparison, we follow the experimental setup of Kipf and Welling[1]. That is, we train the EVGAE and pure VGAE model on an incomplete version of citation datasets. Concretely, the edges of the dataset are divided in training set, validation set and test set. Following [1], we use 85% edges for training, 5% for validation and 10% for testing the performance of the model.

Method	**Cora**		**Citeseer**		**PubMed**	
	AUC	AP	AUC	AP	AUC	AP
DeepWalk	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.0
Spectral Clustering	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
GAE (VGAE[1] with $\beta = 0$)	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	96.4 ± 0.00	96.5 ± 0.0
VGAE [1] ($\beta \sim 10^{-4} - 10^{-5}$)	91.4 ± 0.01	92.6 ± 0.01	90.8 ± 0.02	92.0 ± 0.02	94.4 ± 0.02	94.7 ± 0.0
pure VGAE (β = 1)	79.44 ± 0.03	80.51 ± 0.02	77.08 ± 0.03	79.07 ± 0.02	82.79 ± 0.01	83.88 ± 0.01
EVGAE (β = 1)	$\mathbf{92.96 \pm 0.02}$	$\mathbf{93.58 \pm 0.03}$	$\mathbf{91.55 \pm 0.03}$	$\mathbf{93.24 \pm 0.02}$	$\mathbf{96.80 \pm 0.01}$	$\mathbf{96.91 \pm 0.02}$

Table 2.1: Results of link prediction on citation datasets

We compare the performance of EVGAE with three strong baselines, namely: VGAE[1], spectral clustering[53] and DeepWalk[54]. We also report the performance of pure VGAE (β=1) and GAE (VGAE with β=0). Since DeepWalk and spectral clustering do not employ node features; so VGAE, GAE and EVGAE have an undue advantage over them. The implementation of spectral clustering is taken from [55] with 128 dimensional embedding and for DeepWalk, the standard implementation is used [56]. For VGAE and GAE, we use the implementation provided by Kipf and Welling[1]. EVGAE also follows

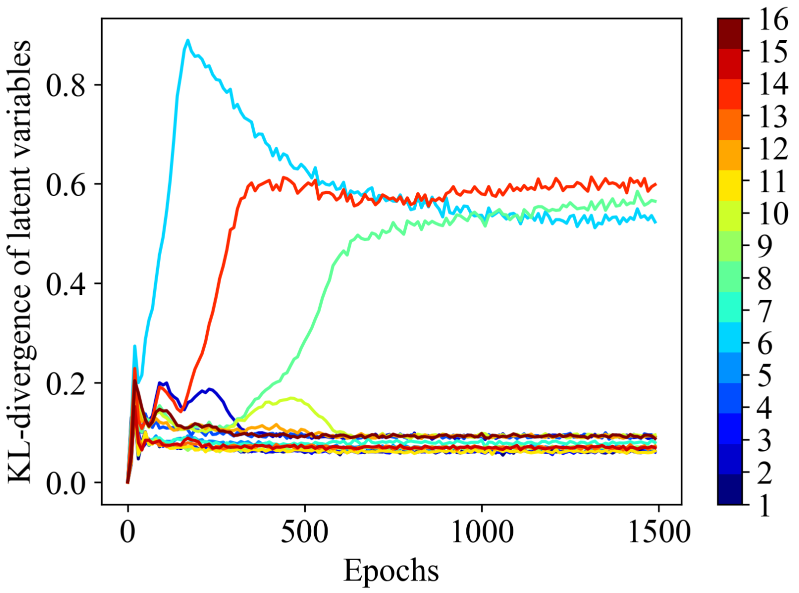

Figure 2.8: KL-divergence of latent variables in EVGAE

a similar structure with latent embedding being 512 dimensional and the hidden layer consisting of 1024 hidden units, half of which learn $\boldsymbol{\mu}_i(.)$ and the other half for learning log-variance. We select 256 epitomes for all three datasets. Each epitome enforces three units to be active, while sharing one unit with neighboring epitomes. This can also be viewed as an extension of the matrix shown in Fig. 2.7. Adam [57] is used as optimizer with learning rate $1e^{-3}$. Further implementation details of EVGAE can be found in the code [58].

For evaluation, we follow the same protocols as other recent works [1][54][53]. That is, we measure the performance of models in terms of area under the ROC curve (AUC) and average precision (AP) scores on the test set. We repeat each experiment 10 times in order to estimate the mean and the standard deviation in the performance of the models.

We can observe from Table 2.1 that the results of EVGAE are competitive or slightly better than other methods. We also note that the performance of variational method pure VGAE is quite bad as compared to our variational method EVGAE. Moreover, the performance of methods with no or poor generative ability (GAE and VGAE [1] with $\beta \sim 10^{-4} - 10^{-5}$) is quite similar.

2.6.3 EVGAE: Over-pruning and Generative Ability

We now show the learning behavior of EVGAE model on our running example of Cora dataset. We select 8 epitomes, each dictating three hidden units to be active. The configuration is shown in Fig. 2.7. Fig. 2.6.2 shows the evolution of KL-divergence and unit activity during training of EVGAE model. By comparing this figure with pure VGAE (Fig. 2.2), we can observe that EVGAE has more active hidden units. This

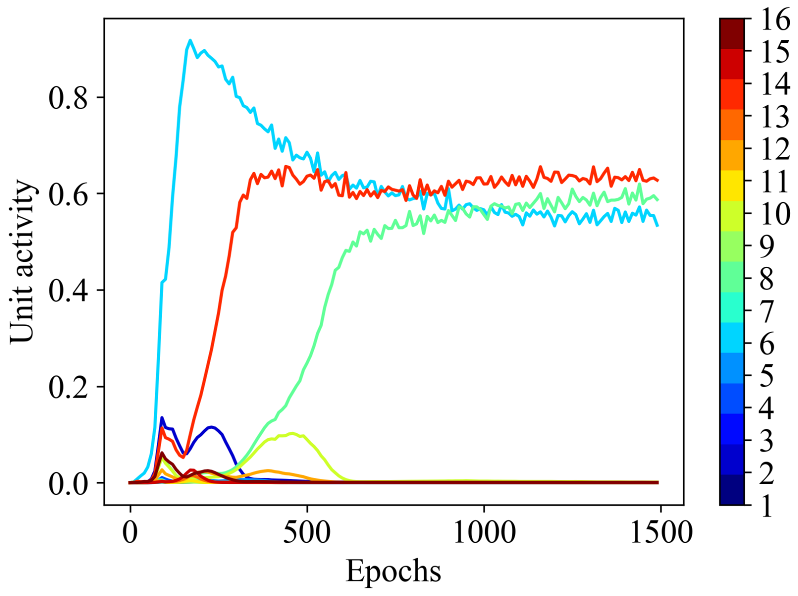

Figure 2.9: Unit activity of 16 hidden units of EVGAE

demonstrates that our model is better than pure VGAE at mitigating the over-pruning issue.

On the other hand, if we compare it to VGAE[1](Fig. 2.3), we observe EVGAE have less active units in comparison. But KL-divergence of the latent variables for VGAE is greater than 1 for all the latent variables (Fig. 2.3). This implies that the latent distribution is quite different from the prior distribution (standard gaussian). In contrast, we observe from Fig. 2.8 that EVGAE has KL-divergence around 0.1 for 13 latent variables and approximately 0.6 for remaining 3 latent variables. This reinforces our claim that VGAE achieves more active hidden units by excessively penalizing the KL-term responsible for generative ability.

In short, although EVGAE has less active units, the distribution matching is better compared to VGAE. VGAE is akin to GAE due to such low weightage to KL-term, i.e. $\beta = 0.0003$.

2.6.4 Impact of Latent Space Dimension

We now look at the impact of latent space dimension on the number of active units and average KL-divergence per active unit. We plot the active units for dimensions $D \in \{16, 32, 64, 128, 256, 512\}$. Fig. 2.6.3 presents an overview of this impact on our running example (Cora dataset). For all values of D, the number of epitomes is set to $\frac{D}{2}$ and one unit is allowed to overlap with neighboring epitomes. Similar to the configuration in Fig. 2.7 for $D = 16$. It is to be noted that we kept the same configuration of epitomes for consistency reasons. Choosing a different configuration of epitomes does not affect the learning behavior of EVGAE.

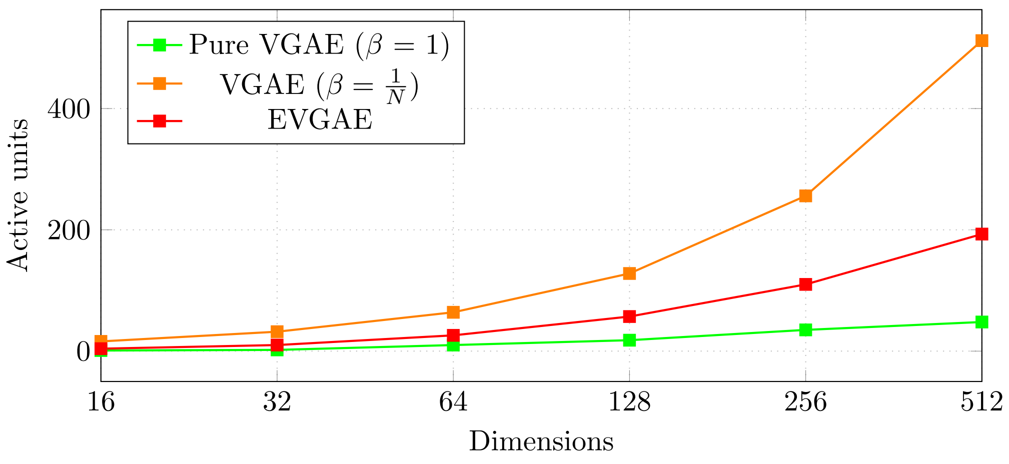

Figure 2.10: Active hidden units with varying latent space dimensions

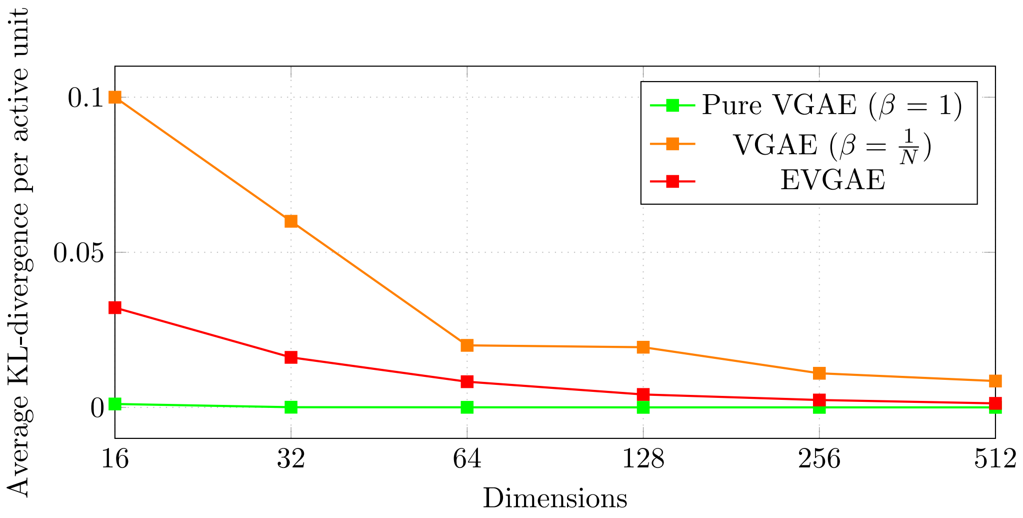

Figure 2.11: Effect of changing latent space dimensions on active units and their KL-divergence. It can be observed that EVGAE has more active units compared to VGAE, and with better generative ability

It can be observed that the number of active units is quite less compared to the available units for VGAE with $\beta = 1$ (pure VGAE). Concretely, for $D = 512$ only 48 units are active. This shows that the over-pruning problem persists even in high dimensional latent space.

Now we observe the behavior of VGAE with $\beta = N^{-1}$ as proposed by Kipf and Welling[1], where N denotes the number of nodes in the graph. All the units are active irrespective of the dimension of latent space. In the case of EVGAE, the number of active units is in between the two. i.e. we are able to mitigate the over-pruning without sacrificing the generative ability ($\beta = 1$). This results in better performance in graph analysis tasks as shown in table 2.1.

To demonstrate that EVGAE achieves better distribution matching than VGAE, we compare the average KL-divergence of active units for different latent space dimensions. Only active units are considered when averaging the KL-divergence because the inactive units introduce a bias towards zero in the results. Fig. 2.11 shows how the distribution matching varies as we increase the number of dimensions. We note that when $\beta = 1$, the average KL-divergence for active units is still quite small, indicating a good match between learned latent distribution and the prior. Conversely, when $\beta = N^{-1}$ the average KL-divergence per active unit is quite high. This supports our claim that original VGAE[1] learns a latent distribution which is quite different from the prior. Thus, when we generate new samples from standard gaussian distribution and pass it through the decoder, we get quite different output than the graph data used for training. In the case of EVGAE, the KL divergence is quite closer to the prior compared to VGAE. For $D = 512$, it is almost similar to the case with $\beta = 1$.

3 Unsupervised Learning of Joint Embeddings for Node Representation and Community Detection

This chapter is based on the following peer-reviewed publication:

*Khan, Rayyan Ahmad [*]; **Anwaar, Muhammad Umer [*]**; Kaddah, Omran, Zhiwei Han and Kleinsteuber, Martin. Unsupervised Learning of Joint Embeddings for Node Representation and Community Detection. In the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), 2021.*
[*] indicates that the authors contributed equally to this work.

3.1 Introduction

Graphs are flexible data structures that model complex relationships among entities, i.e. data points as nodes and the relations between nodes via edges. One important task in graph analysis is community detection, where the objective is to cluster nodes into multiple groups (communities). Each community is a set of densely connected nodes. The communities can be overlapping or non-overlapping, depending on whether they share some nodes or not. Several algorithmic [59, 60] and probabilistic approaches [61, 26, 62, 24] to community detection have been proposed. Another fundamental task in graph analysis is learning the node embeddings. These embeddings can then be used for downstream tasks like graph visualization [63, 64, 65, 62] and classification [66, 67].

In the literature, these tasks are usually treated separately. Although the standard graph embedding methods capture the basic connectivity, the learning of the node embeddings is independent of community detection. For instance, a simple approach can be to get the node embeddings via DeepWalk [27] and get community assignments for each node by using k-means or Gaussian mixture model. Looking from the other perspective, methods like Bigclam [23], that focus on finding the community structure in the dataset, perform poorly for node-representation tasks e.g. node classification. This motivates us to study the approaches that jointly learn community-aware node embeddings.

Recently several approaches, like CNRL [33], ComE [32], vGraph [25] etc, have been proposed to learn the node embeddings and detect communities simultaneously in a unified framework. Several studies have shown that community detection is improved by incorporating the node representation in the learning process [66, 68]. The intuition is that the global structure of graphs learned during community detection can provide useful context for node embeddings and vice versa.

The joint learning methods (CNRL, ComE and vGraph) learn two embeddings for each node. One node embedding is used for the node representation task. The second node embedding is the "context" embedding of the node which aids in community detection. As CNRL and ComE are based on Skip-Gram [69] and DeepWalk [27], they inherit "context" embedding from it for learning the neighbourhood information of the node. vGraph also requires two node embeddings for parameterizing two different distributions. In contrast, we propose learning a single community-aware node representation which is directly used for both tasks.

In this paper, we propose an efficient generative model called **J-ENC** for jointly learning both community detection and node representation. The underlying intuition behind **J-ENC** is that every node can be a member of one or more communities. However, the node embeddings should be learned in such a way that connected nodes are "closer" to each other than unconnected nodes. Moreover, connected nodes should have similar community assignments. Formally, we assume that for i-th node, the node embeddings $\boldsymbol{z}_i$ are generated from a prior distribution $p(\boldsymbol{z})$. Given $\boldsymbol{z}_i$, the community assignments c_i are sampled from $p(c_i|\boldsymbol{z}_i)$, which is parameterized by node and community embeddings. In order to generate an edge (i, j), we sample another node embedding $\boldsymbol{z}_j$ from $p(\boldsymbol{z})$ and respective community assignment c_j from $p(c_j|\boldsymbol{z}_j)$. Afterwards, the node embeddings and the respective community assignments of node pairs are fed to a decoder. The decoder ensures that embeddings of both the nodes and the communities of connected nodes share high similarity. This enables learning such node embeddings that are useful for both community detection and node representation tasks.

It is pertinent to highlight that although both vGraph and **J-ENC** adopt a variational approach but the underlying models are quite different. vGraph assumes that each node can be represented as a mixture of multiple communities and is described by a multinomial distribution over communities, whereas **J-ENC** models the node embedding by a single distribution. For a given node, vGraph, first draws a community assignment and then a connected neighbor node is generated based on the assignment. Whereas, **J-ENC** draws the node embedding from prior distribution and then community assignment is conditioned on a single node only. In simple terms, vGraph also needs edge information in the generative process whereas **J-ENC** does not require it. **J-ENC** relies on the decoder to ensure that embeddings of the connected nodes and their communities share high similarity with each other.

We validate the effectiveness of our approach on several real-world graph datasets. In Sec. 3.3, we show empirically that **J-ENC** is able to outperform the baseline methods including the direct competitors on all three tasks i.e. node classification, overlapping community detection and non-overlapping community detection. Furthermore, we compare the computational cost of training different algorithms. **J-ENC** is up to 40x more time-efficient than its competitors. We also conduct hyperparameter sensitivity analysis which demonstrates the robustness of our approach. Our main contributions are summarized below:

- We propose an efficient generative model called **J-ENC** for joint community detection and node representation learning.
- We adopt a novel approach and argue that a single node embedding is sufficient for learning both the representation of the node itself and its context.
- Training **J-ENC** is extremely time-efficient in comparison to its competitors.

3.2 Methodology

3.2.1 Problem Formulation

Suppose an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ and a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ of F-dimensional node features, N being the number of nodes. Given K as the number of communities, we aim to jointly learn the node embeddings and the community embeddings following a variational approach such that:

- One or more communities can be assigned to every node.
- The node embeddings can be used for both community detection and node classification.

3.2.2 Variational Model

3.2.2.1 Generative Model:

Let us denote the latent node embedding and community assignment for i-th node by the random variables $\boldsymbol{z}_i \in \mathbb{R}^d$ and c_i respectively. The generative model is given by:

$$p(\boldsymbol{A}) = \int \sum_{\boldsymbol{c}} p(\boldsymbol{Z}, \boldsymbol{c}, \boldsymbol{A}) d\boldsymbol{Z}, \tag{3.1}$$

where $\boldsymbol{c} = [c_1, c_2, \cdots, c_N]$ and the matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N]$ stacks the node embeddings. The joint distribution in (3.1) is mathematically expressed as

$$p(\boldsymbol{Z}, \boldsymbol{c}, \boldsymbol{A}) = p(\boldsymbol{Z})\ p_\theta(\boldsymbol{c}|\boldsymbol{Z})\ p_\theta(\boldsymbol{A}|\boldsymbol{c}, \boldsymbol{Z}), \tag{3.2}$$

where θ denotes the model parameters. Let us denote elements of $\boldsymbol{A}$ by a_{ij}. Following existing approaches [1, 20], we consider $\boldsymbol{z}_i$ to be $i.i.d$ random variables. Furthermore, assuming $c_i|\boldsymbol{z}_i$ to be $i.i.d$ random variables, the joint distributions in (3.2) can be factorized as

$$p(\boldsymbol{Z}) = \prod_{i=1}^{N} p(\boldsymbol{z}_i) \tag{3.3}$$

$$p_\theta(\boldsymbol{c}|\boldsymbol{Z}) = \prod_{i=1}^{N} p_\theta(c_i|\boldsymbol{z}_i) \tag{3.4}$$

$$p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z}) = \prod_{i,j} p_\theta(a_{ij}|c_i,c_j,\boldsymbol{z}_i,\boldsymbol{z}_j), \tag{3.5}$$

where Eq. (3.5) assumes that the *edge decoder* $p_\theta(a_{ij}|c_i,c_j,\boldsymbol{z}_i,\boldsymbol{z}_j)$ depends only on $c_i, c_j, \boldsymbol{z}_i$ and $\boldsymbol{z}_j$.

3.2.2.2 Inference Model:

We aim to learn the model parameters θ such that $\log(p_\theta(\boldsymbol{A}))$ is maximized. In order to ensure computational tractability, we introduce the approximate posterior

$$q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I}) = \prod_i q_\phi(\boldsymbol{z}_i,c_i|\mathcal{I}) \tag{3.6}$$

$$= \prod_i q_\phi(\boldsymbol{z}_i|\mathcal{I})q_\phi(c_i|\boldsymbol{z}_i,\mathcal{I}), \tag{3.7}$$

where $\mathcal{I} = (\boldsymbol{A},\boldsymbol{X})$ if node features are available, otherwise $\mathcal{I} = \boldsymbol{A}$.

$$\log(p_\theta(\boldsymbol{A})) = \log\bigg(\int\sum_{\boldsymbol{c}} p_\theta(\boldsymbol{Z},\boldsymbol{c},\boldsymbol{A})d\boldsymbol{Z}\bigg) \tag{3.8}$$

$$= \log\bigg(\int\sum_{\boldsymbol{c}} p(\boldsymbol{Z})\ p_\theta(\boldsymbol{c}|\boldsymbol{Z})\ p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})d\boldsymbol{Z}\bigg) \tag{3.9}$$

$$= \log\bigg(\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\bigg\{\frac{p(\boldsymbol{Z})\ p_\theta(\boldsymbol{c}|\boldsymbol{Z})\ p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{I})q_\phi(\boldsymbol{c}|\boldsymbol{Z},\mathcal{I})}\bigg\}\bigg) \tag{3.10}$$

$$\geq \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\bigg\{\log\bigg(\frac{p(\boldsymbol{Z})\ p_\theta(\boldsymbol{c}|\boldsymbol{Z})\ p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{I})q_\phi(\boldsymbol{c}|\boldsymbol{Z},\mathcal{I})}\bigg)\bigg\} \tag{3.11}$$

$$= \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\bigg\{\log\bigg(\frac{p(\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{I})}\bigg)$$

$$+ \log\bigg(\frac{p_\theta(\boldsymbol{c}|\boldsymbol{Z})}{q_\phi(\boldsymbol{c}|\boldsymbol{Z},\mathcal{I})}\bigg) + \log\bigg(p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})\bigg)\bigg\} \tag{3.12}$$

$$= \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{I})}\bigg\{\log\bigg(\frac{p(\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{I})}\bigg)\bigg\}$$

$$+ \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\bigg\{\log\bigg(\frac{p_\theta(\boldsymbol{c}|\boldsymbol{Z})}{q_\phi(\boldsymbol{c}|\boldsymbol{Z},\mathcal{I})}\bigg)\bigg\}$$

$$+ \mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\bigg\{\log\bigg(p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})\bigg)\bigg\}. \tag{3.13}$$

Where (3.11) follows from Jensen's Inequality. First term of (3.13) is given by:

$$\mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{I})}\bigg\{\log\bigg(\frac{p(\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{I})}\bigg)\bigg\}$$

$$= \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{z}_i\sim q_\phi(\boldsymbol{z}_i|\mathcal{I})}\bigg\{\log\bigg(\frac{p(\boldsymbol{z}_i)}{q_\phi(\boldsymbol{z}_i|\mathcal{I})}\bigg)\bigg\} \tag{3.14}$$

$$= -\sum_{i=1}^{N} D_{KL}(q_\phi(\boldsymbol{z}_i|\mathcal{I})\ ||\ p(\boldsymbol{z}_i)). \tag{3.15}$$

Second term of Eq. (3.13) can be derived as:

$$\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\left\{\log\left(\frac{p_\theta(\boldsymbol{c}|\boldsymbol{Z})}{q_\phi(\boldsymbol{c}|\boldsymbol{Z},\mathcal{I})}\right)\right\}$$

$$= \sum_{i=1}^{N}\mathbb{E}_{(\boldsymbol{z}_i,c_i)\sim q_\phi(\boldsymbol{z}_i,c_i|\mathcal{I})}\left\{\log\left(\frac{p_\theta(c_i|\boldsymbol{z}_i)}{q_\phi(c_i|\boldsymbol{z}_i,\mathcal{I})}\right)\right\} \tag{3.16}$$

$$\approx \sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{c_i\sim q_\phi(c_i|\boldsymbol{z}_i^{(m)},\mathcal{I})}\left\{\log\left(\frac{p_\theta(c_i|\boldsymbol{z}_i^{(m)})}{q_\phi(c_i|\boldsymbol{z}_i^{(m)},\mathcal{I})}\right)\right\} \tag{3.17}$$

$$= -\sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}D_{KL}(q_\phi(c_i|\boldsymbol{z}_i^{(m)},\mathcal{I})\ ||\ p_\theta(c_i|\boldsymbol{z}_i^{(m)})) \tag{3.18}$$

where (3.17) follows from Eq. (3.16) by replacing the expectation over $\boldsymbol{z}_i$ with sample mean by generating M samples $\boldsymbol{z}_i^{(m)}$ from distribution $q(\boldsymbol{z}_i|\mathcal{I})$. Assuming $a_{ij} \in [0,1]\ \forall i$, the third term of Eq. (3.13) is the negative of binary cross entropy (BCE) between observed and predicted edges.

$$\mathbb{E}_{(\boldsymbol{Z},\boldsymbol{c})\sim q_\phi(\boldsymbol{Z},\boldsymbol{c}|\mathcal{I})}\left\{\log\left(p_\theta(\boldsymbol{A}|\boldsymbol{c},\boldsymbol{Z})\right)\right\}$$

$$= \sum_{(i,j)\in\mathcal{E}}\mathbb{E}_{(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j)\sim q_\phi(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j|\mathcal{I})}\Big\{ \log\left(p_\theta(a_{ij}|c_i,c_j,\boldsymbol{z}_i,\boldsymbol{z}_j)\right)\Big\} \tag{3.19}$$

Hence, by substituting Eq. (3.15) and Eq. (3.18) in Eq. (3.13), we get the ELBO bound as:

$$\begin{aligned}\mathcal{L}_{ELBO} \approx\ & -\sum_{i=1}^{N}D_{KL}(q_\phi(\boldsymbol{z}_i|\mathcal{I})\ ||\ p(\boldsymbol{z}_i)) \\ & -\sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}D_{KL}(q_\phi(c_i|\boldsymbol{z}_i^{(m)},\mathcal{I})\ ||\ p_\theta(c_i|\boldsymbol{z}_i^{(m)})) \\ & +\sum_{(i,j)\in\mathcal{E}}\mathbb{E}_{(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j)\sim q_\phi(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j|\mathcal{I})}\Big\{ \log\left(p_\theta(a_{ij}|c_i,c_j,\boldsymbol{z}_i,\boldsymbol{z}_j)\right)\Big\}\end{aligned} \tag{3.20}$$

where $D_{KL}(.||.)$ represents the KL-divergence between two distributions. The distribution $q_\phi(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j|\mathcal{I})$ in the third term of Eq. (**??**) is factorized into two conditionally independent distributions i.e.

$$q_\phi(\boldsymbol{z}_i,\boldsymbol{z}_j,c_i,c_j|\mathcal{I}) = q_\phi(\boldsymbol{z}_i,c_i|\mathcal{I})q_\phi(\boldsymbol{z}_j,c_j|\mathcal{I}). \tag{3.21}$$

3.2.3 Design Choices

In Eq. (3.3), $p(\boldsymbol{z}_i)$ is chosen to be the standard gaussian distribution for all i. The corresponding approximate posterior $q_\phi(\boldsymbol{z}_i|\mathcal{I})$ in Eq. (3.7), used as node embeddings encoder, is given by

$$q_\phi(\boldsymbol{z}_i|\mathcal{I}) = \mathcal{N}\big(\boldsymbol{\mu}_i(\mathcal{I}), \mathrm{diag}(\boldsymbol{\sigma}^2{}_i(\mathcal{I}))\big). \tag{3.22}$$

The parameters of $q_\phi(\boldsymbol{z}_i|\mathcal{I})$ can be learnt by any encoder network e.g. graph convolutional network [70], graph attention network [37], GraphSAGE [29] or even two matrices to learn $\boldsymbol{\mu}_i(\mathcal{I})$ and $\mathrm{diag}(\boldsymbol{\sigma}^2{}_i(\mathcal{I}))$. Samples are then generated using reparametrization trick [71].

For parameterizing $p_\theta(c_i|\boldsymbol{z}_i)$ in Eq. (3.4), we introduce the community embeddings $\{\boldsymbol{g}_1, \cdots, \boldsymbol{g}_K\}$; $\boldsymbol{g}_k \in \mathbb{R}^d$. The distribution $p_\theta(c_i|\boldsymbol{z}_i)$ is then modelled as the softmax of dot products of $\boldsymbol{z}_i$ with $\boldsymbol{g}_k$, i.e.

$$p_\theta(c_i = k|\boldsymbol{z}_i) = \frac{\exp(<\boldsymbol{z}_i, \boldsymbol{g}_k>)}{\sum\limits_{\ell=1}^{K} \exp(<\boldsymbol{z}_i, \boldsymbol{g}_\ell>)}. \tag{3.23}$$

The corresponding approximate posterior $q_\phi(c_i = k|\boldsymbol{z}_i, \mathcal{I})$ in Eq. (3.7) is affected by the node embedding $\boldsymbol{z}_i$ as well as the neighborhood. To design this, our intuition is to consider the similarity of $\boldsymbol{g}_k$ with the embedding $\boldsymbol{z}_i$ as well as with the embeddings of the neighbors of the i-th node. The overall similarity with neighbors is mathematically formulated as the average of the dot products of their embeddings. Afterwards, a hyperparameter α is introduced to control the bias between the effect of $\boldsymbol{z}_i$ and the set $\mathcal{N}_i$ of the neighbors of the i-th node. Finally, a softmax is applied, i.e.

$$\begin{aligned} q_\phi(c_i = k|\boldsymbol{z}_i, \mathcal{G}) = \mathrm{softmax}\Big(&\alpha <\boldsymbol{z}_i, \boldsymbol{g}_k> \\ &+ (1-\alpha)\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} <\boldsymbol{z}_j, \boldsymbol{g}_k>\Big). \end{aligned} \tag{3.24}$$

Hence, Eq. (3.24) ensures that graph structure information is employed to learn community assignments instead of relying on an extraneous node embedding as done in [25, 32]. Finally, the choice of edge decoder in Eq. (3.5) is motivated by the intuition that the nodes connected by edges have a high probability of belonging to the same community and vice versa. Therefore we model the edge decoder as:

$$p_\theta(a_{ij}|c_i = \ell, c_j = m, \boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{\sigma(<\boldsymbol{z}_i, \boldsymbol{g}_m>) + \sigma(<\boldsymbol{z}_j, \boldsymbol{g}_\ell>)}{2}. \tag{3.25}$$

For better reconstructing the edges, Eq. (3.25) makes use of the community embeddings, node embeddings and community assignment information simultaneously. This helps in learning better node representations by leveraging the global information about the graph structure via community detection. On the other hand, this also forces the community assignment information to exploit the local graph structure via node embeddings and edge information.

3.2.4 Practical Aspects

The third term in Eq. (**??**) is estimated in practice using the samples generated by the approximate posterior. This term is equivalent to the negative of binary cross-entropy (BCE) loss between observed edges and reconstructed edges. Since community assignment follows a categorical distribution, we use Gumbel-softmax [72] for backpropagation of the gradients. As for the second term of Eq. (**??**), it is also enough to set $M = 1$, i.e. use only one sample per input node.

For inference, non-overlapping community assignment can be obtained for i-th node as

$$\mathcal{C}_i = \underset{k \in \{1,\cdots,K\}}{\arg\max} \; q_\phi(c_i = k | \boldsymbol{z}_i, \mathcal{I}). \tag{3.26}$$

To get overlapping community assignments for i-th node, we can threshold its weighted probability vector at ϵ, a hyperparameter, as follows

$$\mathcal{C}_i = \left\{ k \;\middle|\; \frac{q_\phi(c_i = k | \boldsymbol{z}_i, \mathcal{I})}{\max\limits_{\ell} q_\phi(c_i = \ell | \boldsymbol{z}_i, \mathcal{I})} \geq \epsilon \right\}, \quad \epsilon \in [0, 1]. \tag{3.27}$$

3.2.5 Complexity

Computation of dot products for all combinations of node and community embeddings takes $\mathcal{O}(NKd)$ time. Solving Eq. (3.24) further requires calculation of mean of dot products over the neighborhood for every node, which takes $\mathcal{O}(|\mathcal{E}|K)$ computations overall as we traverse every edge for every community. Finally, we need softmax over all communities for every node in Eq. (3.23) and Eq. (3.24) which takes $\mathcal{O}(NK)$ time. Eq. (3.25) takes $\mathcal{O}(|\mathcal{E}|)$ time for all edges as we have already calculated the dot products. As a result, the overall complexity becomes $\mathcal{O}(|\mathcal{E}|K + NKd)$. This complexity is quite low compared to other algorithms designed to achieve similar goals [32, 73].

3.3 Experiments

3.3.1 Synthetic Example

We start with a synthetic dataset, consisting of 3 communities with 5 points per community. This dataset is actually a random partition graph generated by the python package `networkx`. The encoder simply consists of two matrices that give $\boldsymbol{\mu}_i(\mathcal{I})$ and $\text{diag}({\boldsymbol{\sigma}^2}_i(\mathcal{I}))$. The results of the community assignments discovered by **J-ENC** are given in Fig. 3.1, where the node sizes are reciprocal to the confidence of **J-ENC** in the community assignments. We choose 3 communities for demonstration because the probabilistic community assignments in such case can be thought of as `rgb` values for coloring the nodes. It can be seen that **J-ENC** discovers the correct community structure. However, the two bigger nodes in the center can be assigned to more than one communities as **J-ENC** is not very confident in case of these nodes. This is evident from the colors that are a mix of red, green and blue. We now proceed to the experiments on real-world datasets.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	K	$\lvert F\rvert$	**Overlap**
CiteSeer	3327	9104	6	3703	N
CiteSeer-full	4230	10674	6	602	N
Cora	2708	10556	7	1433	N
Cora-ML	2995	16316	7	2879	N
Cora-full	19793	126842	70	8710	N
fb0	333	2519	24	N/A	Y
fb107	1034	26749	9	N/A	Y
fb1684	786	14024	17	N/A	Y
fb1912	747	30025	46	N/A	Y
fb3437	534	4813	32	N/A	Y
fb348	224	3192	14	N/A	Y
fb414	150	1693	7	N/A	Y
fb698	61	270	13	N/A	Y
youtube	5346	24121	5	N/A	Y
amazon	794	2109	5	N/A	Y
amazon500	1113	3496	500	N/A	Y
amazon1000	1540	4488	1000	N/A	Y
dblp	24493	89063	5	N/A	Y

Table 3.1: Every dataset has $|\mathcal{V}|$ nodes, $|\mathcal{E}|$ edges, K communities and $|F|$ features. $|F| =$ N/A means that either the features were missing or not used.

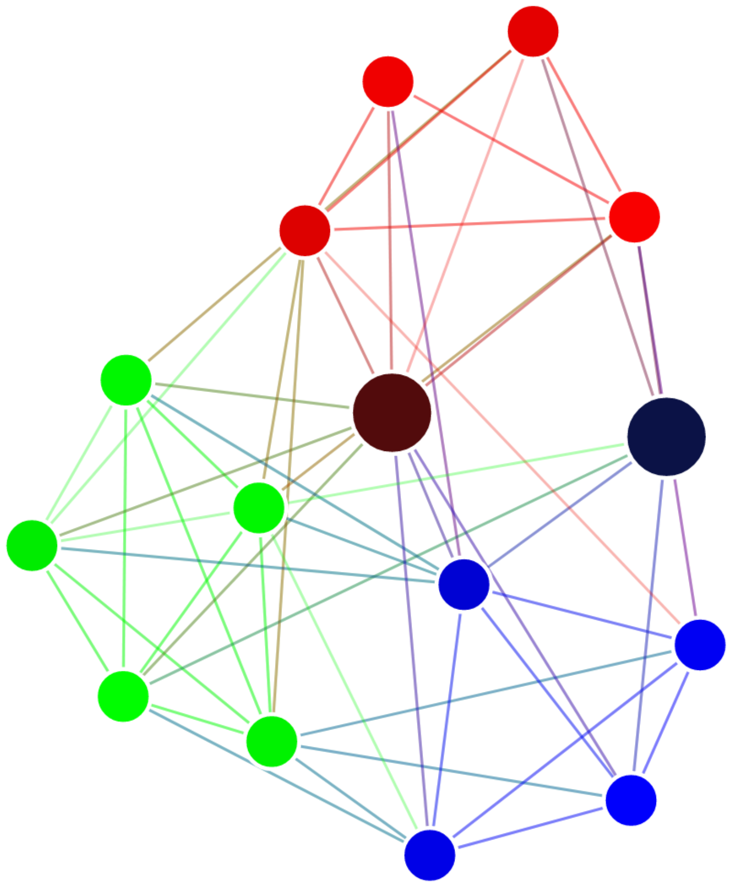

Figure 3.1: Visualization of community assignments discovered by **J-ENC** in the synthetic dataset of 15 points divided in three communities.

3.3.2 Datasets

We have selected 18 different datasets ranging from 270 to 126,842 edges. For non-overlapping community detection and node classification, we use 5 the citation datasets [74, 75]. The remaining datasets [26, 76], used for overlapping community detection, are taken from SNAP repository [77]. Following [25], we take 5 biggest ground truth communities for youtube, amazon and dblp. Moreover, we also analyse the case of large number of communities. For this purpose, we prepare two subsets of amazon dataset by randomly selecting 500 and 1000 communities from 2000 smallest communities in the amazon dataset.

3.3.3 Baselines

For overlapping community detection, we compare with the following competitive baselines:

1. **MNMF**[62] learns community membership distribution by using joint non-negative matrix factorization with modularity based regularization.
2. **BIGCLAM**[23] also formulates community detection as a non-negative matrix factorization (NMF) task. It simultaneously optimizes the model likelihood of observed links and learns the latent factors which represent community affiliations of nodes.
3. **CESNA** [24] extends BIGCLAM by statistically modelling the interaction between the network structure and the node attributes.
4. **Circles** [26] introduces a generative model for community detection in ego-networks by learning node similarity metrics for every community.
5. **SVI** [61] formulates membership of nodes in multiple communities by a Bayesian model of networks.
6. **vGraph** [25] simultaneously learns node embeddings and community assignments by modelling the nodes as being generated from a mixture of communities. **vGraph+**, a variant further incorporates regularization to weigh local connectivity.
7. **ComE** [32] jointly learns community and node embeddings by using gaussian mixture model formulation.
8. **CNRL**[33] enhances the random walk sequences (generated by DeepWalk, node2vec etc) to jointly learn community and node embeddings.
9. **CommunityGAN** (ComGAN)is a generative adversarial model for learning node embeddings such that the entries of the embedding vector of each node refer to the membership strength of the node to different communities.

10. Lastly, we compare the results with the communities obtained by applying k-means to the learned embeddings of **DGI** [31].

For non-overlapping community detection and node classification, we compare **J-ENC** with the following baselines:

1. **DeepWalk** [27] makes use of SkipGram [69] and truncated random walks on network to learn node embeddings.
2. **LINE** [67] learns node embeddings while attempting to preserve first and second order proximities of nodes.
3. **Node2Vec** [28] learns the embeddings using biased random walk while aiming to preserve network neighborhoods of nodes.
4. **GEMSEC** is a sequence sampling-based learning model which aims to jointly learn the node embeddings and clustering assignments.
5. **Graph Autoencoder (GAE)**[1] extends the idea of autoencoders to graph datasets.
6. **VGAE**, variational graph autoencoder, discussed in detail in Chapter 2.
7. **MNMF**, discussed above.
8. **DGI** , discussed above.
9. **CNRL**, discussed above.
10. **CommunityGAN**, discussed above.
11. **vGraph**, discussed above.
12. **ComE**, discussed above.

3.3.4 Settings

For overlapping community detection, we learn mean and log-variance matrices of 16-dimensional node embeddings. We set $\alpha = 0.9$ and $\epsilon = 0.3$ in all our experiments. Following [1], we first pre-train a variational graph autoencoder. We perform gradient descent with Adam optimizer [78] and learning rate = 0.01. Community assignments are obtained using Eq. (3.27). For the baselines, we employ the results reported by [25]. For evaluating the performance, we use *F1-score* and *Jaccard similarity.*

For non-overlapping community detection, since the default implementations of most the baselines use 128 dimensional embeddings, for we use $d = 128$ for fair comparison. Eq. (3.26) is used for community assignments. For vGraph, we use the code provided by the authors. We employ *normalized mutual information (NMI)* and *adjusted random index (ARI)* as evaluation metrics.

For node classification, we follow the training split used in various previous works [75, 70, 31], i.e. 20 nodes per class for training. We train logistic regression using LIBLINEAR [79] solver as our classifier and report the evaluation results on rest of the nodes. For the algorithms that do not use node features, we train the classifier by appending the raw node features with the learnt embeddings. For evaluation, we use *F1-macro* and *F1-micro* scores. All the reported results are the average over five runs.

3.3.5 Discussion of Results

Tables 3.2 and 3.3 summarize the results of the performance comparison for the overlapping community detection task.

First, we note that our proposed method **J-ENC** outperforms the competitors on all datasets in terms of Jaccard score as well as F1-score, with the dataset (*fb0*) being the only exception where **J-ENC** is the second best. These results demonstrate the capability of **J-ENC** to learn multiple community assignments quite well and hence reinforces our intuition behind the design of Eq. (3.24).

Second, we observe that there is no consistent performing algorithm among the competitive methods. That is, excluding **J-ENC** , the best performance is achieved by vGraph/vGraph+ on 5, ComGAN on 4 and ComE on 3 out of 13 datasets in terms of F1-score. A a similar trend can be seen in Jaccard Similarity. It is worth noting that all the methods, which achieve the second-best performance, are solving the task of community detection and node representation learning jointly.

Third, we observe that vGraph+ results are generally better than vGraph. This is because vGraph+ incorporates a regularization term in the loss function which is based on Jaccard coefficients of connected nodes as edge weights. However, it should be noted that this prepossessing step is computationally expensive for densely connected graphs.

Tab. 3.4 and Tab. 3.5 show the results on non-overlapping community detection. First, we observe that MNMF, DeepWalk, LINE and Node2Vec provide a good baseline for the task. However, these methods are not able to achieve comparable performance on any dataset relative to the frameworks that treat the two tasks jointly. Second, **J-ENC** consistently outperforms all the competitors in NMI and ARI metrics, except for *CiteSeer* where it achieves second best ARI. Third, we observe that GCN based models i.e. GAE, VGAE and DGI show competitive performance. That is, they achieve second best performance in all the datasets except *CiteSeer*. In particular, DGI achieves second best NMI results in 3 out of 5 datasets and 2 out of 5 datsets in terms of ARI. Nonetheless, DGI results are not very competitive in Tab. 3.2 and Tab. 3.3, showing that while DGI can be a good choice for learning node embeddings for attributed graphs with non-overlapping communities, it is not the best option for non-attributed graphs or overlapping communities.

The results for node classification are presented in Tab. 3.6 and Tab. 3.7. **J-ENC** achieves best F1-micro and F1-macro scores on 4 out of 5 datasets. We also observe that GCN based models i.e. GAE, VGAE and DGI show competitive performance, following the trend in results of Tab. 3.4 and Tab. 3.5. Furthermore, we note that the node classification results of CommunityGan (ComGAN) are quite poor. We think a potential

Dataset	**MNMF**	**Bigclam**	**CESNA**	**Circles**	**SVI**	**vGraph**	**vGraph+**	**ComE**	**CNRL**	**ComGan**	**DGI**	**J-ENC**
fb0	14.4	29.5	28.1	28.6	28.1	24.4	26.1	31.1	11.5	**35.0**	27.4	34.7
fb107	12.6	39.3	37.3	24.7	26.9	28.2	31.8	39.7	20.2	47.5	35.8	**59.7**
fb1684	12.2	50.4	51.2	28.9	35.9	42.3	43.8	52.9	38.5	47.6	42.8	**56.4**
fb1912	14.9	34.9	34.7	26.2	28.0	25.8	37.5	28.7	8.0	35.6	32.6	**45.8**
fb3437	13.7	19.9	20.1	10.1	15.4	20.9	22.7	21.3	3.9	39.3	19.7	**50.2**
fb348	20.0	49.6	53.8	51.8	46.1	55.4	53.1	46.2	34.1	55.8	54.7	**58.2**
fb414	22.1	58.9	60.1	48.4	38.9	64.7	66.9	55.3	25.3	43.9	56.9	**69.6**
fb698	26.6	54.2	58.7	35.2	40.3	54.0	59.5	45.8	16.4	58.2	52.2	**64.0**
Youtube	59.9	43.7	38.4	36.0	41.4	50.7	52.2	65.5	51.4	43.6	47.8	**67.3**
Amazon	38.2	46.4	46.8	53.3	47.3	53.3	53.2	50.1	53.5	51.4	44.7	**58.1**
Amazon500	30.1	52.2	57.3	46.2	41.9	61.2	60.4	59.8	38.4	59.3	33.8	**67.6**
Amazon1000	19.3	28.6	30.8	25.9	21.6	54.3	47.3	50.3	27.1	52.7	37.7	**60.5**
Dblp	21.8	23.6	35.9	36.2	33.7	39.3	39.9	47.1	46.8	34.9	44.0	**53.9**

Table 3.2: F1 scores (%) for overlapping communities. Best and second best values are bold and blue respectively.

Dataset	MNMF	Bigclam	CESNA	Circles	SVI	vGraph	vGraph+	ComE	CNRL	ComGan	DGI	J-ENC
fb0	08.0	18.5	17.3	18.6	17.6	14.6	15.9	19.5	06.8	24.1	16.8	**24.7**
fb107	06.9	27.5	27.0	15.5	17.2	18.3	21.7	28.7	11.9	38.5	25.3	**46.8**
fb1684	06.6	38.0	38.7	18.7	24.7	29.2	32.7	40.3	25.8	37.9	38.8	**42.5**
fb1912	08.4	24.1	23.9	16.7	20.1	18.6	28.0	18.5	04.6	13.5	22.5	**37.3**
fb3437	07.7	11.5	11.7	05.5	09.0	12.0	13.3	12.5	02.0	33.4	11.6	**36.2**
fb348	11.3	35.9	40.0	39.3	33.6	41.0	40.5	34.4	21.7	23.2	41.8	**43.5**
fb414	12.8	47.1	47.3	34.2	29.3	51.8	55.9	42.2	15.4	53.6	46.4	**58.4**
fb698	16.0	41.9	45.9	22.6	30.0	43.6	47.7	33.8	09.6	46.9	42.1	**50.4**
Youtube	46.7	29.3	24.2	22.1	28.7	34.3	34.8	52.5	35.5	44.0	32.7	**53.3**
Amazon	25.2	35.1	35.0	36.7	36.4	36.9	36.9	34.6	38.7	38.0	29.1	**41.9**
Amazon500	20.8	51.2	53.8	47.2	45.0	59.1	59.6	58.4	41.1	57.3	23.3	**64.9**
Amazon1000	20.3	26.8	28.9	24.9	23.6	54.3	49.7	52.0	26.9	54.1	23.2	**57.1**
Dblp	20.9	13.8	22.3	23.3	20.9	25.0	25.1	27.9	32.8	25.0	29.2	**37.3**

Table 3.3: Jaccard scores (%) for overlapping communities. Best and second best values are bold and blue respectively.

Alg.	CiteSeer	CiteSeer-full	Cora	Cora-ML	Cora-full
MNMF	14.1	09.4	19.7	37.8	42.0
DeepWalk	08.8	15.4	39.7	43.2	48.5
LINE	08.7	13.0	32.8	42.3	40.3
Node2Vec	14.9	22.3	39.7	39.6	48.1
GAE	17.4	55.1	39.7	48.3	48.3
VGAE	16.3	48.4	40.8	48.3	47.0
DGI	37.8	56.7	50.1	46.2	39.9
GEMSEC	11.8	11.1	27.4	18.1	10.0
CNRL	13.6	23.3	39.4	42.9	47.7
ComGAN	03.2	16.2	05.7	11.5	15.0
vGraph	09.0	07.6	26.4	29.8	41.7
ComE	18.8	32.8	39.6	47.6	51.2
J-ENC	**38.5**	**59.0**	**52.7**	**56.3**	**55.2**

Table 3.4: NMI(%): Non-overlapping community detection results. Best and second best values are bold and blue respectively.

Alg.	CiteSeer	CiteSeer-full	Cora	Cora-ML	Cora-full
MNMF	02.6	00.4	02.9	24.1	06.1
DeepWalk	09.5	16.4	31.2	33.9	22.5
LINE	03.3	03.7	14.9	32.7	11.7
Node2Vec	08.1	10.5	25.8	27.9	18.8
GAE	14.1	50.6	29.3	41.8	18.3
VGAE	10.1	40.6	34.7	42.5	17.9
DGI	**38.1**	50.8	44.7	42.1	12.1
GEMSEC	00.6	01.0	04.8	01.0	00.2
CNRL	12.8	20.2	31.9	32.5	22.9
ComGAN	01.2	04.9	03.2	06.7	00.6
vGraph	05.1	04.2	12.7	21.6	14.9
ComE	13.8	20.9	34.2	37.2	19.7
J-ENC	35.2	**60.3**	**45.1**	**49.8**	**28.8**

Table 3.5: ARI(%): Non-overlapping community detection results. Best and second best values are bold and blue respectively.

Alg.	**CiteSeer**	**CiteSeer-full**	**Cora**	**Cora-ML**	**Cora-full**
MNMF	57.4	68.6	60.9	64.2	30.4
DeepWalk	49.0	56.6	69.7	75.8	41.7
LINE	55.0	60.2	68.0	75.3	39.4
Node2Vec	55.2	61.0	71.3	78.4	42.3
GAE	57.9	79.9	71.2	76.5	36.6
VGAE	59.1	74.4	70.4	75.2	32.4
DGI	62.6	**82.1**	71.1	72.6	16.5
GEMSEC	37.5	53.3	60.3	70.6	35.8
CNRL	50.0	58.0	70.4	77.8	41.3
ComGAN	55.9	65.7	56.6	62.5	27.7
vGraph	30.8	28.5	44.7	59.8	33.4
ComE	59.6	69.9	71.6	78.5	42.2
J-ENC	**64.8**	76.8	**73.1**	**80.2**	**43.1**

Table 3.6: F1-macro(%): Node classification results. Best and second best values are bold and blue respectively.

Alg.	**CiteSeer**	**CiteSeer-full**	**Cora**	**Cora-ML**	**Cora-full**
MNMF	60.8	68.1	62.7	64.2	32.9
DeepWalk	52.0	57.3	70.2	75.6	48.3
LINE	57.7	60.0	68.3	74.6	42.1
Node2Vec	57.8	61.5	71.4	78.6	48.1
GAE	61.6	79.6	73.5	77.6	41.8
VGAE	62.2	74.4	72.0	76.4	37.7
DGI	67.9	**81.8**	73.3	75.4	21.1
GEMSEC	39.4	53.5	59.4	72.5	38.9
CNRL	53.2	57.9	70.4	78.4	45.9
ComGAN	59.1	64.9	58.5	62.8	29.4
vGraph	32.1	28.5	44.6	62.3	37.6
ComE	63.1	70.2	74.2	79.5	47.8
J-ENC	**68.2**	77.0	**75.6**	**82.0**	**49.6**

Table 3.7: F1-micro(%): Node classification results. Best and second best values are bold and blue respectively.

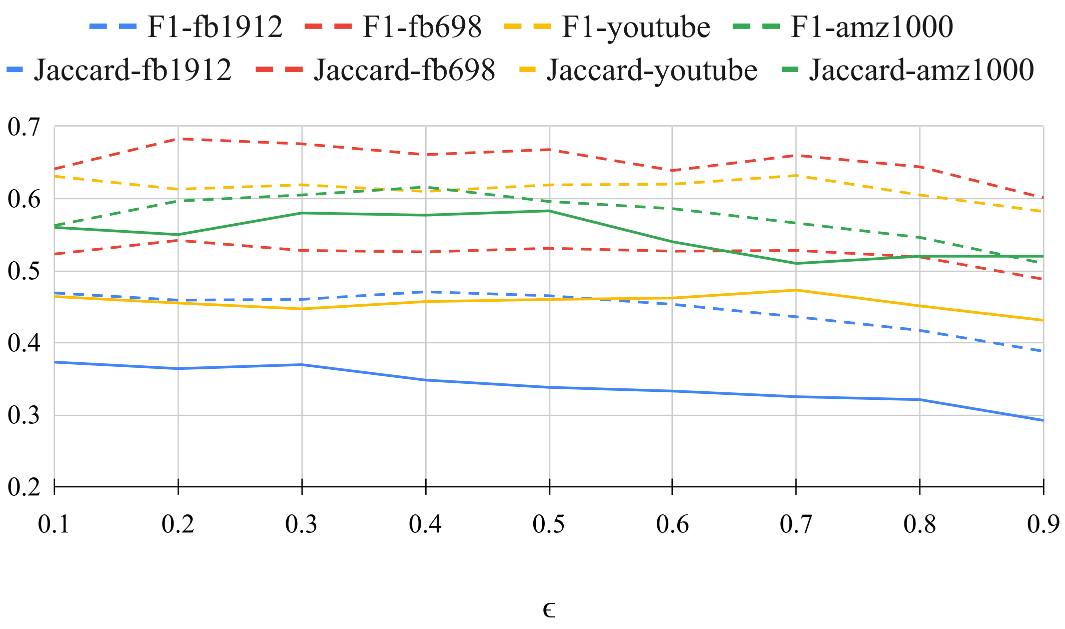

Figure 3.2: Effect of ϵ. Overall a slight decrease in scores can be observed after $\epsilon = 0.7$ mark. F1 and Jaccard scores are in solid and dashed lines respectively.

reason behind it is that the node embeddings are constrained to have same dimensions as the number of communities. Hence, different components of the learned node embeddings simply represent the membership strengths of nodes for different communities. The linear classifiers may find it difficult to separate such vectors.

3.3.6 Hyperparameter Sensitivity

We study the dependence of **J-ENC** on ϵ and α by evaluating on four datasets of different sizes: *fb698*$(N = 61)$, *fb1912*$(N = 747)$, *amazon1000*(N=1540) and *youtube*$(N = 5346)$.

Effect of ϵ: We sweep for $\epsilon = \{0.1, 0.2, \cdots, 0.9\}$. For demonstrating effect of α, we fix $\epsilon = 0.3$ and sweep for $\alpha = \{0.1, 0.2, \cdots, 0.9\}$. The average results of five runs for ϵ and α are given in Fig. 3.2 and Fig. 3.3 respectively. Overall **J-ENC** is quite robust to the change in the values of ϵ and α. In case of ϵ, we see a general trend of decrease in performance when the threshold ϵ is set quite high e.g. $\epsilon > 0.7$. This is because the datasets contain overlapping communities and a very high ϵ will cause the algorithm to give only the most probable community assignment instead of potentially providing multiple communities per node. However, for a large part of sweep space, the results are almost consistent.

Effect of α: When ϵ is fixed and α is changed, the results are mostly consistent except when α is set to a low value. Eq. (3.24) shows that in such a case the node itself is almost neglected and **J-ENC** tends to assign communities based upon neighborhood only, which may cause a decrease in the performance. This effect is most visible in *amazon1000*

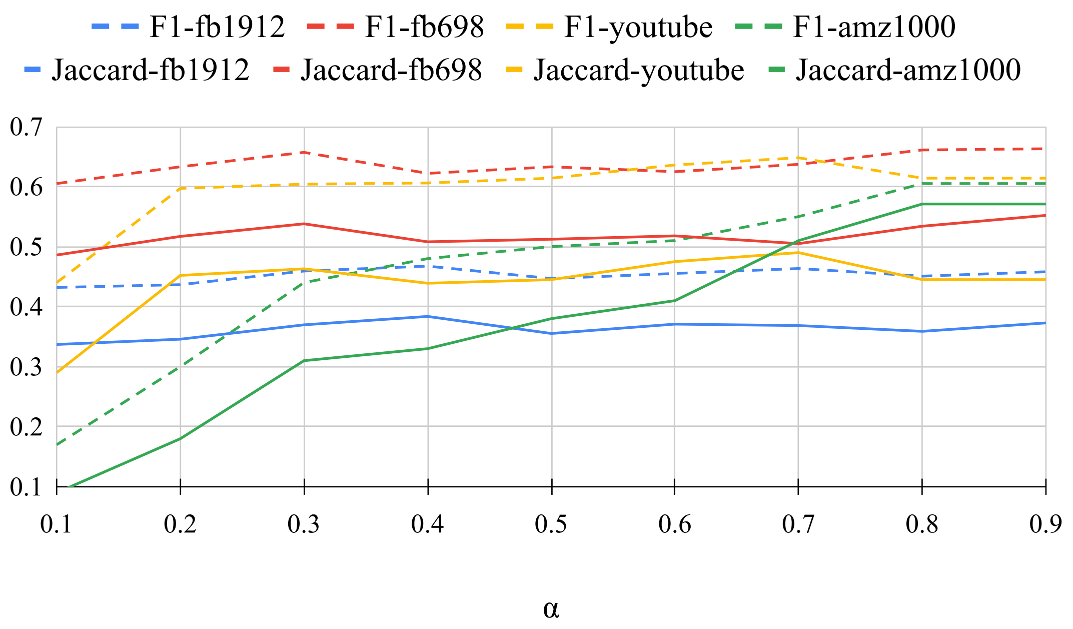

Figure 3.3: Effect of α. The scores generally tend to decrease for small values of α. F1 and Jaccard scores are in solid and dashed lines respectively.

dataset because it has only 1.54 points on average per community. This implies a decent chance for neighbours of a point of being in different communities. Thus, sole dependence on the neighbors will most likely result in poor results.

3.3.7 Training Time

Now we compare the training times of different algorithms in Fig. 3.4. As some of the baselines are more resource intensive than others, we select aws instance type `g4dn.4xlarge` for fair comparison of training times. For vGraph, we train for 1000 iterations and for **J-ENC** for 1500 iterations. For all other algorithms we use the default parameters as used in section 3.3.4. We observe that the methods that simply output the node embeddings take relatively less time compared to the algorithms that jointly learn node representations and community assignments e.g **J-ENC** , vGraph and CNRL. Among these algorithms **J-ENC** is the most time efficient. It consistently trains in less time compared to its direct competitors. For instance, it is about 12 times faster than ComE for *CiteSeer-full* and about 40 times faster compared to vGraph for *Cora-full* dataset. This provides evidence for lower computational complexity of **J-ENC** in Section 3.2.5.

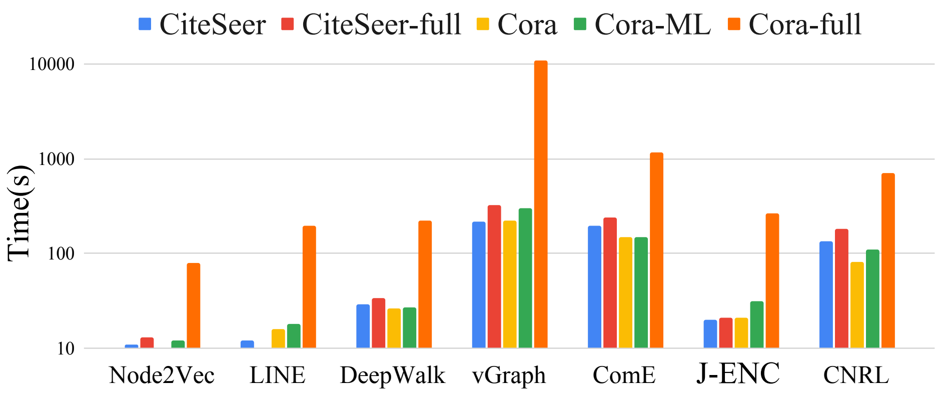

Figure 3.4: Comparison of running times of different algorithms. We can see that **J-ENC** outperforms the direct competitors. The time on y-axis is in log scale.

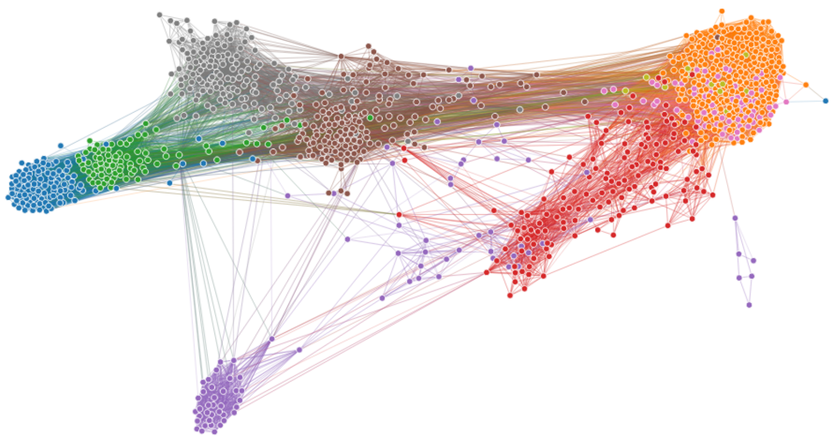

Figure 3.5: fb107. Graph visualization with community assignments (better viewed in color

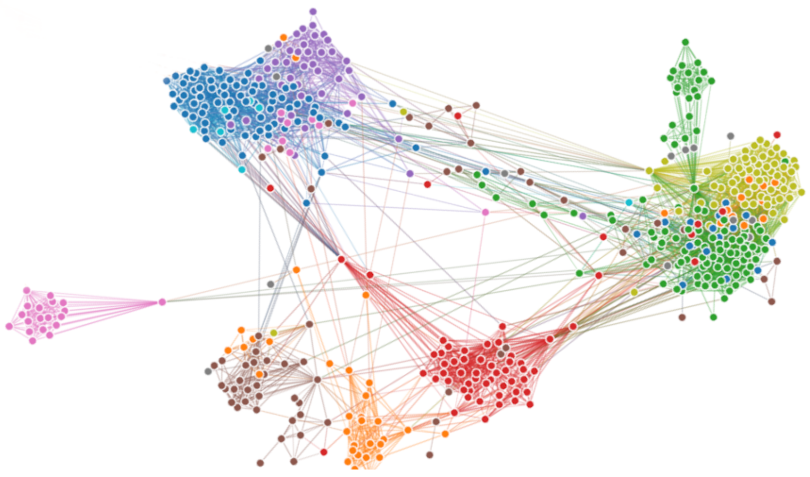

Figure 3.6: fb3437. Graph visualization with community assignments (better viewed in color)

3.3.8 Visualization

Our experiments demonstrate that a single community-aware node embedding is sufficient to aid in both the node representation and community assignment tasks. This is also qualitatively demonstrated by graph visualizations of node embeddings (obtained via t-SNE [80]) and inferred communities for two datasets, fb107 and fb3437, presented in Figures 3.5 and 3.6.

4 Leveraging the Metapath and Entity Aware Subgraphs for Recommendation

This chapter is based on the following publication:

Anwaar, Muhammad Umer[*]*; Han, Zhiwei[*]; Arumugaswamy, Shyam [*]; Khan, Rayyan Ahmad [*]; Weber, Thomas; Qiu, Tianming; Shen, Hao; Liu, Yuanting and Kleinsteuber, Martin. 2022. On Leveraging the Metapath and Entity Aware Subgraphs for Recommendation. In Proceedings of the 1st Workshop on Multimedia Computing towards Fashion Recommendation (MCFR '22). Association for Computing Machinery, New York, NY, USA, 3–10. https://doi.org/10.1145/3552468.3555361*
[*] indicates that the authors contributed equally to this work.

4.1 Introduction

Integrating content information for user preference prediction remains a challenging task in the development of recommender systems. In spite of their effectiveness, most collaborative filtering (CF) methods [81, 82, 83, 84] still suffer from the incapability of modeling content information such as user profiles and item features [85, 86]. Several methods have been proposed to address this problem. Most of them fall in these two categories: factorization and graph-based methods. Factorization methods such as factorization machine (FM) [87], neural factorization machine (NFM) [88] and Wide&Deep models [89] fuse numerical features of each individual training sample. These methods yield competitive performance on several datasets. However, they neglect the dependencies among the content information. Graph-based methods such as NGCF [90], KGAT [91], KGCN [92], Multi-GCCF[93] and LGC [94] represent recommender systems with graph structured data and exploit the graph structure to enhance the node-level representations for better recommendation performance [95, 90, 96, 97].

It is to be noted that learning such node-level representations loses the correspondences and interactions between the content information of users and items. This is because the node embeddings are learned independently as indicated by Zhang et al. [98]. Another disadvantage of previous GNN-based methods is that the sequential nature of connectivity relations are either ignored (Knowledge Graph based methods) or mixed up (GNN-based methods) without the explicit modelling of multi-hop structure.

A natural solution of capturing the inter- and intra-relations between content features and user-item pairs is to explore the high-order information encoded by metapaths

[99, 100]. A metapath denotes a set of composite relations designed for representing multi-hop structure and sequential semantics. To our best knowledge, only a limited number of efforts have been made to enhance GNNs with metapaths. A prominent metapath based method is MAGNN [101]: it aggregates intra-metapath information for each path instance. As a consequence, MAGNN suffers from high memory consumption problem. MEIRec [102] utilizes the structural information in metapaths to improve the node-level representation for intent recommendation, but the method fails to generalize when no user intent or query is available.

To overcome these limitations, we propose Meta**P**ath- and **E**ntity-**A**ware **G**raph **N**eural **N**etwork (PEAGNN), a unified GNN framework, which aggregates information over multiple metapath-aware subgraphs and fuse the aggregated information to obtain node representation using attention mechanism. As a first step, we extract an h-hop enclosing collaborative subgraph (CSG). Each CSG is centered at a user-item pair and aimed to suppress the influence of feature nodes from other user-item interactions. Such local subgraphs contain rich semantic and collaborative information of user-item interactions. One major difference between the CSGs in our work and the subgraphs proposed by Zhang et al. [98] is that in the subgraphs in their work neglect side information by excluding all feature entity nodes.

As a second step, the CSG is decomposed into γ metapath-aware subgraphs based on the schema of the selected metapaths. After that, PEAGNN updates the node representation of the given CSG and outputs a CSG graph-level representation, which distills the collective user-item pattern and sequential semantics encoded in the CSG. A multi-layer perceptron is then trained to predict the recommendation score of a user-item pair. To further exploit the local structure of CSGs, we introduce entity-awareness, a contrastive regularizer which pushes the user and item nodes closer to the connected feature entity nodes, while simultaneously pushing them apart from the unconnected ones. PEAGNN learns by jointly minimizing Bayesian Personalized Rank (BPR) loss and entity-aware loss. Furthermore, PEAGNN can be easily combined with any graph convolution layers such as GAT, GCN and GraphSage.

In contrast to existing metapath based approaches [99, 102, 101, 103, 100], PEAGNN avoids the high computational cost of explicit metapath reconstruction. This is achieved by metapath-guided propagation. The information is propagated along the metapaths "on the fly". This is the primary reason of computational efficiency of PEAGNN as it gets rid of applying message passing on the recommendation graph. Consequently, the redundant information propagated from other interactions (subgraphs) is avoided by only performing metapath-guided propagation on individual metapath-aware subgraph. We discuss this in detail in Sec. 3.

The contributions of our work are summarized as follows:

1. We decouple the sequential semantics conveyed by metapaths into different metapath-aware subgraphs and propose PEAGNN, which explicitly propagates and aggregates multi-hop semantics on metapath-aware subgraphs.

2. We fuse the aggregated information from metapath-aware subgraphs using attention to get representations. For a given CSG, we utilize the graph-level representation and predict the recommendation score of the target user-item pairs.
3. We introduce entity-awareness that acts as a contrastive regularizer on the node embeddings during training.
4. The empirical analysis on three public datasets demonstrate that PEAGNN outperforms other competitive baselines and is capable of learning meaningful metapath combinations.

4.2 Related Work

GNN is designed for learning on graph structured data [104, 105, 106]. GNNs employ message passing algorithm to pass messages in an iterative fashion between nodes to update node representation with the underlying graph structure. An additional pooling layer is typically used to extract graph representation for graph-level tasks, e.g., graph classification or clustering. Due to its superior performance on graphs, GNNs have achieved state-of-the-art performance on node classification [106], graph representation learning [107] and RSs [108]. In the task of RSs, relations such as user-item interactions and user-item features can be presented as multi-typed edges in the graphs. Severel recent works have proposed GNNs to solve recommendation tasks [90, 91, 95]. NGCF [90] embeds bipartite graphs of users and items into node representation to capture collaborative signals. GCMC [95] proposed a graph auto-encoder framework, which produces latent features of users and items through a form of differentiable message passing on the user-item graph. KGAT [91] proposed a knowledge graph based attentive propagation, which enhances the node features by modeling high-order connectivity information. Multi-GCCF [93] explicitly incorporates multiple graphs in the embedding learning process and consider the intrinsic difference between user nodes and item nodes in performing graph convolution.

Prior to GNNs, several efforts have been established to explicitly guide the recommender learning with metapaths [109, 103, 110]. Heitmann et al. [109] utilized linked data from heterogeneous data source to enhance collaborative filtering for the cold-start problem. Sun et al. [103] converted recommendation tasks to relation prediction problems and tackled it with metapath-based relation reasoning. Yu et al. [110] employed matrix factorization framework over meta-path similarity matrices to perform recommendation. Hu et al. [111] proposed user- and item-metapath based co-attention to fuse metapath information for recommendation while ignored the inter-metapath interactions. Zhao et al. [112] fed the semantics encoded by meta-graph in factorization model but neglected the contribution of each individual meta-graph. However, only a limited number of works attempted to enhance GNNs with metapaths. Two recent works are quite prominent in this regard, MAGNN [101] and MEIRec [102]. MAGNN aggregates intra-metapath information for each path instance. But this causes MAGNN to get in the problem of unaffordable memory consumption. MEIRec devised a GNN to perform metapath-

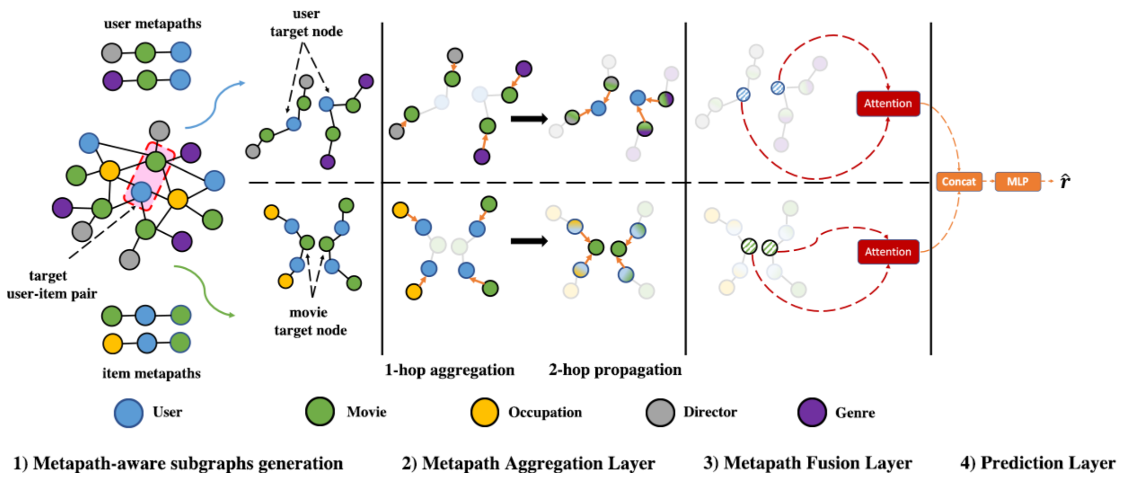

Figure 4.1: Illustration of the proposed PEAGNN model on the MovieLens dataset. Subfigure (1) shows the metapath-aware subgraphs generated from a CSG with the given user- and item metapaths. Subfigures (2), (3) and (4) illustrate the metapath-aware information aggregation and fusion workflow of the PEAGNN model. For simplicity, we have only adopted 2-hop metapaths.

guided propagation. But MEIRec fails to generalize when no user intent is available and does not distinguish the contribution of metapaths. In contrast to these methods, our method saves computational time and memory by adopting a stepwise information propogation over meta-path aware subgraphs. Moreover, PEAGNN employs collaborative subgraph (CSG) to separate semantics introduced by different metapaths and fuses those semantics according to the learned metapath importance, in contrast to existing approaches [98, 101, 102].

4.3 Methodology

4.3.1 Task description

We formulate the recommendation task as: Given a HIN that includes user-item historical interactions as well as their feature entity nodes. The aim is to learn heterogeneous node representations and the graph-level representations from given collaborative subgraphs. The graph-level representations are utilized by a prediction model to predict the interaction score between user-item pair.

4.3.2 Overview of PEAGNN

PEAGNN is a unified GNN framework, which exploits and fuses rich sequential semantics in selected metapaths. To leverage the underlying local structure of the graph for recommendation, we introduce an entity-aware regularizer that distinguishes users and items from their unrelated features in a contrastive fashion. Figure 4.1 illustrates the PEAGNN framework, which consists of three components:

1. A **Metapath Aggregation Layer**, which explicitly aggregates information on metapath-aware subgraphs.
2. A **Metapath Fusion Layer**, which fuses the aggregated node representations from multiple metapath-aware subgraphs using attention mechanism.
3. A **Prediction Layer**, which readouts the graph-level representations of CSGs and estimate the likelihood of potential user-item interactions.

4.3.3 Metapath Aggregation Layer

Sequential semantics encoded by metapaths reveal different aspects towards the connected objects. Appropriate modelling of metapaths can improve the expressiveness of node representations. Our aim is to learn node representations that preserve the sequential semantics in metapaths. PEAGNN saves memory and computation time by performing a step-wise information propagation over metapath-aware subgraphs. This is contrast to Fan et al. [102] which consider each individual path as input.

4.3.3.1 Metapath-aware Subgraph

A metapath-aware subgraph is a directed graph induced from the corresponding CSG by following one specific metapath. As the goal is to learn metapath-aware user-item representation for recommendation, it is intuitive to choose such metapaths which end with either a user or an item node. This ensures that the information aggregation on metapath-aware subgraphs always end on nodes of our primary interest.

4.3.3.2 Information Propagation on Metapath-aware Subgraphs

PEAGNN trains a GNN model to perform step-wise information aggregation on metapath-aware subgraphs. By stacking multiple GNN layers, PEAGNN is capable of not only explicitly exploring the multi-hop connectivity in a metapath but also capturing the collaborative signal effectively. Fig. 4.2 illustrates the flow of information propagation on a given metapath-aware subgraph generated from the metapath mp. Here, $\boldsymbol{X}_{mp,k}$ is the node representations on the metapath mp after kth propagation. $\boldsymbol{A}_{mp,k}$ is the adjacency matrix of the metapath mp at step k. We employ orange and red color to highlight the edges being propagated at a certain aggregation step. Considering the high-order semantic revealed by multi-hop metapaths, we stack multiple GNN layers and recurrently aggregate the representations on the metapaths, so that the high-order semantic is injected into node representations. The metapath-aware information aggregation is shown as follows

$$\boldsymbol{X}_{mp,1} = \sigma(GNN_{mp,1}(\boldsymbol{X}_0, \boldsymbol{A}_{mp,1})), \quad \boldsymbol{X}_{mp,2} = \sigma(GNN_{mp,2}(\boldsymbol{X}_{mp,1}, \boldsymbol{A}_{mp,2})), \tag{4.1}$$

where $\boldsymbol{X}_0$ denotes initial node embeddings. Without loss of generality, by stacking N GNN layers we take into account N-hop neighbours information from the metapath-aware subgraph. Thus, the node representations in the metapath-aware subgraph are given by:

$$\boldsymbol{X}_{mp,n} = \sigma(GNN_{mp,n}(\boldsymbol{X}_{mp,n-1}, \boldsymbol{A}_{mp,n})). \tag{4.2}$$

$\boldsymbol{X}_{mp}$ is the output node representation of the last step on the metapath mp.

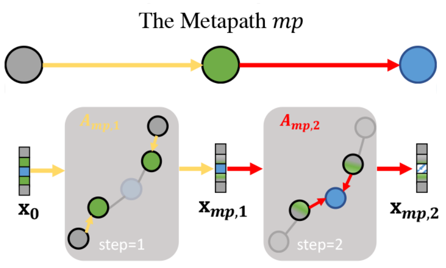

Figure 4.2: Information propagation on a metapath-aware subgraph

4.3.4 Metapath Fusion Layer

After information aggregation within metapath-aware subgraphs, the metapath fusion layer combines and fuses the semantic information revealed by all metapaths. Assume for a node v, a set of its node representations $\{\mathbf{x}^v_{mp_1}, \mathbf{x}^v_{mp_2}, ..., \mathbf{x}^v_{mp_\gamma}\}$ is aggregated from γ metapaths. Semantics disclosed by metapaths are not of equal importance to node representations and the contribution of every metapath should also be adjusted accordingly. Therefore, we leverage soft attention to learn the importance of each metapath, instead of adopting element-wise *mean*, *max* and *add* operators. It is to be noted that PEAGNN applies a node-wise attentive fusion of metapath aggregated node representation.This is contrast to previous works which employ a fixed attention factor for all nodes. Consequently, they fail to capture the node-specific metapath preference. For a given target node v, we apply vector concatenation on its representations from γ metapath-aware subgraphs, denoted as $\mathbf{H}_v = [\mathbf{x}^v_{mp_1}; \mathbf{x}^v_{mp_2}; ...; \mathbf{x}^v_{mp_\gamma}]$. The metapath fusion is performed as follows:

$$\mathbf{c}_v = trace(\mathbf{W}^T \mathbf{H}_v), \tag{4.3}$$

where $\mathbf{c}_v$ is a vector of metapath importance and $\mathbf{W}$ is a matrix with learnable parameters. We then normalize the metapath importance score using softmax function and get the attention factor for each metapath:

$$\text{att}^v_{mp_i} = \frac{\exp(c_v^{mp_i})}{\sum_{j=1}^{\gamma} \exp(c_v^{mp_j})}, \tag{4.4}$$

where $\text{att}^v_{mp_i}$ denotes the normalized attention factor of metapath mp_i on the node v. With the learned attention factors, we can fuse all metapath aggregated node representations to the final metapath-aware node representation, $\mathbf{e}_v$, as:

$$\mathbf{e}_v = \sum_{i=1}^{\gamma} \text{att}^v_{mp_i} \mathbf{x}^v_{mp_i}. \tag{4.5}$$

4.3.5 Prediction Layer

Next, we readout the node representations of CSGs into a graph-level feature vector. In existing works, many pooling methods were investigated such as SumPool, MeanPooling, SortPooling [113] and DiffPooling [114]. However, we adopt a different pooling strategy

Algorithm 2 The training algorithm of PEAGNN

```
1:  Input: A HIN $G$, number of iterations $M$, $\gamma$-metapaths
2:  Output: Learned node representations $\boldsymbol{E}$, a prediction model $\tilde{r}$
3:  for $i = 1$ to $M$ do
4:      Sample a batch $\mathcal{B}$ of training user-item interactions
5:      Construct the CSGs of $\mathcal{B}$ as illustrated in the figure 4.1
6:      Sample the Metapath-aware Subgraphs from the constructed CSGs with the given
        $\gamma$ metapaths
7:      Perform step-wise information propagation over the Metapath-aware Subgraphs as
        per eqs. 4.1 and 4.2
8:      For each node, obtain the metapath-aware node representation by fusing the
        representation from $\gamma$ metapaths with attention factors (see eqs. 4.4 and 4.5)
9:      Use the graph-level representation to calculate the interaction scores of user $u$ and
        item $i$ via eqs. 4.6 and 4.7
10:     Compute the training loss with eqs. 4.8, 4.9 and 4.10 and update model parameters
11: end for
```

which concatenates the aggregated representations of the center user node $\mathbf{e}_u$ and item node $\mathbf{e}_i$ in the CSGs. i.e.,

$$\mathbf{e_g} = \text{concat}(\mathbf{e}_u, \mathbf{e}_i). \tag{4.6}$$

After obtaining the graph-level representation of CSG, we utilize a 2-layer MLP to compute the matching score of a user-item pair. Lets denote a CSG with $\mathcal{G}_{u,i}$, the prediction function for the interaction score of user u and item i can be expressed as follows

$$\tilde{r}(\mathcal{G}_{u,i}) = \mathbf{w}_2^T \sigma(\mathbf{w}_1^T \mathbf{e}_g + \mathbf{b}_1) + \mathbf{b}_2, \tag{4.7}$$

where $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the trainable parameters of the MLP which map the graph-level representation $\mathbf{e}_g$ to a scalar matching score, and σ is the non-linear activation function (e.g. ReLU).

4.3.6 Graph-level representation for recommendation

Compared to the previous GNN-based methods such as NGCF, KGAT, KGCN, Multi-GCCF and LGC that use node-level representations for recommendation, PEAGNN predicts the matching score of a user-item pair by mapping its corresponding metapath-aware subgraph to a scalar as shown in Fig. 4.1. As shown by [98], methods using node-level representation suffer from the over-smoothness problem [115][106]. As their node-level representations are learned independently, they fail to model the correspondence of the structural proximity of a node pair. On the other hand, a graph-level GNN with sufficient rounds of message passing can better capture the interactions between the local structures of two nodes [116].

4.3.7 Training Objective

To train model parameters in an end-to-end manner, we minimize the pairwise Bayesian Personalized Rank (BPR) loss [117], which has been widely used in RSs. The BPR loss can be expressed as follows:

$$\mathcal{L}_{CF} = \sum_{(u,i_+,i_-)\in\mathcal{O}} -\ln\sigma\Big(\tilde{r}(u,i_+) - \tilde{r}(u,i_-)\Big), \tag{4.8}$$

where $\mathcal{O} = \{(u,i_+,i_-)|(u,i_+) \in R_+, (u,i_-) \in R_-\}$ is the training set, R_+ is the observed user-item interactions (positive samples) while R_- is the unobserved user-item interactions (negative samples). The detailed training procedure is illustrated in the Algorithm 2.

Although user and item representations can be derived by information aggregation and fusion on metapath-aware subgraphs, the local structural proximity of user(item) nodes are still missing. Towards this end, we propose ***Entity-Awareness*** to regularize the local structural of user(item) nodes. The idea of *entity-awareness* is to distinguish items or users with their unrelated feature entities in the embedding space. Specifically, *entity-awareness* is a distance-based contrastive regularization term that pulls the related feature entity nodes closer to the corresponding user(item) nodes, while push the unrelated ones apart. The regularization term is defined as following:

$$\begin{aligned}\mathcal{L}_{Entity} = \sum_{(u,i_+,i_-)\in\mathcal{O}} -ln\sigma\Bigg[&\Big(d(\mathbf{x}_u,\mathbf{x}_{f-,u}) - d(\mathbf{x}_u,\mathbf{x}_{f+,u})\Big)\\ &+\Big(d(\mathbf{x}_{i_+},\mathbf{x}_{f-,i_+}) - d(\mathbf{x}_{i_+},\mathbf{x}_{f+,i_+})\Big)\Bigg],\end{aligned} \tag{4.9}$$

where $\mathbf{x}_{f+,u}, \mathbf{x}_{f-,u}$ denote the observed and unobserved feature entity embeddings of user u, $\mathbf{x}_{f+,i_+}, \mathbf{x}_{f-,i_-}$ denote the observed and unobserved feature entity embeddings of positive item i and $d(\cdot,\cdot)$ is a distance measure on the embedding space. The total loss is computed by the weighted sum of these two losses. It is given by:

$$\mathcal{L} = \mathcal{L}_{CF} + \lambda\mathcal{L}_{Entity}, \tag{4.10}$$

where λ is the weight of the *entity-awareness* term. We use mini-batch Adam optimizer [118]. For a batch randomly sampled from training set $\mathcal{O}$, we establish their representation by performing information aggregation and fusion on their embeddings, and then update model parameters via back propagation.

4.4 Experiments

We evaluate the effectiveness our approach via experiments on public datasets. Our experiments aim to address the following research questions:

- **RQ1**: How does PEAGNN perform compared to other baseline methods?
- **RQ2**: How does the *entity-awareness* affect the performance of PEAGNN?
- **RQ3**: What is the impact of different metapaths in recommendation tasks?

	MovieLens-small	**MovieLens-25M**	**Yelp**
#Nodes	2933	33249	89252
#Users	608	14982	60808
#Items	2121	11560	28237
#Interactions	79619	1270237	754425

Table 4.1: Statistics of datasets

4.4.1 Experimental Settings

4.4.1.1 Datasets

The datasets which we included in our experimental evaluation are widely used in related works. That is, Movielens [95, 119, 88] and Yelp [91, 96]. By changing the size of Movielens from small to large, we investigated the effect of dataset scale on the performance of the proposed method and the competitive baselines. We have three datasets of different sizes, namely: MovieLens-small (small), Yelp (medium) and MovieLens-25M (large). The statistics of the three datasets are summarized in Table 4.1.

MovieLens[1] is widely used benchmark dataset for movie recommendation. We use small ($\sim$ 100k ratings) and 25M ($\sim$ 25 million ratings) versions of the dataset. We consider movies as items and ratings as interactions. For ML-small, we use 10-core setting i.e. each user and item will have at least 10 interactions. For ML-25M, we select items which have at least 10 interactions and users with 10 to 300 interactions (from 2018 on-wards) to ensure dataset quality.

Yelp[2] is used for business recommendations and has around 10 million interactions. Here we consider businesses as items; reviews and tips as interactions. The original dataset is highly sparse. So, to ensure dataset quality, we select items which have at least 50 interactions and users with 10 to 20 interactions.

Along with user-item interactions, we use their features as entities to build HIN graph. For MovieLens, we employ user feature of tag and item features like year, genre, actor, director, writer etc. For Yelp, we extract user features like counts of reviews, friends, fans, stars and item features like attributes, categories and counts of stars, reviews and check-ins.

4.4.1.2 Evaluation Strategy and Metrics

Leaving one interaction out evaluation strategy is one of the most commonly followed approach to evaluate recommender systems [88, 117, 120, 121]. Recent research shows that different data splits have a huge impact on the final performance[122]. To avoid the raised concerns in evaluating GNN based methods, we follow [82] and adapt leave-one-out evaluation for a more stable, reliable and fair comparison. For each user, we set the latest interacted item as the test set. The remaining items are employed for training. For each user-item positive interaction in the training set, we employ negative sampling

[1] `https://grouplens.org/datasets/movielens/`
[2] `https://www.yelp.com/dataset`

strategy to get four negative items for that user. For Yelp and ML-25M, we sample randomly while for ML-small, we sample from the unseen items for each user.We use two evaluation metrics: Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). We consider only top-10 positions of the returned results. HR@10 indicates whether the test item is present in top-10 recommendations. NDCG@10 also takes in to account the position at which the correct item appears in the recommendations [119]. We compute the metrics for each test user and report the average score.

4.4.1.3 Hyperparameter Settings

We implemented the PEAGNN and baselines in Pytorch Geometric 1.5.0 [123]. To determine hyper-parameters of our methods, we follow the procedure proposed in [82]. For each user, one random interaction is sampled as the validation data for parameter tuning. We cross validated the batch size of [1024, 2048, 4096], the learning rate of [0.0001 ,0.0005, 0.001, 0.005] and weight of [0.03, 0.1, 0.3, 1] for entity-awareness. For fair comparison, we employed the same embedding dimension for both PEAGNN and the baselines. We set embedding dimension to 64 across all models and datasets. The representation dimension is 16 for GNN-based models and hidden layer size is 64 for factorization and GNN-based models. We use 2-step metapaths and attention channel aggregation for PEAGNN. The number of metapaths, γ, for ML-small, ML-25M and Yelp are 9, 13 and 11 respectively.

Further implementation details of PEAGNN as well as baseline models can be found in the code[3].

4.4.1.4 Baselines

We compare PEAGNN with several kinds of competitive baselines.
1. NFM [88] utilizes neural networks to enhance high-order feature interactions with non-linearity. As suggested by He et al. [88], we apply one hidden layer neural network on input features.
2. CFKG [124] applies TransE [125] to learn heterogeneous node embedding and converts recommendation to a link prediction problem.
3. HeRec [99] extends the matrix factorization model with the joint learning of a set of embedding fusion functions.
4. Metapath2Vec [99] utilizes a skip-gram model to update the node embeddings generataed by metapath-guided random walks. We then use a MLP to predict the matching score with the learned embeddings.
5. NGCF [90] integrates the user-item bipartite graph structure into the embedding process for collaborative filtering.
6. KGCN [92] exploits multi-hop proximity information with a receptive field to learn user preference.
7. KGAT [91] incorporates high-order information by performing attentive embedding propagation with the learned entity attention on knowledge graph.

[3] `https://github.com/ecml-peagnn/PEAGNN`

8. Multi-GCCF [93] explicitly incorporates multiple graphs in the embedding learning process. Multi-GCCF not only models the high-order information but also integrates the proximal information of item-item and user-user paris.
9. LGC [94] simplifies the design of GCN by maintaining only the neighborhood aggregation for collaborative filtering.

4.4.2 Overall Performance Comparison (RQ1)

Table 4.2 summarizes the performance comparison of PEAGNN variants and the competitive baselines. In the following, we discuss these results to gain some important insights into the problem.

First, we note that our proposed method PEAGNN outperforms other methods by a significant margin on all three datasets. In particular, the performance gains achieved by PEAGNN are 7.87%, 2.39%, and 8.23% w.r.t. NDCG@10 on ML-small, ML-25m and Yelp datasets, respectively. The primary reason for outstanding performance of PEAGNN is that none of the existing methods do explicit modelling consequently all node messages get mixed up during message passing. This verifies our claim that explicit modelling and fusing sequential semantics in metapath help in better learning of user-item interactions. Moreover, the superior performance of PEAGNN also reveals the effectiveness of modelling the local graph structure, while other GNN-based methods simply pay no attention to their structure proximity.

Second, we observe that the path-based methods that are not based on GNN significantly underperform all the GNN-based models on all three datasets. Both Metapath2Vec+MLP and HeRec have incorporated the static node embedding using Metapath2vec [99]. The poor performance indicates that learning unsupervised static node embeddings have limited the power of the model to capture the complex collaborative signals and intricate content relations.

Third, we observe that CFKG and NFM prove to be strong baselines for GNN based methods especially on ML-25m and Yelp datasets. For instance, CFKG achieves 0.8729 HR@10 on Yelp and is fourth best performing model. But the performance gap with second and third best models is negligible i.e. 0.4% and 0.2% respectively. CFKG is significantly outperformed only by our method (PEAGAT*) by 4.37%. This demonstrates that our method is able to better leverage the information in the graph-structured data by explicitly modelling the the sequential semantics via metapath aware subgraphs.

4.4.3 Effect of Entity-awareness (RQ2)

The goal of introducing entity-awareness is to take advantage of the first-order structure of CSG, which is not well exploited by pure message passing in graph-based RSs [126]. We study the effect of entity-awareness by comparing the performances of our models with and without entity-awareness. The effect of entity-awareness for different base models are summarized in Table 4.2. Generally, entity-awareness delivers consistently better performance than PEAGNN without entity-awareness. In particular, a more significant performance gain has been observed in the smaller dataset ML-small with a minimum

Model	MovieLens-small		MovieLens-25M		Yelp	
	HR@10	**NDCG@10**	**HR@10**	**NDCG@10**	**HR@10**	**NDCG@10**
NFM	0.477	0.2668	0.8132	0.5347	0.8595	0.6062
CFKG	0.4378	0.2381	0.8152	0.5196	0.8729	0.5826
HeRec	0.2668	0.1449	0.607	0.3291	0.5533	0.3302
Metapath2Vec	0.3063	0.1614	0.7956	0.5051	0.6307	0.402
NGCF	0.5016	0.2755	0.7807	0.4866	0.8068	0.481
KGCN	0.5132	0.2788	0.7771	0.4699	0.8125	0.4668
KGAT	0.5214	0.2846	0.8147	0.5236	0.8762	0.6136
MultiGCCF	0.5230	0.2836	0.8014	0.5153	0.8639	0.6120
LGC	0.5003	0.2815	0.8081	0.5237	0.8744	0.6122
PEAGCN	0.5382	0.2951	0.8185	0.5344	0.9041	0.6379
PEAGCN*	0.5576	0.3036	0.8187	0.5361	0.9125	0.6443
(% improv. w.r.t. best competitor)	6.62%	6.68%	0.43%	0.26%	4.14%	5.00%
PEAGAT	0.5375	0.2983	0.8249	0.5414	0.9057	0.6382
PEAGAT*	0.5477	0.3045	**0.8284**	**0.5475**	**0.9128**	**0.6641**
(% improv. w.r.t. best competitor)	4.72%	6.99%	**1.62%**	**2.39%**	**4.18%**	**8.23%**
PEASage	0.5444	0.3003	0.8176	0.5383	0.8772	0.6247
PEASage*	**0.5609**	**0.307**	0.8273	0.5462	0.8837	0.6308
(% improv. w.r.t. best competitor)	**7.25%**	**7.87%**	1.48%	2.15%	0.86%	2.80%

Table 4.2: Overall Performance Comparison. The scores are average of five runs. Bold indicates best results for the dataset. * denotes entity-awareness.

improvement of 1.9% on HR@10. On the other hand, models with entity-awareness slightly outperform base models on the larger and denser datasets. It indicates that leveraging local structure on sparse datasets proves beneficial. Nonetheless, NDCG@10 benefits more from the entity-awareness in comparison with HR@10 on both MovieLens and Yelp datasets. For instance, on Yelp, PEAGAT shows 4.06% performance gain in terms of NDCG@10. These results signify the importance of explicit modelling of first-order relations in RSs.

4.4.4 Effect of Metapaths (RQ3)

We have conducted various ablation studies, In order to gain insight into the effect of different metapaths on the performance of PEAGNN, we have conducted various experiments. In the interest of space, we only include the results of our best model PEASage on ML-small dataset. We have total 9 metapaths for ML-small dataset. These metapaths are shown as columns in the Table 4.3. We drop only 1 specific metapath at a time and then compare the model performance drop with the original one. The table summarizes the percentage performance drop as compared to the original model. We ran the experiments 5 times with different random seeds for fair evaluation.

First we observe that, there are some metapaths droping which results in significant decrease in the model performance. That is, PEASage learnt three different key metapaths combinations. Namely: U-M-U, M-U-M and Y-M-U. Second, we observe that each

#	U-M-U	M-U-M	Y-M-U	A-M-U	W-M-U	D-M-U	G-M-U	T-M-U	T-U-M
1	**-27.45**	-4.41	**-24.3**	-2.21	-1.9	-0.96	-0.96	-2.53	-6
2	**-30.31**	**-46.98**	**-11.51**	-6.37	-7.28	-8.18	-5.16	-9.71	-5.45
3	-1.88	-5.3	**-40.62**	-4.37	-3.12	-0.3	-0.63	-4.69	**-15.32**
4	-6.38	**-48.77**	**-23.63**	+1.53	-1.85	-0.62	+0.3	-2.06	-7.67
5	-8.97	**-42.24**	**-49.38**	-4.34	-3.1	-1.55	-5.29	-1.55	-4.04

Table 4.3: Percentage drop in the performance of PEASage on ML-small w.r.t. HR@10 when one metapth is removed during training. Bold indicates greater than 10% drop in performance. (Abbreviation for nodes: U-User, M-Movie, Y-Year, A-Actor, W-Writer, D-Director, G-Genre and T-Tag)

metapath is contributing something in the performance of PEASage although the effect of six metapaths is not that significant.

Third, we note that the published year of movies has the most significant impact on users' choice. That is, the metapath Y-M-U comes out as key metapath in all 5 runs. Those metapaths which capture collaborative effects such as U-M-U and M-U-M take critical role in the high performance of PEASage model. Another interesting phenomenon, which warrants further investigation, is that the tag given by users might have a complementary relation of user-item interactions as shown in the third run in the table.

These results also indicate another strength of PEAGNN, i.e., even without prior knowledge and careful selection, the effectiveness of each metapath and different metapath combinations can be verified in a convenient way by comparing attention factors or "disabling" specific metapath. Thus, an incremental training and metapath selection is achievable. Therefore, more insightful research for HCI community can be expected.

Part II

Multi-Modal Data

5 Compositional Learning of Image-Text Query for Image Retrieval

This chapter is based on the following peer-reviewed publication:

Anwaar, Muhammad Umer[*]*; Labintcev, Egor [*] and Kleinsteuber, Martin. Compositional Learning of Image-Text Query for Image Retrieval. In IEEE Winter Conference on Applications of Computer Vision (WACV '21). 2021.*
[*] indicates that the authors contributed equally to this work.

5.1 Introduction

One of the peculiar features of human perception is multi-modality. We unconsciously attach attributes to objects, which can sometimes uniquely identify them. For instance, when a person says *apple* it is quite natural that an image of an apple, which may be green or red in color, forms in their mind. In information retrieval, the user seeks information from a retrieval system by sending a query. Traditional information retrieval systems allow a unimodal query, i.e., either a text or an image. Advanced information retrieval systems should enable the users in expressing the concept in their mind by allowing a multi-modal query.

In this work, we consider such an advanced retrieval system, where users can retrieve images from a database based on a multi-modal query. Concretely, we have an image retrieval task where the input query is specified in the form of an image and natural language expressions describing the desired modifications in the query image. Such a retrieval system offers a natural and effective interface [40]. This task has applications in the domain of E-Commerce search, surveillance systems and internet search. Fig. 5.1 shows a potential application scenario of this task.

Recently, Vo *et al.*[127] have proposed the *Text Image Residual Gating* (TIRG) method for composing the query image and text for image retrieval. They have achieved state-of-the-art (SOTA) results on this task. However, their approach does not perform well for real-world application scenarios, i.e. with long and detailed texts (see Sec. 5.4.4). We think the reason is that their approach is too focused on changing the image space and does not give the query text its due importance. The gating connection takes element-wise product of query image features with image-text representation after passing it through two fully connected layers. In short, TIRG assigns huge importance to query image features by putting it directly in the final composed representation. Similar to [128, 129],

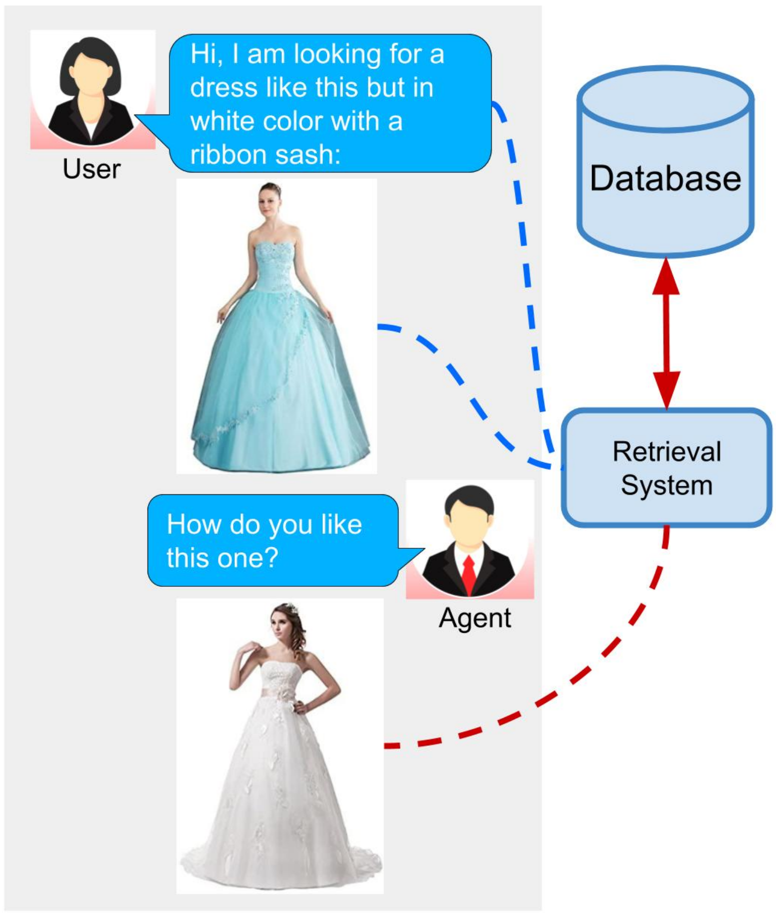
User
Hi, I am looking for a dress like this but in white color with a ribbon sash:
Database
Retrieval System
How do you like this one?
Agent

Figure 5.1: Potential application scenario of this task

they employ LSTM for extracting features from the query text. This works fine for simple queries but fails for more realistic queries.

In this paper, we attempt to overcome these limitations by proposing ComposeAE, an autoencoder based approach for composing the modalities in the multi-modal query. We employ a pre-trained BERT model [130] for extracting text features, instead of LSTM. We hypothesize that by jointly conditioning on both left and right context, BERT is able to give better representation for the complex queries. Similar to TIRG [127], we use a pre-trained ResNet-17 model for extracting image features. The extracted image and text features have different statistical properties as they are extracted from independent uni-modal models. We argue that it will not be beneficial to fuse them by passing through a few fully connected layers, as typically done in image-text joint embeddings [131].

We adopt a novel approach and map these features to a complex space. We propose that the target image representation is an element-wise rotation of the representation of the source image in this complex space. The information about the degree of rotation is specified by the text features. We learn the composition of these complex vectors and their mapping to the target image space by adopting a deep metric learning (DML) approach. In this formulation, text features take a central role in defining the relationship between query image and target image. This also implies that the search space for learning the composition features is restricted. From a DML point of view, this restriction proves to be quite vital in learning a good similarity metric.

We also propose an explicit rotational symmetry constraint on the optimization problem based on our novel formulation of composing the image and text features. Specifically, we require that multiplication of the target image features with the complex conjugate of the query text features should yield a representation similar to the query image features. We explore the effectiveness of this constraint in our experiments (see Sec. 5.4.6).

We validate the effectiveness of our approach on three datasets: MIT-States, Fashion200k and Fashion IQ. In Sec. 5.4, we show empirically that ComposeAE is able to learn a better composition of image and text queries and outperforms SOTA method. In DML, it has been recently shown that improvements in reported results are exaggerated and performance comparisons are done unfairly [132]. In our experiments, we took special care to ensure fair comparison. For instance, we introduce several variants of TIRG. Some of them show huge improvements over the original TIRG. We also conduct several ablation studies to quantify the contribution of different modules in the improvement of the ComposeAE performance.

Our main contributions are summarized below:

- We propose a ComposeAE model to learn the composed representation of image and text query.

- We adopt a novel approach and argue that the source image and the target image lie in a common complex space. They are rotations of each other and the degree of rotation is encoded via query text features.
- We propose a rotational symmetry constraint on the optimization problem.
- ComposeAE outperforms the SOTA method TIRG by a huge margin, i.e., 30.12% on Fashion200k and 11.13% on MIT-States on the Recall@10 metric.
- We enhance SOTA method TIRG [127] to ensure fair comparison and identify its limitations.

5.2 Related Work

Deep metric learning (DML) has become a popular technique for solving retrieval problems. DML aims to learn a metric such that the distances between samples of the same class are smaller than the distances between the samples of different classes. The task where DML has been employed extensively is the cross-modal retrieval, i.e. retrieving images based on text query and getting captions from the database based on the image query [131, 133, 134, 135, 136, 137].

In the domain of Visual Question Answering (VQA), many methods have been proposed to fuse the text and image inputs [128, 138, 139]. We review below a few closely related methods. Relationship [128] is a method based on relational reasoning. Image features are extracted from CNN and text features from LSTM to create a set of relationship features. These features are then passed through a MLP and after averaging them the composed representation is obtained. FiLM [138] method tries to “influence” the source image by applying an affine transformation to the output of a hidden layer in the network. In order to perform complex operations, this linear transformation needs to be applied to several hidden layers. Another prominent method is parameter hashing [139] where one of the fully-connected layers in a CNN acts as the dynamic parameter layer.

In this work, we focus on the image retrieval problem based on the image and text query. This task has been studied recently by Vo *et al.*[127]. They propose a gated feature connection in order to keep the composed representation of query image and text in the same space as that of the target image. They also incorporate a residual connection which learns the similarity between concatenation of image-text features and the target image features. Another simple but effective approach is Show and Tell[129]. They train a LSTM to predict the next word in the sequence after it has seen the image and previous words. The final state of this LSTM is considered the composed representation. Han *et al.*[140] presents an interesting approach to learn spatially-aware attributes from product description and then use them to retrieve products from the database. But their text query is limited to a predefined set of attributes. Nagarajan *et al.*[141] proposed an embedding approach, “Attribute as Operator”, where text query is embedded as a transformation matrix. The image features are then transformed with this matrix to get the composed representation.

This task is also closely related with interactive image retrieval task [142, 143] and attribute-based product retrieval task [144, 127]. These approaches have their limitations such as that the query texts are limited to a fixed set of relative attributes [144], require multiple rounds of natural language queries as input [142, 143] or that query texts can be only one word i.e. an attribute [140]. In contrast, the input query text in our approach is not limited to a fixed set of attributes and does not require multiple interactions with the user. Different from our work, the focus of these methods is on modeling the interaction between user and the agent.

5.3 Methodology

5.3.1 Problem Formulation

Let $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ denote the set of query images, $\mathcal{T} = \{t_1, t_2, \cdots, t_n\}$ denote the set of query texts and $\mathcal{Y} = \{y_1, y_2, \cdots, y_n\}$ denote the set of target images. Let $\psi(\cdot)$ denote the pre-trained image model, which takes an image as input and returns image features in a d-dimensional space. Let $\kappa(\cdot, \cdot)$ denote the similarity kernel, which we implement as a dot product between its inputs. The task is to learn a composed representation of the image-text query, denoted by $g(x, t; \Theta)$, by maximising

$$\max_{\Theta} \kappa(g(x, t; \Theta), \psi(y)), \tag{5.1}$$

where Θ denotes all the network parameters.

5.3.2 Motivation for Complex Projection

In deep learning, researchers aim to formulate the learning problem in such a way that the solution space is restricted in a meaningful way. This helps in learning better and robust representations. The objective function (Equation 5.1) maximizes the similarity between the output of the composition function of the image-text query and the target image features. Thus, it is intuitive to model the query image, query text and target image lying in some common space. One drawback of TIRG is that it does not emphasize the importance of text features in defining the relationship between the query image and the target image.

Based on these insights, we restrict the compositional learning of query image and text features in such a way that: (i) query and target image features lie in the same space, (ii) text features encode the transition from query image to target image in this space and (iii) transition is symmetric, i.e. *some function of the text features* must encode the reverse transition from target image to query image.

In order to incorporate these characteristics in the composed representation, we propose that the query image and target image are rotations (transitions) of each other in a complex space. The rotation is determined by the text features. This enables incorporating the desired text information about the image in the common complex space. The reason for choosing the complex space is that *some function of text features* required

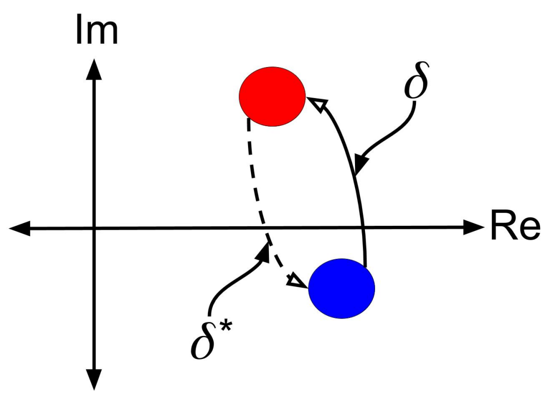

Figure 5.2: Conceptual Diagram of Rotation of the Images in Complex Space. Blue and Red Circle represent the query and the target image respectively. δ represents the rotation in the complex space, learned from the query text features. δ^* represents the complex conjugate of the rotation in the complex space.

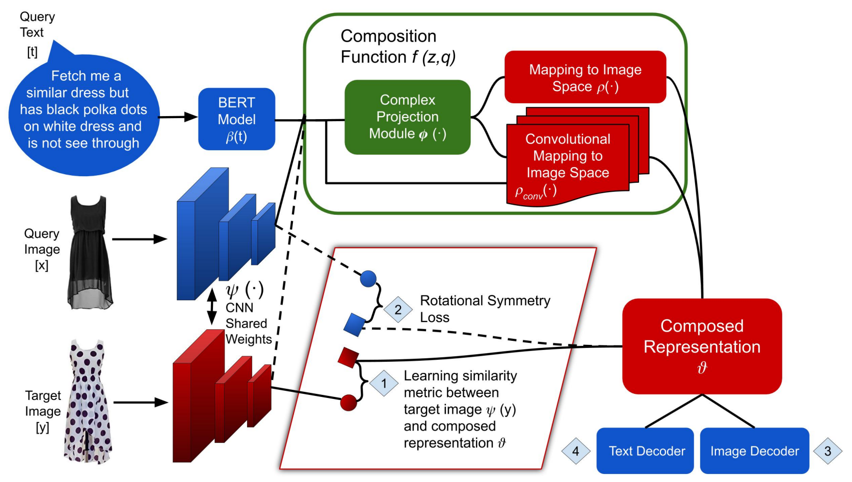
Query Text [t]
Fetch me a similar dress but has black polka dots on white dress and is not see through
BERT Model β(t)
Composition Function *f (z,q)*
Complex Projection Module φ (·)
Mapping to Image Space ρ(·)
Convolutional Mapping to Image Space ρ_{conv}(·)
Query Image [x]
ψ (·) CNN Shared Weights
Target Image [y]
2 Rotational Symmetry Loss
1 Learning similarity metric between target image ψ (y) and composed representation ϑ
Composed Representation ϑ
4 Text Decoder
Image Decoder 3

Figure 5.3: ComposeAE Architecture: Image retrieval using text and image query. Dotted lines indicate connections needed for calculating *rotational symmetry loss* (see Equations 5.12, 5.13 and 5.14). Here 1 refers to L_{BASE}, 2 refers to L_{SYM}^{BASE}, 3 refers to L_{RI} and 4 refers to L_{RT}.

for the transition to be symmetric can easily be defined as the complex conjugate of the text features in the complex space (see Fig. 5.2).

Choosing such projection also enables us to define a constraint on the optimization problem, referred to as *rotational symmetry constraint* (see Equations 5.12, 5.13 and 5.14). We will empirically verify the effectiveness of this constraint in learning better composed representations. We will also explore the effect on performance if we fuse image and text information in the real space. Refer to Sec. 5.4.6.

An advantage of modelling the reverse transition in this way is that we do not require captions of query image. This is quite useful in practice, since a user-friendly retrieval system will not ask the user to describe the query image for it. In the public datasets, query image captions are not always available, e.g. for Fashion IQ dataset. In addition to that, it also forces the model to learn a good "internal" representation of the text features in the complex space.

Interestingly, such restrictions on the learning problem serve as implicit regularization. e.g., the text features only influence angles of the composed representation. This is in line with recent developments in deep learning theory [145, 146]. Neyshabur *et al.*[147] showed that imposing simple but global constraints on the parameter space of deep networks is an effective way of analyzing learning theoretic properties and aids in decreasing the generalization error.

5.3.3 Network Architecture

Now we describe ComposeAE, an autoencoder based approach for composing the modalities in the multi-modal query. Figure 5.3 presents the overview of the ComposeAE architecture.

For the image query, we extract the image feature vector living in a d-dimensional space, using the image model $\psi(\cdot)$ (e.g. ResNet-17), which we denote as:

$$\psi(x) = z \in \mathbb{R}^d. \tag{5.2}$$

Similarly, for the text query t, we extract the text feature vector living in an h-dimensional space, using the BERT model [130], $\beta(\cdot)$ as:

$$\beta(t) = q \in \mathbb{R}^h. \tag{5.3}$$

Since the image features z and text features q are extracted from independent unimodal models; they have different statistical properties and follow complex distributions. Typically in image-text joint embeddings [127, 131], these features are combined using fully connected layers or gating mechanisms.

In contrast to this we propose that the source image and target image are rotations of each other in some complex space, say, $\mathbb{C}^k$. Specifically, the target image representation is an element-wise rotation of the representation of the source image in this complex space. The information of how much rotation is needed to get from source to target image is encoded via the query text features. During training, we learn the appropriate mapping functions which give us the composition of z and q in $\mathbb{C}^k$.

More precisely, to model the text features q as specifying element-wise rotation of source image features, we learn a mapping $\gamma \colon \mathbb{R}^k \to \{D \in \mathbb{R}^{k\times k} \mid D \text{ is diagonal}\}$ and obtain the coordinate-wise complex rotations via

$$\delta = \mathcal{E}\{j\gamma(q)\},$$

where $\mathcal{E}$ denotes the matrix exponential function and j is square root of -1. The mapping γ is implemented as a multilayer perceptron (MLP) i.e. two fully-connected layers with non-linear activation.

Next, we learn a mapping function, $\eta : \mathbb{R}^d \to \mathbb{C}^k$, which maps image features z to the complex space. η is also implemented as a MLP. The composed representation denoted by $\phi \in \mathbb{C}^k$ can be written as:

$$\phi = \delta \; \eta(z) \tag{5.4}$$

The optimization problem defined in Eq. 5.1 aims to maximize the similarity between the composed features and the target image features extracted from the image model. Thus, we need to learn a mapping function, $\rho : \mathbb{C}^k \mapsto \mathbb{R}^d$, from the complex space $\mathbb{C}^k$ back to the d-dimensional real space where extracted target image features exist. ρ is implemented as MLP.

In order to better capture the underlying cross-modal similarity structure in the data, we learn another mapping, denoted as ρ_{conv}. The convolutional mapping is implemented as two fully connected layers followed by a single convolutional layer. It learns 64 convolutional filters and adaptive max pooling is applied on them to get the representation from this convolutional mapping. This enables learning effective local interactions among different features. In addition to ϕ, ρ_{conv} also takes raw features z and q as input. ρ_{conv} plays a really important role for queries where the query text asks for a modification that is spatially localized. e.g., a user wants a t-shirt with a different logo on the front (see second row in Fig. 5.4).

Let $f(z, q)$ denote the overall composition function which learns how to effectively compose *extracted image and text features* for target image retrieval. The final representation, $\vartheta \in \mathbb{R}^d$, of the composed image-text features can be written as follows:

$$\vartheta = f(z, q) = a \; \rho(\phi) + b \; \rho_{conv}(\phi, z, q), \tag{5.5}$$

where a and b are learnable parameters.

In autoencoder terminology, the encoder has learnt the composed representation of image and text query, ϑ. Next, we learn to reconstruct the extracted image z and text features q from ϑ. Separate decoders are learned for each modality, i.e., image decoder and text decoder denoted by d_{img} and d_{txt} respectively. The reason for using the decoders and reconstruction losses is two-fold: first, it acts as regularizer on the learnt composed representation and secondly, it forces the composition function to retain relevant text and image information in the final representation. Empirically, we have seen that these losses reduce the variation in the performance and aid in preventing overfitting.

5.3.4 Training Objective

We adopt a deep metric learning (DML) approach to train ComposeAE. Our training objective is to learn a similarity metric, $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$, between composed image-text query features ϑ and extracted target image features $\psi(y)$. The composition function $f(z,q)$ should learn to map semantically similar points from the data manifold in $\mathbb{R}^d \times \mathbb{R}^h$ onto metrically close points in $\mathbb{R}^d$. Analogously, $f(\cdot,\cdot)$ should push the composed representation away from non-similar images in $\mathbb{R}^d$.

For sample i from the training mini-batch of size N, let ϑ_i denote the composition feature, $\psi(y_i)$ denote the target image features and $\psi(\tilde{y}_i)$ denote the randomly selected negative image from the mini-batch. We follow TIRG [127] in choosing the base loss for the datasets.

So, for MIT-States dataset, we employ triplet loss with soft margin as a base loss. It is given by:

$$\mathcal{L}_{ST} = \frac{1}{MN}\sum_{i=1}^{N}\sum_{m=1}^{M}\log\Big\{1+\exp\{\kappa(\vartheta_i,\psi(\tilde{y}_{i,m})) - \kappa(\vartheta_i,\psi(y_i))\}\Big\}, \tag{5.6}$$

where M denotes the number of triplets for each training sample i. In our experiments, we choose the same value as mentioned in the TIRG code, i.e. 3.

For Fashion200k and Fashion IQ datasets, the base loss is the softmax loss with similarity kernels, denoted as L_{SMAX}. For each training sample i, we normalize the similarity between the composed query-image features (ϑ_i) and target image features by dividing it with the sum of similarities between ϑ_i and all the target images in the batch. This is equivalent to the classification based loss in [127, 148, 149, 150].

$$\mathcal{L}_{SMAX} = \frac{1}{N}\sum_{i=1}^{N} -\log\left\{\frac{\exp\{\kappa(\vartheta_i,\psi(y_i))\}}{\sum_{j=1}^{N}\exp\{\kappa(\vartheta_i,\psi(y_j))\}}\right\}, \tag{5.7}$$

In addition to the base loss, we also incorporate two reconstruction losses in our training objective. They act as regularizers on the learning of the composed representation. The image reconstruction loss is given by:

$$\mathcal{L}_{RI} = \frac{1}{N}\sum_{i=1}^{N}\Big\|z_i - \hat{z_i}\Big\|_2^2, \tag{5.8}$$

where $\hat{z_i} = d_{img}(\vartheta_i)$.

Similarly, the text reconstruction loss is given by:

$$\mathcal{L}_{RT} = \frac{1}{N}\sum_{i=1}^{N}\Big\|q_i - \hat{q_i}\Big\|_2^2, \tag{5.9}$$

where $\hat{q_i} = d_{txt}(\vartheta_i)$.

5.3.5 Rotational Symmetry Loss

As discussed in subsection 5.3.2, based on our novel formulation of learning the composition function, we can include a *rotational symmetry loss* in our training objective. Specifically, we require that the composition of the target image features with the complex conjugate of the text features should be similar to the query image features. In concrete terms, first we obtain the complex conjugate of the text features projected in the complex space. It is given by:

$$\delta^* = \mathcal{E}\{-j\gamma(q)\}. \tag{5.10}$$

Let $\tilde{\phi}$ denote the composition of δ^* with the target image features $\psi(y)$ in the complex space. Concretely:

$$\tilde{\phi} = \delta^* \; \eta(\psi(y)) \tag{5.11}$$

Finally, we compute the composed representation, denoted by ϑ^*, in the following way:

$$\vartheta^* = f(\psi(y), q) = a \; \rho(\tilde{\phi}) + b \; \rho_{conv}(\tilde{\phi}, \psi(y), q) \tag{5.12}$$

The *rotational symmetry constraint* translates to maximizing this similarity kernel: $\kappa(\vartheta^*, z)$. We incorporate this constraint in our training objective by employing softmax loss or soft-triplet loss depending on the dataset.

Since for Fashion datasets, the base loss is L_{SMAX}, we calculate the rotational symmetry loss, L_{SYM}^{SMAX}, as follows:

$$\mathcal{L}_{SYM}^{SMAX} = \frac{1}{N} \sum_{i=1}^{N} -\log \left\{ \frac{\exp\{\kappa(\vartheta_i^*, z_i)\}}{\sum_{j=1}^{N} \exp\{\kappa(\vartheta_i^*, z_j)\}} \right\}, \tag{5.13}$$

Analogously, the resulting loss function, L_{SYM}^{ST}, for MIT-States is given by:

$$\mathcal{L}_{SYM}^{ST} = \frac{1}{MN} \sum_{i=1}^{N} \sum_{m=1}^{M} \log \Big\{ 1 + \exp\{\kappa(\vartheta_i^*, \tilde{z}_{i,m}) - \kappa(\vartheta_i^*, z_i)\} \Big\}, \tag{5.14}$$

The total loss is computed by the weighted sum of above mentioned losses. It is given by:

$$\mathcal{L}_T = \mathcal{L}_{BASE} + \lambda_{SYM} \; \mathcal{L}_{SYM}^{BASE} + \lambda_{RI} \; \mathcal{L}_{RI} + \lambda_{RT} \; \mathcal{L}_{RT}, \tag{5.15}$$

where $BASE \in \{SMAX, ST\}$ depending on the dataset.

5.4 Experiments

5.4.1 Experimental Setup

We evaluate our approach on three real-world datasets, namely: MIT-States[151], Fashion200k [140] and Fashion IQ [152]. For evaluation, we follow the same protocols as

	MIT-States	Fashion200k	Fashion IQ
Total images	53753	201838	62145
# train queries	43207	172049	46609
# test queries	82732	33480	15536
Average length of complete text query	2	4.81	13.5
Average # of target images per query	26.7	3	1

Table 5.1: Dataset statistics

other recent works [127, 140, 138]. We use recall at rank k, denoted as $R@k$, as our evaluation metric. We repeat each experiment 5 times in order to estimate the mean and the standard deviation in the performance of the models.

To ensure fair comparison, we keep the same hyperparameters as TIRG [127] and use the same optimizer (SGD with momentum). Similar to TIRG, we use ResNet-17 for image feature extraction to get 512-dimensional feature vector. In contrast to TIRG, we use pretrained BERT [130] for encoding text query. Concretely, we employ BERT-as-service [153] and use Uncased BERT-Base which outputs a 768-dimensional feature vector for a text query.

5.4.2 Baselines

We compare the results of ComposeAE with several methods, namely: Show and Tell, Parameter Hashing, Attribute as Operator, Relationship, FiLM and TIRG. We explained them briefly in Sec. 5.2.

In order to identify the limitations of TIRG and to ensure fair comparison with our method, we introduce three variants of TIRG. First, we employ the BERT model as a text model instead of LSTM, which will be referred to as *TIRG with BERT*. Secondly, we keep the LSTM but text query contains full target captions. We refer to it as *TIRG with Complete Text Query*. Thirdly, we combine these two variants and get *TIRG with BERT and Complete Text Query*. The reason for complete text query baselines is that the original TIRG approach *generates* text query by finding one word difference in the source and target image captions. It disregards all other words in the target captions.

While such formulation of queries may be effective on some datasets, but the restriction on the specific form (or length) of text query largely constrain the information that a user can convey to benefit the retrieval process. Thus, such an approach of generating text query has limited applications in real life scenarios, where a user usually describes the modification text with multiple words. This argument is also supported by several recent studies [152, 142, 143]. In our experiments, Fashion IQ dataset contains queries *asked by humans* in natural language, with an average length of 13.5 words. (see Table 5.1). Due to this reason, we can not get results of original TIRG on this dataset.

Method	R@1	R@5	R@10
Show and Tell	$11.9^{\pm 0.1}$	$31.0^{\pm 0.5}$	$42.0^{\pm 0.8}$
Att. as Operator	$8.8^{\pm 0.1}$	$27.3^{\pm 0.3}$	$39.1^{\pm 0.3}$
Relationship	$12.3^{\pm 0.5}$	$31.9^{\pm 0.7}$	$42.9^{\pm 0.9}$
FiLM	$10.1^{\pm 0.3}$	$27.7^{\pm 0.7}$	$38.3^{\pm 0.7}$
TIRG	$12.2^{\pm 0.4}$	$31.9^{\pm 0.3}$	$43.1^{\pm 0.3}$
TIRG with BERT	$12.3^{\pm 0.6}$	$31.8^{\pm 0.3}$	$42.6^{\pm 0.8}$
TIRG with Complete Text Query	$7.9^{\pm 1.9}$	$28.7^{\pm 2.5}$	$34.1^{\pm 2.9}$
TIRG with BERT and Complete Text Query	$\underline{13.3}^{\pm 0.6}$	$\underline{34.5}^{\pm 1.0}$	$\underline{46.8}^{\pm 1.1}$
ComposeAE	$\mathbf{13.9}^{\pm 0.5}$	$\mathbf{35.3}^{\pm 0.8}$	$\mathbf{47.9}^{\pm 0.7}$

Table 5.2: Model performance comparison on MIT-States. The best number is in bold and the second best is underlined.

5.4.3 Datasets

Table 5.1 summarizes the statistics of the datasets. The train-test split of the datasets is the same for all the methods.

MIT-States [151] dataset consists of ∼60k diverse real-world images where each image is described by an adjective (state) and a noun (categories), e.g. “ripe tomato”. There are 245 nouns in the dataset and 49 of them are reserved for testing. This split ensures that the algorithm is able to learn the composition on the unseen nouns (categories). The input image (say “unripe tomato”) is sampled and the text query asks to change the state to ripe. The algorithm is considered successful if it retrieves the correct target image (“ripe tomato”) from the pool of *all test images.*

Fashion200k [140] consists of ∼200k images of 5 different fashion categories, namely: pants, skirts, dresses, tops and jackets. Each image has a human annotated caption, e.g. “blue knee length skirt”.

Fashion IQ[152] is a challenging dataset consisting of 77684 images belonging to three categories: dresses, top-tees and shirts. Fashion IQ has two human written annotations for each target image. We report the performance on the validation set as the test set labels are not available.

5.4.4 Discussion of Results

Tables 5.2, 5.3 and 5.4 summarize the results of the performance comparison. In the following, we discuss these results to gain some important insights into the problem.

First, we note that our proposed method ComposeAE outperforms other methods by a significant margin. On Fashion200k, the performance improvement of ComposeAE over the original TIRG and its enhanced variants is most significant. Specifically, in terms of R@10 metric, the performance improvement over the second best method is 6.96% and 30.12% over the original TIRG method . Similarly on R@10, for MIT-

States, ComposeAE outperforms the second best method by 2.35% and by 11.13% over the original TIRG method. For the Fashion IQ dataset , ComposeAE has 2.61% and 3.82% better performance than the second best method in terms of R@10 and R@100 respectively.

Method	R@1	R@10	R@50
Han *et al.* [140]	6.3	19.9	38.3
Show and Tell	$12.3^{\pm 1.1}$	$40.2^{\pm 1.7}$	$61.8^{\pm 0.9}$
Param Hashing	$12.2^{\pm 1.1}$	$40.0^{\pm 1.1}$	$61.7^{\pm 0.8}$
Relationship	$13.0^{\pm 0.6}$	$40.5^{\pm 0.7}$	$62.4^{\pm 0.6}$
FiLM	$12.9^{\pm 0.7}$	$39.5^{\pm 2.1}$	$61.9^{\pm 1.9}$
TIRG	$14.1^{\pm 0.6}$	$42.5^{\pm 0.7}$	$63.8^{\pm 0.8}$
TIRG with BERT	$14.2^{\pm 1.0}$	$41.9^{\pm 1.0}$	$63.3^{\pm 0.9}$
TIRG with Complete Text Query	$18.1^{\pm 1.9}$	$\underline{52.4}^{\pm 2.7}$	$\underline{73.1}^{\pm 2.1}$
TIRG with BERT and Complete Text Query	$\underline{19.9}^{\pm 1.0}$	$51.7^{\pm 1.5}$	$71.8^{\pm 1.3}$
ComposeAE	$\mathbf{22.8}^{\pm 0.8}$	$\mathbf{55.3}^{\pm 0.6}$	$\mathbf{73.4}^{\pm 0.4}$

Table 5.3: Model performance comparison on Fashion200k. The best number is in bold and the second best is underlined.

Method	R@10	R@50	R@100
TIRG with Complete Text Query	$3.34^{\pm 0.6}$	$9.18^{\pm 0.9}$	$9.45^{\pm 0.8}$
TIRG with BERT and Complete Text Query	$\underline{11.5}^{\pm 0.8}$	$\underline{28.8}^{\pm 1.5}$	$\underline{28.8}^{\pm 1.6}$
ComposeAE	$\mathbf{11.8}^{\pm 0.9}$	$\mathbf{29.4}^{\pm 1.1}$	$\mathbf{29.9}^{\pm 1.3}$

Table 5.4: Model performance comparison on Fashion IQ. The best number is in bold and the second best is underlined.

Second, we observe that the performance of the methods on MIT-States and Fashion200k datasets is in a similar range as compared to the range on the Fashion IQ. For instance, in terms of R@10, the performance of *TIRG with BERT and Complete Text Query* is 46.8 and 51.8 on MIT-States and Fashion200k datasets while it is 11.5 for Fashion IQ. The reasons which make Fashion IQ the most challenging among the three datasets are: (i) the text query is quite complex and detailed and (ii) there is only one target image per query (See Table 7.1). That is even though the algorithm retrieves semantically similar images but they will not be considered correct by the recall metric. For instance, for the first query in Fig.5.4, we can see that the second, third and fourth image are semantically similar and modify the image as described by the query text. But if the third image which is the labelled target image did not appear in top-5, then

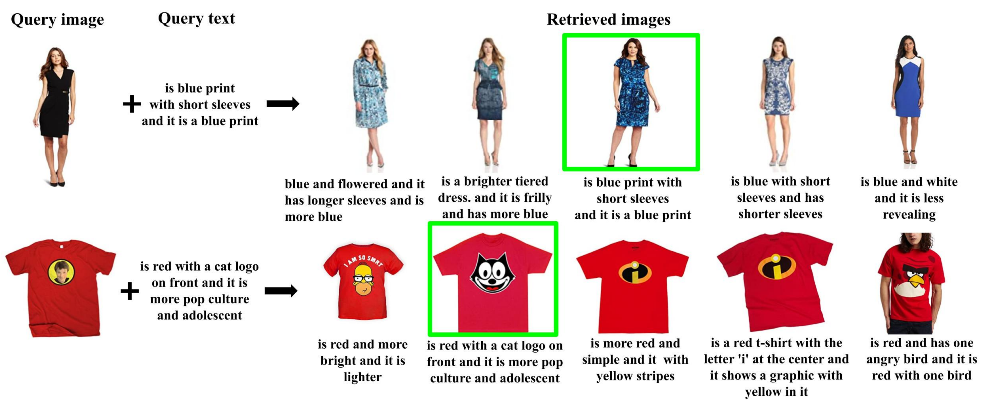
Query image
Query text
Retrieved images
is blue print with short sleeves and it is a blue print
blue and flowered and it has longer sleeves and is more blue
is a brighter tiered dress. and it is frilly and has more blue
is blue print with short sleeves and it is a blue print
is blue with short sleeves and has shorter sleeves
is blue and white and it is less revealing
is red with a cat logo on front and it is more pop culture and adolescent
is red and more bright and it is lighter
is red with a cat logo on front and it is more pop culture and adolescent
is more red and simple and it with yellow stripes
is a red t-shirt with the letter 'i' at the center and it shows a graphic with yellow in it
is red and has one angry bird and it is red with one bird

Figure 5.4: Qualitative Results: Retrieval examples from FashionIQ Dataset

R@5 would have been zero for this query. This issue has also been discussed in depth by Nawaz et al.[154].

Third, for MIT-States and Fashion200k datasets, we observe that the TIRG variant which replaces LSTM with BERT as a text model results in slight degradation of the performance. On the other hand, the performance of the TIRG variant which uses complete text (caption) query is quite better than the original TIRG. However, for the Fashion IQ dataset which represents a real-world application scenario, the performance of TIRG with complete text query is significantly worse. Concretely, TIRG with complete text query performs 253% worse than ComposeAE on R@10. The reason for this huge variation is that the average length of complete text query for MIT-States and Fashion200k datasets is 2 and 3.5 respectively. Whereas average length of complete text query for Fashion IQ is 12.4. It is because TIRG uses the LSTM model and the composition is done in a way which underestimates the importance of the text query. This shows that TIRG approach does not perform well when the query text description is more realistic and complex.

Fourth, one of the baselines (*TIRG with BERT and Complete Text Query*) that we introduced shows significant improvement over the original TIRG. Specifically, in terms of R@10, the performance gain over original TIRG is 8.58% and 21.65% on MIT-States and Fashion200k respectively. This method is also the second best performing method on all datasets. We think that with more detailed text query, BERT is able to give better representation of the query and this in turn helps in the improvement of the performance.

5.4.5 Qualitative Results

Fig.5.4 presents some qualitative retrieval results for Fashion IQ. For the first query, we see that all images are in “blue print” as requested by text query. The second request in the text query was that the dress should be “short sleeves”, four out of top-5 images fulfill this requirement. For the second query, we can observe that all retrieved images share the same semantics and are visually similar to the target images.

Method	Fashion200k	MIT-States	Fashion IQ
ComposeAE	55.3	47.9	11.8
- without L_{SYM}	51.6	47.6	10.5
- Concat in real space	48.4	46.2	09.8
- without ρ_{conv}	52.8	47.1	10.7
- without ρ	52.2	45.2	11.1

Table 5.5: Retrieval performance (R@10) of ablation studies.

5.4.6 Ablation Studies

We have conducted various ablation studies, in order to gain insight into which parts of our approach helps in the high performance of ComposeAE. Table 5.5 presents the quantitative results of these studies.

Impact of L_{SYM}: on the performance can be seen on Row 2. For Fashion200k and Fashion IQ datasets, the decrease in performance is quite significant: 7.17% and 12.38% respectively. While for MIT-States, the impact of incorporating L_{SYM} is not that significant. It may be because the text query is quite simple in the MIT-states case, i.e. 2 words. This needs further investigation.

Efficacy of Mapping to Complex Space: ComposeAE has a complex projection module, see Fig. 5.3. We removed this module to quantify its effect on the performance. Row 3 shows that there is a drop in performance for all three datasets. This strengthens our hypothesis that it is better to map the extracted image and text features into a common complex space than simple concatenation in real space.

Convolutional versus Fully-Connected Mapping: ComposeAE has two modules for mapping the features from complex space to target image space, i.e., $\rho(\cdot)$ and the second with an additional convolutional layer $\rho_{conv}(\cdot)$. Rows 4 and 5 show that the performance is quite similar for fashion datasets. While for MIT-States, ComposeAE without $\rho_{conv}(\cdot)$ performs much better. Overall, it can be observed that for all three datasets both modules contribute in improving the performance of ComposeAE.

5.4.7 Important Notes on Fashion IQ Dataset

In Fashion IQ dataset, $\sim$ 49% annotations describe the target image directly. While $\sim$ 32% annotations compares target and source images, e.g. “is red with a cat logo on front” and the second annotation is, “is more pop culture and adolescent”. The dataset consists of three non-overlapping subsets, namely “dress”, “top-tee” and “shirt”. We join the two annotations with the text “ and it” to get a description similar to a normal sentence a user might ask on an E-Com platform. Now the complete text query is: “is red with a cat logo on front and it is more pop culture and adolescent”. Furthermore, we combine the train sets of all three categories to form a bigger training set and train a *single* model on it. Analogously, we also combine the validation sets to form a single validation set.

A challenge was conducted in ICCV 2019 on Fashion IQ dataset [1]. The website also has some technical reports submitted by the best performing teams. The numbers reported in these reports are quite high, even for TIRG approach. We investigated the reasons and reached the conclusion that these technical reports have have quite different settings. It is not possible for us to compare our results with them in a fair manner. The reasons and differences are delineated briefly as:

- They treat Fashion IQ as three independent datasets and train one model for each category ("dress", "top-tee" and "shirt"). This results in better performance for each category.
- They do pre-training on external datasets like Fashiongen, Fashion200k etc. It is well-known that such transfer learning (via pre-training) inevitably increases the performance of any model.
- They employ product attributes as side information in their models. In our experiments, we do not consider in such side information and rely solely on the image and text query.
- They employ higher capacity models such as ResNet101, ResNet-152 etc. In original TIRG and in all our experiments, we use ResNet17 as image model.
- Since these reports developed models specifically for the competition, they have incorporated several hacks, like ensembeling, data augmentation techniques etc.
- Unfortunately, none of the technical reports have published their code. Thus, we are not able to assess the performance of their model in our experiment setting.

In short, it is neither possible for us to reproduce their results nor are we able to fairly compare the performance of their models in a common experiment setting.

[1]https://sites.google.com/view/lingir/fashion-iq

6 A Contrastive Learning Approach for Compositional Zero-Shot Learning

This chapter is based on the following under-review publication:

Anwaar, Muhammad Umer[*]*; Khan, Rayyan Ahmad [*]; Pan, Zhihui and Kleinsteuber, Martin. A Contrastive Learning Approach for Compositional Zero-Shot Learning. In ICMI '21: Proceedings of the 2021 International Conference on Multimodal Interaction. 2021*
[*] indicates that the authors contributed equally to this work.

6.1 Introduction

Objects, in the real world, exist in certain state(s). Human cognition is well set up to process vague concepts and easily categorize them and assign attributes (states) to objects. For instance, if a user wishes to buy a dress, their mind turns the attention towards what color, size, style, price etc., they would prefer. Traditional information retrieval (e.g. e-commerce websites) offer their customers a unimodal query, i.e., either a text or an image. After the query, the user has to select many filters to "help the algorithm" narrow down the options for the user. Future information retrieval (IR) systems should be smart enough to "help the customer" in expressing the concept in their mind by allowing a multi-modal query (see Fig. **??**).

In this work, we focus our attention to a major aspect of such smart systems: learning good state-object representations. Specifically, (1) we aim to learn such models which understand different states of an object and can recognise even unseen combinations of them. (2) The model should be able to retrieve images based on multi-modal (image-text) query, where the text describes the changes sought by the user in the query image.

In the literature, these two tasks have been treated separately. The first task is referred to as compositional zero shot learning (CZSL). In contrast to image classification, the goal of CZSL classification is to simultaneously identify the class of the object and the state in which the object appears. Sometimes the visual differences of the same object in two states can be huge and that is where traditional classification methods fail. Several CZSL methods have been proposed to address this challenge. Li et al. [155] subject the learning of object embeddings to symmetry constraints under different state transformations. Misra et al. [156] maximize similarity of learned joint state-object embeddings with image features. Nagarajan et al. [157] treat states as operators (transformation matrices) and apply them to objects to yield the joint pair embeddings and then maximize the similarity with image. The second task is image retrieval based on a multi-modal query. Vo et al. [127] proposed the Text Image Residual Gating (TIRG) method for composing the query image and text for image retrieval. Anwaar et al. [158] proposed that the target image representation is an element-wise rotation of the representation of the source image in a complex space. The information about the degree of rotation is specified by the query text features.

The above mentioned methods do not focus on solving the two tasks in a systematic way. Interestingly, the state-of-the-art (SOTA) methods for CZSL task employ such data splits where all the objects and states are seen by the model during training. They refer to it as zero-shot because the test set contains images with pairs (state-object combinations) which were not seen in training. This will prove to be quite limiting for the real-world image retrieval application. Since the model is expected to have seen not only all the objects but also all the attributes the user can come up with. Naturally, a user can use different words which carry the same semantic meaning. Thus, a good model must have the ability to generalize to both unseen objects and unseen attributes. On the other hand, the second class of methods do not learn any object or state classification. They are focused on learning the fusion of query image and text for directly improving the image retrieval. This approach results in poor performance on the first task (see Sec.6.4.2).

In this work, we propose a unified approach, **ContraNet** , which bridges the existing gap in these two tasks. **ContraNet** aims to predict a composition of multiple semantic concepts in images. We adopt a contrastive learning approach to learn embeddings which are visually grounded and semantically meaningful. In recent years, contrastive learning has shown impressive results on a variety of tasks [159, 160, 161, 162]. Our rationale behind adopting contrastive learning is that current SOTA methods (discussed above), overwhelmingly rely on labels for learning. They overlook the fact that the underlying data lives on a much complicated manifold than what sparse labels could capture. Therefore, purely supervised methods converge to rigid solutions. In other words, they lead to good task-specific solutions, rather than learning the multiple semantic concepts in the data. **ContraNet** utilizes pretrained image and text models and then learn their mapping onto a multimodal embedding space via contrastive loss. That is, by maximizing the similarity between actual image-text pairs against random pairs through a bidirectional contrastive loss between the text and image modalities. These embeddings from this multimodal embedding space are then utilized for the downstream tasks. We use cross-entropy losses for the task 1 and soft-triplet loss for the task 2.

Despite the simplicity of our model, **ContraNet** outperforms the SOTA methods on benchmark datasets, namely: MIT-States, UT-Zappos and Fashion200k. Our experimental evaluation shows that projecting the text and image features onto a common embedding space and learning the representations via contrastive loss significantly enhances the performance of **ContraNet** .

Our main contributions are summarized below:

- We propose **ContraNet** , a unified contrastive learning approach which not only can recognize unseen combinations of state-object pairs but is also able to retrieve images of never seen objects based on multi-modal query.
- **ContraNet** outperforms the SOTA methods by a substantial margin on state-object composition zero-shot learning tasks. i.e., 8.7% on UT-Zappos and 8.1% on MIT-States on the best HM metric.
- For the image retrieval task, **ContraNet** utilizes the common embedding space learnt via contrastive loss and surpass the SOTA performance by 4% on MIT-States and 5.3% on Fashion200k dataset on Recall@1 metric.

6.2 Related Work

In computer vision, a bulk of research centered around object recognition and classification, treats attribute as a mid-level "feature" learned from the visual patterns. This has proved amazingly successful in various tasks, e.g., zero-shot recognition, image description [163, 164, 165] and visual question answering [166].

Recently, there has been an increasing interest in compositional learning of attributes (states) and objects. In compositional learning, learning correct attributes is given the same importance as object prediction. That is, the model needs to predict the state-object pair. Several approaches have been proposed to tackle this problem. LabelEmbed (LE) proposed by [156] uses GloVe vectors[167] for state and object. They employ a 3-layered MLP to transform the word embeddings into a transformation matrix. The prediction of the classifier is obtained by the product of transformation matrix with image features. AnalogousAttr [168] trains several linear classifiers for seen compositions and then leverages tensor completion techniques to do predictions for the unseen pairs. AoP [157] uses GloVe vectors for objects. But they consider states (attributes) as linear transformation matrices, which "operate" on the objects to yield pair embeddings. The pair embedding with the minimum distance to image embedding in the joint embedding space is the prediction of the model. Red Wine is another method proposed by [156]. It replaces the GloVe vectors in LE with the SVM weights. TAFE [169] employs word2vec vectors [170] of state-object composition pair to generate binary classifier for each composition. Task-driven Modular Networks (TMN) [171] configures a set of modules (fully connected layers operating in semantic concept space) through a gating function in a task-driven way. It generalizes to unseen compositions by re-weighting these primitive modules. SymNet [155] learns object embeddings showing symmetry under different state transformations. They emphasize that leaning in such a fashion yields better embeddings for the compositional learning task.

The task of image retrieval based on multimodal query has also been explored in literature. Two state of the art methods are: TIRG [127] and ComposeAE [158]. In TIRG, the authors employ gated feature connection in order to keep the composed representation of query image and text in the same space as that of the target image. They also incorporate a residual connection which learns the similarity between concatenation of image-text features and the target image features. The authors of ComposeAE [158] argue that the source image and the target image lie in a common complex space. They are rotations of each other and the degree of rotation is encoded via query text features. Some other approaches which are also closely related to this task are interactive image retrieval task [142, 143] and attribute-based product retrieval task [144]. These approaches have their limitations such as that the query texts are limited to a fixed set of relative attributes [144], require multiple rounds of natural language queries as input [142, 143] or that query texts can be only one word i.e. an attribute [140]. Unlike our task, the focus of these methods is on modeling the interaction between user and the agent.

6.3 Approach

6.3.1 Problem Setting and Overview

Let $\mathcal{X}$ denote the set of images, $\mathcal{S}$ denote the set of states, $\mathcal{O}$ the set of objects and $\mathcal{T} = \mathcal{S} \times \mathcal{O}$ denote the set of composition labels. Each image x is associated with a compositional label $t = (s, o)$.
We tackle the following two tasks:

1. Prediction of composition label (state, object) for a given image. During testing, most of the composition labels are novel i.e. not seen during training. Hence, this task is also called Compositional Zero Shot Learning Task. The model, $f : \mathcal{X} \to \mathcal{T}^{test}$, is trained to maximize the number of correct predictions of composition labels.

2. Image Retrieval based on a multi-modal (image-text) query. Specifically, the query text prompts some *state* modification in to the query image x. The task is to retrieve images with same object label o as in the query image but with the desired state label s. The model, $g : (\mathcal{X}, \mathcal{T}) \to \mathcal{X}^{target}$, is trained to maximize the similarity between composed representation of (image-text) query and the target image representation.

6.3.2 Task # 1: Learning the Compositional Zero Short Prediction

ContraNet is an autoencoder based approach to learn composition of multiple semantic concepts in images. In this section, we discuss the architecture of **ContraNet** and the loss functions involved in CZSL task. Fig. 6.1 presents an overview of **ContraNet** .

In the figure, $\psi(\cdot)$ denotes the pre-trained image model (e.g. ResNet-18), which takes an image as input and returns image features in a d-dimensional space. Analogously, $\beta(\cdot)$ denotes the pre-trained text model (e.g. BERT), which takes an text as input and returns text features in a h-dimensional space. It is to be noted that $\beta(\cdot)$ takes as input both the state and object text and returns a single h-dimensional feature vector. Typically in image-text joint embeddings [127, 131], these features are combined using

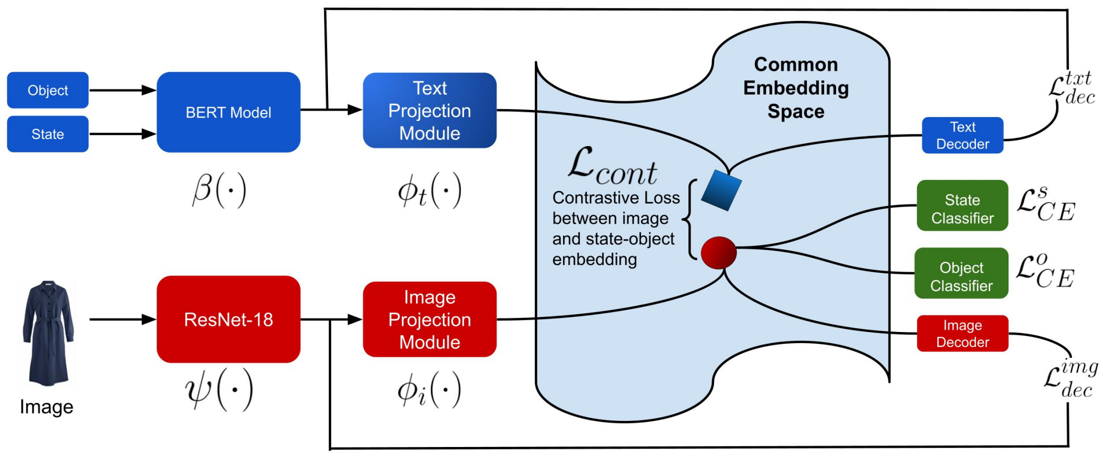

Figure 6.1: ContraNet Architecture: Learning the compositional labels via contrastive learning

fully connected layers or gating mechanisms. In contrast to this, we project these features onto a common multi-modal embedding space via separate projection modules. These modules are denoted by $\phi_t : \mathbb{R}^h \mapsto \mathbb{R}^k$ and $\phi_i : \mathbb{R}^d \mapsto \mathbb{R}^k$ for text and image respectively.

In this common embedding space, we aim to learn such representations which better capture the underlying cross-modal dependencies. **ContraNet** utilizes the text information to learn multiple semantics present in the images. We adopt a deep metric learning approach (DML) to train **ContraNet** in a contrastive fashion. During training, we sample a batch of B input image-text pairs (x, t). We calculate the similarities between the representations of images and texts in the common embedding space and then calculate the contrastive loss, $\mathcal{L}_{cont}$, as follows:

$$\Delta_t = \Big\|_{j=1}^{B} \phi_t(\beta(t_j)), \tag{6.1}$$

$$\Delta_i = \Big\|_{j=1}^{B} \phi_i(\psi(x_j)), \tag{6.2}$$

$$\mathcal{E} = \Delta_t * \Delta_i^T, \tag{6.3}$$

$$\begin{aligned}\mathcal{L}_{cont} = \frac{1}{2B}\sum_{j=1}^{B} & -\log\left\{\frac{\exp\{\mathcal{E}_{jj}\}}{\sum_{p=1}^{B}\exp\{\mathcal{E}_{jp}\}}\right\} \\ & -\log\left\{\frac{\exp\{\mathcal{E}_{jj}\}}{\sum_{p=1}^{B}\exp\{\mathcal{E}_{pj}\}}\right\},\end{aligned} \tag{6.4}$$

where $\|$ denotes the concatenation operation of representation for each sample of the batch, T denotes transpose of a matrix, $*$ denotes matrix multiplication, *exp* denotes exponential function and $\mathcal{E}$ denotes the matrix of similarities of all image-text pairs in a batch. The first term in the loss function corresponds to text-to-image contrastive loss and the second term corresponds to image-to-text contrastive loss.

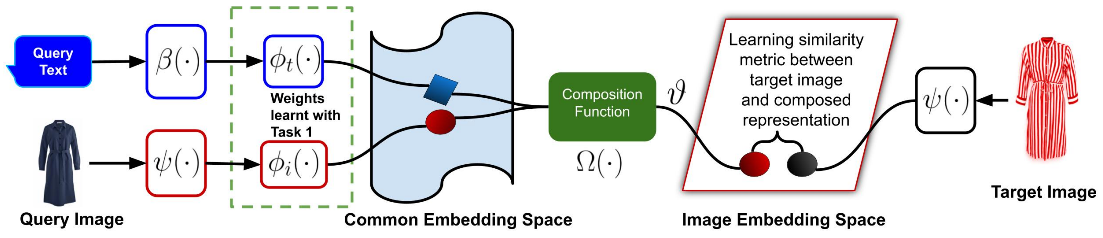

Figure 6.2: ContraNet Architecture: Image retrieval based on multi-modal query. The weights of the projection modules learned during task 1 are frozen.

As for the decoder part of **ContraNet** , we learn two separate decoders from the representations embedded in the multi-modal embedding space. i.e., image decoder and text decoder denoted by d_{img} and d_{txt} respectively. The reason for using the decoders and reconstruction losses is two-fold: first, it acts as regularizer on learning of the embeddings and secondly, it forces the model to retain relevant text and image information in the common embedding space. Empirically, we have seen that these losses reduce the variation in the performance and aid in preventing overfitting.

Thus, we also add two reconstruction losses, $\mathcal{L}_{dec}^{img}$ and $\mathcal{L}_{dec}^{txt}$, in our training objective, each corresponding to d_{img} and d_{txt} respectively. They are given by:

$$\mathcal{L}_{dec}^{img} = \frac{1}{B}\sum_{i=j}^{B} \left\| \psi(x_j) - \hat{z}_i \right\|_2^2, \tag{6.5}$$

$$\mathcal{L}_{dec}^{txt} = \frac{1}{B}\sum_{i=j}^{B} \left\| \beta(t_j) - \hat{q}_i \right\|_2^2, \tag{6.6}$$

where $\hat{z}_i = d_{img}(\cdot)$ and $\hat{q}_i = d_{txt}(\cdot)$.

For this task, during inference, we have to predict the composition label (state, object) for a given image. Thus, **ContraNet** trains two separate classifiers for state and object classification. These classifiers take as input the image representation projected onto the common embedding space. They are implemented as two fully-connected (FC) layers, followed by a softmax layer. They are trained with cross-entropy losses, denoted as $\mathcal{L}_{CE}^{S}$ and $\mathcal{L}_{CE}^{O}$, for state and object classification.

The total loss is computed by the weighted sum of above mentioned losses. It is given by:

$$\mathcal{L} = \mathcal{L}_{cont} + \lambda_i\, \mathcal{L}_{dec}^{img} + \lambda_t\, \mathcal{L}_{dec}^{txt} + \lambda_o\, \mathcal{L}_{CE}^{O} + \lambda_s\, \mathcal{L}_{CE}^{S}, \tag{6.7}$$

6.3.3 Task # 2: Image Retrieval Based on a Multi-Modal Query

As discussed in Sec. 6.3.1, the goal in this task is to retrieve images which have same object label as the query image but possess the state as desired by query text. For instance, we have input image of an “unripe tomato” and the text prompts to retrieve images with the “ripe” state (see Fig. 6.4). Fig. 6.2 presents the modified **ContraNet** architecture for this task. Since the goal is to retrieve images, we hypothesise that it is better to compose the modalities in the multi-modal query and map them to the image embedding space. This ensures that the image and compositional features are “aligned”. In this way, during inference, we can simply compare the composed features with the test images in the image embedding space.

For this task, we learn a composition function, $\Omega : \mathbb{R}^{2k} \mapsto \mathbb{R}^d$, for composing the embeddings of query image and text coming from the common embedding space. This function maps the embeddings directly in to the image embedding space. Lets denote the output of $\Omega(\cdot)$ by ϑ. In our experiments, we work with three different variants of $\Omega(\cdot)$, namely: MLP, residual gating and image rotation based on text (see Sec. 6.4.4 for details.)

We employ triplet loss with soft margin, $\mathcal{L}_{trip}$, for learning this composition function. The loss aims to maximize the similarity between the composed features ϑ and the target image features $\psi(y)$ extracted from the image model. It also pushes the composed representation ϑ away from non-similar images in $\mathbb{R}^d$. Let $\kappa(\cdot,\cdot)$ denote the similarity kernel, which we implement as a dot product between its inputs. The loss is given by:

$$\mathcal{L}_{trip} = \frac{1}{MB}\sum_{j=1}^{B}\sum_{m=1}^{M} \log\Big\{1+\exp\{\kappa(\vartheta_j,\psi(\tilde{y}_{j,m})) \; - \; \kappa(\vartheta_j,\psi(y_j))\}\Big\}, \tag{6.8}$$

where M denotes the number of triplets for each sample j and $\psi(\tilde{y}_j)$ denote the randomly selected negative image from the batch. This is equivalent to the the soft triplet based loss used in [172, 173].

6.4 Experiments

6.4.1 Experimental Setup

Datasets: In our experiments, we use three benchmark datasets, namely: MIT-States[174], UT Zappos [175] and Fashion200k[140]. MIT-States consists of 53753 diverse real-world images where each image is described by an object-attribute composition label, i.e. an attribute (state) and a noun (object), e.g. “ripe tomato”. There are 245 objects, 115 attributes and 1962 possible pairs. UT-Zappos is a dataset of only shoes with fine-grained annotations. A composition label consists of shoe type-material pair. There are 12 shoe types (objects), 16 different materials (states) and 116 possible pairs.

Two different settings of these benchmark datasets are proposed in the literature for evaluating the models. (1) Generalized CZSL (GCZSL) split [171]. Following Chao et al. [176], they argue that performance of the model should also be evaluated on seen pairs. In this setting, MIT-States utilizes 1262 pairs (the seen pairs) for training, whereas the test set has 400 seen and 400 unseen pairs. This split also contains a validation set consisting of 300 seen and 300 unseen pairs. The UT-Zappos dataset makes use of 83 pairs (the seen pairs) for training, whereas the test set has 18 seen and 18 unseen pairs. The validation set has of 15 seen and 15 unseen pairs. (2) CZSL split [156, 157], which ensures that there is no overlap between pairs in the train and test set. For MIT-states, the train set still consists of 1262 pairs/34562 images and the test set now has 700 pairs/19191 images. For UT-Zappos, the train set still has 83 pairs/24898 images and the test set now contains 33 pairs/4228 images.

The third dataset Fashion200k [140] is only used for the second task. It consists of ∼200k images of different fashion categories. Each image has a human annotated caption, e.g. "beige bolero jacket dress". In order to ensure fair comparison, for the second task, we also follow the same train-test split as proposed by our competitors (TIRG [127] and ComposeAE [158]) for both Fashion200k and MIT-States. Now the split for MIT-States is that 196 objects out of total 245 are reserved for training and the rest 49 objects are used during test. This split ensures that there is no overlap between training and testing queries in terms of objects.

Implementation Details: Following [171, 155, 169], we compare the results of **ContraNet** with several baselines as well as previous state-of-the-art (SOTA) methods. Unless otherwise specified, the default image feature extractor for all methods is ResNet-18. We also extract 512-dimensional image features using ResNet18 pretrained on ImageNet [177]. **ContraNet** employs pretrained BERT [130] for encoding texts. Concretely, we employ BERT-as-service [153] and use Uncased BERT-Base++ which outputs a 1024-dimensional feature vector. To ensure fair comparison, we employ the same strategy for hyperparameter tuning as SymNet [155]. We use cross-validation to determine the hyper-parameters, e.g., learning rate, weights, epochs. We employ Adam [118] optimizer. For both datasets, $M = 3$ in $\mathcal{L}_{trip}$ and the weights of the losses are: $\lambda_s = \lambda_o = 1$ and $\lambda_i = \lambda_t = 0.1$. We repeat each experiment 10 times and report the average performance of the models.

Metrics: For the task of CZSL, we follow [157, 155] and report Top-1, 2, 3 accuracies on the unseen test set as evaluation metrics. For the task of GCZSL, we follow the evaluation protocol from TMN [171] and use the same metrics as proposed by them. Namely: best accuracy on only images of seen/unseen compositions (*best seen/best unseen*), best harmonic mean (*best HM*) and Area Under the Curve (AUC) for seen and unseen accuracies by varying the bias values. For the task of image retrieval, following the evaluation protocol from TIRG [127], we use recall at rank k ($R@k$), as our evaluation metric. $R@k$ estimates the proportion of queries where the target (ground truth) image is within the top k retrieved images.

6.4.2 Discussion of Results for the GCZSL and CZSL Tasks

Tables 6.1 and 6.2 summarize the results of the performance comparison on the GCZSL task. In the following, we discuss these results to gain some important insights into the problem.

First, we note that our proposed method **ContraNet** consistently outperforms the SOTA methods by a significant margin on both benchmark datasets. Specifically, on UT-Zappos in terms of AUC metric, the performance improvement over the second best method (TMN) is 18.4% and 32.9% over the third best method (RedWine). Similarly, for best HM metric on UT-Zappos, **ContraNet** outperforms the second best method

Method	MIT-States					
	Attribute	Object	Seen	Unseen	HM	AUC
AoP	21.1	23.6	14.3	17.4	9.9	1.6
Red Wine	22.7	25.1	20.7	17.9	11.6	2.4
LabelEmbed	23.5	26.3	15.0	20.1	10.7	2.0
ComposeAE	23.8	26.4	21.4	22.6	14.9	2.7
TMN	23.3	26.5	20.2	20.1	13.0	2.9
SymNet	24.3	**27.3**	24.2	25.2	16.1	3.0
ContraNet	**28.9**	26.7	**28.1**	**27.4**	**17.4**	**4.7**
- without $\mathcal{L}_{cont}$	22.9	<u>27.1</u>	14.8	18.6	9.7	1.9
- without $\mathcal{L}_{dec}$	<u>28.2</u>	26.5	<u>27.5</u>	<u>26.8</u>	<u>17.2</u>	<u>3.9</u>

Table 6.1: Performance comparison on the Generalized CZSL split of MIT-States. The best performance is in bold and the second best is underlined.

Method	UT-Zappos					
	Attribute	Object	Seen	Unseen	HM	AUC
AoP	38.9	**69.9**	<u>59.8</u>	54.2	40.8	25.9
Red Wine	40.6	69.1	53.6	52.1	41.3	26.1
LabelEmbed	41.2	<u>69.3</u>	53.0	<u>61.9</u>	41.0	25.7
ComposeAE	41.9	68.8	57.2	58.9	44.2	29.2
TMN	40.8	69.2	58.7	60.0	45.0	29.3
SymNet	41.3	68.6	49.8	57.4	40.4	23.4
ContraNet	**52.7**	68.1	**60.7**	**62.5**	**48.9**	**34.7**
- without $\mathcal{L}_{cont}$	42.4	69.2	54.9	52.6	42.9	27.4
- without $\mathcal{L}_{dec}$	<u>51.4</u>	66.7	58.4	60.8	<u>47.2</u>	<u>33.1</u>

Table 6.2: Performance comparison on the Generalized CZSL split of UT-Zappos. The best performance is in bold and the second best is underlined.

(TMN) by 8.67% and by 15.53% over the third best method (RedWine). On MIT-States dataset, **ContraNet** still outperforms the competitive methods but the margins are less than those on UT-Zappos dataset. For instance, in terms of AUC and best HM metric, the performance improvement over the second best method (SymNET) is 3.6% and 8.1% respectively.

Second, we observe a very interesting pattern for the object classification accuracy metric. On UT-Zappos dataset, the top-2 performing methods (AoP and LabelEmbed+) are fairly simple models in comparison to more recent models like SymNET and TMN. For the MIT-States, this pattern is reversed and AoP and LabelEmbed+ are performing worse than advance models. Here, SymNET and **ContraNet** achieves best performance with insignificant difference of 0.7%. We hypothesise that simpler model are better able to capture the object class for UT-Zappos. We test this conjecture by removing $\phi(\cdot)$'s and $\mathcal{L}_{cont}$ from **ContraNet** . The results support this conjecture as the performance is improved by 1.1 and 0.4 for **ContraNet** without $\mathcal{L}_{cont}$ on UT-Zappos and MIT-States respectively. Detailed investigation of this behavior is out of scope for this work.

Third, we note that all the methods perform better on UT-Zappos as compared to MIT-States. This is due to differences in the underlying distributions of the two datasets, which makes one dataset more "difficult" to learn. Another reason is that MIT-states dataset was automatically labeled, which leaves the room for more incorrect labels. Our quantitative analysis (Sec. 6.4.5) provides a better view of this issue. Briefly, even though the algorithm retrieves semantically similar images but they will not be considered correct due to noisy labelling. For instance, for the second query in Fig. 6.4, we can see that the second and fifth image are semantically similar. But according to automated labels, only the first and third images are "correct images". This issue has also been discussed in depth by Nawaz et al.[154].

Table 6.3 presents the results of CZSL task. We observe that **ContraNet** significantly outperforms all the competitive methods. The second best method is also **ContraNet** without $\mathcal{L}_{dec}$. In this variant, we remove both the image and text decoder of our model. Without these "regularizers" on the common embedding space, there is slight drop in performance but still $\mathcal{L}_{cont}$ and classification losses are able to maintain a decent performance.

Finally, we observe from Tables 6.1, 6.2 and 6.3, that the variant of **ContraNet** without $\mathcal{L}_{cont}$ results in a huge degradation in the performance. This reinforces our claim that contrastive learning of image and text embeddings in a common space helps in learning better representations. These representations are then particularly useful for the compositional learning tasks.

6.4.3 Visualization in Latent Space

We plot the representations of images projected unto the common embedding space to visualize how well they are separated with respect to their labels. Fig.6.3 presents these visualizations using t-SNE for the UT-Zappos and MIT-States datasets. It shows that compositions with similar state-object are closer to each other than other compositions.

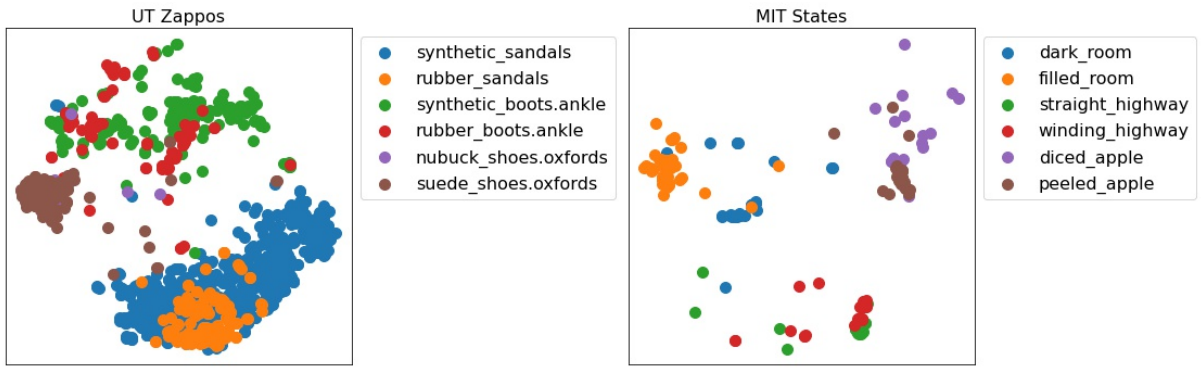

Figure 6.3: Visualization (using t-SNE) of test image instances projected unto the learned common embedding space

Method	MIT-States			UT-Zappos		
	Top-1	Top-2	Top-3	Top-1	Top-2	Top-3
AnalogousAttr	1.4	-	-	18.3	-	-
LabelEmbed	13.4	17.6	22.4	25.8	39.8	52.4
Red Wine	13.1	21.2	27.6	40.3	52.8	67.1
AoP	14.2	19.6	25.1	46.2	56.6	69.2
TAFE-Net	16.4	26.4	33.0	33.2	45.8	57.3
SymNet	19.9	28.2	33.8	52.1	67.8	76.0
ContraNet	**22.1**	**33.7**	**38.2**	**54.6**	**73.1**	**80.4**
- without $\mathcal{L}_{cont}$	14.7	18.8	24.6	39.9	50.2	62.4
- without $\mathcal{L}_{dec}$	<u>20.4</u>	<u>31.2</u>	<u>35.9</u>	<u>53.3</u>	<u>69.7</u>	<u>78.6</u>

Table 6.3: Performance comparison on the CZSL split. The best performance is in bold and the second best is underlined.

Method	R@1	R@5	R@10
Raw Image features only	3.3	12.8	20.9
Raw Text features only	7.4	21.5	32.7
Concatenation [Image,Text]	11.8	30.8	42.1
Show and Tell	11.9	31.0	42.0
AoP	8.8	27.3	39.1
Relationship	12.3	31.9	42.9
FiLM	10.1	27.7	38.3
SymNet	11.2	29.5	41.4
TIRG	12.2	31.9	43.1
ComposeAE	<u>13.9</u>	<u>35.3</u>	<u>47.9</u>
ContraNet -Ω_{res}	**14.5**	**40.7**	**51.4**
ContraNet -Ω_{rot}	13.9	39.1	50.8
ContraNet -Ω_{mlp}	13.7	36.9	48.8
ContraNet -Ω_{res} without Common Embedding Space	12.0	31.2	42.9

Table 6.4: Model performance comparison on MIT-States. The best number is in bold and the second best is underlined.

Method	R@1	R@10	R@50
Raw Image features only	3.5	22.7	43.7
Raw Text features only	1.0	12.3	21.8
Concatenation [Image,Text]	11.9	39.7	62.6
Show and Tell	12.3	40.2	61.8
Param Hashing	12.2	40.0	61.7
Relationship	13.0	40.5	62.4
FiLM	12.9	39.5	61.9
SymNet	11.7	38.6	60.4
TIRG	14.1	42.5	63.8
ComposeAE	<u>22.8</u>	<u>55.3</u>	<u>73.4</u>
ContraNet -Ω_{res}	**24.0**	**58.4**	**79.2**
ContraNet -Ω_{rot}	18.5	54.8	76.3
ContraNet -Ω_{mlp}	22.9	56.7	77.5
ContraNet -Ω_{res} without Common Embedding Space	17.8	50.6	71.1

Table 6.5: Model performance comparison on Fashion200k. The best number is in bold and the second best is underlined.

6.4.4 Discussion of Results for the Image Retrieval Based on Multi-modal Query Task

Tables 6.4 and 6.5 presents the results of this task for MIT-States and Fashion200k respectively. Depending on the composition function Ω, we get three variants of **ContraNet** . Ω_{res} means that we compose the modalities via the gated and residual connections introduced by TIRG. Ω_{rot} denotes that composition is modelled as rotation of image embeddings from common embedding space. This idea was proposed by ComposeAE[158]. However, unlike them, we do not employ any rotational symmetry loss or complex rotation. The third variant, Ω_{mlp}, simply means that we fuse the two modalities via MLP (two-fully connected layers with non-linear activations).

First, we note that the variant **ContraNet** -Ω_{res} achieves impressive gains over all the competitive methods. Specifically, on MIT-Sates in terms of R@10 metric, the performance improvement over the second best method (ComposeAE) is 7.3% and 19.3% over the third best method (TIRG). Similar performance trend can be seen for fashion200k dataset. This supports our claim that the semantic meaning hidden in the labels helps in learning better composition.

Second, we observe that several methods (like AoP, SymNet, FiLM) could not outperform simple baseline like concatenation of raw image and text features. While TIRG, ComposeAE and **ContraNet** consistently outperforms this baseline by a significant margin on both benchmark datasets.

Third, we note that all the variants of **ContraNet** outperform or achieve comparable performance to ComposeAE and TIRG. Although **ContraNet** -Ω_{res} comes out to be the best variant, but **ContraNet** -Ω_{mlp} and **ContraNet** -Ω_{rot} are also able to perform the task reasonably well. We hypothesize that the reason is that the projection modules to the common embedding space learned in the task 1 have learned the underlying shared information between two modalities quite well. Thus, a simple composition function like MLP is also able to achieve competitive results. In order to confirm this intuition, we drop the projection modules to the common embedding space from our best performing variant, i.e., **ContraNet** -Ω_{res} without Common Embedding Space. Consequently, we observe an enormous drop in performance on both datasets.

6.4.5 Qualitative Results

Fig.6.4 presents some qualitative retrieval results for MIT-States dataset. For the first query, we see that two “burnt bush images are retrieved. We can observe that other retrieved images share the same semantics and are visually similar to the target images. In second query, we note that same objects in different states can look drastically different. This highlights the importance of incorporating the text information in the composed representation. Some qualitative retrieval results for Fashion200k dataset are presented in Fig. 6.5. In these results, we observe that the model is able to capture the style and color information quite well. In the first row, we see similar sleeveless dresses with sequin. Similarly, in the second query, the model successfully images from the same product category, i.e. jacket and skirts. Moreover, the retrieved images seem to follow the desired

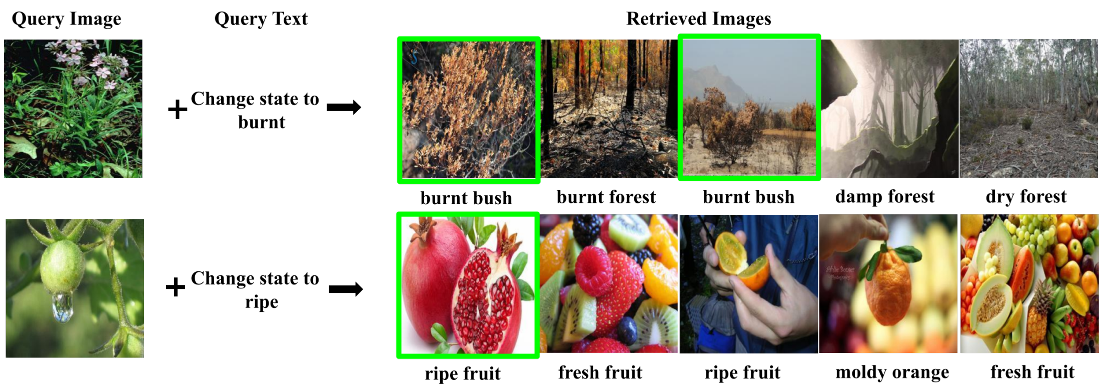

Figure 6.4: Qualitative results for image retrieval task: MIT-States

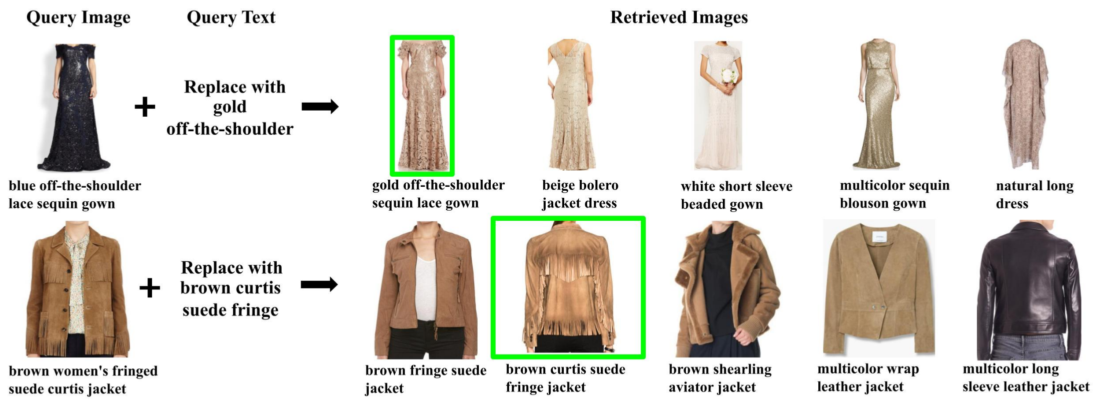
Query Image
Query Text
Retrieved Images
+ Replace with gold off-the-shoulder
blue off-the-shoulder lace sequin gown
gold off-the-shoulder sequin lace gown
beige bolero jacket dress
white short sleeve beaded gown
multicolor sequin blouson gown
natural long dress
+ Replace with brown curtis suede fringe
brown women's fringed suede curtis jacket
brown fringe suede jacket
brown curtis suede fringe jacket
brown shearling aviator jacket
multicolor wrap leather jacket
multicolor long sleeve leather jacket

Figure 6.5: Qualitative results for image retrieval task: Fashion200k

modifications expressed in the query text remarkably well. It is pertinent to highlight that the captions under the images are the ground truth. They are not available to the model as additional input during training or inference.

7 Leveraging Variational Graph Embeddings for Compositional Zero-Shot Learning

This chapter is based on the following publication:

Anwaar, Muhammad Umer*; Pan, Zhihui and Kleinsteuber, Martin. On Leveraging Variational Graph Embeddings for Open World Compositional Zero-Shot Learning. In Proceedings of the 30th ACM International Conference on Multimedia (MM '22), Lisbon, Portugal. Association for Computing Machinery, New York, NY, USA, 4645–4654. https://doi.org/10.1145/3503161.3547798*

7.1 Introduction

In their seminal works in vision and cognitive science, Biederman [178] and Hoffman *et al.*[179] showed the compositional nature of human perception of the visual world. This compositional nature is the motivation behind the feature compositionality which has been exploited by modern vision systems. For instance, learning image features [180, 181] and transfer learning [182, 183, 184]. It is well-known that unfamiliar compositions of concepts dominate the long tailed distribution of visual concepts [185, 186]. However, the bulk of the research is focused on composition in feature space, whereas composition of concepts in the classifier space has received less attention.

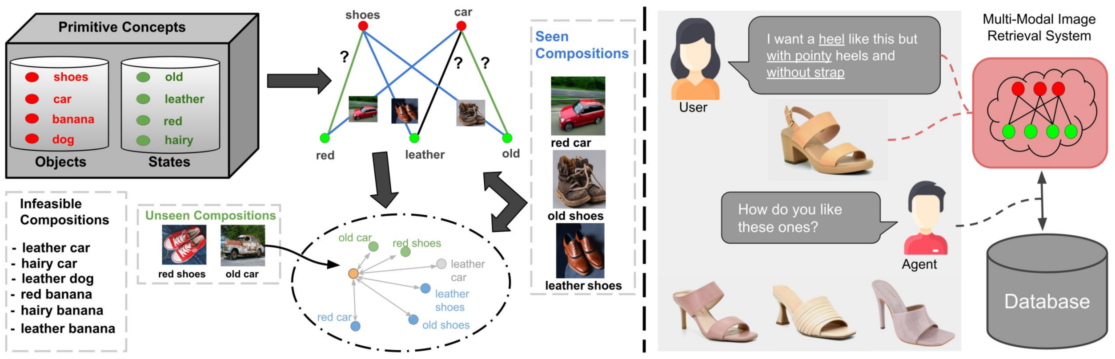

Figure 7.1: Left: Open World CZSL Task. Most compositions of primitive concepts in the model search space are infeasible. **Right**: A potential application scenario, where CZSL aids the image retrieval system in addressing the multi-modal query.

Recently, several works have focused on the so-called Compositional Zero-Shot Learning (CZSL) problem [156, 171, 155]. The task is to learn the composition of concepts (objects and states) in such a way that even their novel compositions can be classified without any supervision. Interestingly, the baseline and state-of-the-art (SOTA) methods in CZSL assume that the novel composition of concepts (objects and states) in the model output space is already known at test time. The only exception is the work by Mancini *et al.*[187], which do not impose any such constraint on the model output space. Following [187], we refer to the former as Close-World (CW) CZSL task and the latter as Open-World (OW) CZSL task. The left part of Fig. 7.1 presents the OW-CZSL task, whereas the right part presents a potential application scenario where OW-CZSL can help E-commerce websites. It is to be noted that underlined object and states can be recognized via Part-of-speech (POS) tagging [188, 189], but this is out of scope of this work.

In this work, we propose **CVGAE** , a Variational Graph Autoencoder (VGAE) based approach for tackling both CW and OW CZSL tasks in a principled way. We argue that objects and states are being generated from a prior distribution. They are treated as nodes of a graph and an edge between them indicates the existence of a compositional pair. This formulation enables us to learn the latent representation of our primitive concepts in a space of reduced dimensionality along with the feasibility of their compositions (edges). The embeddings of the compositional pairs are obtained by simply concatenating the respective state and object node embeddings. This is in contrast to CGE[190], which requires nodes as well as all compositional pairs in the graph. Such problem formulation by Naeem *et al.*[190] limits its applicability for the real-world scenarios. We discuss the computational complexity in detail in subsection 7.4.4.

We hypothesize that there are two fundamental obstacles in learning models which exhibit compositional generalization, i.e., (1) zero-shot nature of unseen labels (distribution-shift) and (2) learning the "disentanglement" of primitive concepts from training samples. In this paper, we do not claim to overcome these challenges. Rather, we recognize these obstacles and draw inspiration from them in developing our model. This is the reason for choosing a generative process for the primitive concepts in our graph. We hypothesize that keeping only the primitives in the graph aids in learning of primitive concepts in a disentangled way. Moreover, the variational modelling process also ensures that **CVGAE** learns the underlying distribution rather than learning such spurious correlations which hurt inference at test time. This enables **CVGAE** to achieve better compositional generalization.

Our model focuses on learning the distribution of the primitive concepts and predicting the existence (feasibility) of an edge between them by exploiting the graph information. We argue that once good primitive representations are learned in a disentangled way, it is more likely to be stable across training and test compositional labels. Following the literature [190, 187, 155], we employ a pretrained image model to extract image features. We learn a mapping of the compositional pair embeddings and image features on the common embedding space. In this space, we employ contrastive loss to learn embeddings which are visually grounded and semantically meaningful. That is, by maximizing the similarity between image embeddings with target compositional pair embeddings against random pairs and vice-versa through a bidirectional contrastive loss. Another big advantage of **CVGAE** is that we exploit the graph information and augment the supervised contrastive loss with VGAE loss. This is in contrast to the current SOTA methods, which overwhelmingly rely on supervisory labels for learning. Such purely supervised methods disregard the fact that the underlying data lives on a much complicated manifold than what sparse labels could capture. This often leads to good task-specific solutions, rather than learning the multiple semantic concepts in the data. We argue that representations learnt by **CVGAE** better capture the semantic meaning of primitive concepts. We evaluate the validity of our approach via experiments on three benchmark datasets (See Sec. 7.4).

Our main contributions are summarized below:

- We propose **CVGAE** , a variational graph-based approach for tackling both CW and OW CZSL tasks. We argue that the node embeddings and edges between the primitive concepts (object and states) are sufficient for achieving good compositional generalization.
- We show that **CVGAE** is computationally cheaper than the current graph-based SOTA method, CGE.
- **CVGAE** achieves better compositional generalization than SOTA methods on three benchmark datasets, i.e., for OW-CZSL task, the performance gain with respect to best harmonic mean metric is 12.5% on MIT-States, 8.3% on UT-Zappos and 25% on C-GQA dataset.

7.2 Related Work

7.2.1 Compositional Zero-Shot Learning

Early works in CZSL see state (attribute) as a mid-level feature between visual patterns and object recognition. It has been proved to help various tasks, including zero-shot recognition, image description [163, 164, 165], visual question answering [166] and so on. [165, 191] treat object as combination of states (attributes). [192] develop a random forest algorithm on top of attribute classifiers. [193] propose using continuous instead of binary state classifier and infer to unseen objects by relating its description to seen objects. Skip-gram word embeddings can be used for both state and object to make predictions on unseen relations [194].

Instead of being used as an intermediate feature, attribute classification becomes another goal of learning in compositional learning, besides object classification. [195] uses matrix factorization to get unseen object-specific attribute classifiers from seen ones. [156] combine attribute and object classifier via a transformation network to get the composition classifier. Instead of training classifiers for primitives, [171] use the state and object label to control a gated network calculating a triplet loss of (image, state, object).

Besides the visual presentation, the semantic information of primitives are also utilized to generalize to unseen compositions. [157] models object using its semantic representation and attribute as linear transformation for the object. [155] learns the coupling and decoupling mechanism between attributes and objects.

7.2.2 Open World Recognition

Most previous works conduct their experiments on the closed-world setting, where they assume prior knowledge of all test compositions. This weakens their generalizability to practical settings. We adopt the more realistic open-world setting from [187]. The search space now contains every possible combination of state and object. We show that most previous models exhibit performance degradation in this setting, while our model can handle this new problem well with the help of Graph Neural Network(GNN).

Both our work and CGE [190] use GNN to get the desired composition representation. However, we propose a variational graph approach which requires much less nodes. This is especially advantageous in the open-world setting considering the already enormous search space.

7.3 Methodology

7.3.1 Task Formulation

Let $\mathcal{X}$ denote the set of images, $\mathcal{S}$ denote the set of states, $\mathcal{O}$ the set of objects and $\mathcal{Y} = \mathcal{S} \times \mathcal{O}$ denote the set of all the compositions. The set $\mathcal{Y}$ can also be expressed as a union of real compositional labels, denoted as $\mathcal{Y}_r$, and hypothetical compositional labels, denoted as $\mathcal{Y}_h$. i.e., $\mathcal{Y} = \mathcal{Y}_r \cup \mathcal{Y}_h$. $\mathcal{Y}_r$ consists of the “real” labels, i.e., we have images in the dataset for these labels. $\mathcal{Y}_r$ is partitioned into two disjoint subsets, depending on whether the corresponding label was “seen” by the model during training or not. We refer to them as “seen” $\mathcal{Y}_s$ and “unseen” $\mathcal{Y}_u$ respectively. i.e., $\mathcal{Y}_r = \mathcal{Y}_s \cup \mathcal{Y}_n$ and $\mathcal{Y}_s \cap \mathcal{Y}_n = \emptyset$. We can write the training set as: $\mathcal{T} = \{(x, y)|x \in \mathcal{X}, y \in \mathcal{Y}_s\}$, where $\mathcal{Y}_s \subset \mathcal{Y}_r$.

In this work, following [190, 187, 171, 196], we adopt the generalized compositional zero-shot split where the test set includes images from both $\mathcal{Y}_s$ and $\mathcal{Y}_u$. The goal of the model $f : \mathcal{X} \to \mathcal{Y}_\xi$ is to predict compositional labels in a space $\mathcal{Y}_\xi \subseteq \mathcal{Y}$. Depending on the output space $\mathcal{Y}_\xi$ of the model, we get two variants of this task: (1) Close-World (CW) CZSL i.e., $\mathcal{Y}_\xi \equiv \mathcal{Y}_r$ and (2) Open-world (OW) CZSL i.e., $\mathcal{Y}_\xi \equiv \mathcal{Y}$. In other words, in CW CZSL, the set of unseen compositions $\mathcal{Y}_u$ is assumed to be known a priori. Thus, we consider $\mathcal{Y}_h = \emptyset$. Whereas, OW CZSL does not have this assumption, which makes it a more challenging task and closer to real-world application scenarios.

7.3.2 Proposed Method: CVGAE

Suppose an undirected and unweighted graph $\mathcal{G}$ consisting of N nodes, with adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and a matrix $\boldsymbol{F} \in \mathbb{R}^{N\times m}$ of m-dimensional node features. For **CVGAE** , $N = |\mathcal{S}| + |\mathcal{O}|$ and each row in $\boldsymbol{F}$ consists of word embedding of the corresponding state or object. This is in contrast to CGE [190], which requires an additional $|\mathcal{Y}_\xi|$ nodes, corresponding embeddings and edges. Such formulation makes CGE computationally very expensive than our approach (see subsection 7.4.4).

CVGAE jointly encodes the information in nodes and edges in a h dimensional latent space. Let us denote the latent node embedding by $\boldsymbol{z}_i \in \mathbb{R}^h$. We assume that these latent random variables $\{\boldsymbol{z}_1, \boldsymbol{z_2}, \cdots, \boldsymbol{z_N}\}$ follow standard gaussian distribution. These latent variables are stacked into a matrix $\boldsymbol{Z} \in \mathbb{R}^{N \times h}$. The joint distribution can be mathematically expressed as

$$p(\mathcal{G}, \boldsymbol{Z}) = p(\boldsymbol{Z}) p_\theta(\mathcal{G}|\boldsymbol{Z}), \tag{7.1}$$

$$p(\boldsymbol{Z}) = \prod_{i=0}^{N} p(\boldsymbol{z}_i), \tag{7.2}$$

$$p(\boldsymbol{z}_i) = \mathcal{N}(\boldsymbol{0}, \mathrm{diag}(\boldsymbol{1})) \ \forall i. \tag{7.3}$$

For an unweighted and undirected graph $\mathcal{G}$, we follow [1] and restrict the decoder to reconstruct only edge information from the latent space. The edge information can be represented by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ where a_{ij} refers to the element in i^{th} row and j^{th} column. If an edge exists between node i and j, we have $a_{ij} = 1$ else 0. Thus, the decoder is given by

$$p_\theta(\boldsymbol{A}|\boldsymbol{Z}) = \prod_{(i,j)=(1,1)}^{(N,N)} p_\theta(a_{ij} = 1|\boldsymbol{z_i}, \boldsymbol{z_j}), \tag{7.4}$$

$$p_\theta(a_{ij} = 1|\boldsymbol{z_i}, \boldsymbol{z_j}) = \sigma(< \boldsymbol{z_i}, \boldsymbol{z_j} >), \tag{7.5}$$

where $< \cdot, \cdot >$ denotes dot product and $\sigma(\cdot)$ is the logistic sigmoid function. It is to be noted that an additional restriction is imposed when sampling the nodes, i.e., $\boldsymbol{z_i} \in \mathcal{S}$ and $\boldsymbol{z_j} \in \mathcal{O}$. In order to ensure computational tractability, we introduce the approximate posterior. i.e.,

$$q_\phi(\boldsymbol{Z}|\mathcal{G}) = \prod_{i}^{N} q_\phi(\boldsymbol{z_i}|\mathcal{G}) \tag{7.6}$$

$$q_\phi(\boldsymbol{z_i}|\mathcal{G}) = \mathcal{N}\Big(\boldsymbol{\mu_i}(\mathcal{G}), \mathrm{diag}(\boldsymbol{\sigma_i^2}(\mathcal{G}))\Big) \tag{7.7}$$

where $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{F})$. We employ GraphSAGE [38] as node encoder to learn $\boldsymbol{\mu}_i(.)$ and $\boldsymbol{\sigma}_i(.)$ Reparameterization trick [9] is used to obtain samples of $q_\phi(\boldsymbol{Z}|\mathcal{G})$ from the mean and variance.

The training objective should be such that the model is able to generate new data and recover graph information from the embeddings simultaneously. We aim to learn the free parameters of our model such that the log probability of $\mathcal{G}$ is maximized i.e.,

$$\begin{aligned} log\Big(p(\mathcal{G})\Big) &= log\Big(\int p(\boldsymbol{Z}) p_\theta(\mathcal{G}|\boldsymbol{Z}) \ d\boldsymbol{Z}\Big) \\ &= log\Big(\int \frac{q_\phi(\boldsymbol{Z}|\mathcal{G})}{q_\phi(\boldsymbol{Z}|\mathcal{G})} p(\boldsymbol{Z}) p_\theta(\mathcal{G}|\boldsymbol{Z}) \ d\boldsymbol{Z}\Big) \\ &= log\Big(\mathbb{E}_{\boldsymbol{Z} \sim q_\phi(\boldsymbol{Z}|\mathcal{G})} \Big\{\frac{p(\boldsymbol{Z}) p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\}\Big), \end{aligned} \tag{7.8}$$

In order to ensure computational tractability, we use Jensen's Inequality [49] to get ELBO bound of Eq. (7.8). i.e.

$$\begin{aligned} log\Big(p(\mathcal{G})\Big) &\geq \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z})p_\theta(\mathcal{G}|\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \quad (7.9)\\ &= \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(p_\theta(\mathcal{G}|\boldsymbol{Z})\Big)\Big\}\\ &+ \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(\frac{p(\boldsymbol{Z})}{q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big)\Big\} \quad (7.10) \end{aligned}$$

We follow Kipf *et al.*[1] and restrict the decoder $p_\theta(\mathcal{G}|\boldsymbol{Z})$ to reconstruct only edge information from the latent space. The edge information is contained in the adjacency matrix $\boldsymbol{A}$. In other words, we choose the decoder to be an edge decoder i.e., $p_\theta(\boldsymbol{A}|\boldsymbol{Z})$.

$$\begin{aligned} log\Big(p(\mathcal{G})\Big) &\geq \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(p_\theta(\boldsymbol{A}|\boldsymbol{Z})\Big)\Big\}\\ &- D_{KL}\Big(q_\phi(\boldsymbol{Z}|\mathcal{G})||p(\boldsymbol{Z})\Big) \quad (7.11) \end{aligned}$$

where, D_{KL} denotes the Kullback-Leibler (KL) divergence between the prior and approximate posterior distributions.

By using (2.2), (2.3), (2.7) and (2.8), the loss function can be formulated as negative of ELBO bound (7.11) i.e.,

$$\begin{aligned} \mathcal{L}_{\text{ELBO}} = &\sum_{i=1}^{N} D_{KL}\Big(\mathcal{N}\Big(\boldsymbol{\mu}_i(\mathcal{G}), \boldsymbol{\sigma}_i^2(\mathcal{G})\Big) \;||\; \mathcal{N}(\boldsymbol{0}, \text{diag}(\boldsymbol{1}))\Big)\\ &- \mathbb{E}_{\boldsymbol{Z}\sim q_\phi(\boldsymbol{Z}|\mathcal{G})}\Big\{log\Big(p_\theta(\boldsymbol{A}|\boldsymbol{Z})\Big)\Big\}. \quad (7.12) \end{aligned}$$

7.3.3 Network Architecture

Fig. 7.2 presents the overview of the architecture of **CVGAE** , a variational graph based approach for learning composition of multiple semantic concepts in images. Here, $\psi(\cdot)$ denotes the pre-trained image model (e.g. ResNet-18), which takes an image as input and returns image features $\boldsymbol{x}$ in a d-dimensional space. We get the embeddings for all the compositional pairs of interest $\mathcal{Y}_\xi$ by concatenating respective state and object node embeddings. i.e., for each $y \in \mathcal{Y}_\xi$, we have the pair embeddings $\boldsymbol{e} = concat(\boldsymbol{z}, \boldsymbol{z'})$, where $\boldsymbol{z} \in \mathcal{S}$ and $\boldsymbol{z'} \in \mathcal{O}$. We project these embeddings onto a common multi-modal embedding space via separate projection modules. These modules are denoted by $\phi_e : \mathbb{R}^{2h} \mapsto \mathbb{R}^k$ and $\phi_i : \mathbb{R}^d \mapsto \mathbb{R}^k$ for compositional pair and image respectively.

In this common embedding space, we aim to learn such representations which better capture the underlying cross-modal dependencies. Our aim is to maximize the similarity between the projected compositional pair embedding $\phi_e(\boldsymbol{e})$ and the corresponding projected image features $\phi_i(\boldsymbol{x})$. We adopt a deep metric learning (DML) approach to train **CVGAE** . Let $\kappa(\cdot,\cdot)$ denote the similarity kernel, which we implement as a dot product

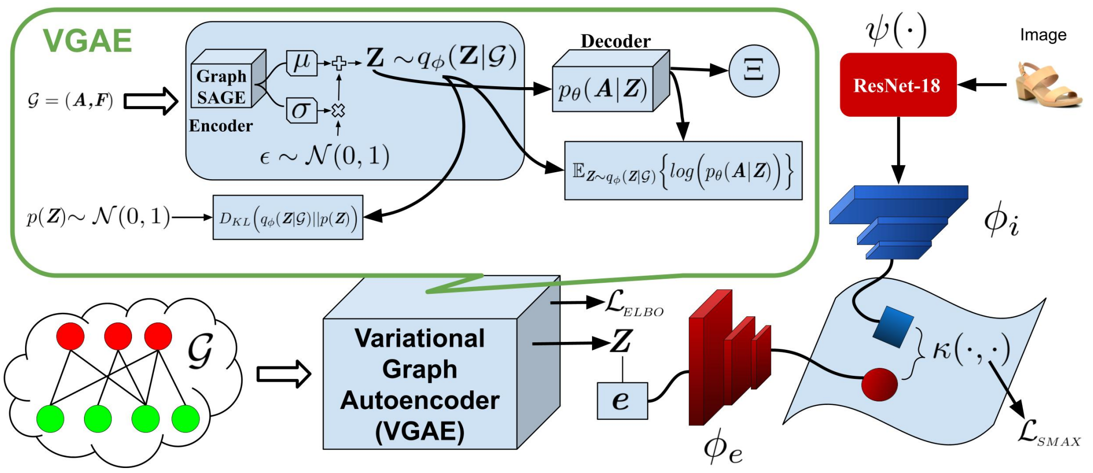
VGAE
$\mathcal{G} = (\boldsymbol{A}, \boldsymbol{F})$
Graph SAGE
Encoder
μ
σ
$\mathbf{Z} \sim q_\phi(\mathbf{Z}|\mathcal{G})$
$\epsilon \sim \mathcal{N}(0,1)$
Decoder
$p_\theta(\boldsymbol{A}|\boldsymbol{Z})$
Ξ
$\mathbb{E}_{Z\sim q_\phi(Z|\mathcal{G})}\{log(p_\theta(\boldsymbol{A}|\boldsymbol{Z}))\}$
$p(\boldsymbol{Z}) \sim \mathcal{N}(0,1)$
$D_{KL}(q_\phi(\boldsymbol{Z}|\mathcal{G})||p(\boldsymbol{Z}))$
$\psi(\cdot)$
Image
ResNet-18
ϕ_i
$\mathcal{G}$
Variational Graph Autoencoder (VGAE)
$\mathcal{L}_{ELBO}$
Z
$\boldsymbol{e}$
ϕ_e
$\kappa(\cdot,\cdot)$
$\mathcal{L}_{SMAX}$

Figure 7.2: CVGAE : a variational graph-based approach for tackling both CW and OW CZSL tasks. The key idea is that the node embeddings and edges between the primitive concepts (object and states) are sufficient for achieving good compositional generalization.

between its inputs. The objective is to learn a similarity metric, $\kappa(\cdot,\cdot) : \mathbb{R}^k \times \mathbb{R}^k \mapsto \mathbb{R}$, between $\phi_e(\boldsymbol{e})$ and $\phi_i(\boldsymbol{x})$.

In other words, **CVGAE** should learn to map semantically similar points from the data manifold in $\mathbb{R}^{2h} \times \mathbb{R}^d$ onto metrically close points in $\mathbb{R}^k$. Analogously, the model should push the projected image features away from the wrong compositional pair embeddings in $\mathbb{R}^k$. For each training sample $(x^{(j)}, y^{(j)})$ in the mini-batch of size B, we normalize the similarity between the projected compositional pair embedding $\phi_e(\boldsymbol{e}^{(j)})$ and the corresponding projected image features $\phi_i(\boldsymbol{x}^{(j)})$ by dividing it with the sum of similarities between $\phi_i(\boldsymbol{x}^{(j)})$ and all the projected compositional pair embeddings. We denote this loss as $\mathcal{L}_{e\mapsto i}$.

$$\mathcal{L}_{e\mapsto i} = \frac{1}{B}\sum_{j=1}^{B} -\log\left\{\frac{\exp\{\kappa(\phi_e(\boldsymbol{e}^{(j)}), \phi_i(\boldsymbol{x}^{(j)})\}}{\sum_{r=1}^{|\mathcal{Y}_\xi|}\exp\{\kappa(\phi_e(\boldsymbol{e}^{(r)}), \phi_i(\boldsymbol{x}^{(j)})\}}\right\}, \tag{7.13}$$

In the other direction, we also divide this similarity with the sum of similarities between $\phi_e(\boldsymbol{e}^{(j)})$ and all the images in the batch. We denote this loss as $\mathcal{L}_{i\mapsto e}$.

$$\mathcal{L}_{i\mapsto e} = \frac{1}{B}\sum_{j=1}^{B} -\log\left\{\frac{\exp\{\kappa(\phi_e(\boldsymbol{e}^{(j)}), \phi_i(\boldsymbol{x}^{(j)})\}}{\sum_{b=1}^{B}\exp\{\kappa(\phi_e(\boldsymbol{e}^{(j)}), \phi_i(\boldsymbol{x}^{(b)})\}}\right\}, \tag{7.14}$$

The total loss can be written as:

$$\mathcal{L} = \mathcal{L}_{ELBO} + \lambda_{ei}\ \mathcal{L}_{e\mapsto i} + \lambda_{ie}\ \mathcal{L}_{i\mapsto e}. \tag{7.15}$$

7.3.4 Feasibility Score

Mancini *et al.*[187] propose two variants of their model CompCos, with separate loss functions for the CW-CZSL and OW-CZSL scenarios. The main difference is that in the OW-CZSL setting, they employ a *"feasibility score"* of the unseen compositional pairs in the loss function. In contrast to their approach, **CVGAE** gets the *feasibility score*, denoted by Ξ, of the unseen compositional pairs as a *byproduct* of using the VGAE. This is because our approach makes use of both the structural information existing in the graph and the variational modelling of the nodes. i.e., **CVGAE** employs an edge decoder, see Eq. (7.5), to reconstruct the edges of the graph. This enables us to leverage the graph structure of states and objects in such a way that "edges" can be predicted between them via a simple dot product of their embeddings. If an edge (link) is predicted between a state-object pair, then it is treated as a proxy of the feasibility of their composition. This major advantage arises naturally due to VGAE and we do not need to change our loss function to incorporate *feasibility scores.* They are simply calculated during inference time and we employ threshold (τ) to make a hard decision on the feasibility of a compositional pair.

$$\Xi = \mathbf{1}(\boldsymbol{z}\boldsymbol{z'}^T \geq \tau), \quad \tau \in [0,1]. \tag{7.16}$$

The threshold τ is calculated using validation set. But for all our experiments we fixed $\tau = 0.2$. Since the performance is quite similar for a long range of τ, i.e., from 0.05 to 0.3.

7.4 Experiments

7.4.1 Experimental Setup

In our experiments, we use three benchmark datasets, namely: MIT-States[174], UT Zappos [175] and C-GQA [190]. MIT-States consists of 53753 diverse real-world images where each image is described by an object-attribute composition label, i.e. an attribute (state) and a noun (object), e.g. "broken glass". UT-Zappos is a dataset of only shoes with fine-grained annotations. A composition label consists of shoe type-material pair. Recently, Naeem *et al.*[190] proposed C-GQA dataset which is built on Stanford GQA dataset. This dataset has the largest label space among the publicly available CZSL datasets. Table 7.1 presents the detailed statistics of the three datasets. Following [176, 171, 190, 187, 155], we use the Generalized CZSL (GCZSL) split of these benchmark datasets. In this split, the model performance is evaluated on both seen and unseen pairs. For the image retrieval task, we follow the same train-test split and evaluation protocol as used by ComposeAE [158], the SOTA method for this task. i.e., the split ensures that there is no overlap between training and testing queries in terms of objects.

Baselines: We compare the results of **CVGAE** with several baselines as well as state-of-the-art (SOTA) methods.

AoP [157] employs GloVe vectors [197] for objects. They model states (attributes) as linear transformation matrices ("operators") and their product with the object embeddings yields corresponding pair embeddings. The compositional pair with the minimum distance to image embedding is considered the prediction of the model.

LabelEmbed (LE+) was introduced by [157]. It employs Glove vectors for both state and object. It projects the image, state and object vectors into a common semantic space. The joint embeddings are then fed to the classifier.

Task-driven Modular Networks (TMN) [171] modifies a set of modules (fully connected layers operating in semantic concept space) through a gating function. They propose to re-weight these primitive modules for generalizing to unseen compositions.

SymNet [155] learns object embeddings showing symmetry under different state transformations. They propose that such symmetry constraints enable learning better embeddings for the CZSL task.

CompCos [187] proposes that the cosine similarity among objects and states can be used as a proxy to estimate the feasibility of each unseen composition. They argue that similar objects share similar states whereas dissimilar objects are less likely to share similar states.

CGE [190] learns a compatibility function between image features and classes of seen and unseen compositions from a graph. The graph consists of not only the primitive concepts but also the compositional pairs of interest.

Implementation Details: It has been recently shown that improvements in reported results in deep metric learning are exaggerated and performance comparisons are done unfairly [132]. In our experiments, we took special care to ensure fair comparison. We follow the same evaluation protocol and metrics as current SOTA methods [190, 187, 171]. Following [187, 155, 157], all the methods considered in this paper use 512-dimensional image features extracted by ResNet18 model pretrained on ImageNet [177]. Resnet18 model is not fine-tuned on the CZSL datasets due to fairness concerns for the baseline methods i.e., since the baseline methods report results with fixed ResNet18 model, thus it would be unfair to report the results of baselines without searching and choosing the optimum learning rate of the image model for the baselines. We employ the same initial graph embeddings as CGE [190], i.e., word2vec [198] for UT-Zappos and C-GQA; and concatenation of word2vec and fasttext[199] embeddings for MIT-States. We use the validation set to determine the hyper-parameters, e.g., learning rate, weights of the losses etc. We employ Adam [118] optimizer with learning rate of $5e^{-5}$. The weights of the losses are: $\lambda_{ei} = 10$, and $\lambda_{ie} = 0.01$. We repeat each experiment 10 times and report the average performance of the models.

Metrics We follow the same evaluation protocol and metrics as current SOTA methods [190, 187, 171]. For the task of GCZSL, we follow the evaluation protocol from TMN [171] and use the same metrics as proposed by them. Namely: best accuracy on only images of seen/unseen compositions (*best seen/best unseen*), best harmonic mean (*best HM*) and Area Under the Curve (AUC) for seen and unseen accuracies by varying the bias values. For the task of image retrieval, following the evaluation protocol from TIRG [127], we use recall at rank k ($R@k$), as our evaluation metric. $R@k$ estimates the proportion of queries where the target (ground truth) image is within the top k retrieved images.

	MIT-States	UT-Zappos	C-GQA
# pairs in output space i.e., $\lvert\mathcal{Y}_\xi\rvert$ for CW/OW	1962 / 28175	116 / 192	9378 / 394110
Percentage of hypothetical pairs in OW $\lvert\mathcal{Y}_h\rvert/\lvert\mathcal{Y}_\xi\rvert$	92.7%	39.6%	97.6%
# states/objects	115 / 245	16 / 12	453 / 870
# images (train/val/test)	30k/10k/13k	23k/3k/3k	26k/7k/5k
# seen pairs (train/val/test)	1262/300/400	83/15/18	6963/1173/1022
# unseen pairs (val/test)	300/400	15/18	1368/1047

Table 7.1: Dataset statistics for CZSL Task

Method	**Concepts**		**Close-World**				**Open-World**			
	State	Object	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC
AoP	21.1	23.6	14.3	17.4	9.9	1.6	16.6	5.7	4.7	0.7
LE+	23.5	26.3	15.0	20.1	10.7	2.0	14.2	2.5	2.7	0.3
TMN	23.3	26.5	20.2	20.1	13.0	2.9	12.6	0.9	1.2	0.1
SymNet	26.3	28.3	24.4	25.2	16.1	3.0	21.4	7.0	5.8	0.8
CompCos	27.9	31.8	25.3	24.6	16.4	4.5	25.4	**10.0**	8.9	1.6
CGE	**27.9**	32.0	**28.7**	25.3	17.2	5.1	25.6	9.3	8.7	1.5
CVGAE	25.6	**32.3**	28.5	**25.5**	**18.2**	**5.3**	**27.3**	9.9	**10.0**	**1.8**

Table 7.2: MIT-States: Comparison of performance on benchmark datasets (in %): We report state and object accuracy of the primitive concepts; area under the curve (AUC), best seen, unseen and harmonic mean (HM) accuracies of the compositional pairs in both close-world and open-world settings. Best performance is in bold.

Method	**Concepts**		**Close-World**				**Open-World**			
	State	Object	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC
AoP	38.9	69.6	59.8	54.2	40.8	25.9	50.9	34.2	29.4	13.7
LE+	41.2	69.2	53.0	61.9	41.0	25.7	60.4	36.5	30.5	16.3
TMN	40.8	69.9	58.7	60.0	45.0	29.3	55.9	18.1	21.7	8.4
SymNet	40.5	71.2	53.3	57.9	39.2	23.9	53.3	44.6	34.5	18.5
CompCos	44.7	73.5	59.8	62.5	43.1	28.7	**59.3**	46.8	36.9	21.3
CGE	45.0	73.9	56.8	**63.6**	41.2	26.4	55.3	46.2	38.5	21.6
CVGAE	**55.0**	**77.2**	**65.0**	62.4	**49.8**	**34.6**	58.6	**48.4**	**41.7**	**22.2**

Table 7.3: UT-Zappos: Comparison of performance on benchmark datasets (in %): We report state and object accuracy of the primitive concepts; area under the curve (AUC), best seen, unseen and harmonic mean (HM) accuracies of the compositional pairs in both close-world and open-world settings. Best performance is in bold.

Method	Concepts		Close-World				Open-World			
	State	Object	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC
AoP	8.3	12.5	11.8	3.9	2.9	0.3	20.2	0.9	3.1	0.1
LE+	7.4	15.6	16.1	5.0	5.3	0.6	20.9	0.9	3.1	0.1
TMN	9.7	20.5	21.6	6.3	7.7	1.1	-	-	-	-
SymNet	14.5	20.2	25.2	9.2	9.8	1.8	24.6	2.1	5.1	0.4
CompCos	9.5	27.5	27.0	10.5	11.7	2.3	25.9	1.8	4.8	0.4
CGE	12.7	26.9	27.5	11.7	11.9	2.5	26.0	1.4	4.0	0.2
CVGAE	**28.2**	**34.9**	**28.2**	**11.9**	**13.9**	**2.8**	**26.6**	**2.9**	**6.4**	**0.7**

Table 7.4: C-GQA: Comparison of performance on benchmark datasets (in %): We report state and object accuracy of the primitive concepts; area under the curve (AUC), best seen, unseen and harmonic mean (HM) accuracies of the compositional pairs in both close-world and open-world settings. Best performance is in bold.

7.4.2 Discussion of Results: CZSL Tasks

We compare **CVGAE** with the state of the art methods on three benchmark datasets, i.e. MIT-States, UT-Zap50k and C-GQA, for both close world and open world settings. It can be seen from Table 7.4 that **CVGAE** outperforms all other methods and establishes a new state-of-the-art for both close world and open world CZSL tasks.

Closed World Results. We observe from Table 7.47.2) that **CVGAE** achieves a leading 5.3 AUC and 18.2 on the best harmonic mean metric on the MIT-States dataset. We can also observe comparable performance on other metrics. We note that even though CGE exhibit a higher accuracy on the state and seen composition predictions, **CVGAE** achieve an overall better results, thanks to more balanced predictions between unseen and seen compositions, as indicated by the harmonic mean metric. On the UT-Zappos dataset (see Table 7.47.3), we observe a huge performance improvement by **CVGAE** over SOTA methods. i.e., AUC and harmonic mean show an significant improvement of 18% (29.3 to 34.6) and 10.7% (45 to 49.8) respectively, over the second best method, TMN. Similarly, we note from Table 7.47.4. that on C-GQA dataset significant performance gains are achieved by our approach in primitive concepts as well as compositional pairs. These consistent performance gains supports on all 3 datasets support our claim that by adopting the variational approach **CVGAE** better captures the underlying data distribution than current SOTA methods.

Open World Results. The challenge of the open world setting can be seen clearly from the performance degradation experienced by all the models (see right side of Table 7.4). Due the the significantly huge search space, the accuracy on unseen compositions, and as a consequence also the AUC and the best harmonic mean, decreases rapidly. On both MIT-States and C-GQA, the performance of the models is halved on these three metrics. On the other hand, the impact is relatively milder for the UT-Zappos dataset. This is because the percentage of hypothetical pairs in the OW setting is just 40% for

Method	**MIT-States**		**UT-Zappos**	
	R@1	R@10	R@1	R@10
ComposeAE	13.9	47.9	64.2	69.1
AoP	8.8	39.1	24.2	59.4
SymNet	11.2	41.4	37.4	62.8
CompCos	17.5	47.3	42.7	63.9
CGE	9.4	35.9	59.7	67.7
CVGAE	**17.7**	**49.7**	**65.4**	**70.3**

Table 7.5: Image retrieval results on MIT states and UT Zappos

UT-Zappos, whereas it is ∼93% and ∼98% for MIT-States and CGQA respectively (see Table 7.1).

We observe from the results that **CVGAE** also exhibits superior performance in the open world setting on all 3 datasets. On MIT-States, with respect to best HM metric, **CVGAE** achieves 15% and 12.5% improvement over CGE and Compcos respectively. Similar performance trend can be observed on UT-Zappos and C-GQA datasets, where **CVGAE** outperforms other models on AUC and best HM metric. The performance gains on the most challenging dataset for OW setting, i.e., C-GQA, are the most promising. For instance, **CVGAE** improves the best HM by 8.3% and 25.5% in comparison to the second best method on UT-Zappos and C-GQA datasets, respectively. This strengthens our claim that by adopting a variational graph based approach and focusing on learning good representation of primitive concepts in a disentangled way leads to better compositional generalization.

One interesting observation is that, on C-GQA dataset, **CVGAE** experiences a huge performance degradation of 75% when comparing its AUC with CW setting. Whereas, CGE experiences a more dramatic performance degradation of 92%. This shows that even when the number of hypothetical pairs is 98% (i.e., C-GQA, see Table 7.1), **CVGAE** is quite effective relative to SOTA methods, in tackling the infeasible pairs in the open world setting. This empirically shows the validity of our hypothesis that once good representation of primitive concepts are learnt in a disentangled way, they are more likely to be stable across seen, unseen and infeasible compositions. Thus, such node embeddings and edges between the primitive concepts are sufficient for achieving good compositional generalization.

7.4.3 Discussion of Results for the Image Retrieval Based on Multi-modal Query Task

In order to gain further evidence that the embeddings learnt via **CVGAE** capture the underlying data distribution better than current SOTA methods, we consider image retrieval task. Recently, this task has been well-studied in the literature. The task is inspired from the potential application scenario, presented in Fig. 1. i.e., a user sends a multi-modal query (an image with a text request for some change in the attributes) and the task is to retrieve the images with desired modifications from the database. In the literature, this task is simplified by limiting the text request to only one word attribute (state) change. ComposeAE [158] is the current state-of-the-art method for this task. We follow the same evaluation protocol as ComposeAE [158]. From Table 7.5, we observe that **CVGAE** outperforms not only the competitive methods (i.e., CompCos and CGE) but also ComposeAE on both datasets. Specifically, on MIT-States and UT-Zappos, **CVGAE** achieves a Recall at position 1 (R@1) of 17.7 and 65.4, whereas CGE achieves 9.4 and 59.7 respectively. Similarly for R@10, the performance improvement over all the methods (including ComposeAE) is quite significant.

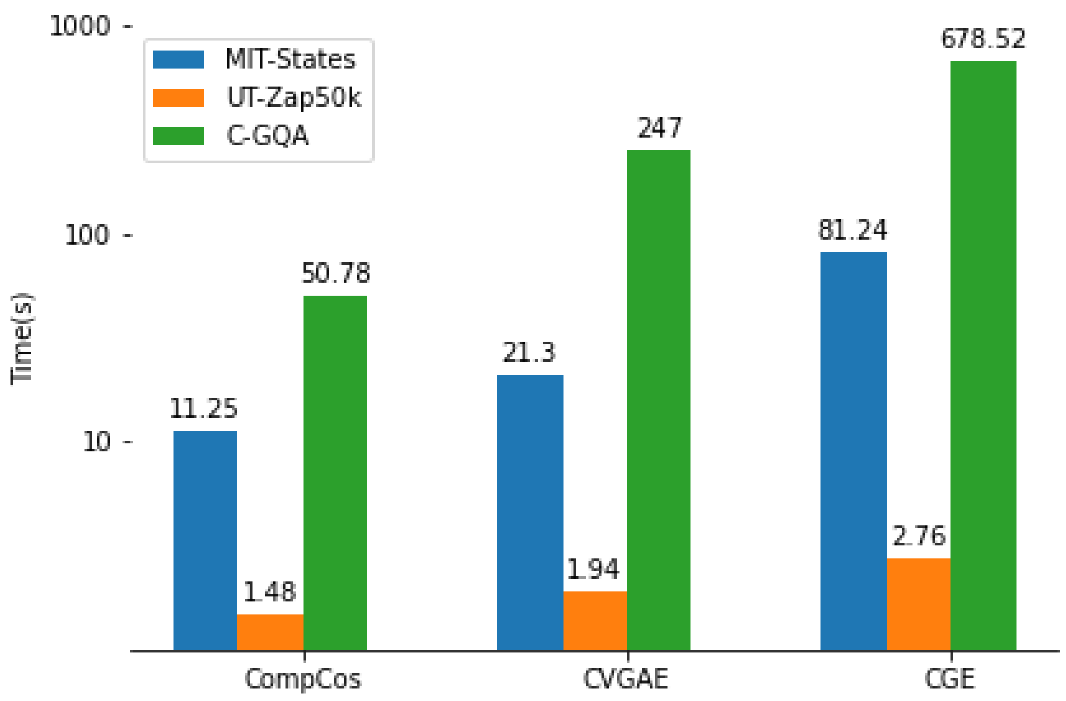

Figure 7.3: Comparison of training times (one epoch) of top-3 approaches on the OW-CZSL task. The y-axis is in log scale. We can see that **CVGAE** is quite time-efficient than its direct graph-based competitor, CGE.

7.4.4 Computational Complexity

We now compare the computational complexity of **CVGAE** with the graph-based SOTA method CGE. The graph in **CVGAE** consists of N nodes, where $N = |\mathcal{S}|+|\mathcal{O}|$. Whereas the graph proposed in CGE [190], consists of $N_{CGE} = |\mathcal{S}| + |\mathcal{O}| + |\mathcal{Y}_\xi|$ nodes. In general, $|\mathcal{Y}_\xi| \gg |\mathcal{S}|+|\mathcal{O}|$, for instance, in the OW setting, $|\mathcal{Y}_\xi| = |\mathcal{S}| \times |\mathcal{O}|$. Such formulation makes CGE model computationally more expensive than **CVGAE** . Formally, we assume that the graph is sparse with the number of edges $|\mathcal{E}| = \mathcal{O}(N)$ and each node is represented by m-dimensional features. Thus, the time and memory complexity for a L-layered GCN

Method	**Concepts**		**Close-World**		**Open-World**	
	State	Object	HM	AUC	HM	AUC
CGAE	24.7	30.3	11.4	2.5	4.2	0.4
ContraNet $_{GAT}$	27.4	33.2	12.8	2.3	5.8	0.6
ContraNet $_{GCN}$	27.9	33.7	13.5	2.6	6.1	0.6
ContraNet $_{GCN-II}$	27.6	33.4	13.1	2.5	5.9	0.6
ContraNet	**28.2**	**34.9**	**13.9**	**2.8**	**6.4**	**0.7**

Table 7.6: Impact of Graph Encoder and Non-Variational approach on the performance of **ContraNet** on the C-GQA dataset

model can be written as $\mathcal{O}(LNm^2)$ and $\mathcal{O}(LNm + Lm^2)$ respectively. We can clearly see that **CVGAE** will be much more efficient than CGE, since $|N_{CGE}| \gg |N|$ for all real-world datasets. e.g. for C-GQA dataset, CGE requires $3.94 \cdot 10^5$ nodes and $1.07 \cdot 10^4$ nodes for OW and CW setting, respectively. While **CVGAE** requires only 1323 nodes for both settings. It is to be noted that the C-GQA dataset contains only 453 states and 870 objects, whereas any real-world application scenario will involve much more number of states and objects. Thus, CGE fails to tackle the problem of scalability.

We also empirically compare the training times of top-3 performing algorithms in Fig. 7.3 for the OW-CZSL task. As some of the methods (e.g. CGE) are more resource intensive than others, we select AWS instance type `g4dn.4xlarge`[1] for fair comparison of training times. All hyperparameters are kept the same, e.g. batchsize is set to 128, for all three methods. As expected, since Compcos does not employ any graph, so it is the fastest among the three methods. We observe that between the two graph methods, **CVGAE** is quite time efficient. e.g. for MIT-States, it is ~4 times faster than CGE. This provides empirical evidence for lower computational complexity of **CVGAE** as discussed above.

7.4.5 Ablation Studies

We have conducted various ablation studies, in order to gain insight into the performance of **CVGAE** . Table 7.6 presents the quantitative results of these studies.

Non-variational approach: CVGAE follows a variational approach to learn the underlying data distributions of the concepts. Instead of VGAE, we used a non-variational GAE to quantify the effect of using variational approach on the performance. Row 1 in Table 4 shows that there is a drop in performance for both the CW and OW settings. This strengthens our hypothesis that it is better to employ a variational approach for learning the representation of concepts. This especially aids in the OW setting, where the drop in performance is most significant. e.g., on AUC metric, there is a performance degradation of 42.8%.

[1]`https://aws.amazon.com/ec2/instance-types/`

k	MIT-States	UT-Zappos	C-GQA
	HM [CW/OW]	HM [CW/OW]	HM [CW/OW]
400	16.9/9.2	46.2/37.3	13.6/6.1
800	17.8/9.6	46.7/38.1	**13.9/6.4**
1200	17.9/9.8	48.6/38.9	13.5/6.2
1600	**18.2/10.0**	**49.8/41.7**	13.2/5.7
2000	18.1/9.7	49.5/40.8	13.1/5.8

Table 7.7: Effect of embedding dimension of common embedding space (k) on the performance of **CVGAE** measured in terms of best Harmonic Mean (HM) in the Close-World (CW) and Open-World (OW) settings

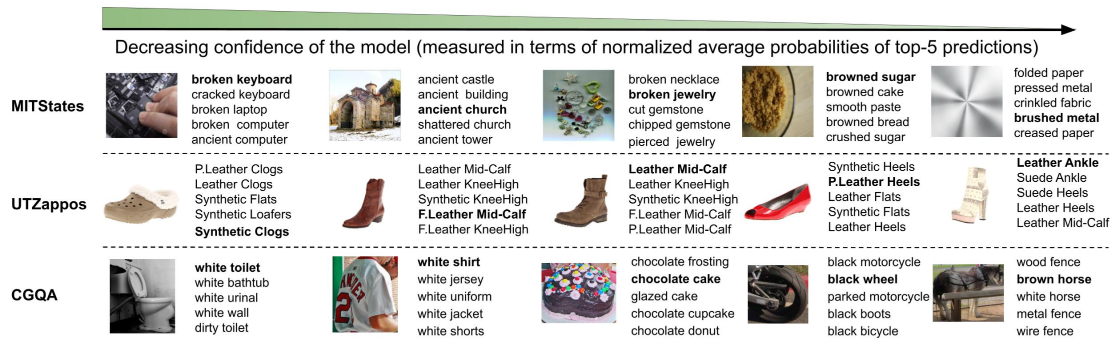
Decreasing confidence of the model (measured in terms of normalized average probabilities of top-5 predictions)
MITStates
broken keyboard, cracked keyboard, broken laptop, broken computer, ancient computer
ancient castle, ancient building, **ancient church**, shattered church, ancient tower
broken necklace, **broken jewelry**, cut gemstone, chipped gemstone, pierced jewelry
browned sugar, browned cake, smooth paste, browned bread, crushed sugar
folded paper, pressed metal, crinkled fabric, **brushed metal**, creased paper
UTZappos
P.Leather Clogs, Leather Clogs, Synthetic Flats, Synthetic Loafers, **Synthetic Clogs**
Leather Mid-Calf, Leather KneeHigh, Synthetic KneeHigh, **F.Leather Mid-Calf**, F.Leather KneeHigh
Leather Mid-Calf, Leather KneeHigh, Synthetic KneeHigh, F.Leather Mid-Calf, P.Leather Mid-Calf
Synthetic Heels, **P.Leather Heels**, Leather Flats, Synthetic Flats, Leather Heels
Leather Ankle, Suede Ankle, Suede Heels, Leather Heels, Leather Mid-Calf
CGQA
white toilet, white bathtub, white urinal, white wall, dirty toilet
white shirt, white jersey, white uniform, white jacket, white shorts
chocolate frosting, **chocolate cake**, glazed cake, chocolate cupcake, chocolate donut
black motorcycle, **black wheel**, parked motorcycle, black boots, black bicycle
wood fence, **brown horse**, white horse, metal fence, wire fence

Figure 7.4: Qualitative Retrieval Results: Top-5 Predictions of **CVGAE** . Bold indicates the true compositional label.

Impact of graph encoder: We now look at the impact of the graph encoder on the performance of **CVGAE** . We employ three graph encoders, namely: Graph Attention Network (GAT) [200], Graph Convolution Network (GCN) [12] and its recent improved version (GCN-II) [201]. Due to space constraints, we show the results on the C-GQA dataset only. Table 7.6 shows the effect of graph encoder on the performance of **CVGAE** . We can observe that there is no significant decrease in the performance. The performance of GCN variants is consistently better than attention based graph encoder (GAT).

Common Embedding Space: Table 7.7 shows the effect of dimensions of the common embedding space on the performance of the model. In the interest of space, we only report best Harmonic Mean (HM) for the Close World (CW) and Open World (OW) settings. Rest of the metrics also follow a similar trend in performance. We observe from the results that the choice of embedding dimension has a significant impact on the performance for all three datasets. Specifically, the performance degradation is 7.1%, 7.2% and 5.6% with respect to CW-HM metric on MIT-States, UT-Zappos and C-GQA datasets respectively.

7.4.6 Qualitative Results

Fig. 7.4 presents some qualitative retrieval results for three datasets. Each row represents image samples from one dataset and each column contains the top-5 composition label predictions by **CVGAE** for an image. First we observe that the model is able to retrieve the compositional labels which share similar semantics. For instance, "ancient church" is actually a subset of "ancient building" in a conceptual world. Similarly, "broken keyboard" and "cracked keyboard" capture same semantic meaning. Second, we note that although model is able to predict several correct compositional labels but model is rewarded only if it predicts the label as annotated in the dataset. We argue that the task of CZSL is inherently multi-label task and benchmark datasets should be amended to cater for this aspect. i.e., prediction should be considered true if the model predicts any of the compositional concepts present in the image. Otherwise, the model has to learn the annotator bias to perform well. For instance, the rightmost image in the third row has both "wood fence" and "brown horse" in it. Similarly, second image in the first row can be seen as both "ancient building" and "ancient church".

8 Conclusion

In this thesis we have leveraged the advances in representation learning and applied them to several problems with graph-structured or multi-modal data. First, we looked at the issue of over-pruning in variational graph autoencoder (VGAE). We demonstrated that the way VGAE [1] deals with this issue results in a latent distribution which is quite different from the standard gaussian prior. We proposed an alternative model based approach EVGAE that mitigates the problem of over-pruning by encouraging more latent variables to actively play their role in the reconstruction. We showed that EVGAE also has a better generative ability than VGAE[1] i.e. better matching between learned and prior distribution. In addition, our proposed algorithm achieves competitive results on the link prediction task.

Second, we focused on simultaneously learning community detection and node representation. We proposed a scalable generative method, **J-ENC** , that learns a single community-aware node embedding for both the representation of the node and its context. Our novel approach is scalable due to its low complexity, i.e. $\mathcal{O}(|\mathcal{E}|K + NKd)$. The experiments on several graph datasets show that **J-ENC** consistently outperforms all the competitive baselines on node classification, overlapping community detection and non-overlapping community detection tasks. Moreover, training the **J-ENC** is highly time-efficient than its competitors.

Third, we investigated the necessity of incorporating sequential semantics in metapath for recommendation. Instead of mixing up multi-hop messages in graphs, we devised a unified GNN framework called **PEAGNN**, which explicitly performs independent information aggregation on generated metapath-aware subgraphs. Specifically, a metapath fusion layer is trained to learn the metapath importance and adaptively fuse the aggregated metapath semantics in an end-to-end fashion. We also introduce a contrastive connectivity regularizer called entity-awareness which exploits the first-order local structure of the graph. For first-order local structure exploitation, the entity-awareness, a contrastive connectivity regularizer, is employed on user nodes and item nodes representation. The experiments on three public datasets have shown the effectiveness of our approach in comparison to competitive baselines. It is to be noted that this work explored the potential of explicitly injecting semantics in metapath into GNNs. However, the advance performance achieved by our model relies on meaningful metapaths. In practice, selection of representative metapaths is a challenging task and thus, it opens the door for future research on the automatic sequential semantics discovery.

Fourth, we explore the task of image retrieval from a database based on a multi-modal query, i.e., the query is specified in the form of an image and natural language expressions describing the desired modifications in the query image. We propose ComposeAE to compose the representation of source image rotated with the modification text in a complex space. This composed representation is mapped to the target image space and a similarity metric is learned. Based on our novel formulation of the problem, we introduce a *rotational symmetry loss* in our training objective. Our experiments on three datasets show that ComposeAE consistently outperforms SOTA method on this task.

Fifth, we address both the image retrieval and CZSL tasks jointly. In order to ensure that the model learns different states of an object and can recognise even unseen combinations of them, we adopt a contrastive learning approach. That is, we propose **ContraNet** , which learns text-aware image representations in a common embedding space in a contrastive manner. The core idea is to ensure that the (state, object, image) triples are distinguishable across the training batches. This prompts **ContraNet** to learn efficient embeddings by simultaneously leveraging the information in multiple modalities. The contrastive loss is complemented by image and text reconstruction losses that not only regularize the learnt embedding but also attempt to preserve the information in the individual modalities. The resultant architecture achieves SOTA performance on generalized CZSL as well as multi-modal query-based image retrieval task, as demonstrated by the extensive comparison with many competitive baselines on three popular benchmark datasets.

At the end, we employ the insights learned in VGAE and compositional learning to tackle the more challenging task of open-world CZSL task. This task is very close to real-world settings and puts less restrictions on the unseen data. We proposed a variational graph autoencoder based approach, **CVGAE** , to learn composition of objects and states, in both close-world and open-world settings. **CVGAE** learns the variational embeddings of the primitive concepts and treats the edge between them as proxy for the feasibility of the compositions. The proposed approach outperforms the current state-of-the-art methods by significant margin on three widely-used benchmark datasets. We also showed that **CVGAE** is computationally cheaper than the SOTA method, CGE [190]. The performance on the image retrieval task validates our claim that **CVGAE** learns better representations of primitive concepts than current SOTA methods.

We now briefly discuss the limitations of current algorithms and datasets for the compositional zero-shot learning tasks. The discussion below highlights the roadblocks which hinders current approaches from achieving compositional generalization. Our work tries to solve part of the problem and get rid of some prior assumptions. But there is still a strong assumption that composition can be only done with two concepts. This is quite limiting in practice. For instance, the approaches discussed in this work can not directly compose "old red car". Although, in comparison to CGE [190], **CVGAE** can be extended relatively easy to more than two concepts i.e., by concatenating the graph embeddings of all three concepts of interest. Furthermore, all the datasets considered also assume that there exists only two kinds of primitive concepts i.e., objects and their states. There is a need of new datasets which consider more kinds of primitive concepts. Another limitation of current work can be observed in Fig. 7.4 i.e., the confidence of model is not the same in predicting the composition of different concepts. The robustness of predictions across different concepts plays a crucial role in the real-world application of the model. This aspect needs to be investigated further in light of recent works [202, 203, 204]. Another important limitation of our work is that we did not take into account the label noise in the datasets. For instance, we can see that there are certain images, like the two rightmost images in the first row of Fig. 7.4, which are difficult for even humans to uniquely annotate. Such issues hinder the model from achieving a good prediction accuracy. In fact, one work has even argued against using MIT-States dataset due to 70% label noise [205].

The journey towards achieving compositional generalization is an arduous one. The discussion above points towards some avenues where future research is required to achieve the lofty goal of compositional generalization.

Bibliography

[1] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[4] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

[5] S. Qi, X. Ning, G. Yang, L. Zhang, P. Long, W. Cai, and W. Li. Review of multi-view 3d object recognition methods based on deep learning. *Displays*, 69:102053, 2021.

[6] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[7] D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2):604–624, 2020.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[9] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[10] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pages 6530–6539, 2017.

[11] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[13] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*, 2017.

[14] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. Sampling of graph signals with successive local aggregations. *IEEE Transactions on Signal Processing*, 64(7):1832–1843, 2015.

[15] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[16] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[17] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *33rd International Conference on Machine Learning (ICML 2016)*, 2016.

[18] S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

[19] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[20] R. A. Khan, M. U. Anwaar, and M. Kleinsteuber. Epitomic variational graph autoencoder, 2020. `arXiv:2004.01468`.

[21] J. Xie, S. Kelley, and B. K. Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):1–35, 2013.

[22] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[23] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596, 2013.

[24] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156. IEEE, 2013.

[25] F.-Y. Sun, M. Qu, J. Hoffmann, C.-W. Huang, and J. Tang. vgraph: A generative model for joint community detection and node representation learning. In *Advances in Neural Information Processing Systems*, pages 514–524, 2019.

[26] J. Leskovec and J. J. Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[28] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[29] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[30] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

[31] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. 2019.

[32] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 377–386, 2017.

[33] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin. A unified framework for community detection and network representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1051–1065, 2018.

[34] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton. Gemsec: Graph embedding with self clustering. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining*, pages 65–72, 2019.

[35] Y. Jia, Q. Zhang, W. Zhang, and X. Wang. Communitygan: Community detection with generative adversarial nets. In *The World Wide Web Conference*, pages 784–794, 2019.

[36] R. A. Khan, M. U. Anwaar, O. Kaddah, Z. Han, and M. Kleinsteuber. Unsupervised learning of joint embeddings for node representation and community detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 19–35. Springer, 2021.

[37] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[38] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs, 2018. `arXiv:1706.02216`.

[39] M. U. Anwaar, Z. Han, S. Arumugaswamy, R. A. Khan, T. Weber, T. Qiu, H. Shen, Y. Liu, and M. Kleinsteuber. Metapath- and entity-aware graph neural network for recommendation, 2021. URL: `https://arxiv.org/abs/2010.11793`, `doi:10.48550/ARXIV.2010.11793`.

[40] A. Saha, M. M. Khapra, and K. Sankaranarayanan. Towards building large scale multimodal domain-aware conversation systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[41] M. U. Anwaar, Z. Pan, and M. Kleinsteuber. URL: `https://arxiv.org/abs/2204.11848`, `doi:10.48550/ARXIV.2204.11848`.

[42] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[43] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[44] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. *arXiv preprint arXiv:1806.01242*, 2018.

[45] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[46] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[47] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[48] T. Kipf. gae. `https://github.com/tkipf/gae/blob/716a46ce579a5cdba84278ccf71891d59e420988/gae/optimizer.py#L33`. URL: `https://github.com/tkipf/gae/blob/716a46ce579a5cdba84278ccf71891d59e420988/gae/optimizer.py#L33`.

[49] E. W. Weisstein. Jensen's inequality. From MathWorld—A Wolfram Web Resource. URL: `https://mathworld.wolfram.com/JensensInequality.html`.

[50] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

[51] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[52] T. Kipf. gae. `https://github.com/tkipf/gae/issues/20#issuecomment-446260981`. URL: `https://github.com/tkipf/gae/issues/20#issuecomment-446260981`.

[53] L. Tang and H. Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.

[54] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[56] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[57] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[58] R. A. Khan. Evgae. `https://github.com/RayyanRiaz/EVGAE`, 2020.

[59] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *nature*, 466(7307):761–764, 2010.

[60] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.

[61] P. K. Gopalan and D. M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.

[62] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community preserving network embedding. In *AAAI*, volume 17, pages 203–209, 2017.

[63] J. Tang, C. Aggarwal, and H. Liu. Node classification in signed social networks. In *Proceedings of the 2016 SIAM international conference on data mining*, pages 54–62. SIAM, 2016.

[64] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[65] S. Gao, L. Denoyer, and P. Gallinari. Temporal link prediction by integrating content and structure information. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1169–1174, 2011.

[66] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 891–900, New York, NY, USA, 2015. Association for Computing Machinery. URL: `https://doi.org/10.1145/2806416.2806512`, `doi:10.1145/2806416.2806512`.

[67] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[68] M. Kozdoba and S. Mannor. Community detection via measure space embedding. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2890–2898, Cambridge, MA, USA, 2015. MIT Press.

[69] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[70] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[71] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[72] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[73] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang. Modularity based community detection with deep learning. In *IJCAI*, volume 16, pages 2252–2258, 2016.

[74] A. Bojchevski and S. Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[75] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[76] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[77] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[78] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[79] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[80] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[81] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[82] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[83] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[84] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[85] Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73, 2002.

[86] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. Springer, 2011.

[87] S. Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[88] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364, 2017.

[89] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[90] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[91] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.

[92] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, 2019.

[93] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He. Multi-graph convolution collaborative filtering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1306–1311. IEEE, 2019.

[94] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[95] R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[96] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM conference on Recommender systems*, 2013.

[97] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2018.

[98] M. Zhang and Y. Chen. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*, 2019.

[99] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144, 2017.

[100] J. Zhang, P. S. Yu, and Z.-H. Zhou. Meta-path based multi-network collective link prediction. In *Proceedings of the 20th ACM SIGKDD*, pages 1286–1295, 2014.

[101] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[102] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD*, pages 2478–2486, 2019.

[103] Y. Sun, J. Han, C. C. Aggarwal, and N. V. Chawla. When will it happen? relationship prediction in heterogeneous information networks. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 663–672, 2012.

[104] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[105] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[106] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[107] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 2017.

[108] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[109] B. Heitmann and C. Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI spring symposium*, 2010.

[110] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI HINA*, 2013.

[111] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1531–1540, 2018.

[112] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 635–644, 2017.

[113] M. Zhang, Z. Cui, M. Neumann, and Y. Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[114] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pages 4800–4810, 2018.

[115] Q. Li, Z. Han, and X.-M. Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, 2018.

[116] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[117] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[118] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[119] X. He, T. Chen, M.-Y. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670, 2015.

[120] I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. 2016.

[121] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR*, SIGIR '16, page 549–558, NY, USA, 2016. ACM.

[122] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation, 2019. `arXiv:1811.05868`.

[123] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[124] Q. Ai, V. Azizi, X. Chen, and Y. Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.

[125] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[126] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[127] N. Vo, L. Jiang, C. Sun, K. Murphy, L.-J. Li, L. Fei-Fei, and J. Hays. Composing text and image for image retrieval - an empirical odyssey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019)*, June 2019.

[128] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[129] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[130] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[131] L. Wang, Y. Li, and S. Lazebnik. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE CVPR*, pages 5005–5013, 2016.

[132] K. Musgrave, S. Belongie, and S.-N. Lim. A metric learning reality check, 2020. `arXiv:2003.08505`.

[133] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE CVPR*, 2016.

[134] Z. Zheng, L. Zheng, M. Garrett, Y. Yang, and Y.-D. Shen. Dual-path convolutional image-text embeddings with instance loss. *ACM TOMM*, 2017.

[135] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. 2018. URL: `https://github.com/fartashf/vsepp`.

[136] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He. Stacked cross attention for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–216, 2018.

[137] Y. Zhang and H. Lu. Deep cross-modal projection learning for image-text matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 686–701, 2018.

[138] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[139] H. Noh, P. Hongsuck Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 30–38, 2016.

[140] X. Han, Z. Wu, P. X. Huang, X. Zhang, M. Zhu, Y. Li, Y. Zhao, and L. S. Davis. Automatic spatially-aware fashion concept discovery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1463–1471, 2017.

[141] T. Nagarajan and K. Grauman. Attributes as operators: factorizing unseen attribute-object compositions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 169–185, 2018.

[142] X. Guo, H. Wu, Y. Cheng, S. Rennie, G. Tesauro, and R. Feris. Dialog-based interactive image retrieval. In *Advances in Neural Information Processing Systems*, pages 678–688, 2018.

[143] F. Tan, P. Cascante-Bonilla, X. Guo, H. Wu, S. Feng, and V. Ordonez. Drill-down: Interactive retrieval of complex scenes using natural language queries. In *Advances in Neural Information Processing Systems*, pages 2647–2657, 2019.

[144] B. Zhao, J. Feng, X. Wu, and S. Yan. Memory-augmented attribute manipulation networks for interactive fashion search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1520–1528, 2017.

[145] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *CoRR*, abs/1412.6614, 2015.

[146] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[147] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In P. Grünwald, E. Hazan, and S. Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 1376–1401, Paris, France, 03–06 Jul 2015. PMLR. URL: `http://proceedings.mlr.press/v40/Neyshabur15.html`.

[148] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[149] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

[150] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE ICCV*, pages 360–368, 2017.

[151] P. Isola, J. J. Lim, and E. H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015.

[152] X. Guo, H. Wu, Y. Gao, S. Rennie, and R. Feris. The fashion iq dataset: Retrieving images by combining side information and relative natural language feedback. *arXiv preprint arXiv:1905.12794*, 2019.

[153] H. Xiao. bert-as-service. `https://github.com/hanxiao/bert-as-service`, 2018.

[154] S. Nawaz, K. Janjua, I. Gallo, A. Mahmood, A. Calefati, and F. Shafait. Do cross modal systems leverage semantic relationships?, 2019. `arXiv:1909.01976`.

[155] Y.-L. Li, Y. Xu, X. Mao, and C. Lu. Symmetry and group in attribute-object compositions. In *CVPR*, 2020.

[156] I. Misra, A. Gupta, and M. Hebert. From red wine to red tomato: Composition with context. In *CVPR*, 2017.

[157] T. Nagarajan and K. Grauman. Attributes as operators: factorizing unseen attribute-object compositions. In *ECCV*, 2018.

[158] M. U. Anwaar, E. Labintcev, and M. Kleinsteuber. Compositional learning of image-text query for image retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1140–1149, January 2021.

[159] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020. `arXiv:2002.05709`.

[160] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. P. Langlotz. Contrastive learning of medical visual representations from paired images and text, 2020. `arXiv:2010.00747`.

[161] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding, 2019. `arXiv:1807.03748`.

[162] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning, 2020. `arXiv:1911.05722`.

[163] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, 2015. `doi:10.1109/ICCV.2015.425`.

[164] N. Kumar, P. Belhumeur, and S. Nayar. Facetracer: A search engine for large collections of images with faces. In *Computer Vision – ECCV 2008*, pages 340–353, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[165] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785, 2009. `doi:10.1109/CVPR.2009.5206772`.

[166] J. Koushik, H. Hayashi, and D. S. Sachan. Compositional reasoning for visual question answering. In *Proceedings of the 34 th International Conference on Machine Learning, 2017*, 2017.

[167] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL: `https://www.aclweb.org/anthology/D14-1162`, `doi:10.3115/v1/D14-1162`.

[168] C.-Y. Chen and K. Grauman. Inferring analogous attributes. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 200–207, USA, 2014. IEEE Computer Society. URL: `https://doi.org/10.1109/CVPR.2014.33`, `doi:10.1109/CVPR.2014.33`.

[169] X. Wang, F. Yu, R. Wang, T. Darrell, and J. E. Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning, 2019. `arXiv:1904.05967`.

[170] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, 2013.

[171] S. Purushwalkam, M. Nickel, A. Gupta, and M. Ranzato. Task-driven modular networks for zero-shot compositional learning. In *ICCV*, 2019.

[172] N. N. Vo and J. Hays. Localizing and orienting street views using overhead imagery. In *European Conference on Computer Vision*, pages 494–509. Springer, 2016.

[173] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[174] P. Isola, J. J. Lim, and E. H. Adelson. Discovering states and transformations in image collections. In *CVPR*, 2015.

[175] A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *CVPR*, 2017.

[176] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild, 2017. `arXiv:1605.04253`.

[177] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[178] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94 2:115–147, 1987.

[179] D. D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1984.

[180] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[181] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[182] J. Choi, M. Rastegari, A. Farhadi, and L. S. Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013.

[183] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV*, 2014.

[184] N. Patricia and B. Caputo. Learning to learn, from transfer learning to domain adaptation: A unifying perspective. In *CVPR*, 2014.

[185] R. Salakhutdinov, A. Torralba, and J. Tenenbaum. Learning to share visual appearance for multiclass object detection. In *CVPR*, 2011.

[186] Y.-X. Wang, D. Ramanan, and M. Hebert. Learning to model the tail. In *NIPS*, 2017.

[187] M. Mancini, M. Naeem, Y. Xian, and Z. Akata. Open world compositional zero-shot learning. In *34th IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2021.

[188] B. Bohnet, R. McDonald, G. Simões, D. Andor, E. Pitler, and J. Maynez. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL: `https://aclanthology.org/P18-1246`, `doi:10.18653/v1/P18-1246`.

[189] C. D. Santos and B. Zadrozny. Learning character-level representations for part-of-speech tagging. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1818–1826, Bejing, China, 22–24 Jun 2014. PMLR. URL: `https://proceedings.mlr.press/v32/santos14.html`.

[190] M. Naeem, Y. Xian, F. Tombari, and Z. Akata. Learning graph embeddings for compositional zero-shot learning. In *34th IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2021.

[191] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1629–1636, 2014. `doi:10.1109/CVPR.2014.211`.

[192] D. Jayaraman and K. Grauman. Zero-shot recognition with unreliable attributes. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3464–3472, 2014.

[193] D. Parikh and K. Grauman. Relative attributes. In *2011 International Conference on Computer Vision (ICCV)*, 2011.

[194] Z. Al-Halah, M. Tapaswi, and R. Stiefelhagen. Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5975–5984, 2016. `doi:10.1109/CVPR.2016.643`.

[195] C.-Y. Chen and K. Grauman. Inferring analogous attributes. In *CVPR*, 2014.

[196] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE TPAMI*, 41(9):2251–2265, 2018.

[197] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[198] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[199] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[200] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[201] Z. W. Ming Chen, B. D. Zengfeng Huang, and Y. Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

[202] A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[203] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 2021.

[204] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[205] Y. Atzmon, F. Kreuk, U. Shalit, and G. Chechik. A causal view of compositional zero-shot recognition. In *NeurIPS*, 2020.