Generative models meet similarity search: robust, heuristic-free and explainable retrieval models

Khoa D. Doan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Chandan K. Reddy (Chair)
Bimal Viswanath
Anuj Karpatne
Lifu Huang
Sathiya Keerthi Selvaraj

August 2, 2021
Arlington, Virginia

Keywords: learning to hash, generative modeling, similarity search, explainable retrieval

Generative models meet similarity search: robust, heuristic-free and explainable retrieval models

Khoa D. Doan

(ABSTRACT)

The rapid growth of digital data, especially visual and textual contents, brings many challenges to the problem of finding similar data. Exact similarity search, which aims to exhaustively find all relevant items through a linear scan in a dataset, is impractical due to its high computational complexity. Approximate-nearest-neighbor (ANN) search methods, especially the Learning-to-hash or Hashing methods, provide principled approaches that balance the trade-offs between the quality of the guesses and the computational cost for web-scale databases. In this era of data explosion, it is crucial for the hashing methods to be both computationally efficient and robust to various scenarios such as when the application has noisy data or data that slightly changes over time (i.e., out-of-distribution).

This Thesis focuses on the development of practical generative learning-to-hash methods and explainable retrieval models. We first identify and discuss the various aspects where the framework of generative modeling can be used to improve the model designs and generalization of the hashing methods. Then we show that these generative hashing methods similarly enjoy several appealing empirical and theoretical properties of generative modeling. Specifically, the proposed generative hashing models generalize better with important properties such as low-sample requirement, and out-of-distribution and data-corruption robustness. Finally, in domains with structured data such as graphs, we show that the computational methods in generative modeling have an interesting utility beyond estimating the data distribution and describe a retrieval framework that can explain its decision by borrowing the algorithmic ideas developed in these methods.

Two subsets of generative hashing methods and a subset of explainable retrieval methods are proposed. For the first hashing subset, we propose a novel adversarial framework that can be easily adapted to a new problem domain and three training algorithms that learn the hash functions without several hyperparameters commonly found in the previous hashing methods. The contributions of our work include: (1) Propose novel algorithms, which are based on adversarial learning, to learn the hash functions; (2) Design computationally efficient Wasserstein-related adversarial approaches which have low computational and sample efficiency; (3) Conduct extensive experiments on several benchmark datasets in various domains, including computational advertising, and text and image retrieval, for performance evaluation. For the second hashing subset, we propose energy-based hashing solutions which can improve the generalization and robustness of existing hashing approaches. The contributions of our work for this task include: (1) Propose data-synthesis solutions to improve the generalization of existing hashing methods; (2) Propose energy-based hashing solutions which exhibit better robustness against out-of-distribution and corrupted data; (3) Conduct extensive experiments for performance evaluations on several benchmark datasets in the

image retrieval domain.

Finally, for the last subset of explainable retrieval methods, we propose an optimal alignment algorithm that achieves a better similarity approximation for a pair of structured objects, such as graphs, while capturing the alignment between the nodes of the graphs to explain the similarity calculation. The contributions of our work for this task include: (1) Propose a novel optimal alignment algorithm for comparing two sets of bag-of-vectors embeddings; (2) Propose a differentiable computation to learn the parameters of the proposed optimal alignment model; (3) Conduct extensive experiments, for performance evaluation of both the similarity approximation task and the retrieval task, on several benchmark graph datasets.

Generative models meet similarity search: robust, heuristic-free and explainable retrieval models

Khoa D. Doan

(GENERAL AUDIENCE ABSTRACT)

Searching for similar items, or similarity search, is one of the fundamental tasks in this information age, especially when there is a rapid growth of visual and textual contents. For example, in a search engine such as Google, a user searches for images with similar content to a referenced image; in online advertising, an advertiser finds new users, and eventually targets these users with advertisements, where the new users have similar profiles to some referenced users who have previously responded positively to the same or similar advertisements; in the chemical domain, scientists search for proteins with a similar structure to a referenced protein. The practical search applications in these domains often face several challenges, especially when these datasets or databases can contain a large number (e.g., millions or even billions) of complex-structured items (e.g., texts, images, and graphs). These challenges can be organized into three central themes: search efficiency (the economical use of resources such as computation and time) and model-design effort (the ease of building the search model). Besides search efficiency and model-design effort, it is increasingly a requirement of a search model to possess the ability to explain the search results, especially in the scientific domains where the items are structured objects such as graphs.

This dissertation tackles the aforementioned challenges in practical search applications by using the computational techniques that learn to generate data. First, we overcome the need to scan the entire large dataset for similar items by considering an approximate similarity search technique called hashing. Then, we propose an unsupervised hashing framework that learns the hash functions with simpler objective functions directly from raw data. The proposed retrieval framework can be easily adapted into new domains with significantly lower effort in model design. When labeled data is available but is limited (which is a common scenario in practical search applications), we propose a hashing network that can synthesize additional data to improve the hash function learning process. The learned model also exhibits significant robustness against data corruption and slight changes in the underlying data. Finally, in domains with structured data such as graphs, we propose a computation approach that can simultaneously estimate the similarity of structured objects, such as graphs, and capture the alignment between their substructures, e.g., nodes. The alignment mechanism can help explain the reason why two objects are similar or dissimilar. This is a useful tool for domain experts who not only want to search for similar items but also want to understand how the search model makes its predictions.

Dedication

To my beloved family.

Acknowledgements

There have been many unforgettable memories in the past five years. Five years have passed so quickly but are seemingly enough to change a person's life and perspectives. My path to the completion of this PhD dissertation is not without challenges and obstacles. Fortunately, I did not walk this journey alone, and I would like to express my sincere gratitude to everyone who helped me on my accomplishment over all these five years.

First and foremost, I would like to extend my deepest appreciation to my advisor, Dr. Chandan K. Reddy, for his guidance and incredible support during my pursuit of this PhD degree. I learned so much from him, from conducting research and expressing an idea in the paper to collaborating with other researchers and advising junior students. I am particularly grateful for his constant complaints and criticism, which helped me become better and better in several aspects and kept me on track during the past five years. To put it simply, it would have been difficult for me to finish my degree without his advice and support, and I am forever indebted for that. I can only hope that one day I can have an opportunity to pass on what I learned from him to other students.

I would also like to thank Dr. Sathiya Keerthi Selvaraj for his inspiration in research. To him, doing research is all about learning, understanding, taking risks and of course, enjoying the accomplishments, each of which is equally important. I took risks, learned, and really enjoyed my achievements, which made the completion of this dissertation truly rewarding. I am, indeed, very fortunate for the opportunity to know and work with him.

I am thankful to other members of my dissertation committee – Dr. Bimal Viswanath, Dr. Anuj Karpatne, and Dr. Lifu Huang – for their important guidance and assistance in the completion of this dissertation. I am especially thankful for their valuable comments and critics during my proposal exam, and research, and final defenses. For me, a critique is as important as a compliment and I felt very fortunate to receive those critiques from my committee members until the last day of my defense. Thank you for that because I could see myself and my dissertation improve substantially after every session of our meetings.

Thank you to my collaborators, including Dr. Saurav Manchanda, Sarkhan Badirli (who will also finish his PhD degree very soon), Dr. Fengjiao Wang, and Dr. Avradeep Bhowmik. I still remember a countless number of hours that we spent on reading papers, sharing knowledge and perspectives, proposing original ideas, writing code, sharing the joys of having a paper

accepted, and catching up on each other's lives. Perhaps, the last part is what I enjoyed the most; I began the PhD journey alone but I reached the finish line having so many good friends beside me. For that, I am grateful.

I am also grateful to the lab members and others at the Sanghani Center for Artificial Intelligence and Data Analytics, including Ping Wang, Tian Shi, Aman Ahuja, Ming Zhu, Nurendra Choudhary, Akshita Jha, Nikhil Muralidhar, to name a few. We learned from and supported each other in research and our graduate-student lives. But again, more importantly, we have become good friends along the way with many happy memories.

Moreover, I would like to thank Roxanne Paul, Wanawsha Hawrami, Sharon Kinder-Potter, and Corinne Julien for their administrative support over the years. Special thanks to Roxanne Paul for her help and patience.

My gratitude to my parents is beyond words. It is their unconditional love and endless support that regularly encouraged me to finish this journey. I am especially grateful to my dad, who is himself a great person and contributed so much to Vietnamese society. He has always believed in me and has taught me many life lessons that were essential elements of my success.

Finally, I want to thank my wife and my daughters, who were always there for me in the past five years. I have been waiting for this moment for so many years simply to say "I did it" and "Thank you" to my wife. It was mostly her continuous love and support that motivated me to stay focus throughout all these years. I thought that my advisor was someone who would be concerned the most about my research publications and various PhD milestones, but my wife was also my second, non-academic advisor for her frequent questions of "When are you done with your paper?" or "When is your final defense?". Indeed, I am fortunate and grateful for her care, trust, and endless support. My achievement is not mine alone, because it is definitely her achievement as well.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid growth of digital data, especially unstructured content such as text and images, not only brings plenty of opportunities, but also presents many challenges in effectively working with massive databases in various applications and tasks. One of the fundamental challenges associated with such massive datasets is to *efficiently* and *effectively* search for data containing *semantically* **similar** content in these datasets. Efficiency involves the economical use of resources such as time and space, while effectiveness refers to the relevancy of the retrieved items. *Exact similarity search*, which aims to exhaustively find all the relevant content via a full linear scan of a database, is often inefficient, thus becoming impractical in practice. A complete linear scan in massive databases is not feasible, especially when the database contains millions (or billions) of samples. Thus, *approximate similarity search* algorithms that can **efficiently** focus on a much smaller subset of potentially relevant candidates and perform the necessary similarity computations only on this subset are being developed for solving this problem.

Hashing is a principled approximate similarity search that projects the original features onto a much smaller locality-preserving *binary* space so that the "candidate" subset can be efficiently discovered from this space. Compact binary codes are storage-efficient and the search-cost in the original high-dimensional space is reduced to calculating their Hamming distances in the reduced space, thus making it computationally much faster. Using bitwise XOR and bit-count operations, Hamming-distance computation takes at most two CPU instructions for 32-bit and 64-bit binary vectors in most conventional 32-bit or 64-bit systems, respectively.

1.1 Research Problems

This research aims to investigate the applications of generative models in existing similarity search methods, especially the hashing-based methods. The major research issues are discussed as follows:

Effective and domain-independent similarity-preserving unsupervised hashing models. While hashing methods ensure efficiency, an important challenge in hashing, especially unsupervised hashing, is to learn a hash function that can *effectively* preserve the semantic similarity between a pair of items, without requiring their explicit labeled similarity. In other words, an unsupervised hashing method must (1) learn a good representation of the data that captures high-level, discriminative concepts in the data and (2) preserve the neighborhood structure (also called locality) of the representations in the discrete space. In real-world applications, representation learning is a challenging task, especially when the data are complex objects such as text documents or users' activities. Most existing unsupervised hashing methods, however, rely on either hand-crafted features or domain-specific structured models which do not automatically adapt to new domains. For example, in the text hashing domain, existing hashing methods still adopt simple, less powerful extracted features such as TF-IDF [115]; in the image hashing domain, to preserve the similarity, some methods rely on various invariant features, e.g., rotation, translation, and flipping of the image [24, 48, 63], which are not always available in other domains such as advertising. Recently, autoencoders have been shown to provide a principled, domain-independent approach for representation learning. Compared to other heuristic-based representation learning approaches, representation learning in autoencoders rely on an assumption that is generally true in many domains: a good low-dimensional encoding of the data makes it easier to reconstruct the data from this encoded representation. However, vanilla autoencoders are less effective compared to well-designed structured models. One explanation for this ineffective learning stems from the fact that it is easy for the vanilla autoencoders to learn an unconditional representation of the input during training. Besides this requirement of representation learning, it is also challenging to *effectively* define the notion of *semantic similarity* for sparse, high-dimensional data. In these cases, existing hashing methods typically produce inferior results in the presence of complex feature interactions. This problem is aggravated when there is a presence of noisy data, which are prevalent in domains such as online advertising and document retrieval.

Hash function learning with fewer hyperparameters. After learning high-level, robust representation of the data, the hashing method should also learn to produce hash codes with bit balance/uncorrelation and low quantization error [134]. Bit-balance and bit-uncorrelation ensure that a uniform number of training data points are assigned to each hash code, thus minimizing the time complexity of the retrieval task in the worst and average cases [57]. Along with low-quantization error, they alleviate the chance of assigning "very" similar data points to different codes. This makes it easier to preserve the original locality structure of

the data in the binary space and improves the retrieval precision [50,57,69]. Despite possessing better retrieval performance, existing deep text-hashing methods, which learn the data representation and the hash function in end-to-end models, either ignore these objectives (for easier optimization) or construct them heuristically. We argue that the retrieval performance is significantly improved by incorporating them in learning the deep hash functions. Furthermore, heuristic objectives to ensure bit balance/uncorrelation and low-quantization error in existing methods lead to more hyperparameters, increasing the complexity of the optimization and the training time (due to the model selection process).

Better generalization and robustness with limited labeled similarity data. While supervised hashing methods demonstrate superior performance over unsupervised ones, they require human-annotated datasets. Annotating massive-scale datasets, which are common in the image hashing domain, is an expensive and tedious task. Without adequate labels, supervised methods can easily overfit or get stuck in bad local optima. Furthermore, data in retrieval applications can change frequently and may be corrupted, further aggravating the train/test distribution mismatch. Generative models can synthesize data, which helps improve the generalization of existing supervised hashing methods. Some generative models, especially the likelihood-based ones, can also benefit downstream problems such as out-of-distribution robustness and imputation of missing data. However, existing data synthesis methods, that are GAN-based, only take advantage of data synthesization and are difficult to train because of the tuning requirement of two separate networks, i.e., the generator and the discriminator.

Interpretable retrieval models. With the popularity of deep models in mainstream applications, there has also been a surge of interest in model interpretability in several domains. In domains such as graphs, data (e.g., a graph) is often represented as a collection of its part (e.g., vertices), which poses an important question: is it possible to explain the similarity of the "wholes" through some type of "correspondence" between their parts? For example, in bioinformatics and chemistry, given two similar proteins, finding the correspondence between their structural elements is an important step in many downstream scientific tasks.

1.2 Research Contributions

In this Thesis, we focus on designing robust and practical retrieval methods. The primary research contributions can be described, as follows:

End-to-end representation and heuristic-free hash function learning via adversarial regularization. We propose end-to-end *unsupervised hashing* models which simultaneously learn optimal representations of the data, and optimal hash functions directly from raw input data with minimal design heuristics in order to improve the quality of the retrieval in both small and large scale applications. We take advantage of the simpler representation learning approach of autoencoders (i.e., via reconstruction learning) and the growing power

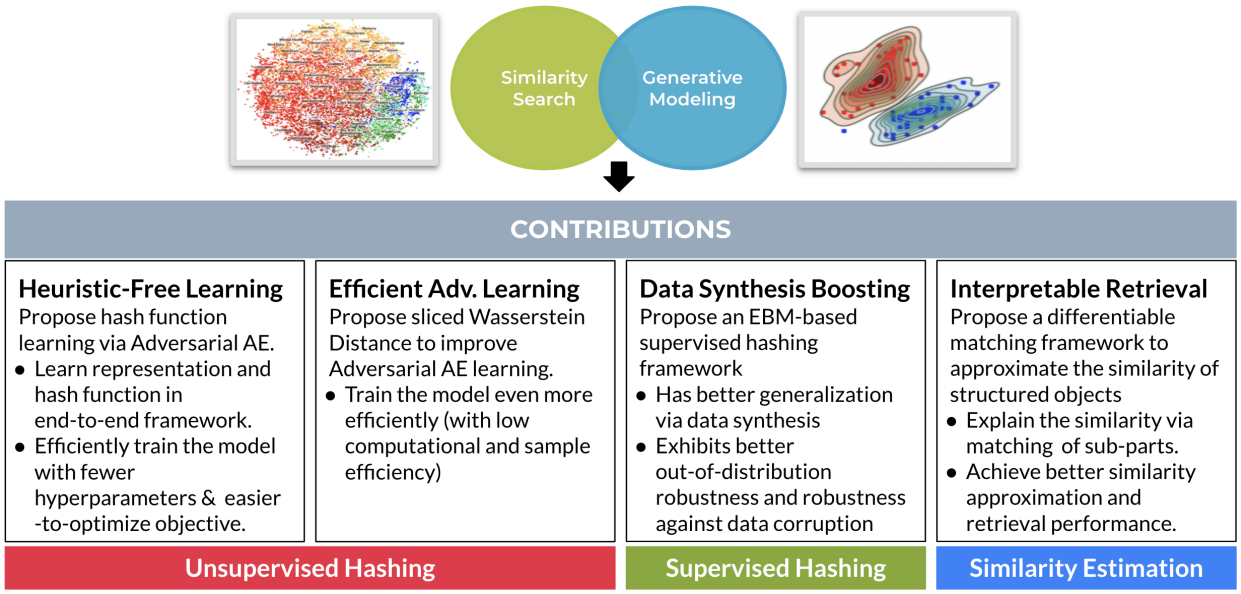


Figure 1.1: Research Problems and Contributions of this Thesis.

of deep encoder-decoder basd neural architectures. We show that end-to-end representation learning directly from the raw data is necessary to improve the retrieval performance, especially in domains with complex, high-dimensional data such as text and advertising. Then, we show that by regularizing the autoencoders with a specific Bernoulli discrete prior, the autoencoders can automatically learn optimal hash functions with minimal design heuristics or fewer hyperparameters. This is very important in large-scale applications such as text and image retrieval because efficient training is crucial, especially when the sizes of these datasets are growing exponentially every day. Finally, we propose training algorithms of the regularized autoencoders which jointly perform representation learning and hash function learning in an end-to-end manner.

Low computational and sample-requirement adversarial regularization. Despite good performance in unsupervised hashing, training the previously-proposed adversarial models by minimizing the Jensen-Shannon or Wasserstein distance is extremely difficult, especially when the dimension of the latent space increases. The scaling difficulty of the adversarial models may be related to one fundamental issue: the generalization property (i.e., the number of samples required to accurately estimate the distance, or sample complexity) of matching distributions. *Without good generalization, the retrieval performance of adversarially trained retrieval models can be sub-optimal.* We propose novel and efficient approaches to perform the distributional matching in the adversarial autoencoders by either directly estimating the Wasserstein distance from its primal domain or a modified sliced-Wasserstein distance (a Monte-Carlo estimate of the Wasserstein distance). The latter proposal, called Discrete Component-wise Wassertein (DCW) distance, has an order-of-magnitude better generalization property and an order-of-magnitude more efficient com-

putation than the existing Wasserstein-based methods. Notably, the proposed approaches also remove the discriminator network, or equivalently the difficult minimax optimization in training the adversarial autoencoders.

Efficiently-trained, energy-based generative hashing models with better generalization and robustness on limited labeled data. We propose a unified generative, *supervised learning-to-hash framework* which takes complete advantage of generative energy-based models and enjoys better generalization and robustness against missing data and out-of-distribution retrieval. Then, we propose a simple yet efficient training procedure, by extending the classic contrastive divergence framework, to train the multipurpose hashing network. This approach learns high-quality binary hash codes and achieves state-of-the-art retrieval performance on several benchmark datasets. Furthermore, it is significantly more robust to out-of-distribution retrieval compared to the existing methods and can handle significant corruption in the data with trivial drops in the retrieval performance.

Explainable retrieval models via optimal alignment. We solve a novel optimal alignment problem between two sets of vector embeddings representing two inputs when approximating their similarity. In the graph domain, this notion is equivalent to the minimum cost of transforming one graph to another in the node-embedding space and is related to classical domain-agnostic similarity measures such as Graph Edit Distance. We propose a novel alignment objective, which directly compares graphs using their (bags of) node embedding vectors, and a differentiable algorithm to learn the parameters of the alignment. The proposed model learns to approximate the similarity between a pair of graphs while providing explanations for such an approximation. The explanation can be obtained by inspecting, (e.g., visualizing) the learned node alignment between two graphs. This interpretable feature is valuable for domain experts, who also want to understand how the model makes its predictions.

Our first two contributions are in **unsupervised hashing**, while our third contribution is in **supervised hashing**. The final contribution is towards **approximating the similarity** of two structured objects, which is central to the graph retrieval task. A visualization of the research problems and our solutions to these problems in this Thesis is shown in Figure 1.1.

1.3 Thesis Organization

The remaining outline and primary contributions of this Thesis are discussed below in the given order:

Chapter 3

We first proceed with a discussion on the importance of end-to-end hash function learning on the retrieval performance via adversarially regularized autoencoders, using the digital advertising domain.

Digital advertising is performed in multiple ways i.e., contextual, display-based, and search-based. Across these avenues, the primary goal of the advertiser is to maximize the return on investment. To realize this, the advertiser often aims to target the advertisements towards a targeted set of audience as this set has a high likelihood to respond positively towards the advertisements. One such form of tailored and personalized, targeted advertising is known as look-alike modeling, where the advertiser provides a set of seed users and expects the machine learning model to identify a new set of users such that the newly identified set is similar to the seed-set with respect to the online purchasing activity. Existing look-alike modeling techniques (i.e., similarity- and regression-based) are limited in their performance due to the implicit constraints (such as good hand-crafted features and efficient hash function learning in similarity-based and large seed sets in regression-based) induced during modeling and training. In addition, the high-dimensional and sparse nature of the advertising data increases the complexity of the task.

To overcome these limitations, we propose a novel Adversarial Factorization Autoencoder that can efficiently learn a binary mapping from sparse, high-dimensional data to a binary address space using an adversarial training procedure. We demonstrate the effectiveness of our proposed approach on a dataset obtained from a real-world setting while also rigorously comparing our proposed approach with existing look-alike modeling baselines.

Chapter 4

While the adversarial autoencoders can learn good representations, thus good hash functions, their model training remains to be a challenging task in practice. In this chapter, we first show that such a problem can result in sub-optimal retrieval performance; then we replace the existing minimax optimization algorithm with a novel, single-objective optimization algorithm which directly estimates the primal Wasserstein distance, with an application in the text retrieval domain.

Searching for documents with semantically similar content is a fundamental problem in the information retrieval domain with various challenges, primarily, in terms of efficiency and effectiveness. Despite the promise of modeling structured dependencies in documents, several existing text hashing methods lack an efficient mechanism to incorporate such vital information. Additionally, the desired characteristics of an ideal hash function, such as robustness to noise, low quantization error, and bit balance/uncorrelation, are not effectively learned with existing methods. This is because of the requirement to either tune additional hyper-parameters or optimize these heuristically constructed cost functions.

We propose a Denoising Adversarial Binary Autoencoder (DABA) model that presents a novel representation learning framework. This framework can capture a structured representation of text documents in the learned hash function. Also, the proposed model takes an alternative direction in learning the hash functions by replacing the explicit formulation of the objective functions on the hash function's output space found in existing methods with an implicit adversarial matching between the output space and a discrete Bernoulli prior. This adversarial matching approach can capture all the desired characteristics of an ideal hash function more effectively. Furthermore, DABA adopts a novel single-objective optimization adversarial training procedure that minimizes the Wasserstein distance in its primal domain to regularize the encoder's output of either a recurrent neural network or a convolutional autoencoder. We empirically demonstrate the effectiveness of our proposed method in capturing the intrinsic semantic manifold of the related documents. The proposed method outperforms the current state-of-the-art shallow and deep unsupervised hashing methods for the document retrieval task on several prominent document collections.

Chapter 5

Despite good performance, training the adversarial autoencoders by minimizing the Jensen-Shannon or Wasserstein distance is extremely difficult, especially when the dimension of the latent space increases. The scaling difficulty of the adversarial autoencoders may be related to one fundamental issue: the generalization property of matching distributions. The Jensen-Shannon divergence and Wasserstein distance do not generalize, in a sense that the generated distribution cannot converge to the target distribution without an exponential number of samples. *Without good generalization, the retrieval performance can be sub-optimal.*

We propose a novel, efficient approach to learn the hash functions by employing a more generalizable variant of the Wasserstein distance that leverages the discrete properties of hashing. It has an order-of-magnitude better generalization property and an order-of-magnitude better computation than the existing Wasserstein-based hashing methods. Then, we demonstrate the superiority of the proposed model over the state-of-the-art hashing techniques on various widely used real-world image retrieval datasets using both quantitative and qualitative performance analysis.

Chapter 6

Supervised methods demonstrate superior performance over the unsupervised ones, but they can easily overfit when there are limited labeled data. To overcome such limitations of supervised hashing, some methods synthesize additional training data to improve the generalization of the hash functions. However, the use of generative models in hashing is currently limited to only data synthesis. Yet, generative models can benefit other downstream problems such as data imputation and out-of-distribution robustness.

In this chapter, we propose a unified energy-based hashing network that simultaneously learns the representations of the images and the hash function. This network learns shared representations of the images that are useful to solve multiple objectives, which include: (1) an explicit joint probability density estimation of an image and its semantic labels, (2) a contrastive hash-function learning, and (3) semantic label prediction. This framework allows the training process to develop a shared set of features as opposed to developing them redundantly in separate networks such as the GAN-based methods. Finally, since our model only trains the EBM, it requires fewer model parameters than approaches that use multiple networks. We demonstrate the advantages of our model over several state-of-the-art hashing techniques on various benchmark retrieval datasets in the image domain.

Chapter 7

In the previous chapters, we demonstrate several benefits of studying generative models in the context of hashing. To achieve data generation capability, the generative methods minimize the distance between the model's generated distribution and the observed data distribution. In this chapter, we show that this idea of distributional matching of two sets of vectors can benefit the problems of approximating the similarity of two structured objects, such as graphs and, consequently, of the retrieval tasks which are based on the corresponding similarity definitions.

We study the problem of estimating the similarity between a pair of graphs, which is critical to several graph applications such as retrieval from graph databases. While there exists several similarity definitions, GED and MCS are two of the most popular similarity measures. While the popularity of GED and MCS can be attributed to the fact that they are domain-agnostic measures, the notion of an edit sequence that transforms one graph into another graph can be important in several domains. In these domains, besides the computed similarity score between a pair of objects, such edit sequence can provide a reasoning or an explanation for the computed score. However, computing GED or MCS is intractable for larger graphs that have more than a few tens of nodes using classical combinatorial approaches. While recently proposed learning-based approaches can efficiently estimate the similarity and find similar graphs to a query graph with high accuracies, none of these methods is able to replicate the ability to explain the similarity of classical methods. We propose a novel model, called GOTSim, which directly compares graphs using their (bag-of) node embedding vectors. GOTSim directly solves the optimal assignment problem on a novel cost matrix formulation that accounts for node substitution, addition, and insertion. Besides its high-performance similarity estimation ability, GOTSim has a distinct feature: it provides an interpretable prediction. When predicting the similarity between a pair of graphs, domain experts can also understand how the model makes its predictions by visually inspecting the optimal matching between the nodes in the two graphs. GOTSim can help in advancing or confirming the domain-specific knowledge in domains that involve graphs.

Chapter 2

Review of Literature

2.1 Learning To Hash

2.1.1 The Retrieval and Hashing Problems

Semantic Similarity Search: Given a query data point e, the *similarity search problem* is to find a set of K similar data points $S(e)$ from a database $X = \{x^{(1)}, x^{(2)}, ..., x^{(n)}\}$ of n documents such that, given $sim(x, y)$ as a pre-defined similarity function that measures the similarity between two data points x and y, we have $sim(e, x) \leq sim(e, y)$ for all $x \in S(e), y \in X \setminus S(e)$. An illustration of the similarity search problem is given in Figure 2.1.

Learning To Hash: Given a dataset $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ of n examples, the goal of a learning-to-hash (or *hashing*) method is to learn a discrete-output, mapping function $\hat{\mathcal{H}} : x \to \{-1, 1\}^K$ that encodes each example x into a K-bit binary vector such that the similarity structure between the examples is preserved in the discrete space. In the supervised hashing setting, each example $x_i \in \mathcal{X}$ is associated with a label c_i. Note that this is a point-wise label of an example. Another common supervised scenario has the pairwise similarity label for each pair of examples. For most retrieval applications, pair-wise labeling is more labor-intensive because a dataset of n data points requires n^2 pairwise labelings.

Note that $\hat{\mathcal{H}}$ is a discrete-output function, which can make the learning problem intractable to solve. In practice, $\hat{\mathcal{H}}$ is typically relaxed to a continuous function, denoted as $\mathcal{H}$, whose range is $[-1, 1]^K$.

Figure 2.1: Example of retrieval problem in the text domain.

2.1.2 Existing learning-to-hash methods

The existing hashing methods can be organized into two categories: shallow hashing and deep hashing. Shallow hashing methods learn linear hash functions and rely on carefully constructed discriminative features which are extracted from any hand-crafted feature extraction techniques or any representation-learning algorithms. Examples of the hand-crafted features are SIFT [91] in the computer vision domain or TF-IDF [115] in the text domain. More powerful deep unsupervised/supervised-trained features, e.g., features extracted from the pre-trained VGG network [123] can be employed to improve the retrieval performance. Essentially, these methods achieve the best performance when the extracted features already capture high-level concepts that are suitable for the target hashing domain. Popular examples of these techniques include Locality Sensitive Hashing (LSH) and Spectral Hashing Iterative Quantization (ITQ) [50], among other techniques [79, 90, 120]. On the other hand, the deep hashing methods combine the feature representation learning phase and the hashing phase into an end-to-end model and have demonstrated significant performance improvements over the hand-crafted feature-based hashing approaches [15,80,140,159]. However, the success of end-to-end hashing models is not replicated in domains other than image applications. For example, while being equally important, there is no prior work in the text domain or advertising recommendation domain to simultaneously learn data representation and an efficient hash function by directly learning end-to-end from raw input data.

Along another dimension, the hashing methods can also be categorized into supervised and unsupervised hashing. Examples of the supervised hashing methods include [14, 45, 120, 140, 151] and examples of unsupervised hashing methods include [24,25,50,62,63,85,113,138,150]. While the supervised methods demonstrate superior performance over the unsupervised ones, supervised methods can easily overfit when there are limited labeled data. To overcome such

limitations of supervised hashing, data synthesis techniques have been successfully used to improve the retrieval performance [14, 105]. These methods employ generative models, such as the popular Generative Adversarial Network (GAN) models to synthesize the contrastive images. The motivation behind these techniques are two folds: 1) generative modeling assists in learning hash functions with better generalization by significantly increasing the amount of training data through their data ability, 2) the learned representation function of the generative models, such as GAN, can be used to extract powerful discriminative features, from which the hash function learns to project to the hash encoding space. The latter strategy is motivated by the assumption that generative models must capture high-level features of the data in order to generate realistic samples from the data distribution. While GAN-based approaches achieve impressive performance in the data generation tasks, such as image generation, they are yet to accomplish a similar success in representation learning [53]. Recently, some energy-based generative approaches [53, 92], which are based on maximum likelihood estimation, demonstrate a more complete achievement of generative models, including powerful representation learning, and missing-data and out-of-distribution data robustness.

2.1.3 Heuristic Hash Function Learning

Prevalent hashing approaches minimize the following training objective:

$$\min_{f} E_{x\sim D_x} L(x, \mathcal{H}(x)) + E_{x\sim D_x} \sum_{k} \lambda_i \times R_k(\mathcal{H}(x)) \tag{2.1}$$

where D_x is the data distribution, $L(x, \mathcal{H}(x))$ is the similarity-preserving loss of the hash function $\mathcal{H}(x)$ and $R_k(\mathcal{H}(x))$ is a regularization objective with λ_k as its corresponding hyperparameter. The role of R_k's is to ensure that the discrepancy between approximate continuous solution and the desired discrete solution is small. Examples of r_k are:

- *Bit balance* [138]: each bit has the same chance of being 0 or 1.
- *Bit uncorrelation* [138]: different bits are uncorrelated.
- *Low quantization error* [50]: low information loss when encoding the real-valued representations in the binary, discrete space.

In hashing, including R_k's is important for improving the retrieval performance. For example, bit-balance and bit-uncorrelation encourage "code-balance", i.e., the condition where a uniform number of training data points is assigned to each hash code. Code-balance minimizes the time complexity of the worst and average cases, thus, improving the retrieval efficiency [57]. Furthermore, code-balance and low-quantization error alleviate the chance

of assigning "very" similar data points to codes that have large Hamming distances. Hence, they better preserve the original locality structure of the data [50,57,69]. Preserving locality improves the retrieval precision. It is critical that these three objectives are jointly optimized in any efficient hashing algorithm. For example, without the bit-uncorrelation constraint, an algorithm can still achieve bit balance. A trivial example is assigning half the number of data points to a code with all 0-bits and the remaining half to a code with all 1-bits. However, the quality of the hash codes are apparently poor because there are only two possible codes regardless of the number of bits.

In spite of possessing better retrieval performance, existing deep text-hashing methods either ignore these objectives (for easier optimization) or construct them heuristically. We argue that the retrieval performance is significantly improved by incorporating them in learning the deep hash functions. Furthermore, we prove that it is possible to *implicitly learn the hashing objectives* in an *adversarial-training framework*. Thus, we can eliminate the need to tune several "hyperparameters" when explicitly defining the heuristic hashing objectives in a learning-to-hash model. In large-scale retrieval applications, such a decrease in training time has a significant value.

2.2 Generative Modeling

Generative Modeling refers to unsupervised machine learning methods which estimate the distribution of the data, or $p(x)$. While some generative models directly estimate the density function of the data distribution, others, such as Generative Adversarial Networks (GANs) [2, 52], provide an implicit way of sampling from such distribution. There are several benefits of studying, and eventually integrating generative models to machine learning applications. Data-sampling generative models, such as GANs, are important tools for manipulating high-dimensional objects, such as images, which can be used to improve the performance of other machine learning tasks. For example, interpolating in the input latent space of the generator of GANs can generate unseen samples, which can help the models in the downstream tasks generalize better. Furthermore, some generative models, especially the maximum-likelihood-based ones [53,92,141,144], can provide predictions on input with missing data, thus improving the smoothness of the downstream models. Besides data manipulation, generative models such as GANs can enable machine learning to work with multi-modal data. For example, in many machine learning tasks, for a given input, there can be several "correct" predictions; however, traditional non-generative machine learning models are not able to produce these many correct answers. Some other computational techniques in generative modeling such as divergence minimization can also have applications in several domains, which we will discuss in more detail next.

2.2.1 Generative Adversarial Networks

GAN has recently gained popularity due to its ability to generate realistic samples from the data distribution [52]. GANs have been successfully used in several applications and tasks, ranging from image synthesis [71, 108] and image-to-image translation [66, 161] to representation learning [32, 33] and regression [1, 98]. A prominent feature of GAN is its ability to "implicitly" *match* the output of a deep network to a pre-defined distribution using the adversarial training procedure. Furthermore, adversarial learning has been leveraged to regularize the latent space, as it helps in learning the intrinsic manifold of the data [30, 97]. For example, the adversarially trained autoencoders can learn a smooth manifold of the data in the low-dimensional latent space [97]. However, training the adversarial models remains challenging and inefficient because of the alternating-optimization procedure (minimax game) between the generator and the discriminator. Specifically, vanilla versions of GANs suffer from mode-collapse and vanishing gradient [2]. Moreover, in the minimax optimization, the generator's loss fluctuates during training instead of "descending", making it extremely challenging to know when to stop training the model.

Wasserstein-based adversarial methods overcome a few of these limitations (specifically, mode-collapse and vanishing gradient) [2,129]. However, because they employ the Kantorovich-Rubinstein dual, the minimax game still exists between the generator and the critic. On the other hand, the work in [65] directly estimates the Wasserstein distance by solving the Optimal Transport (OT) problem. Solving the OT has two main challenges. Firstly, its computational cost is $O(N^{2.5}log(Nd))$ where N is the number of data points and d is the dimension of the data points. This is expensive. Secondly, the OT-estimate of the Wasserstein distance requires an exponential number of samples to generalize (or to achieve a good estimate of the distance) [3]. In practice, both the high computational cost and exponential sample requirement make the OT-based adversarial methods very inefficient.

2.2.2 Energy-based Generative Models

The work in [142] defines an explicit probability distribution on the data space in the form of energy-based models (EBMs). The generative ConvNet parameterizes the energy function by a bottom-up convolutional neural network and trains the network by MCMC-based maximum likelihood estimation. Recent research papers have demonstrated the power of this model in image generation [42,103,142] and image feature learning [42]. This model has also been successfully generalized to synthesizing videos [146], 3D volumetric shapes [145], 3D point clouds [143], trajectories [147], etc.

Xie et al. [141] propose a powerful generative model, called generative cooperative network (CoopNets), which can generate realistic image and video patterns. The CoopNets framework jointly trains an energy-based model (i.e., descriptor network) and a latent variable model (i.e., generator network) via a cooperative learning scheme, where the descriptor net-

work is trained by MCMC-based maximum likelihood estimation [142], while the generator learns from the descriptor and serves as a fast initializer of the MCMC of the descriptor. A conditional version of CoopNets [144] has also been proposed for image-to-image translation. Most of the above works focus on leveraging CoopNets for data synthesis. Furthermore, in practice, CoopNets employ two separate networks which must be carefully designed together to ensure the model converge into a good local minimum [141]. This problem also exists in GANs.

Recently, the works in [53, 92] propose a scalable single-network EBM for the image generation and representation learning tasks. The EBM is able to generate a realistic image and exhibits attractive properties of EBM such as out-of-distribution and adversarial robustness. To estimate the intractable partition function, the authors use MCMC sampling through the Langevin dynamics and build upon the persistent contrastive divergence (PCD) [127] to propagate the MCMC chains during training.

One advantage of generative single-network EBMs over GANs is their easier network design and training stability. In these models, only one single network is required (instead of two, namely, the generator and the discriminator), and this network must be perfectly "matched" for training stability. For example, if the discriminator is more powerful than the generator, the discriminator can easily overfit the training data, which makes it more difficult for the generator to learn to generate the data. While the relationship between the discriminator and generator in CoopNets is "cooperation", instead of being each other's "adversary", this problem still exists. Generative EBMs, in general, and single-network generative EBMs, in specific, provide other important advantages over GANs such as better representation learning. While the GAN's discriminator estimates the divergence between the real and fake data distributions, the EBM-based discriminator directly estimates the likelihood of the real data itself. In other words, EBM-based discriminator is not influenced by the unnecessary information from the fake data and mainly focuses on understanding the real data. This allows the EBM-based discriminator capture highly relevant high-level abstraction of the data itself. Furthermore, this learned representation is shown to have some attractiveness, such as out-of-distribution property.

2.3 Evaluation Metrics

Selecting the right evaluation methods is the key part of studying retrieval models. Evaluation involves two criteria: effectiveness and efficiency. Effectiveness refers to the quality of the retrieval items, usually in the form of a ranked list, returned by a retrieval method. Efficiency is related to the economical use of resources such as computation and space. In this section, we introduce the commonly-used evaluation metrics to assess the effectiveness of a retrieval method and discuss the efficiency metrics in the later chapters for efficiency metrics are usually context-dependent.

The quality criteria to evaluate the effectiveness of the retrieval models can be divided into two categories: truncated metrics and full-ranking metrics. Truncated metrics are calculated based on a fixed number, say 1000, that is retrieved from the database. Since hashing focuses on quick, and often in real-time, retrieval of a small number of highly relevant items, most hashing works employ the truncated metrics. Widely used truncated metrics include precision (**P@K**) or recall (**R@K**), and mean average precision (**MAP@K**). Given the query images, P@K, R@K, and MAP@K are calculated as follows:

$$\text{P@K}(q) = \frac{\sum_{k=1}^{K} \delta(k,q)}{K} \tag{2.2}$$

$$\text{P@}K = \frac{1}{Q} \sum_{q=1}^{Q} \text{P@K}(q) \tag{2.3}$$

$$\text{AP@K}(q) = \frac{1}{N_q} \sum_{k=1}^{K} \text{P@k}(q) \times \delta(k,q) \tag{2.4}$$

$$\text{MAP@K} = \frac{1}{Q} \sum_{q=1}^{Q} \text{AP@K}(q), \tag{2.5}$$

where K is the number of retrieved images, N_q is the number of all relevant images in this set, Q is the size of the query set and $\delta(k,q) = 1$ only when the k-th retrieved image is relevant to the query image q; otherwise $\delta(k,q) = 0$. A retrieved image is relevant if its ground-truth label is the same as the label of the query image. Note that, since P@K and R@K are monotonically related (w.r.t. K), reporting either one of these two metrics is sufficient.

Different from truncated metrics, full-ranking metrics compare the computed ranked list by the retrieval method and the ground-truth ranked list of all items in the database. Examples of these metrics include Spearman's Rank Correlation Coefficient (ρ) [126] and Kendall's Rank Correlation Coefficient (τ) [72].

Chapter 3

Parameter-free Training of Hash Functions via Adversarial Autoencoders for User Retrieval

Approximate similarity search approaches, especially the hashing ones, have applications in several domains. An important domain among these is computational advertising. In this domain, a learning-to-hash method must possess two properties: (1) efficient computation in both training and inference stages, and (2) being able to capture complex relationship, especially the high-order interaction, between features of the input. In this chapter, we begin the development of generative hashing approaches with a model which captures the complex interaction of high-dimensional, sparse input features and preserves the locality information of the data points in the learned hash codes. This model is, then, used to solve an important problem in the advertising domain, called Look-alike Modeling.

3.1 Introduction

To maximize an advertising effort, advertisers execute their campaigns in multiple ways, i.e., digital advertising, search-based advertising, and contextual advertising. The primary aim of the advertisers across these avenues is to maximize the return on their investment. To realize this, the advertiser often aims to *target the advertisements* towards a *targeted set of audiences* as this set has a high likelihood to respond positively towards the advertisements. One such form of tailored and personalized targeted advertising is known as Look-alike Modeling.

Specifically, look-alike modeling, also known as "audience expansion" in the literature, is the task of finding new and relevant users given an existing, often much smaller set (known as seed-set) of users under the assumption that the newly identified look-alike users are similar to the seed-set's users with regards to the activity over the internet (e.g., click, sale or

browsing). Computational look-alike modeling is a fundamental and challenging problem in the advertising domain for which various techniques have been proposed [88,89,93,107,121]. The nature of the advertising data, coupled with constraints induced when modeling look-alike approaches have been a major challenge in the development of robust and sophisticated techniques. In particular, digital advertising data is a collection of information across sources (e.g., user interest, product information, engagement information), thereby making advertising data highly complex and multi-dimensional in nature. In computational advertising, users are often represented by hundreds to thousands, if not more, of categorical and continuous raw features. This often leads to the issue of high-dimensionality (millions of features) when interactions between the raw features or categorization of the raw features are taken into account. Another challenge that arises in modeling advertising data revolves around sparsity. This arises as advertisers collect users' data only when the users are displayed with some kind of product-related advertisements, thus the collected data results in a significant amount of examples where the values are absent (or missing features). For example, a user is not shown advertisements for all the products, but is typically only shown advertisements of products for which the user might have a propensity for a favourable outcome.

Additionally, it is critical for computational look-alike techniques to be efficient, especially in programmatic advertising platforms where there can be hundreds (or thousands) of campaigns that are executed daily. Each campaign requires a different set of look-alike users from the campaign's different, initial seed-set, which potentially involves the creation of many look-alike models on high-dimensional advertising data. For such reasons, despite the improved performance, *regression-based look-alike methods* [107], which separately model the membership probability of a user to the seed-set of a campaign using classification-based methods, are often too computationally expensive to be employed. This is because for each campaign, a classification model is created on a carefully constructed labeled dataset whose positive examples are users from the seed-set. When there are many campaigns, this process is prohibitive, especially when the data is highly sparse and high-dimensional. In addition, when sufficient amount of training data is not available on a newly created campaign, regression-based methods do not perform well; this issue is also known as the "cold-start" problem.

Similarity-based look-alike methods [88,93,94], which involve modeling the pair-wise similarity functions between a pair of users, are preferred because they can be scaled out to millions of users and they do not require creating different models for each campaign. Similarity-based methods include the current state-of-the-art research methods employed Locality Sensitive Hashing (LSH) [88,93,94]. However, LSH-based techniques are known to ignore some of the vital properties of the data (for e.g., they pay little attention to the distribution of the data during the modeling process) and have been shown to be less accurate and slower than data-dependent hashing schemes such as Semantic Hashing [113]. Replacing LSH-based techniques with semantic hashing based techniques is not straightforward, as Semantic Hashing does not work well on sparse, high-dimensional data and typically produces much inferior results in the presence of complex feature interactions. In addition, to motivate the hidden sigmoid-

neuron activations more often near 0 and 1, the learning process relies on employing training tricks, such as adding a deterministic Gaussian noise, with additional hyper-parameters that increase the complexity of using semantic hashing in a real-world environment. To address the aforementioned challenges, in this chapter, we propose a novel Adversarial Factorization Binary Autoencoder that can efficiently learn a mapping from sparse, high-dimensional data to a binary address space through the use of an adversarial training procedure. Our model, which is a "hashing" based approach [134], learns compact, storage-efficient binary codes. The cost of finding look-alike users in the original high-dimensional space is reduced to Hamming-distance calculations (using bit-wise XOR operation which takes only one CPU instruction) in the binary address space (in which a data point's representation requires only 4 to 8 bytes of storage). The main contributions of our work are as follows:

- We propose a novel autoencoder based look-alike model that employs adversarial training procedure to systematically "encourage" the hidden-code layer to be binary. Our method introduces an entirely new and efficient alternative to train binary neurons, which can be employed in other problems that require conditional computations [6].
- We extend the semantic hashing method to be able to efficiently handle extremely-sparse data by modeling an additional feature embedding layer and embedding interaction layer that can automatically learn higher-order feature interactions in an *end-to-end*, completely *unsupervised* manner.
- We demonstrate the effectiveness of our proposed look-alike method on a large real-world advertising data and show both quantitative and qualitative results of the performance.
- The strong offline performance of our proposed model across different partners demonstrate the practical applicability of our approach and thus will motivate advertisers to opt for a targeted advertisement (i.e., look-alike) as compared to a generic re-targeting based advertising.

This chapter is organized is as follows. We discuss the related works in Section 3.2. In Section 3.3, we present information about the data generation process, feature engineering along with some basic data statistics. In Section 3.4, we describe the background, problem formulation and our proposed approach. In Section 3.5, we describe our experimental procedure, baseline methods and results. Lastly, in Section 3.6, we present our conclusion and future work.

3.2 Related Works

In this section, we will describe three research directions that are closely related to the proposed work, namely, look-alike modeling, autoencoders and high-dimensional sparsity.

3.2.1 Look-alike Modeling

Look-alike modeling is the task of reaching new and relevant users in an advertising campaign by extending an existing, smaller set of users (also known as the seed-set). Typically, this extension process can be accomplished by finding "similar" users to those within the seed-set [89, 93, 94, 107, 121]. These approaches can be classified broadly into two categories:

1. *Regression-based look-alike models*: The simplest approach is to predict the class-membership of a user belonging to the seed-set. Given a user with a set of features x, these models compute the propensity score $p(x \in S|x)$ – the probability of the user belonging to the seed-set S – which can be modeled using the sigmoid function $\frac{1}{1+\exp(-\theta^T x)}$ [107]. **Density estimation** approaches can be used to directly estimate $p(x \in S|x)$, but this is generally very difficult both because of the inherent challenges of density estimation and a higher chance of over-fitting when the size of the seed-sets is small. A more popular regression-based look-alike modeling approach is **classification** where negative samples (of the negative class in training the propensity model) can be randomly sampled from all the available users (minus the seed-set). Random negative samples have the following effect: features that correlate with the advertising conversions (such as click-through rate or product purchases) play less important role in similar-user selection. This motivates the sampling of previously exposed (having seen the campaign's advertisements) but non-converted users. However, doing this will lead to the cold-start problem; for example, on new advertising campaigns where there have not been sufficient number of previously exposed users. In spite of this problem, there are a couple of primary advantages of using regression based models. First, using such approaches, the decision to select the extended users is compressed within a model. Second, the performance of regression-based techniques is often significantly better when there is enough, well-constructed supervised training data.

2. *Similarity-based look-alike models*: Instead of learning a class-membership model, using a similarity measures such as Cosine or Jaccard [82], similarity-based systems find look-alike users by directly comparing all possible pairs between seed users and available users using a "pre-defined" pairwise similarity function [88, 93, 94]. A seed-set can be extended by selecting top users from a ranked list of users which is ordered based on their maximum similarity scores to any user in the seed-set. These systems can be scaled out to millions of users using **hashing methods** [134] such as Locality Sensitive Hashing (LSH) [82], but the performance of such systems depend greatly on selecting the right set of features

and similarity functions. Often, they are chosen independent of the data (for e.g., using random projection) and hence, important intrinsic properties of the data are completely ignored.

A key advantage of similarity-based approaches is the fact that its learning process is independent of the number of seed-sets. An implementation of regression-based models requires learning one model for each individual seed-set, which significantly increases the training time when there are thousands of such models to be trained on a regular basis (for e.g., daily), which is a common scenario for many advertisers. Furthermore, extending the seed-sets require a full linear-scan of the users since the model needs to calculate the propensity score for each user in order to rank them. Conversely, similarity-based approaches focus on learning only one desired similarity function, irrespective of the number of seed-sets. In addition, because of the advantage of hashing, these approaches can also limit the retrieval to a small list of candidate users [82]. However, because of the performance superiority of regression-based models, most of the look-alike modeling systems that are in production either employ regression-based models or a hybrid version of regression-based and hashing models (to reduce its learning/training time while preserving the overall performance). In this chapter, we focus on a similarity-based approach (specifically hashing) that automatically learns an efficient similarity function in an end-to-end manner directly from the raw data and demonstrates significant performance improvements compared to both existing regression-based and similarity-based approaches.

3.2.2 Autoencoders and Adversarial Training

Autoencoders

Most of the widely used similarity-based look-alike models employ variants of LSH because such models do not require pre-training and are computationally efficient to deploy in a real production environment [88, 93, 94]. Despite their efficiency, most LSH approaches are data-independent because their hash functions employ random projection, which makes them inappropriate for complex scenarios, especially for high-dimensional sparse data.

Deep neural networks have the ability to discover good representations from the raw input data [81]. The authors of [113] proposed an alternative to the LSH-based approach for approximate similarity search with a pre-defined similarity function by learning a binary vector representation of the input data using autoencoders [81]. The binary vector captures the semantic dependency of the input data in a lower-dimensional space. The advantage of semantic hashing is that, when more hidden layers are used, the network can capture complex interactions among the input features. In addition, finding similar items, after the model is trained, is experimentally shown to be faster while achieving better accuracy compared to LSH-based approach for document retrieval tasks in the text domain [113].

Generative Adversarial Network (GAN)

GAN models have recently gained popularity due to their generative power [52]. One important feature of GAN is the adversarial training procedure which is able to implicitly "match" an output of a deep network to a pre-defined distribution. In fact, the adversarial training mechanism has been used in several works as a regularization of the latent representation of autoencoders [97, 158]. This improves the quality of the projected, low-dimensional manifolds that the autoencoders are able to learn, especially when the high-dimensional input data has a complex structure [158].

Regularizing the autoencoders with an adversarial procedure allows our model to learn an effective representation of the complex, high-dimensional sparse input data while ensuring the latent representation to implicitly match with a pre-defined binary-like distribution. The latter contribution encourages the bottleneck layer of the autoencoder to generate efficient hash codes (similar to that of LSH) but stays closer to the manifold of the original input data.

3.2.3 High-dimensional Sparsity

Advertising data is high-dimensional in nature with many missing feature values and hence, most of the standard machine learning models are prone to the problem of over-fitting. Traditionally, manual feature engineering is used to improve machine learning models' performance but such solutions fail to scale well for higher order feature interactions where the number of features grows exponentially (consequently making the features even sparser). Feature embedding techniques, including both factorization and neural-network models, solve this problem by learning feature interactions directly from raw data [60, 110]. For example, Factorization Machines (FM) [110] model interactions between each pair of features as a dot product between two low-dimensional embedding vectors of the features.

However, to the best of our knowledge, this idea has not been applied in the unsupervised domain, specifically in autoencoders which learn low-dimensional representation of high-dimensional sparse input with complex feature interaction. In the presence of sparse data, the autoencoder's reconstruction output loses its sparsity and easily collapses to the average of the input features [51]. Simply designing autoencoders to be wider and deeper does not help because with too much capacity, autoencoders can learn to "copy" the input without finding a good latent, low-dimensional representation of the data. We propose to tackle the high-dimensionality and sparsity challenges by fusing factorized models with autoencoders, regularized by an adversarial learning procedure in order to find an efficient latent representation of the data.

3.3 Criteo Data

3.3.1 Data Generation Process

Criteo Look-alike Modeling (CLM) is a recent product from Criteo that was launched at the start of 2018 on selected markets and partners. In CLM, partners (i.e., advertisers selling products) provide a selected set of users (i.e., seed-set) and then the goal of CLM is to identify a set of look-alike users (i.e., users who are not in the seed-set, but are similar to the seed-set of users with respect to their online behaviour) from the pool of available users. After the identification of look-alike users, Criteo uses its re-targeting technology to display partner-related advertisements to the look-alike users across publisher websites.

CLM combines different types of data to train machine learning models for identifying look-alike users. CLM collects data about users interacting with product related advertisements primarily from advertisers in an anonymous setting. Also, it collects additional detailed data related to products' baskets, products' sale transactions and clicks of the relevant product advertisements. Figure 3.1 represents the entire end-to-end pipeline of the large-scale data collection process, model training and advertisement display methodology for CLM.

3.3.2 Feature Engineering

Each user is represented by an aggregated set of online activities that are performed across publishers and products. In this chapter, we only consider three major action types for the user (i.e., click on the product advertisement, addition of product to the basket and sale of the product). Furthermore, to overcome the high-dimensionality associated with Billions of products that are available in Criteo's catalog, we use the hierarchical taxonomy from Google[1] to map multiple products to a single relevant category. Hence, Billions of products in Criteo's catalog are mapped to approximately thousands of product categories. In summary, every user is represented using categorical representation across the kind of interaction (i.e., click, basket or sale) and product category. For the sake of replicability of our experiments and results, we have already released the data that is used in this work[2].

[1]https://www.google.com/basepages/producttype/taxonomy.en-US.txt
[2]https://s3-us-west-1.amazonaws.com/criteo-lookalike-dataset/data.zip

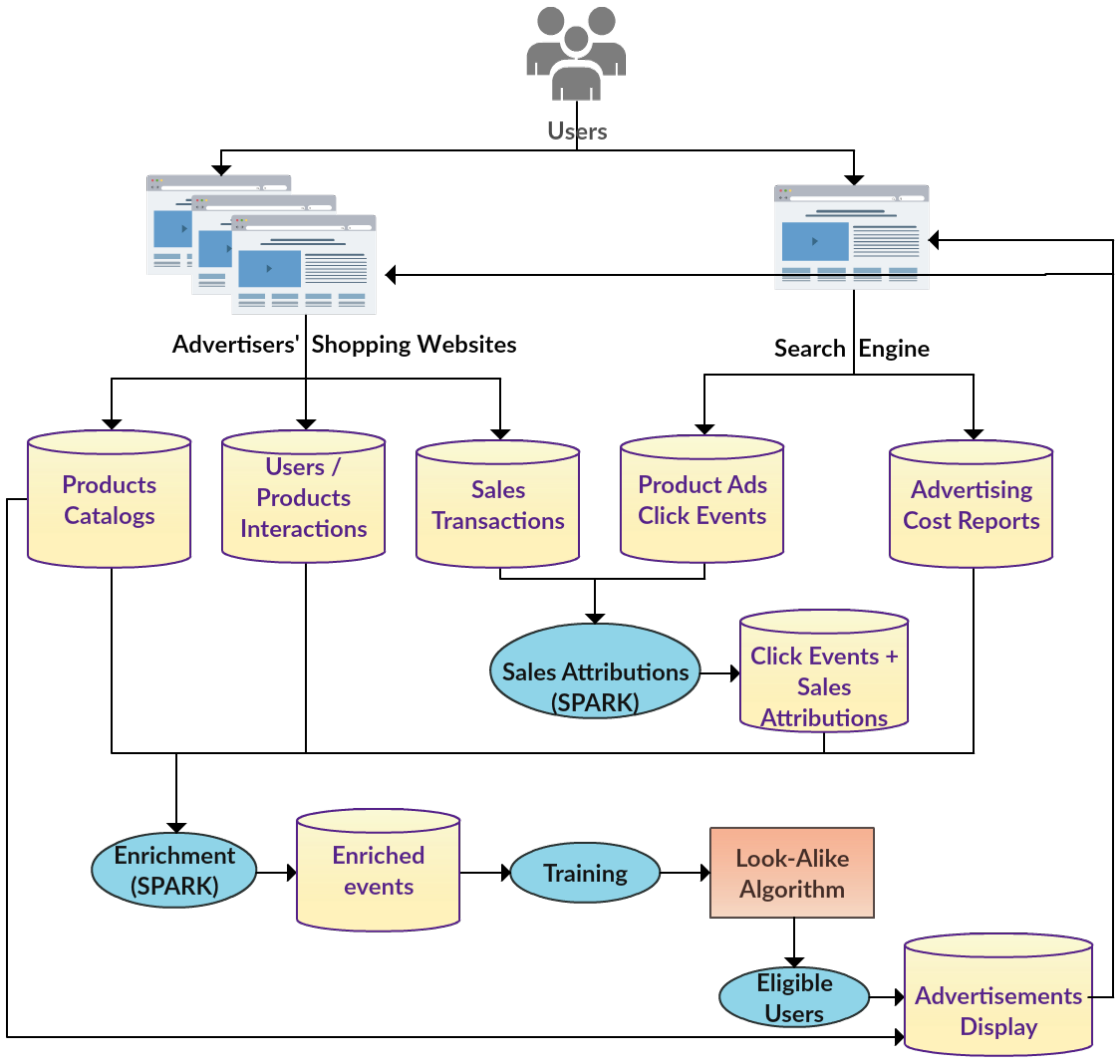


Figure 3.1: Data collection and model training pipeline for Criteo's look-alike modeling product.

3.3.3 Data Statistics

Table 3.1: Statistics of the Criteo partner's datasets.

Partner	Consumer Segment	Seeds	Non-seeds
Apparel-1	*Apparel & Accessories*	18,898	3,038,404
Electronics-1	*Electronics*	398	3,030,559
Home/Garden	*Home & Garden*	34,523	3,020,830
Electronics-2	*Electronics*	114	3,047,652
Animal/Pet	*Animals & Pet-supplies*	3,544	3,047,785
Apparel-2	*Apparel & Accessories*	75,646	3,009,994

The data setting for CLM is as follows: each partner provides a seed-set consisting of approximately a few thousand users. CLM, then, uses its robust ML technology to identify look-alike users (roughly 10 times) who are similar to that of the seed-set users from a pool of Billions of users. However, considering the fact that a partner is often interested in a specific geography (i.e., Europe, North America or Asia-Pacific), the pool of users often is limited to hundreds of millions of potential users (instead of Billions of users).

Among the hundreds of millions of potential users, only a few millions will have a significant online activity. Now we will present some data statistics, provided in Table 3.1. The data contains information about six partners, i.e., the size of the seed-set provided by the partner and the size of the non-seed users available in the data after multiple rounds of filtering. The partners are selected from broad consumer segments, thus bringing significant diversity to the partners' datasets and the numbers of users in the seed-sets.

3.4 The Proposed Look-alike Model

In this section, we will first define the problem statement. We will then describe our proposed framework.

3.4.1 Problem Formulation

Given a seed-set of N users $X_o = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$, the goal of look-alike modeling is to extend X_o with M new users X_e such that the extended users X_e are similar to X_o with respect to their online internet shopping behaviour. Each user $x^{(i)}$ is represented by a sparse vector in $\mathbb{R}^n$. We will now discuss various components of our proposed framework for the look-alike modeling problem. The notations used in this chapter are given in Table 3.2.

Table 3.2: Notations used in chapter.

Notation	Description
$x, \tilde{x}, \hat{x}$	input vector, bi-interaction embedding vector, and reconstructed input vector, respectively
x_i	i^{th} input feature
$x^{(i)}$	i^{th} input sample
e_i	embedding vector of feature i
D_x	data distribution
$I(.)$	bi-interaction function whose output is $\tilde{x}$
f, g	encoding and decoding functions
b	low-dimensional, binary vector – output of $f(.)$
$q(b)$	the posterior distribution of b
L_{BA}, L_{FA}	reconstruction cost of the binary autoencoder and factorization binary autoencoder, respectively
D	the discriminator network
L_D, L_G	original loss function for the generator and discriminator, respectively
Θ_G	parameters of the generator, including parameters of both functions $i(.)$ and $f(.)$
Θ_D	parameters of the discriminator $D(.)$

3.4.2 Binary Autoencoder

Given a dataset $X = \{x|x \sim D_x\}$, a binary autoencoder (BA) defines an encoding function $f : x \to b$ that maps each data point $x \in \mathbb{R}^n$ into a point $b \in \mathbb{R}^B$ in the coding space, where B denotes the dimension of the coding space, and a decoding function $g : b \to \hat{x}$ that recovers x, as $\hat{x}$, from b. In semantic hashing [113], the RBM-based autoencoder uses a sigmoid encoding layer – or bottleneck layer – and *motivates* each component of b, activations of the bottleneck layer, to take values closer to either 0 or 1. As a result, thresholding the activations b will return a binary code vector $c \in \{0, 1\}^B$ which can be considered as the discrete hash code of the input x.

For high-dimensional advertising data, it is important to exploit the higher-order interaction for better model performance, as discussed in Section 3.2. In spite of some drawbacks (which will be discussed later in Section 3.4.3), deep multi-layer encoders can model feature interaction and find a useful discrete, low-dimensional representation of the input. Thus, the binary autoencoder has the following structure:

- *An encoder function* $f : x \to b$ that maps x into a vector b. In our work, we model f as a multi-layer perceptron (MLP) whose last layer has the sigmoid activation function.

- *A decoder function* $g : b \rightarrow \hat{x}$ that reconstructs the input x as $\hat{x}$. Similar to the encoder, we model the decoder with an MLP.
- Similar to Semantic Hashing [113], we employ fixed, random normal noise to force b's values to be binary (closer to 0 or 1).
- BA learns its parameters by minimizing the following squared-error reconstruction cost:

$$L_{BA} = ||\hat{x} - x||_2^2 = ||g(f(x)) - x||_2^2 \tag{3.1}$$

3.4.3 Factorization Binary Autoencoder

Autoencoders are popularly used in learning non-linear data embedding, or data representation. Unfortunately, when the input is high-dimensional and highly sparse, it becomes difficult for the autoencoders to learn a useful representation [51]. Wider and deeper encoders/decoders result in too much model capacity, thus the autoencoders can easily learn to **copy** the input, without extracting any useful information about the data distribution in the low-dimensional latent space [51]. In other words, it is difficult for the binary autoencoders to utilize non-linear, higher-order interactions of the input features to learn a good hash function of the input. Therefore, we propose to model the binary autoencoder with an explicit feature interaction layer. Inspired by the supervised NFM [60], given an embedding vector $e_i \in \mathbb{R}^e$, where e is the dimension of the embedded space, for each feature i, we model the interaction of features using a bi-interaction function $I(x)$ as follow:

$$I(x) = \tilde{x} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i e_i \odot x_j e_j \tag{3.2}$$

where $\odot$ is the element-wise multiplication operation of two vectors and the last term can be efficiently calculated using only non-zero feature positions [60]:

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i e_i \odot x_j e_j = \frac{1}{2} \left[(\sum_{i=1}^{n} x_i e_i)^2 - \sum_{i=1}^{n} (x_i e_i)^2 \right] \tag{3.3}$$

Equation (3.2) captures higher-order feature interactions in the low-dimensional, discrete representation space because the first term achieves memorization, correlation of the input features, while the second term achieves generalization, transitivity of the features' correlation [18]. In this work, we model second-order interaction of the input x in the bi-interaction embedding layer, but it is possible to model the higher-order interaction by extending the embedding vectors e_i into higher-dimensional tensors. It is important to note that computation of the high-order, non-linear interaction is linear in the number of non-zero raw input features, which makes it very efficient for high-dimensional, sparse input domains such as computational advertising.

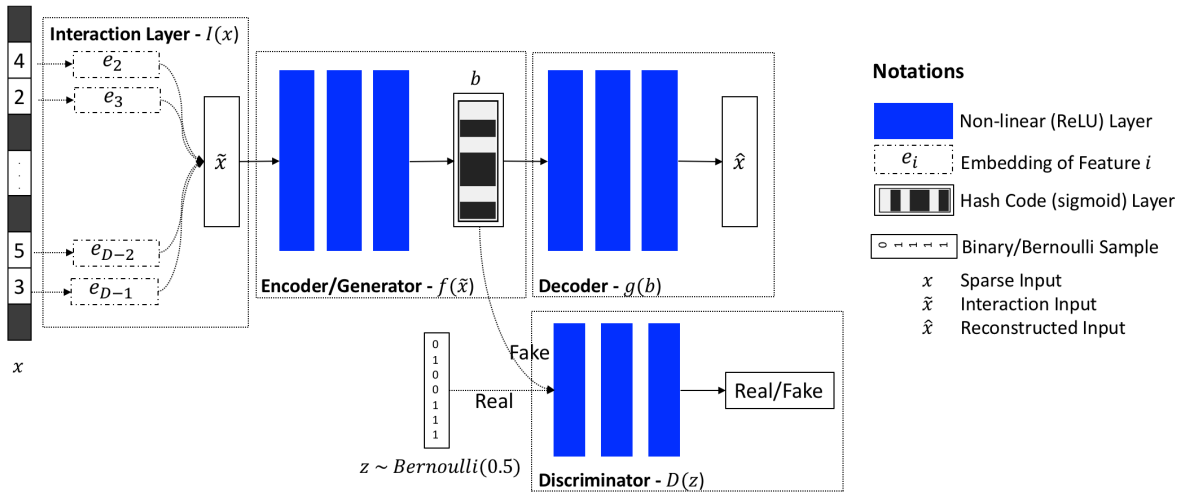


Figure 3.2: Network architecture of the Adversarial Factorization Binary Autoencoder (AFA) model.

In our proposed factorization binary autoencoder (FA) model, the encoder function becomes $f : \tilde{x} \rightarrow b$. On the other hand, instead of employing the similar decoder function $g : b \rightarrow \hat{x}$ as in Section 3.4.2, our decoder, instead, learns to reconstruct the bi-interaction embedding input $\tilde{x}$ by minimizing similar squared-error, reconstruction cost as follows:

$$L_{FA} = ||\hat{x} - \tilde{x}||_2^2 = ||g(f(I(x))) - I(x)||_2^2 \tag{3.4}$$

Reconstructing $\tilde{x}$ instead of x overcomes the difficulties of autoencoders to reconstruct high-dimensional, sparse input. Different from NFM, FA learns to simultaneously find a good interaction representation and a good low-dimensional, discrete representation of the input x because the gradient of the loss function propagates to the embedding space both immediately at the decoder's output and through the autoencoder's network. More importantly, the learning process is performed in a completely unsupervised manner.

3.4.4 Adversarial Factorization Binary Autoencoder

From an alternative perspective, an autoencoder defines an encoding distribution $q(b|x)$ and a decoding distribution $p(x|b)$. Consequently, the encoding step imposes a posterior distribution on the coding space as follows:

$$q(b) = \int_x q(b|x) D_x(x) dx \tag{3.5}$$

Our proposed FA model has several network parameters, from both the bi-interaction layer and autoencoder module, that need to be learned. It is previously shown that when the

encoder and decoder are too powerful, the autoencoder can get stuck in bad local-minima and perform "copying" the input without learning any useful representation from the data [10, 158]. In order words, the distribution $p(x|b)$ easily collapses to $D_x(x)$, and the encoder maps data points in D_x to discrete codes **scattered** in the latent space, thus the locality information in the original input space are not well preserved in the latent, discrete space [158]. To overcome such limitations, we propose to regularize the discrete, hash layer with a discriminator through an adversarial learning procedure [158]. We also take advantage of the **distribution matching** property of the adversarial network to force the low-dimensional, discrete space $q(b)$ to **match** a binary-like distribution such as a "discrete Bernoulli" prior. If we enforce $q(b|x)$ to "agree" with a distribution that has all desired characteristics of a good hashing function, such as "bit balance" and "bit uncorrelation" [134], the adversarial training procedure will force the encoder function $f(x)$ to generate binary codes with the same desired characteristics. Specifically, given an input vector $z \sim Bernoulli(p)$, defined as a vector where each of its components z_i is independently drawn from the same underlying Univariate Bernoulli distribution with parameter p, we regularize the posterior distribution $q(b)$ to be similar to $Bernoulli_z(p)$. The adversarial binary autoencoder (ABA) is trained with dual objectives as follows:

1. Minimize the reconstruction error from the coding space in the decoding step.
2. Constrain the coding posterior distribution, $q(b)$ to agree with the Bernoulli prior, $Bernoulli_z(p)$, where p is set to 0.5. An intuitive explanation of the regularization is that every dimension of b will learn to divide as optimal as possible the original space into two halves (thus its activation has about 50% chance of being closer to 1 or closer to 0 – a "bit balance" [134]), where the points in each half are more similar to those in the same half compared to those in the other half – "bit uncorrelation" [134].

In the adversarial setting, the autoencoder's encoding module takes the role of a generator which is trained to fool the discriminator. The discriminator tries to accurately divide its input into two classes, real and fake for the Bernoulli samples z and the autoencoder's encoding outputs b, respectively. The solution of this adversarial procedure (or the minimax game) is obtained by solving the following objective function:

$$\min_{\Theta_E} \max_{\Theta_D} \; E_{z \sim Bernoulli_z(p)}[\log D(z)] + \; E_{x \sim D_x}[\log(1 - D(f(I(x))))] \tag{3.6}$$

where Θ_G and Θ_D are network parameters of the generator and discriminator, respectively. For an optimal discriminator with parameters Θ_D^*, it can be proven that the training criterion of the generator is as follows:

$$C(G) = -log(4) + 2 * \text{JSD}(Bernoulli_z(p) \;||\; q(b)) \tag{3.7}$$

where $\text{JSD}(p \;||\; q)$ is the Jensen-Shannon Divergence between distributions p and q. The global minimum of $C(G)$ is achieved when the non-negative, symmetric divergence

$$\text{JSD}(Bernoulli_z(p) \;||\; q(b)) = 0 \tag{3.8}$$

and its solution is

$$q(b) = Bernoulli(p) \tag{3.9}$$

In other words, the encoder network perfectly replicates the Bernoulli-data generating process. Since the Bernoulli-data generating process samples each component z_i of z independently (hence, each bit is uncorrelated) from a Univariate Bernoulli distribution with probability 0.5 (hence, each bit has 0.5 probability of being 1), matching $q(b)$ against the $Bernoulli_z(p)$ is exactly equivalent to generating bit-uncorrelated and bit-balance codes. Consequently, our proposed regularized autoencoder learns an effective and optimal hash function because it satisfies the following key characteristics:

- Achieve *low quantization loss* by generating samples that are similar to the sampled, binary vectors z. Note that, using adversarial training, our model minimizes JSD($Bernoulli_z(p)$ || $q(b)$) instead of minimizing the expected quantization error, $E_{D_x}||b - c||^2$ [50].
- Achieve *bit uncorrelation* and *bit balance* by forcing the encoder to replicate the bit-uncorrelated and bit-independence data generation process.
- *Preserves the data manifold* in the low-dimensional discrete space, i.e., the encoder learns to map the original data points to an effective manifold-preserving low-dimensional space in order to have low-reconstruction cost.

We train the network using alternating stochastic gradient descent (SGD) procedure in three stages:

- Train the autoencoder to reconstruct accurate samples from the encoding space b by minimizing L_{FA}
- Train the generator's encoder to confuse the discriminator by minimizing the familiar heuristic generator cost:

$$L_{\mathrm{G}} = E_{x \sim D_x}[-\log(D(f(I(x))))] \tag{3.10}$$

- Train the discriminator to distinguish the true (bernoulli) samples from the fake (autoencoder) ones by minimizing:

$$L_{\mathrm{D}} = E_{z \sim Bernoulli_z(p)}[-\log(D(z))] + E_{x \sim D_x}[-\log(1 - D(f(I(x))))] \tag{3.11}$$

The detail of the training algorithm is described in Algorithm 1.

Algorithm 1 AFA Model Training

Input: Training data X,
Feature embedding size e,
Binary code size B,
Bernoulli sampling parameter p,
Batch size N_b,
Number of training iterations L, l.

Output: W all parameters of the network

for *number of training iterations* L **do**

  for *a few iterations* l **do**

    - Sample a minibatch of N_b examples $\{x^{(1)}, ..., x^{(N_b)}\}$.
    - Sample m vectors $\{z^{(1)}, ..., z^{(N_b)}\}$ where $z^{(i)} \sim Bernoulli_z(p)$.
    - Update the autoencoder parameters by minimizing L_{FA} in Equation (3.4).
    - Update Θ_E by minimizing L_G using Equation (3.10).

  end

  - Sample a minibatch of N_b examples $\{x^{(1)}, ..., x^{(N_b)}\}$.
  - Sample m vectors $\{z^{(1)}, ..., z^{(N_b)}\}$ where $z^{(i)} \sim Bernoulli_z(p)$.
  - Update Θ_D by minimizing L_D in Equation (3.11).

end

3.5 Experimental Analysis

3.5.1 Evaluation Procedure

Our hypothesis is that the carefully constructed segments (i.e., seed-sets), obtained from advertisers, essentially define the advertising targets; for example, instead of executing a campaign with an explicit target such as "reaching users who are movie-goers", the advertiser instead provides a seed-set of users and the look-alike modeling system finds "similar" new users to reach during the campaign. However, it should be noted that advertisers rarely provide the rationale behind the creation of the seed-set. In such a scenario, the only way to assess the off-line performance of look-alike modeling algorithm is to analyze the recall of the seed-set from the test data. A high value of recall indicates that the method is able to discover the rationale behind the creation of a seed-set by the advertiser.

Now, we will describe our evaluation procedure to analyze the performance of the look-alike modeling technique. The estimation of the off-line "Segment Recovery" task is similar to the work of [121]. Given a random subset of a segment A, the goal is to find similar users in order to recover A. In particular, we perform the following procedure:

1. Approximately sample $|A|/2$ number of users from the segment A, i.e., randomly divide set A into two equal subsets denoted by A_1 and A_2. It should be noted that, in this case, the seed-set denoted by X_o will be equal to the set A_1.

2. Given A_1, we assess how many users of A will a method recover at various extension thresholds m, where m is the multiple of the number of users in A_2. We define the following metrics for evaluation:

$$\text{recall}_{@m} = \frac{|X_e \cap A_2|}{|A_2|} \tag{3.12}$$

$$\text{precision}_{@m} = \frac{|X_e \cap A_2|}{|X_e|} \tag{3.13}$$

In addition, we calculate the limited area under the Precision-Recall curves (equivalent to limited mean average precision or mAP) of different methods [117]. For regression-based look-alike models, we rank the users using the probability score of a user being a seed user and select the top $|X_e|$, or $m * |A_2|$ ranked users to compute the precision and recall values. For similarity based look-alike methods, such as LSH or our proposed models, we rank the users by their minimum Hamming distances to seed-users and select the top $|X_e|$ users. When there are ties, we randomly select users in the tied ranks.

The reported recall metric indicates how well each method will recover the original segment X_o at various, specific values of m, while mAP indicates the expected precision under all possible m's values. In look-alike modeling, a method with a higher recall at the first few values of m ("early-rise"), for example, at $m = 1$, is typically more favorable. In order to efficiently compute the evaluation metrics on a single machine, we randomly select 20% of the non-seed users, instead of using the entire set of non-seed users. The reported metrics are averaged on 5 samples.

3.5.2 Comparison Methods

We compare the performance of our proposed method with various state-of-the-art methods developed in the literature for solving look-alike modeling problems.

- *Locality Sensitive Hashing using Random Projection* (**LSH**) [82]: the state-of-the-art similarity-based look-alike modeling method based on LSH with the cosine similarity function.

Table 3.3: Recall values of the Segment Recovery Task at $m = 1$.

Dataset	**LSH**	**ITQ**	**BA**	**1-SVM**	**Poly2**	**LR**	**FM**	**GBT**	**FA**	**AFA**	*p-value*
Apparel-1	0.3487	0.4143	0.4656	0.5223	0.5798	0.6102	0.5769	0.6036	0.6606	**0.8528**	8e-03
Electronics-1	0.3485	0.3939	0.4343	0.2273	0.3535	0.5758	0.4595	0.6566	0.8116	**0.8434**	9e-11
Home/Garden	0.3783	0.4858	0.4570	0.5412	0.5783	0.5818	0.5711	0.6311	0.6389	**0.7581**	8e-08
Electronics-2	0.0217	0.0217	0.0217	0.0001	0.0001	0.0002	0.0068	0.0435	0.5975	**0.6522**	9e-11
Animal/Pet	0.4554	0.4870	0.5476	0.7568	0.8397	0.9137	0.8733	**0.9413**	0.8281	0.8997	4e-13
Apparel-2	0.4536	0.6665	0.6383	0.7568	0.7328	0.7548	0.7239	**0.7764**	0.6650	0.7659	3e-07

- *Iterative Quantization* (**ITQ**) [50]: another similarity-based look-alike modeling method based on the state-of the-art linear, hashing model.
- Binary Autoencoders (**BA**): a similarity-based look-alike modeling method using autoencoder with a fixed random noise for binarization similar to that of semantic hashing defined in Section 3.4.2.
- *1-class SVM* (**1-SVM**): a regression-based look-alike modeling method which learns a probability density function using only the seed-set X_o. We observe that "linear kernel" gives the best results in our experiments.
- *Degree-2 Polynomial* (**Poly2**): a regression-based look-alike modeling method employing the second-order polynomial classification model. The positive class includes seed users while the negative class is created from randomly selected negative users from the database. Different from LR, Poly2 includes second-order feature interaction but the effective input is significantly higher dimensional and sparser.
- *Linear Regression* (**LR**): the most popular regression-based look-alike modeling method by learning a linear classification model using seed users as positive class and randomly selected negative users from the database.
- *Factorization Machine* (**FM**) [110]: a very effective regression-based method that uses Factorization Machine as the classifier. FM can effectively learn second order feature interaction from the original high-dimensional sparse input.
- *Gradient Boosting Tree* (**GBT**) [40]: similar to LR, another popular regression-based method that employs a boosting-based classifier. GBT can learn high-order feature interactions.
- *Factorization Autoencoder* (**FA**): our proposed similarity-based look-alike modeling using factorized autoencoder with similar fix-random noise as that in BA as defined in Section 3.4.3.
- *Adversarial Factorization Binary Autoencoder* (**AFA**): our proposed FA that is regularized by a discriminator in an adversarial training procedure as defined in Section 3.4.4.

Implementation Details: We use the implementation of LSH provided in NearPy package [3]. For FM, we use the implementation from FastFM [4]. The implementations of LR, Poly2, and GBT are from Scikit-Learn [5]. All the autoencoders, including our proposed models, are implemented using Tensorflow [6] (Version 1.10). We use the same encoder's and decoder's network structures for all autoencoders. The encoder consists of two layers with 1000 units in each layer. For adversarial models, we use a similar two-layer architecture with 1000 units in each layer. We train the neural models using ADAM optimizer with a batch size of 100 examples, an initial learning rate of $1e-4$ and a learning decay of 0.96 at every 1000 training steps. We observe that using the batch normalization procedure stabilizes the training process and makes the training procedure converge faster and that dropout does not necessarily improve the generalization of the models. We also observe that employing the feature matching loss [114], which matches the statistics of the last hidden layer of the discriminator for real and fake samples, provides more stability in training our adversarial models. In this experiment, we measure the performance of all the comparison methods for the '*segment recovery task*'. Tables 3.3 and 3.4 show the average recall values of the methods at $m = 1$ and mAP across different segments from six different partners (as described in Section 3.3.3), respectively. To determine if the performance of our proposed methods is significant, we calculate the paired t-tests between our methods and the baselines. The improvements of our models are statistically significant with p-values < 0.01. We report the p-values between AFA and the next best baseline in Table 3.3.

Early-rise Performance Results

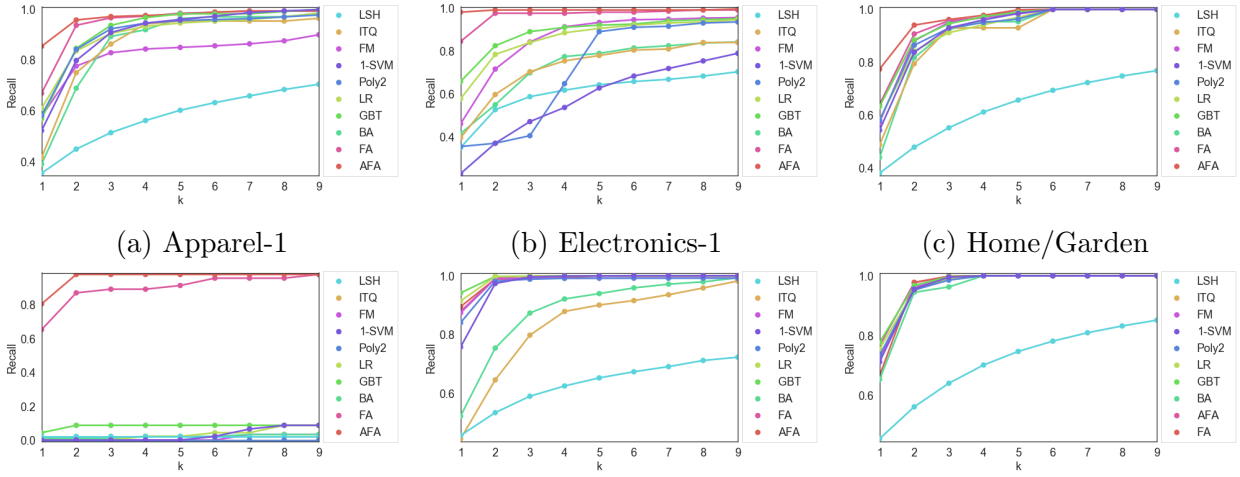
Table 3.3 details how each method performs when the advertisers desire to find a narrower set of look-alike users, a favorable condition known as "early-rise". We observe that similarity-based look-alike models, including linear hashing models (such as LSH and ITQ) and deep models such as BA, have worse recall values than those of regression-based look-alike models, especially when there are more training data (for example, for partners Apparel-1, Home/Garden, Animal/Pet, Apparel-2). As discussed in Section 3.2, this is expected because regression-based models utilize the supervised signal from the labeled data when compressing the original data into the propensity scores. However, when there are less training samples (for example, for partners Electronics-1 and Electronics-2), all similarity-based models outperforms regression-based models. This result shows an important advantage of similarity-based approaches.

While it is expected that regression-based models have significantly better performance than that of similarity-based models, our proposed deep similarity-based approaches significantly outperform even the best regression-based model, GBT, in most cases. Specifically, AFA

[3]https://github.com/pixelogik/NearPy
[4]http://ibayer.github.io/fastFM/
[5]https://scikit-learn.org
[6]https://www.tensorflow.org/

Table 3.4: mAP values for the segment recovery task of the Criteo's partners.

Dataset	**LSH**	**ITQ**	**BA**	**1-SVM**	**Poly2**	**LR**	**FM**	**GBT**	**FA**	**AFA**
Apparel-1	0.2234	0.3411	0.5158	0.4187	0.4667	0.4846	0.4211	0.4655	0.7319	**0.7623**
Electronics-1	0.2178	0.2621	0.7591	0.1307	0.2219	0.4494	0.3627	0.5322	0.9339	**0.9705**
Home/Garden	0.278	0.4104	0.5253	0.458	0.4894	0.4844	0.4924	0.5385	0.6591	**0.6880**
Electronics-2	0.0005	0.0005	0.4979	0.0007	0.0006	0.0007	0.0008	0.0036	0.6427	**0.7015**
Animal/Pet	0.3994	0.3347	0.5171	0.6869	0.7781	0.8777	0.8215	**0.9151**	0.7200	0.8496
Apparel-2	0.455	0.3164	0.4247	0.6366	0.6565	0.6785	0.6502	**0.7054**	0.6678	0.6967



(a) Apparel-1 (b) Electronics-1 (c) Home/Garden

(d) Electronics-2 (e) Animal/Pet (f) Apparel-2

Figure 3.3: Recall values of m between 1 and 5 in the Segment Recovery Task.

outperforms all other models with a significant improvement in for the first four partners while it has comparable performance with GBT for the last two partners, both of which have more training data. The result is important because AFA not only reduces the complexity of building look-alike models without considering the supervised signal (as discussed in Section 3.2), a characteristic of similarity-based look-alike models, but also achieves superior performance compared to that of regression-based look-alike models.

3.5.3 Segment Recovery Results

Quality of Look-alike Users

Complementary to Table 3.3, Figure 3.3 shows the recall curve for different values of m and Table 3.4 shows mAP values of the compared methods. Overall, the results exhibit consistent patterns as our observations from Table 3.3. While regression-based look-alike models outperform linear, similarity-based models and BA, our proposed methods show better look-alike quality in all values of m, as observed in Figure 3.3. Our proposed method, AFA, also achieves a significant improvement on the averaged precisions in most cases, as

shown in Table 3.4. More importantly, when the sizes of the seed sets are smaller (for example, for the two Electronics partners), our proposed models significantly outperform all other look-alike models.

Ablation Study

Tables 3.3 and 3.4, and Figure 3.3 show the contribution of each individual component of our proposed model with respect to all the performance metrics. FA extends BA with the bi-interaction layer and the bi-interaction embedding reconstruction as discussed in Section 3.4.3, and AFA further extends FA with the proposed adversarial training procedure as discussed in Section 3.4.4. The first observation is that the factorization component of FA contributes significantly to the improvement of the proposed models. It demonstrates that the presence of the factorization component is very important. Furthermore, employing the proposed adversarial training further improves the performance.

Discussion

To summarize, our experimental results demonstrate the following conclusions:

- *BA has worse performance than regression-based look-alike models and our proposed models*: vanilla autoencoders could not efficiently handle sparse, high-dimensional data. Such data causes deep models, in general, and specifically autoencoders to settle in bad local minima, thus the low-dimensional discrete representation is sub-optimal. The empirical results are consistent with our discussion in Section 3.2. More importantly, our proposed models efficiently learn and embed feature interactions in the low-dimensional discrete representation.
- *FA has comparable performance with regression-based look-alike models while AFA performs even significantly better in most datasets.* It exploits higher-order interactions among the features and reduces the dimensionality of the input. The output of the autoencoder effectively allowS the encoder to learn better low-dimensional discrete representation of the high-dimensional, sparse input.
- *AFA outperforms FA*: adversarial training effectively regularizes the autoencoder models.
- *Both FA and AFA outperforms regression-based look-alike models when the seed sets are small*: one advantage of regression-based models is their exploitation of the supervised signals of the seed sets but they require a more involved training process when there are many seed-sets and a full, computationally expensive linear-scan in order to find look-alike users. However, our models demonstrate that we can have a computationally

efficient method which also produces high-quality look-alike users, even in the cases where regression-based models do not perform well.

3.5.4 Parameter Sensitivity

An important hyper-parameter while using similarity-based look-alike models is the "size" of the discrete space, or the code size B. Figure 3.4 show AFA's recall values at $m = 1$ for various values of B for all the partners. We observe that AFA's performance does not noticeably change when B reaches a certain value. One possible reason is that when the discrete space is large enough, the encoder is able to map a very small number of data points to a unique discrete code, thus making it more flexible to **reconstruct** the original manifold in the discrete space. More specifically, the number of possible codes in a discrete space of size B is 2^B; for example, when $B = 4$, there are 16 possible codes and when $B = 32$ there are roughly 500 million possible codes, which makes the space large enough to accommodate the mappings of all the data points in our datasets.

To find look-alike users, similarity-based look-alike models build a "candidate set" which likely has similar users and limit the search to only this subset. This is much more computationally efficient than a full-linear scan as in the case of regression-based look-alike models. Although each model build its candidate set differently, one criteria to determine the computational efficiency of a similarity-based look-alike model is the distribution of the data points to the codes in the discrete space. A model which has better look-alike quality and lower number of data points, on average, assigned to each code is more favorable because it is also more suitable to distribute clustered data points in a distributed system for faster look-alike retrieval.

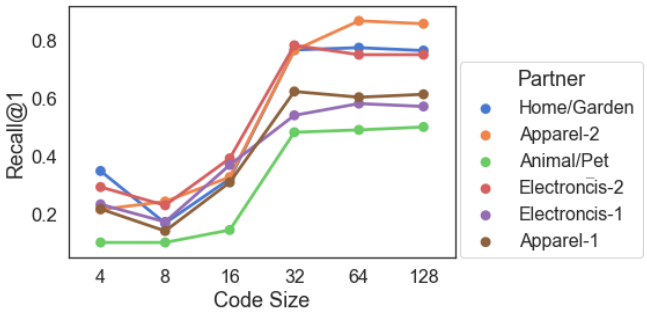
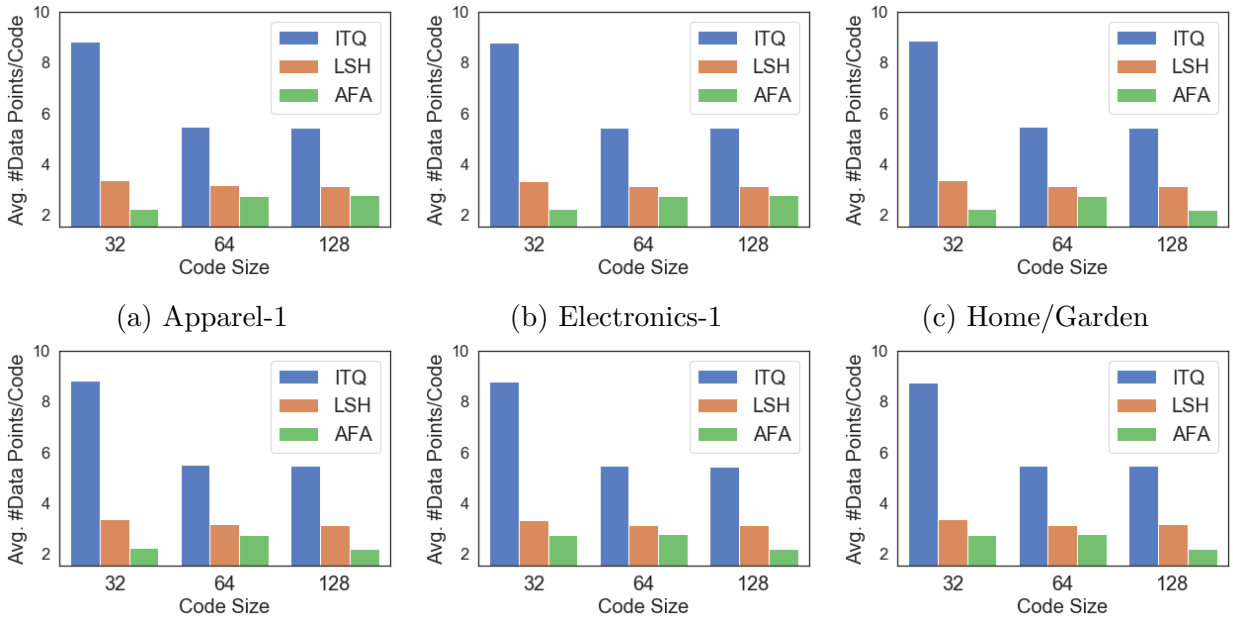


Figure 3.4: Recall values at $m = 1$ for the segment recovery task at different code sizes B.

Figure 3.5 shows the expected number of data points assigned to a discrete code of our proposed AFA model and other baseline similarity-based look-alike models. We observe that AFA has a lower expected number of points per code compared to both LSH and ITQ in Figure 3.5. This shows that our proposed model produces better quality look-alike users with fewer computations when it is deployed in a production environment.



(a) Apparel-1 (b) Electronics-1 (c) Home/Garden

(d) Electronics-2 (e) Animal/Pet (f) Apparel-2

Figure 3.5: Expected number of data points assigned to each discrete code at different code sizes for similarity-based look-alike methods. Smaller values are better.

3.6 Summary

In this chapter, we discussed the importance of end-to-end hash function learning on the retrieval performance in solving the Look-alike Modeling problem in the digital advertising domain. To this end, we proposed a novel and efficient similarity-based look-alike modeling approach that learns to embed sparse, high-dimensional users' data with complex feature interaction into a search-efficient, discrete representation space. Specifically, we developed a new and effective neural network-based interaction layer which learns higher-order interactions between features and a mechanism to constrain the autoencoder's low-dimensional embedding to become binary by employing an adversarial training procedure. We have shown that the proposed model outperforms other existing state-of-the-art methods for the segment recovery task while being both computationally efficient and easier for parallelization in a distributed computing environment.

Chapter 4

End-to-end Representation and Hash Function Learning for Document Retrieval

In Chapter 3, we have shown that adversarial autoencoders can learn good representation capturing high-order feature interaction from raw data and effective hash functions without the additional hyperparameters usually found in previous hashing approaches. In this chapter, we continue to demonstrate the effectiveness of adversarial autoencoders in learning discriminative representation of text documents by employing the two popular sequential neural models (i.e., recurrent neural networks and convolutional neural networks) for the encoders and decoders. Furthermore, the previously proposed discrete space regularization of the adversarial autoencoders use the Jensen-Shannon (JSD), which is estimated through a separate discriminator network. There are several limitations in minimizing the JSD : difficult minimax optimization, ill-defined for non-overlapping supports, frequent mode-dropping during training, and high-sample complexity. On the other hand, Wasserstein distance considers the geometry of the data and provides a robust description for the distance between distributions. In this chapter, we develop an efficient estimation for the Wasserstein distance without the difficult minimax optimization which exists in the previous adversarial learning approaches. Then, we show the effectiveness of proposed algorithm in training the adversarial autoencoder in the text retrieval domain.

4.1 Introduction

One of the crucial challenges associated with mining massive text corpora is to *efficiently* and *effectively* search for documents with **similar** *semantic* content, a problem known as "semantic similarity search" in the information retrieval domain. *Exact similarity search*, which aims to exhaustively find all the relevant documents, is often impractical due to its computational complexity. Specifically, it requires an inefficient linear scan of all documents in the database. The number of documents in the database range in the order of millions (or sometimes even billions). Thus, *approximate similarity search* algorithms that *efficiently* examine a smaller subset of candidate documents provide a principled approach for solving this problem. For instance, in *hashing*, a classic approximate-similarity search approach, we project the original input in the high-dimensional space onto a much smaller locality-preserving *binary* space. Therefore, the full-linear scan is no longer imperative. Compact binary codes are storage-efficient and the search-cost in the original high-dimensional space is reduced to calculating Hamming distances. The binary representation of a document often requires 4 to 8 bytes of storage and computing the pairwise Hamming distance using "bitwise" XOR needs a single CPU instruction.

While hashing methods ensure efficiency, an important challenge is to learn a hash function that *effectively* captures the semantic similarity between a pair of text documents, especially, in the presence of noisy data. Historically, supervised hashing techniques display a better retrieval performance, but they require human-annotated documents that are expensive to obtain on massive-scale datasets [16, 45, 90, 135]. Unsupervised hashing methods do not require supervised signals for training, thus are more suitable for massive-scale corpora [16, 50, 61, 113, 119, 138, 153].

For analyzing text data, majority of the existing unsupervised hashing methods employ shallow, linear hash functions. On the other hand, deep models are capable of learning non-linear hash functions [16, 113, 119, 153]. However, despite the improved performance, they heavily rely on manually-engineered feature vectors such as Term Frequency-Inverse Document Frequency (TF-IDF) [115]. To the best of our knowledge, no prior work in the literature attempts to learn hash functions that capture the semantic and syntactic representation of raw text documents. Although, respectable performance is observed in the context of other text mining problems [76, 101, 154, 155].

In addition to capturing optimal representation of the documents in the hash functions, the learned hash codes should also achieve bit balance/uncorrelation and low-quantization error [134]. Bit-balance and bit-uncorrelation ensure that a uniform number of training data points are assigned to each hash code. Thus, minimizing the time complexity of the retrieval task in the worst and average cases [57]. Along with low-quantization error, they alleviate the chance of assigning "very" similar data points to different codes. This preserves the original locality structure of the data in the binary space and improves the retrieval precision [50, 57, 69]. In spite of possessing better retrieval performance, existing deep text-

Table 4.1: Characteristics of hashing algorithms along with the representative methods. Our proposed method satisfies all these characteristics in an end-to-end framework without introducing complex cost functions and hyperparameters. ∘ indicates the method achieves the objective, but the procedure is ad-hoc.

	LSH [82]	SpecHash [138]	ITQ [50]	SemHash [113]	STH [153]	VDSH [16]	NASH [119]	DABA (Ours)
Non-linear	×	×	×	✓	✓	✓	✓	✓
Low Quantization Error	×	×	✓	∘	✓	×	×	✓
Bit Balance	×	✓	✓	∘	✓	✓	✓	✓
Bit Uncorrelation	×	✓	✓	✓	✓	✓	✓	✓
Data Dependent	×	✓	✓	✓	✓	✓	✓	✓
Learn from Raw Text	×	×	×	×	×	×	×	✓
Robust to Noisy Data	×	×	×	×	×	×	×	✓

hashing methods either ignore these objectives or construct them heuristically. We argue that the retrieval performance is significantly improved by incorporating them in learning the deep hash functions. Furthermore, we prove that it is possible to *implicitly learn the hashing objectives* in an *adversarial-training framework*. Thus, we can eliminate the need to tune several "hyperparameters" when explicitly defining the heuristic hashing objectives in a learning-to-hash model.

To address these aforementioned challenges, we propose *a novel unsupervised **D**enoising **A**dversarial **B**inary **A**utoencoder* (**DABA**) model for the text hashing problem. The proposed DABA model learns a non-linear hash function that captures an optimal representation directly from sequential input data and is robust to the noise in the input data through the application of an adversarially-trained denoising autoencoder. The main contributions of the chapter are as follows:

- Propose a novel unsupervised deep generative autoencoder-based hashing model that jointly learns an efficient representation of text documents directly from the raw text and a robust, deep hash function in an end-to-end framework. To the best of our knowledge, this is the first work in text hashing that learns a hash function directly from the raw text data.
- Employ adversarial training procedure; the proposed method introduces a new and efficient alternative to train binary neurons that implicitly generate balanced and discriminative hash codes without introducing additional hyperparameters or complex hashing objective functions. Thus, fulfilling the desired characteristics of an ideal hashing algorithm (as shown in Table 4.1).
- Propose a novel algorithm to train the adversarial, binary autoencoder that directly estimates the Wasserstein distance from the primal domain by solving the Optimal Transport (OT) problem. The proposed algorithm does not employ the familiar min-max game, which is harder to train, because of the fluctuating generator's cost. Thus, increasing its suitability for large-scale and real-world text hashing problems.

- Demonstrate the effectiveness of our method in large-scale text datasets and demonstrate the superiority of the proposed model over state-of-the-art hashing techniques with both quantitative and qualitative analysis of the performance.

The rest of the chapter is organized as follows. We discuss the related work in Section 4.2. In Section 4.3, we describe the details of our proposed method. Finally, we present quantitative and qualitative experimental results in Section 4.4 and conclude in Section 4.5.

4.2 Related Work

In this section, we describe two lines of research that are closely related to the proposed work: hashing and deep learning.

4.2.1 Similarity Search and Hashing

Hashing

The most representative class of efficient mechanisms for semantic similarity search is hashing. Hashing techniques can be broadly classified into supervised [16, 45, 90, 135] and unsupervised hashing [16,50,61,113,138,153]. Although supervised hashing methods demonstrate better performance [16, 119], they require human-annotated documents that are expensive to obtain on massive-scale datasets, which are increasingly prevelant in the current scenario. Scarcity of labeled training data introduces the problem of train/test distribution mismatch and poor local optima converge in supervised learning frameworks. Therefore, their search performance degrades significantly.

Unsupervised hashing approaches address these problems by learning hash functions without any supervised signal. Thus, these methods are more suitable for large-scale corpora. Predominantly, unsupervised hashing methods include shallow and deep models. Locality sensitive hashing (LSH) approaches, which learn the hash functions using random projection, are popular shallow models with appealing theoretical properties [34, 82, 125, 133].

Data-dependent shallow-hashing methods, notably Spectral Hashing (SpecHash) [138] and Semantic Hashing (SemHash) [113], demonstrate significant performance improvements over LSH. SpecHash learns compact, balanced and uncorrelated-bit codes by solving the Eigenvector problem. However, the method strongly assumes that the data points are uniformly distributed in a hyper-rectangle, which restricts its application potential. Spherical Hashing (SpheHash) [61], on the other hand, employs hypersphere-based partitioning to achieve better and coherent quantization. Iterative Quantization (ITQ) additionally minimizes the

quantization error to generate hash functions that better preserves the original locality structure of the input data. However, these shallow models are linear and hence cannot be applied to real-world high-dimensional datasets.

Unsupervised, deep-hash models [16,113,119,153], display remarkable performance improvements compared to the shallow models by learning non-linear hash functions. Self-taught hashing (STH) [153] decomposes the algorithm into learning the hash function (similar to that of SemHash) and learning a hash function on unseen data using a supervised learning method. Variational Deep Semantic Hashing (VDSH) [16] employs variational autoencoders to learn a generative model to reconstruct the original documents. Neural Architecture for Semantic Hashing (NASH) [119] adopts a similar variational autoencoder architecture as that of VDSH but models the hashing codes as Bernoulli latent-variable. These existing deep hash models, in spite of having better performance, do not exploit the latent semantic and syntactic dependencies in the sequential text data to learn the hash functions. For example, both VDSH and NASH employ the hand-crafted TF-IDF input. To the best of our knowledge, our proposed model is the first end-to-end hashing model that captures the structured representation of raw text documents.

Characteristics of Hash Functions

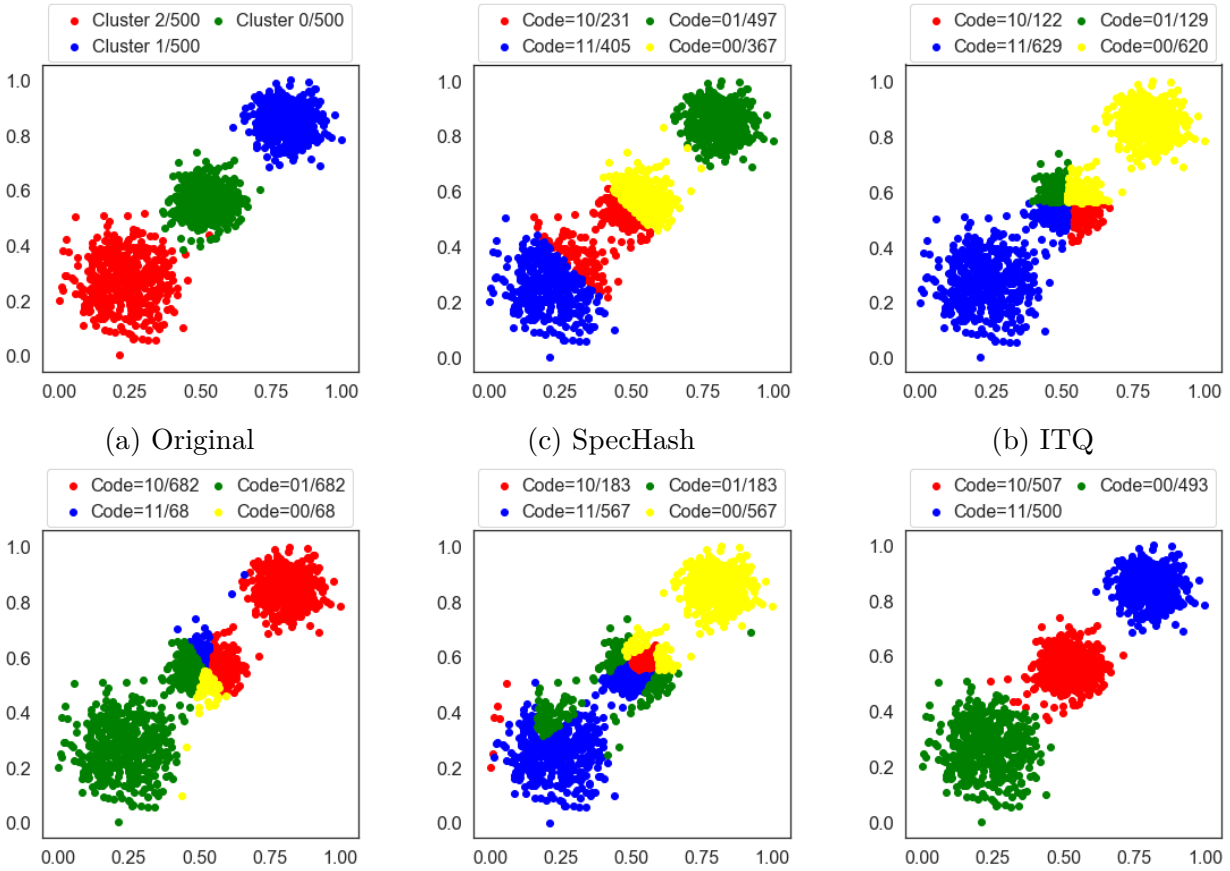
Prevalent hashing approaches minimize the training objective as follows:

$$\min_{f} E_{x\sim D_x} L(x, f(x)) + E_{x\sim D_x} \sum_{k} \lambda_i \times H_k(f(x)) \tag{4.1}$$

where D_x is the data distribution, $L(x, f(x))$ is the locality-preserving loss of the hash function $f(x)$ and $H_k(f(x))$ is a hashing objective with λ_k as its corresponding hyperparameter. Examples of H_k are:

- *Bit balance* [138]: each bit has the same chance of being 0 or 1.
- *Bit uncorrelation* [138]: different bits are uncorrelated.
- *Low quantization error* [50]: low information loss by encoding the real-valued representation space by the binary, discrete space.

In hashing, including H_k's are important for improving the retrieval performance. For example, bit-balance and bit-uncorrelation encourage "code-balance", i.e., the condition where a uniform number of training data points is assigned to each hash code. Code-balance minimizes the time complexity of the worst and average cases, thus, improving the retrieval efficiency [57]. Furthermore, code-balance and low-quantization error alleviate the chance of assigning "very" similar data points to codes that have large Hamming distances. Hence, they better preserve the original locality structure of the data [50,57,69]. Preserving locality



(a) Original (c) SpecHash (b) ITQ

(d) VDSH (e) NASH (f) DABA-MLP

Figure 4.1: A synthetic example of 1500 data points generated from three Gaussian clusters – 500 data points per cluster (Figure 4.1a). Each method learns a 2-bit hash function without using the supervised cluster-labels. An optimal hash function achieves "code balance" and recovers the original cluster assignment as much as possible.

improves the retrieval precision. It is critical that these three objectives are jointly optimized in any efficient hashing algorithm. For example, without the bit-uncorrelation constraint, an algorithm can still achieve bit balance. A trivial example is assigning half the number of data points to a code with all 0-bits and the remaining half to a code with all 1-bits. However, the quality of the hash codes are apparently poor because there are only two possible codes regardless of the number of bits.

Figure 4.1 depicts a concrete example of data points that are independently drawn from three two-dimensional Gaussian clusters (500 data points per cluster). Figure 4.1a illustrates the original generated data points. SpecHash and ITQ, the shallow-hashing methods, could not learn a non-linear separation of the data points (Figures 4.1b-4.1c). On the other hand, VDSH and NASH, the deep-hashing methods, are able to learn non-linear hash functions

Table 4.2: Notations used in this chapter.

Notation	Description
$x, \tilde{x}, \hat{x}$	the original, corrupted and reconstructed input
$x^{(i)}$	the i^{th}-indexed sample input document
$\hat{w}_{x,l}, w_{x,l}$	predicted and actual words at position l in document x
$\hat{x}_l$	predicted word vector at position l in document x
D_x	data distribution
b, c	sigmoid-representation vector and the corresponding thresholded binary-code vector
z	the sampled input (from $B_p(z)$) to the discriminator
z_i	the i^{th} component of the vector z
$b^{(i)}, z^{(j)}$	the representation of the i^{th}-indexed document and the j^{th}-indexed Bernoulli sample, respectively
$q(b)$	posterior distribution of b
f, g	encoding and decoding functions
$W_e[v]$	output (of dimension e) of the embedding lookup of word v
L_A	reconstruction loss function for the autoencoder
L_M	distribution-matching loss (i.e., divergence)
Θ_E, Θ_D	parameters of the encoder (generator) and the decoder

but they do not achieve code-balance (the number of data points per hash code has high variance, as shown in Figures 4.1d and 4.1e). The reason is that they ignore bit-balance, bit-uncorrelation and low-quantization error. Only the proposed model, DABA-MLP, is able to recover the original cluster's data point assignments almost perfectly and achieve code-balance. DABA-MLP is a non-linear technique and, more importantly, is constrained to generate both bit-balance and bit-uncorrelation. Furthermore, distinct from existing approaches, the hashing objectives (or H_k) are **implicitly optimized** (using adversarial training) without tuning additional hyperparameters. A summary of the characteristics of different text hash algorithms is given in Table 4.1.

4.2.2 Semantic Hashing and Deep Learning

SemHash [113] is one of the early deep hash models. SemHash learns a binary vector representation of the input data using Restricted Boltzmann Machine (RBM) tuned autoencoders. The hashed binary vectors capture the semantic dependency of the input data in a lower-dimensional space. However, RBM is harder to train for achieving a generative power similar to that of a denoising autoencoder [7]. Furthermore, denoising autoencoders produce noise-invariant low-dimensional representations of the data leading to more robust hash functions [132].

To encourage the code layer to be "binary", SemHash regularizes the sigmoid units with deterministic Gaussian noise during training. This introduces additional hyper-parameters that increase the complexity of the learning process. Although, SemHash and similar semantic models [135, 153] do not require additional hashing objective functions such as the quantization or bit-entropy loss (i.e., is H_k's), the sigmoid layer outputs more activation values near 0 than near 1 (because of the asymmetry between 0 and 1 of the RBM's energy function). Thus, selection of a threshold value becomes heuristic and can be suboptimal. Our DABA training squashes activations of the sigmoid layer almost equally near 0 and 1, achieving "bit balance" [134] (see Figure 4.4) and encouraging the choice of the same threshold value (0.5) in all datasets.

Representation Learning

Recurrent neural network (RNN) provides a natural and conceptual way to capture the characteristics of sequential data [51]. RNNs have been successfully applied in several domains [101, 156]. In [76], the authors proposed a method to compress a sequence into its vector representation. Similar to RNN, Convolutional Neural Networks (CNN) are also proposed for learning an efficient representation that captures semantic and syntactic structures of text documents for various problems [154, 157]. To the best of our knowledge, there is no prior work in the text domain to simultaneously learn document representation and an efficient hash function by directly exploiting such structures from raw textual data.

Generative Adversarial Network and Adversarial Training

Generative Adversarial Network (GAN) has recently gained popularity due to its ability to generate realistic samples from the data distribution [52]. A prominent feature of GAN is its ability to "implicitly" *match* outputs of a deep network to a pre-defined distribution using the adversarial training procedure. In fact, adversarial training has been selected in several works as a regularization of the latent representation of the autoencoders [97, 158]. [158] report that adversarially-trained autoencoders efficiently learn the intrinsic manifold of the data, especially, structured data without overfitting and poor local minima convergence. Furthermore, in contrast to other generative models such as Variational Autoencoders [74], it is possible to learn an "optimal" hash function implicitly by adversarial regularization of the hash function with a "binary" distribution to guide the learning process without imposing any additional constraints H_k's [30]. However, training adversarial autoencoders remains challenging and inefficient because of the alternating-optimization procedure (min-max game) between the generator and the discriminator. For example, [30] employs the original min-max GAN objective [52], which suffers from mode-collapse and vanishing gradient [2]. Moreover, in min-max optimization, the generator's loss fluctuates during training instead of "descending", making it extremely challenging to know when to stop the training process. Wasserstein GAN overcomes a few of these limitations (specifically, mode-collapse

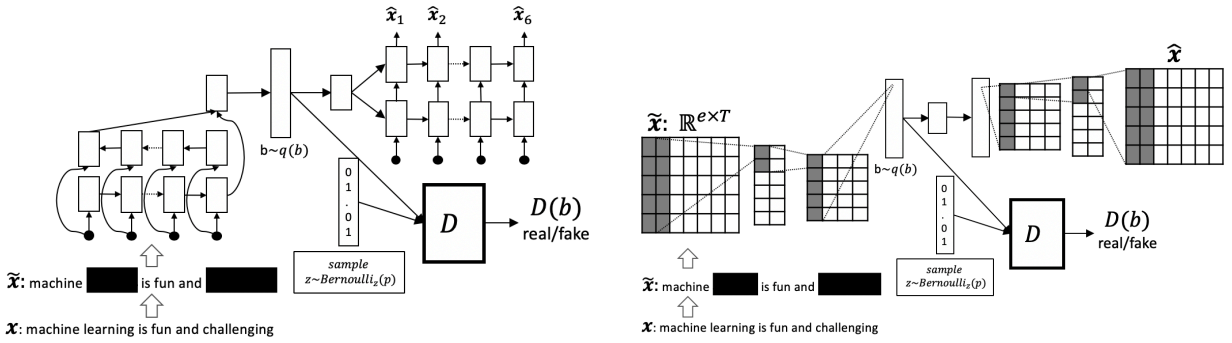
and vanishing gradient) [2]. However, it approximates the Wasserstein distance by employing the Kantorovich-Rubinstein dual, thus, resulting in a similar min-max game between the generator and the critic. The authors of [129] propose Wasserstein Autoencoder but still employs the min-max game. We propose to directly approximate the Wasserstein distance from the primal direction by solving the Optimal Transport problem [13].

4.3 Proposed Model

4.3.1 Problem Formulation

Given a query document e, the *similarity search problem* is to find a set of K similar documents $S(e)$ from a database $X = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ of N documents such that, given $sim(x, y)$ as a pre-defined similarity function that measures the similarity between two documents x and y, we have $sim(e, x) \leq sim(e, y)$ for all $x \in S(e), y \in X \setminus S(e)$.

In hashing, the goal is to find a hash function $c = f(x)$ whose output c, called "hash code", is a low-dimensional, binary vector in $\{0, 1\}^{|c|}$, where the notation $|c|$ is the cardinality (or dimension) of c, such that the relationship between x and y, through $sim(x, y)$, is preserved in $\{0, 1\}^{|c|}$. Learning a discrete-output function f is intractable [134], thus we instead learn a continuous function $b = f(x) \in \mathbb{R}^{|b|}$, from which c is obtained by thresholding b. In the text domain, each data point x or y is a vector in a high-dimensional space $\mathbb{R}^n$, such as the TF-IDF vector, or a sequence of one-hot vectors of the words representing the raw text document. $|b|$ is typically small and takes a value between 16 to 128. The notations used in this chapter are given in Table 4.2. In the following sections, we will discuss the components of the proposed method.



(a) DABA-RNN (b) DABA-CNN

Figure 4.2: Overall network architecture of the proposed Denoising Adversarial Binary Autoencoder (DABA) model.

4.3.2 Binary Autoencoder

Given a dataset $X = \{x|x \sim D_x\}$, a binary autoencoder defines an encoding function $f : x \to b$ that maps each data point x into a point $b \in \mathbb{R}^{|b|}$ in the coding space and a decoding function $g : b \to \hat{x}$ that recovers x from b. When the encoder's output layer (i.e. the last layer) is sigmoid, thresholding the activations b (e.g., at 0.5) will return a binary code vector $c \in \{0, 1\}^{|b|}$ which is considered as the discrete hash code of the input x. Modeling the hash function as a structured, deep encoder, captures complex latent document representation from the raw text. Therefore, we define the binary autoencoder with the following network structure:

- *An encoder function* $f : x \to b$ maps a sequence of text x into a low-dimensional vector b. In our work, we model f as either a bidirectional RNN or a convolutional neural network (CNN), followed by a single-layer projection onto the sigmoid layer:
  - The RNN encoder (Figure 4.2a) is a multi-layer, bidirectional RNN with Long-Short-Term-Memory (LSTM) Units. We observe similar performance using more than one layer and other types of hidden units, such as Gated Recurrent Units [19].
  - The CNN encoder (Figure 4.2b) consists of multiple convolutional layers. For example, in our experiments on 20NewsGroups and Reuters datasets, given an embedding input $\tilde{x} \in \mathbb{R}^{e \times T}$ where T is the max length of input sequences and e the word's embedding dimension, we find that the best performance is achieved by using two convolutional layers with Rectified Linear Unit (ReLU) activation function with kernels $\{5 \times e, 5 \times 1\}$, strides $\{2, 2\}$ and numbers of filters $\{300, 600\}$, resulting in $\{c_1 \times 300, c_2 \times 600\}$ feature maps, respectively (where c_1 and c_2 are the dimensions of the feature maps at layer 1 and layer 2, respectively). This is followed by a max-out layer of 300 filters of $c_2 \times 1$ kernel and a stride of 1.
- *A decoder function* $g : b \to \hat{x}$ that reconstructs the input x as $\hat{x}$. The function g is modeled as a single-layer linear projection followed by an RNN (for an RNN-encoder) or a CNN (for a CNN-encoder) to reconstruct the input x:
  - The RNN decoder (Figure 4.2a) is a multi-layer LSTM network whose output $\hat{x}_l$ at position l in the reconstructed sequence $\hat{x}$ of L words is the predicted softmax probability of word $\hat{w}_{x,l}$ at position l in the reconstructed sequence $\hat{x}$, defined as

$$p(\hat{w}_{x,l} = v) = \phi(\hat{h}_l, W_e[v]) \tag{4.2}$$

    where $\hat{h}_l$ is the hidden state representation of the LSTM decoder at time-step l, $W_e[v]$ is the embedding of word v, and $\phi(v_1, v_2)$ is the softmax function.
  - The CNN decoder (Figure 4.2b) consists of deconvolutional (convolutional transpose) layers to decode the hidden representation back to the original embedded

input [157]. The probability that the predicted word at position l in the reconstructed sequence $\hat{x}$, $\hat{w}_{x,l}$, is v can be defined as follows:

$$p(\hat{w}_{x,l} = v) = \frac{\exp(\tau^{-1} cos(\hat{x}_l, W_e[v]))}{\sum_{v' \in V} \exp(\tau^{-1} cos(\hat{x}_l, W_e[v']))} \tag{4.3}$$

where τ is the Gumbel-softmax parameter, V is the vocabulary, and $cos(.)$ is the cosine distance function. We use $\tau = 0.01$ in our experiments.

It is important to note that modeling the encoder/decoder with other network architectures such as Attention [130] is a straightforward extension to this chapter. The objective function of the binary autoencoder described above can be written as the word-wise negative log-likelihood, averaged across all documents, as follows:

$$L_{\mathrm{A}} = E_{x \sim D_x} \left[\frac{1}{L} \sum_{l=1..L} - \log \ p(\hat{w}_{x,l} = w_{x,l}) \right] \tag{4.4}$$

where $w_{x,l}$ is the actual word at position l in the input document x.

4.3.3 Denoising Binary Autoencoder

As discussed in Section 4.2, denoising autoencoders achieve generative power similar to that of RBM-based autoencoders which are arduous to train [7]. Moreover, the denoising component allows the autoencoder to robustly learn the hash function of corrupted data by utilizing the semantic relationship of uncorrupted words in the text. Therefore, we propose to extend the binary autoencoder, as defined in Section 4.3.2, with a denoising component.

Specifically, for each word-position in a document, we randomly remove the word with probability p_d. p_d is, therefore, the fraction of the words of the input sequences that need to be removed. The input to the model is the corrupted input $\tilde{x}$ and the model attempts to recover x. Denoising autoencoders are also known to help in learning a smooth, low-dimensional manifold of the data similar to that of RBMs [9]. We call this model Denoising BA (DBA).

4.3.4 Denoising Adversarial Binary Autoencoder

An autoencoder defines an encoding distribution $q(b|x)$ and a decoding distribution $p(x|b)$. Consequently, the encoding step imposes a posterior distribution on the coding space as follows:

$$q(b) = \int_x q(b|x) D_x(x) dx \tag{4.5}$$

The posterior distribution is also synonymous to the hash's output distribution. Therefore, if we force $q(b)$ to "agree" with a binary-like distribution, such as a discrete "Bernoulli prior", that has all desired characteristics of a good hashing function as discussed in Section 4.2, the adversarial training procedure will force the encoder function $f(x)$ to generate binary codes with the same desired characteristics. Specifically, an input vector z is a sample from a Bernoulli distribution $B_p(z)$ with a parameter p if each of its components z_i is independently drawn from the Univariate Bernoulli distribution with parameter p. Given $z \sim B_p(z)$, we regularize the posterior distribution $q(b)$ to be similar to $B_p(z)$. The adversarial binary autoencoder (ABA) is trained with dual objectives as follows:

$$\min_{\Theta_E, \Theta_D} L_A + L_M \tag{4.6}$$

where Θ_E is encoder's parameters, Θ_D is the decoder's parameters and

1. L_A is the reconstruction error from the encoding space b in the decoding step, defined in Section 4.3.2.

2. L_M is the regularization term. It is the divergence of the encoding posterior distribution $q(b)$ and the Bernoulli-prior distribution $B_p(z)$ where p is set to 0.5 (hereafter, we denote $B_{0.5}(z)$ as $B(z)$). An intuitive explanation of the regularization is that every component of b will learn to divide as optimal as possible the original space into two halves (thus its activation has about 50% chance of being closer to 1 or closer to 0 – a "bit balance" [134]), where the points in each half are more similar to those in the same half compared to those in the other half – "bit uncorrelation" [134].

We solve Equation (4.6) by playing a game between the generator and discriminator, equivalently minimizing the Jensen Shannon divergence [158]; in other words, the training objective is as follows:

$$\min_{\Theta_E, \Theta_D} \left(L_A + \max_{\Theta_{Disc}} L_M \right) \tag{4.7}$$

where Θ_{Disc} is the parameters of the discriminator. The authors of [129] generalize this framework to Wasserstein distance, but it employs a similar min-max game. In this chapter, we follow the primal direction of estimating the Wasserstein distance, thus, removing the min-max game. Specifically, in the primal domain, Wasserstein distance is defined as follows:

$$W(q(b), B(z)) = \inf_{\gamma \in \Pi(q(b), B(z))} \int_{(b,z)\sim\gamma} p(b,z) d(b,z) dbdz \tag{4.8}$$

where $\Pi(q(b), B(z))$ is the set of all possible joint distributions of b and z whose marginals are $q(b)$ and $B(z)$, respectively, and $d(b,z)$ is the cost of transporting one unit of mass from b to z. This approach is equivalent to solving the optimal transport (OT) problem,

whose objective is to find the optimal transport plan to move masses from the generated distribution $q(b)$ to the true distribution $B(z)$. Given two finite samples of N examples of b and N examples of z, the empirical transport cost can be formulated as the following Linear Programming (LP) problem:

$$\hat{W}(q(b), B(z)) = \min_{M} \sum_{i}^{N} \sum_{j}^{N} M_{i,j} d(b^{(i)}, z^{(j)}) = \min_{\Theta_E} M \odot D \tag{4.9}$$

where M is the assignment matrix, D is the cost matrix where $D_{ij} = d(b^{(i)}, z^{(j)})$ and $\odot$ is the Hadamard product. The LP formulation has the following constraints:

$$\sum_{i}^{N} M_{i,j} = 1, \forall j = 1, ..., N \tag{4.10}$$

$$\sum_{j}^{N} M_{i,j} = 1, \forall i = 1, ..., N \tag{4.11}$$

$$M_{i,j} \in \{0, 1\}, \forall i = 1, ..., N, \forall j = 1, ..., N \tag{4.12}$$

It is important to note that both $M_{i,j}$ and $d(b^{(i)}, z^{(j)})$ are functions of Θ_E. *The "distribution matching cost" L_M in Equation (4.6) is equivalent to $\hat{W}(q(b), B(z))$.* Given the optimal assignment matrix $M*$ that solves this LP problem and a learning rate α, we can update the parameters of the generator, Θ_E, as follows:

$$\Theta_E^{\text{new}} = \Theta_E - \alpha \frac{\partial \hat{W}^*(q(b), B(z))}{\partial \Theta_E} \tag{4.13}$$

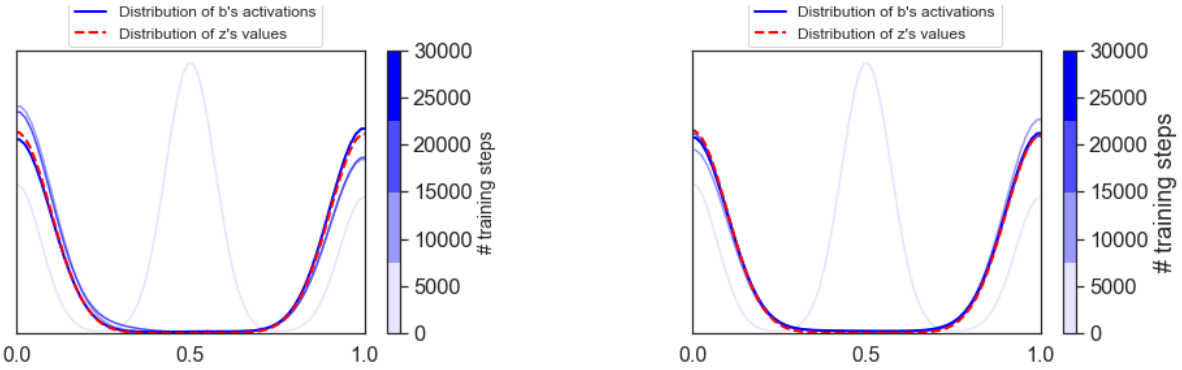
$$\frac{\partial \hat{W}^*(q(b), B(z))}{\partial \Theta_E} = \frac{\partial M^*}{\partial \Theta_E} \odot D + M^* \odot \frac{\partial D}{\partial \Theta_E} \tag{4.14}$$

While $M*$ depends on Θ_E, it is known that a small perturbation of Θ_E does not change the transport plan; that is $\frac{\partial M^*}{\partial \Theta_E} = 0$. Thus, we update the generator using only the second term in Equation (4.14), $M^* \odot \frac{\partial D}{\partial \Theta_E}$, which is easy to compute given a differentiable distance function $d(b, z)$. In our work, we use the Euclidean distance where $d(b, z) = ||b - z||_2$.

Minimizing L_M drives the distribution $q(b)$ closer to that of $B(z)$. Intuitively, this is assigning each data point b to the closest sample z in the Euclidean space (given that $d(b, z)$ is the Euclidean distance). Since the Bernoulli-data generating process samples each component z_k of z independently (hence, each bit is uncorrelated) from a Univariate Bernoulli distribution with probability 0.5 (hence, each bit has 0.5 probability of being 1), matching $q(b)$ against

the $B(z)$ is exactly equivalent to maximizing bit-uncorrelation and bit-balance codes after the assignment. The assigned z of the vector b can also be viewed as its quantized vector c. In other words, c is approximately the same as z. Consequently, it minimizes the quantization error when L2-norm cost $d(b, z) = ||b - z||_2$ is applied (because the quantization error is defined as $||b - c||_2$).

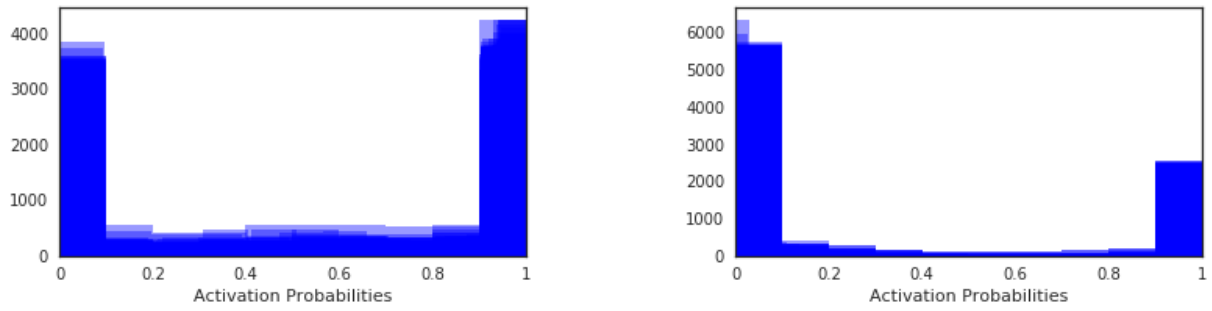
The implicit distribution-matching process between b and z is illustrated in Figure 4.3. At the beginning of the training process, the distribution of b-neurons' output (or activations) is a Gaussian (a result of uniform-random weights' initializations). However, after several training steps, the distribution of b's activations become very similar to that of the sampled values of z from $B(z)$.



(a) Training Data (b) Validation Data

Figure 4.3: Distribution of activations of b after several training steps and distribution of values of z, sampled from $Bernoulli_z(0.5)$, of DABA created on the 20Newsgroup dataset.

We call this model as Denoising Adversarial BA (DABA). The overall architecture of our DABA-RNN and DABA-CNN models is shown in Figure 4.2. The adversarial training



(a) DABA-Epoch 60 (b) SM-Epoch 60

Figure 4.4: Activations (values of b) of DABA and Semantic Hashing (SM) after 60 epochs.

Algorithm 2 DABA Model Training

Input: Training data X,
Denoising parameter p_d,
Binary code size $|b|$,
Bernoulli sampling parameter p,
Number of training iterations K.

Output: $\{\Theta_E\}$ parameters of the encoder

for *number of training iterations* K **do**

- Sample a minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$.
- Randomly drop elements of $x^{(i)}$ with probability p_d to $\tilde{x}^{(i)}$.
- Sample m vectors $\{z^{(1)}, ..., z^{(m)}\}$ where $z^{(i)} \sim B_p(z)$.
- Update the autoencoder parameters, Θ_E and Θ_D by minimizing L_A in Eq. (4.4).
- Solve the LP problem given in Eq. (4.9) to find M^*.
- Update the encoder parameters Θ_E using the update rule described in Eqs. (4.13) and (4.14).

end

algorithm is described in Algorithm 2. For training our model, we adopt the standard minibatch stochastic gradient descent procedure, using Adam optimizer [73] with $p = 0.5$.

Figure 4.4 depicts the distribution of the activations (b) of encoder's output layer on the 20NewsGroups dataset for Semantic Hashing and DABA with TF-IDF input [115] and multilayer perceptrons (MLP) as encoder/decoder (called DABA-MLP with similar network architecture as that of Semantic Hashing). In both methods, we observe that the activations squashed out to the borders near 0 and 1. DABA (shown in Figure 4.4a), however, learns to output activations that are more equally squashed out than those of Semantic Hashing (shown in Figures 4.4a). This allows us to simply pick 0.5 as a threshold for binarizing the activations, as compared to a heuristic suggestion of 0.1 for Semantic Hashing [113].

4.4 Experimental Results

4.4.1 Datasets Used

We utilize the following datasets in the performance evaluation experiments of the proposed DABA model.

- **20NewsGroups** [1]: A collection of 18,821 newsgroup documents categorized uniformly into 20 different newsgroups. The data is split into 90% of the documents that serve as the database and 10% of the documents for querying.
- **Reuters** [2]: A collection of 12,902 Newswire stories provided by Reuters, Ltd. and organized into 135 categories. We randomly split 90% of the documents as the database and the remaining 10% of the documents for querying.
- **DBpedia** [155]: A collection of 630,000 documents classified into 14 non-overlapping ontology classes. The data is split into 560,000 training documents that serve as the database and 70,000 testing documents for querying.

4.4.2 Evaluation Procedure

For evaluating the performance of the proposed model, we follow the standard mechanism that is widely accepted in the context of the ranking problem- the **precision** metric at various threshold values m ($P@m$):

$$\text{precision}@m = \frac{|\text{retrieved, relevant documents}|}{|\text{retrieved documents}|} \tag{4.15}$$

where the threshold m is the number of top ranked candidate documents, based on their Hamming distances to the query document, obtained by the algorithms for the retrieval task. For the experiments, we set m to be 10, 50, and 100.

In addition to this, we calculate the limited areas under the Precision-Recall curves (equivalent to limited mean average precision – or MAP) up to a certain threshold m, called MAP@m for different methods [117]. In our experiments, we report MAP@1000.

4.4.3 Comparison Methods

We compare the performance of the proposed method with various representative similarity-search methods.

- *Locality Sensitive Hashing* (**LSH**) [82]: the popular data-independent, shallow hashing method using random projection.
- *Spectral Hashing* (**SpecHash**) [138]: an unsupervised shallow hashing method whose goals are to preserve locality and find balanced, uncorrelated hashes by solving the Eigenvector problem.

[1] `http://qwone.com/~jason/20Newsgroups/`
[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

- *Spherical Hashing* (**SpheHash**) [61]: a hypersphere-based space-partitioning shallow hashing method.
- *Iterative Quantization* (**ITQ**) [50]: the state-of-the-art shallow hashing method that alternately minimizes the quantization error to achieve better hash codes.
- *Semantic Hashing* (**SemHash**) [113]: the widely-used unsupervised deep hashing method where the hash function is modeled as a deep generative model using Restricted Boltzmann Machine.
- *Self-taught Hashing* (**STH**) [153]: an extension of SemHash that also learns a hash function on unseen data.
- *Variational Deep Semantic Hashing* (**VDSH**) [16]: the deep hashing method that learns the hash function through a generative model using variational Autoencoders (VAE) with TF-IDF input.
- *Neural Architecture for Semantic Hashing* (**NASH**) [119]: the state-of-the-art deep hashing method that, similar to VDSH, learns the optimal hashing using VAE whose input are TF-IDF vectors. NASH models the hashing codes as a Bernoulli latent-variable.
- *Denoising Binary Autoencoder* (**DBA**): the binary autoencoder as defined in Section 4.3.2, with denoising input.
- *Denoising Adversarial Binary Autoencoder* (**DABA-MLP**): our proposed adversarially regularized autoencoder (described in Section 4.3.4) with multi-layer perceptrons as encoder/decoder whose input are the TF-IDF vectors of the documents.
- *DABA with RNN encoder/decoder* (**DABA-RNN**): our proposed DABA model (in Section 4.3.4) with RNN encoder and RNN decoder.
- *DABA with CNN encoder/decoder* (**DABA-CNN**): our proposed DABA model (in Section 4.3.4) with a Convolutional encoder and Deconvolutional decoder.

First, we pre-process the data by removing common stop words. Then, we transform each document into its TF-IDF [115] vector representations with 2,000 components for 20Newsgroup and Reuters and 5,000 components for DBpedia. For DABA-CNN and DABA-RNN, we learn the hash functions directly from the sequential text data.

We run the experiments of LSH using random projection on TF-IDF input. For SemHash and STH, we employ the network sizes similar to the ones used in the original papers [113, 153] for both the 20Newsgroup and Reuters datasets and two layers of $1000 \rightarrow 500$ units for DBpedia. We use similar network sizes for DABA-MLP. For 20Newsgroup and Reuters, VDSH's networks use 1,000 hidden nodes (as suggested in [16]), while NASH's networks have two layers of $500 \rightarrow 500$ units (as suggested in [119]); for DBpedia, 1,500 hidden nodes

for VDSH and two layers of 1000 $\rightarrow$ 500 units for NASH. For DABA-RNN, we use an embedding size of 200 for 20Newsgroup and Reuters and 300 for DBpedia. For all DABA methods, the discriminator D is a MLP with two layers of 500 $\rightarrow$ 500 hidden units. To choose the denoising parameter p_d, we perform a grid search with p_d ranging from 0 to 0.5. For all these methods, we report the average MAP@1000 (shown as MAP) and Precision@k by repeating the experiments three times in order to ensure statistically meaningful results.

We implemented our proposed methods using Tensorflow[3]. In our experiments, we employ the learning rate of 0.001 for 20Newsgroup and Reuters datasets; 0.01 for the DBpedia dataset. We use a dropout rate of 0.2 and 0.5 for the small and large datasets, respectively. We also apply batch normalization [64] for our DABA-CNN method. Given N examples, the best method of solving the OT's LP program has a cost of approximately $O(N^{2.5}\log(N\mathcal{D}))$, where $\mathcal{D} = max_{i,j}d(b^{(i)}, z^{(j)})$ and the distances $d(b^{(i)}, z^{(j)})$ are scaled up to be integers [13]. While this is computationally expensive for large N, the cost is less than 100ms (where N is less than 2048) in a conventional CPU [4]. Therefore, it is entirely possible to implement this LP program in a mini-batch SGD training. In our experiments, we operate on a large mini-batch size of 500 examples in training DABA. This reduces the variance of the empirical primal Wasserstein estimate, while making the training process very efficient. The code of our proposed method is released on Github [5].

4.4.4 Results

Retrieval Performance

Table 4.3: Performance comparison of different methods using standard metrics (precision at various threshold values m and mean average precision) for the document retrieval task on various text datasets.

	20Newsgroup				Reuters				DBpedia			
	P@10	**P@50**	**P@100**	**MAP**	**P@10**	**P@50**	**P@100**	**MAP**	**P@10**	**P@50**	**P@100**	**MAP**
LSH	0.456	0.388	0.301	0.259	0.455	0.429	0.386	0.389	0.507	0.485	0.417	0.296
SpecHash	0.511	0.448	0.382	0.307	0.782	0.719	0.651	0.429	0.519	0.455	0.431	0.323
SpheHash	0.512	0.455	0.385	0.316	0.752	0.710	0.664	0.421	0.463	0.427	0.408	0.304
ITQ	0.524	0.457	0.384	0.320	0.799	0.731	0.667	0.430	0.516	0.457	0.422	0.336
SemHash	0.520	0.435	0.390	0.322	0.780	0.703	0.650	0.462	0.553	0.503	0.452	0.313
STH	0.523	0.501	0.565	0.328	0.817	0.798	0.755	0.476	0.568	0.506	0.483	0.312
VDSH	0.552	0.550	0.431	0.339	0.839	0.815	0.775	0.495	0.634	0.531	0.475	0.324
NASH	0.579	0.544	0.539	0.349	**0.867**	0.840	0.799	0.501	0.610	0.557	0.505	0.331
DBA	0.491	0.511	0.562	0.301	0.810	0.788	0.679	0.471	0.596	0.526	0.479	0.325
DABA-MLP	0.559	0.542	0.565	0.339	0.836	0.831	0.772	0.497	0.629	0.541	0.511	0.341
DABA-RNN	0.628	0.579	0.565	**0.366**	0.859	0.833	**0.830**	**0.520**	0.641	0.576	**0.539**	0.348
DABA-CNN	**0.639**	**0.612**	**0.590**	0.341	0.860	**0.861**	0.820	0.509	**0.655**	**0.598**	0.537	**0.358**

In this experiment, we measure the performance of the compared methods on the *document retrieval task*. Table 4.3 shows the average precision values at various retrieval thresholds m and the average MAP. Improvements of our models over the compared methods are

[3] `https://www.tensorflow.org/`

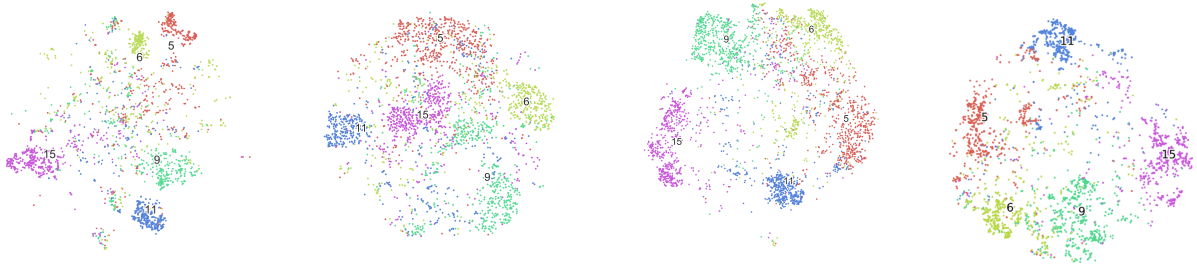
[4] https://github.com/gatagat/lap

[5] https://github.com/khoadoan/daba-hashing

statistically significant according to the corresponding paired t-tests (p-value < 0.01). As expected, shallow-hash functions have worse performance than other deep hashing methods. SemHash and DBA have comparable performance across the metric, which supports our discussion/claims in Section 4.3.3 that denoising autoencoders achieve a similar result as that of RBM-based autoencoders. However, RBM-based autoencoders are harder to train.

While there are mixed results when comparing the performance of the existing shallow hashing methods and the compared deep models, it is clearly observed that the proposed DABA-MLP significantly improves the retrieval quality compared to all these methods, including DBA – the DABA-MLP version that is not adversarially regularized. This supports our discussion/claims in Section 4.2 that adversarial training, besides generating better hash codes, helps the network learn a better semantic manifold of the data than that of a similar network but without adversarial regularization (this statement will again be supported by the qualitative experiments about the learned manifold of the data in Section 4.4.4). Finally, from Table 4.3, it is clearly noticed that DABA-RNN and DABA-CNN significantly outperform all other methods for most of the retrieval metrics. Also, DABA-CNN achieves better results compared to DABA-RNN, albeit comparable. We conjecture that the superior performance of all DABA methods is due to the following reasons: (1) the hash functions learned by DABA better preserve the semantic manifold of the documents (DABA-MLP performs better than DBA and SemHash/STH), (2) exploiting the semantic and syntactic structures of text documents (one of the main contributions of this chapter) results in better retrieval quality (DABA-RNN/CNN perform better than DABA-MLP) and (3) the learned hash functions generate better hash codes (DABA-MLP performs better than all other shallow and deep models).

Hashing Space Manifold



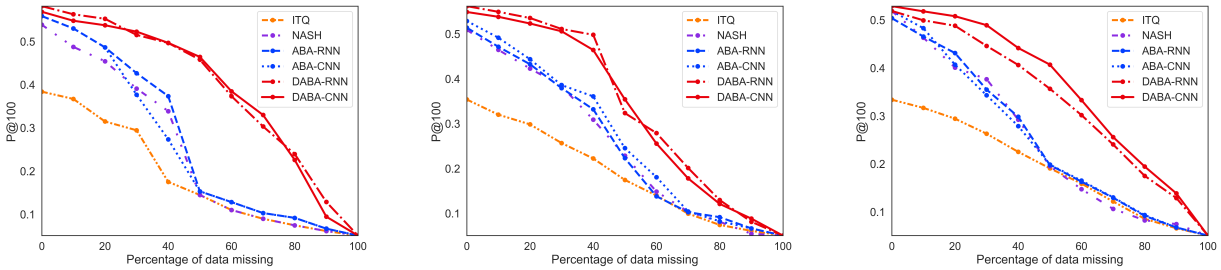
(a) SpecHash (b) VDSH (c) NASH (d) DABA-RNN

Figure 4.5: Two-dimensional t-SNE manifold visualization of the coding space (values of b) for five randomly selected topics, namely, Computer/Windows (5, Red), For Sale (6, Yellow-Green), Sport/Baseball (9, Green), Science/Cryptology (11, Blue), and Religion/Christian (15, Purple).

In this experiment, we plot the embedding of the documents in the binary address space us-

ing the 2-D t-SNE projection [95], of the learned binary codes of various methods. In Figure 4.5 (created using the 20NewsGroups dataset) the documents belonging to the same topic are represented using the same color. As observed in this figure, in the DABA-RNN/CNN results, more documents belonging to the same topics are clustered together (while the clusters are more distant from each other) compared to the selected shallow and deep methods. Most "Religion/Christian" documents (purple) or "Sport/Baseball" documents (green), for example, are better separated from the rest of the documents. This demonstrates the effectiveness of the proposed method in preserving the semantic manifold of the documents in the coding space.

Analysis with Missing Data



(a) 32-bit code (b) 64-bit code (c) 128-bit code

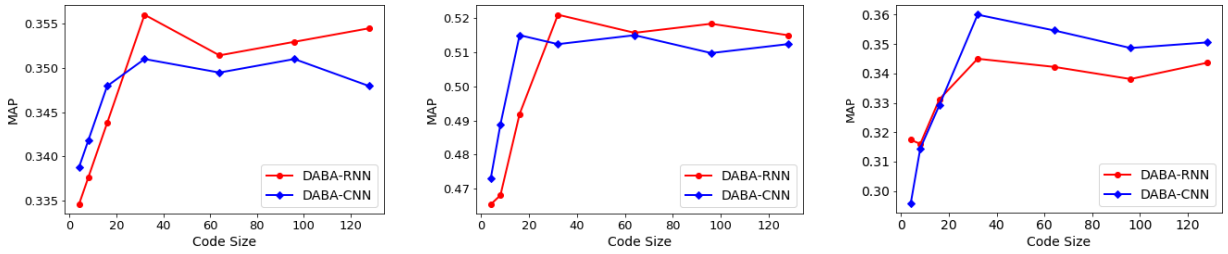
Figure 4.6: Precision@100 of various methods including DABA-CNN and DABA-RNN when data is missing for various bit sizes on the 20Newsgroup dataset. ABA-RNN and ABA-CNN are our proposed methods without the denoising component.

In this experiment, we evaluate the performance, specifically Precision@100, of various selected methods against our proposed models when the data contains missing values. In the 20NewsGroups dataset, for each document, we randomly remove a predefined fraction of the words ranging from 0% to 100% in the query documents. We investigate the performance of our methods, DABA-CNN and DABA-RNN, when the denoising components are removed, resulting in ABA-CNN and ABA-RNN, respectively. This allows us to carry out an ablation study to understand the contribution of the proposed denoising component with regards to learning robust hash functions.

Figure 4.6 shows the performance of the methods at various percentages of missing data for different bit sizes on the 20Newsgroup dataset. The compared models, including ABA-CNN and ABA-RNN, have steeper performance decline than those of DABA-CNN and DABA-RNN when the missing data rate increases. DABA-CNN and DABA-RNN are more robust than the other methods in the presence of missing data. Note that all methods converge to the same performance, that is of the random guess, when the missing data rate is approaching 100%. The results demonstrate the robustness of employing the denoising component within

our hashing framework toward missing data, a realistic scenario when deploying the similarity search system in a real-world environment.

Parameter Sensitivity



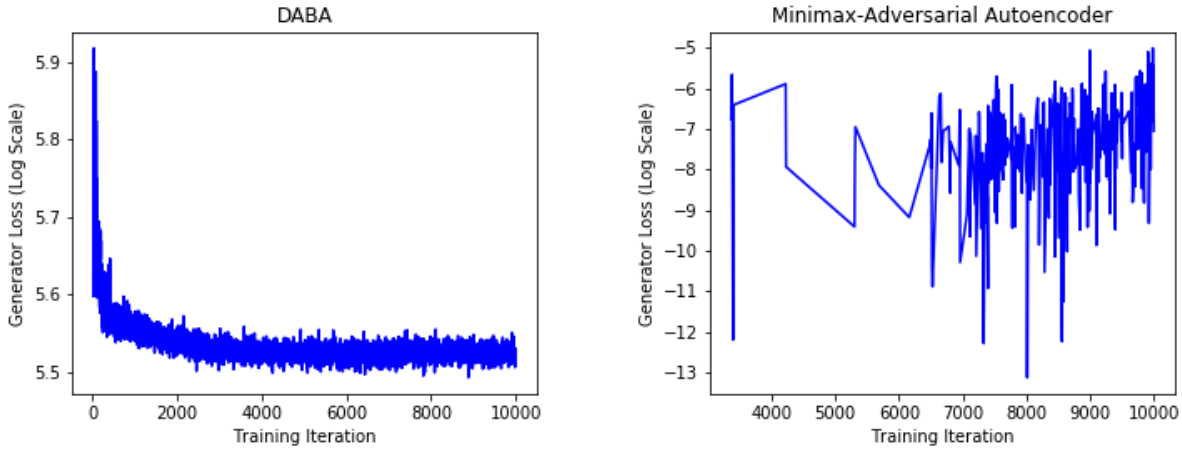
(a) 20NewsGroups (b) Reuters (c) DBpedia

Figure 4.7: Performance results (MAP) when varying the size of the learned binary codes on 20Newsgroup and DBpedia.

In this experiment, we study the relationship between the embedding size B of the binary code b and the retrieval performance. Figure 4.7 illustrates the MAP performance of DABA-CNN as the value of B is varied from 4 to 128. We observe that the retrieval performance quickly increases during the initial phase (when using smaller codes), following which it reaches the optimal value. Increasing the code sizes after that optimal code size results in neither performance gain nor any significant performance loss. This suggests that if we train DABA networks with a sufficiently large code size B, we are guaranteed to obtain a near-optimal performance. This experiment illustrates that the performance of the proposed DABA model is not sensitive to the learned code size parameter.

Stability during training

One of the subtleties of training a min-max adversarial network is to decide the stopping criterion for the training process, that is "when does the model reach the equilibrium?". Figure 4.8 illustrates the generator's loss, L_M, for Wasserstein-Autoencoder [129] and DABA in the 20Newsgroup training. We observe that the loss fluctuates in the min-max game during training, while our proposed OT loss gradually descends. This displays the training efficiency of our proposed training algorithm. This effectively suggests that we can utilize the criteria to detect convergence such as early stopping to automatically train DABA, making it suitable for real-world hashing problems.



(a) DABA Training

(b) Min-max Training

Figure 4.8: Convergence behavior of our proposed loss functions during the training process of 20Newsgroup. DABA is trained using the empirical OT loss without the min-max game. The min-max autoencoder is trained using the Wasserstein Kantorovich-Rubinstein dual. In this game, the loss has poor fluctuation, thus is it difficult to know when to stop training.

4.5 Summary

We proposed a novel and efficient denoising adversarial binary autoencoder for the text hashing problem. To achieve this, we developed a new and effective mechanism to constrain the network's hidden layer and force them to become binary by regularizing the hash-function learning process with an adversarial network with a discrete distribution prior and corrupted input that the denoising autoencoder must reconstruct. Employing the OT formulation, we train our proposed model without the min-max game and significantly stabilize the training process of the adversarial autoencoder. We demonstrate that the proposed model using RNN or CNN encoder/decoder, that effectively exploits the semantic and syntactic dependencies of text documents, outperforms other existing state-of-the-art semantic-similarity search methods.

Chapter 5

Image Hashing by Minimizing Discrete Component-wise Wasserstein Distance

In Chapter 3 and Chapter 4, we have shown that the adversarial autoencoders are effective models for learning the hash functions without domain heuristic model designs and additional hyperparameters to enforce the discrete constraints, in the computational advertising and text retrieval domains, respectively. In addition, the Linear Sum Assignment formulation, proposed in Chapter 4, allows us to learn the model more effectively without the difficult minimax optimization. However, the Wasserstein distance, as well as the Jensen-Shannon Divergence, still has high-sample complexity; i.e., the number of samples required to accurately estimate the distance grows exponentially with respect to the dimension of the latent space. Furthermore, solving the Linear Sum Assignment is computationally expensive. These issues make the previous adversarial models highly unsatisfactory to be deployed in real applications. In this section, we consider another variant of the Wasserstein distance, called the Sliced Wasserstein Distance, and develop an efficient estimation algorithm which exploits the properties of the learned discrete space. Specifically, the proposed Sliced Wasserstein Distance has a polynomial sample complexity and a significantly-lower computational cost than the previous approaches. We demonstrate the effectiveness of proposed approach in image retrieval domain.

5.1 Introduction

The rapid growth of visual data, especially images, brings many challenges to the problem of finding similar items. Exact similarity search, which aims to exhaustively find all relevant images, is impractical due to its computational complexity. This is due to the fact that a complete linear scan of all the images in such massive databases is not feasible, especially when the database contains millions (or billions) of items. Hashing is an approximate similarity search method which provides a principled approach for web-scale databases. In hashing, high-dimensional data points are projected onto a much smaller locality-preserving *binary* space via a hash function $f : x \rightarrow \{0, 1\}^m$, where m is the dimension of the binary space. *Approximate search for similar images can be efficiently performed* in this binary space using Hamming distance [82]. Furthermore, the compact binary codes are storage-efficient.

The existing hashing methods can be broadly grouped into supervised and unsupervised hashing. Although supervised hashing offers a superior performance, unsupervised hashing is more suitable for large databases because it learns the hash function without any labeled data. Unsupervised hashing can also be sub-categorized as shallow hashing methods [50, 138], and deep hashing methods [47, 149, 150]. Among the deep hashing methods, adversarial autoencoder-based methods show superior performance in the advertising and text domains [29, 30].

Despite good performance, training the adversarial autoencoders by minimizing the Jensen-Shannon or Wasserstein distance is extremely difficult, especially when the dimension of the latent space increases. The scaling difficulty of the adversarial autoencoders may be related to one fundamental issue: the generalization property of matching distributions. The authors of [3] show that Jensen-Shannon divergence and Wasserstein distance do not generalize, in a sense that the generated distribution cannot converge to the target distribution without an exponential number of samples. *Without good generalization, the retrieval performance can be sub-optimal.*

To address these aforementioned challenges, we propose a novel unsupervised Discrete Component-wise Wasserstein Autoencoder (DCW-AE) model for the image hashing problem. DCW-AE implicitly learns the optimal hash function using a novel and efficient divergence minimization framework. The main contributions of this chapter are as follows:

- Demonstrate that the ability to match the distribution of the output of the learned hash function to the target discrete distribution is closely related to the retrieval performance. Specifically, employing a distance measure with an easier convergence to the target distribution (called generalization) results in better retrieval performance. To this end, the existing Wasserstein-based Adversarial Autoencoders have poor generalization; thus they have a sub-optimal retrieval performance.
- Propose a novel, efficient approach to learn the hash functions by employing a more

generalizable variant of the Wasserstein distance, that leverages the discrete properties of hashing. It has an order-of-magnitude better generalization property and an order-of-magnitude more efficient computation than the existing Wasserstein-based hashing methods.

- Demonstrate the superiority of the proposed model over the state-of-the-art hashing techniques on various widely used real-world datasets using both quantitative and qualitative performance analysis.

The rest of the chapter is organized as follows. We discuss the related work in Section 5.2. In Section 5.3, we describe the details of the proposed method. Finally, we present quantitative and qualitative experimental results in Section 5.4 and conclude our discussion in Section 5.5.

5.2 Related Work

5.2.1 Image Hashing

Various supervised [14,45,120,140,151] and unsupervised methods [24,25,50,58,61–63,85,113, 138, 150] have been developed for hashing. While supervised methods demonstrate superior performance over unsupervised ones, they require human-annotated datasets. Annotating massive-scale datasets, which are common in the image hashing domain, is an expensive and tedious task. Furthermore, besides the train/test distribution-mismatch problem, supervised methods easily get stuck in bad local optima when labeled data are limited. Thus, exploring the unsupervised hashing techniques is of great interest.

Hashing methods can also be categorized as shallow [48, 61, 138] and deep hashing [24, 25, 63, 85, 150]. The deep hashing methods can learn non-linear hash functions and have shown superiority over the shallow approaches.

Until recently, the deep hashing methods relied on "heuristic" loss functions to learn the optimal hash codes where the heuristic definitions altered from domain to domain [30]. The authors of [29,30] show that by matching the latent space of the autoencoder with an optimal discrete prior, we can implicitly learn a good hash function f while simultaneously satisfying the constraints. However, their adversarial methods are unstable in practice and do not show a good generalization property. In Section 5.4, we will show that poor generalization results in a sub-optimal retrieval performance when the model is trained with stochastic optimization techniques such as Stochastic Gradient Descent (SGD).

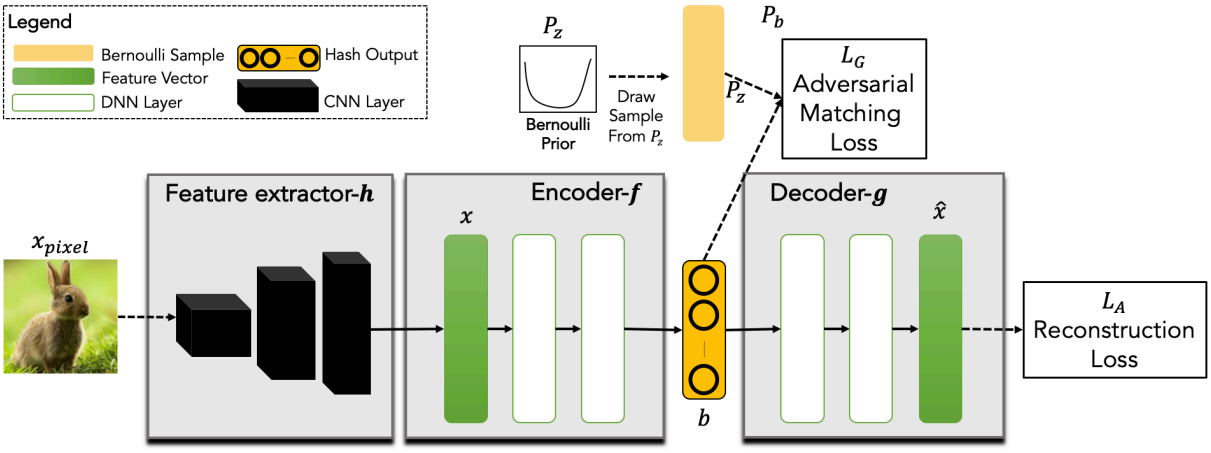


Figure 5.1: The network architecture of the proposed DCW-AE model. During the training phase, the parameters of f and g are trained, while the parameters of h can be fixed or trained. For the sake of a fair comparison with the other methods, we fixed the parameters of h in our experiments. The encoder/hash function f learns to preserve the locality in the original space in the hash code space through the reconstruction loss L_A and generate the balanced hash codes b through the adversarial loss L_G.

5.2.2 Adversarial Learning and Generalization

Generative Adversarial Network (GAN) has recently gained popularity due to its ability to generate realistic samples from the data distribution [52]. A prominent feature of GAN is its capability to "implicitly" *match* output of a deep network to a pre-defined distribution using the adversarial training procedure. Furthermore, adversarial learning has been leveraged to regularize the latent space, as it helps in learning the intrinsic manifold of the data [30, 97]. For example, the adversarially trained autoencoders can learn a smooth manifold of the data in the low-dimensional latent space [97]. However, training the adversarial autoencoders remains challenging and inefficient because of the alternating-optimization procedure (minimax game) between the generator and the discriminator. Mode-collapse and vanishing gradient [2] are two main problems of the minimax game [52]. Moreover, in the minimax optimization, the generator's loss fluctuates during training instead of "descending", making it extremely challenging to know when to stop training the model.

Wasserstein-based adversarial methods overcomes a few of these limitations (specifically, mode-collapse and vanishing gradient) [2,129]. Nonetheless, as they employ the Kantorovich-Rubinstein dual, the minimax game still exists between the generator and the critic. The work in [29, 65], on the other hand, directly estimates the Wasserstein distance by solving the Optimal Transport (OT) problem. Solving the OT has two main challenges. Firstly, its computation cost is $O(N^{2.5}log(Nd))$ where N is the number of data points and d is the dimension of the data points. This is expensive considering N is in the order of thou-

sands or millions of data points. Secondly, the OT-estimate of the Wasserstein distance requires an exponential number of samples to generalize (or to achieve a good estimate of the distance) [3]. In practice, both the high-computational cost and exponential-sample requirement make the OT-based adversarial methods highly inefficient. Recently, Doan et al. [28] employ the Fréchet distance to approximate the Wasserstein-2 distance without the discriminator similar to the OT approaches, but their proposed method still has exponential sample requirement.

Another variant of the Wassertein distance is Sliced Wasserstein Distance (SWD), which approximates the Wasserstein distance by averaging the one-dimensional Wasserstein distances of the data points when they are projected onto many random, one-dimensional directions [23]. SWD is more generalizable than the OT, with polynomial sample complexity [22]. Furthermore, the SWD estimate has a computational cost of $O(N_\omega N \log(Nd))$, where N_ω is the number of random directions. Hence, the computational complexity of SWD estimate is better than that of the OT *only when* N_ω *is smaller than* $N^{1.5}$. Nevertheless, in the high dimensional space, it becomes very likely that many random, one-dimensional directions do not lie on the manifold of the data. In other words, along several of these directions, the projected distances are close to zero. Consequently, in practice, the number of random directions N_ω is often larger than $N^{1.5}$. For example, in [23], for a mini-batch size of 64, SWD needs $N_\omega = 10,000$ projections, which is significantly larger than $64^{1.5}$, to generate good visual images. To address this problem, Max-SWD finds the best direction and estimates the Wasserstein distance along this direction [22]. In this chapter, we address the limitations of these GAN-based approaches by robustly and efficiently minimizing a novel variant of the Wasserstein distance. By exploiting the properties of the target distributions in hashing, the proposed adversarial hashing method is significantly more efficient than both the existing OT-based and SWD-based approaches.

5.3 Proposed Method

5.3.1 Problem statement and Notations

Given a data set $X = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ of N images where $x^{(i)} \in R^d$, the goal of unsupervised hashing is to learn a hash function $f : x \to b$ that can generate binary hash code $b \in \{0, 1\}^m$ of the image x. The length of the hash code b is denoted by m and it is typically much smaller than d.

Table 5.1 shows the notations used in this chapter.

Table 5.1: Notations used in this chapter.

Notation	Description
x_{pixel}, x	the pixel vector and the extracted feature vector of the image, respectively
h, f, g	feature extractor, encoder, and decoder, respectively
W_h, W_f, W_g	parameters of the feature extractor, encoder, and decoder, respectively
d	dimension of the data
m	dimension of the discrete space
c, b	discrete code and its continuous representation
$W, \hat{W}$	Wasserstein distance and its empirical estimate, respectively
z	sample from the discrete prior P_z
P_z, P_b	distributions of z and b, respectively
$\mathcal{D}, \mathcal{F}$	empirical samples of P_z and P_b, respectively
N	number of samples
ω_k	vector that defines the projection onto a random one-dimensional direction.
N_ω	number of random projections
L_A	autoencoder's reconstruction loss
L_G	adversarial matching loss (also called the distributional distance)

5.3.2 Network architecture

We propose the DCW-AE network. Figure 5.1 shows the architecture of DCW-AE. Similar to the existing image hashing approaches [149,150], we choose to employ a feature extractor, such as the VGG network [124], and represent an image by its extracted feature vector x. In particular, the feature extractor is defined as the function $h : x_{pixel} \to x$, where x_{pixel} is the pixel-representation of the image. Note that, h can be trained in an end-to-end framework similar to that of [30]. However, in our work, we choose to use the pretrained VGG-feature extractor h and do not retrain its parameters.

The encoder, represented by the function $f : x \to b$ and parameterized by W_f, computes the low-dimensional representation b. Given the feature vector x of an image, the output $b = f(x)$ is represented by the m independent probabilities $b_i = p(c_i = 1|x, W_f)$. To generate the hash codes, we simply compute $c_i = \mathbf{1}_{[b_i > 0.5]}$. Note that the encoder f is also the hash function for our purpose. We then regularize the posterior distribution of b, called P_b with a predefined, discrete prior P_z by minimizing their distributional distance L_G (discussed in Sections 5.3.4 and 5.3.5). The decoder, represented by the function $g : b \to x$ and parameterized by W_g, reconstructs the input, denoted as $\hat{x}$. We train our model by minimizing the reconstruction loss L_A (discussed in Section 5.3.3). In the following sections, we will discuss the details of the proposed method.

5.3.3 Locality preservation of the hash codes

The autoencoder is trained to minimize the mean-squared error between the input and the reconstructed output, as below:

$$L_A = \frac{1}{N}\sum_{j}^{N} ||\hat{x}^{(j)} - x^{(j)}||_2^2 = \frac{1}{N}\sum_{j}^{N} ||g(f(x^{(j)})) - x^{(j)}||_2^2 \tag{5.1}$$

It is possible to show that minimizing the reconstruction loss L_A is equivalent to preserving locality information of the data in the original input space [21]. In other words, the proposed autoencoder model learns the hash function f that preserves the original input locality. Our approach to preserve the locality of the input in the discrete space using an autoencoder is different from the approaches taken by SSDH [149] and DistillHash [150]. These methods heuristically constructs the semantic, pairwise similarity matrix from the representation x. Consequently, the retrieval performance closely depends on the quality of the representation x and the constructed similarity matrix. In several domains, especially new domains, pre-trained feature extractors are not readily available or appropriate. As we shall see in Section 5.4, when a good feature extractor h is not available, our approach significantly outperforms these methods.

5.3.4 Implicit optimal hash function learning

We regularize the encoder's output b to match a pre-defined binary prior. Specifically, we sample a vector z as the real data. Each component of z is independently and identically sampled from a one-dimensional Bernoulli distribution with a parameter p. The sampling procedure defines a distribution P_z over z while the encoder defines a distribution P_b over the latent space b. The encoder, which is the generator in the GAN game, learns its parameters W_f by minimizing the Wasserstein distance as follows:

$$L_G = W(P_b, P_z) = \inf_{\gamma \in \Pi(P_b, P_z)} \int_{(b,z)\sim\gamma} p(b,z)d(b,z)dbdz \tag{5.2}$$

where $\Pi(P_b, P_z)$ is the set of all possible joint distributions of b and z whose marginals are P_b and P_z, respectively, and $d(b, z)$ is the cost of transporting one unit of mass from b to z.

Given a finite, N-sample $\mathcal{F} = \{b^{(1)}, b^{(2)}, ..., b^{(N)}\}$ from P_b and a finite, N-sample $\mathcal{D} = \{z^{(1)}, z^{(2)}, ..., z^{(N)}\}$ from P_z, one approach is to minimize the empirical Optimal Transport (OT) cost as follows:

$$\hat{W}(\mathcal{D}, \mathcal{F}) = \min_{W_f} \sum_{i}^{N}\sum_{j}^{N} M_{ij} d(b^{(i)}, z^{(j)}) = \min_{W_f} M \odot D \tag{5.3}$$

where M is the assignment matrix, D is the cost matrix where $D_{ij} = d(b^{(i)}, z^{(j)})$ and $\odot$ is the Hadamard product. This Linear Programming (LP) program has the following constraints:

$$\sum_{i}^{N} M_{ij} = 1, \forall j, \sum_{j}^{N} M_{ij} = 1, \forall i, M_{ij} \in \{0, 1\}, \forall i, j$$

As discussed in Section 5.2, the OT is computationally expensive to solve has an exponential sample complexity. The combination of these two factors makes the OT less suitable in practice, where large mini-batches are necessary for the models to perform well.

5.3.5 Discrete Component-wise Wasserstein Distance

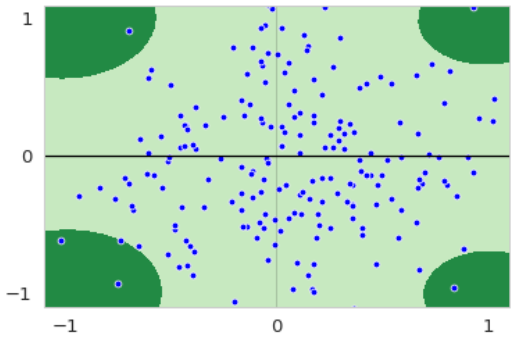
In our experiments, we observe that it is difficult to match P_b to P_z by solving the OT. We conjecture that the reason is because the OT is a poor estimate of the Wasserstein distance. It has high variance when the number of samples N in the mini-batches is small [65]. On the other hand, SWD is a more "generalizable" distance estimate than the OT [22]. Generalization refers to the number of samples the algorithm needs to converge to the target distribution.

The empirical SWD of two samples $\mathcal{D}$ and $\mathcal{F}$ is estimated as follows:

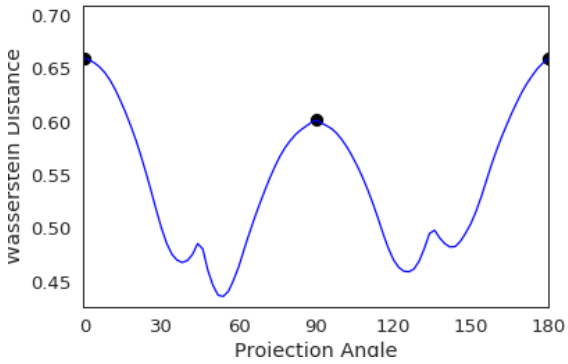
$$\hat{W}(\mathcal{D}, \mathcal{F}) = \frac{1}{|N_\omega|} \sum_{k=1}^{N_\omega} W(\mathcal{D}\omega_k, \mathcal{F}\omega_k) \tag{5.4}$$

where $\omega_k \in R^d$ is a vector which defines the projection onto a random, one-dimensional direction and N_ω is the number of such random projections. While SWD has a better sample complexity than the OT, it needs a high number of random directions.

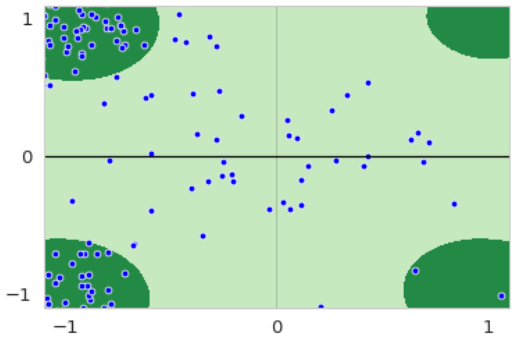
In hashing, and especially in matching the latent space of b to the target, discrete prior P_z, each sample z of P_z lies at the vertices of the hypercube. Without loss of generality, assume that z is two dimensional, therefore in $\{-1, 1\}^2$ (this is similar to sampling $z \in \{0, 1\}^2$ where the only difference is the activation function *tanh* instead of *sigmoid* after the logit output of the encoder f). A discrete sample z falls into one of the four possible corners, as seen in Figures 5.2a and 5.2c. Figures 5.2b and 5.2d show the one-dimensional Wasserstein distances for different directions at different angles from the vector $(1, 0)$. In Figure 5.2b, when the generated data points b are further away from the corners, the projections onto the directions along the axes (those with angles $45r^\circ$ for different integer values r) have the most distances, thus best describe the separation between the samples of b and z. The projections onto other directions will underestimate this separation between the samples of z and b. In Figure 5.2d, when more data points are closer to the corners, the projections onto 0° or 180°-degree direction still best separate the samples of z and b. In other words, if we project the data points onto these axes and average the one-dimensional Wasserstein distances along



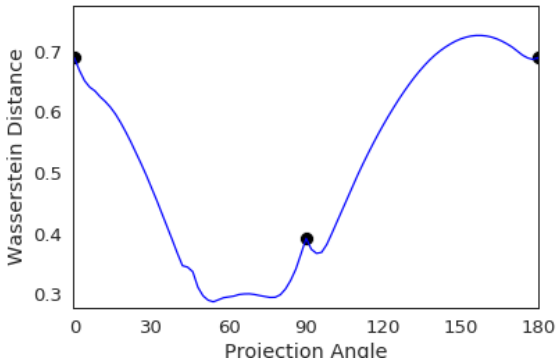
(a) P_b is far from P_z.



(b) 1-D $\hat{W}$ along different projection angles (in degrees) when P_b is far from P_z.



(c) P_b is close to P_z.



(d) 1-D $\hat{W}$ along different projection angles (in degrees) when P_b is close to P_z.

Figure 5.2: The hash output b (fake, scattered points, in blue) versus the prior, target hash codes z (real, the corner clusters, in green) in the 2-D discrete space. In Figures 5.2a and 5.2c, the objective is to push the data points closer to the corners. Clearly, projections onto the axes (black vertical and horizontal lines) result in the most correction. In Figures 5.2b and 5.2d, we can clearly see this effect.

these axes, the resulting Wasserstein distance is a better distance estimate compared to the random projections. Therefore, this motivates us to estimate the Wasserstein distance by averaging the distances along each dimension or b and z, as follows:

$$L_G = \tilde{W}(\mathcal{D}, \mathcal{F}) = \frac{1}{m} \sum_{i}^{m} \hat{W}(\mathcal{D}_i, \mathcal{F}_i) \tag{5.5}$$

where $\mathcal{D}_i = \{z_i^{(1)}, z_i^{(2)}, ..., z_i^{(N)}\}$ and $\mathcal{F}_i = \{b_i^{(1)}, b_i^{(2)}, ..., b_i^{(N)}\}$. Solving the OT in the one-dimensional space has a significantly small computational cost [23]. The cost of such operation is equivalent to the one-dimensional array sort plus the distance calculation. Therefore,

the Wasserstein-p distance L_G can be calculated as follows:

$$L_G = \tilde{W}(\mathcal{D}, \mathcal{F}) = \frac{1}{m} \sum_{i}^{m} \sum_{j}^{N} ||z^{(\sigma_i(j))} - b^{(\sigma_i(j))}||_p \tag{5.6}$$

where σ_i is the sorting operation applied to dimension i, and $z^{(\sigma_i(j))}$ and $b^{(\sigma_i(j))}$ are the ranked j^{th} values of the sets $\mathcal{D}_i$ and $\mathcal{F}_i$, respectively.

Computational and Sample Complexity: The cost of solving the proposed Wasserstein distance estimate is $O(Nmlog(Nm))$. Since m is fixed and smaller than N, this is an order of magnitude faster than the high-dimensional OT's computational cost of $O(N^{2.5}\log(Nm))$. Now, we show that the proposed L_G calculation is a valid distance measure and its sample complexity is polynomial. Similar to the work in [22], we introduce the following definition of *generalizability* of a distributional distance:

Definition 1. *1.[**Generalizability**] Consider a family of distributions $\mathcal{P}$ over $\mathbb{R}^m$. A distance $\mathbf{d}(.)$ is said to be $\mathcal{P}$-generalizable if there exists a polynomial g such that for any two distributions $\mu, \nu \in \mathcal{P}$, and their empirical data samples $\hat{\mu}, \hat{\nu}$ with size $n = g(m, 1/\epsilon), \epsilon > 0$, the following holds:*

$$|\mathbf{d}(\mu, \nu) - \mathbf{d}(\hat{\mu}, \hat{\nu})| \leq \epsilon \quad w.p. \geq 1 - polynomial(-n).$$

Give the generalizability definition, we can prove the following result:

Theorem 1. *The proposed Wasserstein-p calculation, $\tilde{W}(\mathcal{D}, \mathcal{F})$, is a valid distance and it is $\mathcal{P}$-generalizable.*

Proof. Consider the SWD, $\hat{W}$, defined in Equation (5.4) where the m projection directions, instead of being randomly sampled, are chosen as follows. Let $\omega_k \in R^m$, be the column-k vector of the identity matrix $I \in R^{m \times m}$. We can see that $\mathcal{D}\omega_k$ and $\mathcal{F}\omega_k$ are projections of the data matrices onto the direction along the k-th basis. It is easy to see that this formulation of $\hat{W}$ is equivalent to $\tilde{W}$, defined in Equation (5.6). Therefore, similar to $\hat{W}$, $\tilde{W}$ is a valid distance metric. Moreover, since $\hat{W}$ is $\mathcal{P}$-generalizable (Claim 1 in [22]), $\tilde{W}$ is also $\mathcal{P}$-generalizable. That is, $\tilde{W}$ has a polynomial sample complexity. □

Unlike SWD which employs a large number of random directions, the proposed estimate uses the directions that best separate the generated b and the real data z. Under Theorem 1 and similar to SWD, estimating the Wasserstein distance from the m one-dimensional projections has a polynomial sample complexity. This is an important advantage over the OT estimation, which has an exponential sample complexity. The proposed estimate is also related to Max-SWD [22]. Max-SWD finds the single best projected dimension that best describes the separation of the two samples, by employing the discriminator that classifies real and fake data points. This results in the problematic minimax game. Our proposed

Algorithm 3 DCW-AE Model Training

Input: Training data X,
Discrete, latent code size m,
Number of training iterations K.
Number of reconstruction steps per one adversarial matching step l.
Learning rate η.
Output: $\{W_f\}$ parameters of the encoder
for *number of training iterations* K **do**
  for *number of reconstruction steps* l **do**
    Sample a minibatch of N examples $\{x^{(1)}_{pixel}, ..., x^{(N)}_{pixel}\}$.
    Compute the feature vectors $x^{(j)} = h(x^{(j)}_{pixel})$ for $j = \{1, ..., N\}$.
    Compute $L_A = ||g(f(x)) - x||_2^2$.
    Update $W_f \leftarrow W_f - \eta \nabla_{W_f} L_A$.
    Update $W_g \leftarrow W_g - \eta \nabla_{W_g} L_A$.
  end
  Sample a minibatch of N examples $\{x^{(1)}_{pixel}, ..., x^{(N)}_{pixel}\}$.
  Compute the feature vectors $x^{(j)} = h(x^{(j)}_{pixel})$ for $j = \{1, ..., N\}$.
  Sample N vectors $\{z^{(1)}, ..., z^{(N)}\}$ where $z^{(i)} \sim P_z$.
  Compute L_G
  for *each dimension* i **do**
    $L_G(i) = \hat{W}(\{z_i^{(1)}, z_i^{(2)}, ..., z_i^{(N)}\}, \{b_i^{(1)}, b_i^{(2)}, ..., b_i^{(N)}\})$
  end
  Set $L_G = \frac{1}{m} \sum_i^m L_G(i)$
  Update $W_f \leftarrow W_f - \eta \nabla_{W_g} L_G$.
end

calculation can estimate the similar averaged distance without the discriminator. For example, in Figure 5.2b, we can show that, given the optimal discriminator, Max-SWD finds the single direction whose distance would be the average of the distances along the 0°-direction and 90°-direction. Our approach will also estimate the same average, but without using the discriminator.

We call the adversarial autoencoder with the proposed loss calculation **D**iscrete **C**omponent-wise **W**asserstein **A**uto**E**ncoder (DCW-AE). The objective function of the DCW-AE can be written as follows:

$$L = L_A + L_G \tag{5.7}$$

We summarize the training algorithm of DCW-AE in Algorithm 3. While we can have a single minimization step on L, we find that alternatively minimizing L_A and L_G works better in practice (similar to Alternating Least Squares approach traditionally used in matrix factorization). Specifically, we minimize L_A for a few steps l on every minimization step of L_G. In all of our experiments, we set $l = 5$. Note that this is not a minimax game because L_A and L_G are different losses and are not related through a divergence or a value function, as in WGAN [2] and Jensen-Shannon GAN [52].

5.4 Experiments

In this section, we present the experimental results to demonstrate the effectiveness of the proposed hashing method over the existing hashing adversarial autoencoders and other hashing methods.

5.4.1 Datasets Used

We utilize the following datasets in our performance evaluation experiments:

- **MNIST** [1]: This dataset consists of of 70,000 digit images. We randomly select 10,000 images as the query set and the remaining images for the training and retrieval sets.
- **CIFAR10** [77]: This dataset consists of 60,000 natural images categorized uniformly into 10 labels. We randomly select 1,000 images from each label for the query set and use the remaining images for the training and retrieval sets. Hence, the query set contains 10,000 images and the training/retrieval set contains the same 50,000 images.
- **FLICKR25K** [11]: This dataset consists of 25,000 social photographic images downloaded from Flickr [2]. There are a total of 250 different class labels. We randomly select 20 images from each label for the query set and similarly use the remaining images for the training and retrieval sets. The final query dataset contains 5,000 images and the training/retrieval set contains the same 20,000 images.
- **PLACE365** [3]: This dataset consists of 1.8 millions of scenery images organized into 365 categories (labels). We randomly select 10 images from each label for the query set and 500 images from each label for the trainin and retrieval sets. The final query dataset contains 3,650 images and the training/retrieval set contains 182,500 images.

[1] http://yann.lecun.com/exdb/mnist/
[2] https://www.flickr.com/
[3] http://places2.csail.mit.edu/

5.4.2 Evaluation Metrics

For evaluating the performance of the proposed model, we follow the standard evaluation mechanism that is widely accepted for the problem of image hashing - the **precision@R** (**P@R**) and **mean average precision** (**MAP**). Given the query images, P@R and MAP are calculated as follows:

$$\text{Precision}(R, q) = \frac{\sum_{r=1}^{R} \delta(r, q)}{R} \tag{5.8}$$

$$\text{P@}R = \frac{1}{Q} \sum_{q=1}^{Q} \text{Precision}(R, q) \tag{5.9}$$

$$\text{AP}(q) = \frac{1}{N_q} \sum_{r=1}^{N} \text{Precision}(r, q) \times \delta(r, q) \tag{5.10}$$

$$\text{MAP} = \frac{1}{Q} \sum_{q=1}^{Q} AP(q), \tag{5.11}$$

where N is the size of the retrieval set, R is the number of retrieved images, N_q is the number of all relevant images in this set, Q is the size of the query set and $\delta(r, q) = 1$ only when the r-th retrieved image is relevant to the query image q; otherwise $\delta(r, q) = 0$. A retrieved image is relevant if its ground-truth label is the same as the label of the query image.

5.4.3 Comparison Methods

We compare the performance of the proposed method with various representative unsupervised image hashing methods.

- *Locality Sensitive Hashing* (**LSH**) [82]: a widely-used data-independent, shallow hashing method using random projection.
- *Spectral Hashing* (**SpecHash**) [138]: an unsupervised shallow hashing method whose goal is to preserve locality while finding balanced, uncorrelated hashes by solving the Eigenvector problem.
- *Iterative Quantization* (**ITQ**) [50]: the state-of-the-art shallow hashing method that alternately minimizes the quantization error to achieve better hash codes.
- *Stochastic Generative Hashing* (**SGH**) [21]: a representative hashing method that, similar to the proposed method, also minimizes the reconstruction loss in an autoencoder model.

Table 5.2: Performance comparison of different methods using P@1000. The best P@1000 value for each experiment is in bold.

Method	**MNIST**		**CIFAR10**		**FLICKR**		**PLACE**	
	32b	**64b**	**32b**	**64b**	**32b**	**64b**	**32b**	**64b**
LSH	0.314	0.430	0.172	0.173	0.013	0.018	0.005	0.010
SpecHash	0.441	0.504	0.199	0.198	0.021	0.023	0.007	0.007
ITQ	0.598	0.608	0.242	0.258	0.026	0.030	0.008	0.009
SGH	0.471	0.522	0.167	0.180	0.013	0.013	0.006	0.006
SSDH	0.441	0.469	0.221	0.176	0.026	0.027	0.014	0.016
DistillHash	0.481	0.491	0.231	0.233	0.026	0.028	0.015	0.016
WGAN-AE.	0.596	0.597	0.194	0.217	0.021	0.026	0.015	0.017
OT-AE	0.608	0.611	0.237	0.240	0.022	0.027	0.014	0.018
DCW-AE	**0.645**	**0.654**	**0.269**	**0.274**	**0.028**	**0.033**	**0.020**	**0.025**

- *Semantic Structure-based Deep Hashing* (**SSDH**) [149]: an unsupervised deep hashing method that learns the hash function by preserving heuristically-defined semantic structure of the data. The semantic structure is extracted from a pre-trained neural network (such as VGGNet).
- *Deep Hashing by Distilling Data Pairs* (**DistillHash**) [149]: the state-of-the-art unsupervised deep hashing method that is, in principle, similar to SSDH. However, the semantic structure is constructed by distilling data pairs that are consistent with the Bayes optimal classifier.
- *Wasserstein Adversarial Autoencoder* (**WGAN-AE**): adversarial autoencoder model for hashing which employs the critic that estimates the Wasserstein from the dual domain. This is an improved version of the adversarial autoencoder defined in [30].
- *OT-Wasserstain Adversarial Autoencoder* (**OT-AE**): the adversarial autoencoder model for hashing which directly minimizes the Wasserstein distance using the OT formulation in the primal domain.
- **DCW-AE**, which is the proposed method.

Table 5.3: Performance comparison of different methods using MAPs. The best MAP values are shown in bold.

Method	MNIST		CIFAR10		FLICKR		PLACE	
	32b	**64b**	**32b**	**64b**	**32b**	**64b**	**32b**	**64b**
LSH	0.259	0.261	0.139	0.1477	0.033	0.034	0.009	0.010
SpecHash	0.380	0.393	0.126	0.126	0.053	0.053	0.015	0.012
ITQ	0.542	0.543	0.180	0.182	0.053	0.053	0.010	0.016
SGH	0.414	0.420	0.128	0.137	0.019	0.019	0.004	0.005
SSDH	0.390	0.397	0.160	0.166	0.049	0.050	0.010	0.010
DistillHash	0.421	0.439	0.178	0.183	0.056	0.057	0.015	0.017
WGAN-AE	0.537	0.541	0.171	0.177	0.062	0.064	0.015	0.016
OT-AE	0.553	0.562	0.171	0.177	0.064	0.065	0.015	0.016
DCW-AE	**0.589**	**0.601**	**0.194**	**0.208**	**0.079**	**0.079**	**0.017**	**0.018**

Table 5.4: Additional performance results for $m = 128$ bits.

Method	MNIST		CIFAR10		FLICKR		PLACE	
	P@1000	MAP	P@1000	MAP	P@1000	MAP	P@1000	MAP
LSH	0.441	0.264	0.216	0.157	0.022	0.035	0.008	0.013
SpecHash	0.522	0.401	0.193	0.127	0.022	0.053	0.007	0.016
ITQ	0.611	0.549	0.270	0.190	0.030	0.053	0.010	0.012
SGH	0.611	0.431	0.188	0.139	0.014	0.020	0.013	0.006
SSDH	0.503	0.396	0.176	0.171	0.028	0.052	0.028	0.011
DistillHash	0.536	0.437	0.250	0.185	0.015	0.059	0.028	0.018
WGAN-AE	0.603	0.547	0.228	0.179	0.028	0.064	0.018	0.016
OT-AE	0.622	0.572	0.239	0.179	0.029	0.066	0.024	0.016
DCW-AE	**0.669**	**0.609**	**0.296**	**0.211**	**0.036**	**0.080**	**0.030**	**0.019**

5.4.4 Performance Results

In this experiment, we measure the performance of various methods proposed in the image hashing domain for the image retrieval task. Table 5.2 shows the P@1000 results across different lengths of the hash codes. DCW-AE consistently outperforms all the baseline methods at different lengths of the hash codes. Specifically, DCW-AE has a relative performance improvement of more than 10% in CIFAR10 and FLICKR. Similarly, in Table 5.3, we report the MAP results for all the methods for $m = 32$ and $m = 64$. Again, DCW-AE consistently achieves the best MAP results. Additional performance results for both P@1000 and MAP when $m = 128$ bits are presented in Table 5.4. The improvements of our method over the baselines are statistically significant according to the corresponding paired t-tests (p-value < 0.01).

The P@1000 and MAP results demonstrate the superiority of our method compared to various state-of-the-art approaches for the image hashing problem. One important result is the improvement in performance of DCW-AE compared to OT-AE. This supports our claim that the existing adversarial autoencoders cannot learn the optimal hash function compared

Table 5.5: MAP performance of various Wasserstein losses used in Adversarial Autoencoders. Starting from the based autoencoder (AE), different formulations of the Wasserstein Distance (WD) is added to regularize the AE, resulting in different versions of the adversarial AE. DCW denotes the proposed Discrete Component-Wise formulation.

Method	**CIFAR10**		**FLICKR**	
	32b	**64b**	**32b**	**64b**
AE	0.148	0.151	0.048	0.054
DualWD+AE	0.170	0.177	0.060	0.064
OT+AE	0.171	0.177	0.064	0.065
SWD+AE	0.179	0.180	0.067	0.068
DCW+AE	**0.192**	**0.208**	**0.074**	**0.079**

to our DCW-AE model. This demonstrates the significance of a generalizable Wasserstein estimate. A lower sample-complexity estimate makes it easier for the algorithm to converge to the target distribution using mini-batch training algorithms.

5.4.5 Analysis of Wasserstein Losses

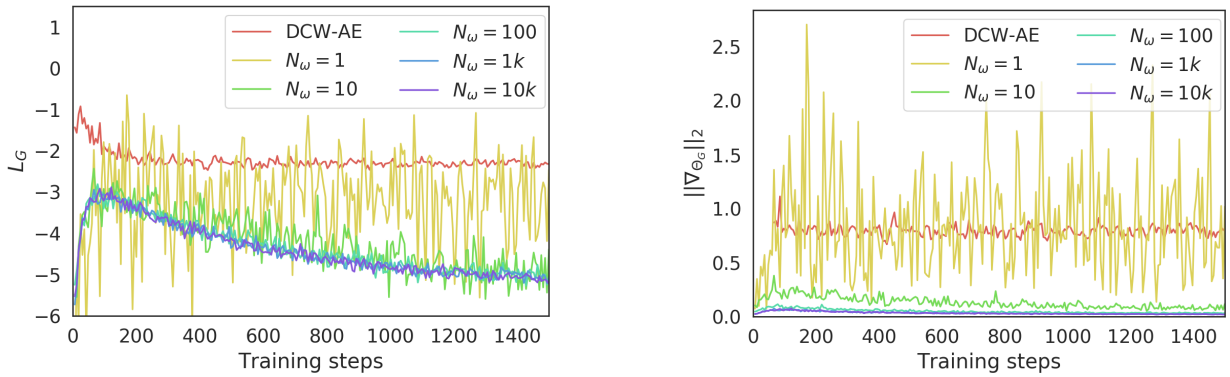
In this section, we examine the influence of different formulations of the Wasserstein Distance on the Adversarial Autoencoder with respect to the retrieval performance. Table 5.5 shows the Precision-Recall curves of the different Autoencoder-based hashing models. AE denotes the vanilla Autoencoder without any adversarial regularization. SWD+AE is the Adversarial Autoencoder whose adversarial matching cost is SWD. We observe that all adversarial-based Autoencoder models outperform AE. This demonstrates the importance of the adversarial learning for Autoencoders in image hashing. Furthermore, replacing the dual Wasserstein Distance (in DualWD+AE) and the OT estimate (in OT+AE) with our proposed estimate (DCW) further improves the retrieval performance. DCW+AE's performance is significantly better than that of OT-AE and SWD-AE. We hypothesize that this improvement is primarily due to the following reasons:

- The proposed distance estimate of DCW-AE has a better sample complexity (i.e., more generalizable) than OT. Better sample complexity allows mini-batch optimization techniques such as SGD to converge better to the target distribution.
- The proposed distance estimate of DCW-AE is a better distance compared to SWD. Our distance estimate induces a weaker topology than SWD [2]. In other words, our estimate makes it easier to converge to the target distribution.

(a) ITQ (b) DistillHash (c) OT-AE (d) DCW-AE

Figure 5.3: T-SNE embedding of the generated discrete hash codes on the MNIST dataset. The digit identities are color-coded.

5.4.6 Qualitative Evaluation



(a) Distance Estimates (b) Gradient Norm

Figure 5.4: Wasserstein distance estimates (left) and the gradient norm of the encoder's parameters (right) during training of DCW-AE and SWD (with varying N_ω values).

Qualitatively, we can visually compare the quality of the hash codes generated by DCW-AE, OT-AE and two best non-adversarial hashing baselines, ITQ and DistillHash. Figure 5.3 shows the two-dimensional t-SNE embeddings [95] of the generated hash codes on the query set. In this example, the similarity matrix of SSDH is constructed directly from the image-pixel space. Notice that SSDH generates unreliable hash codes. This shows that, without a reliable construction of the similarity matrix, the retrieval performance of both SSDH and DistillHash is significantly deteriorated. On the other hand, DCW-AE learns a very efficient discrete embedding of the original data; for example, DCW-AE can even separate the most similar digits 9 and 4.

5.4.7 DCW-AE's Wasserstein Estimation

In this Section, we show the efficiency of the proposed Wassertein estimate and compare it with SWD. Figure 5.4 shows the distance estimates and the gradient norm of the parameters ($||W_f||_2$) during training of the proposed calculation in DCW-AE and of the original SWD on the CIFAR-10 dataset. SWD is estimated with different number of random projections N_ω. On the extreme case, when $N_\omega = 1$, both the distance and the gradient fluctuates significantly during training. This makes adversarial training become very unstable. When N_ω is higher, we can observe that the distance estimates of SWD are lower. This is because the one-dimensional Wasserstein distances across several random directions are very small and do not contain useful signal for training (see the corresponding gradient). Even when increasing N_ω from 10 to 10K, the estimate is not generally better. On the other hand, the estimates of DCW-AE are higher and its gradient is more stable. This is due to the fact that the proposed distance estimate averages the distances of directions along which the P_z and P_b are most dissimilar.

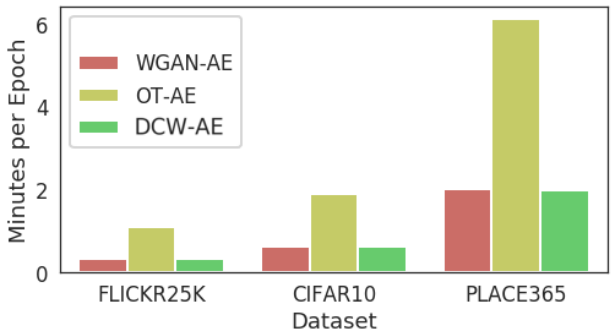


Figure 5.5: Comparison of training times (in minutes) per one epoch (using a mini-batch size of 128 examples).

5.4.8 Computational efficiency of DCW-AE

In this experiment, we compare the training time of DCW-AE and the related adversarial hashing methods. The training time of WGAN-AE, OT-AE and DCW-AE are shown in Figure 5.5 for three datasets, CIFAR10, FLICKR and PLACE. We report the average training time per epoch. In Figure 5.5, the training time of DCW-AE is significantly reduced compared to the training time of OT-AE.

5.4.9 Sampled queries

In Table 5.6, we show the top-10 retrieved digit images of two query images corresponding to digits 3 and 9. DCW-AE method has successfully retrieved relevant digits; when the retrieved images are false positives, we can still see that they contains similar appearances (e.g. some handwritten digit 4's are similar to the digit 9). As expected, when increasing the size of the binary code) (m), the model makes fewer mistakes.

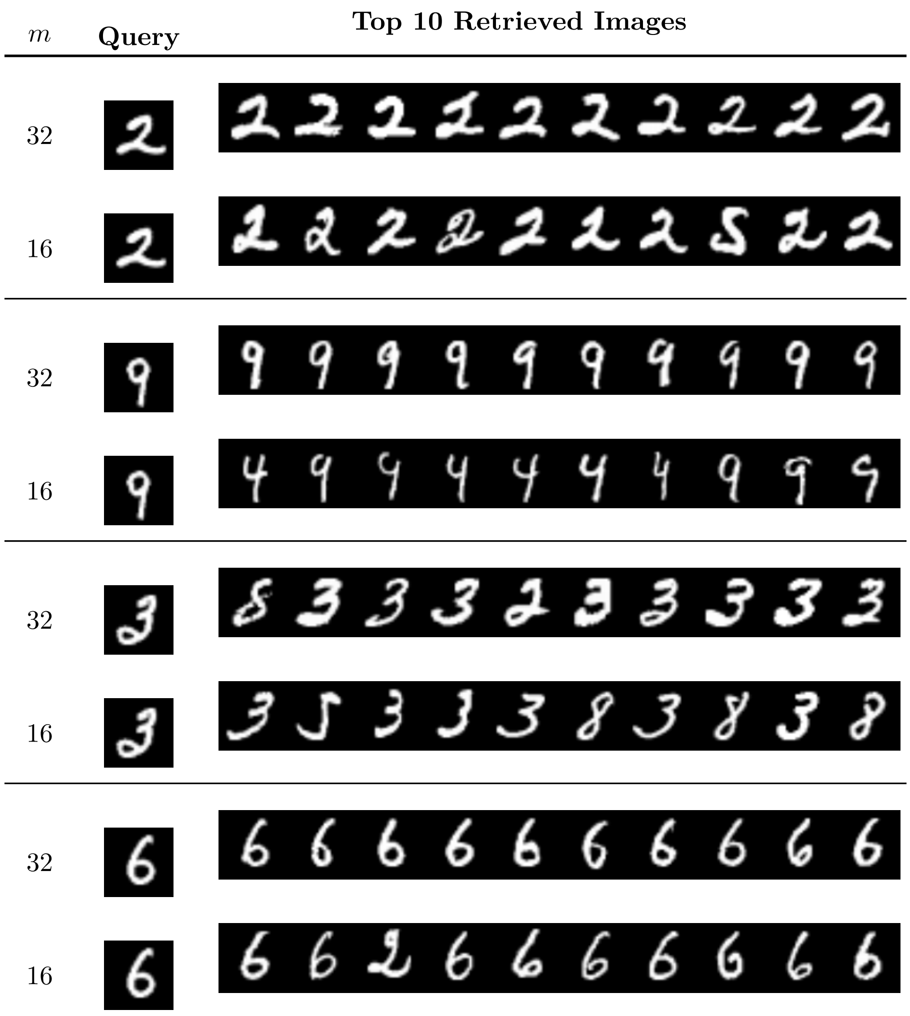


Table 5.6: An illustration of the top-10 retrieved MNIST digits for a given query image using code length (m) of 32 and 16 bits.

5.5 Summary

We proposed a novel adversarial autoencoder model for the image hashing problem. Our model learns hash codes that preserves the locality information in the original data by employing the adversarial autoencoder. By directly minimizing a novel variant of SWD, our model has a much better generalization property than the existing adversarial approaches, thus is able to achieve significant performance gains. Furthermore, the proposed model trains significantly faster than the existing primal-Wasserstein adversarial autoencoders. Our experiments validate that the proposed hashing method outperforms all the existing state-of-the-art image hashing methods. Our work makes one leap towards leveraging an efficient, robust adversarial autoencoder for the image hashing problem and we envision that our model will serve as a motivation for improving other adversarially-trained hashing models.

Chapter 6

Multipurpose Generative Hashing Network

In the previous chapters, we developed a suite of unsupervised, generative hashing models that efficiently learn hash functions without domain-specific heuristics to preserve the locality information and additional hyperparameters of the regularizers to minimize the gap between continuous and discrete optimizations. The recent development of generative models, on the other hand, shows an impressive data generation quality, especially in the computer vision domain. Such data synthesizing ability can be useful in improving the generalization of machine learning models. In this section, we consider the supervised case where a hashing method can exploit the semantic labels of samples in the dataset. We then propose a supervised, generative hashing model which is based on maximum-likelihood estimation of the data. In this so-called Energy-based Model (EBM) framework, the density function of the data is modeled as the Gibbs distribution of the energy function. Since we focus our efforts on developing robust and efficient hash-function learning approaches, we will focus on discussing the advantages of the proposed generative hashing approach, specifically on efficient training and robustness to corrupted and out-of-distribution data, compared to other popular models such as GANs.

6.1 Introduction

Searching for similar items (such as images) is an important yet challenging problem in this digital world. Accurate retrieval within a constrained response time is crucial, especially in large databases with several millions of images. This motivates the need for approximate nearest-neighbor (ANN) methods instead of using an intractable linear scan of all the images for such massive datasets. Hashing is a widely used ANN method with a principled retrieval approach for web-scale databases. In hashing, high-dimensional data points are

projected onto a much smaller locality-preserving *binary* space. Searching for similar images is reduced to searching for similar discrete vectors in this binary space using computationally efficient Hamming distance [82]. Searching for an item in the binary space is extremely fast because each Hamming distance calculation (e.g., for 64-bit binary space) only needs 1-CPU instruction in most modern hardware. Furthermore, the compact binary codes are storage-efficient, thus the entire index of items can be kept in fast-access memory; for example, a million 64-bit vectors only occupy approximately 8 megabytes. This chapter focuses on the learning-to-hash methods that "learn" hash functions for efficient image retrieval.

The mapping between the original image x and the k-bit discrete vectors is expressed through a hash function $f : x \to \{-1, 1\}^k$. Learning and deploying such a hash function in real-world applications face many challenges. First, the hash function should capture the similarity relationship between images in the binary space, for example, represented in the annotated similarity between items. However, with a massive amount of data, annotated similarity is scarce. This leads to a poor generalization in methods which exclusively rely on such annotated information. Furthermore, real-world data contains amendable missing information (e.g., a part of an image is corrupted during lossy compression or transmission between systems) and gradually changes over time (i.e., the underlying data distribution changes). A hash function which is not robust to such scenarios is not suitable for real-world applications because its expected retrieval performance will quickly degrade.

Several learning-to-hash methods have been proposed for efficient ANN search [15, 49, 79, 80, 120, 140, 159]. Supervised methods, which leverage the labeled similarity of the images in learning the hash functions, demonstrate a superior performance over unsupervised methods. However, subjected to the scarcity of supervised similarity information, the supervised methods run into problems such as overfitting and train/test distribution mismatch, resulting in a significant loss in retrieval performance. Recently, some methods employ generative models, specifically generative adversarial networks (GAN), to synthesize additional training data for improving the generalization of the learned hash functions. Nevertheless, these GAN-based methods do not take full advantage of generative models beyond synthetically generating the data. The main reason is that GAN is an implicit generative model that does not directly estimate the density function of the data. On the other hand, explicit generative models, specifically energy-based models (EBMs), are able to synthesize images and recover the missing information in those images through the inference of the EBMs. For example, when the EBM explicitly models the density of the data $p(x)$, we can recover the missing information in a data point x by revising x through the MCMC inference of the EBM in order to find the most probable version of x in the data distribution, thus effectively recovering x. Such ability of explicit generative models is extremely useful for real-world retrieval applications, especially when data loss or corruption can happen at any stage during the data collection or transmission process in these applications. By recovering the corrupted data, we hope to preserve much of the retrieval performance of the model.

In this chapter, we propose a unified energy-based generative hashing framework (GenHash) which simultaneously learns the representation of the images and the hash function. Our

hashing network consists of a shared representation network. This network learns shared representation of the images that are useful to solve multiple objectives. Each objective is modeled as a light-weight head (a multi-layer perceptron) on top of the shared network and solves a specific task. The tasks include: 1) an explicit joint probability density estimation of an image and its semantic labels (energy head), 2) a contrastive hash-function learning (hash head) and 3) semantic label prediction (classification head). Consequently, this multipurpose network simultaneously learns to represent the images from multiple perspectives and allows the training process to develop a shared set of features as opposed to developing them redundantly in separate networks such as the GAN-based methods. Finally, since our model only trains the EBM for data synthesis, it requires fewer model parameters than approaches that use multiple networks (e.g., a generator and discriminator in GAN-based approaches). The main contributions of the chapter are summarized below:

- We propose a unified generative, supervised learning-to-hash framework which takes complete advantage of generative energy-based models and enjoys better generalization and robustness towards missing data and out of distribution retrieval. The core component of this unified framework is the multi-headed or multipurpose hashing network, which combines density estimation (i.e., MCMC teaching process) and hash coding (i.e., contrastive loss).
- We propose a simple yet efficient training procedure to train the multipurpose hashing network. Specifically, we propagate the MCMC chains during training with two persistent contrastive divergence (PCD) buffers. One PCD buffer "explores" different modes of the model during training while the other PCD buffer is responsible for "exploiting" the learned modes to assist the contrastive hash function learning. The two PCD buffers jointly improve the efficiency of the MCMC teaching, thus allowing the training process to converge faster in practice.
- We demonstrate the advantages of our model over several state-of-the-art hashing techniques through an extensive set of experiments on various benchmark retrieval datasets.

The rest of the chapter is organized as follows. We discuss the related work in Section 6.2. In Section 6.3, we describe the details of the proposed generative hashing network. We present quantitative and qualitative experimental results in Section 6.4 and conclude our discussion in Section 6.5.

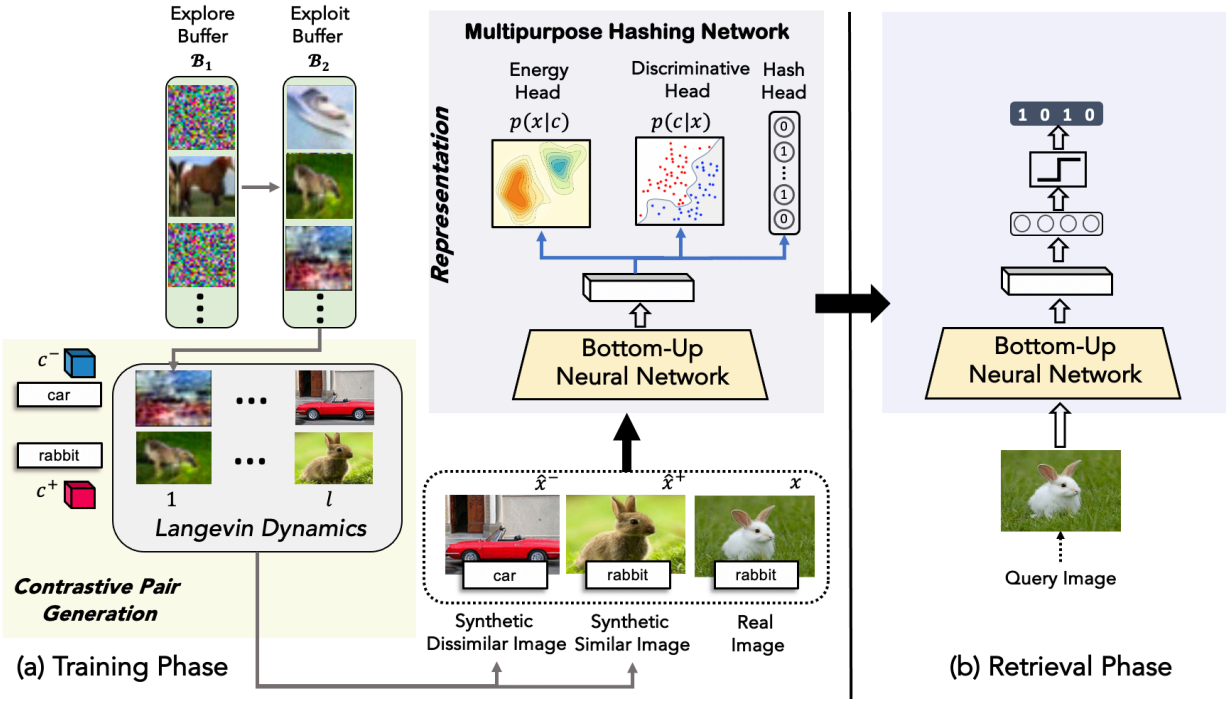


Figure 6.1: The main component of GenHash is a multipurpose hashing network (light blue block) that describes the images in multiple ways, including an explicit density model $p(x|c)$, a discriminative model $p(c|x)$, and a hashing model. All three models share a base bottom-up representational network. The multipurpose hashing network is trained by a loss including negative maximum likelihood, triplet-ranking loss, and classification loss. To compute the triplet-ranking loss, GenHash relies on the Contrastive Pair Generation process (light yellow block) that takes a label c^+ as input and synthesizes (i.e., samples from the replay buffer $\mathcal{B}_2$) a contrastive image pair $\{\hat{x}^+, \hat{x}^-\}$ from the same class c^+ and a different class c^-. In the retrieval phase, only the hashing computational path (Hash Head and shared representational network) is used; the binary hash codes are the signs of the real-valued outputs.

6.2 Related Works

In this section, we review the previous research works related to two topics, namely, image hashing and energy-based generative models.

6.2.1 Image Hashing

Learning to hash, and especially image hashing, has been heavily investigated in both theory and practice. With the advance of deep neural networks, a plethora of image hashing methods have been proposed. In general, these existing image hashing methods can be organized into two categories: shallow hashing and deep hashing. Shallow hashing methods learn linear hash functions and rely on carefully-constructed discriminative features which are extracted from any hand-crafted feature extraction techniques or any representation-learning algorithms. On the other hand, the deep hashing methods combine the feature representation learning phase and the hashing phase into an end-to-end model and have demonstrated significant performance improvements over the hand-crafted feature-based hashing approaches [15, 36, 55, 67, 80, 83, 140, 159, 160].

Some hashing methods leverage the labeled information of the images when they are available. The works in [56, 120] regress from the hash code of an image to its semantic label. Li et al. [55, 83] predict the class label of an image given its hash code. On the other hand, the works in [79, 140] preserve the consistency between the hash codes approximated from the similarity matrix and the hash codes approximated from the deep representation networks. A pairwise similarity objective or triplet ranking objective can also be formulated by randomly drawing the similar and dissimilar examples of an image from the dataset [80, 159]. Our work also models the relationship between the hash code of an image and its semantic label. However, the proposed GenHash method ensures that the hash codes of the synthetic samples are also consistent with their sampled labels. Furthermore, while the proposed contrastive hashing objective (in Section 6.3) has some similarity to other previously proposed triplet ranking loss, the contrastive samples (similar and dissimilar images) are synthetic (i.e., generated from a generative models) instead of being drawn from the same empirical datasets. The primary reason for these differences is to improve the generalization of the learned hash function.

Generative Supervised Hashing

Hashing methods can also be divided into unsupervised [14, 26, 45, 120, 140, 151] and supervised hashing [24, 25, 49, 62, 63, 85, 113, 138, 150]. Supervised methods demonstrate a superior performance over the unsupervised ones, but they can easily overfit when there are limited labeled data. To overcome such limitations of supervised hashing, some methods synthesize additional training data to improve the generalization of the hash functions [42, 106]. These methods employ the popular Generative Adversarial Network (GAN) to synthesize the contrastive images.

The use of generative models in hashing is currently limited to only data synthesis. Yet, generative models can benefit other downstream problems such as data imputation and out-of-distribution robustness. Our work belongs to the deep, supervised hashing category and we aim to jointly learn an energy-based generative model (EBM) and the hash function in an end-to-end manner. Borrowing the strengths of EBM, we not only improve the retrieval performance over the GAN-based approaches but also significantly improve robustness of the hash function in real-world problems in terms of handling missing data and out-of-distribution retrieval.

6.2.2 Energy-based Generative Models

Xie et al. [141] proposed a powerful generative model, called generative cooperative network (CoopNets), which is able to generate realistic image and video patterns. The CoopNets framework jointly trains an energy-based model (i.e., descriptor network) and a latent variable model (i.e., generator network) via a cooperative learning scheme, where the descriptor network is trained by MCMC-based maximum likelihood estimation [142], while the generator learns from the descriptor and serves as a fast initializer for the MCMC of the descriptor. Xie et al. [144] studied the conditional version of CoopNets for supervised image-to-image translation. Compared to GANs, the CoopNets framework has several advantages: First, the cooperative learning of two probabilistic models is based on maximum likelihood estimation, which generally does not suffer from GAN's mode collapse issue. Secondly, after the training is finished, GANs' bottom-up discriminator becomes invalid because it fails to distinguish the real and fake examples, and only the generator remains to be useful for the data synthesis purpose. In contrast, in the CoopNets framework, both bottom-up EBM and top-down generator are valid models for representation and generation. In practice, however, CoopNets still employ two separate networks which must be carefully designed together to ensure the model to converge into a good local minima [141]. This problem also exists in GANs.

Du et al. [92] proposed a scalable single-network EBM for the image generation task. The EBM is able to generate a realistic image and exhibits attractive properties of EBM such as out-of-distribution and adversarial robustness. Grathwohl et al. [53] reinterpreted the

discriminative classification task, which estimate the conditional probability $p(x|y)$, with an energy-based model for the joint probability $p(x, y)$. To estimate the intractable partition function, the authors use MCMC sampling through the Langevin dynamics. Specifically, they build upon the persistent contrastive divergence (PCD) [127] and maintain a replay buffer to propagate the MCMC chains during training. This allows shorter mixing times than initialization of the chains from random noise, while occasionally re-initializing samples from random noise in the buffer allows the training to explore different modes of the model.

Our work studies generative hashing based on the framework of a single EBM with multi-purpose objectives. However, as we shall see later, the current PCD training procedure of existing EBM works does not work well for contrastive hash function learning where the loss function involves data synthesis of similar and dissimilar examples. Instead, we propose to train the EBMs by mixing between an exploration buffer and an exploitation buffer.

6.3 Multipurpose Generative Hashing Network

The proposed Multipurpose Generative Hashing Network consists of a shared representation network and multiple light-weight heads. They are jointly trained by an MCMC-based learning algorithm. Figure 6.1 provides an overview of our hashing network.

6.3.1 Problem Statement

Given a dataset $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ of n images, the goal of a hashing method is to learn a discrete-output, nonlinear mapping function $\mathcal{H} : x \to \{-1, 1\}^K$, which encodes each image x into a K-bit binary vector such that the similarity structure between the images is preserved in the discrete space. In the supervised hashing setting, each example $x_i \in \mathcal{X}$ is associated with a label c_i. Note that this is a point-wise label of an image. Another common supervised scenario has the pairwise similarity label for each pair of images. However, for most image applications, pair-wise labeling is significantly labor-intensive because a dataset of n images requires n^2 pairwise labelings.

6.3.2 Multipurpose Energy-based Model

The multipurpose EBM aims at representing the images from different perspectives. We propose to parameterize this network by a multi-headed bottom-up neural network, where each branch accounts for one different representation of the image. The proposed network assembles three types of representational models of data in a single network in the sense that all models share a base network but have separate lightweight heads built on top of the base network for different representational purposes. Let $f_0(x;\theta_0)$ be the shared base network with parameters θ_0. In the following sections, we will describe the purpose of each head in more detail.

Conditional Energy head

The energy head h_E along with the base network f_0 specifies an energy function $f_E(x,c;\Theta_E)$, where observed image-label pairs (which come from the real data distribution) are assigned lower energy values than unobserved ones. For notational simplicity, let the parameters be $\Theta_E = (\theta_0, \theta_E)$ and the energy function be $f_E(x,c;\Theta_E) = h_E(c, f_0(x,\theta_0);\theta_E)$. With the energy function f_E, the energy head explicitly defines a probability distribution of x given its label c in the form of an energy-based model as follows:

$$p(x|c;\Theta_E) = \frac{p(x,c;\Theta_E)}{\int p(x,c;\Theta_E)dx} = \frac{\exp[-f_E(x,c;\Theta_E)]}{Z(c;\Theta_E)}, \tag{6.1}$$

where $Z(c;\Theta_E) = \int \exp[-f_E(x,c;\Theta_E)]dx$ is the intractable normalizing constant. Eq. (6.1) is also called generative modeling of neural network f_E [142]. Specifically, fixing the label c, $f_E(x,c;\Theta_E)$ defines the value of the compatible solution x and $-f_E(x,c;\Theta_E)$ defines the conditional energy function. Note that, for each value c, there are many compatible solutions x, i.e., there are several x's with similar, low conditional energies.

The training of θ_E in this context can be achieved by maximum likelihood estimation, which will lead to the "analysis by synthesis" algorithm [54]. Given a set of training images with labels $\{(c_i, x_i)\}_{i=1}^n$, we train Θ_E by minimizing the negative log-likelihood (NLL):

$$\mathcal{L}_{\mathrm{E}}(\Theta_E) = -\frac{1}{n}\sum_{i=1}^{n} \log p(x_i|c_i;\Theta_E), \tag{6.2}$$

The gradient of the above loss function is given by

$$\frac{1}{n}\sum_{i=1}^{n}\left\{\mathbb{E}_{p(x|c_i;\Theta_E)}\left[\frac{\partial f_E(x,c_i;\Theta_E)}{\partial \Theta_E}\right] - \frac{\partial f_E(x_i,c_i;\Theta_E)}{\partial \Theta_E}\right\}, \tag{6.3}$$

where the $\mathbb{E}_{p(x|c_i;\Theta_E)}$ denotes the intractable expectation with respect to $p(x|c_i;\Theta_E)$.

Following the works in [53, 92], we use persistent contrastive divergence (PCD) [127] to estimate the intractable expectation since it only requires short-run MCMC chains. This gives an order of magnitude savings in computation compared to initializing new chains with long mixing time at each iteration. Intuitively, the PCD supplies the learning process with initial solutions from a replay buffer of past generated samples. The learning process then refines these solutions at high value region around a mode of the objective function. The model's parameters are then updated so that the objective function shifts its high value region around the mode towards the observed solution. In the next iteration, the refined solution will (hopefully) get closer to the observed solution.

The MCMC sampling strategy with the replay buffer can be summarized in two steps: (i) the algorithm first samples $\hat{x}$ from a replay buffer $\mathcal{B}_1$ with a probability $p_{\mathcal{B}_1}$ and from uniform noise with a probability $1 - p_{\mathcal{B}_1}$, and then (ii) it refines $\hat{x}$ by finite steps of Langevin updates [162], which is an example of MCMC, to obtain final $\tilde{x}$, as follows:

$$\tilde{x}_{t+1} = \tilde{x}_t - \frac{\delta^2}{2}\frac{\partial f(\tilde{x}_t, c; \Theta_E)}{\partial \tilde{x}} + \delta \mathcal{N}(0, I_D), \tilde{x}_0 = \hat{x}, \tag{6.4}$$

where t indexes the Langevin time steps, and δ is the step size. The Langevin dynamics in Eq. (6.4) is a gradient-based MCMC, which is equivalent to a stochastic gradient descent algorithm that seeks to find the minimum of the objective function defined by $f(x, c; \Theta_E)$. The replay buffer $\mathcal{B}_1$ stores past generated samples. Occasionally re-sampling from random uniform noise is crucial to the learning process since different modes of the model can be explored in training. On the other hand, between the parameters' update steps, the model only slightly changes, thus sampling from past samples, which should be reasonably close to the model distribution, allows the algorithm to simulate longer MCMC chains on the samples. However, we call $\mathcal{B}_1$ an explore buffer because of its primary function, which is to seek and cover possible modes of the model.

With the MCMC examples, we can compute the gradient of the negative log-likelihood objective by

$$\nabla(\Theta_E) \approx \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\partial f_E(\tilde{x}_i, c_i; \Theta_E)}{\partial \Theta_E} - \frac{\partial f_E(x_i, c_i; \Theta_E)}{\partial \Theta_E}\right]. \tag{6.5}$$

Contrastive Hashing head

The head h_H learns to represent the input images as binary hash codes. The hash head h_H and the base network f_0 form a hash function $f_H(x; \Theta_H) = h_H(f_0(x; \theta_0); \theta_H)$, where $\Theta_H = (\theta_0, \theta_H)$. The hash function aims at mapping images with similar high-level concepts to similar hash codes and those of unrelated concepts to dissimilar codes. With a generative model, one effective way to learn such hash function is: for each image x, we "draw" a positive sample x^+ which is conceptually similar to x and a negative sample x^- which is

conceptually dissimilar to x, and train the hash function to produce similar hash codes for x and x^+, and dissimilar hash codes for x and x^-.

Such contrastive learning can be achieved by recruiting labeled generated samples from the inference of the conditional EBM. These synthetic samples from each class can be similarly generated using MCMC sampling. To avoid long mixing time where the MCMC chains are initialized from random noise, we can reuse past generated samples from the replay buffer $\mathcal{B}_1$. However, $\mathcal{B}_1$ may contain several past samples that have been less rigorously refined through the Langevin dynamics (i.e., only refined for a few times). These "young" samples can still be closer to random noise. Therefore, when being selected for contrastive hash learning, there is not useful difference between samples from a different class. That is, the current sample x and their similar and dissimilar synthetic samples x^+ and x^-, respectively, do not form an informative contrastive triplet. In our experiments, we observe that only relying on the explore buffer $\mathcal{B}_1$ for contrastive samples learn a hash function whose performance is significantly worse than desired.

This problem is further illustrated in Figure 6.2, where some samples from each class are similar to random noise. There are more of these samples in CIFAR10 than in MNIST because the generation of natural images in CIFAR10 requires a significantly more complex model than the generative model of MNIST. Even further into training, we observe that there are still similar MCMC samples due to the occasional sampling from random noise which requires longer MCMC chains when the data distribution is complex (as in the case of CIFAR10). One naive solution is to increase the number of Langevin steps; however, larger chains make the EBM significantly more computationally expensive to train.

We propose a simple, yet effective sampling strategy to solve this problem. First, we introduce a second replay buffer $\mathcal{B}_2$, where a sample $x \in \mathcal{B}_2$ is required to be initialized from supposedly longer MCMC chains. To avoid explicitly increasing the number of Langevin steps to achieve such longer MCMC chains, we leverage the existing replay buffer $\mathcal{B}_1$ by "copying" samples that have been revised several times in $\mathcal{B}_1$. Intuitively, while sampling from $\mathcal{B}_1$ allows the training process to explore different modes of the models, sampling from $\mathcal{B}_2$ exploits the learned modes. However, since we regularly copy the samples from $\mathcal{B}_1$ to $\mathcal{B}_2$ when they are "matured", sampling from $\mathcal{B}_2$ also explores all previously learned modes in for the contrastive hash learning.

Under this sampling strategy, to learn the contrastive hash function, for each observed image x and its label c, we sample a synthetic image x^+, conditioned on the label c, and a synthetic image x^-, conditioned on a different label $c^- \neq c$ using the replay buffer $\mathcal{B}_2$. The three examples form a real-synthetic triplet (x, x^+, x^-). The hash function f_H can be trained to minimize the Hamming distance (a discrete distance function which is typically approximated by the continuous L_2 distance) between $f_H(x)$ and $f_H(x^+)$ and maximize the distance between $f_H(x)$ and $f_H(x^-)$. This triplet-ranking loss is defined as follows:

Algorithm 4 Learning of GenHash

Input:
(1) Training images with labels $\{(x_i, c_i), i = 1, ..., n\}$
(2) Numbers of Langevin steps l
(3) Number of learning iterations T
(4) Learning rate γ
(5) Replay buffer sizes $|\mathcal{B}_1|$ and $|\mathcal{B}_2|$
(6) Number of mature Langevin steps k
Output: Learning parameters of the hash function Θ_H

Initialize $\Theta = (\Theta_E, \Theta_H, \Theta_C)$.
Let $t \leftarrow 0$.
Let $\mathcal{B}_1 \leftarrow \{(x_i, c_i) : x_i \sim \text{uniform}(0, 1), c_i = i \bmod |C|, i = 1, ..., |\mathcal{B}_1|\}$

for $t \leftarrow 1$ **to** T **do**

Sample real images: Randomly draw a n-sample mini-batch $\{(x_i, c_i) : i = 1, ..., n\}$.

Sample synthetic images $\mathcal{B}_1$: Randomly draws $(\hat{x}_i, c_i) \sim \mathcal{B}_1$ with 95% probability and $\hat{x}_i \sim \text{uniform}(0, 1)$ otherwise.

Sample synthetic contrastive image pair from $\mathcal{B}_2$: For $i = 1, ..., n$, sample a dissimilar label c_i^- such that $c_i^- \neq c_i$, then sample a similar image $\hat{x}_i^+$ with label c_i such that $(\hat{x}_i^+, c_i) \in \mathcal{B}_2$ and a dissimilar image $\hat{x}_i^-$ with label c_i^- such that $(\hat{x}_i^-, c_i^-) \in \mathcal{B}_2$ to form a triplet $(x_i, \hat{x}_i^+, \hat{x}_i^-)$.

Refine synthetic images by the descriptor: For $i = 1, ..., n$, starting from $\hat{x}$, $\hat{x}_i^+$ and $\hat{x}_i^-$, run l steps of Langevin dynamics to obtain the refined images $\tilde{x}_i$, $\tilde{x}_i^+$ and $\tilde{x}_i^-$, respectively, each step following Eq. (6.4).

Update descriptor: With the observed and the synthetic examples, we update $\Theta^{(t+1)} = \Theta^{(t)} - \gamma_D \mathcal{L}'_{\text{Des}}(\Theta^{(t)})$, where $\mathcal{L}_{\text{Des}}(\Theta^{(t)})$ is defined in Eq. (6.9)

Update the explore buffer: $\mathcal{B}_1 \leftarrow \mathcal{B}_1 \cup \{(\tilde{x}_i, c_i), i = 1, ..., n\}$.

Update the exploit buffer: $\mathcal{B}_2 \leftarrow \mathcal{B}_2 \cup \{(x, c)\}$ such that $(x, c) \in \mathcal{B}_1$ and x_j has been revised for more than k steps, or (x, c) are contrastive samples, that is $(x, c) = (\tilde{x}_i^-, c_i^+)$ or $(x, c) = (\tilde{x}_i^+, c_i^-)$.

end

(a) MNIST (b) CIFAR10

Figure 6.2: Samples from the explore buffer $\mathcal{B}_1$. Images on each row are from the same semantic class.

$$\begin{aligned} &\mathcal{L}_{\mathrm{H}}(\Theta_H) = ||f_H(x) - f_H(x^+)||_H + max(m - ||f_H(x) - h(x^-)||_H, 0) \\ &s.t.\ \ f_H(x) \in \{-1, 1\}, f_H(x^+) \in \{-1, 1\}, f_H(x^-) \in \{-1, 1\} \end{aligned} \tag{6.6}$$

where $||.||_H$ denotes the Hamming distance. The first term in the objective function preserves the similarity between images with similar semantic concepts, while the second term penalizes the mapping of semantically dissimilar images to similar hash codes if their distance is within a margin m. Essentially, this is a contrastive objective which avoids collapsed solutions because it only considers the dissimilar pairs having distances within a certain margin to contribute to the loss.

The objective of $\mathcal{L}_h$ is a discrete optimization problem, hence it is computationally intractable to solve and is not suitable for a gradient-based backpropagation algorithm. A natural solution is to approximate the discrete constraints with real-valued output and replace the Hamming distance with Euclidean distance. For the thresholding procedure, a commonly-used trick is to employ the *tanh* or *sigmoid* function. However, we find that *tanh* or *sigmoid* makes the learning process more difficult to converge to good local optima. To overcome this, we propose to directly regularize the real-valued output of the hash function to the desired discrete values. The final triplet-ranking loss is as follows:

$$\begin{aligned} \mathcal{L}_{\mathrm{H}}(\Theta_H) =& ||f_H(x) - f_H(x^+)||_2 + \max(m - ||f_H(x) - f_H(x^-)||_2, 0) \\ &+ \lambda(|||f_H(x)| - 1||_2 + |||f_H(x^+)| - 1||_2 + |||f_H(x^-)| - 1||_2) \end{aligned} \tag{6.7}$$

where $f_H(.)$ is now the relaxed function with real-valued vector outputs and $|.|$ is element-wise absolute operation. The first and second terms in the objective function approximate the Hamming distances in Eq. (6.7). The last term minimizes the quantization error of approximating the discrete solution with the real-value relaxation. Intuitively, it centers the relaxed, continuous output of the hash function around the desired binary value of -1 or 1. Note that the max operation is non-differentiable; however, we can define the subgradient of the max function to be 1 at the non-differential points.

Discriminative head

Image labels provide not only knowledge for training a classification model but also a supervised signal for extracting high-level information of the images. On the other hand, the learned hash codes should also capture high-level abstractions of the images, therefore should be predictive of the image labels. This relationship can be modeled through a multi-class classification problem. Specifically, we propose a classification head h_C which predicts the class label of an image given its hash code. For each image x_i, let $\hat{c}_i$ be the predicted label. The multi-class classification loss can be defined as follows:

$$\mathcal{L}_C(\Theta_C) = \mathcal{L}_{\mathrm{C}}(x|c;\theta_C) = -\frac{e^{\theta_{c,:}^T h(x)}}{\sum_j e^{\theta_{j,:}^T h(x)}} \tag{6.8}$$

where $\theta_C \in \mathbb{R}^{K\times L}$ is the parameter of the linear layer which maps each hash code into the class labels. We additionally denote $\Theta_C = (\theta_0, \theta_C)$ as the parameters of this classification network. This objective function is optimal when the discrete space is approximately linear separable with respect to the class labels. In other words, the hash codes of the images from the same semantic class have small Hamming distances between them, while the hash codes of the images from different classes have larger Hamming distances.

6.3.3 Optimization

At each iteration, the energy-based model $p(x|c;\Theta_E)$ samples synthetic contrastive image pairs by following the strategy described in Section 6.3.2. With synthetic images, we train the multipurpose hashing network to simultaneously describe the images from multiple representational perspectives. The overall training objective of the network, which combines the negative log-likelihood $\mathcal{L}_{\mathrm{E}}(\Theta_E)$, the triplet-ranking loss $\mathcal{L}_{\mathrm{H}}(\Theta_H)$, and the classification loss $\mathcal{L}_{\mathrm{C}}(\Theta_C)$, is given by:

$$\mathcal{L}(\Theta_E, \Theta_H, \Theta_C) = \mathcal{L}_E(\Theta_E) + \mathcal{L}_H(\Theta_H) + \mathcal{L}_C(\Theta_C) \tag{6.9}$$

In general, EBMs are less computationally efficient than GAN-based models [92]. However, the increased computational cost (compared to GAN-based methods) is only in the train-

ing phase, which can be mitigated with larger hardware and distributed training. With short-run MCMCs (typically only 15-20 Langevin steps in our experiments), we can already significantly reduce the training time of the EBM component. During the testing phase, since we only use the hash function and discard the remaining components (Figure 6.1b), the computation is similar to the computation of the models in other hashing methods.

6.4 Experiments

In this section, we present the evaluation results on several real-world datasets to demonstrate the effectiveness of the proposed method, compared to the existing representative and state-of-the art hashing methods.

6.4.1 Experimental Setup

Datasets: We evaluate our method on three widely-used datasets in the image hashing domain. **NUS-WIDE** [20] dataset contains 269,648 images, each of which belongs to at least one of the 81 concepts. We randomly select 5,000 images for the query set, with the remaining images used as the retrieval set. 10,000 images randomly selected in the retrieval set are used for training. **COCO** [86] dataset contains 123,287 images, labeled with at least 1 out of 80 semantic concepts. Similarly, the query set is randomly constructed with 5,000 images, with the remaining images used as the retrieval set (10,000 randomly selected images in this set are used for training). **CIFAR-10** [1] contains 60,000 images, out of which 10,000 images are randomly selected as the query set, and the remaining images used as the retrieval set. Similarly, 5,000 images randomly selected from the retrieval set are used for training. Note that this experimental setup is adopted in the previous works [14]. In real-world hashing applications, curating labeled training data for the whole retrieval set, which typically contains millions to billions of images, is an impractical task. Consequently, the labeled training data is a small fraction of the retrieval set, which matches the proposed experimental setup. In addition to these benchmark datasets, which are used for evaluating the retrieval performance, we propose to evaluate the out-of-distribution retrieval performance of the methods on two widely-used and related datasets MNIST and SVHN. **MNIST** [2] dataset contains 70,000 images, out of which 10,000 images are randomly selected as the query set, and the remaining images used as both the retrieval and training set. **SVHN** [3] contains 99,289 real-world, digit images, out of which 10,000 images are randomly selected as the query set, and the remaining images used as both the training and retrieval set. Since both MNIST and SVHN contain digit images, they are conceptually

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`

[2] `http://yann.lecun.com/exdb/mnist/`

[3] `http://ufldl.stanford.edu/housenumbers/`

related and represent a realistic change when the images in the query set slightly (but not significantly) change with respect to the images in the trained and retrieval sets. Note that, if the query images significantly change, it is extremely difficult for any hashing method to reasonably perform as well as the case where there train/retrieval and the query sets are samples from the same distribution.

Evaluation Metrics: We evaluate the retrieval performance of the methods using the standard retrieval metrics in image hashing: Mean Average Precision at K (mAP@K) and Precision at K (P@K). Following the standard evaluation protocol in the previous works [14, 15, 159], an image is relevant to the query image if they share the same label for the single-label case, and if they share at least one of the labels for the multi-label cases. In addition to reporting the Precision@1000 for evaluating precision-first retrieval in image hashing, we also follow HashNet [159] and HashGAN [14] and calculate mAP@54000 for the CIFAR-10 dataset, mAP@60000 for the MNIST dataset, and mAP@5000 for the NUS-WIDE and COCO datasets. The selected thresholds (K) for these datasets are widely used as benchmark thresholds in the previous hashing works [14, 15, 49, 55, 159].

Baselines: We compare our method against several representative approaches from image hashing. ITQ [49], BRE [79], and KSH [90]) are shallow supervised hashing approaches. CNNH [140], DNNH [80], DHN [159], DSDH [83], DVStH [87], HashNet [15], and the state-of-the-art supervised method, HashGAN [14] are deep supervised hashing approaches. SDH [120], and FastHash [56] regress the hash codes to the corresponding labels. CNNH [140] first approximately learns the hash codes, then simultaneously fine-tunes the image features and the hash functions using a deep-representation network. DNNH [80] improves over CNNH by simultaneously learn the representation and the hash function. DHN [159] further improves DNNH by preserving the pairwise similarity and control the quantization error simultaneously. HashNet [15] improves DHN by balancing the positive and negative training pairs to trade off between precision and recall. DSDH [83] also relies on a deep network to learn the hash codes but leverage the fact that the learned hash codes are ideal for classification. DVStH [87] is a variational method with a structured representation layer. HashGAN [14], the state-of-the-art GAN-based supervised method, first learns to synthesize data via GAN, then learns the hash function with additional data generated by the learned generator. Additionally, we include the results of HashGAN-1, which is the HashGAN method where the GAN model and the hash function are jointly learned in one stage.

Implementation Details: Following the evaluation approaches in [14, 15], for the shallow-hashing methods, we extract the image features (4096-dimensional vectors) from DeCAF7 [31]. Similarly, for the deep-hashing methods, we use AlexNet [78] as the backbone. The experiments on these methods are performed with the original source codes from the corresponding papers. For HashGAN, we follow the original paper and employ a four-layer

ResNet architecture [59] for the discriminator and generator in HashGAN. To train the proposed hashing network, we adopt Adam optimizer (β_1=0.9 and β_2=0.999) with a batch size of 128 and a learning rate of $1e-4$. We run 15 steps through the MCMC Langevin dynamics to train the EBM with a learning rate of 10, and standard deviation of 0.01 for the white noise. Finally, in GenHash, the parameter k depends on the complexity of the datasets. In our experiments, we observe that after setting $k = 10$ for MNIST and $k = 20$ for the other datasets, we achieve the reported results. Furthermore, if the training process only uses the sample buffer $\mathcal{B}_1$, the performance of the hash function is significantly and consistently worse in all experiments. For example, in CIFAR10, the mAP results are below 0.2 in all runs.

6.4.2 Retrieval Results

Table 6.1: Mean Average Precision (mAP) for different number of bits on the three image datasets.

Method	NUS-WIDE				CIFAR-10				COCO			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
ITQ [49]	0.460	0.405	0.373	0.347	0.354	0.414	0.449	0.462	0.566	0.562	0.530	0.502
BRE [79]	0.503	0.529	0.548	0.555	0.370	0.438	0.468	0.491	0.592	0.622	0.630	0.634
KSH [90]	0.551	0.582	0.612	0.635	0.524	0.558	0.567	0.569	0.521	0.534	0.534	0.536
SDH [120]	0.588	0.611	0.638	0.667	0.461	0.520	0.553	0.568	0.555	0.564	0.572	0.580
CNNH [140]	0.570	0.583	0.593	0.600	0.476	0.472	0.489	0.501	0.564	0.574	0.571	0.567
DNNH [80]	0.598	0.616	0.635	0.639	0.559	0.558	0.581	0.583	0.593	0.603	0.605	0.610
FastHash [56]	0.502	0.515	0.516	0.517	0.524	0.566	0.597	0.613	0.601	0.609	0.612	0.618
DHN [159]	0.637	0.664	0.669	0.671	0.568	0.603	0.621	0.635	0.677	0.701	0.695	0.694
DSDH [83]	0.650	0.701	0.705	0.709	0.655	0.660	0.682	0.687	0.659	0.688	0.710	0.731
DVStH [87]	0.661	0.680	0.698	0.702	0.667	0.695	0.708	0.714	0.689	0.709	0.713	0.721
HashNet [15]	0.662	0.699	0.711	0.716	0.643	0.667	0.675	0.687	0.687	0.718	0.730	0.736
HashGAN [14]	0.715	0.737	0.744	0.748	0.668	0.731	0.735	0.749	0.697	0.725	0.741	0.744
GenHash	**0.745**	**0.759**	**0.778**	**0.801**	**0.715**	**0.742**	**0.781**	**0.799**	**0.751**	**0.772**	**0.781**	**0.789**

Table 6.2: Precision@1000 for different number of bits on the three image datasets.

Method	NUS-WIDE				CIFAR-10				COCO			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
ITQ [49]	0.489	0.572	0.590	0.610	0.289	0.271	0.305	0.334	0.489	0.518	0.545	0.580
BRE [79]	0.521	0.603	0.627	0.641	0.398	0.445	0.471	0.488	0.520	0.535	0.559	0.571
KSH [90]	0.598	0.656	0.667	0.669	0.580	0.612	0.641	0.669	0.519	0.540	0.558	0.578
SDH [120]	0.640	0.702	0.712	0.715	0.655	0.671	0.651	0.680	0.696	0.695	0.710	0.719
CNNH [140]	0.601	0.651	0.672	0.670	0.533	0.545	0.578	0.591	0.671	0.690	0.718	0.721
DNNH [80]	0.620	0.689	0.707	0.719	0.651	0.678	0.691	0.720	0.713	0.701	0.728	0.725
DHN [159]	0.655	0.713	0.726	0.730	0.659	0.701	0.725	0.735	0.703	0.731	0.750	0.761
DSDH [83]	0.658	0.728	0.752	0.765	0.678	0.710	0.729	0.756	0.721	0.735	0.754	0.765
HashNet [15]	0.680	0.729	0.741	0.760	0.720	0.721	0.741	0.761	0.745	0.746	0.753	0.780
HashGAN [14]	0.720	0.759	0.772	0.789	0.735	0.751	0.762	0.801	0.755	0.768	0.783	0.791
GenHash	**0.749**	**0.780**	**0.808**	**0.820**	**0.780**	**0.799**	**0.823**	**0.835**	**0.789**	**0.795**	**0.811**	**0.818**

In this section, we present the results of querying for similar images. Table 6.1 shows the mAP results for all the methods. Compared to the shallow hashing methods, GenHash improves at least 14% on NUS-WIDE, 13% on CIFAR-10 and 20% on COCO. Compared to the state-of-the-art deep hashing method which does not have data synthesis (i.e., HashNet), GenHash improves at least 9%, 9%, and 5% on NUS-WIDE, CIFAR-10 and COCO, respectively. GenHash outperforms HashGAN, the state-of-the-art supervised, data-synthesis method by statistically significant margins in all the datasets.

Table 6.3: mAP performance of OOD Retrieval experiments.

Train (*Test*)	HashNet	HashGAN	GenHash
SVHN (*MNIST*)	0.181	0.354	0.609
SVHN (*SVHN*)	0.837	0.889	0.895
MNIST (*SVHN*)	0.193	0.280	0.498
MNIST (*MNIST*)	0.957	0.990	0.991

The mAP results provide empirical evidence to support our discussion in Sections 6.1 and 6.2. **First**, generating synthetic data improves the performance of a supervised hashing method. This could be explained by the fact that generative models improve the amount of "labeled" training data and increase its diversity, both of which improve the method's generalization capacity. **Second**, we can observe that the performance of HashGAN significantly decreases without the fine-tuning step (HashGAN-1's results). Training GAN-based models is difficult with problems such as mode collapse; thus in GAN-based models such as HashGAN, a second fine-tuning step, where only the hash function is trained with the synthetic data while the other components are fixed, is needed to avoid difficulties in simultaneously training the generator and discriminator. This increases the computational requirement of the GAN-based methods. Finally, GenHash, which simultaneously trains the generative model and the hash function, has better retrieval performance than the state-of-the-art, two-stage HashGAN. This supports our claim that the one-stage scheme learns better similarity-preserving hash codes of the images.

In addition, we present the Precision@1000 results in Table 6.2. Similarly, the proposed GenHash significantly outperforms all the compared methods. The Precision@1000 is calculated at the common retrieval threshold (1000) in image applications. This makes GenHash very desirable for practical, precision-oriented retrieval systems.

6.4.3 Out-of-distribution Retrieval

In this section, we show that GenHash, which is a multipurpose EBM, exhibits better out-of-distribution (OOD) robustness in retrieval than other methods. In real-world retrieval applications, the arrival of new data instances or new data format are common. This results in conceptual drift or change in the underlying data distribution where the hashing methods are trained on. A hashing method that is robust (i.e., its performance is not significantly worse) to slight changes in such underlying distributional change in the data is preferred because it takes a longer time for the trained model to become obsolete.

We propose to simulate a minor but realistic distributional change in the data as follows. In the learning phase, each hashing method is trained on a source dataset. In the testing or evaluation phase, we use a different test dataset that is conceptually similar to the source dataset but comes from a (slightly) different data distribution. We choose MNIST and SVHN as the conceptually-related datasets. One dataset is selected as both the train and retrieval dataset, while the test queries are sampled from the other dataset (we use 10,000 query samples).

Table 6.3 shows the retrieval results of GenHash, HashNet (a deep hashing method), HashNet and HashGAN (a GAN-based hashing method). As can be observed, GenHash significantly outperforms both HashNet and HashGAN in OOD retrieval, with more than 25% when using MNIST for querying, and 20% when using SVHN for querying. GenHash's mAP performance is still roughly more than 50% when the test data distribution changes. This makes GenHash still useful in practice, while in other methods, the retrieval performance significantly drops closer to the performance of a random retrieval.

While the OOD retrieval performance falls significantly, compared to the retrieval performance using data from the same distribution as that of the training data, data-synthesis methods (HashGAN and GenHash) are more robust toward distributional changes, compared to the conventional deep hashing method HashNet. In addition, when being trained on a more complex dataset (SVHN), the retrieval performance of GenHash significantly improves in our OOD tests, while the OOD retrieval performances of the other methods only slightly improve.

6.4.4 Missing-data Robustness in Retrieval

GenHash is a multipurpose EBM. Similar to other explicit generative EBMs [53, 92], we can additionally model the energy function of x. This provides us with an important advantage over other implicit generative models: we can revise (or reconstruct) a sample with corruption by initializing the input chain into the Langevin dynamics with the corrupted samples. Through the Langevin revision, the corrupted samples can be re-constructed. Note that, such feature of the EBM is not directly available in other generative hashing methods, such as HashGAN.

Table 6.4: mAP results for data corruption experiments (using 32 bits).

	Type	HashNet	HashGAN	GenHash
SnP	Clean	0.513	0.608	0.680
	Corrupted	0.223	0.281	0.652
RRM	Clean	0.471	0.615	0.654
	Corrupted	0.243	0.298	0.607

We perform the missing data experiments as follows. First, we assume that both training data and test data may contain corrupted input images. During training of GenHash, we train on the clean input as mentioned previously. For corrupted input we initialize the MCMC chains with the corrupted samples and revise these samples through the Langevin dynamics.

We corrupt the images in both the training and test sets of the CIFAR10 dataset. We corrupt 20% of the data using salt-and-pepper noise (denoted by SnP) or random rectangular mask (denoted by RRM, where the rectangles randomly cover approximately 10-20% of the images at random locations) on the images for both training and query sets. Then, the model in each hashing method is trained with the corrupted training set and the evaluation is performed

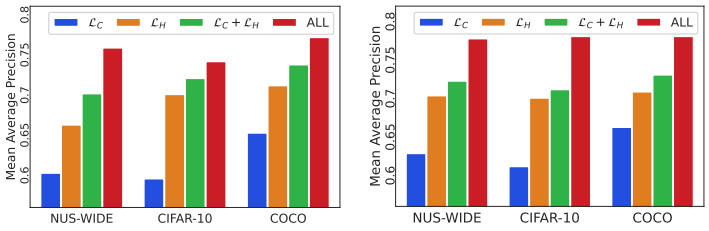
on the corrupted test set.

In Table 6.4, we show the results for the missing-data robustness experiments. As can be observed, the performances of the baseline methods, including the generative hashing HashGAN method, are significantly degraded when there are corruptions in the data. On the other hand, GenHash's retrieval performance only slightly drops when the data is corrupted. This shows the advantages of the proposed EBM-based multipurpose generative hashing network.

6.4.5 Ablation Study

In this section, we investigate the contributions of different components of GenHash through an ablation study. We evaluate four variants of GenHash:

- $\mathcal{L}_{\text{C}}$: only discriminative head.
- $\mathcal{L}_{\text{H}}$: only contrastive hashing head. In this variant, the contrastive samples are selected from the available training data (instead of being synthetic samples).
- $\mathcal{L}_{\text{C}} + \mathcal{L}_{\text{H}}$: the combination of discriminative and contrastive heads. Similar to the $\mathcal{L}_{\text{H}}$ variant, the contrastive samples are selected from the training data.
- ALL: the final GenHash model with all three heads.



(a) 32 bits (b) 64 bits

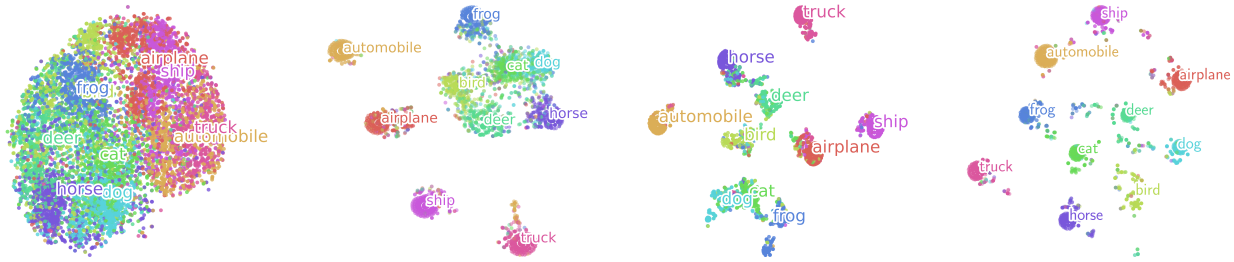
Figure 6.3: Ablation Study: mAP results of the different variants of the GenHash model obtained by removing different components of the loss function. $\mathcal{L}_{\text{C}}$: GenHash with only the Discriminative head. $\mathcal{L}_{\text{H}}$: GenHash with the Contrastive hashing head. $\mathcal{L}_{\text{C}} + \mathcal{L}_{\text{H}}$: GenHash without the Energy head. ALL: the complete GenHash model.

Figure 6.3 shows the mAP values of these variants on the three datasets. As can be observed, each addition of any proposed head contributes toward improving the retrieval performance. When we train the network for either only classification objective ($\mathcal{L}_{\text{C}}$) or only discriminative hashing objective objective ($\mathcal{L}_{\text{H}}$), the retrieval performance significantly degrades. Note that $\mathcal{L}_{\text{C}}$ variant is not built for hashing, since both the contrastive loss between images and especially the quantization error are not included in the objective to train the model. Since the retrieval phase involves a quantization step on real-valued vectors, considerable quantization errors may be accumulated, which makes the final binary codes in this variant less effective. The fact that the $\mathcal{L}_{\text{H}}$ variant significantly improves over the $\mathcal{L}_{\text{C}}$ shows us

the importance of these hashing losses with respect to the hashing-based retrieval. In the $\mathcal{L}_{\mathrm{C}} + \mathcal{L}_{\mathrm{H}}$ variant, the retrieval performance increases compared to the variant with only one of these two heads. Finally, the addition of the energy head (ALL) further improves the retrieval performance. When the model is trained with the synthetically generated data, it is expected to have better generalization.

6.4.6 Qualitative Analysis

Visualization of the hash Codes



(a) ITQ (b) HashNet (c) HashGAN (d) GenHash

Figure 6.4: The t-SNE visualizations of the quantized 32-bit hash codes learned by ITQ, HashNet, HashGAN and GenHash on the CIFAR-10 dataset.

We present the visualization of the hash codes of the hash functions in different methods (i.e., ITQ, HashNet, HashGAN and GenHash) in Figure 6.4. We project the hash codes into 2-dimensional embeddings using the t-SNE method. As Figure 6.4d displays, the learned hash codes of ITQ cluster tightly together, which makes its retrieval performance suboptimal. While HashNet's learned structure is significantly better than that of ITQ, its learned hash codes of HashNet still exhibit a more entangled structure than the learned hash codes of both HashGAN and GenHash. The learned discrete space of GenHash exhibits a more separated structure, compared to other methods. This explains the superior retrieval performance of GenHash compared to the other baselines.

Generated Images

In this section, we show the randomly selected MNIST digits and CIFAR-10 in the exploit buffer $\mathcal{B}_2$ in order to understand the learning process of GenHash. Figures 6.5a and 6.5b show the MNIST and CIFAR10 generated images, respectively, where each row contains images of the same label. Note that the training process uses very short MCMC chains; however, because the images, which begin with random noise, are sampled from the replay buffer multiple times and revised through the Langevin dynamics, the training process intuitively use a much longer mixing time for these samples.

As can be observed, GenHash can successfully generate a variety of samples from each class in both datasets. While there are a few outliers in each class, these images are likely in the younger MCMC chains; however, the visual appearances of these images are still closer to the digits of their classes than to random noise, as can be seen in the corresponding version of the explore buffer $\mathcal{B}_1$ in other works [53,92]. This difference in the generated images by the proposed exploit buffer $\mathcal{B}_2$ makes our learning process more stable, and make the contrastive hashing possible with a single-network generative model.

(a) MNIST (b) CIFAR10

Figure 6.5: Samples from the exploit buffer $\mathcal{B}_2$. Images on each row are from the same semantic class.

6.5 Summary

This chapter proposes a unified generative framework, called GenHash to solve the image hashing problem. The framework learns a multipurpose hashing network to represent images from multiple perspectives, including classification, hashing, and probability density estimation. This approach learns high-quality binary hash codes and achieves state-of-the-art retrieval performance on several benchmark datasets. Furthermore, GenHash is significantly more robust to out-of-distribution retrieval compared to the existing methods and is able to handle significant corruption in the data with trivial drops in the retrieval performance. In GenHash, we also train a single EBM-based network, which makes it easier for the practitioner to design better architectures. This is preferred compared to other GAN-based approaches because designing the discriminator and generator networks is not a trivial task with various problems when one network has more capacity than the other.

Chapter 7

Interpretable Graph Similarity Computation via Differentiable Optimal Alignment of Node Embeddings

In the previous chapters, we demonstrated several benefits of studying generative models in the context of the ANN method, called hashing. One particularly interesting concept in implicit generative models is the distance between two distributions. Certain types of distributional distances, such as the Wasserstein distance, consider the geometry of the data. For example, the optimal transport between two distributions essentially measures the minimal transformation between two sets of vectors in the vector space. In this chapter, we explore this idea of minimal transformation (so-called alignment in this chapter) and develop an interpretable model to solve an important task in similarity search: approximating the similarity between two structured objects, specifically graphs. We then show that the proposed model not only learns to approximate the similarity between a pair of graphs (consequently, resulting in better retrieval results) but is also able to "explain" the decision produced by the model. The explainability is realized through the fact that we can visualize the transformation between one graph into the other graph, and can help in advancing or confirming the domain-specific knowledge in domains which involve graphs.

7.1 Introduction

Graphs are non-linear data structures used to model a set of inter-connected objects (nodes) and their relationships (edges). Recently, the ubiquitous expressive power of graphs has received immense attention in data modeling across various research areas including social networks, natural sciences (drugs and protein-protein interaction networks), and knowledge graphs. Consequently, a wide range of graph analytical techniques are being developed to learn and extract useful patterns from graph data. One of the challenging problems when dealing with graph databases is to compute the similarity between a pair of graphs, which is critical to several graph applications such as retrieval from graph databases and graph clustering. To tackle this problem, different graph similarity measures have been proposed in the literature. Among them, Graph Edit Distance (GED) and Maximum Common Subgraphs (MCS) are two examples of the most prevalent measures. GED and MCS, which are domain-agnostic measures of structural similarity between the graphs, define the similarity as a function of pairwise alignment of different entities (such as nodes, edges, and subgraphs) in the two graphs. As such, GED and MCS have important applications in many domains such as bioinformatics and cheminformatics [43, 44], because the explicit explainability of the similarity score provides transparency and justification, as well as additional informative inductive knowledge towards decision making in downstream analysis in these domains. For example, in medicinal chemistry in particular, the availability of 'rules of thumb' underscores the willingness, in certain situations, to sacrifice accuracy in favor of models that better fit the human intuition [68]. In addition to being domain-agnostic similarity measures, GED and MCS can be deployed in a real application without any prior domain knowledge. For-

Table 7.1: Characteristics of graph similarity algorithms along with the representative methods. × indicates the method lacks the specific characteristic. ✓indicates the method achieves the objective. 'N/A' stands for Not Applicable. A desirable approach will have *high accuracy*, *high interpretability*, and *low computation*. Our proposed GOTSim method satisfies all these characteristics by employing an *end-to-end* learning-based approach.

	Bipartite [37, 111]	SimGNN [4]	GraphSim [5]	Graph Matching [17, 84, 100]	OAK [104]	GOTSim (Ours)
Learning-based	×	✓	✓	✓	✓	✓
End-to-End	×	✓	✓	✓	×	✓
Accuracy	Low	High	High	High	Low	High
Interpretability	High	N/A	N/A	Low	Low	High
Computation	High	Low	Low	Low	Low	Low

mally, GED finds the number of edit operations which transforms one graph to another, and MCS finds the most common subgraphs between the two graphs. GED, under a particular cost function setup, is equivalent to MCS [12]. Exact computation of GED or MCS, using classical, combinatorial search approaches, however, is NP-Hard. Computing GED and MCS

is intractable for larger graphs that have more than a few tens of nodes.

Recently, by learning a similarity function to approximate the similarity metric of interest, the works in [4,5] show that one can efficiently find similar graphs to a query graph, compared to using classical combinatorial methods such as Beam [102] or Bipartite [37, 111]. These methods model the similarity function as a neural network either on the two graph-level embeddings representing the two graphs or on the two sets of node embeddings representing the two graphs. While their performance in both similarity approximation and retrieval tasks approaches that of the classical computations of GED and MCS, they lack several important characteristics that are typically seen in the classical methods, specifically the explainable alignment between nodes or edges between the two graphs.

A similar research setting is also seen in several works solving the graph matching problem [17, 38, 84, 100]. However, their primary goal is fundamentally different. Specifically, graph matching focuses on approximating the similarity between graph entities such as nodes or edges, regardless of the overall structural similarity between the entire graphs. The learned matching is only required to be a discrete distribution over potential correspondences of a node in a graph to each node in the other graph, but not necessarily an injective mapping between nodes. Furthermore, some methods require fine-grained annotations such as node-node or edge-edge similarities in order to learn a sparse correspondence distribution [38]. Our work is different from these methods and primarily focuses on approximating not only the injective assignment, but also capturing the explanation for node deletion and insertions. Hence, this will be similar to the explicit explanability found in classical combinatorial methods by using only annotated graph-level similarities.

It should be noted that a low similarity computation cost and an explainable alignment as part of the computation are two of the desired properties of graph similarity search. To the best of our knowledge, *none of the existing learning-based methods can achieve the explicit node alignment results while approximating the graph-graph similarity score somewhat identical to classical methods.* In this chapter, we denote such explicit node-alignment as the "interpretability" of the graph similarity algorithm. In this work, we first propose to learn the graph embedding through a context-aware Graph Neural Network (GNN) model, such as Graph Convolutional Network (GCN), so that we can capture the local and global structures around the nodes in the embedding space. Then, we propose to directly solve for the optimal alignment between nodes of the two graphs from their node-node similarity matrix. However, *our proposed optimal alignment formulation is guaranteed to have a stable and differentiable solution, which is crucial for both gradient-based stochastic optimization of the neural models and finding sparse exact alignment solutions similar to those of classical GED computations.*

We name our method GOTSim. Given two graphs, GOTSim first learns node embeddings using any suitable graph embedding framework, such as GCN [75]. Then, GOTSim directly solves the optimal assignment problem between the two sets of embeddings using a novel differentiable algorithm. Our contributions are as follows:

- Propose a novel optimal assignment objective based on the graphs' node embeddings to approximate graph similarity. In this framework, the similarity between the two graphs is related to the optimal assignment cost, which is the minimum cost of transforming one graph to another in the (node) embedding space. This learning-based framework works on any notion of graph similarity, such as GED or MCS.
- Develop an efficient differentiable algorithm to solve the optimal assignment objective with a polynomial computational complexity, which is also the complexity of existing neural learning-based graph-similarity methods. The algorithm is suitable for any gradient-based learning (such as SGD).
- Demonstrate the effectiveness of our approach on various widely used graph benchmark datasets. GOTSim achieves the lowest error in similarity estimation and superior ranking performance (on several ranking metrics) compared to existing state-of-the-art graph similarity methods.

For domain experts, GOTSim can provide valuable insights into the downstream tasks that are being studied; for example, one can visualize the learned node alignment between two predictably similar protein graphs for further biological investigation. Thus, GOTSim fulfills the desired characteristics of an ideal graph similarity algorithm (as shown in Table 7.1).

The rest of the chapter is organized as follows. We discuss the related work in Section 7.2. In Section 7.3, we describe the problem definition and the details of the proposed GOTSim method. Finally, we present quantitative and qualitative experimental results in Section 7.4 and conclude our discussion in Section 7.5.

7.2 Related Work

We first discuss representation learning in graphs and then discuss other works related to graph similarity and graph matching.

7.2.1 Graph Representation Learning

GNNs have gained huge popularity in the past few years. These models apply deep neural networks on graph data and can address the limitations of earlier representation learning methods. GCNs are a special class of GNNs, that apply message passing on graphs and compute node representations with input node/edge features and graph structures. These models can be trained with any downstream task to learn node representations specifically for a given task and achieve better performance. Specifically, GCNs use a convolutional layer to perform neighborhood aggregation, thus generating a node's representation by aggregating

its own features and its neighbors' features, where the neighbors generally correspond to directly connected nodes. By stacking multiple such convolutional layers, GCNs enable us to estimate context-aware representations of the nodes in graphs.

The initial GCN network, proposed by [75], uses all nodes that are directly connected to v (the target node for which we want to estimate the representation), in addition to v itself (i.e., self-edge), as the neighbors of v. The message is constructed as a projection of the features of the neighboring nodes, and the aggregation is modeled by taking the sum of the incoming messages and normalizing the sum. GAT [131] extends GCN by introducing the attention mechanism as a substitute for the statically normalized convolution operation. The attention weights are used to perform a weighted combination of the messages from the neighbors, as opposed to the vanilla sum used in GCN. We can apply message passing on a graph multiple times to have a node gather information from nodes multiple hops away, instead of just its direct neighbors. When considering tasks on graphs, one critical component to design is the readout function, i.e., the mapping from the set of node representations to a fixed-size vector representation of the graph. A readout function can be a pooling operation (such as mean, max etc., similar to CNNs) or can also have learnable parameters.

The message passing formulation of GCNs has also been extended to heterogeneous graphs, i.e., graphs with multiple node-types and edge-types. Popular examples include Relational Graph Convolution Network [116], Heterogeneous Graph Attention Network [137], Metapath Aggregated Graph Neural Network [41] and Deep Heterogenous Graph Convolutional Networks [99].

7.2.2 Graph Similarity Search

Searching for graphs involves selection of the appropriate pairwise, graph-graph similarity measure and an efficient similarity computation. In this section, we provide a detailed discussion in this direction.

Characteristics of graph similarity algorithms

Computing similarity between the structured graph objects is a challenging problem with many important real-world applications, in particular, similarity-based retrieval in graph databases. For example, in the field of computer security, we need to search for similar binary functions, where given a binary which may or may not contain code with known vulnerabilities, we wish to check whether any control-flow graph in this binary is sufficiently similar to a database of known vulnerable functions [84].

Graph similarity is typically defined based on (sub-)graph isomorphism [8, 118] or some structural similarity measure such as GED or MCS [109, 139]. Exact similarity computation is known to be computationally expensive in practice. For example, exact GED calculation

is NP-Hard and does not scale well to graphs with more than a few tens of nodes. Thus, approximation algorithms have been designed to calculate these similarity measures. These approximate methods can be broadly divided into two categories: (i) search-based similarity computation, and (ii) function-estimation similarity computation.

Search-based similarity computation

Examples of the search-based methods (specifically GED) include Beam search [102], Bipartite approximation [37, 111], and Hausdorff approximation [39]. Beam heuristically explores the search-graph by expanding the most promising node in a limited set, thus approximating the A^* algorithm to find the GED. Bipartite provides an upper bound, while Hausdorff provides a lower bound to the GED. These algorithms essentially estimate the set of edit operations with a minimal cost of transforming one graph to another graph and this minimal cost represents the edit distance between the pair of graphs. Consequently, such optimal edit operations can be used as an explanation of the associated edit similarity score. Such explainable characteristic of these classical algorithms is extremely useful for downstream tasks in domains such as bioinformatics and cheminformatics.

Function-estimation similarity computation

Recently, several function-estimation similarity computation methods have been proposed, including SimGNN [4], GraphSim [5] and Graph Matching Network (GMN) [84]. Function-estimation algorithms have a significant computational advantage over search-based approaches. These approaches learn their functional parameters from empirical data using the principle of empirical risk minimization, and hence they are also suitable for any similarity definition, including GED and MCS. By employing end-to-end deep GNN models, recent works [4, 5] have demonstrated a superior performance over classical methods such as Bipartite or Hausdorff. Although these existing methods effectively estimate the graph similarity, their lack of explanation in node alignments is a crucial limitation.

Our proposed method also belongs to the category of learning-based similarity computation. Unlike existing learning-based approaches, our method is capable of both accurately approximating the graph similarity and providing similar explanation which is found in classical search-based approaches when approximating similarity measures such as the GED.

7.2.3 Graph Matching

Identifying correspondences between nodes of graphs, such as in the case of calculating GED, is a prevalent problem that arises in different domains, and thus has been studied under various terminologies. A set of fundamentally different techniques commonly referred to as graph matching or graph alignment have been developed in bioinformatics and computer vision. The techniques developed in these areas, however, are non-exact because large networks without any specific structural properties are commonly studied. In graph matching, given two graphs G_s and G_t with node sets $\mathcal{V}_s$ and $\mathcal{V}_t$, respectively, and adjacency matrices A_s and A_t, respectively, the goal is to find correspondences that can avoid mapping of adjacent nodes in the source graph to different regions in the target graph [112], as expressed in the following maximization objective function:

$$\sum_{(i,i')\in\mathcal{V}_s} \sum_{(j,j')\in\mathcal{V}_t} A_{i,i'} A_{j,j'} S_{i,j} S_{i',j'} \tag{7.1}$$

Recently, various deep graph-matching methods which employ GNN embeddings have been developed. In soft graph matching methods, such as Graph Matching Network (GMN) [84], Graph Optimal Transport [100], and Optimal Assignment Kernel (OAK) [104], the rectangular matching matrix $S \in \mathbb{R}^{\mathcal{V}_s \times \mathcal{V}_t}$ is a doubly stochastic matrix. In sparse graph matching methods, such as in [38], S is constrained to be a rectangular permutation matrix. To learn the model parameters, either node-level (each pair of nodes of two graphs have a similarity label) or graph-level (each pair of graphs has a similarity label) matching annotation is used.

It can be shown that optimizing the objective in Equation (7.1) is equivalent to solving Equation (7.2) in Section 7.3.2 under certain constraints. When GED is used as a graph-level annotation, graph matching approximates the GED. However, the learned assignments between graph matching methods and classical GED computation can be very different. For example, soft graph matching learns a doubly stochastic matrix, while GED computation learns a permutation matrix. While some of the graph matching works [38] attempt to approximate a sparse assignment matrix (equivalently, this can be seen as an approximation to the permutation matrix), these methods require domain-specific node-level annotation. Thus, they are not capable of estimating the graph-level similarity scores. Furthermore, as discussed previously, the requirement of this type of annotation significantly limits the applications of these approaches in several domains. Contrary to this, our work can approximate the graph similarity while simultaneously learning a sparse permutation assignment only using the graph-level annotation. Furthermore, our method can additionally provide explanation for operations such as deletion and insertion similar to those of classical methods.

Table 7.2: Notations used in this chapter.

Notation	Description
$\mathbf{h}_{G,k,i}$	Embedding of i^{th} node of graph G at layer k in the GCN.
$\mathbf{H}_{G,k}$	Set of node embeddings of a graph G at layer k in the GCN.
$\mathbf{C}_{G_1,G_2,k}$	Pairwise similarity matrix between two graphs G_1 and G_2 at layer k in the GCN.
$c_{k,i,j}$	Distance (in embedding space) between the nodes v_i and v_j of G_1 and G_2, respectively, i.e., the substitution cost.
$\mathbf{d}_{k,i}$	Deletion cost of node v_i in G_1.
$\mathbf{a}_{k,j}$	Insertion cost of node v_j in G_2.
$\mathbf{s}(G_1, G_2)$	True similarity score of graphs G_1 and G_2.
$\hat{\mathbf{s}}(G_1, G_2)$	Predicted similarity score of graphs G_1 and G_2.
θ	Parameters of the network.
$\mathbf{M}^*$	Permumation matrix that is the optimal solution to the LP problem.
η	Learning rate.
K_e	Number of training epochs.

7.3 The proposed GOTSim Model

We can formally define graph as $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, where $\mathcal{V}$ represents the node set consisting of nodes $\{v_i\}_{i=1,...,N}$ and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the corresponding adjacency matrix where $\mathbf{A}_{ij}$ represents the edge weight between nodes v_i and v_j. We only consider undirected and unweighted graphs in this work and thus $\mathbf{A}$ is symmetric and generally sparse. However, our method is a general graph-similarity computation framework, and can be extended to other graph types. The notations used in this chapter are given in Table 7.2.

Problem Statement: Given two graphs $G_1 = \{\mathcal{V}_1, \mathbf{A}_1\}$ and $G_2 = \{\mathcal{V}_2, \mathbf{A}_2\}$, the goal of the graph similarity computation task is to approximate the similarity score between the graphs, denoted by $\hat{\mathbf{s}}(G_1, G_2)$.

The proposed GOTSim model consists of two sequential modules: 1) multiple-scaled convolutional aggregations and 2) differentiable graph matching. For a pair of graphs, GOTSim first computes the vector embeddings of the nodes in the graphs at multiple GCN layers; then, at each layer, GOTSim computes the pairwise similarity matrix between the two sets of node embeddings representing the graphs; finally, GOTSim computes the optimal graph transformation cost, at each layer, from the similarity matrix and approximates the similarity score by aggregating the optimal costs from all GCN layers. The parameters of all computational operations are learned in an end-to-end manner.

7.3.1 Multiple-scaled Node Embeddings

In our proposed approach we use context-aware node embeddings. To compute them, we employ GCN [35, 75] which uses a convolutional layer to perform neighborhood aggregation, thus generating a node's representation by aggregating its own features and its neighbors' features, where the neighbors generally correspond to directly connected nodes. By stacking multiple convolutional layers, GCNs enable us to learn context-aware representations of the nodes in graphs, when a K-depth GCN is able to estimate the representation of a node using the context within K-hops. More concretely, the Graph Convolution operator at the k-th

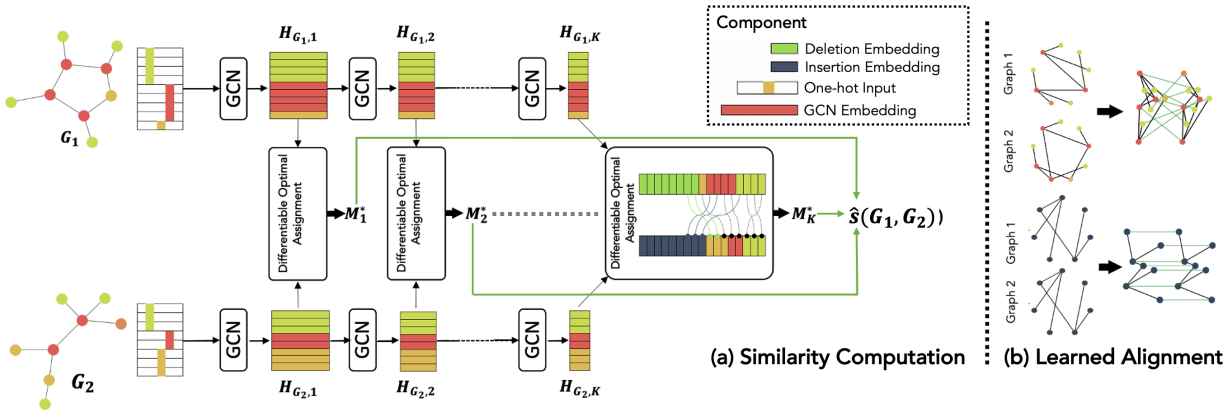


Figure 7.1: GOTSim model (a) and an illustration of the learned graph matching (b). **(a)** the one-hot encoded node input are given as input to the GCN, which estimates the node embeddings; the 'Differentiable Optimal Assignment' block computes the optimal assignment between two sets of node embeddings; the final similarity score is the average of the optimal assignment costs for each GCN layer. Solving the optimal assignment problem at each GCN layer results in an interpretable alignment **(b)** between the two graphs using the information at that layer; for two isormorphic graphs, GOTSim gives the perfect alignment.

GCN layer transforms the representation of node v_i as follows:

$$\mathbf{h}_{G,k,i} = \sigma(\hat{\mathbf{h}}_{G,k-1,i}\mathbf{W}^k + \mathbf{b}^k), \quad \hat{\mathbf{h}}_{G,k-1,i} = \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{\mathbf{d}_i \mathbf{d}_j}} \mathbf{h}_{G,k-1,j}$$

where $\sigma(\cdot)$ is nonlinear activation, $\mathbf{d}_i$ is the degree of node v_i plus 1 (assuming self-loop), $\mathcal{N}(i)$ represents the set of first-order neighbors of v_i and $\{\mathbf{W}^k, \mathbf{b}^k\}$ represent the filter parameters of the k-th GCN layer, which are shared by all nodes.

In multi-layer GCN, each layer has the effect of capturing a particular higher-order structural information around node v in its representation. While comparing two graphs using their node embeddings, using a particular higher-order representation is usually not sufficient. An intuitive and effective approach is to utilize the embeddings at multiple GCN layers. Specifically, let $\mathbf{H}_{G,k} = \{\mathbf{h}_{G,k,i} \mid v_i \in \mathcal{V}_G,\ i = 1, ..., N\}$ be the set of node embeddings of a

graph G at layer k in the GCN, G is represented by K sets of vectors as $\mathbf{H}_G = \{\mathbf{H}_{G,k} \mid k = 1, ..., K\}$.

At each layer k, we can compute the pairwise similarity matrix $\mathbf{C}_{G_1,G_2,k}$ between two graphs G_1 and G_2. Since node orderings are not graph invariant, it is very strict to assume an order in the similarity matrix. For this reason, we take a completely different modeling approach than existing learning-based graph similarity methods and compare graphs using their bag-of-vectors representations.

7.3.2 Optimal-Assignment Similarity

As discussed earlier, a graph can be represented as a "bag" of its building blocks, e.g., context-aware nodes where the context encodes the structural information around the node. Given two bag-of-vectors $\mathbf{H}_{G_1,k}$ and $\mathbf{H}_{G_2,k}$ from graphs G_1 and G_2, respectively, at layer k, and $\mathbf{c}$ is a pairwise distance function between two node embeddings, we define the cost matrix between G_1 and G_2 as:

$$\mathbf{C}_{G_1,G_2,k} = \begin{bmatrix} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N,1} & \cdots & \mathbf{c}_{k,N,N} \end{bmatrix}$$

where $\mathbf{c}_{k,i,j} = \mathbf{c}(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{G_2,k,j})$ is the distance (in embedding space) between nodes v_i and v_j from G_1 and G_2, respectively. Examples of $\mathbf{c}_{k,i,j}$ include the Cosine or Euclidean distance. First, let us assume that G_1 and G_2 have the same cardinality N. The graph similarity at layer k can be defined via the optimal transformation cost from G_1 to G_2 as follows:

$$\mathbf{L}_{G_1,G_2,k} = \min_{\mathbf{M}} \mathbf{M} \circ \mathbf{C}_{G_1,G_2,k}, \tag{7.2}$$

where $\mathbf{M}$ is subject to the following constraints:

$$\sum_i \mathbf{M}_{ij} = 1 \;\; \forall j, \quad \sum_j \mathbf{M}_{ij} = 1 \;\; \forall i, \quad \mathbf{M}_{ij} \geq 0 \;\; \forall i, j \tag{7.3}$$

Intuitively, this objective is the optimal transformation cost from G_1 to G_2 in the embedding space. When a node v_i in G_1 is assigned to a node v_j in G_2, they should have similar local and global structures, which are captured in the embedding space. In other words, the embedding vectors of $\mathbf{h}_{G_1,*,i}$ (and $\mathbf{h}_{G_2,*,j}$) of v_i (and v_j) are closer in the embedding space. Therefore, the minimal assignment cost directly approximates the similarity between two graphs. Note that this formulation is order invariant. In fact, any permutation of the similarity matrix still results in the same optimal solution.

However, the above formulation has two major problems. First, its solution and the gradients are highly unstable when M is a doubly stochastic matrix [46]. Second, it is only applicable

when G_1 and G_2 have the same cardinality N. By changing to a new square matrix M in the formulation of the optimal assignment problem (proposed in this Section) and enforcing a permutation matrix solution (proposed in Section 7.3.3), we simultaneously achieve both stable solution/gradients and a highly interpretable model.

Without loss of generality, let us now assume that $N_1 \neq N_2$, where $N_1 = |G_1|$ and $N_2 = |G_2|$. In this scenario, besides the node-assignment cost, we need to account for the costs of the node-deletion and the node-insertion operations. Specifically, from the perspective of G_1, there are at most N_2 node-insertion and at most N_1 node-deletion operations. This motivates us to extend the similarity matrix $C_{G_1,G_2,k}$ as follows:

$$\mathbf{C}_{G_1,G_2,k} = \left[\begin{array}{ccc|ccc} \mathbf{c}_{k,1,1} & \cdots & \mathbf{c}_{k,1,N_2} & \mathbf{d}_{k,1} & \cdots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{c}_{k,N_1,1} & \cdots & \mathbf{c}_{k,N_1,N_2} & \infty & \cdots & \mathbf{d}_{k,N_1} \\ \hline \mathbf{a}_{k,1} & \cdots & \infty & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \infty & \cdots & \mathbf{a}_{k,N_2} & 0 & \cdots & 0 \end{array}\right] \tag{7.4}$$

The original similarity matrix is extended to include the cost blocks for deletion and insertion operations. $\mathbf{d}_{k,i}$ denotes the deletion cost of node v_i in G_1 while $\mathbf{a}_{k,j}$ denotes the insertion cost of node v_j in G_2. Note that, a deletion of a node in a graph is equivalent to an insertion of another node in the other graph, thus, this configuration covers both node insertions and deletions in both the graphs.

There are several ways to define $\mathbf{d}_{k,i}$ and $\mathbf{a}_{k,j}$ from node-embedding vectors $\mathbf{h}_{G_1,k,i}$ and $\mathbf{h}_{G_2,k,j}$ of the two graphs. One can define $\mathbf{d}_{k,i}$ and $\mathbf{a}_{k,i}$ as constant costs, i.e., $\forall i, j$ and $\mathfrak{d}_k, \mathfrak{a}_k \in \{x : x \in \mathbb{R}, 0 \leq x < \infty\}$, $\mathbf{d}_{k,i} = \mathfrak{d}_k$ and $\mathbf{a}_{k,j} = \mathfrak{a}_k$, which are independent of the nodes. However, such definition results in hyperparameter choices, which in turn, require additional training time for model selection. Therefore, we propose to approximate such cost by first defining two global embedding vectors $\mathbf{h}_{k,d}$ and $\mathbf{h}_{k,a}$ and then computing deletion and insertion costs as follows:

$$\mathbf{d}_{k,i} = c(\mathbf{h}_{G_1,k,i}, \mathbf{h}_{k,d}), \quad \mathbf{a}_{k,i} = c(\mathbf{h}_{G_2,k,i}, \mathbf{h}_{k,a}) \tag{7.5}$$

The global embedding vectors $\mathbf{h}_{k,d}$ and $\mathbf{h}_{k,a}$ are learned in model training, along with other GCN vectors $\mathbf{h}_{G_1,k,i}$ and $\mathbf{h}_{G_2,k,j}$ of the two graphs. We can show that this is related to the Hungarian formulation [111] of approximating the GED. However, our formulation is more powerful because it generalizes to graph embedding and is suitable for similarity function estimation (because of its differentiable solution, which will be discussed in the next section).

Given the optimal assignment cost at each layer-k of the GCN, the final similarity score can be computed from the weighted average of the normalized assignment costs at all GCN levels as:

$$\hat{\mathbf{s}}(G_1, G_2) = 1 - \frac{1}{K}\sum_{k}^{K} \frac{\mathbf{L}_{G_1,G_2,k}}{0.5 \times (N_1 + N_2)} \tag{7.6}$$

We learn the parameters of GOTSim by minimizing the empirical mean squared error, as follows:

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{(G_1,G_2)\in\mathcal{D}} ||\hat{\mathbf{s}}(G_1, G_2) - \mathbf{s}(G_1, G_2)||_2^2, \tag{7.7}$$

where $\mathbf{s}(G_1, G_2)$ is the target similarity score of graphs G_1 and G_2 and $\mathcal{D}$ is the set of training graph pairs.

7.3.3 Solving the Optimal Assignment Problem

The objective in Equation (7.2) admits the solution $\mathbf{M}^*$ that is doubly stochastic. This means that each node v_i in G_1 can be partially assigned to a node v_j in G_2. However, it is known that this Linear Programming (LP) formulation has an unstable and not always unique solution [46] i.e., $\mathbf{L}_k$ is not continuously differentiable.

Fortunately, since our objective is to learn an explainable solution similar to that of the classical GED computations, we have a more stringent constraint: a node in G_1 can be mapped to one and only one node in G_2. This is also true for the insertion and deletion operations. We can easily show that enforcing this sparsity constraint turns our optimal assignment problem into a linear sum assignment program [13]. Consequently, instead of the doubly stochastic matching matrix $\mathbf{M}^*$, the solution is a permutation matrix, i.e., $\mathbf{M}_{ij} \in \{0, 1\}$. The permutation-matrix is also a more stable solution and we can easily compute the derivative of the objective $\mathbf{L}_{G_1,G_2,k}$ with respect to the model's parameters. Specifically, Theorem 1 shows that the model parameters can be efficiently computed by solving for the optimal solution.

Theorem 2. *The derivative of $L_{G_1,G_2,k}$ can be calculated as follows:*

$$\frac{\partial\ \mathbf{L}_{G_1,G_2,k}}{\partial\theta} = \mathbf{M}_k^* \frac{\partial\ \mathbf{C}_{G_1,G_2,k}}{\partial\theta} \tag{7.8}$$

where $\mathbf{M}_k^*$ is the permutation-matrix solution at layer k.

Proof. Using the chain rule, the derivative of $\mathbf{L}_{G_1,G_2,k}$ is:

$$\frac{\partial\ \mathbf{L}_{G_1,G_2,k}}{\partial\theta} = \frac{\partial\ \mathbf{M}_k^*}{\partial\theta}\mathbf{C}_{G_1,G_2,k} + \mathbf{M}_k^* \frac{\partial\ \mathbf{C}_{G_1,G_2,k}}{\partial\theta} \tag{7.9}$$

Using sensitivity analysis, we can see that the first derivative $\frac{\partial \mathbf{M}_k^*}{\partial\theta} = 0$ for non-degenerate solutions M_k^*. Intuitively, a infinitesimal change in θ results in the exact same optimal solution $\mathbf{M}_k^*$. Therefore, we arrive at the formulation of the derivative in the Theorem. □

Given Theorem 1, any continuously differentiable distance function $\mathbf{c}$ is suitable. Some examples are the euclidean distance or the cosine distance. In our chapter, we employ the cosine distance, which is also used in other learning-based graph similarity methods.

Algorithm 5 GOTSim Model Training

Require: Training graph pairs (G_i, G_j), number of training epochs K_e, learning rate η.
Output: parameters of the network $\{\theta\}$.
for *number of training epochs* K_e **do**
  Sample a minibatch $\{(G_1, G_2)_l | l = 1, \cdots, m\}$ of m graph pairs.
  for *each graph pair* (G_1, G_2) **do**
    for *GCN layer* k **do**
      Compute $h_{G_1,i}$ and $h_{G_2,j}$ $\forall$ $v_i \in G_1$, $v_j \in G_2$.
      Compute similarity matrix $C_{G_1,G_2,k}$ using Eq. (7.4).
      Solve for the solution M_k^* of RHS of Eq. (7.2) and compute $\mathbf{L}_{G_1,G_2,k}$ using Eq. (7.2).
      Compute the gradient of $\mathbf{L}_{G_1,G_2,k}$ with respect to the network parameters using the Eq. (7.8).
    end
    Compute $\hat{s}(G_1, G_2)$ in Eq. (7.6) using $\mathbf{L}_{G_1,G_2,k}$ computed at each GCN layer k.
  end
  Compute the mini-batch loss $\mathcal{L}$ using Eq. (7.7).
  Update $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$.
end

The algorithm of training GOTSim is described in Algorithm 5. GOTSim has a computational complexity of $O(max(N_1, N_2)^{2.5})$ (dominated by the optimal assignment complexity [70]). Also, similar to high computation costs in existing learning-based graph-similarity methods [5, 84], this is a non-trivial cost. The price to pay for the generalization ability of (neural) learning-based graph-similarity methods is that they cannot directly be used for large graphs. In practice, however, GOTSim is very efficient when being trained on all real-world datasets which are evaluated in this chapter. Also note that, GOTSim, and other learning-based methods, can easily approximate the similarity between graphs with an order of magnitude more number of nodes than the current limit of existing classical, non-learning based approaches.

7.4 Experimental Results

We evaluate GOTSim against several existing state-of-the-art baselines for GED and MCS computation, with a primary goal of addressing the following aspects of the graph similarity task:

- **(RQ1) Effectiveness**: How accurate is GOTSim, compared to the state-of-the-art approaches, in terms of both similarity score approximation and graph similarity retrieval

tasks? The results are presented in Section 7.4.2.

- **(RQ2) Interpretability**: Can we explain how GOTSim makes its prediction, in a similar way to classical methods like the exact GED algorithm A^* and the approximate GED algorithm (Bipartite)? We discuss this in Section 7.4.3.

7.4.1 Experimental Setup

Datasets

We employ four widely-used real-world datasets AIDS, LINUX, IMDB and PTC. For each dataset, we employ five-fold cross validation and split the dataset into 5 subsets: one for validation, one for testing and the rest for training. **AIDS** [136] is a collection of antivirus screen chemical compounds from the Developmental Therapeutics Program at NCI/NIH[1], and has been used in several existing works on graph similarity search [136, 152]. **LINUX** [136] dataset consists of 48,747 Program Dependence Graphs (PDG) generated from the Linux kernel. Each graph represents a function where a node represents one statement and an edge represents the dependency between any two statements. **PTC** [128] dataset consists of 344 chemical compound graphs that report the carcinogenicity for male and female rats. Each node in the PTC dataset has one out of 17 possible labels. **IMDB** [148] dataset consists of 1,500 ego-networks of movie actors, where there is an edge if two people appear in the same movie. We follow the same pre-processing steps that are described in [5]. These are the standard datasets that have been used in existing works [4, 84]. While the number of graphs may appear to be small, the similarity is computed between each pair of the graphs, making the total number of unique data-instances (~100K for PTC, ~0.5M for AIDS, ~1M for LINUX, ~2.25M for IMDB).

Evaluation metrics

For evaluation on the similarity approximation task, we report the average Mean Square Error (MSE) and Mean Absolute Error (MAE). For evaluation on the retrieval task, we report Spearman's Rank Correlation Coefficient (ρ) [126], Kendall's Rank Correlation Coefficient (τ) [72], and Precision at k (P@k).

[1] `https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data`

Comparison methods

We evaluate GOTSim against a variety of competing approaches, including various similarity-learning neural approaches and classic approximation approaches. Since our method is related to graph matching approaches and most of these graph matching methods are not specifically designed for the graph similarity task, we select a representative graph matching method (GMN) which has been successfully used to predict graph similarity in our experiments for a fair evaluation. Specifically, we compare our method against the following baselines:

- Bipartite matching [37, 111]: It is a GED approximation deduced from a linear sum assignment of the nodes of the two graphs, which can be efficiently computed in polynomial time.
- Hausdorff matching [39]: Hausdorff-matching is a quadratic-time approximation of graph edit distance based on Hausdorff matching, which underestimates the true distance.
- EmbMean, EmbMax: employ mean-readout and max-readout functions, respectively, on the node-level embeddings to get the graph-level embedding.
- EmbGated [84]: EmbGated employs a projection on the node-level embeddings through a gated network, followed by mean-readout to get the graph-level embedding.
- GMN [84]: a representative graph-matching method which learns a soft matching between nodes.
- SimGNN [4]: SimGNN employs a differentiable attention-based readout function and a non-differentiable histogram function on the node-level embeddings to get the graph-level embedding.
- GraphSim [5]: GraphSim estimates the similarity function by using CNN on the similarity matrices of the node-embeddings.

Implementation details

For a fair comparison, all learning-based neural methods use GCN to learn the node embeddings. We the same network architecture for the proposed method and the learning-based graph similarity baselines, for all the datasets. For the GCN-based baselines, the number of GCN layers is 3 and the activation function is ReLU. For GraphSim and SimGNN, we used the same network architectures as reported in their papers.

7.4.2 Effectiveness of Graph Similarity Approximation (RQ1)

Table 7.3: Results of graph similarity approximation when training with GED targets on AIDS and LINUX. The ground-truth targets are provided by A^* algorithm (ExactGT). The values are in 10^{-3} unit.

	AIDS		LINUX	
	MSE	MAE	MSE	MAE
ExactGT	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Bipartite	130.68 ± 9.91	337.28 ± 16.11	266.11 ± 5.60	485.26 ± 6.66
Hausdorff	77.48 ± 5.64	255.39 ± 11.58	58.07 ± 1.46	221.84 ± 3.41
EmbMean	8.34 ± 0.50	69.60 ± 2.15	15.29 ± 1.92	151.12 ± 4.23
EmbMax	9.37 ± 0.24	74.44 ± 0.90	13.11 ± 1.88	140.73 ± 4.27
EmbGated	5.92 ± 0.32	65.46 ± 2.35	14.54 ± 2.01	119.01 ± 5.17
GMN	5.01 ± 0.25	61.23 ± 6.56	7.23 ± 0.94	55.78 ± 3.38
SimGNN	2.70 ± 0.30	38.34 ± 0.30	4.43 ± 0.62	50.41 ± 3.52
GraphSim	8.60 ± 0.33	37.06 ± 0.21	4.75 ± 0.78	45.72 ± 5.12
GOTSim	**2.36 ± 0.12**	**35.19 ± 0.15**	**4.25 ± 0.60**	**44.27 ± 3.23**

Table 7.4: Results of graph similarity approximation when training with GED targets on PTC and IMDB. The ground-truth target of each pair is the minimum GED returned by Bipartite and Beam. The values are in 10^{-3} unit.

	PTC		IMDB	
	MSE	MAE	MSE	MAE
ExactGT	–	–	–	–
Bipartite	60.68 ± 1.10	177.72 ± 3.36	13.22 ± 0.60	69.23 ± 3.93
Hausdorff	171.57 ± 7.94	388.30 ± 9.62	24.98 ± 1.71	90.91 ± 4.39
EmbMean	10.98 ± 1.22	79.92 ± 7.07	63.72 ± 2.37	102.63 ± 4.24
EmbMax	10.60 ± 1.52	80.18 ± 8.29	54.74 ± 1.94	98.12 ± 3.38
EmbGated	5.71 ± 2.21	51.05 ± 16.10	6.28 ± 3.23	99.23 ± 4.87
GMN	5.82 ± 1.89	56.7 ± 8.20	74.12 ± 4.22	168.02 ± 9.73
SimGNN	1.98 ± 0.43	27.85 ± 3.02	9.05 ± 4.12	78.01 ± 3.01
GraphSim	5.85 ± 0.83	55.17 ± 4.79	6.87 ± 4.02	111.39 ± 7.96
GOTSim	**1.90 ± 0.43**	**26.79 ± 0.43**	**5.92 ± 3.19**	**75.31 ± 3.33**

In this section, we present the results of graph similarity estimation and graph retrieval ranking. For the GED problem, Tables 7.3 and 7.4 show the similarity score estimation results while Tables 7.5 and 7.6 show the ranking results. Similarly, for the MCS problem, Tables 7.7 and 7.8 show the similarity score estimation results while Tables 7.9 and 7.10 show the similarity score estimation and ranking results for the MCS problem. GOTSim achieves the lowest estimation error, compared to other approaches (see similarity score estimation results in Tables 7.3, 7.4, 7.7, and 7.8). In graph retrieval, GOTSim also shows the highest ranking performance across all the datasets. Note that, in the PTC and IMDB experiments

Table 7.5: Graph similarity retrieval results when training with GED targets on AIDS and LINUX. The ground-truth targets are provided by A^* (ExactGT).

	AIDS			LINUX		
	τ	ρ	P@10	τ	ρ	P@10
ExactGT	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Bipartite	0.26 ± 0.02	0.37 ± 0.02	0.32 ± 0.04	0.32 ± 0.02	0.43 ± 0.03	0.45 ± 0.03
Hausdorff	0.48 ± 0.02	0.63 ± 0.03	0.54 ± 0.05	0.78 ± 0.01	0.88 ± 0.01	0.78 ± 0.05
EmbMean	0.41 ± 0.04	0.53 ± 0.04	0.60 ± 0.10	0.55 ± 0.02	0.60 ± 0.02	0.51 ± 0.01
EmbMax	0.42 ± 0.02	0.57 ± 0.01	0.61 ± 0.06	0.57 ± 0.03	0.65 ± 0.01	0.71 ± 0.01
EmbGated	0.43 ± 0.02	0.66 ± 0.02	0.70 ± 0.04	0.58 ± 0.03	0.88 ± 0.01	0.81 ± 0.01
GMN	0.47 ± 0.05	0.69 ± 0.02	0.72 ± 0.02	0.78 ± 0.03	0.88 ± 0.01	0.80 ± 0.01
SimGNN	0.67 ± 0.03	0.82 ± 0.02	0.84 ± 0.04	0.80 ± 0.01	0.92 ± 0.01	0.82 ± 0.05
GraphSim	0.68 ± 0.03	0.57 ± 0.03	0.76 ± 0.03	0.83 ± 0.02	0.92 ± 0.02	0.84 ± 0.03
GOTSim	**0.72 ± 0.02**	**0.86± 0.02**	**0.87± 0.03**	**0.89 ± 0.02**	**0.92± 0.01**	**0.86± 0.02**

Table 7.6: Graph similarity retrieval results when training with GED targets on PTC and IMDB. The ground-truth target is the minimum GED returned by Bipartite and Beam.

	PTC			IMDB		
	τ	ρ	P@10	τ	ρ	P@10
ExactGT	–	–	–	–	–	–
Bipartite	0.65 ± 0.02	0.82 ± 0.02	0.79 ± 0.03	0.85 ± 0.01	0.92 ± 0.01	0.97 ± 0.01
Hausdorff	0.78 ± 0.01	0.91 ± 0.01	0.90 ± 0.02	0.64 ± 0.01	0.76 ± 0.01	0.88 ± 0.04
EmbMean	0.16 ± 0.04	0.23 ± 0.06	0.41 ± 0.08	0.56 ± 0.03	0.09 ± 0.02	0.09 ± 0.02
EmbMax	0.21 ± 0.03	0.29 ± 0.03	0.49 ± 0.03	0.58 ± 0.02	0.10 ± 0.01	0.23 ± 0.02
EmbGated	0.66 ± 0.10	0.81 ± 0.13	0.84 ± 0.08	0.60 ± 0.03	0.82 ± 0.01	0.41 ± 0.01
GMN	0.40 ± 0.02	0.71 ± 0.03	0.70 ± 0.04	0.46 ± 0.03	0.62 ± 0.02	0.35 ± 0.03
SimGNN	0.80 ± 0.02	0.93 ± 0.01	0.91 ± 0.01	0.71 ± 0.03	0.66 ± 0.02	0.63 ± 0.02
GraphSim	0.79 ± 0.01	0.91 ± 0.01	0.91± 0.02	0.74 ± 0.02	0.71± 0.02	0.66 ± 0.02
GOTSim	**0.82 ± 0.01**	**0.95± 0.01**	**0.94± 0.01**	**0.80 ± 0.03**	**0.85± 0.01**	**0.73± 0.02**

reported in Tables 7.4 and 7.6, Bipartite and Hausdorff provide the approximate upperbound on the ranking performance because the ground-truth targets are created by the minimum of the estimated GEDs of Beam and Bipartite. It is computationally very expensive to compute the exact ground-truth in these datasets, whose graphs have much higher number of nodes than the graphs in AIDS and LINUX[2]. Note that, all these methods are extremely efficient than classical methods (Bipartite and Hausdorff), which require a significant amount of computation. When the exact ground-truth can be computed (AIDS and LINUX for GED in Tables 7.3 and 7.6), GOTSim outperforms Bipartite and Hausdorff. Improvements of our models over the compared methods are statistically significant according to the corresponding paired t-tests (p-value < 0.01).

[2]Both GOTSim and GraphSim have very similar running times, which are about 3%-5% longer than SimGNN. It is expected that SimGNN is slightly faster than GOTSim and GraphSim which are more expressive models with superior performance.

Table 7.7: Results of graph similarity approximation when training with MCS targets on AIDS and LINUX. The ground-truth targets are provided by MCSPLIT algorithm (EXACTGT). Note that BIPARTITE is not included because it is not an MCS-estimation algorithm. The values are in 10^{-3} unit.

	AIDS		LINUX	
	MSE	MAE	MSE	MAE
EXACTGT	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
EMBMEAN	7.83 ± 1.19	71.65 ± 6.44	1.11 ± 0.41	30.46 ± 1.27
EMBMAX	5.84 ± 0.23	61.76 ± 1.62	1.07 ± 0.37	30.45 ± 1.16
EMBGATED	5.91 ± 0.46	62.19 ± 3.50	1.03 ± 0.32	56.49 ± 4.03
GMN	3.73 ± 0.32	48.55 ± 2.32	0.90 ± 0.06	28.39 ± 2.57
SIMGNN	3.65 ± 0.65	47.25 ± 4.67	0.71 ± 0.06	19.90 ± 1.07
GRAPHSIM	7.28 ± 0.79	68.93 ± 3.97	0.69 ± 0.23	22.18 ± 1.35
GOTSIM	**2.26 ± 0.53**	**32.94 ± 3.90**	**0.65 ± 0.12**	**21.78 ± 1.01**

Table 7.8: Results of graph similarity approximation when training with MCS targets on PTC and IMDB. The ground-truth targets are provided by MCSPLIT algorithm (EXACTGT). Note that BIPARTITE is not included because it is not an MCS-estimation algorithm. The values are in 10^{-3} unit.

	PTC		IMDB	
	MSE	MAE	MSE	MAE
EXACTGT	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
EMBMEAN	8.22 ± 0.68	72.93 ± 3.65	8.63 ± 0.59	109.12 ± 3.75
EMBMAX	7.92 ± 0.70	72.53 ± 4.49	11.84 ± 0.55	117.32 ± 3.99
EMBGATED	5.32 ± 0.65	69.89 ± 2.37	4.59 ± 0.50	98.75 ± 3.45
GMN	5.01 ± 0.65	49.00 ± 0.94	3.75 ± 0.23	50.23 ± 3.47
SIMGNN	3.09 ± 0.70	42.14 ± 5.07	2.66 ± 0.31	21.78 ± 3.23
GRAPHSIM	8.61 ± 0.72	39.32 ± 1.48	2.60 ± 0.35	21.65 ± 2.89
GOTSIM	**2.97 ± 0.65**	**38.17 ± 1.57**	**2.38 ± 0.40**	**21.12 ± 1.37**

A Case Study on Graph Ranking: We present the ranking results of two example queries on the AIDS (labeled) and IMDB (unlabeled) datasets. In each example, the ground-truth ranking results (GT) are provided for comparison. In Figure 7.11, when GOTSIM approximates the lower bound of BIPARTITE and BEAM (IMDB dataset), GOTSIM ranks several isomorphic graphs (with respect to the query) in the database on the top of the list. GOTSIM's retrieved graphs are as relevant to the query as those graphs retrieved by BIPARTITE. However, it is clear that when the exact ground truth is used as the targets (AIDS dataset), GOTSIM top-ranked, retrieved graphs are very close to the ground truths.

Table 7.9: Graph similarity retrieval results when training with MCS targets on AIDS and LINUX. The ground-truth targets are provided by Mcsplit algorithm (ExactGT). Note that Bipartite is not included because it is not an MCS-estimation algorithm.

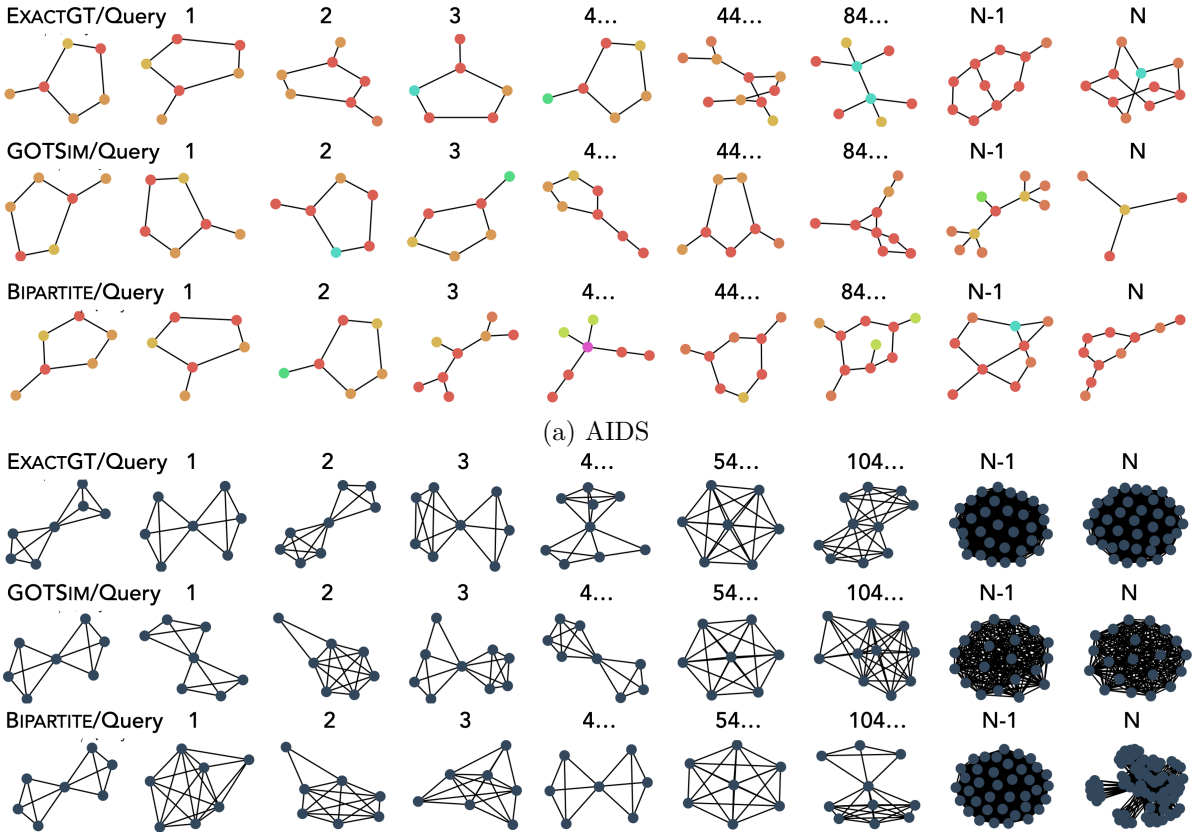
	AIDS			LINUX		
	τ	ρ	P@10	τ	ρ	P@10
ExactGT	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
EmbMean	0.36 ± 0.09	0.46 ± 0.12	0.20 ± 0.04	0.58 ± 0.03	0.40 ± 0.02	0.23 ± 0.01
EmbMax	0.50 ± 0.03	0.64 ± 0.03	0.31 ± 0.01	0.57 ± 0.02	0.41 ± 0.02	0.25 ± 0.01
EmbGated	0.51 ± 0.05	0.65 ± 0.05	0.32 ± 0.04	0.56 ± 0.04	0.43 ± 0.02	0.29 ± 0.02
GMN	0.57 ± 0.02	0.71 ± 0.03	0.45 ± 0.04	0.35 ± 0.03	0.37 ± 0.02	0.25 ± 0.03
SimGNN	0.56 ± 0.04	0.72 ± 0.05	0.47 ± 0.06	0.47 ± 0.02	0.58 ± 0.03	0.59 ± 0.09
GraphSim	0.57 ± 0.04	0.61 ± 0.05	0.27 ± 0.02	0.56 ± 0.02	0.62 ± 0.04	0.65 ± 0.05
GOTSim	**0.58 ± 0.03**	**0.82± 0.03**	**0.67± 0.02**	**0.61 ± 0.01**	**0.68± 0.03**	**0.79± 0.03**

Table 7.10: Graph similarity retrieval results when training with MCS targets on PTC and IMDB. The ground-truth targets are provided by Mcsplit algorithm (ExactGT). Note that Bipartite is not included because it is not an MCS-estimation algorithm.

	PTC			IMDB		
	τ	ρ	P@10	τ	ρ	P@10
ExactGT	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
EmbMean	0.32 ± 0.06	0.42 ± 0.70	0.30 ± 0.01	0.50 ± 0.02	0.67 ± 0.03	0.27 ± 0.04
EmbMax	0.45 ± 0.02	0.57 ± 0.03	0.33 ± 0.04	0.45 ± 0.02	0.38 ± 0.02	0.10 ± 0.03
EmbGated	0.47 ± 0.03	0.72 ± 0.05	0.51 ± 0.08	0.53 ± 0.03	0.62 ± 0.02	0.38 ± 0.02
GMN	0.27 ± 0.03	0.39 ± 0.03	0.31 ± 0.04	0.45 ± 0.02	0.39 ± 0.02	0.28 ± 0.02
SimGNN	0.61 ± 0.04	0.61 ± 0.04	0.61 ± 0.06	0.78 ± 0.03	0.63 ± 0.01	0.49 ± 0.02
GraphSim	0.63 ± 0.02	0.54 ± 0.04	0.42 ± 0.04	0.79 ± 0.01	0.65 ± 0.02	0.50 ± 0.02
GOTSim	**0.69 ± 0.02**	**0.73± 0.03**	**0.65± 0.02**	**0.81 ± 0.02**	**0.73± 0.03**	**0.52± 0.02**

7.4.3 Interpretable Graph Matching (RQ2)

In this section, we demonstrate how well GOTSim learns the ground-truth node-assignment in GED estimation, similar to that of classical methods, including the exact A^* (ExactGT) and the approximate Bipartite approaches. Specifically, we visualize the node matching between a pair of graphs which has the optimal assignment cost. Figure 7.12 shows examples of the AIDS and PTC datasets (with labeled nodes) and examples of the LINUX and IMDB datasets (with unlabeled nodes). The selected pair of graphs have similar number of nodes for simpler visualization and discussion. As we can observe in this figure, GOTSim learns a more similar matching to ExactGT than the matching computed by Bipartite, one of the most representative classical matching algorithms. In the unlabeled graph datasets (LINUX and IMDB), GOTSim recovers almost the exact same ground-truth node matchings between a pair of graphs. In the multi-labeled graph dataset, AIDS and PTC, while it is more difficult to visually investigate the node matchings, we can still observe that GOTSim matches nodes



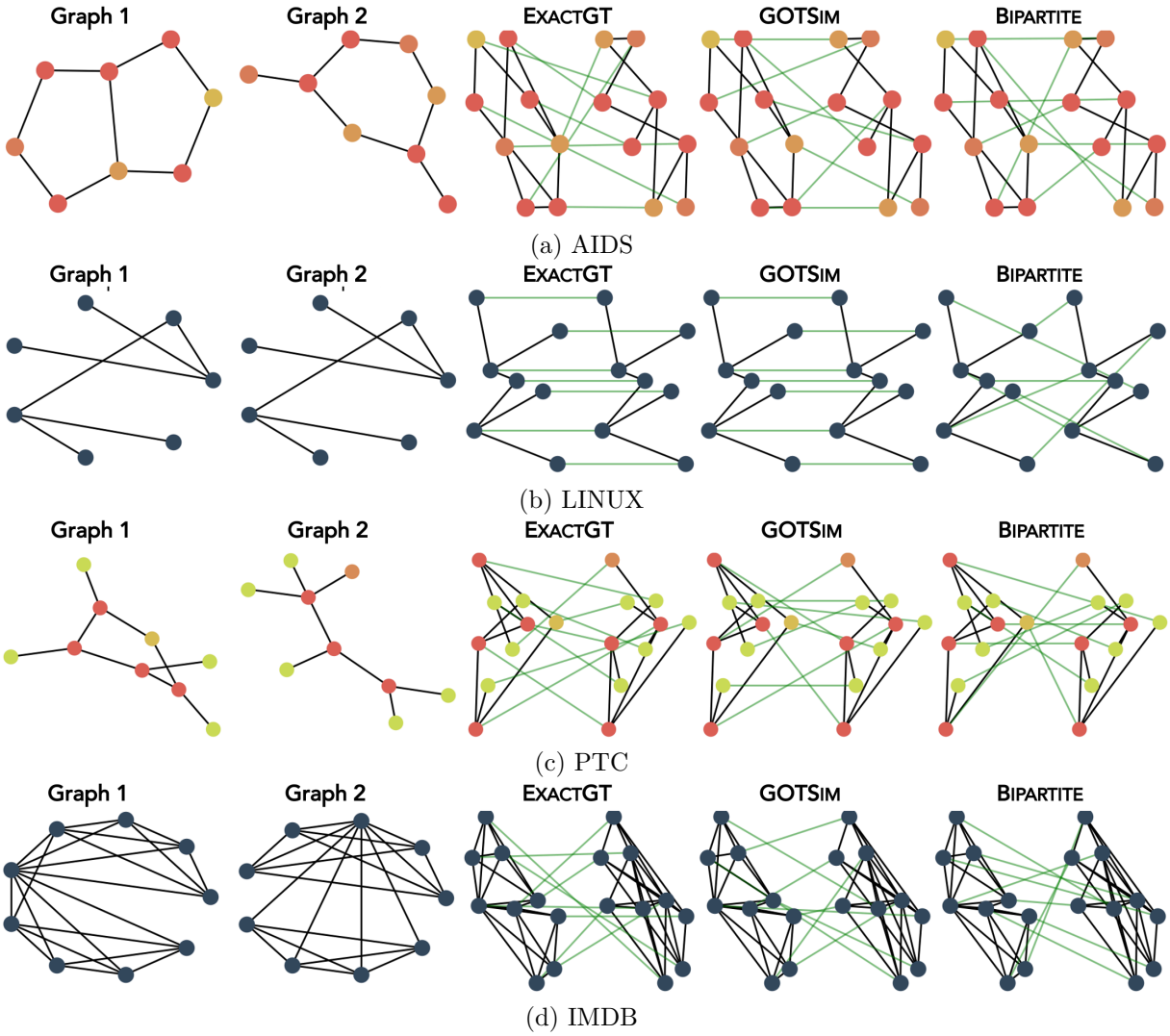
(a) AIDS

(b) IMDB

Figure 7.11: A sample of ranking results under the GED metric on AIDS (labeled) and IMDB (unlabeled) datasets. Each query graph (first column) is presented with a ranked list of graphs. Node labels are color-coded in AIDS.

with similar types better than that of Bipartite and is also closer to the ground-truth matchings. In the PTC and IMDB datasets, the result is even more significant because the training labels of all the learning based methods, including GOTSim, are provided by Bipartite and Beam. This implies that GOTSim generalizes better than the non-learning based approaches.

Different from soft graph matching models such as GMN [84] and other approaches [17, 100], GOTSim's matching is highly interpretable because we can easily see the injective mapping between the nodes of one graph to the other. Note that, this experiment is not possible with GraphSim and SimGNN.



(a) AIDS

(b) LINUX

(c) PTC

(d) IMDB

Figure 7.12: Graph matching for GOTSim on the last GCN layer on AIDS (labeled), PTC (labeled), LINUX (unlabeled), and IMDB (unlabeled) datasets. For each pair of graphs, the ExactGT matching is provided by A^*. The cross-graph matchings are shown in green. AIDS Node labels are color-coded.

7.5 Summary

In this chapter, we studied the problem of graph similarity and proposed a learning-based graph similarity computation method. Compared to the standard graph prediction problem, graph similarity prediction poses unique challenges and has potential advantages. For example, learning the similarity function directly from the set of node embeddings is non-trivial when there does not exist a canonical ordering of the nodes in the graphs. The

proposed GOTSim model directly compares graphs using their (bag of) node embedding vectors which can be learned by using a graph neural network framework. GOTSim directly solves the optimal assignment problem on a novel cost matrix formulation which accounts for node substitution, addition, and insertion. Computationally, the optimal assignment objective can be solved efficiently in polynomial time. Unlike existing optimal assignment based approaches, this objective has a stable optimal matching solution and is differentiable (which allows GOTSim to be efficiently trained using gradient learning). GOTSim also has a distinct feature: it provides an interpretable prediction. When predicting the similarity between a pair of graphs, domain experts can also understand how the model makes its predictions by visually inspecting the optimal matching between the nodes in the two graphs. To the best of our knowledge, GOTSim is the first learning-based graph-similarity method that provides such interpretable results. GOTSim can help in advancing or confirming the domain-specific knowledge in domains which involve graphs.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This Thesis focused on the development of efficient and robust retrieval models for large-scale retrieval applications in several domains, including computational advertising, text search, image search, and graph search. Specifically, we focused on four major challenges: (1) domain-independent hashing models, (2) efficient training of the hash functions, (3) better generalization and robustness of the hash functions, and (4) explainable retrieval of structured objects.

In Chapters 3, 4, and 5, we presented novel adversarial autoencoders which can efficiently preserve the locality information of the data in the discrete hash codes. The autoencoders achieve this objective by learning to reconstruct the input. This is fundamentally different from previous hashing approaches, which rely on domain-specific heuristics in order to preserve the input locality. The advantage of this approach is that the model designers only need to focus on the architectures of the encoder and decoder. With the rapid progress of deep neural networks in recent years, there are a large number of powerful architectures for several domains. For example, in Chapter 3, we presented an MLP encoder with a factorization layer for the hash function and an MLP decoder for data reconstruction. In Chapter 4, we presented both RNNs and CNNs as encoders and decoders. In Chapter 5, we presented a CNN encoder and a CNN decoder. Across these domains, there are minimal changes in the objective functions to preserve the input's locality information.

To learn the hash functions with low-quantization error and bit-balance, in Chapter 3, we presented a novel adversarial regularization framework. The adversarial regularization "matches" the latent space of the autoencoder with a Bernoulli distribution. This discrete distribution is chosen so that the regularization results in a near-optimal distribution of the data points into the available hash codes, effectively achieving hash code balance for the retrieval problem.

In Chapter 4, we built upon the adversarial autoencoder proposed in Chapter 3 and proposed an efficient algorithm to train the adversarial autoencoder. Specifically, we replaced the Jensen-Shannon divergence, which is known to be unsuitable for distributions with non-overlapping supports and has training problems such as mode collapses, with an Optimal-Transport formulation of the Wasserstein distance in the optimization objective. The proposed OT formulation results in a single optimization objective, instead of the previously proposed minimax objective. We showed that such improvement resulted in a model which was easier to train and achieved better retrieval performance in practice.

While the single-optimization model is easier to train, the Wasserstein distance, as well as the Jensen-Shannon divergence, has a high sample requirement. Furthermore, the OT estimate of the Wasserstein distance is computationally expensive. Chapter 5 considered a variant of the Wasserstein distance called Sliced Wasserstein Distance, and presented an efficient estimation algorithm which exploits the properties of the learned discrete space. The proposed Sliced Wasserstein Distance has a polynomial sample complexity and a significantly lower computational cost than the previous approaches. We showed the effectiveness of the proposed approach for the image retrieval task.

In Chapter 6, we discussed supervised hashing methods and presented a multipurpose generative hashing network that can learn a better hash function with limited labeled data. The proposed network is an energy-based generative model, which is solved via maximum-likelihood estimation (MLE). MLE results in more stable training than that of the implicit, adversarial models, GANs. In the image retrieval domain, we showed that the learned hash function also generalized better, resulting in better retrieval performance, and exhibited several desired characteristics such as the robustness toward out-of-distribution data and data corruption.

In Chapter 7, we extended the idea of distributional matching in adversarial models to measure the difference between two sets of vectors. We presented a novel graph-similarity estimation model which estimates the similarity score via the optimal transformation (or alignment) costs between two sets of vectors representing the graphs. Besides its high-performance similarity estimation ability, the proposed model has a distinct feature: it provides an interpretable prediction. When predicting the similarity between a pair of graphs, domain experts can also understand how the model makes its predictions by visually inspecting the optimal matching between the nodes in the two graphs, which helps in advancing or confirming the domain-specific knowledge in domains that involve graphs.

This Thesis makes a step towards leveraging generative models, especially their computational solutions, for a variety of retrieval problems, where the key bottlenecks are the computational efficiency, data efficiency, and interpretability of the models. We envision that this work will serve as a motivation for other applications that requires latent models or rely on limited labeled training data, which is expensive and time-consuming.

8.2 Future works

There are several interesting future directions that can follow this Thesis. In this section, we discuss some possible directions. The first direction is to combine the proposed unsupervised adversarial autoencoders and supervised EBMs in a semi-supervised learning-to-hash setting. This is a useful problem to solve because we can further utilize the availability of unlabeled data in order to improve the generalization of the proposed supervised generative hashing approach. Another possible future work is to improve the training process of the supervised generative EBMs by considering a generator, which replaces the replay buffers in training the EBMs. This is different from GANs: while the generator and discriminator in GANs play an adversarial game, the generator plays a cooperative game with the discriminator [144]. One of the challenges is to scale up this cooperative game to more complex image datasets that are widely used in hashing. Finally, as this Thesis has shown the possibility of learning an injective mapping between a pair of graphs, the proposed computational method can be extended to other applications that involve graphs [27]. For example, in the textual entailment task [96, 122], which determines whether a text logically follows from another text, providing an explanation for the underlying reasoning process is an important requirement. Our proposed alignment idea can be used to solve this problem, by first building a commonsense knowledge graph from the lexical definitions of the texts, then learning an alignment between these two graphs. This learned alignment can hopefully help explain the underlying reasoning process.

Bibliography

[1] K. Aggarwal, M. Kirchmeyer, P. Yadav, S. S. Keerthi, and P. Gallinari. Conditional generative adversarial networks for regression. *ArXiv190512868 Cs Stat.(10)*, 2019.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[3] S. Arora, R. Ge, Y. Liang, T. Ma, and Y. Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232. PMLR, 06–11 Aug 2017.

[4] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 384–392, New York, NY, USA, 2019. Association for Computing Machinery.

[5] Y. Bai, H. Ding, K. Gu, Y. Sun, and W. Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3219–3226, Apr 2020.

[6] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[7] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[8] S. Berretti, A. Del Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 23(10):1089–1094, Oct 1999.

[9] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2018.

[10] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[11] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse. Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression. *Computers & Graphics*, 71:77–87, 2018.

[12] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[13] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment problems: revised reprint*. Society for Industrial and Applied Mathematics, 2012.

[14] Y. Cao, B. Liu, M. Long, and J. Wang. Hashgan: Deep learning to hash with pair conditional wasserstein gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1287–1296, 2018.

[15] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[16] S. Chaidaroon and Y. Fang. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM, 2017.

[17] L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, and J. Liu. Graph optimal transport for cross-domain alignment. In *International Conference on Machine Learning*, pages 1542–1553. PMLR, 2020.

[18] 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. ACM, 2016.

[19] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[20] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pages 1–9, 2009.

[21] B. Dai, R. Guo, S. Kumar, N. He, and L. Song. Stochastic generative hashing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 913–922. JMLR. org, 2017.

[22] I. Deshpande, Y.-T. Hu, R. Sun, A. Pyrros, N. Siddiqui, S. Koyejo, Z. Zhao, D. Forsyth, and A. G. Schwing. Max-sliced wasserstein distance and its use for gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10648–10656, 2019.

[23] I. Deshpande, Z. Zhang, and A. G. Schwing. Generative modeling using the sliced wasserstein distance. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3483–3491, 2018.

[24] K. G. Dizaji, F. Zheng, N. S. Nourabadi, Y. Yang, C. Deng, and H. Huang. Unsupervised deep generative adversarial hashing network. In *CVPR 2018*, 2018.

[25] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, pages 219–234. Springer, 2016.

[26] K. D. Doan, S. Manchanda, S. Badirli, and C. K. Reddy. Image hashing by minimizing discrete component-wise wasserstein distance. *arXiv preprint arXiv:2003.00134*, 2020.

[27] K. D. Doan, S. Manchanda, S. Mahapatra, and C. K. Reddy. Interpretable graph similarity computation via differentiable optimal alignment of node embeddings. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 665–674, New York, NY, USA, 2021. Association for Computing Machinery.

[28] K. D. Doan, S. Manchanda, F. Wang, S. K. Selvaraj, A. Bhowmik, and C. K. Reddy. Image generation via minimizing fréchet distance in discriminator feature space. *arXiv preprint arXiv:2003.11774*, 2020.

[29] K. D. Doan and C. K. Reddy. Efficient implicit unsupervised text hashing using adversarial autoencoder. In *Proceedings of The Web Conference 2020*, pages 684–694, 2020.

[30] K. D. Doan, P. Yadav, and C. K. Reddy. Adversarial factorization autoencoder for look-alike modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2803–2812, 2019.

[31] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[32] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. 2017.

[33] J. Donahue and K. Simonyan. Large scale adversarial representation learning. volume 32, pages 10542–10552, 2019.

[34] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li. Modeling lsh for performance tuning. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 669–678. ACM, 2008.

[35] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[36] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2475–2483, 2015.

[37] S. Fankhauser, K. Riesen, and H. Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 102–111. Springer, 2011.

[38] M. Fey, J. E. Lenssen, C. Morris, J. Masci, and N. M. Kriege. Deep graph matching consensus. In *International Conference on Learning Representations*, 2019.

[39] A. Fischer, R. Plamondon, Y. Savaria, K. Riesen, and H. Bunke. A hausdorff heuristic for efficient computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 83–92. Springer, 2014.

[40] J. H. Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[41] 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.

[42] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9155–9164, 2018.

[43] C. Garcia-Hernandez, A. Fernandez, and F. Serratosa. Ligand-based virtual screening using graph edit distance as molecular similarity measure. *Journal of chemical information and modeling*, 59(4):1410–1421, 2019.

[44] C. Garcia-Hernandez, A. Fernandez, and F. Serratosa. Learning the edit costs of graph edit distance applied to ligand-based virtual screening. *Current topics in medicinal chemistry*, 20(18):1582–1592, 2020.

[45] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *European Conference on Computer Vision*, pages 250–264. Springer, 2014.

[46] A. Genevay, G. Peyré, and M. Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617, 2018.

[47] K. Ghasedi Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, and H. Huang. Unsupervised deep generative adversarial hashing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3664–3673, 2018.

[48] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[49] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 35(12):2916–2929, 2012.

[50] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[51] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[53] W. Grathwohl, K. Wang, J. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[54] U. Grenander, M. I. Miller, M. Miller, et al. *Pattern theory: from representation to inference*. Oxford university press, 2007.

[55] J. Gui and P. Li. R 2 sdh: Robust rotated supervised discrete hashing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1485–1493, 2018.

[56] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan. Fast supervised discrete hashing. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):490–496, 2017.

[57] J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer. Compact hashing with joint optimization of search accuracy and time. In *CVPR 2011*, pages 753–760. IEEE, 2011.

[58] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013.

[59] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[60] 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. ACM, 2017.

[61] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2957–2964. IEEE, 2012.

[62] C. Huang, C. Change Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5175–5184, 2016.

[63] S. Huang, Y. Xiong, Y. Zhang, and J. Wang. Unsupervised triplet hashing for fast image retrieval. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 84–92. ACM, 2017.

[64] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[65] A. Iohara, T. Ogawa, and T. Tanaka. Generative model based on minimizing exact empirical wasserstein distance, 2019.

[66] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[67] Q.-Y. Jiang and W.-J. Li. Asymmetric deep supervised hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[68] J. Jiménez-Luna, F. Grisoni, and G. Schneider. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10):573–584, 2020.

[69] A. Joly and O. Buisson. Random maximum margin hashing. In *CVPR 2011*, pages 873–880. IEEE, 2011.

[70] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[71] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[72] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[73] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

[74] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of 2nd International Conference on Learning Representations*, 2014.

[75] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[76] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[77] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto. 2009*, 2009.

[78] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[79] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050, 2009.

[80] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3270–3278, 2015.

[81] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[82] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[83] Q. Li, Z. Sun, R. He, and T. Tan. Deep supervised discrete hashing. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2479–2488, 2017.

[84] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning*, pages 3835–3845. PMLR, 2019.

[85] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[86] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[87] V. E. Liong, J. Lu, L.-Y. Duan, and Y.-P. Tan. Deep variational and structural hashing. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):580–595, 2018.

[88] H. Liu, D. Pardoe, K. Liu, M. Thakur, F. Cao, and C. Li. Audience Expansion for Online Social Network Advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 165–174. ACM, 2016.

[89] H. Liu, D. Pardoe, K. Liu, M. Thakur, F. Cao, and C. Li. Audience Expansion for Online Social Network Advertising. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 165–174. ACM, 2016.

[90] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2074–2081. IEEE, 2012.

[91] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[92] Y. lun Du and I. Mordatch. Implicit generation and modeling with energy based models. In *NeurIPS*, 2019.

[93] M. Ma, Z. Wen, D. Chen, et al. A sub-linear massive-scale look-alike audience extension system. In *Proceedings of the 5th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2016.

[94] Q. Ma, E. Wagh, J. Wen, Z. Xia, R. Ormandi, and D. Chen. Score look-alike audiences. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 647–654. IEEE, 2016.

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

[96] B. MacCartney and C. D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, 2008.

[97] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *International Conference on Learning Representations*, 2016.

[98] S. Manchanda, D. K. Doan, P. Yadav, and S. S. Keerthi. Regression via implicit models and optimal transport cost minimization. *arXiv preprint arXiv:2003.01296*, 2020.

[99] S. Manchanda, D. Zheng, and G. Karypis. Schema-aware deep graph convolutional networks for heterogeneous graphs, 2021.

[100] H. P. Maretic, M. El Gheche, G. Chierchia, and P. Frossard. Got: an optimal transport framework for graph comparison. In *Advances in Neural Information Processing Systems*, pages 13876–13887, 2019.

[101] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[102] M. Neuhaus, K. Riesen, and H. Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 163–172. Springer, 2006.

[103] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Advances in Neural Information Processing Systems*, pages 5232–5242, 2019.

[104] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis. Matching node embeddings for graph similarity. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[105] Z. Qiu, Y. Pan, T. Yao, and T. Mei. Deep semantic hashing with generative adversarial networks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 225–234, 2017.

[106] Z. Qiu, Y. Pan, T. Yao, and T. Mei. Deep semantic hashing with generative adversarial networks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 225–234. ACM, 2017.

[107] Y. Qu, J. Wang, Y. Sun, and H. M. Holtan. Systems and methods for generating expanded user segments, Feb. 18 2014. US Patent 8,655,695.

[108] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[109] J. W. Raymond, E. J. Gardiner, and P. Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[110] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[111] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

[112] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. Neighbourhood consensus networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1658–1669, 2018.

[113] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[114] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[115] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[116] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[117] G. Schröder, M. Thiele, and W. Lehner. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, USA*, volume 23, page 53, 2011.

[118] D. Shasha, J. T. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 39–52, 2002.

[119] D. Shen, Q. Su, P. Chapfuwa, W. Wang, G. Wang, R. Henao, and L. Carin. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2041–2050, 2018.

[120] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 37–45, 2015.

[121] J. Shen, S. C. Geyik, and A. Dasdan. Effective audience extension in online advertising. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2099–2108. ACM, 2015.

[122] V. S. Silva, A. Freitas, and S. Handschuh. Xte: Explainable text entailment. *arXiv preprint arXiv:2009.12431*, 2020.

[123] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[124] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[125] M. Slaney, Y. Lifshits, and J. He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[126] C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471, 1987.

[127] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071, 2008.

[128] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, 19(10):1183–1193, 2003.

[129] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. 2017.

[130] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[131] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. In *6th International Conference on Learning Representations*, 2018.

[132] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[133] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[134] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[135] Q. Wang, D. Zhang, and L. Si. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 213–222. ACM, 2013.

[136] X. Wang, X. Ding, A. K. Tung, S. Ying, and H. Jin. An efficient graph indexing method. In *2012 IEEE 28th International Conference on Data Engineering*, pages 210–221. IEEE, 2012.

[137] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.

[138] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[139] P. Willett, J. M. Barnard, and G. M. Downs. Chemical similarity searching. *Journal of chemical information and computer sciences*, 38(6):983–996, 1998.

[140] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 2156–2162. AAAI Press, 2014.

[141] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *IEEE transactions on pattern analysis and machine intelligence*, 42(1):27–45, 2018.

[142] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[143] J. Xie, Y. Xu, Z. Zheng, S.-C. Zhu, and Y. Nian Wu. Generative pointnet: Energy-based learning on unordered point sets for 3d generation, reconstruction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2021.

[144] J. Xie, Z. Zheng, X. Fang, S.-C. Zhu, and Y. N. Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[145] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8629–8638, 2018.

[146] J. Xie, S.-C. Zhu, and Y. Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[147] Y. Xu, J. Xie, T. Zhao, C. Baker, Y. Zhao, and Y. N. Wu. Energy-based continuous inverse optimal control. *arXiv preprint arXiv:1904.05453*, 2019.

[148] P. Yanardag and S. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[149] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao. Semantic structure-based unsupervised deep hashing. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1064–1070, 2018.

[150] E. Yang, T. Liu, C. Deng, W. Liu, and D. Tao. Distillhash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2955, 2019.

[151] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):437–451, 2018.

[152] Z. Zeng, A. K. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

[153] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR'10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM, 2010.

[154] X. Zhang and Y. LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[155] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[156] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *AAAI*, pages 1369–1375, 2014.

[157] Y. Zhang, D. Shen, G. Wang, Z. Gan, R. Henao, and L. Carin. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179, 2017.

[158] J. Zhao, Y. Kim, K. Zhang, A. Rush, and Y. LeCun. Adversarially regularized autoencoders. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911. PMLR, 10–15 Jul 2018.

[159] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[160] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. volume 30, Mar 2016.

[161] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[162] S. C. Zhu and D. Mumford. Grade: Gibbs reaction and diffusion equations. In *International Conference on Computer Vision (ICCV)*, pages 847–854, 1998.