November 2017

Graph-based Latent Embedding, Annotation and Representation Learning in Neural Networks for Semi-supervised and Unsupervised Settings

Ismail Ozsel Kilinc
University of South Florida, ozsel@mail.usf.edu

Graph-based Latent Embedding, Annotation and Representation Learning in Neural Networks for Semi-supervised and Unsupervised Settings

by

Ismail Ozsel Kilinc

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ismail Uysal, Ph.D.
Sanjukta Bhanja, Ph.D.
Selcuk Kose, Ph.D.
Lawrence Hall, Ph.D.
Umut Ozertem, Ph.D.

Date of Approval:
November 30, 2017

Keywords: Machine Learning, Deep Learning, Graph-based Regularization, Clustering, Auto-clustering Output Layer

DEDICATION

To mom, dad and Sule.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my major professor Dr. Ismail Uysal for his guidance and support in my study. His inspiring suggestions and meticulous feedback in every step of this dissertation enabled me to write it and made it an invaluable experience for me. It has been a pleasure to write this dissertation under his guidance.

I would also wish to express my sincere gratitude to my dissertation committee members Dr. Sanjukta Bhanja, Dr. Selcuk Kose, Dr. Larry Hall and Dr. Umut Ozertem as they kindly accepted to share their invaluable comments and helpful suggestions with me. I also thank to Dr. Kyle Reed as he kindly accepted to chair my dissertation defense meeting.

I express my dearest thanks to Sule Akdogan who did not leave me alone getting through the hardest times. Her presence and her belief in me have been reassuring throughout this study. I am deeply indebted to her for her understanding, patience, respect in what I am doing and encouragement.

Last but not least, most special thanks and love go to my family, Nevin and Tacettin, who have supported me in everything I have done in my life. It could have been impossible to write this dissertation without their love and support. They are the true possessors of my success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Machine learning has been immensely successful in supervised learning with outstanding examples in major industrial applications such as voice and image recognition. Following these developments, the most recent research has now begun to focus primarily on algorithms which can exploit very large sets of unlabeled examples to reduce the amount of manually labeled data required for existing models to perform well. In this dissertation, we propose graph-based latent embedding/annotation/representation learning techniques in neural networks tailored for semi-supervised and unsupervised learning problems. Specifically, we propose a novel regularization technique called Graph-based Activity Regularization (GAR) and a novel output layer modification called Auto-clustering Output Layer (ACOL) which can be used separately or collaboratively to develop scalable and efficient learning frameworks for semi-supervised and unsupervised settings.

First, singularly using the GAR technique, we develop a framework providing an effective and scalable graph-based solution for semi-supervised settings in which there exists a large number of observations but a small subset with ground-truth labels. The proposed approach is natural for the classification framework on neural networks as it requires no additional task calculating the reconstruction error (as in autoencoder based methods) or implementing zero-sum game mechanism (as in adversarial training based methods). We demonstrate that GAR effectively and accurately propagates the available labels to unlabeled examples. Our results show comparable performance with state-of-the-art generative approaches for this setting using an easier-to-train framework.

Second, we explore a different type of semi-supervised setting where a coarse level of labeling is available for all the observations but the model has to learn a fine, deeper level of latent annotations for each one. Problems in this setting are likely to be encountered in many domains such as text categorization, protein function prediction, image classification as well as in exploratory scientific studies such as medical and genomics research. We consider this setting as simultaneously performed supervised classification (per the available coarse labels) and unsupervised clustering (within each one of the coarse labels) and propose a novel framework combining GAR with ACOL, which enables the network to perform concurrent classification and clustering. We demonstrate how the coarse label supervision impacts performance and the classification task actually helps propagate useful clustering information between sub-classes. Comparative tests on the most popular image datasets rigorously demonstrate the effectiveness and competitiveness of the proposed approach.

The third and final setup builds on the prior framework to unlock fully unsupervised learning where we propose to substitute real, yet unavailable, parent- class information with pseudo class labels. In this novel unsupervised clustering approach the network can exploit hidden information indirectly introduced through a pseudo classification objective. We train an ACOL network through this pseudo supervision together with unsupervised objective based on GAR and ultimately obtain a k-means friendly latent representation. Furthermore, we demonstrate how the chosen transformation type impacts performance and helps propagate the latent information that is useful in revealing unknown clusters. Our results show state-of-the-art performance for unsupervised clustering tasks on MNIST, SVHN and USPS datasets with the highest accuracies reported to date in the literature.

CHAPTER 1: INTRODUCTION

Artificial intelligence (AI) is a flourishing field with many practical applications and active research topics such as automating routine work, understanding and interpreting speech, text and images, supporting basic scientific research and diagnosis in medicine [19]. In the early days of AI dominant approaches were rule-based systems where the knowledge needed to solve the tasks is explicitly hard-coded by a programmer. These kinds of approaches have been successful in solving problems that can be easily described by a small set of formal rules but can require a lot of search, such as playing chess; however, they failed to solve those that cannot be formally described, such as recognizing objects or speech, and those requiring an immense amount of knowledge, such as playing Go. While there are approximately 400 possible next moves in chess, for Go this number goes up to 130,000 [13]. Also, it is impossible for a programmer to hard-code the knowledge required to recognize faces through if-else based rules. Therefore, researchers have reached a consensus that, in order to behave in an intelligent way, AI systems need the ability to acquire their own knowledge from raw data, which is known as the field of machine learning, just as human-beings learn from experience.

Neural networks, when first introduced as universal estimators, created wide spread enthusiasm especially with innovative and bio-inspired machine learning methodologies such as error back-propagation [54]. However, their potential wasn't fully realized until greater availability of computational power to simulate networks orders of magnitude larger than their early counterparts and massive datasets to help train them. This approach to AI, called deep learning with reference to deep structures with many layers, has been proposed to mimic the operational and organizational behavior of the human brain, which works through

abstraction [3]. For example, objects or sounds are represented as electrical signals traveling through different types of connections with different strengths between neurons in visual or auditory cortexes respectively [46]. Deep learning takes inspiration from this and relies on higher-level representations of features (or characteristics) embedded in the data instead of human engineered characteristics.

One great example is in computer vision in regards to how methods have changed transformatively over the last several years. For instance, a particular implementation of deep learning called Convolutional Neural Networks (CNN) has been applied to the image classification problem with remarkable success [31]. Besides pure performance numbers, the most interesting finding was how expert human engineered features, such as textures and contrasts, which have been used for decades were ultimately represented at the deeper and hidden layers of the neural network without any explicit instructions. On the other hand, for tasks involving temporal input such as speech and text, Recurrent Neural Networks (RNN) are widely in use since they provide better prediction for the future elements thanks to the architecture implicitly keeping the information about past elements, such as Long Short Term Memory (LSTM) networks [34], [23].

Specifically in the last five year period, these models have successful produced many practical industrial applications for supervised learning problems where the overall task is to learn to map one vector (input) to another (label), given enough examples of the mapping. To achieve good performance, algorithms used for these applications require large amounts of labeled data whose labels are typically entered by a staff of human supervisors. However, labeling is mostly a challenging task that sometimes requires expert knowledge and in fact, especially in exploratory research, defines the scientific problem itself. Therefore, in recent years, researchers have focused approaches that are able to exploit a large set of unlabeled examples to reduce the amount of labeled data required for existing models to perform well. These approaches adopt some form of unsupervised or semi-supervised learning. In a

semi-supervised setting, the goal is to take advantage of a large set of unlabeled examples to improve the performance that is obtained using only labeled examples whose amount is typically much smaller than that of unlabeled ones. On the other hand, in an unsupervised setting, as there is no labeling available, the task is generally to determine the underlying distribution generating the dataset and to discover the unknown labels.

In this dissertation, we propose graph-based latent embedding, annotation and representation learning in neural networks for semi-supervised and unsupervised settings. Specifically, we propose a novel regularization technique called Graph-based Activity Regularization (GAR) and a novel output layer modification called Auto-clustering Output Layer (ACOL) which are separately and collaboratively used to develop scalable and efficient learning frameworks for semi-supervised and unsupervised settings. More specifically, major contributions of this dissertation are as follows:

- The first graph-based technique in literature which displays competitive performance with deep generative approaches based on the test performances on mid-size image datasets of digits, such as MNIST [35] and SVHN [44]
  - $6.98\%(\pm 0.82)$[1] vs. 4.28% [41] error rate on SVHN using a randomly chosen 1000 labeled examples with stratification
  - $1.56\%(\pm 0.09)$[1] vs. $0.93\%(\pm 0.07)$ [56] error rate on MNIST using a randomly chosen 100 labeled examples with stratification

  and statistically significantly outperforms all other graph-based methods
  - $1.56\%(\pm 0.09)$[1] vs. $7.75\%(\pm 0.07)$ [67] error rate on MNIST using a randomly chosen 100 labeled examples with stratification

  for semi-supervised learning.

[1] $\pm$ shows the 0.95 confidence interval

- The first model in literature which enables neural networks to learn previously unknown annotations in observations for which coarse labeling is available with empirical demonstrations of how inter-class differences can help explore subclasses of the provided coarse labeling.

- Statistically significant and superior performance with respect to other methods modified to operate in this particular setting based on the test performances on MNIST, i.e. mid-size image dataset of hand-written digits

  - $1.39\%(\pm 0.12)^1$ vs. 8.18% [25] error rate on MNIST when providing the coarse labeling as whether a digit is smaller or greater than 5

- The first graph-based technique in literature which outperforms all other known approaches for unsupervised clustering based on the test performances on mid-size image datasets of digits, such as MNIST [35] and SVHN [44]

  - $76.80\%(\pm 1.30)^1$ vs. $57.30\%(\pm 3.90)$ [24] clustering accuracy on SVHN
  - $98.32\%(\pm 0.08)^1$ vs. $98.40\%(\pm 0.40)$ [24] and 96.10 [72] clustering accuracy on MNIST

  and defines the current state-of-the-art for unsupervised clustering for the SVHN dataset, as observed through statistically significantly better result with a wide margin.

The following section describes the outline of the dissertation and briefly summarizes the proposed framework in each chapter.

1.1 Dissertation Outline

1.1.1 GAR: An Efficient and Scalable Graph-based Activity Regularization for Semi-supervised Learning

In Chapter 2, we propose a novel graph-based approach for the semi-supervised learning setting in which there exist a large number of observations but only a small subset of them have ground-truth labels. In this approach, the adjacency of the examples is inferred using the predictions of a neural network model, which is first initialized by supervised training using the small subset of labeled examples. Then, during the subsequent unsupervised portion of the training that propagates the available labels toward the unlabeled examples, the inferred adjacency matrix is simultaneously updated along with the predictions of the network. However, unlike traditional graph-based methods propagating the labels using an $m \times m$ adjacency matrix of m examples, the proposed approach propagates the labels through a scalable regularization objective defined on an $n \times n$ adjacency matrix of n output classes, where typically $n \ll m$. Ultimately, the proposed framework provides an effective and scalable graph-based solution which is natural for the classification framework on neural networks as it requires no additional task calculating the reconstruction error (as in autoencoder based methods) or implementing zero-sum game mechanism (as in adversarial training based methods). Our results show comparable performance with state-of-the-art generative approaches for semi-supervised learning using an easier-to-train framework.

1.1.2 Auto-clustering Output Layer: Automatic Learning of Latent Annotations in Neural Networks

In Chapter 3, we discuss a different type of semi-supervised setting: a coarse level of labeling is available for all observations but the model has to learn a fine level of latent annotation for each one of them. Problems in this setting are likely to be encountered in many domains such as text categorization, protein function prediction, image classification as

well as in exploratory scientific studies such as medical and genomics research. We consider this setting as simultaneously performed supervised classification (per the available coarse labels) and unsupervised clustering (within each one of the coarse labels) and propose a novel output layer modification called auto-clustering output layer (ACOL) that allows concurrent classification and clustering based on Graph-based Activity Regularization (GAR) technique. As the proposed output layer modification duplicates the softmax nodes at the output layer for each class, GAR allows for competitive learning between these duplicates on a traditional error-correction learning framework to ultimately enable a neural network to learn the latent annotations in this partially supervised setup. We demonstrate how the coarse label supervision impacts performance and helps propagate useful clustering information between sub-classes. Comparative tests on three image datasets MNIST, SVHN and CIFAR-100 rigorously demonstrate the effectiveness and competitiveness of the proposed approach.

1.1.3 Learning Latent Representations in Neural Networks for Unsupervised Clustering through Pseudo Supervision and Graph-based Activity Regularization

In Chapter 4, we propose a novel unsupervised clustering approach exploiting the hidden information that is indirectly introduced through a pseudo classification objective. Specifically, we randomly assign a pseudo parent-class label to each observation which is then modified by applying the domain specific transformation associated with the assigned label. Generated pseudo observation-label pairs are subsequently used to train a neural network with Auto-clustering Output Layer (ACOL) that introduces multiple softmax nodes for each pseudo parent-class. Due to the unsupervised objective based on Graph-based Activity Regularization (GAR) terms, softmax duplicates of each parent-class are specialized as the hidden information captured through the help of domain specific transformations is propagated during training. Ultimately we obtain a k-means friendly latent representation. Furthermore, we demonstrate how the chosen transformation type impacts performance and

helps propagate the latent information that is useful in revealing unknown clusters. Our results show state-of-the-art performance for unsupervised clustering tasks on MNIST, SVHN and USPS datasets, with the highest accuracies reported to date in the literature.

CHAPTER 2: GAR: AN EFFICIENT AND SCALABLE GRAPH-BASED ACTIVITY REGULARIZATION FOR SEMI-SUPERVISED LEARNING[1]

The idea of utilizing an auxiliary unsupervised task to help supervised learning dates back to 90s [59]. As an example to one of its numerous achievements, unsupervised pretraining followed by supervised fine-tuning was the first method to succeed in the training of fully connected architectures [22]. Although today it is known that unsupervised pretraining is not a must for successful training, this particular accomplishment played an important role enabling the current deep learning renaissance and has become a canonical example of how a learning representation for one task can be useful for another one [19]. There exist a variety of approaches to combine supervised and unsupervised learning in the literature. More specifically, the term *semi-supervised* is commonly used to describe a particular type of learning for applications in which there exists a large number of observations, where a small subset of them has ground-truth labels. Proposed approaches aim to leverage the unlabeled data to improve the generalization of the learned model and ultimately obtain a better classification performance.

One approach is to introduce additional penalization into training based on the reconstruction of the input through autoencoders [50]. Recently, there have been significant improvements in this field following the introduction of a different generative modeling technique which uses the variational equivalent of deep autoencoders integrating stochastic latent variables into the conventional architecture [29] [53]. First, [28] have shown that such modifications make generative approaches highly competitive for semi-supervised learning.

[1]This chapter has been submitted to peer-reviewed Elsevier Neurocomputing Journal and is now under the second revision.

Later, [37] further improved the results obtained using variational autoencoders by introducing auxiliary variables increasing the flexibility of the model. Furthermore, [52] has applied Ladder networks [61], a layer-wise denoising autoencoder with skip connections from the encoder to the decoder, for semi-supervised classification tasks.

On the other hand, these recent improvements have also motivated researchers to offer radically novel solutions such as virtual adversarial training [42] motivated by Generative Adversarial Nets [20] proposing a new framework corresponding to a minimax two-player game. Alternatively, [73] revisited graph-based methods with new perspectives such as invoking the embeddings to predict the context in the graph. Conventional graph-based methods aim to construct a graph propagating the label information from labeled to unlabeled observations and connecting similar ones using a graph Laplacian regularization in which the key assumption is that nearby nodes are likely to have the same labels [74], [7]. However, the requirement for eigenanalysis of the graph Laplacian severely limits the scalability of these approaches. In a different work, [67] have shown that the idea of combining an embedding-based regularizer with a supervised learner to perform semi-supervised learning can be generalized to deep neural networks and the resulting models can be trained by stochastic gradient descent. A possible bottleneck with this approach is that the optimization of the unsupervised part of the loss function requires precomputation of the weight matrix specifying the similarity or dissimilarity between the examples whose size grows quadratically with the number of examples. For example, a common approach to computing the similarity matrix is to use the k-nearest neighbor algorithm which is computationally very expensive for a large number of samples. Therefore, it is approximated using sampling techniques.

In this chapter, we propose a novel framework for semi-supervised learning which can be considered a variant of graph-based approach. This framework can be described as follows.

- Instead of a graph between examples, we consider a bipartite graph between examples and output classes. To define this graph, we use the predictions of a neural network

model initialized by a supervised training process that uses a small subset of samples with known labels.

- We use this bipartite graph to obtain two disjoint graphs, which are then employed to interpret two adjacencies: One between the examples and another between the output nodes. We introduce two regularization terms for the graph between the output nodes and during the unsupervised portion of training, the predictions of the network are updated only based on these two regularizers.

- These terms implicitly provide that the bipartite graph between the examples and the output classes becomes a biregular graph and the inferred graph between the examples becomes a disconnected graph of regular subgraphs. Ultimately, because of this observation, the predictions of the network yield embeddings that we try to find.

The proposed framework is naturally inductive, where predictions can be generalized to never-seen-before examples. More importantly, it is scalable and it can be applied to datasets regardless of the sample size or the dimensionality of the feature set. Furthermore, the entire framework operationally implements the same feedforward and backpropagation mechanisms of the state of the art deep neural networks as the proposed regularization terms are added to the loss function in the same way as adding standard $L1$, $L2$ regularizations [45] and similarly optimized using stochastic gradient descent [12].

2.1 Related Work

Consider a semi-supervised learning problem where out of m observations, corresponding ground-truth labels are only known for a subset of m_L examples and the labels of the complimentary subset of m_U examples are unknown where $m = m_L + m_U$ and typically $m_L \ll m_U$. Let $\boldsymbol{x}_{1:m}$ and $y_{1:m}$ denote the input feature vectors and the output predictions respectively and $t_{1:m_L}$ denote the available output labels. The main objective is to train a

classifier $f : \boldsymbol{x} \rightarrow y$ using all m observations that is more accurate than another classifier trained using only the labeled examples. Graph-based semi-supervised methods consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of which vertices $\mathcal{V}$ correspond to all m examples and edges $\mathcal{E}$ are specified by an $m \times m$ adjacency matrix $\boldsymbol{A}$ whose entries indicate the similarity between the vertices. There have been many different approaches about the estimation of the adjacency matrix $\boldsymbol{A}$. [74] derived $\boldsymbol{A}$ according to simple Euclidean distances between the samples while [67] precomputed $\boldsymbol{A}$ using a k-nearest neighbor algorithm. They also suggest that, in case of a sequential data, one can presume consecutive instances are also neighbors in the graph. [73], on the other hand, consider the specific case where $\boldsymbol{A}$ is explicitly given and represents additional information. The most important common factor in all these graph-based methods is the fact that $\boldsymbol{A}$ is a fixed matrix throughout the training procedure with the key assumption that nearby samples on $\mathcal{G}$, which is defined by $\boldsymbol{A}$, are likely to have the same labels. Hence, the generic form of the loss function for these approaches can be written as:

$$\sum_{i=1}^{m_L} \mathcal{L}\big(f(\boldsymbol{x}_i), t_i\big) + \lambda \sum_{i,j=1}^{m} \mathcal{U}\big(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j), A_{ij}\big) \tag{2.1}$$

where $\mathcal{L}(.)$ is the supervised loss function such as log loss, hinge loss or squared loss, $\mathcal{U}(.)$ is the unsupervised regularization (in which a multi-dimensional embedding $g(\boldsymbol{x}_i) = \boldsymbol{z}_i$ can also be replaced with one-dimensional $f(\boldsymbol{x}_i)$) and λ is a weighting coefficient between supervised and unsupervised metrics of the training $\mathcal{L}(.)$ and $\mathcal{U}(.)$. One of the commonly employed embedding algorithms in semi-supervised learning is Laplacian Eigenmaps [6] which describes the distance between the samples in terms of the Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is the diagonal degree matrix such that $D_{ii} = \sum_j A_{ij}$. Then, unsupervised regularization becomes:

$$\sum_{i,j=1}^{m} \mathcal{U}\big(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j), A_{ij}\big) = \sum_{i,j=1}^{m} A_{ij} ||g(\boldsymbol{x}_i) - g(\boldsymbol{x}_j)||^2 = \mathrm{Tr}(\boldsymbol{Z}^T \boldsymbol{L} \boldsymbol{Z}) \tag{2.2}$$

subject to the balancing constraint $\boldsymbol{Z}^T\boldsymbol{D}\boldsymbol{Z} = \boldsymbol{I}$, where $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_m]^T$. [74] used this regularization together with a nearest neighbor classifier while [7] integrates hinge loss to train an SVM. Both methods impose regularization on labels $f(\boldsymbol{x}_i)$. On the other hand, [67] employ a margin-based regularization by [21] such that

$$\mathcal{U}\big(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j), A_{ij}\big) = \begin{cases} ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||^2 & \text{if } A_{ij} = 1 \\ max(0, \gamma - ||f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)||^2) & \text{if } A_{ij} = 0 \end{cases} \tag{2.3}$$

to eliminate the balancing constraints and enable optimization using gradient descent. They also propose to learn multi-dimensional embeddings on neural networks such that $g(\boldsymbol{x}_i) = f^l(\boldsymbol{x}_i) = \boldsymbol{y}_i^l$, where $\boldsymbol{y}_i^l$ is the output of the l^{th} hidden layer corresponding to the i^{th} sample.

2.2 Proposed Framework

2.2.1 Bipartite Graph Approach

Instead of estimating the adjacency matrix $\boldsymbol{A}$ using an auxiliary algorithm such as nearest neighbor or auxiliary external knowledge, we propose to use the actual predictions of a neural network model initialized by a supervised training step using m_L labeled examples.

Suppose that after supervised training, predictions of the network, $\boldsymbol{B}$, for all m examples are obtained as an $m \times n$ matrix, where n is the number of output classes and B_{ij} is the probability of the i^{th} example belonging to j^{th} class. We observe that these predictions, indeed, define a bipartite graph $\mathcal{G}^* = (\mathcal{V}^*, \mathcal{E}^*)$ whose vertices $\mathcal{V}^*$ are m examples together with n output nodes. However, $\mathcal{V}^*$ can be divided into two disjoint sets $\mathcal{M}$ and $\mathcal{N}$, such that $\mathcal{G}^* = (\mathcal{M}, \mathcal{N}, \mathcal{E}^*)$, where $\mathcal{M}$ is the set of examples, $\mathcal{N}$ is the set of output nodes and an edge $\mathsf{e} \in \mathcal{E}^*$ connects an example $\mathsf{m} \in \mathcal{M}$ with an output node $\mathsf{n} \in \mathcal{N}$. As there is no lateral connection between m examples and between n output nodes, $(m + n) \times (m + n)$ adjacency

matrix $\boldsymbol{A}^*$ of graph $\mathcal{G}^*$ has the following form

$$\boldsymbol{A}^* = \begin{pmatrix} \mathbf{0}_{m\times m} & \boldsymbol{B}_{m\times n} \\ \boldsymbol{B}^T_{n\times m} & \mathbf{0}_{n\times n} \end{pmatrix} \tag{2.4}$$

where $\boldsymbol{B}$ corresponds to $m \times n$ biadjacency matrix of graph $\mathcal{G}^*$ which is by itself unique and sufficient to describe the entire $\mathcal{E}^*$.

In graph $\mathcal{G}^*$, the examples are connected with each other by even-length walks through the output nodes whereas the same is true for the output nodes through the samples. In this case, the square of the adjacency matrix $\boldsymbol{A}^*$ represents the collection of two-walks (walks with two edges) between the vertices. It also implements two disjoint graphs $\mathcal{G}_\mathcal{M} = (\mathcal{M}, \mathcal{E}_\mathcal{M})$ and $\mathcal{G}_\mathcal{N} = (\mathcal{N}, \mathcal{E}_\mathcal{N})$ such that

$$\boldsymbol{A}^{*2} = \begin{pmatrix} \boldsymbol{B}\boldsymbol{B}^T_{m\times m} & \mathbf{0}_{m\times n} \\ \mathbf{0}_{n\times m} & \boldsymbol{B}^T\boldsymbol{B}_{n\times n} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{N} \end{pmatrix} \tag{2.5}$$

where $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$ and $\boldsymbol{N} = \boldsymbol{B}^T\boldsymbol{B}$ are the adjacency matrices specifying edges $\mathcal{E}_\mathcal{M}$ and $\mathcal{E}_\mathcal{N}$, respectively. Unlike a simple graph $\mathcal{G}$ considered by conventional graph-based methods, $\mathcal{G}_\mathcal{M}$ also involves self-loops. However, they have no effect on the graph Laplacian and thus on the embeddings. Hence, one can estimate the adjacency of examples using the predictions of the network, i.e. $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$, and then find the embeddings by applying a standard unsupervised objective such as Laplacian Eigenmap minimizing $\mathrm{Tr}(\boldsymbol{Z}^T\boldsymbol{L}_\boldsymbol{M}\boldsymbol{Z})$ as defined in (2.2), where $\boldsymbol{L}_\boldsymbol{M} = \boldsymbol{D}_\boldsymbol{M} - \boldsymbol{M}$. It is important to note that conventional graph-based algorithms assume a fixed adjacency matrix during loss minimization whereas in the proposed framework, we consider an adaptive adjacency which is updated throughout the unsupervised training process as described in the following section.

2.2.2 Adaptive Adjacency

As derived adjacency $\boldsymbol{M}$ depends only on $\boldsymbol{B}$, during the unsupervised training, $\boldsymbol{B}$ needs to be well-constrained to preserve the learned latent embeddings. Otherwise, the idea of updating the adjacency matrix throughout an unsupervised task might be catastrophic and results in offsetting the effects of the supervised training step.

There are two constraints derived from the natural expectation of the specific form of the $\boldsymbol{B}$ matrix for a classification problem: i) first, a sample is to be assigned to one class with the probability of 1, while remaining $n-1$ classes have 0 association probability, and ii) second, for balanced classification tasks, each class is expected to involve approximately the same number of the examples. Let us first consider the case where $\boldsymbol{B}$ assigns m/n examples to each one of the n classes with the probability of 1. $\boldsymbol{B}$ in this particular form implies that graph $\mathcal{G}^*$ becomes $(1, m/n)$-biregular since

$$\deg(\mathsf{m}_i) = 1, \forall \mathsf{m}_i \in \mathcal{M} \quad \text{and} \quad \deg(\mathsf{n}_i) = m/n, \forall \mathsf{n}_i \in \mathcal{N} \tag{2.6}$$

Subsequently graph $\mathcal{G}_{\mathcal{N}}$ turns into a disconnected graph including only self-loops and its adjacency matrix $\boldsymbol{N}$ becomes a scaled identity matrix indicating mutually exclusive and uniform distribution of the examples across the output nodes. Similarly, $\mathcal{G}_{\mathcal{M}}$ also becomes a disconnected graph including n disjoint subgraphs. Each one of these subgraphs is m/n-regular, where each vertex also has an additional self-loop. In this particular form, $\boldsymbol{B}$ becomes the optimal embedding of $\boldsymbol{M}$ as it satisfies both $\boldsymbol{B}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{0}$ and $\boldsymbol{B}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{I}$ where $\boldsymbol{L}_{\boldsymbol{M}}$ is the Laplacian and $\boldsymbol{D}_{\boldsymbol{M}}$ is the diagonal degree matrix of $\boldsymbol{M}$ (as shown in the Section 2.2.3). As they all depend only on $\boldsymbol{B}$, the relationships between $\mathcal{G}^*$, $\mathcal{G}_{\mathcal{M}}$ and $\mathcal{G}_{\mathcal{N}}$ are biconditional, which can be written as follows:

$$\mathcal{G}^* \text{ is } (1, m/n)\text{-biregular} \Leftrightarrow \mathcal{G}_{\mathcal{M}} : \underset{\boldsymbol{Z}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{Z}=\boldsymbol{I}}{\operatorname{argmin}} \operatorname{Tr}(\boldsymbol{Z}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{Z}) = \boldsymbol{B} \Leftrightarrow \mathcal{G}_{\mathcal{N}} : \boldsymbol{N} = m/n\boldsymbol{I} \tag{2.7}$$

Depending on this relation, we propose applying regularization during the unsupervised task in order to ensure that $\boldsymbol{N}$ becomes the identity matrix. Regularizing $\boldsymbol{N}$ instead of $\boldsymbol{M}$ enables us to devise a scalable framework as typically $n \ll m$. The first one of the proposed two regularization terms constrains $\boldsymbol{N}$ to be a diagonal matrix, whereas the second one forces it into becoming a scaled identity matrix by equalizing the value of the diagonal entries. The second term ultimately corresponds to constraining $\boldsymbol{B}$ to obtain a uniform distribution of samples across the output classes. Obviously, this condition is not valid for every dataset, but a balancing constraint is required to prevent collapsing onto a subspace of dimension less than n and it is analogous to the constraint of $\boldsymbol{Z}^T\boldsymbol{D}\boldsymbol{Z} = \boldsymbol{I}$ in [6].

2.2.3 The Optimality of $\boldsymbol{B}$

Following [6], the optimal embedding that can be found on $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$ can be written as

$$\underset{\boldsymbol{Z}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{Z}=\boldsymbol{I}}{\operatorname{argmin}} \operatorname{Tr}(\boldsymbol{Z}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{Z}) \tag{2.8}$$

where $\boldsymbol{L}_{\boldsymbol{M}}$ is the Laplacian and $\boldsymbol{D}_{\boldsymbol{M}}$ is the diagonal degree matrix of $\boldsymbol{M}$ such that $\boldsymbol{L}_{\boldsymbol{M}} = \boldsymbol{D}_{\boldsymbol{M}} - \boldsymbol{M}$. For any embedding $\boldsymbol{Z}$, one can simply write that

$$\boldsymbol{Z}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{Z} = \boldsymbol{Z}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{Z} - \boldsymbol{Z}^T\boldsymbol{M}\boldsymbol{Z} \tag{2.9}$$

Replacing $\boldsymbol{M}$ with $\boldsymbol{B}\boldsymbol{B}^T$, then (2.9) becomes

$$\boldsymbol{Z}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{Z} = \boldsymbol{Z}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{Z} - \boldsymbol{Z}^T\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{Z} \tag{2.10}$$

If the regularization turns $\boldsymbol{N} = \boldsymbol{B}^T\boldsymbol{B}$ into the identity matrix, using $\boldsymbol{B}$ as the embedding yields that

$$\boldsymbol{B}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{B}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{B} - \boldsymbol{B}^T\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{B} \tag{2.11}$$

$$\boldsymbol{B}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{B}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{B} - \boldsymbol{I} \tag{2.12}$$

and this equality is satisfied when $\boldsymbol{B}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{0}$ and $\boldsymbol{B}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{B} = \boldsymbol{I}$. Then,

$$\underset{\boldsymbol{Z}^T\boldsymbol{D}_{\boldsymbol{M}}\boldsymbol{Z}=\boldsymbol{I}}{\operatorname{argmin}} \operatorname{Tr}(\boldsymbol{Z}^T\boldsymbol{L}_{\boldsymbol{M}}\boldsymbol{Z}) = \boldsymbol{B} \tag{2.13}$$

Assuming that the proposed regularization successfully turns $\boldsymbol{N}$ into the identity matrix, this simply tells us that no additional step is necessary to find the optimal embedding of $\boldsymbol{M}$ and $\boldsymbol{B}$ automatically becomes an optimal embedding.

2.2.4 Activity Regularization

Consider a neural network with $L-1$ hidden layers where l denotes the individual index for each hidden layer such that $l \in \{1, ..., L-1\}$. Let $\boldsymbol{Y}^{(l)}$ denote the output of the nodes at layer l. $\boldsymbol{Y}^{(0)} = \boldsymbol{X}$ is the input and $f(\boldsymbol{X}) = f^{(L)}(\boldsymbol{X}) = \boldsymbol{Y}^{(L)} = \boldsymbol{Y}$ is the output of the entire network. $\boldsymbol{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weights and biases of layer l, respectively. Then, the feedforward operation of the neural networks can be written as

$$\boldsymbol{Y}^{(l)} = f^{(l)}\big(\boldsymbol{X}\big) = h^{(l)}\big(\boldsymbol{Y}^{(l-1)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}\big) \tag{2.14}$$

where $h^{(l)}(.)$ is the activation function applied at layer l.

In the proposed framework, instead of using the output probabilities of the softmax nodes, we use the activations at their inputs to calculate the regularization. The intuition here is that regularizing linear activations rather than nonlinear probabilities defines an easier optimization task. Since the multiplication of two negative activations yields a positive (false) adjacency in $\boldsymbol{M}$, we rectify the activations first, i.e. $f(x) = \max(0, x)$. Then, $\boldsymbol{B}$ becomes

$$\boldsymbol{B} = g\big(\boldsymbol{X}\big) = \max\left(\boldsymbol{0}, \big(\boldsymbol{Y}^{(L-1)}\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)}\big)\right) \tag{2.15}$$

Recall that $\boldsymbol{N}$ is an $n \times n$ symmetric matrix such that $\boldsymbol{N} := \boldsymbol{B}^T\boldsymbol{B}$ and let $\boldsymbol{v}$ be a $1 \times n$ vector representing the diagonal entries of $\boldsymbol{N}$ such that $\boldsymbol{v} := \begin{bmatrix} N_{11} & N_{22} & \dots & N_{nn} \end{bmatrix}$. Then, let us define $\boldsymbol{V}$ as an $n \times n$ symmetric matrix such that $\boldsymbol{V} := \boldsymbol{v}^T\boldsymbol{v}$. Then, the two proposed regularization terms can be written as

$$\textit{Affinity} = \alpha\big(\boldsymbol{B}\big) := \frac{\sum\limits_{i\neq j}^{n} N_{ij}}{(n-1)\sum\limits_{i=j}^{n} N_{ij}} \tag{2.16}$$

and

$$\textit{Balance} = \beta\big(\boldsymbol{B}\big) := \frac{\sum\limits_{i\neq j}^{n} V_{ij}}{(n-1)\sum\limits_{i=j}^{n} V_{ij}} \tag{2.17}$$

While *affinity* penalizes the non-zero off-diagonal entries of $\boldsymbol{N}$, *balance* attempts to equalize diagonal entries. One might suggest minimizing the off-diagonal entries of $\boldsymbol{N}$ directly without normalizing, however, normalization is required to bring both regularizers within the same range for optimization and ultimately to ease hyperparameter adjustment. Unlike regularizing $\boldsymbol{N}$ to simply become a diagonal matrix, equalizing the diagonal entries is not an objective that we can reach directly by minimizing some entries of $\boldsymbol{N}$. Hence, we propose to use (2.17) that takes values between 0 and 1 where 1 represents the case where all diagonal entries of $\boldsymbol{N}$ are equal. Respectively, we propose to minimize the normalized term (2.16) instead of the direct summation of the off-diagonal entries. However, during the optimization, denominators of these terms increase with the activations which may significantly diminish the effects of both regularizers. To prevent this phenomenon, we apply the L^2 norm to penalize the increase of the sum of squared activities. Recall that the Frobenius norm for $\boldsymbol{B}$, $||\boldsymbol{B}||_F$, is analogous to the L^2 norm of a vector and defined as

$$||\boldsymbol{B}||_F = \sqrt{\sum B_{ij}^2} \tag{2.18}$$

Hence, the proposed overall unsupervised regularization loss ultimately becomes

$$\mathcal{U}\big(g(\boldsymbol{X})\big) = \mathcal{U}(\boldsymbol{B}) = c_\alpha \alpha(\boldsymbol{B}) + c_\beta\big(1 - \beta(\boldsymbol{B})\big) + c_F||\boldsymbol{B}||_F^2 \tag{2.19}$$

and can be written in terms of $\boldsymbol{B}$ as

$$\mathcal{U}(\boldsymbol{B}) = c_\alpha \left(\frac{2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{m} B_{ki} B_{kj}}{(n-1) \sum_{i=1}^{n} \sum_{k=1}^{m} B_{ki}^2} \right) + c_\beta \left(\frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\sum_{k=1}^{m} B_{ki}^2 - \sum_{k=1}^{m} B_{kj}^2 \right)^2}{(n-1) \sum_{i=1}^{n} \sum_{k=1}^{m} B_{ki}^4} \right) + c_F \left(\sum_{i=1}^{n} \sum_{k=1}^{m} B_{ki}^2 \right) \tag{2.20}$$

2.2.5 Training

Training of the proposed framework consists of two sequential steps: Supervised training using only the labeled examples and subsequent unsupervised regularization using the entire dataset. We adopt stochastic gradient descent in the mini-batch mode [12] for optimization of both steps. Indeed, mini-batch mode is required for the unsupervised task since the proposed regularizers implicitly depend on the comparison of the examples with each other. Algorithm 1 below describes the entire training procedure. The first stage of the training is a typical supervised training task in which m_L examples $\boldsymbol{X}_L = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{m_L}]^T$ are introduced to the network with the corresponding ground-truth labels $\boldsymbol{t}_L = [t_1, ..., t_{m_L}]^T$ and the network parameters are updated to minimize the log loss $\mathcal{L}(.)$. After the supervised training step is completed, this supervised objective is never revisited and the labels $\boldsymbol{t}_L$ are never reintroduced to the network in any part of the unsupervised task. Hence, the remaining unsupervised objective is driven only by the proposed regularization loss $\mathcal{U}(.)$ defined in (2.19). The examples $\boldsymbol{X}_L$ used in the supervised training stage can also be batched together with the upcoming unlabeled examples $\boldsymbol{X}_U = [\boldsymbol{x}_{m_L+1}, ..., \boldsymbol{x}_m]^T$ in order to ensure a more stable regularization. As the network is already introduced to them, b_L examples randomly chosen from $\boldsymbol{X}_L$ can be used as guidance samples for the remaining b_U examples of $\boldsymbol{X}_U$ in

that batch. Such blended batches help the unsupervised task especially when the examples in the dataset have high variance.

Algorithm 1: Model training

Supervised training:

Input : $\boldsymbol{X}_L = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{m_L}]^T$, $\boldsymbol{t}_L = [t_1, ..., t_{m_L}]^T$, batch size b

repeat

$\{(\acute{\boldsymbol{X}}_1, \acute{\boldsymbol{t}}_1), ..., (\acute{\boldsymbol{X}}_{m_L/b}, \acute{\boldsymbol{t}}_{m_L/b})\} \longleftarrow (\boldsymbol{X}_L, \boldsymbol{t}_L)$ `// Shuffle and create batch pairs, e.g. if` $m_L = 1000$ `and` $b = 100$`, then 10 pairs vs. 100 for batch size`

for $i \leftarrow 1$ ***to*** m_L/b **do**

Take i^{th} pair $(\acute{\boldsymbol{X}}_i, \acute{\boldsymbol{t}}_i)$

Take a gradient step for $\mathcal{L}\big(f\big(\acute{\boldsymbol{X}}_i\big), \acute{\boldsymbol{t}}_i\big)$

until stopping criteria is met

return *model*

Unsupervised training:

Input : *model*, $\boldsymbol{X}_L = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{m_L}]^T$, $\boldsymbol{X}_U = [\boldsymbol{x}_{m_L+1}, ..., \boldsymbol{x}_m]^T$, b_L, b_U

repeat

$\big\{\acute{\boldsymbol{X}}_1, ..., \acute{\boldsymbol{X}}_{m_U/b_U}, \big\} \longleftarrow \boldsymbol{X}_U$ `// Shuffle and create input batches`

for $i \leftarrow 1$ ***to*** m_U/b_U **do**

Take i^{th} input batch $\acute{\boldsymbol{X}}_i$

$\ddot{\boldsymbol{X}} \longleftarrow$ `random`$(\boldsymbol{X}_L, b_L)$ `// Randomly sample` b_L `examples from` $\boldsymbol{X}_L$

Take a gradient step for $\mathcal{U}\big(g\big(\begin{bmatrix}\acute{\boldsymbol{X}}_i^T & \ddot{\boldsymbol{X}}^T\end{bmatrix}^T\big)\big)$

until stopping criteria is met

2.3 Experiments

The models have been implemented in Python using Keras [14] and Theano [60]. Open source code is available at http://github.com/ozcell/LALNets that can be used to reproduce the experimental results obtained on the three image datasets, MNIST [35], SVHN [44] and NORB [36] have been used by previous researchers [18, 28, 33, 37, 41, 42, 52, 56, 65, 67] publishing in the field of semi-supervised learning at NIPS and other similar venues.

2.3.1 Datasets Used in the Experiments

Figures 2.1, 2.2 and 2.3 respectively illustrate 200 examples from MNIST, SVHN and NORB datasets and Table 2.1 summarizes the properties of these datasets used in the experiments. The MNIST is a dataset of handwritten digits in which digits have been size-normalized and centered in a fixed-size image [35]. SVHN is a real-world color image dataset that can be seen as similar in flavor to MNIST, but comes from a significantly harder, real world problem, i.e. recognizing digits and numbers in natural scene images [44]. NORB, on the other hand, is a dataset of grayscale images of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations and 18 azimuths. Its training set is composed of 5 instances of each category and test set of the remaining 5 instances [36]. The following preprocessing steps have been applied to these datasets.

- MNIST: Images were normalized by dividing by 256.
- SVHN: We applied centering and normalization per each channel, i.e. we set each sample mean to 0 and divided each input by its standard deviation.
- NORB: Following [37], images were downsampled to 32×32 using nearest-neighbor interpolation. We added uniform noise between 0 and 1 to each pixel value. First, we normalized the NORB dataset by dividing by 256, then applied centering and normalization per each channel and also set input mean to 0 over the dataset and divided inputs by standard deviation of the dataset.

2.3.2 Models Used in the Experiments

Table 2.2 summarizes all models used in the experiments. Reported MNIST results have been obtained using the model named *6-layer CNN*, whereas SVHN results were obtained

Figure 2.1: MNIST Dataset.

Table 2.1: Datasets used in the experiments.

	Data type	Number of examples	Dimension	Number of classes	% of largest class
MNIST	Image: Hand-written digits	Train: 60000, Test: 10000	$1 \times 28 \times 28$	10	11%
SVHN	Image: Street-view digits	Train: 73257, Test: 26032, Extra: 531131	$3 \times 32 \times 32$	10	19%
NORB	Image: Objects	Train: 24300, Test: 24300	$2 \times 96 \times 96$	5	20%

using the *9-layer CNN-2* model and NORB results were obtained using the *9-layer CNN* model. The results obtained on different models are also presented in the following sections to show the effect of the chosen model on the test accuracy. The ReLU activation function [43], i.e. $f(x) = \max(0, x)$, has been used for all models. For both supervised training and unsupervised regularization, models were trained using stochastic gradient descent with following settings: learning rate= 0.01, decay= $1e^{-6}$, momentum= 0.95 with Nesterov updates, where the momentum direction is first applied and then this direction is corrected with a gradient update [8].

Figure 2.2: SVHN Dataset.

Table 2.2: Specifications of the models used in the experiments†.

Model name	Specification
4-layer MLP	FC(2048) - Drop(0.5) - FC(2048) - Drop(0.5) - FC(2048) - Drop(0.5) - FC(n)
6-layer CNN	2*Conv(32x3x3) - MP(2x2) - Drop(0.2) - 2*Conv(64x3x3) - MP(2x2) - Drop(0.3) - FC(2048) - Drop(0.5) - FC(n)
9-layer CNN	2*Conv(32x3x3) - MP(2x2) - Drop(0.2) - 2*Conv(64x3x3) - MP(2x2) - Drop(0.3) - 3*Conv(128x3x3) - MP(2x2) - Drop(0.4) - FC(2048) - Drop(0.5) - FC(n)
9-layer CNN-2	2*Conv(64x3x3) - MP(2x2) - Drop(0.2) - 2*Conv(128x3x3) - MP(2x2) - Drop(0.3) - 3*Conv(256x3x3) - MP(2x2) - Drop(0.4) - FC(2048) - Drop(0.5) - FC(n)

† Inputs of the models are determined according to the dimensions of the dataset being used for the training. n is the number of classes.
FC(i): Fully connected layer with i units
Drop(i): Applying dropout[58] where the probability of retaining a unit is $1 - i$
Conv($i \times j \times k$): Convolution layer where i corresponds to the number of filters and $j \times k$ to the kernel size
MP($i \times j$): Max pooling with pool size of $i \times j$, i.e. factors by which to downscale (vertical $\times$ horizontal)

2.3.3 Results

Coefficients of the proposed regularization term have been chosen as $c_\alpha = 3, c_\beta = 1$ and $c_F = 0.000001$ in all of the experiments. For the sake of fairness, in the same manner as followed by the models used for the comparison [42], a validation set of 1000 examples has been chosen randomly without stratification among the training set examples to determine the hyperparameters of the proposed approach ($c_\alpha, c_\beta, c_F, {}^{b_L}/_{b_U}$), the specifications of the models used for the experiments (given in Table 2.2) and the epoch to report the test accuracy. We

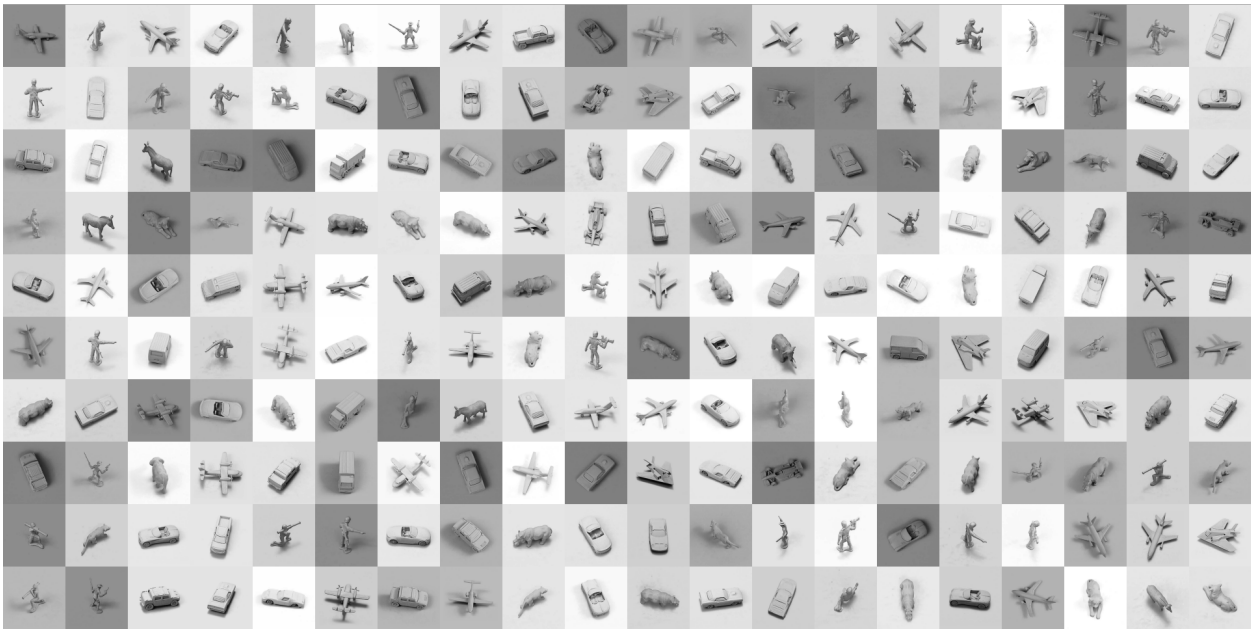

Figure 2.3: NORB Dataset.

used a batch size of 128 for both supervised training and unsupervised regularization steps. For each dataset, the same strategy is applied to decide on the ratio of labeled and unlabeled data in the unsupervised regularization batches, i.e. b_L/b_U: among a hyperparameter set of $\{16, 32, 64, 96, 112\}$, b_L is assigned as the setting closest to one tenth of the number of all labeled examples m_L, and then b_U is chosen to complement the batch size up to 128, i.e. $b_U = 128 - b_L$. Each experiment has been repeated for 10 times. For each repetition, to assign $\boldsymbol{X}_L$, m_L examples have been chosen through stratified random sampling to keep the same ratio for each class.

2.3.3.1 MNIST

On the MNIST dataset, experiments have been performed using 4 different settings for the number of labeled examples, i.e. $m_L = \{100, 600, 1000, 3000\}$ respectively corresponding to $\{10, 60, 100, 300\}$ labeled examples from each class, following the literature used for

comparative results [28, 41, 42, 52, 67]. Following the strategy described in the previous section, the unsupervised regularization batches have been formed by choosing $b_L = 16$ and $b_U = 112$ for the $m_L = 100$ case. However, we didn't follow this strategy for $m_L = \{600, 1000, 3000\}$ cases, but kept the same b_L/b_U ratio, i.e. $16/112$, to reduce the runtime of the experiments as we haven't observed any statistically significant differences when further increasing b_L ratio for these 3 cases on MNIST.

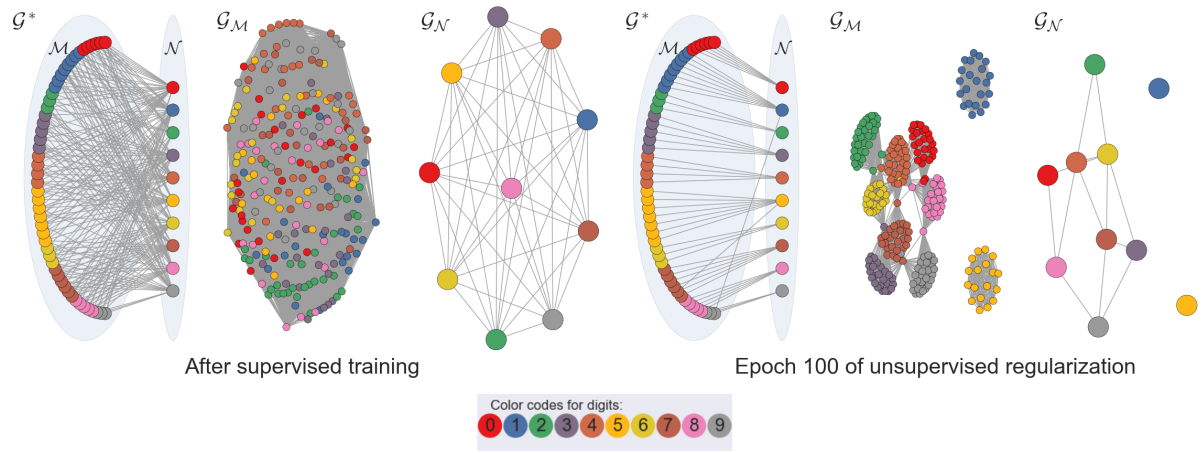

Figure 2.4: Visualizations of the graphs $\mathcal{G}^*$, $\mathcal{G}_\mathcal{M}$ and $\mathcal{G}_\mathcal{N}$ for a randomly chosen 250 test examples from MNIST for the $m_L = 100$ case. As implied through the relation defined in (2.7) and discussed in detail in Section 2.2.3, when we regularize $\boldsymbol{N}$ to become the identity matrix: i) $\mathcal{G}^*$ becomes a biregular graph, ii) $\mathcal{G}_\mathcal{M}$ turns into a disconnected graph of n m/n-regular subgraphs, iii) $\mathcal{G}_\mathcal{N}$ turns into a disconnected graph of n nodes having self-loops only (self-loops are not displayed for the sake of clarity), and automatically iv) $\boldsymbol{B}$ becomes the optimal embedding of $\boldsymbol{M}$. Color codes denote the ground-truths for the examples. This figure is best viewed in color.

Figure 2.4 shows a visualization of the realization of the graph-based approach described in this chapter using real predictions for the MNIST dataset. After the supervised training, in the bipartite graph $\mathcal{G}^*$ (defined by $\boldsymbol{B}$), most of the examples are connected to multiple

output nodes at the same time. In fact, the graph between the examples $\mathcal{G}_{\mathcal{M}}$ (inferred by $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^T$) looks like a sea of edges. However, thanks to supervised training step, some of these edges are actually quite close to the numerical probability value of 1. Through a scalable regularization objective, which is defined on the graph between the output nodes $\mathcal{G}_{\mathcal{N}}$ (inferred by $\boldsymbol{N} = \boldsymbol{B}^T\boldsymbol{B}$), stronger edges are implicitly propagated in graph $\mathcal{G}_{\mathcal{M}}$. In other words, as hypothesized, when $\boldsymbol{N}$ turns into the identity matrix, $\mathcal{G}^*$ closes to be a biregular graph and $\mathcal{G}_{\mathcal{M}}$ closes to be a disconnected graph of n $^m/_n$-regular subgraphs. As implied through the relation defined in (2.7) and discussed in detail in Section 2.2.3, we expect $\boldsymbol{B}$ to automatically become the optimal embedding of $\boldsymbol{M}$. Figure 2.5 presents the t-SNE [38] visualizations of the embedding spaces inferred by $\boldsymbol{B}$. t-SNE is a popular method creating two-dimensional maps from data with thousands of dimensions. Recently, it has become widespread in the field of machine learning to explore high-dimensional data [66]. For semi-supervised and unsupervised settings, well-separated clusters on these two-dimensional maps indicate the quality of the obtained latent representation. However, the distances between well-separated clusters in a t-SNE plot may have no further meaning [66]. As clearly observed from Figure 2.5, as the unsupervised regularization exploits the unlabeled data, clusters become well-separated and simultaneously the test accuracy increases.

Table 2.3 summarizes the semi-supervised test accuracies observed with four different m_L settings. Results of a broad range of recent existing solutions are also presented for comparison. These solutions are grouped according to their approaches to semi-supervised learning. While [28], [52], [37] employ autoencoder variants, [42], [41] and [56] adopt adversarial training, [67] and [65] are another graph-based methods. To show the baseline of the unsupervised regularization step in our framework, the performance of the network after the supervised training is also given. For MNIST, GAR outperforms existing graph-based methods and all the contemporary methods other than some cutting-edge approaches that use generative models.

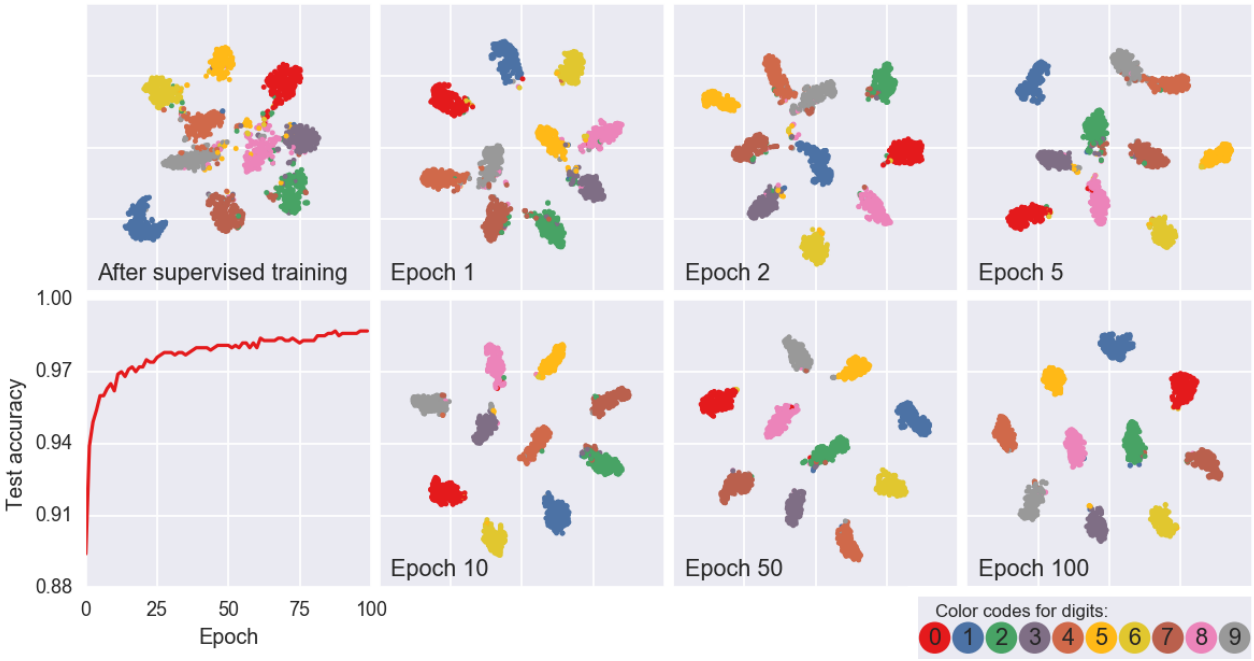

Figure 2.5: t-SNE visualization of the embedding spaces inferred by $\boldsymbol{B}$ for a randomly chosen 2000 test examples for the $m_L = 100$ case. Color codes denote the ground-truths for the examples. Note the separation of clusters from epoch 0 (right after supervised training step) to epoch 100 of the unsupervised training. For reference, accuracy for the entire test set of 10000 examples is also plotted with respect to the unsupervised training epochs. This figure is best viewed in color.

2.3.3.2 SVHN and NORB

SVHN and NORB datasets are both used frequently in recent literature on semi-supervised classification benchmarks [18, 28, 33, 37, 41, 42, 56]. Either dataset represents a significant jump in difficulty for classification when compared to the MNIST dataset. Table 2.4 summarizes the semi-supervised test accuracies observed on SVHN and NORB. For SVHN experiments, 1000 labeled examples have been randomly chosen with stratification among 73,257 training examples. Two experiments are conducted where the SVHN *extra* set (an additional training set including 531,131 more samples) is either omitted from the unsupervised training or not. The same batch ratio has been used in both experiments as $b_L = 96, b_U = 32$.

Table 2.3: Benchmark results of the semi-supervised test accuracies on MNIST for few labeled samples, $m_L \in \{100, 600, 1000, 3000\}$. Results of a broad range of recent existing solutions are also presented for comparison. The last row demonstrates the benchmark scores of the proposed framework in this chapter.

	$m_L = 100$	$m_L = 600$	$m_L = 1000$	$m_L = 3000$
M1+M2[28]	3.33%(±0.14)	2.59%(±0.05)	2.40%(±0.02)	2.18%(±0.04)
Ladder Network[52]	1.06%(±0.37)	-	0.84%(±0.08)	-
AGDM[37]	0.96%(±0.02)	-	-	-
VAT[42]	2.12%	1.39%	1.36%	1.25%
Extended VAT[41]	1.36%	-	1.27%	-
Improved GAN[56]	0.93%(±0.07)	-	-	-
EmbedCNN[67]	7.75%	3.82%	2.73%	2.07%
HAGR[65]	11.34%(±1.23)	-	-	-
Supervised training (Baseline)	14.82%(±1.03)	4.16%(±0.26)	3.25%(±0.11)	1.73%(±0.08)
Supervised training + GAR	1.56%(±0.09)	1.15%(±0.07)	1.10%(±0.07)	0.93%(±0.05)

On the other hand, for NORB experiments, to follow the previously published works reporting their results on the NORB dataset [28, 37] using 1000 labeled examples and to further create a more challenging scenario using less number of labeled examples, 2 different settings have been used for the number of labeled examples, i.e. $m_L = \{300, 1000\}$ with the same batch ratio of $b_L = 32, b_U = 96$, where labeled examples have been randomly chosen with stratification. Both results are included for comparative purposes.

Deep generative approaches [18, 28, 33, 37, 41, 42, 52, 56] have gained popularity especially over the last two years for semi-supervised learning and achieved superior performance for the problems in this setting in spite of the difficulties in their training. As it requires no additional work in calculating the reconstruction error (as in autoencoder based methods [28, 37, 52]) or implementing a zero-sum game mechanism (as in GAN based methods [41, 42, 56]), GAR is natural for the classification framework on neural networks and it is therefore a low-cost and efficient competitor for generative approaches and achieves

comparable performance with these state-of-the-art models while statistically significantly outperforming all existing graph-based methods. Most importantly, GAR is open to further improvements with standard data enhancement techniques such as augmentation, ensemble learning.

Table 2.4: Benchmark results of the semi-supervised test accuracies on SVHN and NORB. Results of a broad range of most recent existing solutions are also presented for comparison. The last row demonstrates the benchmarks for the proposed framework in this chapter.

	SVHN		NORB	
	$m_L = 1000^{\dagger}$	$m_L = 1000$	$m_L = 300$	$m_L = 1000$
M1+TSVM[28]	55.33%(±0.11)	-	-	18.79%(±0.05)
M1+M2[28]	36.02%(±0.10)	-	-	-
SGDM[37]	29.82%	16.61%(±0.24)	-	9.40%(±0.04)
VAT[42]	24.63%	-	-	9.88%
Extended VAT[41]	5.77%	-	-	-
Extended VAT+EntMin[41]	4.28%	-	-	-
Improved GAN[56]	8.11%(±1.30)	-	-	-
ALI[18]	-	7.42%(±0.65)	-	-
Π Model [33]	5.43%(±0.25)	-	-	-
Supervised training (Baseline)	19.11%(±1.09)	19.11%(±1.09)	17.93%(±1.07)	10.22%(±1.00)
Supervised training + GAR	8.67%(±0.65)	6.98%(±0.82)	12.19%(±1.46)	7.10%(±0.57)

† This column presents the results obtained when SVHN *extra* set is omitted from the unsupervised training. Unless otherwise specified, reported results for other approaches are assumed to represent this scenario.

2.3.3.3 Effects of Hyperparameters

Figure 2.6 presents the test accuracy curves with respect to the unsupervised training epochs obtained using different models. The proposed unsupervised regularization improves the test accuracy in all models. However, the best case depends on the chosen model specifications.

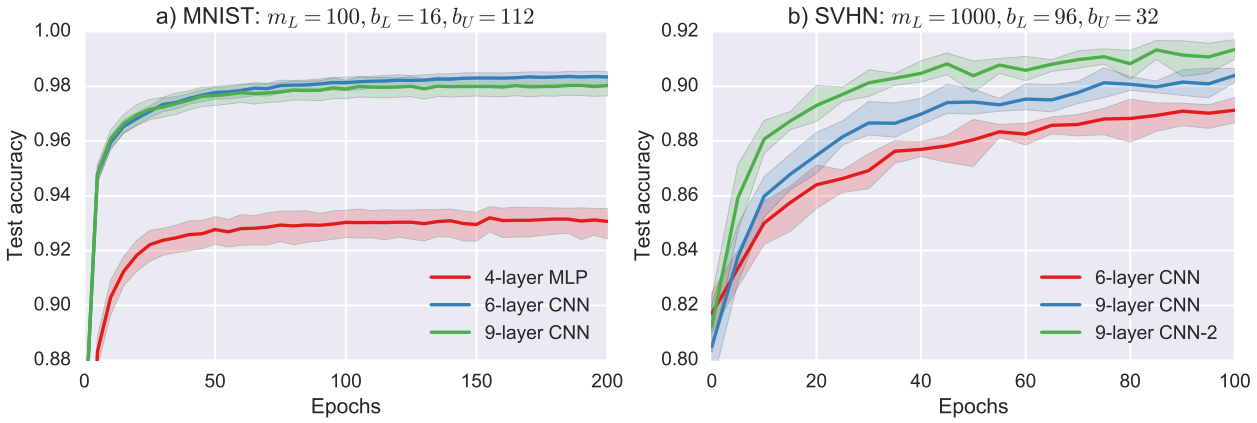

Figure 2.6: Test accuracies obtained using different models with respect to unsupervised training epochs. a) On the left, MNIST results are given for the settings that $m_L = 100$ and $b_L = 16, b_U = 112$ and b) On the right, SVHN results are given for the settings that $m_L = 1000$ and $b_L = 96, b_U = 32$. Model specifications are given in Table 2.2. Shaded regions show the 0.95 confidence interval.

The labeled/unlabeled data ratio of unsupervised training batches is the most critical hyperparameter of the proposed regularization. Figure 2.7 visualizes the effect of this ratio for MNIST and SVHN datasets. These two datasets have different characteristics. MNIST dataset has a lower variance among its samples with respect to SVHN. As a result, even when the labeled examples introduced to the network during the supervised training are not blended with the unsupervised training batches, i.e. $b_L = 0$, this does not affect the performance dramatically. However, for the SVHN dataset, reducing the b_L proportion of the unsupervised training batches significantly affects the accuracy and further decrease of b_L reduces the stability of the regularization.

One can also observe another phenomenon through the MNIST results in Figure 2.7. That is, as b_L approaches m_L, the generalization of the model reduces. This effect can be better observed in Figure 2.8 including a further step, i.e. $b_L = 96$ when $m_L = 100$. Since the same examples start to dominate the batches of unsupervised regularization, overfitting occurs and ultimately test accuracy significantly reduces. Figure 2.8 also presents the effect

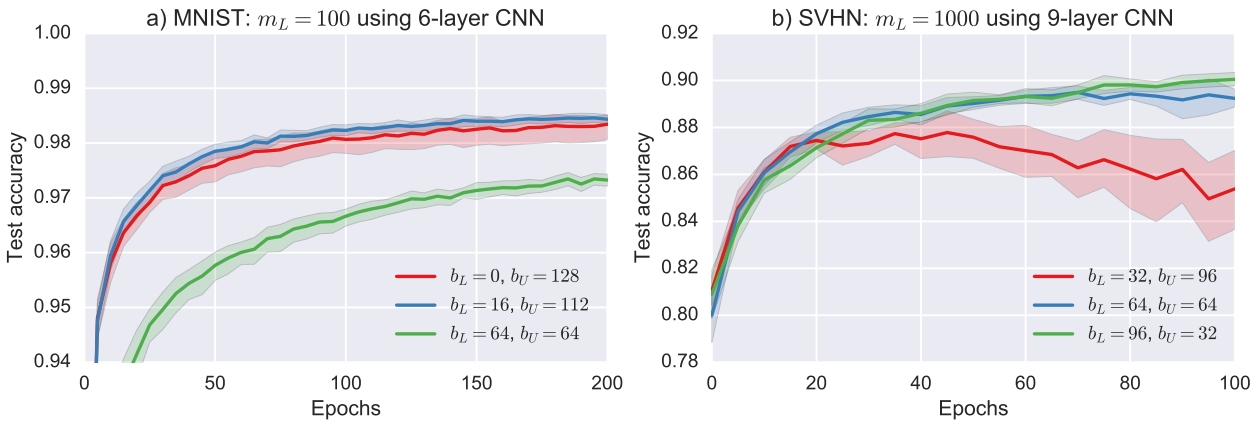

Figure 2.7: The effect of ${}^{b_L}/_{b_U}$ ratio for MNIST and SVHN datasets. Shaded regions show the 0.95 confidence interval.

of applying dropout during the unsupervised regularization. Dropping out the weights the during unsupervised training dramatically affects the accuracy. This effect is more obvious when b_L is smaller. Hence, for the experiments, we removed the dropouts from the models specified in Table 2.2 during the unsupervised training.

The effects of regularization coefficients are presented in Figure 2.9 for the MNIST dataset. Part (a) of the figure visualizes the case when c_F is held constant, but the ratio of ${}^{c_\alpha}/_{c_\beta}$ changes. And part (b) of the figure illustrates the case when ${}^{c_\alpha}/_{c_\beta}$ is held constant, but c_F changes. We can say that as long as $c_\alpha \geq c_\beta$, the ratio of ${}^{c_\alpha}/_{c_\beta}$ does not affect the accuracy significantly. Furthermore, the value of c_F is not so critical (close performances both with $c_F = 1e^{-6}$ and $c_F = 1e^{-15}$) unless it is too big to distort the regularization. Therefore, we can say that the proposed unsupervised regularization term is considerably robust with respect to the coefficients c_α, c_β and c_F. This can also be seen through the fact that we have applied the same coefficients for the experiments of all three datasets.

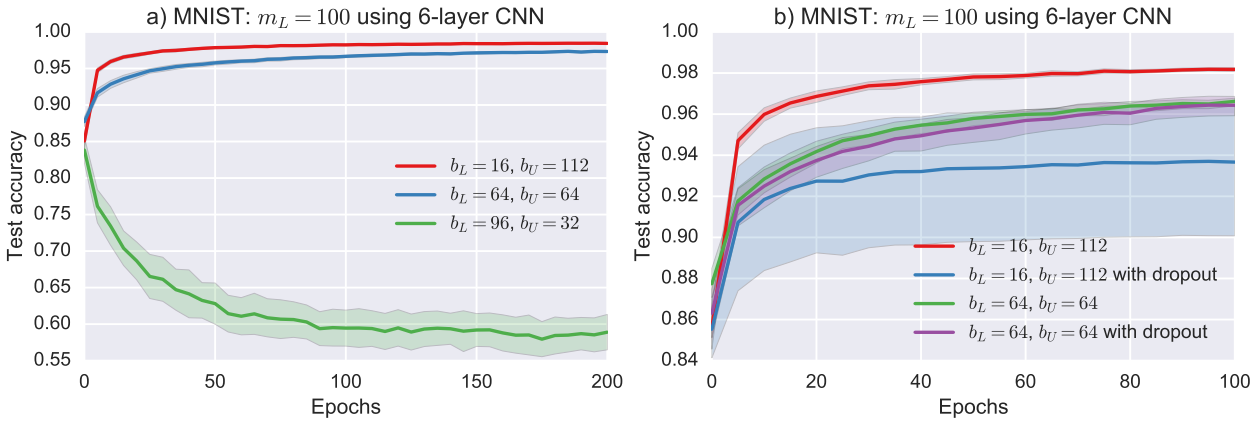

Figure 2.8: The effects of a) on the left, choosing $b_L \approx m_L$ and b) on the right, applying dropout during the unsupervised training on MNIST dataset. Note that the figures show different ranges of training epochs, i.e. 200 vs. 100. Shaded regions show the 0.95 confidence interval.

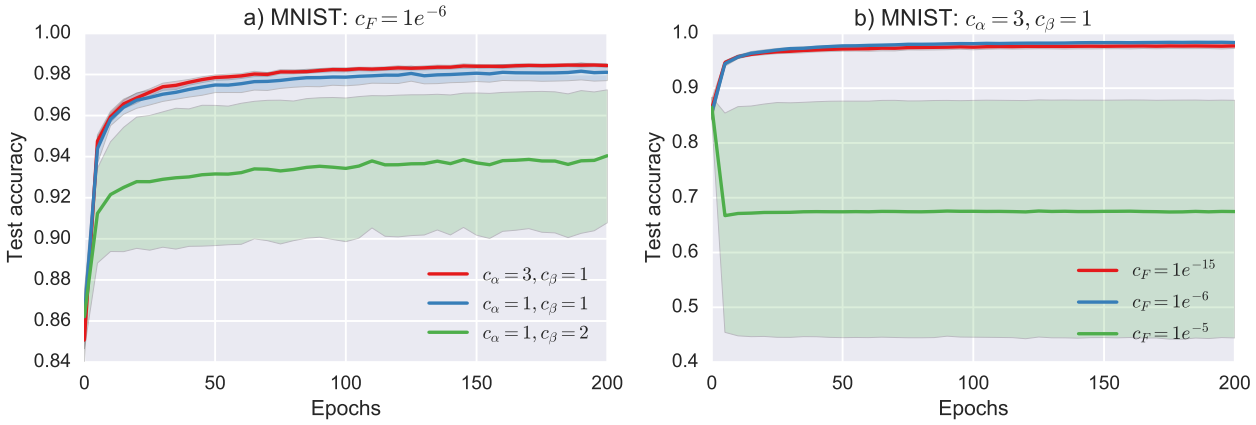

Figure 2.9: The effects of a) c_α/c_β and b) c_F on MNIST dataset. Shaded regions show the 0.95 confidence interval. The size of green shaded regions implies the instability of the model when using the specified settings as the obtained accuracy varies drastically per epoch.

CHAPTER 3: AUTO-CLUSTERING OUTPUT LAYER: AUTOMATIC LEARNING OF LATENT ANNOTATIONS IN NEURAL NETWORKS[1]

Combinations of supervised and unsupervised learning have resulted in many fruitful developments throughout the machine learning literature. Among many achievements, unsupervised feature learning where unsupervised training is used as a pretraining stage for initializing hidden layer parameters [22], was the first method to succeed in the training of fully connected deep architectures and played a key role in igniting the third wave of machine learning research by creating a paradigm shift we now call deep learning [19].

In current literature, different kinds of approaches exist to combine supervised and unsupervised learning. In this context, *semi-supervised* is frequently used to specify certain applications where a large number of observations exist with only a small subset having ground-truth labels. Solutions suggested for these applications seek ways of exploiting the unlabeled data to improve the model generalization. There have been significant developments recently in this field. Following the introduction of Bayesian inference to the conventional autoencoder architecture [29, 53], these variational autoencoders have been proven to make deep generative models highly competitive for semi-supervised learning [28, 37]. Virtual adversarial training [42] motivated by Generative Adversarial Nets [20] and the denoising autoencoder variant named Ladder networks [61] have also been successfully employed for semi-supervised learning problems [52]. On the other hand, in the previous chapter, we have proposed a scalable and efficient graph-based method that is natural for the classification framework on neural networks as it requires no additional work calculating the reconstruction

[1]This chapter has been submitted to peer-reviewed IEEE Transactions on Neural Networks and Learning Systems and is now under the second revision.

error (as in autoencoder based methods) or implementing zero-sum game mechanism (as in adversarial training based methods) and we reported competitive performance with respect to other approaches.

In this chapter, we consider a different kind of semi-supervised setting in which a coarse level of labeling (e.g. tree, flower) is available for all observations, but the model still needs to learn a fine level of latent annotations, which are previously unknown to the user (e.g. maple, rose), for each one of them. Provided labeling can be interpreted as parent-class information and latent annotations to be explored can be conceived as the sub-classes. Since partial supervision does not involve any explicit information, sub-class exploration is considered an unsupervised task. Therefore, the overall learning procedure can be considered semi-supervised. For clarification, let us assume that we are given a dataset of hand-written digits such as MNIST [35] where the overall task is complete categorization of each digit, but the only available supervision is whether a digit is smaller or greater than 5, as visualized in Figure 3.1. While being trained to categorize each example as a member of parent-classes, $\{0, 1, 2, 3, 4\}$ or $\{5, 6, 7, 8, 9\}$, the model also needs to learn to distinguish the digits, sub-classes under the same parent from each other. Since provided labeling involves no explicit information about the difference between the samples of digit 0 and the samples of digit 1, their separation is performed as an unsupervised task. As we use partial supervision to help overall categorization, the entire procedure is semi-supervised.

Outside the neural network literature, the learning of latent variable models when supervision for more general classes than those of interest is provided has previously been studied. In the natural language processing (NLP) field, following the introduction of Latent Dirichlet Allocation (LDA), a completely unsupervised algorithm that models each document as a mixture of topics [11], several modifications have been proposed to incorporate supervision [10, 32, 48, 49]. These ideas have also been extended for simultaneous image classification and annotation [51, 64].

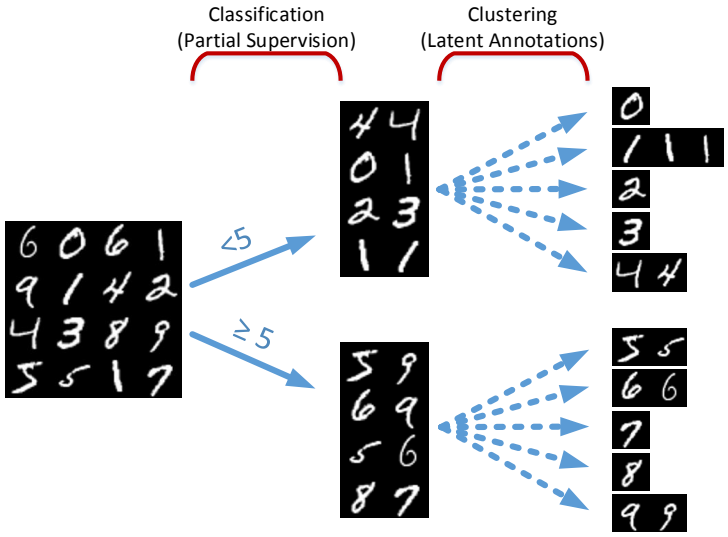

Figure 3.1: Particular semi-supervised setting discussed in this chapter. That is, a coarse level of labeling is available for all observations but the model has to learn a fine level of latent annotation for each one of them. We propose a novel approach that considers these problems as simultaneously performed supervised classification (per provided parent-classes) and unsupervised clustering (within each parent) tasks and devise a framework to enable a neural network to learn the latent annotations in this partially supervised setup.

From the viewpoint of parent/sub-class interpretation of the provided supervision and latent annotations, an analogy can be established between the semi-supervised problems discussed in this chapter and the hierarchical classification tasks in the literature. Hierarchical classification has previously been studied in neural networks [68]; however, proposed approaches consider the completely supervised case where every observation in the dataset has a parent-class label but only a few of them also have additional sub-class labels. The literature about hierarchical classification is scattered across very different application domains such as text categorization, protein function prediction, music genre classification and image classification [26].

Given partial supervision, latent annotation learning is also a common problem in these domains as well as in exploratory scientific research such as in medicine and genomics [5],

as the tasks in these domains naturally involve both already-explored (hence labeled) classes and extraction of not-yet-explored (hence hidden) patterns. In this chapter, we propose a novel approach that considers these problems as simultaneously performed supervised classification (per provided parent-classes) and unsupervised clustering (within each parent) tasks and devise a framework to enable a neural network to learn the latent annotations in this partially supervised setup. This framework involves a novel output layer modification called the auto-clustering output layer (ACOL) that allows concurrent classification and clustering tasks where clustering is performed according to the Graph-based Activity Regularization (GAR) technique proposed in the previous chapter. ACOL duplicates the softmax nodes at the output layer and GAR allows for competitive learning between these duplicates in a traditional error-correction learning framework.

This chapter is organized as follows. The next section briefly summarizes the activity regularization proposed in the previous chapter which we adopt as the objective of the unsupervised portion of the training. In the third section, we describe the proposed output layer modification and its integration with the GAR technique and experimental results are presented in the fourth section.

3.1 Related Work

GAR is a scalable and efficient graph-based approach which was originally proposed for the classical type of semi-supervised learning problems where a large number of observations with only a small subset of corresponding labels exist. In this chapter, we adopt the same regularization terms and show that these terms can be employed to reveal the latent annotations in a supervised setup through ACOL. After describing ACOL, Section 3.2.2 describes how ACOL is integrated with the GAR technique.

3.2 Auto-clustering Output Layer

3.2.1 Output Layer Modification

Neural networks define a family of functions parameterized by weights and biases which define the relation between inputs and outputs. In multi-class categorization tasks, outputs correspond to class labels, hence in a typical output layer structure there exists an individual output node for each class. An activation function, such as softmax is then used to calculate normalized exponentials to convert the previous hidden layer's activities, i.e. scores, into probabilities, such that $f(x_i) = {e^{x_i}}/{\sum e^{x_j}}$.

Unlike a traditional output layer structure, ACOL defines more than one softmax node (k_s duplicates) per parent-class. Outputs of k_s duplicated softmax nodes that belong to the same parent are then combined in a subsequent pooling layer for the final prediction. Training is performed in the configuration shown in Figure 3.2 where n_p is the number of parent-classes. This might look like a classifier with redundant softmax nodes. However, the duplicated softmax nodes of each parent are specialized using GAR throughout the training in a way that each one of $n = n_p k_s$ softmax nodes represent an individual sub-class of a parent, i.e. annotation.

ACOL does not change feedforward and backpropagation mechanisms of the network drastically. During feedforward operation of the network, the pooling layer calculates final parent-class predictions through sub-class probabilities. Pooling does not affect backpropagation in terms of derivatives and ACOL behaves in a similar fashion to a traditional output layer. However, labels are now implicitly applied to multiple softmax nodes each representing an individual sub-class under the same parent. In other words, even if the labels are provided as a one-hot encoded vector at the output, due to the pooling layer, it turns into k_s-hot encoded vector at the augmented softmax layer. k_s softmax nodes simultaneously receive the

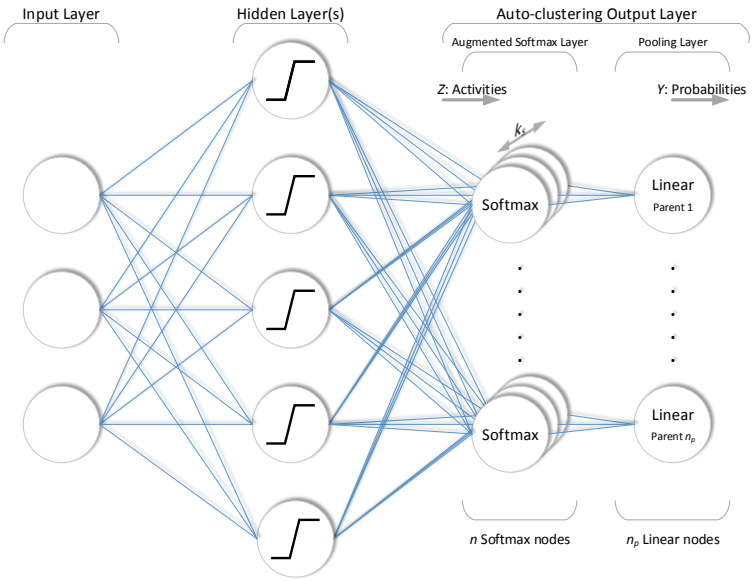

Figure 3.2: Neural network structure with the ACOL. Each softmax node corresponds to an individual sub-class of a parent, i.e. annotation. During feedforward operation of the network, the pooling layer calculates final parent-class predictions through sub-class probabilities.

error between the label and the prediction and then backpropagate it towards the previous hidden layers.

This structure carries the ability to learn latent annotations as ACOL introduces extra trainable weights between the previous hidden layer and itself. Each softmax node is connected to the previous hidden layer through non-shared weights. Due to random initialization, these weights may ultimately converge to different values at the end of training and duplicated softmax nodes may be specialized for only a subset of the samples of that parent-class. However, this mechanism is completely uncontrollable. Furthermore, during the weight updates, if any one of the duplicated softmax nodes get activated to generate a significantly lower error, through the pooling layer, this will also eliminate the backpropagated error to other $k_s - 1$ softmax nodes of that parent. Therefore, not only the one reducing the error, but all k_s duplicated softmax nodes diminish backpropagating the error to previous

layers. That is to say, without any additional mechanism, there is no actual competition between the duplicated softmax nodes of a parent.

Therefore, we adopt the GAR objective defined in (2.19) as the unsupervised regularization term to create competition between the duplicates which ultimately results in specialized but equally-active softmax nodes each representing a latent annotation within a parent. The following subsection mathematically describes ACOL and its collaboration with GAR.

3.2.2 Mathematical Description and GAR Integration

Using the same notation in Chapter 2, for traditional output layer structure, let us define the activities at the input of softmax layer as $\boldsymbol{Z}$ such that

$$\boldsymbol{Z} := \boldsymbol{Y}^{(L-1)}\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)} \tag{3.1}$$

In a neural network with ACOL, due to the subsequent pooling layer, (3.1) is modified as

$$\boldsymbol{Z} := \boldsymbol{Y}^{(L-2)}\boldsymbol{W}^{(L-1)} + \boldsymbol{b}^{(L-1)} \tag{3.2}$$

and now $\boldsymbol{Z}$ corresponds to an $m \times n$ matrix representing the activities going into the augmented softmax layer. Recall that n is the total number of all softmax nodes at the augmented softmax layer such that $n = n_p k_s$, where n_p is the number of parent-classes and k_s is the number of softmax duplicates per each parent-class. Then, the output of the ACOL applied network can be written in terms of $\boldsymbol{Z}$ as

$$f\big(\boldsymbol{X}\big) = \boldsymbol{Y} = h^{(L)}\Bigg(h^{(L-1)}\big(\boldsymbol{Z}\big)\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)}\Bigg) \tag{3.3}$$

where $\boldsymbol{Y}$ is an $m \times n_p$ matrix whose cell Y_{ij} represents the probability of i^{th} example belonging to j^{th} parent. Since $h^{(L-1)}(.)$ and $h^{(L)}(.)$ respectively correspond to softmax and linear activation functions and $\boldsymbol{b}^{(L)} := \boldsymbol{0}$ for ACOL networks, then (3.3) further simplifies into

$$\boldsymbol{Y} = \text{softmax}\big(\boldsymbol{Z}\big)\boldsymbol{W}^{(L)} \tag{3.4}$$

where $\boldsymbol{W}^{(L)}$ (hereafter will be denoted as $\boldsymbol{W}$ for simplicity) is an $n \times n_p$ matrix representing the constant weights between the augmented softmax layer and the pooling layer such that

$$\boldsymbol{W} := \boldsymbol{W}^{(L)} = \begin{bmatrix} \boldsymbol{I}_{n_p} \\ \boldsymbol{I}_{n_p} \\ \vdots \\ \boldsymbol{I}_{n_p} \end{bmatrix} \tag{3.5}$$

and simply sums up the output probabilities of the softmax nodes belonging to the same parent. Since the output of the augmented softmax layer is already normalized, no additional averaging is needed at the pooling layer and summation alone is sufficient to calculate final parent-class probabilities.

Recalling that GAR is applied to the positive part of activities going into the augmented softmax layer, i.e. $\boldsymbol{B} := \max\left(\boldsymbol{0}, \boldsymbol{Z}\right)$, the overall objective cost function of the training can be written as

$$\mathcal{L}\big(\boldsymbol{Y}, \boldsymbol{t}\big) + c_\alpha \alpha\big(\boldsymbol{B}\big) + c_\beta\big(1 - \beta\big(\boldsymbol{B}\big)\big) + c_F ||\boldsymbol{B}||_F^2 \tag{3.6}$$

where $\mathcal{L}(.)$ is the supervised log loss function and $\boldsymbol{t} = [t_1, ..., t_m]^T$ is the vector of provided parent-class labels such that $t_i \in \{1, ..., n_p\}$. Also, recall that $\alpha(.)$ and $\beta(.)$ are the unsupervised regularization terms respectively defined in (2.16)-(2.17) and $||\boldsymbol{B}||_F$ corresponds to the Frobenius norm for $\boldsymbol{B}$.

Algorithm 2: Model training

Input : $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_m]^T$,
$\boldsymbol{t} = [t_1, ..., t_m]^T$ such that $t_i \in \{1, ..., n_p\}$,
batch size b, weighing coefficients c_α, c_β, c_F

repeat

$\{(\acute{\boldsymbol{X}}_1, \acute{\boldsymbol{t}}_1), ..., (\acute{\boldsymbol{X}}_{m/b}, \acute{\boldsymbol{t}}_{m/b})\} \longleftarrow (\boldsymbol{X}, \boldsymbol{t})$

`// Shuffle and create batch pairs`

for $i \leftarrow 1$ ***to*** m/b **do**

Take i^{th} pair $(\acute{\boldsymbol{X}}_i, \acute{\boldsymbol{t}}_i)$

Forward propagate for $\acute{\boldsymbol{Y}}_i = f(\acute{\boldsymbol{X}}_i)$ and $\acute{\boldsymbol{B}}_i = g(\acute{\boldsymbol{X}}_i)$

Take a gradient step for $\mathcal{L}(\acute{\boldsymbol{Y}}_i, \acute{\boldsymbol{t}}_i) + c_\alpha \alpha(\acute{\boldsymbol{B}}) + c_\beta(1 - \beta(\acute{\boldsymbol{B}})) + c_F ||\acute{\boldsymbol{B}}||_F^2$

until stopping criteria is met

3.2.3 Training and Annotation Assignment

Training of the proposed framework is performed according to simultaneous supervised and unsupervised updates resulting from the objective function given in (3.6). We adopt stochastic gradient descent in mini-batch mode [12] for optimization. Algorithm 2 below describes the entire training procedure.

After the training phase is completed, the network is simply truncated by completely disconnecting the pooling layer as shown in Figure 3.3 and the rest of the network with trained weights is used to assign the annotations to each example. This assignment can be described as

$$y_i := \underset{1 \leq j \leq n}{\operatorname{argmax}} Z_{ij} \tag{3.7}$$

where y_i is the annotation assigned to the i^{th} example such that $y_{i_p} := (y_i - 1) \bmod n_p + 1$ and $y_{i_s} := (y_i - 1) \backslash n_p + 1$ are corresponding parent (learned through the provided supervision) and sub-class (learned through the unsupervised exploration) indices, respectively.

3.2.4 Graph Interpretation of the Proposed Framework

GAR regularizers, *affinity* and *balance*, were originally proposed for the classical type of semi-supervised learning problems where the number of labeled observations is much

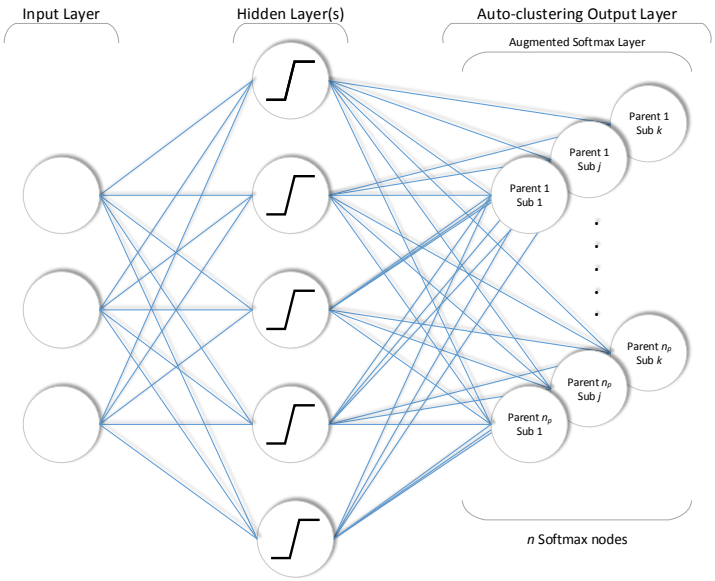

Figure 3.3: After training, the pooling layer is simply disconnected. The rest of the network with trained weights is used to obtain the assigned annotations. This operation can be described as $y_i := \text{argmax}_{1 \leq j \leq n} Z_{ij}$ where $y_{i_p} := (y_i - 1) \bmod n_p + 1$ and $y_{i_s} := (y_i - 1) \backslash n_p + 1$ are corresponding parent and sub-class indices, respectively.

smaller than the number of unlabeled observations. These two terms are used to propagate these labels to unlabeled observations across the graph $\mathcal{G}_{\mathcal{M}}$ which is defined by $\boldsymbol{B}\boldsymbol{B}^T$. Unlike these problems, in this chapter, we consider a different case in which a coarse level of labeling is available for all observations but the model has to explore a fine level of latent annotations using this partial supervision. Therefore, a graph interpretation of the proposed framework in this chapter is rather different than the one described in the previous section.

To better understand how the provided partial supervision is propagated in order to reveal latent annotations, let us first note that even though softmax($\boldsymbol{Z}$) produces probabilistic output, it is an increasing function of $\boldsymbol{Z}$ similar to $\max\big(\boldsymbol{0}, \boldsymbol{Z}\big)$. Assuming softmax($\boldsymbol{Z}$) $\approx$ $\max\big(\boldsymbol{0}, \boldsymbol{Z}\big)$ allows us to explicitly express $\boldsymbol{Y}$ in terms of $\boldsymbol{B}$ such that

$$\boldsymbol{Y} \approx \boldsymbol{B}\boldsymbol{W} \tag{3.8}$$

Noting that $\boldsymbol{W}\boldsymbol{W}^T$ is an $n \times n$ symmetric matrix such that

$$\boldsymbol{W}\boldsymbol{W}^T = \begin{bmatrix} \boldsymbol{I}_{n_p} & \boldsymbol{I}_{n_p} & \dots & \boldsymbol{I}_{n_p} \\ \boldsymbol{I}_{n_p} & \boldsymbol{I}_{n_p} & \dots & \boldsymbol{I}_{n_p} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}_{n_p} & \boldsymbol{I}_{n_p} & \dots & \boldsymbol{I}_{n_p} \end{bmatrix} \tag{3.9}$$

this assumption helps us visualize graph $\mathcal{G}_{\mathcal{M}}$ (whose edges are described by $\boldsymbol{B}\boldsymbol{B}^T$) as the spanning subgraph of $\mathcal{G}_{\mathcal{Y}}$ (whose edges are described by $\boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{B}\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{B}^T$). In other words, these two graphs are made up of the same vertices. However, while propagating the supervised adjacency introduced by $\mathcal{G}_{\mathcal{Y}}$ across $\mathcal{G}_{\mathcal{M}}$, GAR regularizers eliminate some of the edges of $\mathcal{G}_{\mathcal{Y}}$ from $\mathcal{G}_{\mathcal{M}}$ in a way that $\mathcal{G}_{\mathcal{M}}$ ultimately becomes a disconnected graph of n disjoint subgraphs. This propagation/elimination process is better explained in the following section through empirical demonstration on real data along with the impact of provided supervision on the exploration of latent annotations.

3.3 Experimental Results

The models have been implemented in Python using Keras [14] and Theano [60]. Open source code is available at http://github.com/ozcell/lalnets that can be used to reproduce the experimental results obtained on the three image datasets, MNIST [35], SVHN [44] and CIFAR-100 [30] commonly used by previous researchers publishing in the field of semi-supervised learning at NIPS, TNNLS and other similar venues. Figures 2.1, 2.2 and 3.4 respectively illustrate 200 examples from MNIST, SVHN and CIFAR-100 datasets and Table 3.1 summarizes the properties of these datasets used in the experiments. The CIFAR-100 dataset consists of 60000 32×32 color images in 100 classes. These 100 classes are grouped into 20 *superclasses*. Each image comes with a fine label, i.e. the *class* to which it belongs and a coarse label, i.e. the *superclasses* to which it belongs.

Figure 3.4: CIFAR-100 Dataset.

Table 3.1: Datasets used in the experiments. Refer to Table 2.1 for MNIST and SVHN.

	Data type	Number of examples	Dimension	Number of classes	% of largest class
CIFAR-100	Image: Objects	Train: 50000, Test: 10000	$3 \times 32 \times 32$	Fine: 100, Coarse: 20	Fine: 1%, Coarse: 5%

All experiments have been performed using the 6-layer convolutional neural network (CNN) model described in Table 3.2. For MNIST and SVHN experiments, coefficients of GAR terms have been set as $c_\alpha = 0.1, c_\beta = 0.1$, $c_F = 0.0003$ based on the experiments on a validation set whose examples were randomly chosen from the training set and supervision is introduced as a two parent-class classification problem, i.e. $n_p = 2$, as further explained in the following sections. CIFAR-100 naturally involves two levels of labeling where $n_p = 20$. For CIFAR-100, we use the same values for c_α and c_β but change c_F to 10^{-7} based on the validation set experiments, which might possibly be caused due to the difference in n_p settings of the datasets. For all experiments, we used a batch size of 128. Each experiment has been

repeated 10 times using different random initializations of the network. A validation set of 1000 examples has been chosen randomly among the training set examples to determine the epoch to report the test performance, which is obtained by application to the examples not introduced to the model during training, as is standard. For the sake of fairness, to obtain the performances of the models used for comparison, all training examples of the datasets are used for the pretraining of autoencoder-based models, and datasets are later pre-divided into two subsets according to the created problem (e.g. whether a digit is smaller or larger than 5) where two individual clusterings are performed within these subsets. The overall performances are obtained by combining the results of these two clusterings. Following [25], we evaluate test performance with unsupervised clustering accuracy given as

$$ACC = \max_{\mathfrak{f} \in \mathfrak{F}} \frac{\sum_{i=1}^{m} 1\{t_i^* = \mathfrak{f}(y_i)\}}{m} \tag{3.10}$$

where t_i^* is the ground-truth label, y_i is the annotation assigned in (3.7), and $\mathfrak{F}$ is the set of all possible one-to-one mappings between assignments and labels. In case of mixed clusters between multiple classes, finding the best mapping maximizing (3.10) corresponds to assigning a cluster to the class that occurs most in that cluster

Table 3.2: Specifications of the CNN model used in the experiments†.

Model name	Specification
6-layer CNN	2*Conv(32x3x3) - MP(2x2) - Drop(0.2) - 2*Conv(64x3x3) - MP(2x2) - Drop(0.3) - FC(2048) - Drop(0.5) - ACOL(n_p, k_s)

† Inputs of the models are determined according to the dimensions of the dataset being used for the training.
FC(i): Fully connected layer with i units
Drop(i): Applying dropout where the probability of retaining a unit is $1 - i$
Conv($i \times j \times k$): Convolution layer where i corresponds to the number of filters and $j \times k$ to the kernel size
MP($i \times j$): Max pooling with pool size of $i \times j$, i.e. factors by which to downscale (vertical $\times$ horizontal)
ACOL(n_p, k_s): ACOL with n_p parent-classes where k_s is the number of softmax duplicates per each parent-class

3.3.1 MNIST

To empirically demonstrate the label propagation process and compare the proposed approach with other methods, we created a semi-supervised problem using MNIST by providing parent-class supervision on whether a digit is smaller or larger than 5 such that

$$t_i = \begin{cases} 0 & \text{if } digit < 5 \\ 1 & \text{otherwise} \end{cases} \tag{3.11}$$

Figure 3.5 visualizes the realization of label propagation using the real predictions obtained for the test set of 10000 examples. It's worth noting that, for the sake of clarity, Figure 3.5 visualizes only a randomly selected 250 examples out of the entire test set of 10000 examples. Colored circles denote the ground-truths for the vertices, i.e. examples, and gray lines denote the edges, i.e. weighted connections between the examples representing their similarity. Note that, for vertices in graph $\mathcal{G}_\mathcal{Y}$, there are two different colors indicating true parent-class label assigned in (3.11), albeit ten different colors indicating the real *digit* identity for vertices in graph $\mathcal{G}_\mathcal{M}$. Graph $\mathcal{G}_\mathcal{M} = (\mathcal{M}, \mathcal{E})$ shares the same vertices $\mathcal{M}$ with graph $\mathcal{G}_\mathcal{Y} = (\mathcal{M}, \mathcal{E}_\mathcal{Y})$, which is constructed per the provided supervision. However, $\mathcal{E}$ is a subset of $\mathcal{E}_\mathcal{Y}$ as some of the edges in graph $\mathcal{G}_\mathcal{Y}$, such as those between the examples of digit 0 and 1, are eliminated in graph $\mathcal{G}_\mathcal{M}$ due to GAR regularization terms. As training continues, the provided supervision turns graph $\mathcal{G}_\mathcal{Y}$ into a disconnected graph of $n_p = 2$ disjoint subgraphs and implicitly propagates to its spanning subgraph $\mathcal{G}_\mathcal{M}$ in a way that $\mathcal{G}_\mathcal{M}$ becomes a disconnected graph of $n = 10 = n_p k_s$ disjoint subgraphs where $k_s = 5$ for this experiment.

Figure 3.6 presents the t-SNE [38] visualization of the latent space inferred by $\boldsymbol{Z}$ for a randomly chosen 2000 test examples without stratification from MNIST, where the portion of the largest class is 11%. It's also worth noting that this reduction, which is due to

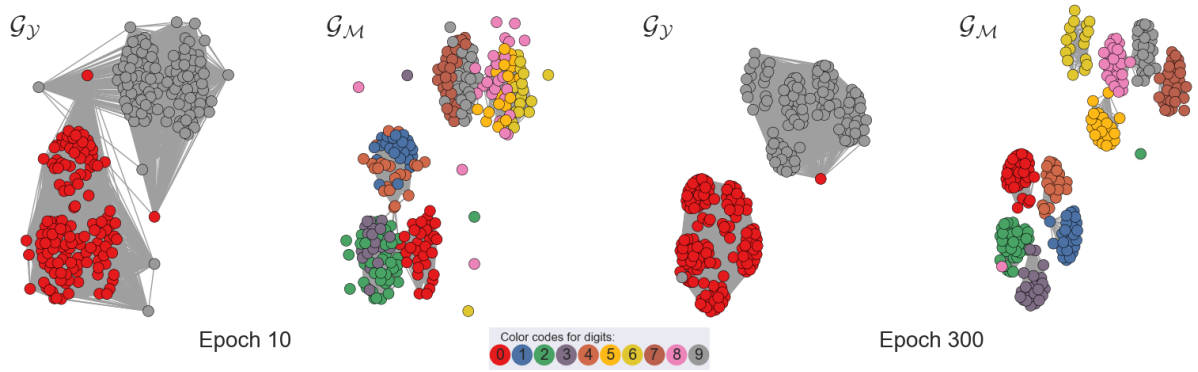

Figure 3.5: Visualizations of the graph $\mathcal{G}_{\mathcal{Y}}$ and its spanning subgraph $\mathcal{G}_{\mathcal{M}}$ for a randomly chosen 250 test examples from MNIST. Colored circles denote the ground-truths for the vertices, i.e. examples, and gray lines denote the edges, i.e. weighted connections between the examples representing their similarity. Note that, for vertices in graph $\mathcal{G}_{\mathcal{Y}}$, there are two different colors indicating true parent-class label assigned in (3.11), albeit ten different colors indicating the real *digit* identity for vertices in graph $\mathcal{G}_{\mathcal{M}}$. As training continues, provided supervision turns graph $\mathcal{G}_{\mathcal{Y}}$ into a disconnected graph of $n_p = 2$ disjoint subgraphs and implicitly propagates to graph $\mathcal{G}_{\mathcal{M}}$ in a way that $\mathcal{G}_{\mathcal{M}}$ becomes a disconnected graph of $n = 10 = n_p k_s$ disjoint subgraphs where $k_s = 5$ for this experiment. This figure is best viewed in color.

the computational complexity of the t-SNE technique, is performed only for visualization purposes; however, all numeric performance metrics are calculated using the entire test sets. From epoch 1 to epoch 300 of the training, clusters become well-separated and simultaneously the test accuracy increases. As clearly observed from this figure, using the provided partial supervision, the neural network also reveals some hidden patterns useful to distinguish the examples of different digits under the same parent-class and ultimately learns to categorize each one of the ten digits. Also for comparison, Figure 3.7 provides latent space visualizations obtained using three other approaches along with ACOL. In order to introduce the same two-parent supervision to other approaches, the dataset is first divided into two subsets according to the provided supervision and then distinct latent spaces obtained for each one of the subsets are combined for the final result. Table 3.3 summarizes the test error rates calculated using the unsupervised clustering accuracy metric given in (3.10) for MNIST with

$k_s = 5$. Results of a broad range of recent existing solutions are also presented for comparison. VaDE [25], unsupervised generative clustering framework combining Variational Autoencoders (VAE) and Gaussian Mixture Model (GMM) together, produces more competitive results with respect to other approaches as it adopts variational inference during the reconstruction process and enables the simultaneous updating of the GMM parameters and the network parameters. On the other hand, unlike VaDE and other similar approaches based on the reconstruction of the input, ACOL motivates neural networks to learn the latent space representation through the provided partial supervision (e.g. whether a digit is smaller or larger than 5), which is typically more general than the overall categorization interest (e.g. finding the real digit identity $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$). This motivation yields a better separation of the clusters in the latent space; however, its quality depends on the provided supervision as further explored in the following section.

Table 3.3: Benchmark results for the two-parent case (whether a digit is smaller or larger than 5) on MNIST. The last row demonstrates the benchmark scores of the proposed framework in this chapter. Note that given values represent the test error.

	MNIST $k = k_s = 5$
AE+k-means	21.80%
AE+GMM	23.80%
VaDE [25]	8.18%
ACOL	1.39%($\pm$0.12)

3.3.2 MNIST - Impact of the Provided Supervision on Performance

We perform two experiments on MNIST in order to observe the impact of provided supervision and whether useful information is propagated between parent-classes for better

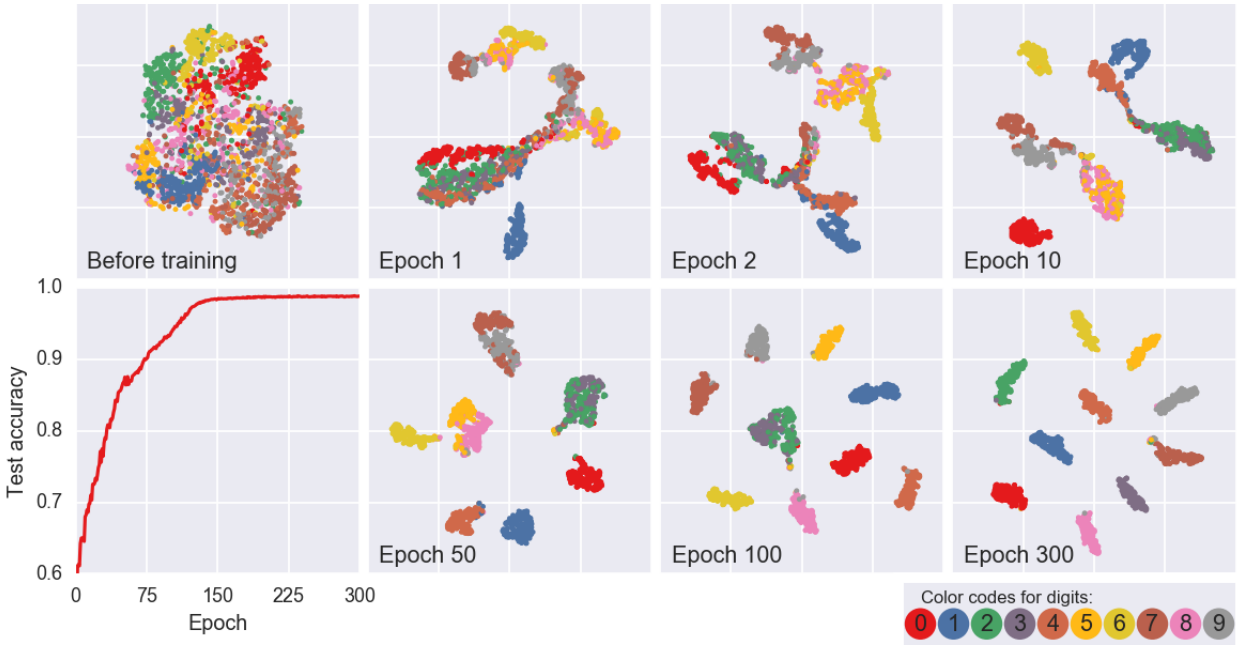

Figure 3.6: t-SNE visualization of the latent space inferred by $\boldsymbol{Z}$ for a randomly chosen 2000 test examples from MNIST. Color codes denote the ground-truths for the examples. Note the separation of clusters from epoch 1 to epoch 300 of the training. For reference, accuracy for the entire test set is also plotted with respect to the training epochs. This figure is best viewed in color.

sub-classification. In the first experiment, we used the same supervision described in (3.11) by leaving the first parent-class unchanged throughout the experiment but discarding all examples of a digit from the second parent-class in each new iteration. For all five iterations of this experiment, Figure 3.8 presents the t-SNE visualization of the latent space representation observed for a randomly chosen 2000 test examples only from the first parent-class, i.e. {0,1,2,3,4}, and Table 3.4 summarizes the overall test errors calculated only over the classes whose examples are included in the training. One might expect to observe better performance when the clustering problem under one of the parent-classes is simplified. On the contrary, a more challenging objective forces the network to reveal more latent patterns needed to better differentiate each of the digits. More specifically, when the second parent-class consists only

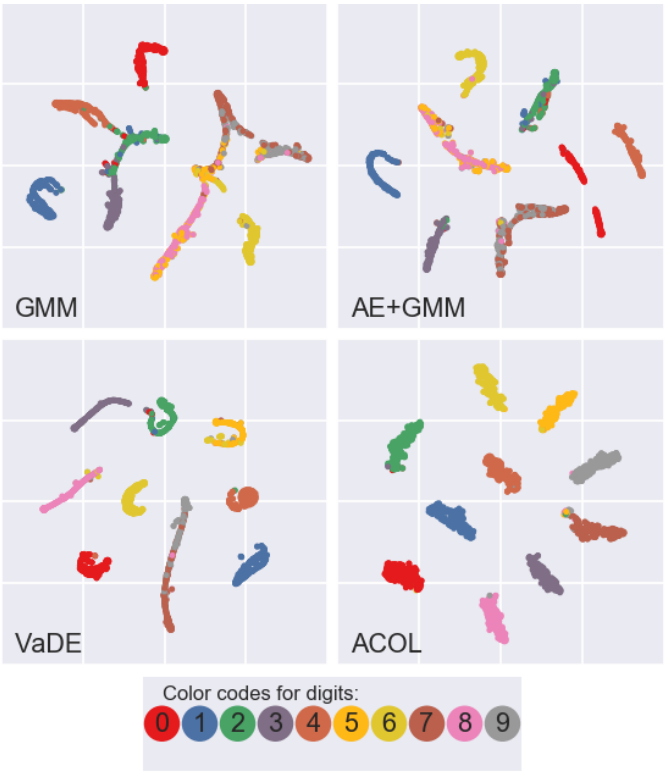

Figure 3.7: t-SNE visualization of the latent spaces obtained using four different approaches for a randomly chosen 2000 test examples from MNIST. In order to introduce the same two-parent supervision to other approaches, the dataset is first divided into two subsets according to the provided supervision and then distinct latent spaces obtained for each one of the subsets are combined for the final result. Color codes denote the ground-truths for the examples. Note the more definitive separation of clusters when using ACOL. This figure is best viewed in color.

of the examples of the digit 5, the network learns only to distinguish its examples from those of the first five digits. Adding the examples of digit 6, the network now has to extract more hidden patterns identifying the unique differences between the set of the digits $5, 6$ and the set of digits $0, 1, 2, 3, 4$. These extra hidden patterns also contribute to the differentiation of the digits $0, 1, 2, 3, 4$ from each other, which we identify as the *inter-parent* effect of the provided supervision.

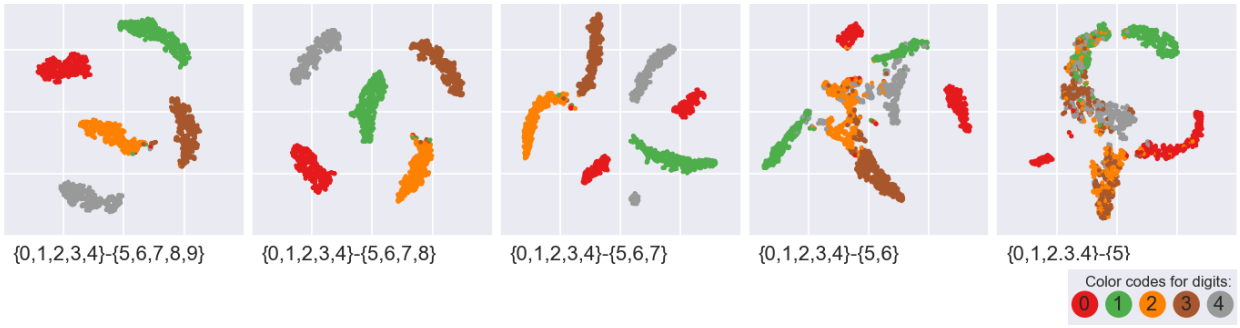

Figure 3.8: t-SNE visualization of the latent spaces obtained for the first parent-class, $\{0, 1, 2, 3, 4\}$, showing the inter-parent effect of the provided supervision. In this scenario, the first parent-class is left unchanged throughout the experiment but all examples of a digit are discarded from the second parent-class in each new iteration. When the classification objective becomes more challenging due to more distinct digits in the second parent-class, the network is more capable of revealing the hidden patterns needed to better differentiate the examples of the first parent. Color codes denote the ground-truths for the examples. This figure is best viewed in color.

In the second experiment, rather than using the rule given in (3.11), we randomly assign the parent-classes. That is, examples of a randomly chosen five digits are used to construct the first parent-class and those of the remaining five digits form the second one. The experiment is repeated 74 times with a new selection of parent-classes where combination repetitions are prevented. Histogram of the test accuracies observed for these 74 repetitions is given in Figure 3.9 and Table 3.5 summarizes the best, the median and the worst cases along with the overall average. For reference, k-means results obtained for the same 74 scenarios are also provided. For k-means, initialization was performed by following k-means++ technique [4], which is based on the idea that spreading out the k initial cluster centers can prevent arbitrarily bad clusterings. This method chooses cluster centers at random from the data points, but weighs the data points according to their squared distance from the closest cluster center already chosen [4]. One can observe that ACOL is less sensitive to variations in the provided supervision as a majority of the iterations are concluded with a test accuracy within 1.5% range. For k-means, provided supervision only determines the difficulty of the

Table 3.4: Benchmark results for the two-parent case on MNIST to observe the inter-parent effect of the provided supervision. Reported overall test accuracies are calculated only over the classes whose examples are included in the training.

1^{st} Parent	2^{nd} Parent	Test Error
{0,1,2,3,4}	{5,6,7,8,9}	1.39%(±0.12)
{0,1,2,3,4}	{5,6,7,8}	3.83%(±2.09)
{0,1,2,3,4}	{5,6,7}	4.04%(±0.92)
{0,1,2,3,4}	{5,6}	16.44%(±3.33)
{0,1,2,3,4}	{5}	26.55%(±2.64)

subsequent clustering tasks. As these clusterings are performed individually, they don't have any effect on performances of each other. However, in ACOL, there is a much more complex relation between the provided supervision and the observed performance. Even though the assigned parent-classes yield a more difficult unsupervised clustering task (e.g. when digits 4 and 9 are under the same parent-class), ACOL can compensate this effect with the help of latent patterns learned through inter-parent comparisons due to the classification objective. Therefore, when a coarse level of labeling is available for a dataset, simultaneous classification and clustering through ACOL produces better separated and more accurate latent embeddings than individual clustering tasks within each of the known labels as ACOL enables neural networks to exploit the provided supervision.

3.3.3 SVHN and CIFAR-100

We also test the proposed approach for more realistic and challenging scenarios using the SVHN and CIFAR-100 datasets. For SVHN, we adopt the same supervision as defined in (3.11). On the other hand, the CIFAR-100 dataset naturally defines two levels of labeling. Each image has a coarse label indicating the *superclass* to which it belongs, such as *trees* and also a fine label indicating the *class* to which it belongs, such as *maple*. The 100 *classes*

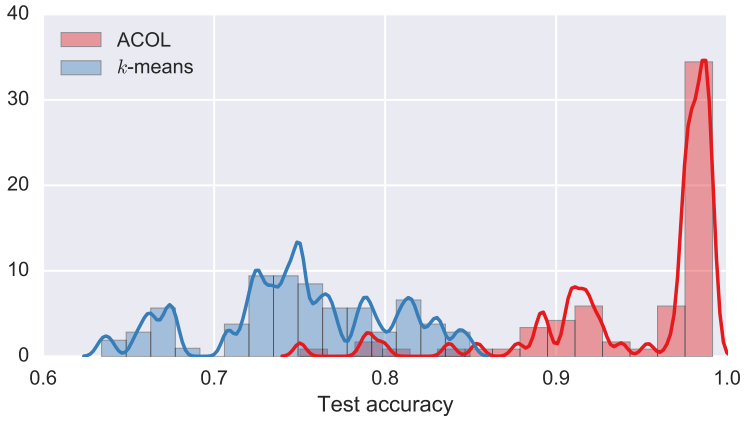

Figure 3.9: Normalized histogram of the test accuracies obtained using ACOL for a randomly chosen 74 non-repeating two-parent supervision scenarios on MNIST. For comparison, the same scenarios are also tested using the k-means algorithm.

Table 3.5: Test errors for worst, median and best cases among 74 non-repeating two-parent supervision scenarios on MNIST. For k-means, k is set as 5, i.e. $k = k_s = 5$, and initialization is performed by the k-means++ technique based on the idea that spreading out the k initial cluster centers can prevent arbitrarily bad clusterings [4].

	Worst	Median	Best	Mean
k-means	36.58%	24.99%	15.04%	24.94%($\pm$1.23)
ACOL	24.98%	2.25%	0.85%	5.09%($\pm$1.28)

in the CIFAR-100 are grouped into 20 *superclasses*. For the CIFAR-100 experiment, coarse labels are provided as the supervision and fine labels are targeted as the annotations to be observed. Tables 3.6 and 3.7 respectively summarize the test performances obtained for the SVHN and CIFAR-100 datasets with two different k_s settings. Results of a broad range of other approaches in the current literature are also presented for comparison.

It is worth noting that, unlike MNIST and SVHN, the proposed approach suffers from a limitation when using CIFAR-100 dataset. That is, the parent-wise classification performance of the model affects the accuracy of assigned annotations. In the MNIST and SVHN datasets,

once trained with the provided parent-classes, the networks generalize well to the test sets based on this supervision. Hence, when the sub-class annotations are obtained for training and test examples, models yield approximately the same accuracy. However, in CIFAR-100, training with 20 *superclasses*, the network cannot generalize well to the test examples as it achieves 97.76% classification accuracy on the training set but cannot go beyond 70% on the test set. In other words, generalization problem observed on the CIFAR-100 dataset is not caused by the regularization but the supervised part of the training. To monitor this effect, training set performances are also presented in Table 3.7. We would like to emphasize that coarse labels (parent-classes) are introduced to the network only for the training set examples, not for those in the test set. Fine labels, on the other hand, are never introduced to the model in any part of the training for any example.

Table 3.6: Benchmark error results for the two-parent case on SVHN. The last row demonstrates the benchmark scores of the proposed framework in this chapter. Note that given values represent the test error.

	SVHN $k = k_s = 5$	SVHN $k = k_s = 10$
AE+k-means	67.29%	62.42%
AE+GMM	69.38%	63.89%
ACOL	36.90%($\pm$6.22)	21.66%($\pm$1.49)

Table 3.7: Benchmark error results for the twenty-parent case on CIFAR-100. The last row demonstrates the benchmark scores of the proposed framework in this chapter. Note that given values represent the test error.

	Training $k = k_s = 5$	Test $k = k_s = 5$	Training $k = k_s = 10$	Test $k = k_s = 10$
AE+k-means	64.60%	63.81%	61.17%	59.63%
AE+GMM	65.94%	65.45%	62.17%	61.20%
ACOL	44.64%($\pm$0.79)	62.17%($\pm$0.32)	37.41%($\pm$0.42)	58.60%($\pm$0.18)

CHAPTER 4: LEARNING LATENT REPRESENTATIONS IN NEURAL NETWORKS FOR UNSUPERVISED CLUSTERING THROUGH PSEUDO SUPERVISION AND GRAPH-BASED ACTIVITY REGULARIZATION[1]

Clustering, the unsupervised process of grouping similar examples together, is one of the most fundamental challenges in machine learning research and has been studied extensively in different aspects such as feature selection, distance functions, grouping methods, etc. [1]. k-means [39] and Gaussian Mixture Models (GMM) [9] are two well-known conventional clustering algorithms that are applicable to a wide range of problems. Traditionally, these methods are applied to low-level features such as gradient-orientation histograms (HOG) for images. Therefore, their distance metrics are limited to local relations in the data space and inadequate to represent hidden dependencies in latent spaces. On the other hand, spectral clustering [63] is another conventional approach producing more flexible distance metrics than k-means and GMM. However, these types of solutions are not scalable to large datasets as they need to compute the full graph Laplacian matrix. Approximation approaches proposed to trade off performance for speed can scale spectral clustering to large datasets [71].

In recent years, researchers have focused on the unsupervised learning of high-level features on which to apply clustering and shown that learning good representations is important for the accuracy and robustness of the clustering task. Deep Embedding Clustering (DEC) [69] was proposed to simultaneously learn feature representations and cluster assignments using deep neural networks (DNN). In this approach, first DNN parameters are initialized with a layer-wise trained deep autoencoder [62] and then the initialized DNN

[1]This chapter has been submitted to 6th International Conference on Learning Representations (ICLR 2018) and is now under review.

is used to obtain the latent representation on which to perform k-means clustering for the initialization of cluster centers. This complicated initialization is followed by a challenging optimization process that minimizes the Kullback–Leibler (KL) divergence between the centroid-based probability distribution and the auxiliary target distribution derived from the soft cluster assignments. Similarly, Joint Unsupervised Learning (JULE) [72] combines agglomerative clustering with convolutional neural networks (CNN) and formulates them as a recurrent process. Although JULE proposes an end-to-end learning framework, it suffers scalability issues due to its agglomerative clustering.

Novel deep generative models that can be trained via direct backpropagation have recently been proposed avoiding the difficulties in preexisting generative models such as Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN) and Deep Boltzmann Machines (DBM) that are trained by MCMC-based algorithms [22, 55]. Among two canonical examples of these models, Variational Autoencoders (VAE) [29, 53] integrate stochastic latent variables into the conventional autoencoder architecture while Generative Adversarial Networks (GAN) [20] propose an adversarial training procedure implementing a min-max adversarial game between two neural networks: the discriminator and the generator. Following these advances, researchers have started to study new hybrid models with the goal of performing unsupervised clustering through deep generative models. For example, Variational Deep Embedding (VaDE) [25] proposed a clustering framework combining VAE and GMM together. Also, Gaussian Mixture Variational Autoencoder (GMVAE) [16] built upon the semi-supervised model by [28] to perform unsupervised clustering within the VAE framework with a Gaussian mixture as a prior distribution. GAN-based methods include: Categorical Generative Adversarial Networks (CatGAN) [57], an approach incorporating neural network classifiers with an adversarial generative model, and Adversarial Autoencoder (AAE) [40], a probabilistic autoencoder variant integrating traditional reconstruction error with adversarial training criterion of GANs. Besides, [47] proposes to fuse the disentangled features learned

by Information Maximizing Generative Adversarial Networks (InfoGAN), an extension to GANs that uses mutual information to induce representation, with k-means clustering.

In this chapter, we propose a novel unsupervised clustering approach building upon the previous study on learning of latent annotations in a particular semi-supervised setting where a coarse level of supervision is available for all observations, i.e. parent-class labels, but the model has to learn a fine level of latent annotations, i.e. sub-classes, under each one of these parents. For clarification, as before assume that we are given a dataset of hand-written digits such as MNIST [35] where the overall task is the complete categorization of each digit, but the only available supervision is whether a digit is smaller or greater than 5. To study this particular semi-supervised setting on neural networks, the previous chapter proposed a novel output layer modification, Auto-clustering Output Layer (ACOL). ACOL allows simultaneous supervised classification (per provided parent-classes) and unsupervised clustering (within each parent) where clustering is performed through Graph-based Activity Regularization (GAR) technique proposed in the second chapter. More specifically, as ACOL duplicates the softmax nodes at the output layer for each class, GAR allows for competitive learning between these duplicates on a traditional error-correction learning framework.

To learn latent annotations in a fully unsupervised setup, we substitute the real, yet unavailable, parent-class information with a pseudo one. More specifically, we choose a domain specific transformation to be applied to the observations in a dataset to generate examples for a pseudo parent-class. The transformed dataset constitutes the examples of that pseudo parent-class and every new transformation generates a new one. Regarding the MNIST example for this fully unsupervised setting, now we simply augment the dataset by applying a transformation to examples, e.g. rotating by 90^o, and label transformed examples as *rotated* and non-transformed examples as *original*. This new augmented dataset is provided to the network as a two-class classification problem with pseudo classes labeled as *original* and *rotated* as visualized in Figure 4.1. While being trained over this pseudo supervision,

through ACOL and GAR, the neural network learns the latent representation distinguishing the real digit identities in an unsupervised manner.

The idea of employing an auxiliary task to learn a good data representation has been previously studied for different domains [2, 15]. Most recent study, Exemplar CNN [17], proposed to use a regularizer enforcing the feature representation to be approximately invariant to the transformations while training the network to discriminate between a set of pseudo parent-classes ("surrogate classes" with their definition). This approach requires thousands of transformations to obtain a good representation and also it cannot exploit more than 300 examples per "surrogate class" severely limiting its scalability. Furthermore, some elementary transformations, such as rotation, have only a minor impact on the performance. In comparison, in our approach, only 8 pseudo parent-classes generated by rotation-based transformations, i.e. 0^o, 90^o, 180^o, 270^o rotations of original images and 0^o, 90^o, 180^o, 270^o rotations of horizontally flipped images, provide a rich latent representation to obtain state-of-the-art unsupervised clustering performance.

4.1 Background

In the second chapter, GAR has been originally proposed for the classical type of semi-supervised setting where the number of labeled observations is much smaller than the number of unlabeled observations. We have shown that defining the objective of the regularization over the matrix $\boldsymbol{N}$ yields a scalable and efficient graph-based solution and that the entire operation corresponds to propagating the available labels across the graph $\mathcal{G}_{\mathcal{M}}$ whose edges are specified by the $m \times m$ symmetric matrix $\boldsymbol{M} := \boldsymbol{B}\boldsymbol{B}^T$ that infers the adjacency of the examples based on the predictions of the neural network. More specifically, it has been shown that as the matrix $\boldsymbol{N}$ turns into the identity matrix, $\mathcal{G}_{\mathcal{M}}$ becomes a disconnected graph including n disjoint subgraphs each of which is $^m/_n$-regular. This indicates

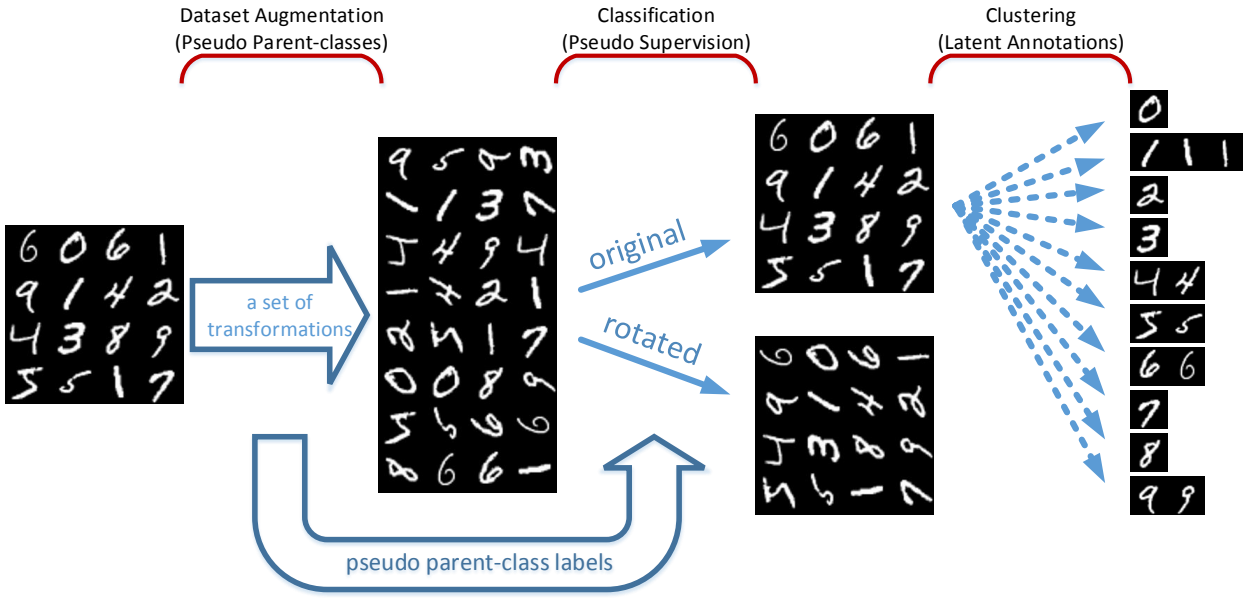

Figure 4.1: Assume that we are given a dataset of hand-written digits such as MNIST where the overall task is the complete categorization of each digit. Then, we simply augment the dataset by applying a transformation to examples, e.g. rotating by 90^o, and label each of them either as *original* or as *rotated*. This new augmented dataset is provided to the network as a two-class classification problem. While being trained over this pseudo supervision, through ACOL and GAR, the neural network also learns the latent representation distinguishing the real digit identities in unsupervised an manner.

that the strong adjacencies in the matrix $\boldsymbol{M}$ get stronger, weak ones diminish and each label is propagated to m/n examples through the strong adjacencies.

On the other hand, in the particular semi-supervised setting considered in the third chapter (i.e. a coarse level of labeling is available for all observations but the model still needs to learn a fine level of latent annotation for each one of them), when applied to an ACOL network, GAR provides that the latent information introduced by the coarse supervision is propagated from the graph $\mathcal{G}_{\mathcal{Y}}$ (whose edges are specified by $m \times m$ symmetric matrix $\boldsymbol{Y}\boldsymbol{Y}^T$) to its spanning subgraph $\mathcal{G}_{\mathcal{M}}$ to reveal deeper latent annotations. In other words, although these two graphs are made up of the same vertices (m examples) while propagating the latent information that is captured through supervised adjacency introduced by $\mathcal{G}_{\mathcal{Y}}$ across

$\mathcal{G}_{\mathcal{M}}$, GAR terms eliminate some of the edges of $\mathcal{G}_{\mathcal{Y}}$ from $\mathcal{G}_{\mathcal{M}}$ in a way that $\mathcal{G}_{\mathcal{M}}$ ultimately becomes a disconnected graph of n disjoint subgraphs each of which now corresponds to a latent annotation.

4.2 Proposed Framework

4.2.1 Objective Function

The unsupervised clustering approach proposed in this chapter adopts the same framework introduced in the previous chapter. Since the real parent-class labels (a digit is smaller or greater than 5) are unavailable in a fully unsupervised setting, we randomly assign pseudo parent-class labels each of which is associated with a domain specific transformation used to generate the examples of that pseudo parent-class.

In this setting, n_p now corresponds to the number of pseudo parent-classes and $\tilde{\boldsymbol{t}} = [\tilde{t}_1 \dots \tilde{t}_m]^T$ is a vector of randomly assigned pseudo parent-class labels which are uniformly distributed across n_p pseudo parent-classes such that $\tilde{t}_i \in \{1, ..., n_p\}$. Also, there exists a set of transformations $\mathcal{S}_{\mathcal{T}} = \{\mathcal{T}_1, ..., \mathcal{T}_{n_p}\}$ where transformation $\mathcal{T}_j$ is used to generate the examples of the j^{th} pseudo parent-class such that $\tilde{\boldsymbol{x}}_i = \mathcal{T}_j(\boldsymbol{x}_i)$. $\mathcal{S}_{\mathcal{T}}$ also includes non-transformation $\mathcal{T}_1$ providing $\tilde{\boldsymbol{x}}_i = \mathcal{T}_1(\boldsymbol{x}_i) = \boldsymbol{x}_i$ to ensure that the original observations are introduced to the network during training. $\tilde{\boldsymbol{t}}$ is associated with a vector of transformations $\boldsymbol{T} = [T_1 \dots T_m]^T$ such that $T_i = \mathcal{T}_{\tilde{t}_i}$.

Let $\odot$ be an element-wise operation defined between the vector of transformations $\boldsymbol{T}$ and the original input feature map $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_m]^T$ such that

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1 \\ \tilde{\boldsymbol{x}}_2 \\ \vdots \\ \tilde{\boldsymbol{x}}_m \end{bmatrix} = \boldsymbol{T} \odot \boldsymbol{X} = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_m \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_m \end{bmatrix} = \begin{bmatrix} T_1(\boldsymbol{x}_1) \\ T_2(\boldsymbol{x}_2) \\ \vdots \\ T_m(\boldsymbol{x}_m) \end{bmatrix} = \begin{bmatrix} \mathcal{T}_{\tilde{t}_1}(\boldsymbol{x}_1) \\ \mathcal{T}_{\tilde{t}_2}(\boldsymbol{x}_2) \\ \vdots \\ \mathcal{T}_{\tilde{t}_m}(\boldsymbol{x}_m) \end{bmatrix} \tag{4.1}$$

where $\tilde{\boldsymbol{X}}$ corresponds to the modified input feature map by the vector of transformations $\boldsymbol{T}$, which is associated with the randomly assigned pseudo labels $\tilde{\boldsymbol{t}}$. The output of the entire network and the positive part of the augmented softmax layer activities respectively become $\boldsymbol{Y} = f(\tilde{\boldsymbol{X}})$ and $\boldsymbol{B} = g(\tilde{\boldsymbol{X}})$. Then, the objective function defined in (3.6) can simply be adopted by substituting the real, yet unavailable, observation-label pair $(\boldsymbol{X}, \boldsymbol{t})$ with a pseudo one $(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{t}})$ such that

$$\mathcal{L}\big(f(\tilde{\boldsymbol{X}}), \tilde{\boldsymbol{t}}\big) + \mathcal{U}\big(g(\tilde{\boldsymbol{X}})\big) = \mathcal{L}\big(\boldsymbol{Y}, \tilde{\boldsymbol{t}}\big) + c_\alpha \alpha\big(\boldsymbol{B}\big) + c_\beta\big(1 - \beta\big(\boldsymbol{B}\big)\big) + c_F ||\boldsymbol{B}||_F^2 \tag{4.2}$$

4.2.2 Modified *Affinity* and *Balance* Terms

Recall that an $n \times n$ symmetric matrix $\boldsymbol{N} = \boldsymbol{B}^T\boldsymbol{B}$ specifies the edges of the graph between the softmax duplicates and that GAR terms have been proposed to regularize the matrix $\boldsymbol{N}$ in a way that it turns into the identity matrix. While the objective of *affinity*, i.e. penalizing the non-zero off-diagonal entries of $\boldsymbol{N}$, corresponds to assigning an example to only one softmax node with the probability of 1, the objective of *balance*, i.e. equalizing diagonal entries of $\boldsymbol{N}$, corresponds to preventing collapsing onto a subspace of dimension less than n.

Among the off-diagonal entries of $\boldsymbol{N}$ determining the *affinity* cost, for each one of n softmax nodes, there exist $k_s - 1$ entries describing its relation with the other duplicates of the same parent-class (let us define them as *intra-parent* entries) and $(n_p - 1)k_s$ entries describing its relation with the softmax nodes belonging to other parent-classes (let us define them as *inter-parent* entries). While *inter-parent* entries are explicitly affected by the pseudo classification objective as well as the regularization, *intra-parent* entries do not experience the classification directly. Therefore, the *affinity* cost due to *inter-parent* entries is minimized at a different rate than the *affinity* cost due to *intra-parent* entries. On the other hand, as it is calculated over the diagonal entries of $\boldsymbol{N}$, the *balance* cost does not either experience the

pseudo classification objective explicitly. As a result, due to the direct impact of the pseudo classification objective which is observed only on the *affinity* cost, the weighting between the regularization terms actively alters during the training and needs to be re-tuned through the hyperparameters c_α and c_β. This effect can be observed more clearly as n_p, the number of parent-classes, increases.

To ensure a more robust regularization we introduce a modification for the *affinity* and *balance* terms: We discard all *inter-parent* entries of $\boldsymbol{N}$ and represent the remaining ones as a three dimensional tensor $\tilde{\boldsymbol{N}}$. Thus, $\tilde{\boldsymbol{N}}$ is a $k_s \times k_s \times n_p$ tensor such that $\tilde{\boldsymbol{N}}_{:,:,k}$ specifies the relations between k_s softmax duplicates of the k^{th} parent-class where $k \in \{1, ..., n_p\}$. Also, $\tilde{\boldsymbol{V}}$ is another $k_s \times k_s \times n_p$ tensor defined as

$$\tilde{\boldsymbol{V}}_{:,:,k} = [\tilde{N}_{1,1,k} \dots \tilde{N}_{k_s,k_s,k}]^T [\tilde{N}_{1,1,k} \dots \tilde{N}_{k_s,k_s,k}] \tag{4.3}$$

Then, the modified *affinity* and *balance* terms can be respectively written as

$$\tilde{\alpha}\big(\boldsymbol{B}\big) := \frac{1}{n_p} \sum_{k=1}^{n_p} \frac{\sum\limits_{i \neq j}^{k_s} \tilde{N}_{ijk}}{(k_s - 1) \sum\limits_{i=j}^{k_s} \tilde{N}_{ijk}} \tag{4.4}$$

$$\tilde{\beta}\big(\boldsymbol{B}\big) := \frac{1}{n_p} \sum_{k=1}^{n_p} \frac{\sum\limits_{i \neq j}^{k_s} \tilde{V}_{ijk}}{(k_s - 1) \sum\limits_{i=j}^{k_s} \tilde{V}_{ijk}} \tag{4.5}$$

and simply correspond to calculating the original terms given in (2.16), (2.17) on each 2-D $k_s \times k_s \times 1$ slice of $\tilde{\boldsymbol{N}}$ and $\tilde{\boldsymbol{V}}$ tensors and then averaging the results for n_p of them. Replacing these modified terms in (4.2), the overall modified objective function becomes

$$\mathcal{L}\big(f(\tilde{\boldsymbol{X}}), \tilde{\boldsymbol{t}}\big) + \mathcal{U}\big(g(\tilde{\boldsymbol{X}})\big) = \mathcal{L}\big(\boldsymbol{Y}, \tilde{\boldsymbol{t}}\big) + c_\alpha \tilde{\alpha}\big(\boldsymbol{B}\big) + c_\beta\big(1 - \tilde{\beta}\big(\boldsymbol{B}\big)\big) + c_F ||\boldsymbol{B}||_F^2 \tag{4.6}$$

4.2.3 Training and Cluster Assignments

Network parameters are trained by implementing the stochastic optimization method Adam [27] based on the objective given in (4.6). After training, k-means clustering is performed on the representation space observed in the hidden layer preceding the augmented softmax layer such that

$$\boldsymbol{F} = \boldsymbol{Y}^{(L-2)} = f^{(L-2)}(\boldsymbol{X}) \tag{4.7}$$

Recalling that the original examples are already introduced to the network as the examples of first pseudo parent-class through transformation $\mathcal{T}_1$, we obtain the latent space representation only for the original examples to perform k-means clustering.

One might suggest performing k-means clustering on the representation observed in the augmented softmax layer ($\boldsymbol{Z}$ or softmax($\boldsymbol{Z}$)) rather than $\boldsymbol{F}$. Properties and respective clustering performance of these representation spaces are empirically demonstrated in the following sections.

Algorithm 3 below describes the entire training and cluster assignment procedure.

4.3 Experiments

4.3.1 Experimental Setup and Datasets

The models have been implemented in Python using Keras [14] and Theano [60]. Open source code is available http://github.com/ozcell/lalnets that can be used to reproduce the experimental results obtained on three benchmark image datasets, MNIST [35], SVHN [44] and USPS. Figures 2.1, 2.2 and 4.2 respectively illustrate 200 examples from MNIST, SVHN and USPS datasets and Table 4.1 summarizes the properties of these datasets used in the experiments. The USPS dataset is a 16×16 grayscale image dataset of handwritten digits in which digits have been size-normalized. There are 7291 training observations and 2007 test observations.

Algorithm 3: Model training and cluster assignments

Input : $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_m]^T$, n_p,
a set of transformations $\mathcal{S}_{\mathcal{T}} = \{\mathcal{T}_1, ..., \mathcal{T}_{n_p}\}$,
batch size b, weighing coefficients c_α, c_β, c_F, the number of clusters k

repeat

- $\tilde{\boldsymbol{t}} \longleftarrow$ `random`(n_p) `// Randomly assign labels across` n_p `classes`
- $\boldsymbol{T} \longleftarrow [\mathcal{T}_{\tilde{t}_1}, ..., \mathcal{T}_{\tilde{t}_m}]$ `// Obtain the vector of transformations corresponding to` $\tilde{\boldsymbol{t}}$
- $\tilde{\boldsymbol{X}} \longleftarrow \boldsymbol{T} \odot \boldsymbol{X}$ `// Obtain the modified input`
- $\{(\acute{\boldsymbol{X}}_1, \acute{\boldsymbol{t}}_1), ..., (\acute{\boldsymbol{X}}_{m/b}, \acute{\boldsymbol{t}}_{m/b})\} \longleftarrow (\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{t}})$ `// Shuffle and create batch pairs`
- **for** $i \leftarrow 1$ ***to*** m/b **do**
  - Take i^{th} pair $(\acute{\boldsymbol{X}}_i, \acute{\boldsymbol{t}}_i)$
  - Forward propagate for $\acute{\boldsymbol{Y}}_i = f(\acute{\boldsymbol{X}}_i)$ and $\acute{\boldsymbol{B}}_i = g(\acute{\boldsymbol{X}}_i)$
  - Take a gradient step for $\mathcal{L}(\acute{\boldsymbol{Y}}_i, \acute{\boldsymbol{t}}_i) + c_\alpha \tilde{\alpha}(\acute{\boldsymbol{B}}_i) + c_\beta(1 - \tilde{\beta}(\acute{\boldsymbol{B}}_i)) + c_F ||\acute{\boldsymbol{B}}_i||_F^2$

until stopping criteria is met

$\boldsymbol{F} \longleftarrow f^{(L-2)}(\boldsymbol{X})$ `// Obtain latent space representation` $\boldsymbol{F}$ `for the original examples`

$\boldsymbol{y} \longleftarrow$ `kmeans`$(\boldsymbol{F}, k)$ `// Assign clusters by performing` k`-means on` $\boldsymbol{F}$

return : Cluster assignments $\boldsymbol{y}$

Table 4.1: Datasets used in the experiments. Refer to Table 2.1 for MNIST and SVHN.

	Data type	Number of examples	Dimension	Number of classes	% of largest class
USPS	Image: Hand-written digits	Train: 7291, Test: 2007	$1 \times 16 \times 16$	10	17%

All experiments have been performed on a 6-layer convolutional neural network (CNN) model whose specifications are given in Table 4.2 where coefficients of GAR terms have been set as $k_s = 20, c_\alpha = 0.1, c_\beta = 1$, $c_F = 0.000001$. Specifications of the 6-layer CNN model and coefficients of GAR terms have been chosen based on experiments on the entire training set. During training, pseudo supervised objective is introduced as an 8 pseudo parent-class

Figure 4.2: USPS Dataset.

classification problem, i.e. $n_p = 8$, through the following rotation-based transformations:

$$\mathcal{T}_i = \begin{cases} i = 1: & \text{No transformation} \\ i = 2: & \text{Rotate by } 90^o \\ i = 3: & \text{Rotate by } 180^o \\ i = 4: & \text{Rotate by } 270^o \\ i = 5: & \text{Flip horizontally} \\ i = 6: & \text{Flip horizontally + Rotate by } 90^o \\ i = 7: & \text{Flip horizontally + Rotate by } 180^o \\ i = 8: & \text{Flip horizontally + Rotate by } 270^o \end{cases} \tag{4.8}$$

We used the same batch size of 400 for all experiments, but it's worth noting that we didn't observe any significant difference during the experiments performed using different batch size settings, i.e. $\{128, 400, 1000\}$. Each experiment has been repeated 10 times using different random initializations of the network weights. To ensure that the representation obtained

through the proposed approach is well-generalized for never-seen-before data, we train the neural network parameters using only the training set examples of each dataset and obtain the clustering performances using k-means with $k = 10$ on the latent space representation $\boldsymbol{F}$ of the untransformed test set examples (through $\mathcal{T}_1$).

Table 4.2: Specifications of the CNN model used in the experiments†.

Model name	Specification
6-layer CNN	2*Conv(32x3x3) - MP(2x2) - Drop(0.2) - 2*Conv(64x3x3) - MP(2x2) - Drop(0.3) - FC(2048) - Drop(0.5) - ACOL(n_p, k_s)

† Inputs of the models are determined according to the dimensions of the dataset being used for the training.
FC(i): Fully connected layer with i units
Drop(i): Applying dropout where the probability of retaining a unit is $1 - i$
Conv($i \times j \times k$): Convolution layer where i corresponds to the number of filters and $j \times k$ to the kernel size
MP($i \times j$): Max pooling with pool size of $i \times j$, i.e. factors by which to downscale (vertical $\times$ horizontal)
ACOL(n_p, k_s): ACOL with n_p pseudo parent-classes where k_s is the number of softmax duplicates per each pseudo parent-class

4.3.2 Quantitative Comparison

Following [25] and [72], we evaluate the test performances using normalized mutual information (NMI) [70] and unsupervised clustering accuracy given in (3.10). Both metrics range between $[0, 1]$ where a larger value indicates more precise clustering results.

Figure 4.3 presents the t-SNE [38] visualizations of the latent space $\boldsymbol{F}$ throughout the training for a randomly selected 2000 untransformed test examples without stratification from MNIST, where the portion of the largest class is 11%. Each group corresponds to a cluster (i.e. a digit) under the first pseudo parent-class (i.e. the class of untransformed examples including all ten digits). Color codes denote the ground-truths for the digits. From epoch 1 to epoch 400 of the unsupervised (but pseudo supervised) training, clusters become well-separated and simultaneously the clustering accuracy increases. As clearly observed from this figure, using the pseudo supervision, the neural network also reveals some hidden patterns useful to distinguish the real digit identities and ultimately learns to categorize

each one of them. It is also worth noting that a high level of clustering accuracy is achieved relatively quickly (after only 50 epochs) as seen both in the t-SNE and test accuracy plots.

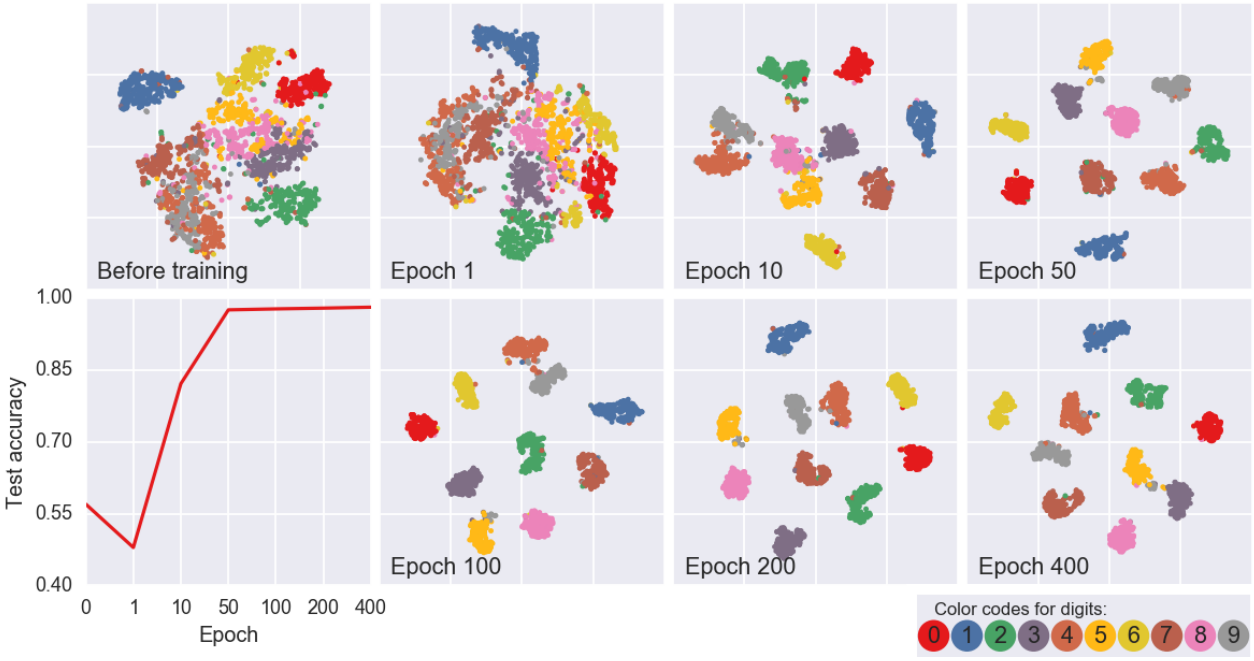

Figure 4.3: t-SNE visualization of the latent space $\boldsymbol{F}$ throughout the training for a randomly selected 2000 untransformed test examples from MNIST. Color codes denote the ground-truths for the digits. Note the separation of clusters from epoch 1 to epoch 400 of the unsupervised (but pseudo supervised) training. For reference, clustering accuracy for the entire test set is also provided. This figure is best viewed in color.

Tables 4.3 and 4.4 summarize quantitative unsupervised clustering performances observed on three datasets respectively in terms of accuracy (ACC) and normalized mutual information (NMI). Results of a broad range of recent existing solutions are also presented for comparison. These solutions are grouped according to their approaches to unsupervised clustering. Following the very recent developments in deep generative models, VaDE [25] and GMVAE [16] employ variational autoencoders while CatGAN [57], AAE [40] and IMSAT [24]

adopt adversarial training. DEC [69] simultaneously learns feature representations and cluster assignments using DNNs. On the other hand, JULE [72] combines agglomerative clustering with CNNs. Also, the performance of a conventional approach, applying k-means on the autoencoder representation, is provided to establish a baseline for unsupervised clustering performances. Our approach statistically significantly outperforms all the contemporary methods that reported unsupervised clustering performance on MNIST except IMSAT [24] displaying very competitive performance with our approach, i.e. 98.32%($\pm$0.08) vs. 98.40%($\pm$0.40). However, results obtained on the SVHN dataset show that our approach statistically significantly outperforms IMSAT on this realistic dataset and defines the current state-of-the-art for unsupervised clustering. Besides, the USPS dataset provides another basis of comparison between our approach and JULE - one of the models reporting highest unsupervised clustering accuracies on MNIST in the literature.

4.3.3 Representation Properties

Recall that, for the 6-layer CNN model employed in the experiments, $\boldsymbol{F} = \boldsymbol{Y}^{(L-2)}$ corresponds to the output of the fully-connected layer of 2048 ReLU nodes, $\boldsymbol{Z} = \boldsymbol{F}\boldsymbol{W}^{L-1}+\boldsymbol{b}^{L-1}$ is the input of the augmented softmax layer of 160 nodes, i.e. $n = n_p k_s$, where 8 pseudo parent-classes are represented by 20 softmax duplicates each.

Figure 4.4 provides the average value for each dimension of $\boldsymbol{F}$, $\boldsymbol{Z}$ and softmax($\boldsymbol{Z}$) observed with respect to untransformed test set examples and the norm of the associated weights. Note that the representation on $\boldsymbol{F}$ is not distributed to the entire space but the weights associated to these unused dimensions do not decay. On the other hand, due to the pseudo supervision task, the output of the augmented softmax layer i.e. softmax($\boldsymbol{Z}$), becomes a one-hot encoded representation of which 140 dimensions, i.e. $(n_p - 1)k_s$, are inactive for the untransformed examples; however, the representation at its input is distributed to all dimensions. Figure 4.4 also summarizes how the dimension size of $\boldsymbol{F}$, i.e. the number of

Table 4.3: Quantitative unsupervised clustering performance (ACC) on MNIST, USPS and SVHN datasets. Results of a broad range of recent existing solutions are also presented for comparison. The last row demonstrates the benchmark scores of the proposed framework in this chapter. 95% confidence interval is also provided to show the statistical significance of the obtained results.

	k	MNIST-*test*	USPS-*full*†	SVHN-*test*
VaDE [25]	10	94.06%	-	-
GMVAE [16]	10	82.31%($\pm$3.75)	-	-
GMVAE [16]	16	87.82%($\pm$5.33)	-	-
GMVAE [16]	30	92.77%($\pm$1.60)	-	-
CatGAN [57]	20	90.30%	-	-
AAE [40]	16	90.45%($\pm$2.05)	-	-
AAE [40]	30	95.90%($\pm$1.13)	-	-
IMSAT [24]	10	98.40%($\pm$0.40)	-	57.30%($\pm$3.90)
k-means [69]	10	53.49%	-	-
AE+k-means [69]	10	81.84%	-	-
DEC [69]	10	84.30%	-	11.9%($\pm$0.40)††
JULE [72]	10	96.10%	95.00%	-
Our approach	10	98.32%($\pm$0.08)	96.51%($\pm$0.26)	76.80%($\pm$1.30)

† Only for USPS dataset, following JULE [72], we reported unsupervised clustering performance over the full dataset for a fair comparison.
†† Excerpted from [24].

ReLU nodes in the fully-connected layer, affects the clustering performance. Decreasing the number of dimensions of $\boldsymbol{F}$ up to a point, i.e. ≈ 1024, does not significantly affect the clustering accuracy. However, further decrease beyond this point dramatically reduces the performance.

For comparison, Figure 4.5 presents t-SNE visualizations of these latent representations observed with respect to a randomly selected 2000 untransformed test examples without stratification from MNIST, where the portion of the largest class is 11%. One can clearly see that clusters are not well-separated on one-hot encoded softmax($\boldsymbol{Z}$); however, separations of the clusters are quite similar and clear on the representation spaces $\boldsymbol{F}$ and $\boldsymbol{Z}$. Hence, one can

Table 4.4: Quantitative unsupervised clustering performance (NMI) on MNIST, USPS and SVHN datasets. The last row demonstrates the benchmark scores of the proposed framework in this chapter.

	k	MNIST-*test*	USPS-*full*†	SVHN-*test*
JULE [72]	10	91.50%	91.30%	-
Our approach	10	95.64%($\pm$0.15)	92.76%($\pm$0.30)	68.90%($\pm$0.43)

† Only for USPS dataset, following JULE [72], we reported unsupervised clustering performance over the full dataset for a fair comparison.

also obtain similar clustering accuracy on the test set of 10000 examples, i.e. $= 98.16\%\pm(0.14)$, by applying k-means on the representation space $\boldsymbol{Z}$.

4.3.4 Graph Interpretation of the Latent Information Propagation through GAR

Recall that GAR terms have been originally proposed to propagate the available labels towards the unlabeled examples in a semi-supervised setting and, in the previous chapter, we have shown that these terms can also be adopted to propagate the hidden information that is introduced by a coarse level of supervision and which is useful to discover a deeper level of latent annotations. In the fully unsupervised setting considered in this chapter, as no real supervision is available, hidden information useful to discover unknown clusters is now captured through the help of domain specific transformations and propagated by GAR terms as well.

Figure 4.6 visualizes the realization of this propagation using the real predictions obtained on MNIST. Colored circles denote the ground-truths for the vertices, i.e. examples, and gray lines denote the edges, i.e. non-zero weighted connections between the examples representing their similarity. Note that, for vertices in graph $\mathcal{G}_y$, there are two different colors indicating true pseudo parent-class labels assigned per the applied transformation (for simplicity, out of 8, only the examples of the first two pseudo parent-classes are used for this illustration), albeit ten different colors indicating the real digit identity for vertices in graph

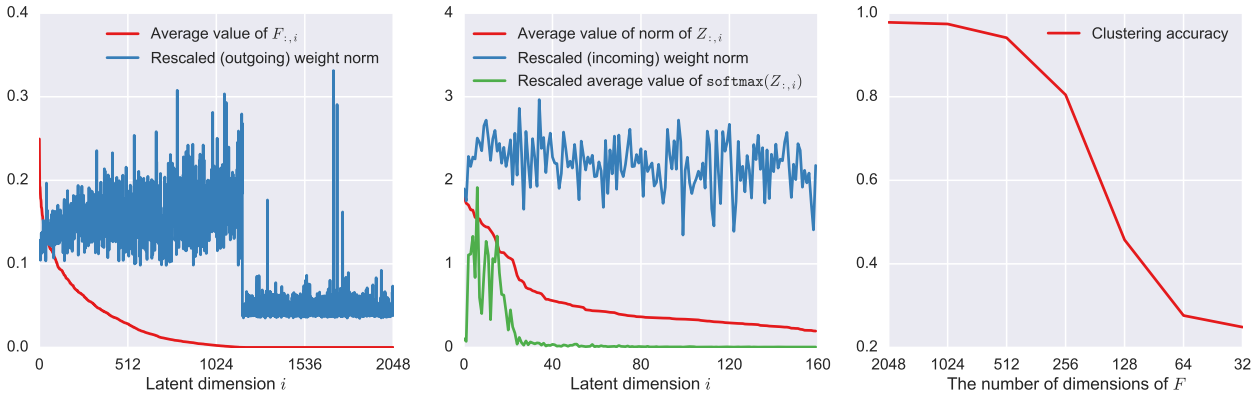

Figure 4.4: The average value for each dimension of $\boldsymbol{F}$, $\boldsymbol{Z}$ and softmax($\boldsymbol{Z}$) observed with respect to untransformed test set examples and the norm of the associated weights. Note that the representation on $\boldsymbol{F}$ is not distributed to the entire space but the weights associated to these unused dimensions do not decay. On the other hand, due to the pseudo supervision task, the output of the augmented softmax layer i.e. softmax($\boldsymbol{Z}$), becomes a one-hot encoded representation of which 140 dimensions are inactive for the untransformed examples; however, the representation at its input is distributed to all dimensions. The last plot shows how the dimension size of $\boldsymbol{F}$ affects the clustering performance. This figure is best viewed in color.

$\mathcal{G}_{\mathcal{M}}$. Recall that edges of these two graphs, $\mathcal{E}_{\mathcal{Y}}$ and $\mathcal{E}$, are respectively inferred by matrices $\boldsymbol{Y}\boldsymbol{Y}^T$ and $\boldsymbol{B}\boldsymbol{B}^T$ where $\boldsymbol{B} = \max(0, \boldsymbol{Z})$ and that $\mathcal{G}_{\mathcal{M}}$ is the spanning subgraph of $\mathcal{G}_{\mathcal{Y}}$. That is, $\mathcal{G}_{\mathcal{M}} = (\mathcal{M}, \mathcal{E})$ shares the same vertices $\mathcal{M}$ with graph $\mathcal{G}_{\mathcal{Y}} = (\mathcal{M}, \mathcal{E}_{\mathcal{Y}})$, which is constructed per the pseudo supervision; however, $\mathcal{E}$ is a subset of $\mathcal{E}_{\mathcal{Y}}$ as some of the edges in graph $\mathcal{G}_{\mathcal{Y}}$, such as those between the examples of digit 0 and 1, are eliminated in graph $\mathcal{G}_{\mathcal{M}}$ due to GAR regularization terms. As training continues, pseudo supervision eliminates the edges between the examples of different pseudo parent-classes and turns graph $\mathcal{G}_{\mathcal{Y}}$ into a disconnected graph of $n_p = 8$ disjoint subgraphs (only two of them are illustrated). Simultaneously, GAR terms eliminate the edges between the examples of the same parent-class in graph $\mathcal{G}_{\mathcal{M}}$ to discover previously unknown clusters. Ultimately, $\mathcal{G}_{\mathcal{M}}$ becomes disconnected graphs of δ disjoint subgraphs where $n_p \leq \delta \leq n_p k_s$ and each disjoint subgraph corresponds to a cluster.

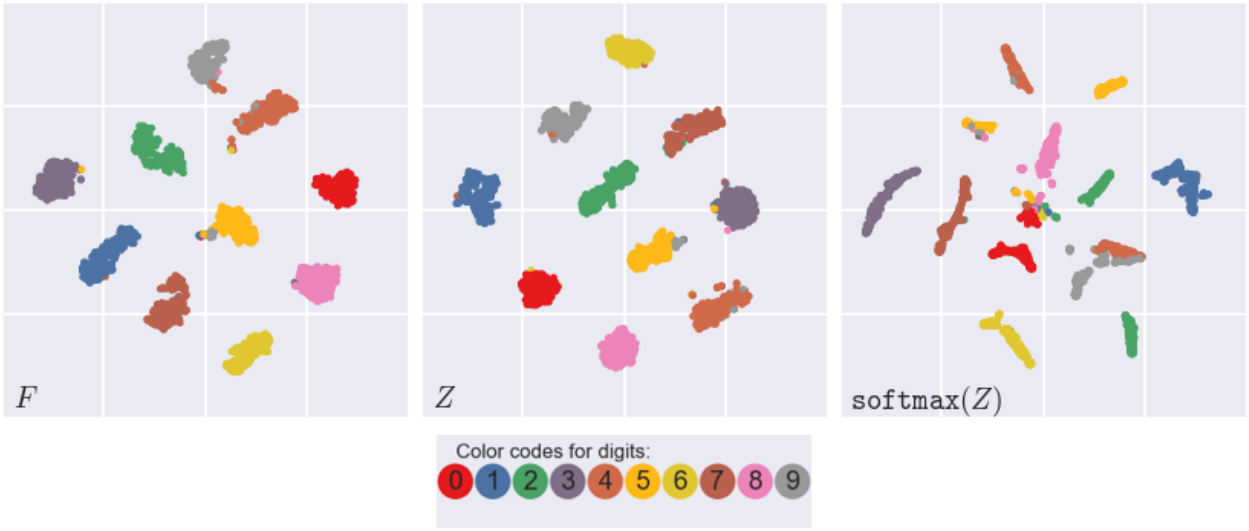

Figure 4.5: Comparison of t-SNE visualizations of the latent spaces $\boldsymbol{F}$, $\boldsymbol{Z}$ and softmax($\boldsymbol{Z}$) for 2000 test examples from MNIST. Color codes denote the ground-truths for the examples. Color codes denote the ground-truths for the digits. Clusters are not well-separated on one-hot encoded softmax($\boldsymbol{Z}$); however, separations of the clusters are quite similar and clear on the representation spaces $\boldsymbol{F}$ and $\boldsymbol{Z}$. This figure is best viewed in color.

4.3.5 The Impact of the Number of Clusters k

For the quantitative clustering results, we set the number of clusters for the k-means to the number of classes assuming a prior knowledge, i.e. $k = 10$. To demonstrate the representation power of the proposed approach as an unsupervised clustering model, on MNIST, we deliberately choose different k values for the k-means clustering applied on the representation space $\boldsymbol{F}$. For two different k settings i.e. 7 and 20, Figure 4.7 illustrates a few examples of each cluster. One can see that when k is smaller than the actual number of classes, digits with similar appearances are grouped together, such as digits 4 and 9, 5 and 8, 0 and 6. When k is set to a bigger value than the number of classes, some digits are divided into subclasses based on visually identifiable image properties such as digit tilt, roundness, etc. Note the differences between upright and oblique digit 1 as shown in clusters 2 and 20,

between two styles of digit 6 as shown in clusters 18 and 19, and between two styles of digit 2 as shown in clusters 7 and 12.

4.3.6 The Impact of Transformations

As the revealed unknown clusters are directly related with the captured latent information through pseudo parent-classes, choosing the right set of transformations for the clustering task of concern is crucial for the performance. Figure 4.8 presents t-SNE visualizations of the representation spaces observed when different sets of transformations are adopted.

The first row of Figure 4.8 illustrates the clustering results when one of four different transformation types, i.e. scaling, shearing, translation and random permutation of the pixels, is applied variably to generate 8 pseudo parent-classes. One can observe some level of grouping with scaling and shearing-based transformations; however, the clusters defined by these groupings do not represent real digit identities (as shown by the colored dots) and may indicate other features of images. On the other hand, translating the images or randomly permuting the pixel positions do not provide any useful knowledge to discover any well-defined clustering.

The second row of Figure 4.8 presents the results obtained when rotation-based transformations listed in (4.8) are adopted. One can easily observe that only two or four pseudo parent-classes generated using rotation-based transformations are sufficient to obtain decent clustering representing the real digit identities. Considering that, for MNIST, the clustering accuracy obtained using all 8 transformations in (4.8) is 98.32%($\pm$0.08), we have achieved 97.80%($\pm$0.18) accuracy using $\mathcal{S}_{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}$, 72.52%($\pm$6.20) accuracy using $\mathcal{S}_{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_2\}$ and 96.84%($\pm$0.29) accuracy using $\mathcal{S}_{\mathcal{T}} = \{\mathcal{T}_1, \mathcal{T}_3\}$. Recalling that $\mathcal{T}_2$ and $\mathcal{T}_3$ respectively correspond to rotating the images by 90^o and 180^o, one can say that comparing the untransformed images with their 180^o rotated versions is more effective in terms of

capturing the latent information that is useful to distinguish the real digit identities. In fact, $\mathcal{T}_3$ alone is sufficient to achieve state-of-the-art clustering accuracy on MNIST. Adding more rotation-based transformations to $\mathcal{S}_{\mathcal{T}}$ further improves the clustering performance. It's worth noting that max pooling layers of the CNN model used in the experiments provide basic invariance to rotations and translations in images. This might be the reason of not being able to obtain any well-defined clustering when using translation-based transformations as shown in the first row of Figure 4.8. To prevent any similar effects when using rotation-based transformations and also the need for interpolation, we applied the rotations in multiples of 90^o. To summarize, the type of the transformation generating the pseudo parent-classes is more important than their number and different transformations can reveal different clustering patterns. Therefore, finding the right transformation type for the clustering task of concern is crucial for the proposed approach in this chapter and it remains an important research question how to identify the kind of transformation most optimized for the clustering task at hand.

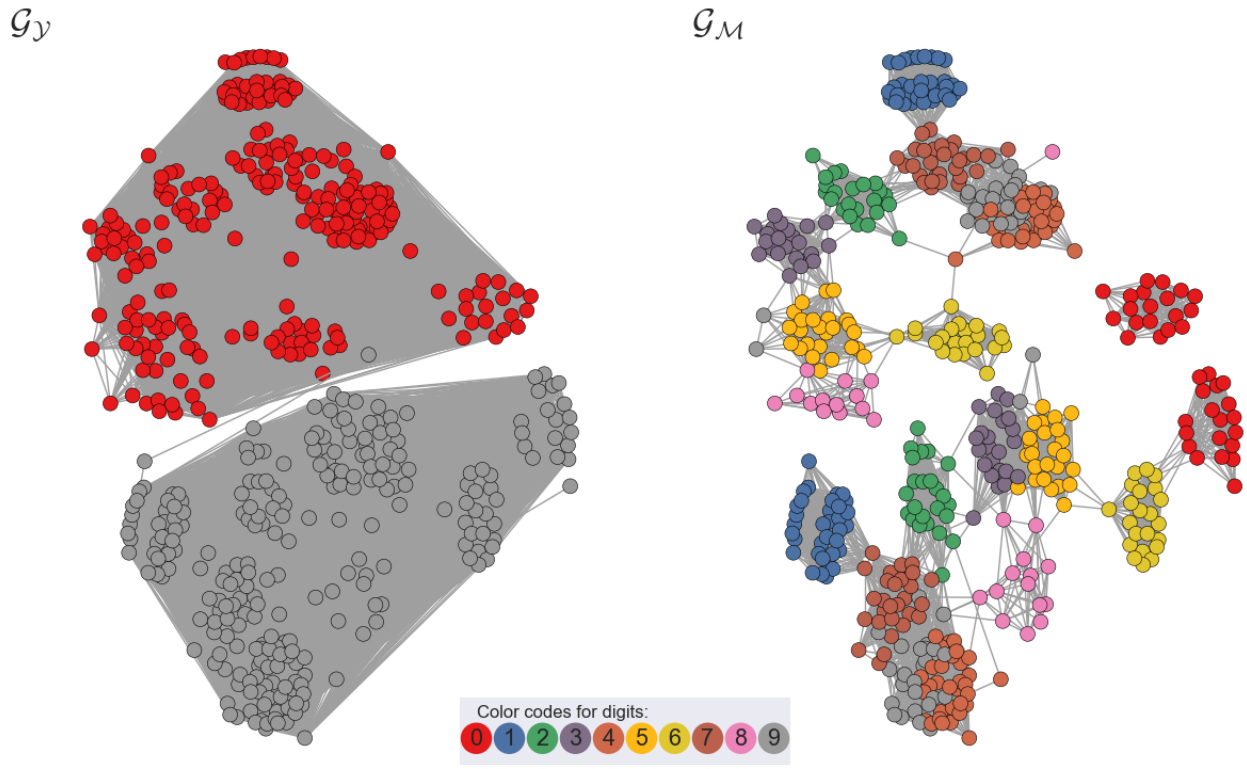

Figure 4.6: Visualizations of the graph $\mathcal{G}_{\mathcal{Y}}$ and its spanning subgraph $\mathcal{G}_{\mathcal{M}}$ for randomly chosen 250 test examples from MNIST. Colored circles denote the ground-truths for the vertices, i.e. examples, and gray lines denote the edges, i.e. non-zero weighted connections between the examples representing their similarity. Note that, for vertices in graph $\mathcal{G}_{\mathcal{Y}}$, there are two different colors indicating true pseudo parent-class labels assigned according to the applied transformation (for simplicity, out of 8, only the examples of first two pseudo parent-classes are used for this illustration), albeit ten different colors indicating the real digit identity for vertices in graph $\mathcal{G}_{\mathcal{M}}$. As training continues, pseudo supervision eliminates the edges between the examples of different pseudo parent-classes and turns graph $\mathcal{G}_{\mathcal{Y}}$ into a disconnected graph of $n_p = 8$ disjoint subgraphs (only two of them are illustrated). Simultaneously, GAR terms eliminate the edges between the examples of the same parent-class in graph $\mathcal{G}_{\mathcal{M}}$ to discover previously unknown clusters. Ultimately, $\mathcal{G}_{\mathcal{M}}$ becomes disconnected graphs of δ disjoint subgraphs where $n_p \leq \delta \leq n_p k_s$ and each disjoint subgraph corresponds to a cluster. This figure is best viewed in color.

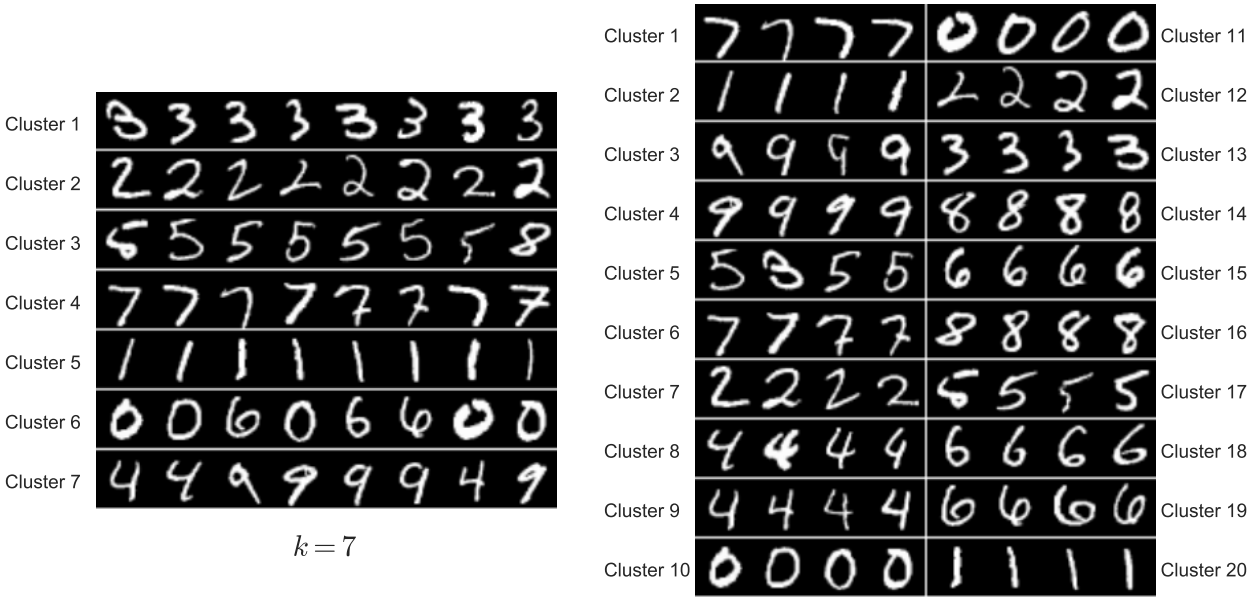

Figure 4.7: Illustration of a few examples of each cluster for two different k settings. When k is smaller than the actual number of classes, digits with similar appearances are grouped together, such as digits 4 and 9, 5 and 8, 0 and 6. When k is set to a bigger value than the number of classes, some digits are divided into subclasses based on visually identifiable image properties such as digit tilt, roundness, etc. Note the differences between upright and oblique digit 1 as shown in clusters 2 and 20, between two styles of digit 6 as shown in clusters 18 and 19, and between two styles of digit 2 as shown in clusters 7 and 12.

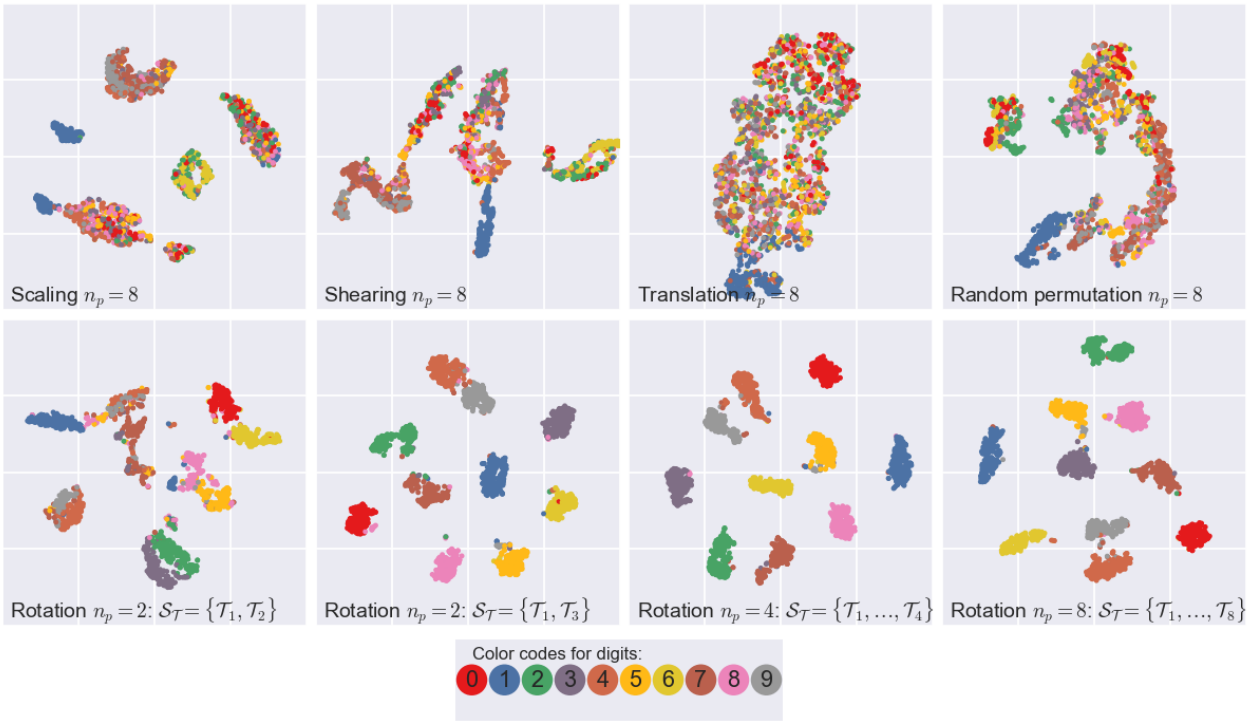

Figure 4.8: t-SNE visualizations of the representation spaces observed when different sets of transformations are adopted. The first row illustrates the clustering results when one of four different transformation types, i.e. scaling, shearing, translation and random permutation of the pixels, is applied variably to generate 8 pseudo parent-classes. The second row presents the results obtained when rotation-based transformations listed in (4.8) are adopted. To summarize, the type of the transformation generating the pseudo parent-classes is more important than their number and different transformations can reveal different clustering patterns. Therefore, finding the right transformation type for the clustering task of concern is crucial for the proposed approach in this chapter. Color codes denote the ground-truths for the examples. This figure is best viewed in color.

CHAPTER 5: CONCLUSIONS

It is now a universally accepted fact that machine learning, especially in recent years, has achieved tremendous success in practical industrial applications, specifically on supervised learning problems with large amounts of properly labeled data. Requiring such large amounts of supervised data to obtain good accuracy, these algorithms are difficult to adopt to domains where labeling is a challenging task that needs expert knowledge. Therefore, in recent years, machine learning research has had a focus on algorithms that are able to exploit a large set of unlabeled examples to reduce the amount of labeled data required for existing models to perform well. In this dissertation, we presented novel, scalable and efficient learning frameworks for semi-supervised and unsupervised settings based on two ideas: Graph-based Activity Regularization (GAR) and Auto-clustering Output Layer (ACOL).

The second chapter introduced Graph-based Activity Regularization (GAR) technique for semi-supervised learning problems. Specifically, in Chapter 2, we proposed a novel graph-based framework considering adaptive adjacency of the examples, $\boldsymbol{M}$, which is inferred using the predictions of a neural network model. When well-constrained, the adaptive adjacency approach contributes to improved accuracy results and automatically yields that the predictions of the network become the optimal embedding of $\boldsymbol{M}$ without requiring any additional step. We satisfied these constraints by defining a regularization over the adjacency of the output nodes, $\boldsymbol{N}$, which is also inferred using the predictions of the network. Such regularization helped us devise an efficient and scalable framework that is natural for the classification framework on neural networks as it requires no additional task calculating the reconstruction error or implementing zero-sum game mechanism unlike competing state-of-

the-art autoencoder or adversarial network based approaches. Through this low-cost and easy-to-train framework, we obtained comparable performance with state-of-the-art generative approaches for semi-supervised learning.

In the third chapter, we introduced Auto-clustering Output Layer (ACOL), a novel modification to the output layer of a neural network, to automatically identify the latent annotations via partial supervision of coarse class labels. We use GAR terms in training the model to search for sub-classes under parent-classes without supervision. The proposed learning framework can be used in many domains such as text categorization, protein function prediction, image classification as well as in exploratory scientific studies such as medical and genomics research. Our major contributions in Chapter 3 are four-fold: First, we explored a different type of semi-supervised setting for neural networks. That is, every observation in a dataset has a corresponding ground-truth label; however, this label is more general than the main categorization interest. Hence, the aim of this particular semi-supervised setting is to explore the more definite latent annotations when this general supervision is provided as parent-class labels. Second, we propose a simple yet efficient output layer modification, ACOL, which enables simultaneous supervised classification and unsupervised clustering on neural networks. ACOL introduces duplicated softmax nodes for each one of the parent-classes. Then, we adopted GAR terms for the unsupervised portion of the objective function and showed that these terms efficiently guide the optimization in a way that each softmax duplicate is specialized during the training to represent a proper latent annotation. Most interestingly, we demonstrate that the neural network can learn from existing differences between different parent-class labels and translate that knowledge to better identify sub-classes within each parent-class. Finally the proposed approach was validated on three popular image benchmark datasets, MNIST, SVHN and CIFAR-100, through t-SNE visualizations and unsupervised clustering accuracy metrics compared to

well-accepted approaches implemented for the particular semi-supervised setting discussed in Chapter 3.

In the fourth chapter, we introduced a novel unsupervised clustering approach building upon ACOL and GAR. To discover unknown clusters in a fully unsupervised setup, we substitute the real, yet unavailable, partial supervision with a pseudo one. More specifically, we randomly assign pseudo parent-class labels each of which is associated with a different domain specific transformation. Each observation is modified by applying the transformation corresponding to the assigned pseudo label. Generated observation-label pairs are used to train an ACOL network that introduces multiple softmax nodes for each pseudo parent-class. Due to the unsupervised regularization based on GAR terms, each softmax duplicate under a parent-class is specialized as the latent information captured with the help of domain specific transformations is propagated throughout the training. Ultimately we obtain a k-means friendly latent representation. Furthermore, we demonstrate that the neural network can learn by comparing differently transformed examples and translate that knowledge to reveal unknown clusters. The proposed approach is validated on three image benchmark datasets, MNIST, SVHN and USPS, through t-SNE visualizations and unsupervised clustering accuracy exceeds those reported by well-accepted approaches in the literature. Future work will extend this approach to other domains such as sequential data. We will also explore how to optimize domain specific transformations based on known or otherwise identifiable characteristics of the dataset being considered for clustering.

In addition to competitive and state-of-the-art performance, compared to other traditional graph-based approaches, the proposed frameworks in this dissertation are fully scalable to very large datasets. Besides, unlike other state-of-the-art approaches based on generative models recently proposed in the literature, i.e. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), all three approaches provide easy-to-train frameworks that are fit well with the operation of neural networks as i) GAR terms can readily

be added to the loss function as simply as adding standard $L1$, $L2$ regularizations and ii) ACOL brings no additional overhead to the training. Ultimately, when their performance is coupled with their simplicity and ease-of-implementation, the proposed approaches are strong candidates to replace current state-of-the-art techniques in the literature for semi-supervised and unsupervised settings.

LIST OF REFERENCES

[1] Aggarwal, C. C. and Reddy, C. K., editors (2014). *Data Clustering: Algorithms and Applications*. CRC Press.

[2] Ahmed, A., Yu, K., Xu, W., Gong, Y., and Xing, E. P. (2008). Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*, pages 69–82.

[3] Arel, I., Rose, D. C., and Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18.

[4] Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035.

[5] Bair, E. (2013). Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361.

[6] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

[7] Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

[8] Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. (2013). Advances in optimizing recurrent networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8624–8628.

[9] Bishop, C. M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer.

[10] Blei, D. M. and McAuliffe, J. D. (2007). Supervised topic models. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 121–128.

[11] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[12] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[13] Burmeister, J. and Wiles, J. (1995). The challenge of go as a domain for ai research: a comparison between go and chess. In *Intelligent Information Systems, 1995. ANZIIS-95. Proceedings of the Third Australian and New Zealand Conference on*, pages 181–186. IEEE.

[14] Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

[15] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[16] Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. (2016). Deep unsupervised clustering with gaussian mixture variational autoencoders. *CoRR*, abs/1611.02648.

[17] Dosovitskiy, A., Fischer, P., Springenberg, J. T., Riedmiller, M. A., and Brox, T. (2016). Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(9):1734–1747.

[18] Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. C. (2016). Adversarially learned inference. *CoRR*, abs/1606.00704.

[19] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[20] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

[21] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 1735–1742.

[22] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[23] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[24] Hu, W., Miyato, T., Tokui, S., Matsumoto, E., and Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1558–1567.

[25] Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2017). Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1965–1972.

[26] Jr., C. N. S. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72.

[27] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[28] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*.

[29] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[30] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

[31] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[32] Lacoste-Julien, S., Sha, F., and Jordan, M. I. (2008). Disclda: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 897–904.

[33] Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. *CoRR*, abs/1610.02242.

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

[35] LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database of handwritten digits.

[36] LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–104. IEEE.

[37] Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1445–1453.

[38] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[39] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[40] Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. J. (2015). Adversarial autoencoders. *CoRR*, abs/1511.05644.

[41] Miyato, T., Maeda, S., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *CoRR*, abs/1704.03976.

[42] Miyato, T., Maeda, S., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional smoothing by virtual adversarial examples. *CoRR*, abs/1507.00677.

[43] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814.

[44] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.

[45] Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

[46] Patterson, K., Nestor, P. J., and Rogers, T. T. (2007). Where do you know what you know? the representation of semantic knowledge in the human brain. *Nature Reviews Neuroscience*, 8(12):976–987.

[47] Premachandran, V. and Yuille, A. L. (2016). Unsupervised learning using generative adversarial training and clustering.

[48] Ramage, D., Hall, D. L. W., Nallapati, R., and Manning, C. D. (2009a). Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 248–256.

[49] Ramage, D., Heymann, P., Manning, C. D., and Garcia-Molina, H. (2009b). Clustering the tagged web. In *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, pages 54–63.

[50] Ranzato, M. and Szummer, M. (2008). Semi-supervised learning of compact document representations with deep networks. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 792–799.

[51] Rasiwasia, N. and Vasconcelos, N. (2013). Latent dirichlet allocation models for image classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2665–2679.

[52] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3546–3554.

[53] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1278–1286.

[54] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[55] Salakhutdinov, R. and Hinton, G. E. (2009). Deep boltzmann machines. In *International conference on artificial intelligence and statistics*, pages 448–455.

[56] Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234.

[57] Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *CoRR*, abs/1511.06390.

[58] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[59] Suddarth, S. C. and Kergosien, Y. L. (1990). Rule-injection hints as a means of improving network performance and learning time. In *Neural Networks, EURASIP Workshop 1990, Sesimbra, Portugal, February 15-17, 1990, Proceedings*, pages 120–129.

[60] Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

[61] Valpola, H. (2014). From neural PCA to deep unsupervised learning. *CoRR*, abs/1411.7783.

[62] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408.

[63] von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416.

[64] Wang, C., Blei, D. M., and Li, F. (2009). Simultaneous image classification and annotation. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 1903–1910.

[65] Wang, M., Fu, W., Hao, S., Liu, H., and Wu, X. (2017). Learning on big graph: Label inference and regularization with anchor hierarchy. *IEEE Trans. Knowl. Data Eng.*, 29(5):1101–1114.

[66] Wattenberg, M., Viégas, F., and Johnson, I. (2016). How to use t-sne effectively. *Distill*.

[67] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 639–655.

[68] Xiao, Z., Dellandrea, E., Dou, W., and Chen, L. (2007). Automatic hierarchical classification of emotional speech. In *Multimedia Workshops, 2007. ISMW'07. Ninth IEEE International Symposium on*, pages 291–296. IEEE.

[69] Xie, J., Girshick, R. B., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 478–487.

[70] Xu, W., Liu, X., and Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 267–273.

[71] Yan, D., Huang, L., and Jordan, M. I. (2009). Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 907–916.

[72] Yang, J., Parikh, D., and Batra, D. (2016a). Joint unsupervised learning of deep representations and image clusters. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5147–5156.

[73] Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016b). Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 40–48.

[74] Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919.