Correlational Neural Networks for Common Representation Learning

A THESIS

submitted by

SARATH CHANDAR A P

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

July 2015

THESIS CERTIFICATE

This is to certify that the thesis titled **Correlational Neural Networks for Common Representation Learning**, submitted by **Sarath Chandar A P**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Science**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Balaraman Ravindran
Research Guide
Associate Professor
Dept. of CSE
IIT-Madras, 600 036

Place: Chennai

Date: 06 July 2015

ACKNOWLEDGEMENTS

I am extremely lucky to be surrounded by an amazing set of people and I am grateful to all of them. Firstly, I would like to thank my adviser Balaraman Ravindran who gave me the freedom to explore the field of Machine Learning. It would have been impossible for me to delve into a wide range of fields like Reinforcement Learning, Natural Language Processing, Bayesian Non Parametrics, Deep Learning, Social Network Analysis, without his support and guidance. His versatility and enthusiasm to work are always my sources of inspiration.

Next I would like to extend my gratitude to Mitesh Khapra. I was very fortunate to have Mitesh as my mentor during my internship at IBM Research Lab. It was his encouragement that led me to explore Deep Learning. I have learnt a lot from him, particularly the practical aspects of NLP.

I would also like to thank Raghavendra Rao for his excellent course on theory toolkit which helped me to develop a strong foundation for understanding theoretical aspects of Machine Learning. We had several discussions on theoretical Machine Learning problems. He also played a major role in developing central ideas in my Graph grammars work. I get inspired by his mathematical knowledge every time I leave his office after a healthy technical discussion.

Hugo Larochelle deserves a big thanks though I met him only towards the end of my course. His collaboration strengthened my Deep Learning knowledge further and he is definitely an excellent researcher to work with. I spent my 2014 summer with his research group and worked on Neural Machine Translation and document network modelling.

I am also happy that I got opportunity to collaborate with several excellent students during my stay at IIT. Exploring several fields would not have been possible without the excellent set of collaborators. Prasanna worked with me in Bayesian Reinforcement Learning and Jana worked with me in several problems in Deep Learning. Jana's questions have always been the food for my thoughts and it has shaped a lot of my research

ideas. He has a strong knowledge about Correlational Neural Networks and helped me develop several new insights. I would also like to thank Gowthaman, with whom I explored spectral learning. Thanks to Stanislaus Lauly for the fruitful collaboration which resulted in a NIPS 2014 paper.

Apart from these people, I am also grateful to several of my friends at IIT Madras who made my entire stay memorable. Special thanks to Deepak for being my source of inspiration and for supporting me morally whenever I faced rejections. My life at IITM would have been not-so-interesting without Sabari. Sabari was an excellent team mate in many of the courses I undertook and he backed me whenever I felt low. A simple thanks would not suffice to all his help. I would also like to thank Saket, Avijit, Arpita, Prasanna, Praveen, Jana for making my lab experience awesome. I will definitely miss the late night discussions and coffee at night canteens. A special thanks to Arpita for providing me with her computing machines whenever I needed more computing power to run experiments (I always needed more machines!). Several wonderful people from outside my lab also have made my life at IITM easy. Thanks to Anoop, Sahitya, Vaishnavh, Subhashree and Aditya for all the wonderful technical and non-technical discussions we had. Subhashree deserves a special thanks. She was there whenever I needed any help. Thanks to Vaishnavh and Akhilesh for their wonderful questions in classroom which was a major reason for my several hours of preparations for every single class that I taught in Machine Learning and Graphical Models course. I take this as an opportunity to thank again Ravi for letting me lecture in his courses which helped me harness my teaching skills to a great extent.

Personally I would like to thank Priyesh, Nandhini, Abhishek and Nivas for all their moral support without which I would not have completed my course. Thanks to Susan for introducing me to research. Thanks to Sibi for the wonderful stay during the course. If there was one reason I wanted to finish up the work early everyday, it was Sibi. He is my saviour to break the mechanical research cycles whenever I got stuck inside one without life. Thanks to my sister Parani for always believing me. At last but not least, thanks to my Mother Prema who is the main reason for where I am today. I can say with very high confidence that my each and every success is only due to her prayers and love. I wish I will make her proud and happy every single moment hereafter. I dedicate this thesis to my Father Parthipan who always wanted to see me as a Computer Scientist. I am very close to achieving your dream dad !

ABSTRACT

KEYWORDS: Representation Learning; Deep Learning; Transfer Learning; Neural Networks; Auto-encoders; Common Representations.

Classical Machine Learning (ML) algorithms learn to do a specific task, say classification by using the available training data and classify the test data which is unknown during the training. However, all these algorithms assume that the data is represented using a set of features, which are mostly hand-crafted by human experts. Learning useful features or representations from the raw data, also known as Representation Learning is one of the hardest problems in Machine Learning. Deep Learning is an emerging field in Machine Learning which solves the problem of representation learning using Deep Neural Networks (DNNs). These representations learned using Deep Neural Networks, when coupled with simple classification algorithms, perform significantly better than the complex state-of-the-art procedures in text, image and speech data.

Common Representation Learning (CRL), wherein different descriptions (or views) of the data are embedded in a common subspace, is receiving a lot of attention recently. Two popular paradigms in CRL world are Canonical Correlation Analysis (CCA) based approaches and Autoencoder (AE) based approaches. CCA based approaches learn a joint representation by maximizing correlation of the views when projected on the common subspace. AE based methods learn a common representation by minimizing the error of reconstructing the two views. Each of these approaches has its own advantages and disadvantages. For example, though CCA based approaches outperform AE based approaches for the task of transfer learning, they are not as scalable as the latter.

In this thesis, we propose Correlational Neural Networks (CorrNet), a class of neural networks that can learn common representation for two different views. CorrNet is an AE based approach, that explicitly maximizes correlation among the views when projected on the common subspace. The model can be efficiently trained with batch gradient descent and is thus scalable to big data. Apart from learning common representation for different views, the model also learns to predict the features in one view

given the features in the other view, which has several interesting applications. CorrNet, unlike CCA, can be easily extended to more than two views. Experiments show that CorrNet outperforms CCA in learning common representation. Thus scalability is gained without compromising performance. The CorrNets come with another advantage, that is, it can make use of single view data when there are less parallel data and this is not possible in CCA.

CorrNet has been successfully applied in Natural Language Processing (NLP) to several cross lingual tasks where a learner has to learn from one language and perform in a different language. In the task of Cross Language Document Classification (CLDC), CorrNet based bilingual word representation learning algorithm performs significantly better than the current state of the art procedures and a strong Machine Translation baseline. The model is also applied in Transliteration Mining. It has been tested with several language pairs and works well even if the languages differ in scripts, since the model itself is not language dependent.

CorrNet can be considered as a variant of auto-encoder to handle multi-view data. In this thesis, we also propose ways to make the CorrNet deep. One of the deep versions of the CorrNet performs much better than several strong baselines in the task of Cross Language Sentiment Analysis (CLSA). All the mentioned studies leads us to believe that CorrNet is a promising tool from common representation learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

"Can machines think?" Alan Turing, a British mathematician proposed to consider this question in 1950 (Turing, 1950). We humans can think and act based on our intelligence. Can a (non-living) machine think? Can a machine possess human level intelligence? Artificial Intelligence (AI) tries to answer this question by designing algorithms that can acquire knowledge and perform reasoning based on its acquired knowledge.

Machine Learning (ML) is a sub-field of AI which believes that intelligence can be achieved only by *learning*, which is how humans are believed to gain their knowledge. Machine Learning is widely applied to problems related to text, images, videos and speech. One simple application of ML is spam classification. In spam classification, the task is to classify the mails as spam or ham. The learning algorithm may face any of the following scenarios:

Given a set of sample spams and hams, the algorithm is supposed to learn the characteristics of spam and ham and classify the new mail as per these characteristics. This is called as *Supervised Learning* since the algorithm receives supervision in the form of training data, which contains sample mails and corresponding classes (spam or ham).

The learning algorithm might be forced to directly perform, without any supervision and get some feedback in the form of rewards for each decision it takes. It will eventually learn to classify by trying to maximize the rewards. This form of learning is called as *Reinforcement Learning*.

Given a set of mails without any labels, the algorithm might be asked to learn to characterize the mails and detect anomalies in the new mail (spam). This form of learning is called as *Unsupervised Learning*.

In the case of Supervised Learning, the algorithm learns a decision function from the given set of examples (training data). However, the algorithm is expected to perform well even when it sees previously unseen data (test data). This is known as generalization capability of the algorithm and better the generalization, better the performance.

In all the three approaches for learning, data plays a major role. The agent (algorithm) learns to perform a task by learning from the data. The data is usually represented in the form of a set of features. For example, in the case of e-mail classification problem, the presence or absence of a certain set of discriminative words can be considered as features. In an image classification problem, features could be the pixel information in the image.

Now we can formally define the problem of supervised learning. Given $\{(X_i, Y_i)_{i=1}^N\}$ where N is the number of training samples, $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}$, learn a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ which fits the data well such that given a new instance X, the learnt function can predict the corresponding Y as accurate as possible. This is called as *regression* problem and when Y is discrete, it is called as *classification* problem.

In any supervised problem, the features used for learning has a big impact on the learnability of the problem with respect to a particular learning algorithm. We will illustrate this fact with a simple toy example. Consider an XOR problem over two variables. The data points and the corresponding class labels are given in Table1.1.

x_1	x_2	class
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.1: XOR Problem

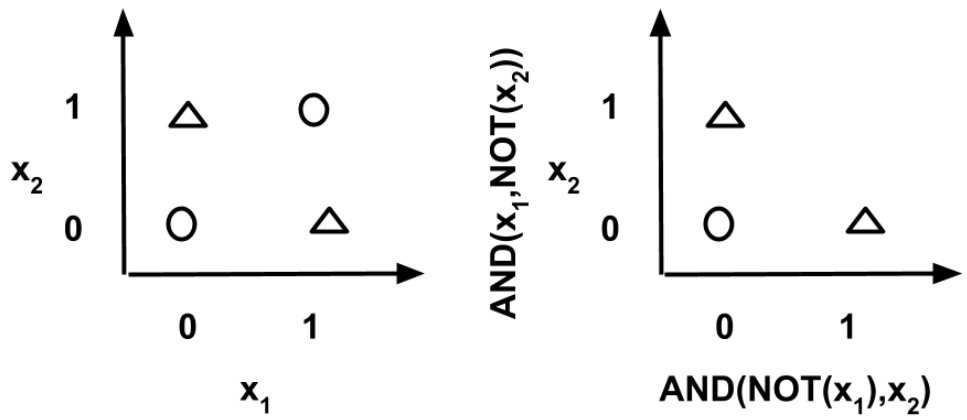

Figure 1.1: Two different projections of data points in XOR problem.

Let us consider the perceptron algorithm which can learn linear decision functions.

Let us consider the features to the algorithm to be x_1 and x_2. In this scenario, the data lies in this two dimensional space and it is visually represented in Figure 1.1 (left). By visualizing the data, it is easy to verify that a linear perceptron can never achieve more than 75% accuracy. With a linear decision function, it should definitely misclassify at least one point. However, if we project the data into some other representation, like the one illustrated in Figure 1.1 (right), then the problem can be easily solved by a linear perceptron. The same problem which was previously linearly inseparable became linearly separable! This is the power of good representation. A good representation makes learning relatively much easier.

1.1 Representation Learning

Good representations are *expressive* in nature. According to Bengio *et al.* (2013), there are three major characteristics that a good representation should possess:

- **distributed** : A distributed representation is the one where multiple features can be independently varied. The features are not mutually exclusive. Each concept is represented by multiple features and each feature is involved in representing many concepts.
- **invariant** : A good representation would be invariant to most local changes in the input. This results in a proper abstraction of the data which is good enough to generalize.
- **disentangling factors of variation** : A good representation should be able to disentangle the underlying factors of variation.

Consider the case of document representation. One simplest way of representing documents is to represent it as a binary bag of words (*bbow*) vector. *bbow* is a binary vector of size equal to the vocabulary. Each entry in the vector is 1 if the corresponding word in the vocabulary is present in the document or 0 otherwise. Clearly this is not a good representation since it is not distributed. Also, as the vocabulary size increases, the representation becomes too sparse and the curse of dimensionality is high. It is also clear that designing better representations manually, as we did for the XOR problem is practically infeasible in this scenario. The standard machine learning approach to this problem is to learn a better representation from the data itself.

Learning representations from the data is one of the most challenging problems in machine learning. It is challenging because we are interested in learning good representations while we do not know which representation is good for the application in hand. Earliest method which is successfully applied for learning representation is Principal Component Analysis (PCA) (Pearson, 1901) which learns a linear projection of the unlabelled data such that the variance in projected dimensions is maximum. Linear Discriminant Analysis (LDA) (Fisher, 1936) can be considered as a supervised counterpart for PCA. LDA learns linear projections of the data such that it separates the data points from different classes. While both these models are linear models, kernel versions of these models (Scholkopf *et al.*, 1998; Mika *et al.*, 1999) which find non-linear projections also exist.

Once we have a good representation learnt using a representation learner, we use the learnt representation to train a classifier for the task in hand. Neural Networks are a class of algorithms which are capable of learning representations and decision surfaces simultaneously. Consider a three layer neural network. The hidden layer can be considered to learn a projection of the data (representation learning) and the output layer to learn a decision function (classification learning).

Autoencoders and Restricted Boltzmann Machines (RBMs) are the most successful Neural Network based approaches for representation learning. A three layer linear autoencoder learns representation of the data which is exactly similar to the representation learnt by a PCA. However, using multiple layers in Autoencoders or RBMs was initially not succesful. With the invention of greedy layer-wise unsupervised pre-training (Hinton *et al.*, 2006), the deep neural networks are capable of learning richer representations which produce state of the art results in speech recognition (Hannun *et al.*, 2014), image recognition (Krizhevsky *et al.*, 2012), text processing (Zhang and LeCun, 2015) and several other fields.

1.2 Motivation

In any real life application, data comes with several constraints. The same data can be viewed from multiple views. For example, a movie clip consists of an audio view and a video view. A text document can be viewed from multiple languages. It is desirable to

make the best use of the available information in all the views. For example, viewing the same document in multiple languages benefits in word sense disambiguation and hence better performance (Mihalcea *et al.*, 2007). Traditional machine learning algorithms can just consider all the views as a single set of features. However, this might result in the loss of several inter-view and intra-view dependencies. Learning representations such that it captures all these inter and intra view dependencies in multiple modalities is necessary in such situations.

Another motivation to learn a common representation for multiple views is transfer learning. In transfer learning, we are interested in training an agent to perform in one view and expect it to generalize its knowledge to perform in another view. This has several potential applications. If we have a good transfer learning system, we can use all available resources for language processing in English to design better systems for language processing in other languages in which we do not have enough resources.

Any ideal common representation should have the following characteristics.

- If x and y form a parallel view, then they should be mapped to same vector or closest possible vector in the common representation space.
- Similar data points should be mapped closer in the common representation while dissimilar data points should be mapped far away. In other words, the common representation should preserve the semantics of the data.
- The representation of one view in this common space should be predictive of the features in the other view.

A good shared representation has a wide range of applications as listed below.

- Common representation helps in transfer learning across the views. If you consider these views as languages, then essentially we can train a classifier with the training data available in one language and transfer that knowledge to classify instances in another language. This is most preferable when you do not have enough training data in one language.
- In some applications, one view might be easy to get while the second view might be hard to get or costly to compute online. Unfortunately, the second view might be most representative of the classes. In such scenario, a common representation which is predictive of the second view can improve the performance of the classifier even though it has access to the first view only.
- Common representation captures the characteristics of both the views and hence projecting the data to the common representation to train a classifier solves multiview learning.

- Common representation for multimodal data solves multimodal learning. For example consider a query system for image search. When you have a common representation for images and text, we can query the system with either text or an image and the system will retrieve the images that lie close to this query in the shared space. Hence common representation helps in multimodal tasks as well.

1.3 Problem Statement

In the previous section, we discussed the characteristics, an ideal common representation should have. In this section we formally define the problem of Common Representation Learning (CRL). We define the task of learning common representation for two views and the definition can be easily extended to multiple views.

Consider some data $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^N$ which has two views: X and Y. Each data point $\mathbf{z}_i$ can be represented as a concatenation of these two views : $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i \in \mathbb{R}^{d_1}$ and $\mathbf{y}_i \in \mathbb{R}^{d_2}$. Common Representation Learning is defined as learning two functions, h_X and h_Y, such that $h_X(\mathbf{x}_i) \in \mathbb{R}^k$ and $h_Y(\mathbf{y}_i) \in \mathbb{R}^k$ are projections of $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively in a common subspace ($\mathbb{R}^k$) such that for a given pair $\mathbf{x}_i$, $\mathbf{y}_i$:

1. $h_X(\mathbf{x}_i)$ and $h_Y(\mathbf{y}_i)$ should be highly correlated.
2. It should be possible to reconstruct $\mathbf{y}_i$ from $\mathbf{x}_i$ (through $h_X(\mathbf{x}_i)$) and vice versa.

Canonical Correlation Analysis (CCA) satisfies only the first condition. Existing Autoencoder based approaches satisfy only the second condition. We need both these conditions to be satisfied for meaningful common representation. The model should also learn this common representation with as less multi-view data as possible since multi-view data is costly when compared to single view data. The model should also be able to use the available single view data.

1.4 Contributions of the thesis

In this thesis, we propose a novel algorithm for learning common representation based on neural networks. We call our learning algorithm as Correlational Neural Network (CorrNet). The proposed learning algorithm has several advantages as described below.

- The CorrNet can predict the other view based on one view. This predictive capability improves the quality of the representations learnt.
- The proposed learning algorithm is scalable with the available huge amount of data while most of the existing algorithms are not scalable. CorrNets work well with batch gradient descent as demonstrated in the experiments section.
- Another major advantage with CorrNets, which is not available in other methods is that this can be trained with single view data also along with parallel view data to improve the quality of reconstruction. This helps in better generalization.
- The CorrNets are very simple when compared to the existing shared representation learners and easy to implement.
- This model can be easily extended to more than 2 views.

In this thesis, we do an extensive analysis of CorrNets and compare its characteristics with other competing algorithms. Experiments on several tasks prove that CorrNets are better than other common representation learning algorithms. The thesis also demonstrates the application of CorrNet for several cross language tasks. We show that we can learn meaningful bilingual word representations without any word-aligned data, such that similar words in two different languages lie closer in the common subspace.

We apply CorrNet to transfer learning tasks like Cross Language Document Classification (CLDC) and Cross Language Sentiment Analysis (CLSA). Specifically, in Cross Language Document Classification CorrNet achieves around 14% improvement over the previous state of the art for English to German transfer task. We also apply CorrNet to a matching task : Transliteration Equivalence. In this thesis, we propose two different ways to extend CorrNets to Deep Correlational Networks and provide experimental evidences for their performance.

1.5 Organisation of the thesis

Rest of this thesis is organized as follows:

- In chapter 2, we will review the background in Neural Networks and Deep Learning which is essential to understand the rest of the thesis. We will also describe Canonical Correlation Analysis (CCA) which also does common representation learning.
- Chapter 3 introduces Correlational Neural Network, the major contribution of this thesis. This chapter has an extensive analysis of CorrNet which helps to

characterize the algorithm. Experiments on MNIST dataset show that CorrNet is superior to the existing algorithms for common representation learning in various aspects. We also discuss how to apply CorrNet for transliteration mining.

- Chapter 4 talks about the application of CorrNets for Cross Language Learning. We propose a way to learn common representation for words in two languages and the learnt word representations are used for Cross Language Document Classification (CLDC). Our model outperforms the previous state of the art and a strong machine translation baseline by a huge margin.

- In chapter 5, we motivate the need for Deep Correlational Networks and propose two ways to extend CorrNet to deep versions. We state experimental results for one variant of Deep CorrNet applied for Cross Language Sentiment Analysis (CLSA).

- Chapter 6 concludes the thesis by providing a summary of the work done and discussing possible future work.

CHAPTER 2

BACKGROUND

In this chapter, we will briefly describe the neural network and deep learning background necessary to understand Correlational Neural Networks. We will also describe Canonical Correlation Analysis (CCA) which is a de facto model for learning common representation for two views.

2.1 Neural Networks

The field of Neural Networks takes inspiration from how the human brain works. Human Neural System is an extremely complicated and massively parallel system which is responsible for most of our thinking and decision making.

The *artificial neuron* or *perceptron* (Rosenblatt, 1957) is a simple computational unit that mimics the process of a biological neuron. However, it should be remembered that the functionality of even a single biological neuron is highly complex and still unclear. Perceptron is just a simple mathematical abstraction of how a biological neuron acts.

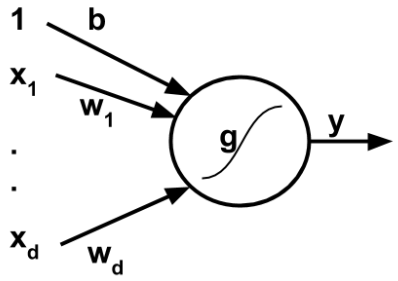

Figure 2.1: An Artificial Neuron

The neuron model is illustrated in Figure-2.1. Let $X \in \mathbb{R}^d$ where d is the number of features. The neuron computes a pre-activation function $a(x)$ given by

$$a(x) = b + \sum_{i=1}^{d} w_i x_i \tag{2.1}$$

where $W = (w_1, ..., w_d)$ is a d-dimensional weight vector and b is the bias. The pre-activation function can also be written as

$$a(x) = b + W^T X \tag{2.2}$$

in vectorial representation. Now the neuron applies some activation function $g : \mathbb{R} \to \mathbb{R}$ on the pre-activation to compute the output y.

$$y = g(a(X)) = g(b + W^T X) \tag{2.3}$$

The activation function can be any linear or non-linear transformation. Few commonly used activation functions are listed below.

1. Linear activation function $g(a) = a$.
2. Sigmoid activation function $g(a) = sigm(a) = \frac{1}{1+exp(a)}$.
3. Hyperbolic tangent activation function $g(a) = tanh(a)$.
4. Rectified linear activation function $g(a) = reclin(a) = max(0, a)$.

A single neuron described above can solve any linearly separable problem and *only* linearly separable problems. In chapter 1, we saw an example (XOR problem) where the perceptron can fail (Minsky and Papert, 1969). We also saw that if the input is transformed to a suitable representation, then the problem becomes easy for the perceptron (linearly separable). This is the motivation for designing Neural Networks.

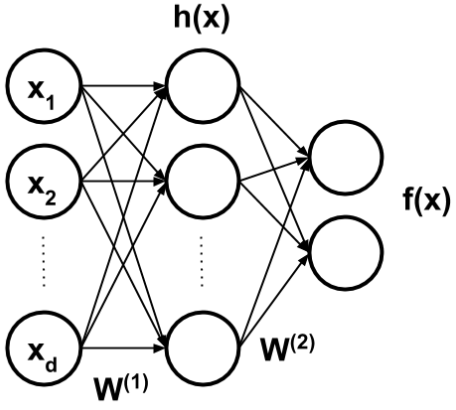

Figure 2.2: Standard three layer neural network. Bias connections are not shown in the figure.

A neural network consists of multiple layers with each layer being defined as a set of neurons. A standard Neural Network as shown in Figure-2.2 consists of three layers: An input layer, a hidden layer and an output layer. Neurons in the input layer are just placeholders for the input and each input neuron is connected to every neuron in the hidden layer. Similarly each neuron in the hidden layer is connected to every neuron in the output layer. Given input X, the hidden layer activation is defined as follows:

$$h(x) = g(b^{(1)} + W^{(1)}X) \tag{2.4}$$

where $W^{(1)}$ is the weight matrix, $b^{(1)}$ is the bias vector and g is the hidden layer activation function. This hidden layer can be considered as a projection of the input features onto some other feature space such that the problem becomes simpler. Now, the hidden layer activation (new features) is given as the input to the output layer neuron to compute the output $f(x)$.

$$f(x) = o(b^{(2)} + W^{(2)}h(x)) \tag{2.5}$$

where $W^{(2)}$ is the weight matrix and $b^{(2)}$ is the bias vector and o is the output activation function. $W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$ are the parameters of the model to be learnt. In the next section, we will discuss an efficient algorithm for learning these parameters given the training data.

When we deal with a multiclass classification problem with k classes, the output layer will contain k neurons and softmax activation is usually used which gives us a valid probability distribution over the k classes. The softmax activation function is defined as follows:

$$softmax(a) = \left[\frac{exp(a_1)}{\sum_i exp(a_i)}, ..., \frac{exp(a_k)}{\sum_i exp(a_i)}\right] \tag{2.6}$$

The universal approximation theorem (Hornik, 1991) says that a three layer neural network described above can model any arbitrary function provided *enough* number of hidden neurons. However the number of hidden neurons required grows exponentially as the complexity of the problem increases.

To give more expressive power to neural networks with relatively smaller number of hidden neurons, one obvious solution is to add more hidden layers which results in

a highly non-linear transformation. One should note that adding more layers makes sense only if the hidden neurons have non-linear activation functions. In case of linear neurons, multiple hidden layers can be replaced by an equivalent single hidden layer with suitable weights and biases.

2.2 Training Neural Networks

In the previous section, we described an artificial neuron and network of neurons. In this section, we will see how to train them efficiently. We will begin the discussion with training a single neuron.

Given $\{(X_i, Y_i)_{i=1}^N\}$ where N is the number of training samples, $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}$, a neuron learns the function f which maps the input X to the output Y. The performance of the neuron is measured by some loss function, say $l(f(X), Y)$ which determines the deviation of the network output $f(X)$ from the true output Y. Some of the commonly used loss functions are listed below.

1. **Squared error function** $l(f(X), Y) = (f(X) - Y)^2$. This is suitable when the output is a real value.
2. **Negative Log Likelihood** $l(f(X), Y) = -\sum_i \mathbb{I}_{Y=i} log f(X)_i$. This is suitable when the output is a probability distribution. NLL is exactly equivalent to cross-entropy between Y and $f(X)$ when the output is binary.

Given a loss function, the training proceeds by learning a set of parameters $\Theta = \{W, b\}$ for the neuron such that the loss over the training data is minimized or the likelihood of the training data is maximized. This is usually done by gradient descent where you take the gradient of the loss function with respect to the parameters and update the parameters based on the gradient such that the training proceeds towards a local minimum of the loss function.

The vanila version of gradient descent which takes one example at a time and updates the parameters is called as online *Stochastic Gradient Descent* (SGD) and the update equation for parameter θ is given by,

$$\begin{aligned} v_{t+1} &= -\alpha \nabla l(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.7}$$

where α is the learning rate. The learning rate decides the magnitude of the step taken by SGD towards the local minimum. Usually we will go through the training set multiple passes and each pass through the training set is called as an epoch. This simple first order gradient descent is sufficient in most of the cases. However, to train a neural network, we will see more advanced optimization procedures in the next section.

Training a neural network in general is not as trivial as the training procedure described for the single neuron. The loss function for a neural network with multiple layers is a composition of several sub-functions and thus deriving gradients with respect to each parameter is difficult. Rumelhart *et al.* (1986) introduced what we now call as the *backpropagation* algorithm which is an efficient way of computing gradients for function compositions based on the chain rule for derivatives.

The idea of backpropagation is simple. When we train a neural network, we compute each layer output based on the input from the previous layer. Doing this computation from input to the output layer is known as forward propagation. Now we compute the gradient for the output layer and backpropagate the gradients until the input layer using chain rule. This is computationally very efficient.

However, deriving the gradient update rules for each model is a tedious process and there are automatic gradient computation tools available which can be used to automate the backpropagation. Most commonly used tools are Theano (Bergstra *et al.*, 2010) and Torch (Collobert *et al.*, 2011*a*). We have used Theano for all of our experiments.

Initialization of weights and the type of activation functions used play a major role in the performance and they are decided based on the kind of problem and data. When we have huge data, which is typical in most of the experiments we will see, using mini-batch gradient descent is more effective. In mini-batch gradient descent we update the parameters after seeing a mini-batch of training examples rather than a single example. This is computationally more efficient since it can exploit the available advancements in doing fast matrix multiplications.

2.3 Advanced Gradient Descent techniques

We discussed about a vanilla Stochastic Gradient Descent and its mini-batch version in the previous section. However, there are several improvements over SGD which tries to accelerate the learning process.

2.3.1 Classical Momentum

The momentum based method to accelerate gradient descent was proposed in (Polyak, 1964). We refer to this method as Classical Momentum (CM) method. CM is a technique for accelerating gradient descent by accumulating a velocity vector in directions of persistent reduction in the objective across iterations. The update equations are given by:

$$\begin{aligned} v_{t+1} &= \mu v_t - \alpha \nabla l(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.8}$$

where $\mu \in [0, 1]$ is the momentum coefficient.

2.3.2 Nesterov's Accelerated Gradient

Nesterov's Accelerated Gradient (Nesterov, 1983) is another way to accelerate the GD and is typically faster than CM in many situations. While CM computes the gradient update from the current position θ_t, NAG first performs a partial update to θ_t, computing $\theta_t + \mu v_t$, which is similar to θ_{t+1}, but missing the as yet unknown correction. This results in faster convergence. The update equations for NAG are given by:

$$\begin{aligned} v_{t+1} &= \mu v_t - \alpha \nabla l(\theta_t + \mu v_t) \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.9}$$

2.3.3 Adagrad

Recently there has been some work on designing first order optimization algorithms that has some properties of second order methods. The update rule in Adagrad (Duchi

et al., 2011) is given by:

$$\begin{aligned} v_{t+1} &= -\frac{\alpha}{\sqrt{\sum_{\tau=1}^{t}(\nabla l(\theta_\tau))^2}}\nabla l(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.10}$$

This method uses a global learning rate α. However, it also has its own dynamic rate for each dimension which grows with the inverse of the gradient magnitudes. This makes sure that the progress along each dimension evens out over time. However, the learning rate will decrease continuously due to the increase in the denominator and learning will eventually stop.

2.3.4 RMSProp

RMSProp (Tieleman and Hinton, 2012) avoids the limitation of Adagrad by using a running average of the denominator term instead of the fixed average. The update equations for RMSProp are given as follows:

$$\begin{aligned} E[\nabla l(\theta_t)^2] &= \rho E[\nabla l(\theta_{t-1})^2] + (1-\rho)(\nabla l(\theta_t))^2 \\ RMS[\nabla l(\theta_t)] &= \sqrt{E[\nabla l(\theta_t)^2] + \epsilon} \\ v_{t+1} &= -\frac{\alpha}{RMS[\nabla l(\theta_t)]}\nabla l(\theta_t) \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.11}$$

where ρ is a decay constant and ϵ is added for numerical stability.

2.3.5 Adadelta

Adadelta (Zeiler, 2012) is similar to RMSProp with unit corrections which leads to proper updates. In case of Adadelta, the weight updates are done as follows:

$$\begin{aligned} E[\nabla l(\theta_t)^2] &= \rho E[\nabla l(\theta_{t-1})^2] + (1-\rho)(\nabla l(\theta_t))^2 \\ v_{t+1} &= -\frac{RMS[v_{t-1}]}{RMS[\nabla l(\theta_t)]}\nabla l(\theta_t) \\ E[v_t^2] &= \rho E[v_{t-1}^2] + (1-\rho)v_t^2 \\ \theta_{t+1} &\leftarrow \theta_t + v_{t+1} \end{aligned} \tag{2.12}$$

Designing better optimization algorithm for training neural networks is one of the active areas of research and there are several other methods proposed in the recent literature (Martens, 2010; Dauphin *et al.*, 2014). We will explicitly mention the optimization procedure we use in all the experiments.

2.4 Autoencoders

An auto-encoder consists of an encoder followed by a decoder (Rumelhart *et al.*, 1986). The encoder is a function f that maps an input $x \in \mathbb{R}^{d_x}$ to hidden representation $h(x)$ $\in \mathbb{R}^{d_h}$. It can be defined as

$$h = f(x) = s_f(Wx + b_h) \tag{2.13}$$

where s_f is a nonlinear activation function like sigmoid function. The parameters of the encoder are a d_h X d_x weight matrix W and a bias vector $b_h \in \mathbb{R}^{d_h}$.

The decoder function g maps the hidden representation h back to a reconstruction y such that,

$$y = g(h) = s_g(W'h + b_y) \tag{2.14}$$

where s_g is the decoder's activation function, typically either the identity or a sigmoid. The decoder's parameters are the matrix W' and a bias vector $b_y \; in \; \mathbb{R}^{d_x}$. In general, W' = W^T.

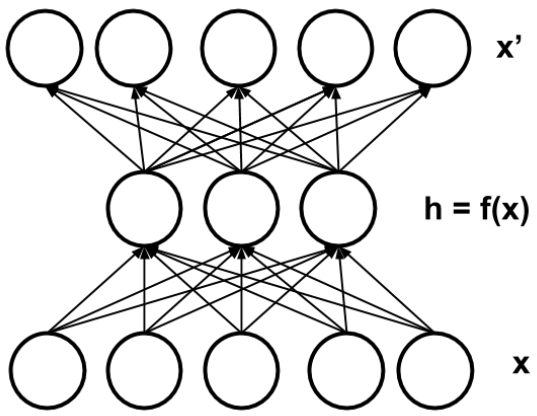

Figure 2.3: An Autoencoder

The auto-encoder is trained to find the parameters θ = $\{W, W', b_h, b_x\}$ such that

the reconstruction error is minimum. If D_n is the set of training examples, then the objective function to be minimized is given by,

$$\mathcal{J}_{AE}(\theta) = \sum_{x \in D_n} L(x, g(f(x)) \tag{2.15}$$

where L is the reconstruction error. Squared error function is typically used.

Autoencoders are typically used for dimensionality reduction and when the hidden layer is linear, then the autoencoder learns the top principal components of the data. In other words, it is equivalent to doing Principal Component Analysis (PCA) on the data. Autoencoders are also the building blocks for unsupervised feature learning in Deep Learning as explained in the later section.

2.5 Restricted Boltzmann Machines (RBMs)

An RBM (Smolensky, 1986) is an energy based model which can be considered as a Markov Random Field associated with a bipartite undirected graph. It consists of m visible units $V = (v_1, v_2, ..., v_m)$ to represent the observable data and n hidden units $H = (h_1, h_2, ..., h_n)$ to capture dependencies between observed variables. Consider binary RBMs where each visible or hidden unit can take only binary values. The energy function for a particular configuration of (V, H) is given by,

$$E(V, H) = -\sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} a_i h_i \tag{2.16}$$

where W is the interaction weight matrix between hidden and visible units, b is the bias vector associated with visible units and a is the bias vector associated with hidden units.

Now the joint probability distribution is given by the Gibbs distribution

$$p(V, H) = \frac{1}{Z} e^{-E(V,H)} \tag{2.17}$$

where Z is the normalization constant (also known as the partition function).

RBM is called as $Restricted$ Boltzmann Machine because of the restriction in the connections. RBM has only connections across the visible and hidden layers and not inside the layers. This makes one hidden variables independent given the state of the visible variables and vice versa.

$$\begin{aligned} p(h|v) &= \prod_{i=1}^{n} p(h_i|v) \\ p(v|h) &= \prod_{i=1}^{m} p(v_i|h) \end{aligned} \tag{2.18}$$

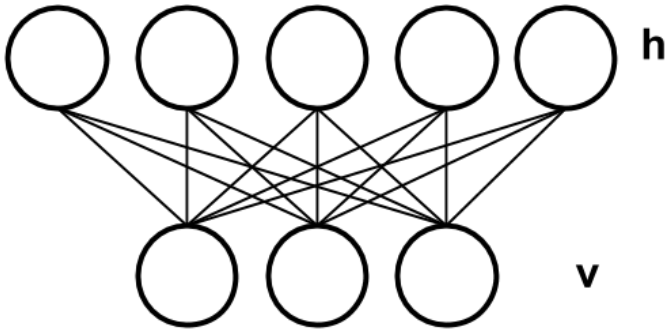

Figure 2.4: A Restricted Boltzmann Machine

RBMs can also be interpreted as a stochastic neural network. The conditional probability of a single variable being one can be defined as follows:

$$p(h_i = 1|v) = \sigma\left(\sum_{j=1}^{m} W_{ij} v_j + a_i\right) \tag{2.19}$$

$$p(v_j = 1|h) = \sigma\left(\sum_{i=1}^{n} W_{ij} h_i + b_j\right) \tag{2.20}$$

RBMs are trained by maximizing the likelihood of the data and computing likelihood is intractable in RBMs because of the partition function. So this is often approximated and most common way to do this is by training based on Contrastive Divergence (CD) (Hinton, 2002). RBMs along with Autoencoders form the building blocks of Deep Neural Networks.

2.6 Greedy Layerwise Training and Deep Learning

A single hidden layer auto-encoder basically provides a non-linear transformation of the input to the hidden layer. A novel methodology to stack multiple auto-encoders to design a deep neural network was proposed in (Bengio *et al.*, 2007). Similar greedy layer-wise training using Restricted Boltzman Machines was proposed in (Hinton *et al.*, 2006). Both the methods consists of an unsupervised pre-training phase followed by supervised fine-tuning phase. We will describe the auto-encoder based training (Bengio *et al.*, 2007) since it is most relevant to the thesis.

1. Train the first layer of the autoencoder to minimize some form of reconstruction error of the raw input. This is purely unsupervised.
2. The outputs of the hidden units of this autoencoder is used as input for another layer which is also trained to be an autoencoder. Again, this is also unsupervised.
3. Iterate as in (2) to add the desired number of layers.
4. Take the last hidden layer output as input to a supervised layer and initialize its parameters.
5. Fine-tune all the parameters of this deep architecture with respect to the supervised criterion. Alternately, unfold all the autoencoders into a very deep autoencoder and fine-tune the global reconstruction error.

This unsupervised pretraining results in better initialization of the parameters of the network which helps in the supervised training phase. The entire procedure is illustrated in Figure-2.5.

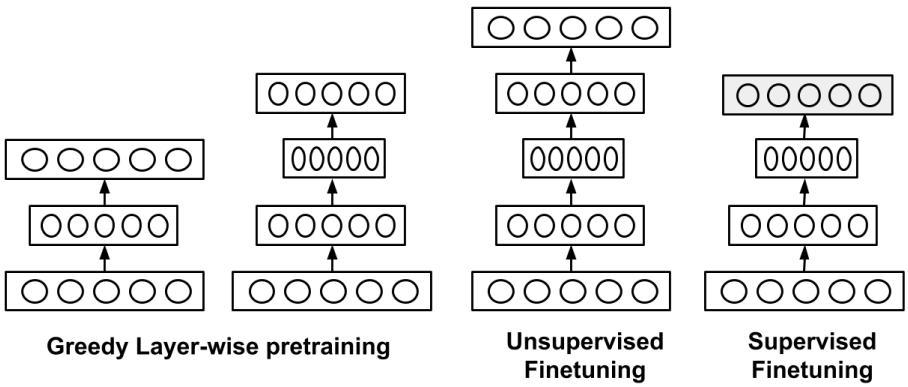

Figure 2.5: Procedure to train a deep neural network using autoencoders. After greedy layer-wise pretraining, we can do either unsupervised finetuning or supervised finetuning.

2.7 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) (Hotelling, 1936) is a de-facto tool for learning common representation for two different views in the literature (Udupa and Khapra, 2010; Dhillon *et al.*, 2011).

Let $X_1 \in \mathbb{R}^{n1}$ and $X_2 \in \mathbb{R}^{n2}$ denote random vectors with covariances Σ_{11} and Σ_{22} respectively. Let Σ_{12} be the cross-covariance. CCA finds pairs of linear projections of the two views: $w_1'X_1$ and $w_2'X_2$ that are maximally correlated:

$$\begin{aligned}(w_1^*, w_2^*) &= \underset{w_1,w_2}{\arg\max}\ corr(w_1'X_1, w_2'X_2) \\ &= \underset{w_1,w_2}{\arg\max}\ \frac{w_1'\Sigma_{12}w_2}{\sqrt{w_1'\Sigma_{11}w_1 w_2'\Sigma_{22}w_2}}\end{aligned} \tag{2.21}$$

Kernel CCA (Akaho, 2001) uses the standard kernel trick to find pairs of non-linear projections of the two views. Deep CCA, a deep version of CCA is also introduced in (Andrew *et al.*, 2013).

Even though CCA and its variants are used widely for learning common representations, we would like to highlight few limitations of CCA which makes them unusable in certain situations.

- CCA is easily not scalable. Even though there are several work on scaling up CCA (Lu and Foster, 2014), they are all approximation to CCA and hence the decrease in the performance.
- It is not very trivial to extend CCA to multiple views. However there are some recent work along this line (Tenenhaus and Tenenhaus, 2011; Luo *et al.*, 2015) which requires complex computations.
- CCA is unidirectional in projections. In other words, you cannot get back the data point given its projection. If you can have a bidirectional projection, then it will facilitate translating data points from one view to another view.
- CCA can work only on parallel data. However, in real life situations, parallel data is costly when compared to single view data. So it is ideally expect to make best use of the single view data as well.

All the above-mentioned limitations suggest that CCA is not a final solution for Common Representation Learning. However, CCA has been widely used because of its superior performance over other existing CRL algorithms.

2.8 Summary

This chapter provides a summary of Neural networks and Deep Learning which is essential to understand the rest of the thesis. We also described Canonical Correlation Analysis, a standard CRL algorithm and its limitations. In the next chapter, we will see a Neural network based approach for common representation learning.

CHAPTER 3

CORRELATIONAL NEURAL NETWORKS

In this chapter, we introduce Correlational Neural Networks. Correlational Neural Network (CorrNet) is a three layer neural network that learns a common representation for two different views of the data. CorrNet is for multi-view data, as an autoencoder is for single view data.

3.1 Introduction

Let us restate the problem of Common Representation Learning. Consider some data $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^{N}$ which has two views: X and Y. Each data point $\mathbf{z}_i$ can be represented as a concatenation of these two views : $\mathbf{z}_i = (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i \in \mathbb{R}^{d_1}$ and $\mathbf{y}_i \in \mathbb{R}^{d_2}$. In this work, we are interested in learning two functions, h_X and h_Y, such that $h_X(\mathbf{x}_i) \in \mathbb{R}^k$ and $h_Y(\mathbf{y}_i) \in \mathbb{R}^k$ are projections of $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively in a common subspace ($\mathbb{R}^k$) such that for a given pair $\mathbf{x}_i$, $\mathbf{y}_i$:

1. $h_X(\mathbf{x}_i)$ and $h_Y(\mathbf{y}_i)$ should be highly correlated.
2. It should be possible to reconstruct $\mathbf{y}_i$ from $\mathbf{x}_i$ (through $h_X(\mathbf{x}_i)$) and vice versa.

Canonical Correlation Analysis (CCA) (Hotelling, 1936) is a commonly used tool for learning such common representations for two-view data (Udupa and Khapra, 2010; Dhillon *et al.*, 2011). By definition, CCA aims to produce correlated common representations but, it suffers from some drawbacks. First, it is not easily scalable to very large datasets. Of course, there are some approaches which try to make CCA scalable (for example, (Lu and Foster, 2014)), but such scalability comes at the cost of performance. Further, CCA does not have any reconstruction capabilities, *i.e.*, it cannot be used to reconstruct one view from the other. Finally, CCA cannot benefit from additional non-parallel, single-view data. This puts it at a severe disadvantage in several real world situations, where in addition to some parallel two-view data, abundant single view data is available for one or both views.

Recently, Multimodal Autoencoders (MAEs) (Ngiam *et al.*, 2011) have been proposed to learn a common representation for two views/modalities. The idea in MAE is to train an autoencoder to perform two kinds of reconstruction. Given any one view, the model learns both self-reconstruction and cross-reconstruction (reconstruction of the other view). This makes the representations learnt to be predictive of each other. However, it should be noticed that the MAE does not get any explicit learning signal encouraging it to share the capacity of its common hidden layer between the views. In other words, it could develop units whose activation is dominated by a single view. This makes the MAE not suitable for transfer learning, since the views are not guaranteed to be projected to a common subspace. This is indeed verified by the results reported in (Ngiam *et al.*, 2011) where they show that CCA performs better than deep MAE for the task of transfer learning.

These two approaches have complementary characteristics. On one hand, we have CCA and its variants which aim to produce correlated common representations but lack reconstruction capabilities. On the other hand, we have MAE which aims to do self-reconstruction and cross-reconstruction but does not guarantee correlated common representations. In this thesis, we propose Correlational Neural Network (CorrNet) as a method for learning common representations which combines the advantages of the two approaches described above. The main characteristics of the proposed method can be summarized as follows:

- It allows for self/cross reconstruction. Thus, unlike CCA (and like MAE) it has predictive capabilities. This can be useful in applications where a missing view needs to be reconstructed from an existing view.
- Unlike MAE (and like CCA) the training objective used in CorrNet ensures that the common representations of the two views are correlated. This is particularly useful in applications where we need to match items from one view to their corresponding items in the other view.
- CorrNet can be trained using Gradient Descent based optimization methods. Particularly, when dealing with large high dimensional data, one can use Stochastic Gradient Descent with mini-batches. Thus, unlike CCA (and like MAE) it is easy to scale CorrNet.
- The procedure used for training CorrNet can be easily modified to benefit from additional single view data. This makes CorrNet useful in many real world applications where additional single view data is available.

We use the MNIST hand-written digit recognition dataset to compare CorrNet with other state of the art CRL approaches. In particular, we evaluate its (i) ability to

self/cross reconstruct (ii) ability to produce correlated common representations and (iii) usefulness in transfer learning. In this setup, we use the left and right halves of the digit images as two views. Next, we use CorrNet for the task of transliteration equivalence where the aim is to match a name written using the script of one language (first view) to the same name written using the script of another language (second view). Here again, we demonstrate that with its ability to produce better correlated common representations, CorrNet performs better than CCA and MAE.

3.2 The Model

As described earlier, our aim is to learn a common representation from two views of the same data such that: (i) any single view can be reconstructed from the common representation, (ii) a single view can be predicted from the representation of another view and (iii) like CCA, the representations learned for the two views are correlated. The first goal above can be achieved by a conventional autoencoder. The first and second can be achieved together by a Multimodal autoencoder but it is not guaranteed to project the two views to a common subspace. We propose a variant of autoencoders which can work with two views of the data, while being explicitly trained to achieve all the above goals. In this section and the next section, we describe our model and the training procedure. We start by proposing a neural network architecture which contains three layers: an input layer, a hidden layer and an output layer. Just as in a conventional single view autoencoder, the input and output layers have the same number of units, whereas the hidden layer can have a different number of units. For illustration, we consider a two-view input $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. For all the discussions, $[\mathbf{x}, \mathbf{y}]$ denotes a concatenated vector of size $d_1 + d_2$.

Given $\mathbf{z} = (\mathbf{x}, \mathbf{y})$, the hidden layer computes an encoded representation as follows:

$$h(\mathbf{z}) = f(\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{y} + \mathbf{b}) \tag{3.1}$$

where $\mathbf{W}$ is a $k \times d_1$ projection matrix, $\mathbf{V}$ is a $k \times d_2$ projection matrix and $\mathbf{b}$ is a $k \times 1$ bias vector. Function f can be any non-linear activation function, for example *sigmoid* or *tanh*. The output layer then tries to reconstruct $\mathbf{z}$ from this hidden representation by

computing

$$\mathbf{z}' = g([\mathbf{W}'h(\mathbf{z}), \mathbf{V}'h(\mathbf{z})] + \mathbf{b}') \tag{3.2}$$

where $\mathbf{W}'$ is a $d_1 \times k$ reconstruction matrix, $\mathbf{V}'$ is a $d_2 \times k$ reconstruction matrix and $\mathbf{b}'$ is a $(d_1 + d_2) \times 1$ output bias vector. Vector $\mathbf{z}'$ is the reconstruction of $\mathbf{z}$. Function g can be any activation function. This architecture is illustrated in Figure 3.1. The parameters of the model are $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{W}', \mathbf{V}', \mathbf{b}, \mathbf{b}'\}$. In the next section we outline a procedure for learning these parameters.

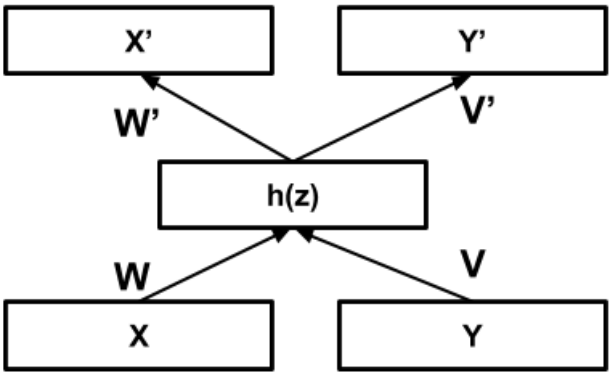

Figure 3.1: Correlational Neural Network

3.3 Training

In this section, we will describe how to train CorrNets. As mentioned before, CorrNets can be trained with only parallel data or with both parallel and single view data.

3.3.1 Training with parallel data

Restating our goals more formally, given a two-view data $\mathcal{Z} = \{(\mathbf{z}_i)\}_{i=1}^N = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, for each instance, $(\mathbf{x}_i, \mathbf{y}_i)$, we would like to:

- Minimize the self-reconstruction error, *i.e.*, minimize the error in reconstructing $\mathbf{x}_i$ from $\mathbf{x}_i$ and $\mathbf{y}_i$ from $\mathbf{y}_i$.
- Minimize the cross-reconstruction error, *i.e.*, minimize the error in reconstructing $\mathbf{x}_i$ from $\mathbf{y}_i$ and $\mathbf{y}_i$ from $\mathbf{x}_i$.
- Maximize the correlation between the hidden representations of both views.

We achieved this by finding the parameters $\theta = \{\mathbf{W}, \mathbf{V}, \mathbf{W}', \mathbf{V}', \mathbf{b}, \mathbf{b}'\}$ which minimize the following objective function:

$$\mathcal{J}_{\mathcal{Z}}(\theta) = \sum_{i=1}^{N}(L(\mathbf{z}_i, g(h(\mathbf{z}_i))) + L(\mathbf{z}_i, g(h(\mathbf{x}_i))) + L(\mathbf{z}_i, g(h(\mathbf{y}_i)))) - \lambda \operatorname{corr}(h(X), h(Y)) \tag{3.3}$$

$$\operatorname{corr}(h(X), h(Y)) = \frac{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})(h(\mathbf{y}_i) - \overline{h(Y)})}{\sqrt{\sum_{i=1}^{N}(h(\mathbf{x}_i) - \overline{h(X)})^2 \sum_{i=1}^{N}(h(\mathbf{y}_i) - \overline{h(Y)})^2}} \tag{3.4}$$

where L is the reconstruction error, λ is the scaling parameter to scale the fourth term with respect to the remaining three terms, $\overline{h(X)}$ is the mean vector for the hidden representations of the first view and $\overline{h(Y)}$ is the mean vector for the hidden representations of the second view. If all dimensions in the input data take binary values then we use cross-entropy as the reconstruction error otherwise we use squared error loss as the reconstruction error. For simplicity, we use the shorthands $h(\mathbf{x}_i)$ and $h(\mathbf{y}_i)$ to note the representations $h((\mathbf{x}_i, 0))$ and $h((0, \mathbf{y}_i))$ that are based only on a single view[1]. The correlation term in the objective function is calculated considering the hidden representation as a random vector.

In words, the objective function decomposes as follows. The first term is the usual autoencoder objective function which helps in learning meaningful hidden representations. The second term ensures that both views can be predicted from the shared representation of the first view alone. The third term ensures that both views can be predicted from the shared representation of the second view alone. The fourth term interacts with the other objectives to make sure that the hidden representations are highly correlated, so as to encourage the hidden units of the representation to be shared between views.

We can use stochastic gradient descent (SGD) to find the optimal parameters. For all our experiments, we used mini-batch SGD. The fourth term in the objective function is then approximated based on the statistics of a minibatch. Approximating second order statistics using minibatches for training was also used successfully in the batch normalization training method of Ioffe and Szegedy (2015).

The model has three hyperparameters: (i) the number of units in its hidden layer, (ii) λ and (iii) the SGD learning rate. The first hyperparameter is dependent on the specific

[1]They represent the generic functions h_X and h_Y mentioned in the introduction.

task at hand and can be tuned using a validation set (exactly as is done by other competing algorithms). The second hyperparameter is only to ensure that the correlation term in the objective function has the same range as the reconstruction errors. This is again easy to approximate based on the given data. The final hyperparameter, the learning rate is common for all neural network based approaches.

Once the parameters are learned, we can use the CorrNet to compute representations of views that can potentially generalize across views. Specifically, given a new data instance for which only one view is available, we can compute its corresponding representation ($h(\mathbf{x})$ if $\mathbf{x}$ is observed or $h(\mathbf{y})$ if $\mathbf{y}$ is observed) and use it as the new data representation.

3.3.2 Training with single view data

In practice, it is often the case that we have abundant single view data and comparatively little two-view data. For example, in the context of text documents from two languages (X and Y), typically the amount of monolingual (single view) data available in each language is much larger than parallel (two-view) data available between X and Y. Given the abundance of such single view data, it is desirable to exploit it in order to improve the learned representation. CorrNet can achieve this, by using the single view data to improve the self-reconstruction error as explained below.

Consider the case where, in addition to the data $\mathcal{Z} = \{(\mathbf{z}_i)\}_{i=1}^{N} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, we also have access to the single view data $\mathcal{X} = \{(\mathbf{x}_i)\}_{i=N+1}^{N_1}$ and $\mathcal{Y} = \{(\mathbf{y}_i)\}_{i=N+1}^{N_2}$. Now, during training, in addition to using $\mathcal{Z}$ as explained before, we also use $\mathcal{X}$ and $\mathcal{Y}$ by suitably modifying the objective function so that it matches that of a conventional autoencoder. Specifically, when we have only $\mathbf{x}_i$, then we could try to minimize

$$\mathcal{J}_{\mathcal{X}}(\theta) = \sum_{i=N+1}^{N_1} L(\mathbf{x}_i, g(h(\mathbf{x}_i)))$$

and similarly for $\mathbf{y}_i$.

In all our experiments, when we have access to all three types of data (*i.e.*, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$), we construct 3 sets of mini-batches by sampling data from $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively. We then feed these mini-batches in random order to the model and perform a gradient

update based on the corresponding objective function.

3.4 Related Models

In this section, we briefly describe other neural network based common representation learning models and explain how CorrNet differs from them.

Hsieh (2000) is one of the earliest Neural Network based model for nonlinear CCA. This method uses three feedforward neural networks. The first neural network is a double-barreled architecture where two networks project the views to a single unit such that the projections are maximally correlated. This network is first trained to maximize the correlation. Then the inverse mapping for each view is learnt from the corresponding canonical covariate representation by minimizing the reconstruction error. There are clear differences between this Neural CCA model and CorrNet. First, CorrNet is a single neural network which is trained with a single objective function while Neural CCA has three networks trained with different objective functions. Second, Neural CCA does only correlation maximization and self-reconstruction, whereas CorrNet does correlation maximization, self-reconstruction and cross-reconstruction, all at the same time.

Multimodal Autoencoder (MAE) (Ngiam *et al.*, 2011) is another Neural Network based CRL approach. Even though the architecture of MAE is similar to that of CorrNet there are clear differences in the training procedure used by the two. Firstly, MAE only aims to minimize the following three errors: (i) error in reconstructing z_i from x_i (E_1), (ii) error in reconstructing z_i from y_i (E_2) and (iii) error in reconstructing z_i from z_i (E_3). More specifically, unlike the fourth term in our objective function, the objective function used by MAE does not contain any term which forces the network to learn correlated common representations. Secondly, there is a difference in the manner in which these terms are considered during training. Unlike CorrNet, MAE only considers one of the above terms at a time. In other words, given an instance $z_i = (x_i, y_i)$ it first tries to minimize E_1 and updates the parameters accordingly. It then tries to minimize E_2 followed by E_3. Empirically, we observed that a training procedure which considers all three loss terms together performs better than the one which considers them separately.

Deep Canonical Correlation Analysis (DCCA) (Andrew *et al.*, 2013) is a recently proposed Neural Network approach for CCA. DCCA employs two deep networks, one

per view. The model is trained in such a way that the final layer projections of the data in both the views are maximally correlated. DCCA maximizes only correlation whereas CorrNet maximizes both, correlation and reconstruction ability.

3.5 Experiments on MNIST handwritten digits

In this section, we analyze and experimentally verify the use of Correlational Networks in various scenarios where shared representation learning is needed.

In this section, we perform a set of experiments to compare CorrNet, CCA (Hotelling, 1936), Kernel CCA (KCCA) (Akaho, 2001) and MAE (Ngiam *et al.*, 2011) based on:

- ability to reconstruct a view from itself
- ability to reconstruct one view given the other
- ability to learn correlated common representations for the two views
- usefulness of the learned common representations in transfer learning.

For CCA, we used a C++ library called *dlib* (King, 2009). For KCCA, we used an implementation provided by the authors of (Arora and Livescu, 2012). We implemented CorrNet and MAE using Theano (Bergstra *et al.*, 2010).

3.5.1 Data Description

We used the standard MNIST handwritten digits image dataset for all our experiments. This data consists of 60,000 train images and 10,000 test images. Each image is a 28 * 28 matrix of pixels; each pixel representing one of 256 grayscale values. We treated the left half of the image as one view and the right half of the image as another view. Thus each view contains $14 * 28 = 392$ dimensions. We split the train images into two sets. The first set contains 50,000 images and is used for training. The second set contains 10,000 images and is used as a validation set for tuning the hyper-parameters of the four models described above.

3.5.2 Performance of Self and Cross Reconstruction

Among the four models listed above, only CorrNets and MAE have the ability to construct a view from itself as well as from the other view. So, in this sub-section, we consider only these two models. Table 3.1 shows the Mean Squared Errors (MSEs) for self and cross reconstruction when the left half of the image is used as input.

Model	MSE for self reconstruction	MSE for cross reconstruction
CorrNet	3.6	4.3
MAE	**2.1**	**4.2**

Table 3.1: Mean Squared Error for CorrNet and MAE for self reconstruction and cross reconstruction

The above table suggests that CorrNet has a higher self reconstruction error and almost the same cross reconstruction error as that of MAE. This is because unlike MAE, in CorrNet, the emphasis is on maximizing the correlation between the common representations of the two views. This goal captured by the fourth term in the objective function obviously interferes with the goal of self reconstruction. As we will see in the next sub-section, the embeddings learnt by CorrNet for the two views are better correlated even though the self-reconstruction error is sacrificed in the process.

Figure 3.2 shows the reconstruction of the right half from the left half for a few sample images. The figure reiterates our point that both CorrNet and MAE are equally good at cross reconstruction.

Figure 3.2: Reconstruction of right half of the image given the left half. First block shows the original images, second block shows images where the right half is reconstructed by CorrNet and the third block shows images where the right half is reconstructed by MAE.

3.5.3 Correlation between representations of two views

As mentioned above, in CorrNet we emphasize on learning highly correlated representations for the two views. To show that this is indeed the case, we follow (Andrew *et al.*, 2013) and calculate the total/sum correlation captured in the 50 dimensions of the common representations learnt by the four models described above. The training, validation and test sets used for this experiment were as described in section 3.5.1. The results are reported in Table 3.2.

Model	**Sum Correlation**
CCA	17.05
KCCA	30.58
MAE	24.40
CorrNet	**45.47**

Table 3.2: Sum/Total correlation captured in the 50 dimensions of the common representations learned by different models using MNIST data.

The total correlation captured in the 50 dimensions learnt by CorrNet is clearly better than that of the other models.

Next, we check whether this is indeed the case when we change the number of dimensions. For this, we varied the number of dimensions from 5 to 80 and plotted the sum correlation for each model (see Figure 3.3). For all the models, we tuned the hyper-parameters for $dim = 50$ and used the same hyper-parameters for all dimensions. For this experiment, we used 10,000 images for training.

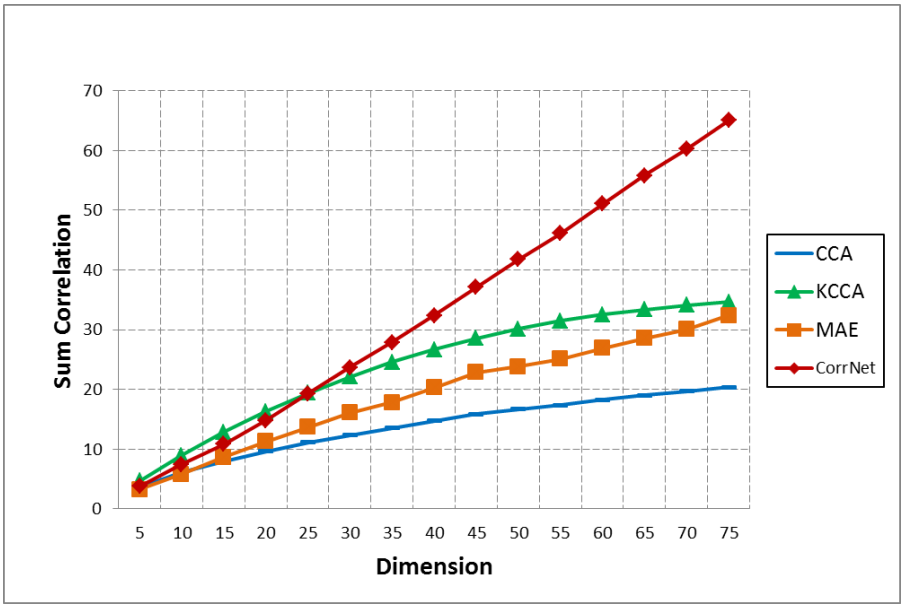

Figure 3.3: Sum/Total correlation as a function of the number of dimensions in the common representations learned by different models using MNIST data.

Again, we see that CorrNet clearly outperforms the other models. CorrNet thus achieves its primary goal of producing correlated embeddings with the aim of assisting transfer learning.

3.5.4 Transfer Learning across views

To demonstrate transfer learning, we take the task of predicting digits from only one half of the image. We first learn a common representation for the two views using 50,000 images from the MNIST training data. For each training instance, we take only one half of the image and compute its 50 dimensional common representation using one of the models described above. We then train a classifier using this representation. For each test instance, we consider only the other half of the image and compute its common representation. We then feed this representation to the classifier for prediction. We use the linear SVM implementation provided by (Pedregosa *et al.*, 2011) as the classifier for all our experiments and do 5-fold cross validation using 10000 test images. We report accuracy for two settings (i) Left to Right (training on left view, testing on right view) and (ii) Right to Left (training on right view, testing on left view).

Model	Left to Right	Right to Left
CCA	65.73	65.44
KCCA	68.1	75.71
MAE	64.14	68.88
CorrNet	**77.05**	**78.81**

Table 3.3: Transfer learning accuracy using the representations learned using different models on the MNIST dataset.

Once again, we see that CorrNet performs significantly better than the other models. To verify that this holds even when we have less data, we decrease the data for learning common representation to 10000 images. The results as reported in Table 3.4 show that even with less data, CorrNet perform betters than other models.

Model	Left to Right	Right to Left
CCA	66.13	66.71
KCCA	70.68	70.83
MAE	68.69	72.54
CorrNet	**76.6**	**79.51**

Table 3.4: Transfer learning accuracy using the representations learned using different models trained with 10000 instances from the MNIST dataset.

3.5.5 Relation with MAE

At the face of it, it may seem that both CorrNet and MAE differ only in their objective functions. Specifically, if we remove the last correlation term from the objective function of CorrNet then it would become equivalent to MAE. To verify this, we conducted experiments using both MAE and CorrNet without the last term (say CorrNet(123)). When using SGD to train the networks, we found that the performance is almost similar. However, when we use some advanced optimization technique like RMSProp, CorrNet(123) starts performing better than MAE. The results are reported in Table 3.5.

Model	Optimization	Left to Right	Right to Left
MAE	SGD	63.9	67.98
CorrNet(123)	SGD	63.89	67.93
MAE	RMSProp	64.14	68.88
CorrNet(123)	RMSProp	67.82	72.13

Table 3.5: Results for transfer learning across views

This experiment sheds some light on the understanding why MAE is inferior. Even-though the objective of MAE and CorrNet(123) is same, MAE tries to solve it in a stochastic way which adds more noise. However, CorrNet(123) performs better since it is actually working on the combined objective function and not the stochastic version (one term at a time) of it.

3.5.6 Analysis of Loss Terms

The objective function defined in Section 3.3.1 has the following four terms:

- $L_1 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{z}_i))$
- $L_2 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{x}_i))$
- $L_3 = \sum_{i=1}^{N} L(\mathbf{z}_i, g(h(\mathbf{y}_i))$
- $L_4 = \lambda \operatorname{corr}(h(X), h(Y))$

In this section, we analyze the importance of each of these terms in the loss function. For this, during training, we consider different loss functions which contain different combinations of these terms. In addition, we consider two more loss terms for our analysis.

- $L_5 = \sum_{i=1}^{N} L(\mathbf{y}_i, g(h(\mathbf{x}_i))$
- $L_6 = \sum_{i=1}^{N} L(\mathbf{x}_i, g(h(\mathbf{y}_i))$

where $L5$ and $L6$ essentially capture the loss in reconstructing only one view (say, $\mathbf{x}_i$) from the other view ($\mathbf{y}_i$).

For this, we first learn common representations using different loss functions as listed in the first column of Table 3.6. We then repeated the transfer learning experiments using common representations learned from each of these models. For example, the sixth row in the table shows the results when the following loss function is used for learning the common representations.

$$\mathcal{J}_{\mathcal{Z}}(\theta) = L_1 + L_2 + L_3 + L_4$$

which is the same as that used in CorrNet.

Loss function used for training	**Left to Right**	**Right to Left**
L_1	24.59	22.56
$L_1 + L_4$	65.9	67.54
$L_2 + L_3$	71.54	75
$L_2 + L_3 + L_4$	76.54	**80.57**
$L_1 + L_2 + L_3$	67.82	72.13
$L_1 + L_2 + L_3 + L_4$	**77.05**	78.81
$L_5 + L_6$	35.62	32.26
$L_5 + L_6 + L_4$	62.05	63.49

Table 3.6: Comparison of the performance of transfer learning with representations learned using different loss functions.

Each even numbered row in the table reports the performance when the correlation term (L_4) was used in addition to the other terms in the row immediately before it. A pair-wise comparison of the numbers in each even numbered row with the row immediately above it suggests that correlation term (L_4) in the loss function clearly produces representations which lead to better transfer learning.

3.6 Transliteration Equivalence

In the previous section, we analyzed the characteristics of CorrNet using MNIST dataset as a benchmark. In this section, we show a real application of CorrNet in Natural Language Processing. Specifically, we use CorrNet for matching equivalent items across views. As a case study, we consider the task of determining transliteration equivalence

of named entities wherein given a word u written using the script of language X and a word v written using the script of language Y the goal is to determine whether u and v are transliterations of each other. Several approaches have been proposed for this task and the one most related to our work is an approach which uses CCA for determining transliteration equivalence.

We consider English-Hindi as the language pair for which transliteration equivalence needs to be determined. For learning common representations we used approximately 15,000 transliteration pairs from NEWS 2009 English-Hindi training set (Li *et al.*, 2009). We represent each Hindi word as a bag of 2860 bigram characters. This forms the first view ($\mathbf{x}_i$). Similarly we represent each English word as a bag of 651 bigram characters. This forms the second view ($\mathbf{y}_i$). Each such pair $(\mathbf{x}_i, \mathbf{y}_i)$ then serves as one training instance for the CorrNet.

For testing we consider the standard NEWS 2010 transliteration mining test set (Kumaran *et al.*, 2010). This test set contains approximately 1000 Wikipedia English Hindi title pairs. The original task definition is as follows. For a given English title containing T_1 words and the corresponding Hindi title containing T_2 words identify all pairs which form a transliteration pair. Specifically, for each title pair, consider all $T_1 \times T_2$ word pairs and identify the correct transliteration pairs. In all, the test set contains 5468 word pairs out of which 982 are transliteration pairs. For every word pair $(\mathbf{x}_i, \mathbf{y}_i)$ we obtain a 50 dimensional common representation for $\mathbf{x}_i$ and $\mathbf{y}_i$ using the trained CorrNet. We then calculate the correlation between the representations of $\mathbf{x}_i$ and $\mathbf{y}_i$. If the correlation is above a threshold we mark the word pair as equivalent. This threshold is tuned using an additional 1000 pairs which were provided as training data for the NEWS 2010 transliteration mining task. As seen in Table 3.7 CorrNet clearly performs better than the other methods. Note that our aim is not to achieve state of the art performance on this task but to compare the quality of the shared representations learned using different CRL methods considered in this paper.

Model	F1-measure (%)
CCA	49.68
KCCA	42.36
MAE	72.75
CorrNet	**81.56**

Table 3.7: Performance on NEWS 2010 En-Hi Transliteration Mining Dataset

3.7 Summary

In this chapter, we proposed Correlational Neural Networks as a method for learning common representations for two views of the data. The proposed model has the capability to reconstruct one view from the other and it ensures that the common representations learned for the two views are aligned and correlated. Its training procedure is also scalable. Further, the model can benefit from additional single view data, which is often available in many real world applications. We employ the common representations learned using CorrNet for transliteration equivalence detection. We show that the representations learned using CorrNet perform better than other methods.

CHAPTER 4

CROSS LANGUAGE LEARNING

In this chapter, we will discuss an approach for Cross Language Learning based on Correlational Neural Networks. We propose Bilingual Autoencoding, a way to use correlational networks to learn common representation for words in two different languages. In experiments on 3 language pairs, we show that our approach achieves state-of-the-art performance in Cross Language Document Classification.

4.1 Motivation

Languages show different levels of maturity with respect to their Natural Language Processing (NLP) capabilities. This maturity in terms of the quality and number of NLP tools available for a given language is directly proportional to the amount of annotated resources available for that language. As a result, languages such as English which have plenty of annotated resources at their disposal are better equipped than other languages which are not so fortunate in terms of annotated resources. For example, high quality POS taggers (Toutanova *et al.*, 2003), parsers (Socher *et al.*, 2013), sentiment analyzers (Liu, 2012) are already available for English but this is not the case for many other languages such as Hindi, Marathi, Bodo, Farsi, Urdu, *etc*. This situation was acceptable in the past when only a few languages dominated the digital content available online and elsewhere. However, the ever increasing number of languages on the web today has made it important to accurately process natural language data in such less fortunate languages also. An obvious solution to this problem is to improve the annotated inventory of these languages but the involved cost, time, and effort act as a natural deterrent to this.

While the majority of previous work on vectorial text representations has concentrated on the monolingual case, there has been considerable interest in learning word and document representations that are aligned across languages (Klementiev *et al.*,

2012; Zou *et al.*, 2013; Mikolov *et al.*, 2013). Such aligned representations can potentially allow the use of resources from a resource fortunate language to develop NLP capabilities in a resource deprived language (Yarowsky and Ngai, 2001; Das and Petrov, 2011; Mihalcea *et al.*, 2007; Wan, 2009; Padó and Lapata, 2009). For example, if a common representation model is learned for representing English and German documents, then a classifier trained on annotated English documents can be used to classify German documents (provided we use the learned common representation model for representing documents in both languages).

Such reuse of resources across languages has been tried in the past by projecting parameters learned from the annotated data of one language to another language (Yarowsky and Ngai, 2001; Das and Petrov, 2011; Mihalcea *et al.*, 2007; Wan, 2009; Padó and Lapata, 2009) These projections are enabled by a bilingual resource such as a Machine Translation (MT) system. Recent attempts at learning common bilingual representations (Klementiev *et al.*, 2012; Zou *et al.*, 2013; Mikolov *et al.*, 2013) aim to eliminate the need of such a MT system. Such bilingual representations have been applied to a variety of problems, including cross-language document classification (Klementiev *et al.*, 2012) and phrase-based machine translation (Zou *et al.*, 2013). A common property of these approaches is that a *word-level alignment* of translated sentences is leveraged, *e.g.*, to derive a regularization term relating word embeddings across languages (Klementiev *et al.*, 2012; Zou *et al.*, 2013). Such methods not only eliminate the need for an MT system but also outperform MT based projection approaches.

In this work, we experiment with a method to learn bilingual word representations that *does not require word-to-word alignment* of bilingual corpora during training. Unlike previous approaches (Klementiev *et al.*, 2012), we only require aligned sentences and do not rely on word-level alignments (e.g., extracted using GIZA++, as is usual), which simplifies the learning procedure. To do so, we propose a bilingual autoencoder model, that learns hidden encoder representations of paired bag-of-words sentences which are not only informative of the original bag-of-words but also predictive of each other. Word representations can then easily be extracted from the encoder and used in the context of a supervised NLP task. Specifically, we demonstrate the quality of these representations for the task of cross-language document classification, where a labeled data set can be available in one language, but not in another one. As we'll see, our approach is able to reach state-of-the-art performance, achieving up to 10-14 percentage

point improvements over the best previously reported results.

4.2 Autoencoder for Bags-of-Words

Let $\mathbf{x}$ be the bag-of-words representation of a sentence. Specifically, each x_i is a word index from a fixed vocabulary of V words. As this is a bag-of-words, the order of the words within $\mathbf{x}$ does not correspond to the word order in the original sentence. We wish to learn a D-dimensional vectorial representation of our words from a training set of sentence bags-of-words $\{\mathbf{x}^{(t)}\}_{t=1}^{T}$.

We propose to achieve this by using an autoencoder model that encodes an input bag-of-words $\mathbf{x}$ with a sum of the representations (embeddings) of the words present in $\mathbf{x}$, followed by a non-linearity. Specifically, let matrix $\mathbf{W}$ be the $D \times V$ matrix whose columns are the vector representations for each word. The encoder's computation will involve summing over the columns of $\mathbf{W}$ for each word in the bag-of-word. We will denote this encoder function $\phi(\mathbf{x})$. Then, using a decoder, the autoencoder will be trained to optimize a loss function that measures how predictive of the original bag-of-words is the encoder representation $\phi(\mathbf{x})$.

There are different variations we can consider in the design of the encoder/decoder and the choice of loss function. One must be careful however, as certain choices can be inappropriate for training on word observations, which are intrinsically sparse and high-dimensional. In this work, we explore one such approach.

4.3 Binary bag-of-words reconstruction training with merged bags-of-words

We start from the conventional autoencoder architecture, which minimizes a cross-entropy loss that compares a binary vector observation with a decoder reconstruction. We thus convert the bag-of-words $\mathbf{x}$ into a fixed-size but sparse binary vector $\mathbf{v}(\mathbf{x})$, which is such that $v(\mathbf{x})_{x_i}$ is 1 if word x_i is present in $\mathbf{x}$ and otherwise 0.

From this representation, we obtain an encoder representation by multiplying $\mathbf{v}(\mathbf{x})$

with the word representation matrix $\mathbf{W}$

$$\mathbf{a}(\mathbf{x}) = \mathbf{c} + \mathbf{W}\mathbf{v}(\mathbf{x}), \;\; \phi(\mathbf{x}) = \mathbf{h}(\mathbf{a}(\mathbf{x})) \tag{4.1}$$

where $\mathbf{h}(\cdot)$ is an element-wise non-linearity such as the sigmoid or hyperbolic tangent, and $\mathbf{c}$ is a D-dimensional bias vector. Encoding thus involves summing the word representations of the words present at least once in the bag-of-words.

To produce a reconstruction, we parameterize the decoder using the following non-linear form:

$$\widehat{\mathbf{v}}(\mathbf{x}) = \mathrm{sigm}(\mathbf{V}\phi(\mathbf{x}) + \mathbf{b}) \tag{4.2}$$

where $\mathbf{V} = \mathbf{W}^T$, $\mathbf{b}$ is the bias vector of the reconstruction layer and $\mathrm{sigm}(a) = 1/(1 + \exp(-a))$ is the sigmoid non-linearity.

Then, the reconstruction is compared to the original binary bag-of-words as follows:

$$\ell(\mathbf{v}(\mathbf{x})) = -\sum_{i=1}^{V} v(\mathbf{x})_i \log(\widehat{v}(\mathbf{x})_i) + (1 - v(\mathbf{x})_i) \log(1 - \widehat{v}(\mathbf{x})_i) \; . \tag{4.3}$$

Training proceeds by optimizing the sum of reconstruction cross-entropies across the training set, *e.g.*, using stochastic or mini-batch gradient descent.

Note that, since the binary bags-of-words are very high-dimensional (the dimensionality corresponds to the size of the vocabulary, which is typically large), the above training procedure which aims at reconstructing the complete binary bag-of-word, will be slow. Since we will later be training on millions of sentences, training on each individual sentence bag-of-words will be expensive.

Thus, we propose a simple trick, which exploits the bag-of-words structure of the input. Assuming we are performing mini-batch training (where a mini-batch contains a list of the bags-of-words of adjacent sentences), we simply propose to merge the bags-of-words of the mini-batch into a single bag-of-words and perform an update based on that merged bag-of-words. The resulting effect is that each update is as efficient as in stochastic gradient descent, but the number of updates per training epoch is divided by the mini-batch size . As we'll see in the experimental section, this trick produces good word representations, while sufficiently reducing training time. We note that, additionally, we could have used the stochastic approach proposed by Dauphin *et al.* (2011) for

reconstructing binary bag-of-words representations of documents, to further improve the efficiency of training. They use importance sampling to avoid reconstructing the whole V-dimensional input vector.

4.4 Bilingual autoencoders

Let's now assume that for each sentence bag-of-words $\mathbf{x}$ in some source language $\mathcal{X}$, we have an associated bag-of-words $\mathbf{y}$ for this sentence translated in some target language $\mathcal{Y}$ by a human expert.

Assuming we have a training set of such $(\mathbf{x}, \mathbf{y})$ pairs, we'd like to use it to learn representations in both languages that are aligned, such that pairs of translated words have similar representations.

To achieve this, we propose to augment the regular autoencoder proposed in Section 4.3 so that, from the sentence representation in a given language, a reconstruction can be attempted of the original sentence in the other language. Specifically, we now define language specific word representation matrices $\mathbf{W}^x$ and $\mathbf{W}^y$, corresponding to the languages of the words in $\mathbf{x}$ and $\mathbf{y}$ respectively. Let $V^{\mathcal{X}}$ and $V^{\mathcal{Y}}$ also be the number of words in the vocabulary of both languages, which can be different. The word representations however are of the same size D in both languages. For the binary reconstruction autoencoder, the bag-of-words representations extracted by the encoder become

$$\phi(\mathbf{x}) = \mathbf{h}\left(\mathbf{c} + \mathbf{W}^{\mathcal{X}}\mathbf{v}(\mathbf{x})\right) \ , \ \ \phi(\mathbf{y}) = \mathbf{h}\left(\mathbf{c} + \mathbf{W}^{\mathcal{Y}}\mathbf{v}(\mathbf{y})\right) \tag{4.4}$$

Notice that we share the bias $\mathbf{c}$ before the non-linearity across encoders, to encourage the encoders in both languages to produce representations on the same scale.

From the sentence in either languages, we want to be able to perform a reconstruction of the original sentence in both the languages. In particular, given a representation in any language, we'd like a decoder that can perform a reconstruction in language $\mathcal{X}$ and another decoder that can reconstruct in language $\mathcal{Y}$. Again, we use decoders of the form proposed in Section 4.3 (see Figure 4.1), but let the decoders of each language have their own parameters $(\mathbf{b}^{\mathcal{X}}, \mathbf{V}^{\mathcal{X}})$ and $(\mathbf{b}^{\mathcal{Y}}, \mathbf{V}^{\mathcal{Y}})$.

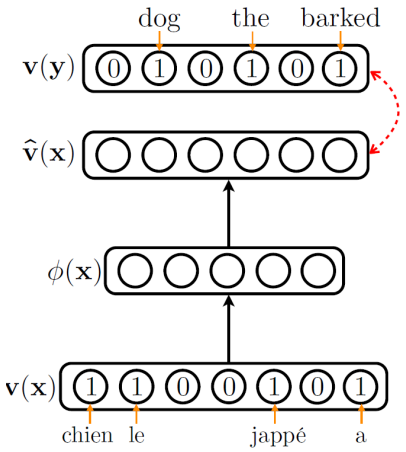

Figure 4.1: Bilingual autoencoder based on the binary reconstruction error. In this example, the model reconstruct the bag-of-words for the English sentence "*the dog barked*" from its French translation "*le chien a jappé*".

This encoder/decoder decomposition structure allows us to learn a mapping within each language and across the languages. Specifically, for a given pair $(\mathbf{x}, \mathbf{y})$, we can train the model to (1) construct $\mathbf{y}$ from $\mathbf{x}$ (loss $\ell(\mathbf{x}, \mathbf{y})$), (2) construct $\mathbf{x}$ from $\mathbf{y}$ (loss $\ell(\mathbf{y}, \mathbf{x})$), (3) reconstruct $\mathbf{x}$ from itself (loss $\ell(\mathbf{x})$) and (4) reconstruct $\mathbf{y}$ from itself (loss $\ell(\mathbf{y})$). We follow this approach in our experiments and optimize the sum of the corresponding 4 losses during training.

4.4.1 Joint reconstruction and cross-lingual correlation

Inspired by the transfer learning capabilities of Correlational Neural Networks, we also considered incorporating two additional terms to the loss function, in an attempt to favour even more meaningful bilingual representations:

$$\ell(\mathbf{x}, \mathbf{y}) + \ell(\mathbf{y}, \mathbf{x}) + \ell(\mathbf{x}) + \ell(\mathbf{y}) + \ell([\mathbf{x}, \mathbf{y}], [\mathbf{x}, \mathbf{y}]) - \lambda \cdot cor(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y})) \tag{4.5}$$

The term $\ell([\mathbf{x}, \mathbf{y}], [\mathbf{x}, \mathbf{y}])$ is simply a joint reconstruction term, where both languages are simultaneously presented as input and reconstructed. The second term $cor(\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y}))$ encourages correlation between the representation of each language. It is the sum of the scalar correlations between each pair $a(\mathbf{x})_k, a(\mathbf{y})_k$, across all dimensions k of the vectors $\mathbf{a}(\mathbf{x}), \mathbf{a}(\mathbf{y})$[1]. To obtain a stochastic estimate of the correlation, during training,

[1]While we could have applied the correlation term on $\phi(\mathbf{x}), \phi(\mathbf{y})$ directly, applying it to the pre-activation function vectors was found to be more numerically stable.

small mini-batches are used.

4.4.2 Document representations

Once we learn the language specific word representation matrices $\mathbf{W}^x$ and $\mathbf{W}^y$ as described above, we can use them to construct document representations, by using their columns as word vector representations. Given a document $\mathbf{d}$ written in language $\mathcal{Z} \in \{\mathcal{X}, \mathcal{Y}\}$ and containing m words, $z_1, z_2, \ldots, z_m$, we represent it as the tf-idf weighted sum of its words' representations $\psi(\mathbf{d}) = \sum_{i=1}^{m} \textit{tf-idf}(z_i) \cdot \mathbf{W}^{\mathcal{Z}}_{\cdot,z_i}$. We use the document representations thus obtained to train our document classifiers, in the cross-lingual document classification task described in Section 4.6.

4.5 Related Work

Recent work that has considered the problem of learning bilingual representations of words usually has relied on word-level alignments. Klementiev *et al.* (2012) propose to train simultaneously two neural network languages models, along with a regularization term that encourages pairs of frequently aligned words to have similar word embeddings. Thus, the use of this regularization term requires to first obtain word-level alignments from parallel corpora. Zou *et al.* (2013) use a similar approach, with a different form for the regularizer and neural network language models as in (Collobert *et al.*, 2011*b*). In our work, we specifically investigate whether a method that does not rely on word-level alignments can learn comparably useful multilingual embeddings in the context of document classification.

Looking more generally at neural networks that learn multilingual representations of words or phrases, we mention the work of Gao *et al.* (2014) which showed that a useful linear mapping between *separately trained* monolingual skip-gram language models could be learned. They too however rely on the specification of pairs of words in the two languages to align. Mikolov *et al.* (2013) also propose a method for training a neural network to learn useful representations of phrases, in the context of a phrase-based translation model. In this case, phrase-level alignments (usually extracted from word-level alignments) are required. Recently, Hermann and Blunsom (2014*a*,*b*) pro-

posed neural network architectures and a margin-based training objective that, as in this work, does not rely on word alignments. We will briefly discuss their work in the experiments section. A tree based bilingual autoencoder with similar objective function is also proposed in (Chandar A P *et al.*, 2014).

4.6 Experiments

The technique proposed in this work enables us to learn bilingual embeddings which capture cross-language similarity between words. We propose to evaluate the quality of these embeddings by using them for the task of cross-language document classification. We followed closely the setup used by Klementiev *et al.* (2012) and compare with their method, for which word representations are publicly available[2]. The set up is as follows. A labeled data set of documents in some language $\mathcal{X}$ is available to train a classifier, however we are interested in classifying documents in a different language $\mathcal{Y}$ at test time. To achieve this, we leverage some bilingual corpora, which is not labeled with any document-level categories. This bilingual corpora is used to learn document representations that are coherent between languages $\mathcal{X}$ and $\mathcal{Y}$. The hope is thus that we can successfully apply the classifier trained on document representations for language $\mathcal{X}$ directly to the document representations for language $\mathcal{Y}$. Following this setup, we performed experiments on 3 data sets of language pairs: English/German (EN/DE), English/French (EN/FR) and English/Spanish (EN/ES).

4.6.1 Data

For learning the bilingual embeddings, we used sections of the Europarl corpus (Koehn, 2005) which contains roughly 2 million parallel sentences. We considered 3 language pairs. We used the same pre-processing as used by Klementiev *et al.* (2012). We tokenized the sentences using NLTK (Bird Steven and Klein, 2009), removed punctuations and lowercased all words. We did not remove stopwords.

As for the labeled document classification data sets, they were extracted from sections of the Reuters RCV1/RCV2 corpora, again for the 3 pairs considered in our exper-

[2] `http://klementiev.org/data/distrib/`

iments. Following Klementiev *et al.* (2012), we consider only documents which were assigned exactly one of the 4 top level categories in the topic hierarchy (CCAT, ECAT, GCAT and MCAT). These documents are also pre-processed using a similar procedure as that used for the Europarl corpus. We used the same vocabularies as those used by Klementiev *et al.* (2012) (varying in size between $35,000$ and $50,000$).

For each pair of languages, our overall procedure for cross-language classification can be summarized as follows:

Train representation: Train bilingual word representations $\mathbf{W}^x$ and $\mathbf{W}^y$ on sentence pairs extracted from Europarl for languages $\mathcal{X}$ and $\mathcal{Y}$. We also use the monolingual documents from RCV1/RCV2 to reinforce the monolingual embeddings (this choice is cross-validated). These non-parallel documents can be used through the losses $\ell(\mathbf{x})$ and $\ell(\mathbf{y})$ (*i.e.* by reconstructing $\mathbf{x}$ from $\mathbf{x}$ or $\mathbf{y}$ from $\mathbf{y}$). Note that Klementiev *et al.* (2012) also used this data when training word representations.

Train classifier: Train document classifier on the Reuters training set for language $\mathcal{X}$, where documents are represented using the word representations $\mathbf{W}^x$ (see Section 4.4.2). As in Klementiev *et al.* (2012) we used an averaged perceptron trained for 10 epochs, for all the experiments.

Test-time classification: Use the classifier trained in the previous step on the Reuters test set for language $\mathcal{Y}$, using the word representations $\mathbf{W}^y$ to represent the documents.

We call our model[3] **BAE-cr** which stands for bilingual autoencoder based on cross-lingual correlation.

Models were trained for up to 20 epochs using the same data as described earlier. We used mini-batch (of size 20) stochastic gradient descent. All results are for word embeddings of size $D = 40$, as in Klementiev *et al.* (2012). Further, to speed up the training for BAE-cr we merged each 5 adjacent sentence pairs into a single training instance, as described in Section 4.3. For all language pairs, the joint reconstruction β was fixed to 1 and the cross-lingual correlation factor λ to 4. The other hyperparameters were tuned to each task using a training/validation set split of 80% and 20% and using the performance on the validation set of an averaged perceptron trained on the smaller training set portion (notice that this corresponds to a monolingual classification

[3]Our word representations and code are available at `http://www.sarathchandar.in/crl.html`

experiment, since the general assumption is that no labeled data is available in the test set language).

4.6.2 Comparison of the performance of different models

We now present the cross language classification results obtained by using the embeddings produced by our proposed model. We also compare our models with the following approaches:

BAE-tr: This model uses similar encoder-decoder architecture as in BAE-cr while the decoder is a tree based decoder (Chandar A P *et al.*, 2014). However, this model does not include the cross lingual correlation term. The correlation does not help in BAE-tr because of the nature of the architecture.

Klementiev et al.: This model uses word embeddings learned by a multitask neural network language model with a regularization term that encourages pairs of frequently aligned words to have similar word embeddings. From these embeddings, document representations are computed as described in Section 4.4.2.

MT: Here, test documents are translated to the language of the training documents using a standard phrase-based MT system, MOSES[4] which was trained using default parameters and a 5-gram language model on the Europarl corpus (same as the one used for inducing our bilingual embeddings).

Majority Class: Test documents are simply assigned the most frequent class in the training set.

For the EN/DE language pairs, we directly report the results from Klementiev *et al.* (2012). For the other pairs (not reported in Klementiev *et al.* (2012)), we used the embeddings available online and performed the classification experiment ourselves. Similarly, we generated the MT baseline ourselves.

Table 4.1 summarizes the results. They were obtained using 1000 RCV training examples. We report results in both directions, *i.e.* language $\mathcal{X}$ to $\mathcal{Y}$ and vice versa. The best performing method in all the pairs except one is BAE-cr. In particular, BAE-cr often outperforms the approach of Klementiev *et al.* (2012) by a large margin.

[4] `http://www.statmt.org/moses/`

We also mention the recent work of Hermann and Blunsom (2014*b*), who proposed two neural network architectures for learning word and document representations using sentence-aligned data only. Instead of an autoencoder paradigm, they propose a margin-based objective that aims to make the representation of aligned sentences closer than non-aligned sentences. While their trained embeddings are not publicly available, they report results for the EN/DE classification experiments, with representations of the same size as here ($D = 40$) and trained on 500K EN/DE sentence pairs. Their best model in that setting reaches accuracies of 83.7% and 71.4% respectively for the EN $\rightarrow$ DE and DE $\rightarrow$ EN tasks. One clear advantage of our model is that unlike their model, it can use additional monolingual data. Indeed, when we train BAE-cr with 500k EN/DE sentence pairs, plus monolingual RCV documents (which come at no additional cost), we get accuracies of 87.9% (EN $\rightarrow$ DE) and 76.7% (DE $\rightarrow$ EN), still improving on their best model. If we do not use the monolingual data, BAE-cr's performance is worse but still competitive at 86.1% for EN $\rightarrow$ DE and 68.8% for DE $\rightarrow$ EN. Finally, without constraining D to 40 (they use 128) and by using additional French data, the best results of Hermann and Blunsom (2014*b*) are 88.1% (EN $\rightarrow$ DE) and 79.1% (DE $\rightarrow$ EN), the latter being, to our knowledge, the current state-of-the-art.

Table 4.1: Cross-lingual classification accuracy for 3 language pairs, with 1000 labeled examples.

	EN $\rightarrow$ DE	DE $\rightarrow$ EN	EN $\rightarrow$ FR	FR $\rightarrow$ EN	EN $\rightarrow$ ES	ES $\rightarrow$ EN
BAE-cr	**91.8**	**74.2**	**84.6**	**74.2**	49.0	**64.4**
BAE-tr	81.8	60.1	70.4	61.8	**59.4**	60.4
Klementiev et al.	77.6	71.1	74.5	61.9	31.3	63.0
MT	68.1	67.4	76.3	71.1	52.0	58.4
Majority Class	46.8	46.8	22.5	25.0	15.3	22.2

We also evaluate the effect of varying the amount of supervised training data for training the classifier. For brevity, we report only the results for the EN/DE pair, which are summarized in Figure 4.2 and Figure 4.3. We observe that BAE-cr clearly outperforms the other models at almost all data sizes. More importantly, it performs remarkably well at very low data sizes (100), suggesting it learns very meaningful embeddings, though the method can still benefit from more labeled data (as in the DE $\rightarrow$ EN case).

Table 4.2: Example English words along with 8 closest words both in English (en) and German (de), using the Euclidean distance between the embeddings learned by BAE-cr

january		president		said	
en	de	en	de	en	de
january	januar	president	präsident	said	gesagt
march	märz	i	präsidentin	told	sagte
october	oktober	mr	präsidenten	say	sehr
july	juli	presidents	herr	believe	heute
december	dezember	thank	ich	saying	sagen
1999	jahres	president-in-office	ratspräsident	wish	heutigen
june	juni	report	danken	shall	letzte
month	1999	voted	danke	again	hier
oil		**microsoft**		**market**	
en	de	en	de	en	de
oil	öl	microsoft	microsoft	market	markt
supply	boden	cds	cds	markets	marktes
supplies	befindet	insider	warner	single	märkte
gas	gerät	ibm	tageszeitungen	commercial	binnenmarkt
fuel	erdöl	acquisitions	ibm	competition	märkten
mineral	infolge	shareholding	handelskammer	competitive	handel
petroleum	abhängig	warner	exchange	business	öffnung
crude	folge	online	veranstalter	goods	binnenmarktes

Table 4.2 also illustrates the properties captured within and across languages, for the EN/DE pair. For a few English words, the words with closest word representations (in Euclidean distance) are shown, for both English and German. We observe that words that form a translation pair are close, but also that close words within a language are syntactically/semantically similar as well.

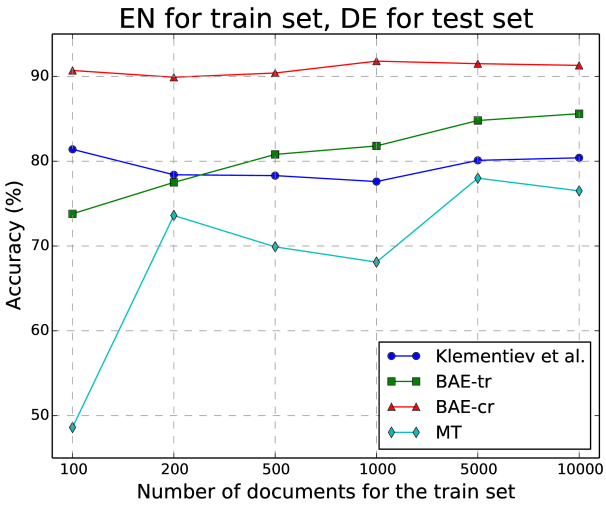

Figure 4.2: Cross-lingual classification accuracy results for EN → DE

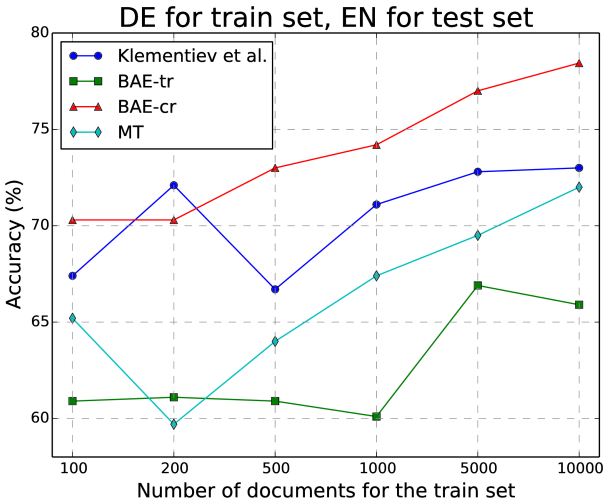

Figure 4.3: Cross-lingual classification accuracy results for DE → EN

The excellent performance of BAE-cr suggests that merging several sentences into single bags-of-words can still yield good word embeddings. In other words, not only we do not need to rely on word-level alignments, but exact sentence-level alignment is also not essential to reach good performances. We experimented with the merging of 5, 25 and 50 adjacent sentences into a single bag-of-words. Results are shown in Table 4.3. They suggest that merging several sentences into single bags-of-words does not necessarily impact the quality of the word embeddings. Thus they confirm that exact sentence-level alignment is not essential to reach good performances as well.

Table 4.3: Cross-lingual classification accuracy for 3 different pairs of languages, when merging the bag-of-words for different numbers of sentences. These results are based on 1000 labeled examples.

	# sent.	EN → DE	DE → EN	EN → FR	FR → EN	EN → ES	ES → EN
BAE-cr	5	91.75	72.78	84.64	74.2	49.02	64.4
	25	88.0	64.5	78.1	70.02	68.3	54.68
	50	90.2	49.2	82.44	75.5	38.2	67.38

4.7 Conclusion

We presented evidence that meaningful bilingual word representations could be learned without relying on word-level alignments or using fairly coarse sentence-level alignments. In particular, we showed that even though our model does not use word level alignments, it is able to reach state-of-the-art performance, even compared to a method that exploits word-level alignments. In addition, it also outperforms a strong machine translation baseline.

For future work, we would like to explore the possibility of conversion of our bilingual model to a multilingual model, which can learn common representations for multiple languages, given different amounts of parallel data between these languages. We would also like to investigate extensions of our bag-of-words bilingual autoencoder to bags-of-n-grams, where the model would also have to learn representations for short phrases. Such a model should be particularly useful in the context of a machine translation system.

CHAPTER 5

DEEP CORRELATIONAL NEURAL NETWORKS

In this chapter we will discuss how to extend Correlational Neural Networks to Deep Correlational Neural Networks.

5.1 Motivation

Correlational Neural Networks are capable of learning a common representation for two views given enough parallel data. In the last two chapters, we experimentally verified that the representations learnt by CorrNets help transfer learning algorithms perform better. However, the views that we considered in all our experiments are not so skewed. Consider multimodal learning, where we are interested in learning common representation for images and text. Now, the image view is too dense while the text view is too sparse. Learning a common representation such that we can reproduce the image view given the text view is harder than our previously discussed scenarios. Learning a deep network might help us to learn this highly skewed mapping.

Another motivation to design deep networks is the nature of the problem. Some problems might have a highly non-linear dependency across the views and a simple three layer network might fail to capture such non-linear dependencies. Learning a deep network will help us to capture such non-linear dependencies. In this chapter, we will discuss two ways to construct deep Correlational Neural Networks. The first way, as proposed in (Chandar A P *et al.*, 2013) is really not a deep CorrNet. It employs a shallow CorrNet on deep representations. We call such methods as Quasi deep methods. In this thesis, we propose another way to make CorrNets deep. It essentially involves stacking CorrNets as pretraining.

5.2 Quasi Deep Correlational Neural Networks

Our goal is to learn deep common representations for two views. In this section, we will see a CorrNet which does this by learning common representations for two views projected into a deep representation. We call such networks Quasi-deep networks.

We will illustrate Quasi-deep CorrNets with cross language learning as an application. In this quasi-deep approach, we will learn language specific deep representations and then use it to learn a common representation. Now the classifier can be trained and tested in the common space. The entire procedure is described below.

1. Language specific deep representation: In this phase, we are interested in obtaining a language specific representation (p_i or q_i) for an entity in L_1 or L_2 respectively. If the entity is a document this representation can be as simple as a set of binary features indicating the presence or absence of a n-gram in the vocabulary of the language. Alternatively, if the entity is a word then each feature could indicate the presence or absence of any n-gram character in the language or some such suitable representation. We train a k-layered stacked auto-encoder to learn an abstract representation (Figure 5.1) for a given entity using its raw representation (n-gram words, n-gram characters, co-occurrence vectors, etc.).

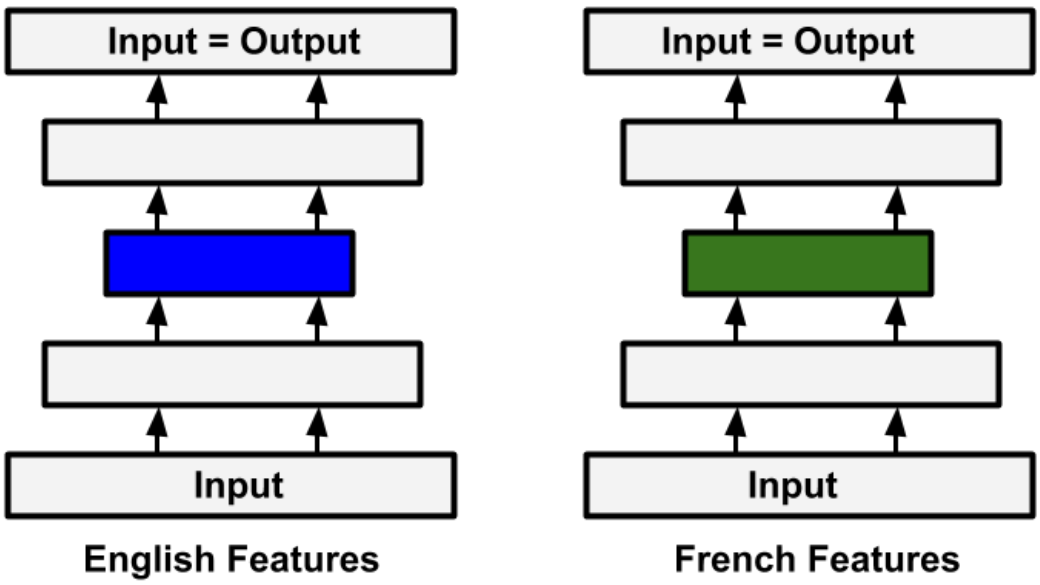

Figure 5.1: Language Specific Representation

2. Shared Representation Learning(SRL): For the next phase we need a pair of

parallel entities in L_1 and L_2 wherein the representation p_i (or q_i) of an entity in L_1 (or L_2) is obtained using unsupervised feature learning as described above. A sample $Z = \{(p_i, q_i)\}_{i=1}^{n}$ of such parallel entities is then passed to the CorrNet to learn a shared representation. This process is illustrated in Figure 5.2.

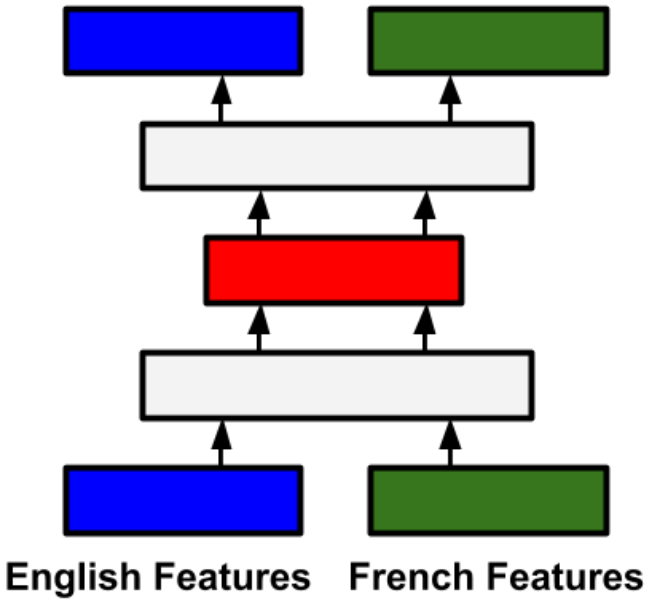

Figure 5.2: Shared Representation Learning

3. Source Language Training: Now we come to the crux of cross language training where the aim is to train a model using the data available in L_1 and apply this model to data from L_2. Lets assume we have a sample $D = \{x_i, y_i\}_{i=1}^{k}$ of training data available in L_1 where x_i is the input and y_i is the label. For each x_i we first learn the abstract representation p_i in L_1 using the auto-encoder in phase 1. Next for each p_i we obtain the compact representation $f((p_i, \mathbf{0})) = f(z_i^1)$ using the CorrNet trained in phase 2. Effectively, we have projected the original input x_i to a space in which entities from L_2 can also be represented. Thus, a model trained using this projected data $\{f(z_i^1)\}_{i=1}^{n}$ can be applied to entities belonging to L_2 after projecting them to this space. This process of training is illustrated in Figure 5.3.

4. Target Language Testing: Finally, the model trained above is applied to test data from L_2 by first projecting it to the common space as illustrated in Figure5.4. For each x_i, we first learn the abstract representation q_i in L_2 using the auto-encoder in phase 1. Next for each q_i we obtain the compact representation $f((\mathbf{0}, q_i)) = f(z_i^2)$ using the CorrNet trained in phase 2. Now use the classifier to classify the test instance.

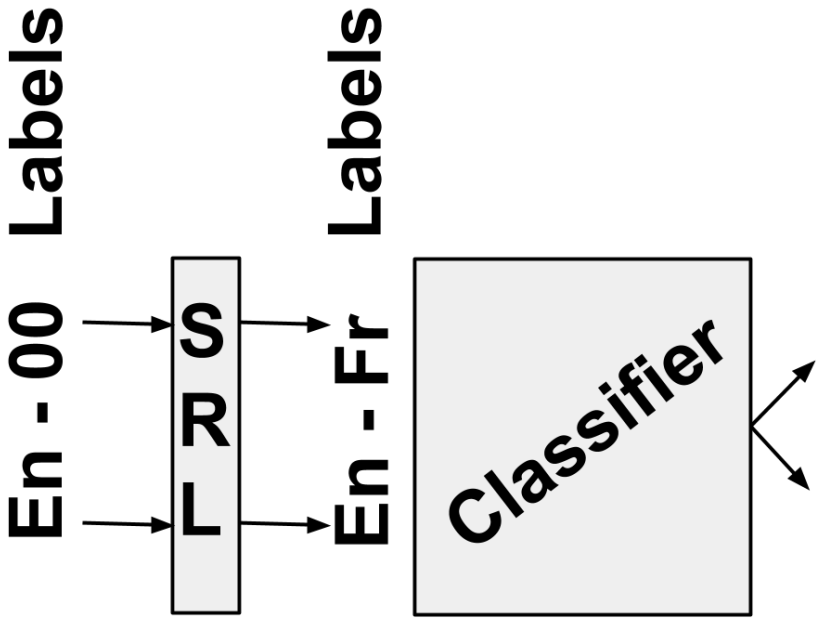

Figure 5.3: Source Language Training

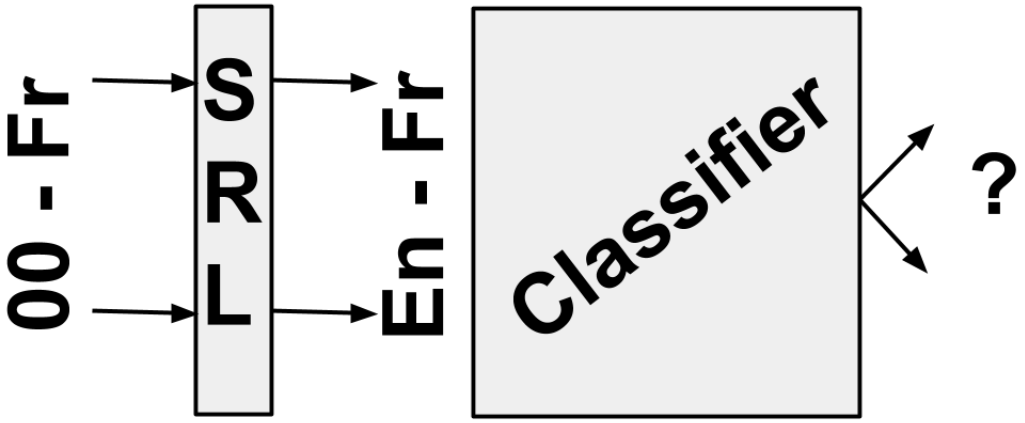

Figure 5.4: Target Language Testing

5.3 Cross Language Sentiment Learning

We evaluate the performance of the proposed framework on the task of Cross Language Sentiment Analysis where the goal is to detect the sentiment polarity (positive or negative) of a document in language L_2 using training data available in language L_1. For the purpose of this evaluation, we created a Multilingual Dataset for Sentiment Analysis similar to the Multi-domain dataset used in Blitzer *et al.* (2007). Specifically, we

collected reviews for English and French DVDs from amazon [1]. These reviews are accompanied with a reviewer rating on a scale of 1 to 5 (5 indicating excellent and 1 indicating poor). We considered reviews with ratings 4 and 5 to be positive and reviews with ratings 1 and 2 to be negative. The details of the dataset are provided in Table 5.1.

Language	Training instances			Test instances		
	Positive	Negative	Neutral	Positive	Negative	Neutral
English	20000	20000	10000	2000	2000	-
French	20000	20000	10000	2000	2000	-

Table 5.1: Multilingual Sentiment Dataset Description

The dataset will be made publicly available and will hopefully help in furthering the research on multilingual sentiment analysis.

5.3.1 Experimental Setup

As mentioned in section 5.2, our approach has four phases (i) language specific representation (ii) shared representation learning (iii) task specific supervised training and (iv) cross language testing. We describe the procedure followed for executing each of these phases.

For monolingual deep learning, we used 50,000 training documents for each language from the Multilingual Dataset. Note that we do not use parallel corpora in this phase. We chose an arbitrary set of documents in each language drawn from the same domain as the test documents. This is to ensure that there is a strong overlap in the vocabulary of the corpus used in phases 1 and 2. We used a five layered stacked auto-encoder for this phase. To feed the first layer of the auto-encoder we converted the documents to a feature vector comprising of the top 40,000 unigrams in the vocabulary. In subsequent layers of the stacked auto-encoder, we reduced the number of hidden neurons from 10,000 to 5000 to 2500 to 500. The representation from the last layer consisting of 500 hidden neurons is used as the deep representation of a given document.

[1] www.amazon.com and www.amazon.fr

Next, for the second phase, we need English-French parallel documents. For this, we translated the 50,000 English documents used above to French using a state of the art Machine Translation Tool (Al-Onaizan and Papineni, 2006) thus creating a English French parallel corpus. We then obtain the language specific deep representations for each of these documents and then use these parallel deep representations to train the predictive auto-encoder. In the third phase, we need to train a sentiment classifier using 40K English training data. Instead of using a simple unigram based feature representation for the documents we first obtain the language specific deep representation of these documents and then project this representation into the common space using the predictive auto-encoder. We then train a classifier using this shared representation of the English documents as the feature vector. Finally, we take the test documents from French, repeat the above process of projecting them to the common space and then feed them to the trained model for inference.

We compared the above approach with some standard approaches for CLSA. The empirical upper bound for the performance is obtained by using a classifier trained on French training data. In addition, we consider two baselines: (i) a classifier trained using English data and tested on French data after translating it to English using a MT system and (ii) a classifier trained after translating the training instances into French and then tested on French instances. For the two baseline approaches we use a ungiram feature representation. For all the methods we use SVM (Chang and Lin, 2011) as the classifier. The accuracy of the different approaches are reported in Table 5.2.

S.No	Approach	Train data	Test data	Accuracy
1	Self Training	Fr	Fr	86.1%
2	Translate and Train	En - trans. to Fr	Fr	63.4%
3	Translate and Test	En	Fr - trans. to En	65.15%
4	Common Representation Learning	En	Fr	72%

Table 5.2: Accuracy of different approaches for Cross Language Sentiment Analysis

5.4 Deep Correlational Neural Networks

As mentioned previously, quasi-deep CorrNets are actually shallow. In Quasi-deep approach, we learn deep representation for each view separately and use it along with a shallow CorrNet to learn a common representation. However, feeding non-linear deep representations to a shallow CorrNet makes it harder to train the CorrNet. In this section we propose Deep Correlational Neural Netork with an efficient pre-training procedure.

We use the following procedure to train a Deep CorrNet.

1. Train a shallow CorrNet with the given data (see step-1 in Figure 5.5). At the end of this step, we have learned the parameters $\mathbf{W}$, $\mathbf{V}$ and $\mathbf{b}$.
2. Modify the CorrNet model such that the first input view connects to a hidden layer using weights $\mathbf{W}$ and bias $\mathbf{b}$. Similarly connect the second view to a hidden layer using weights $\mathbf{V}$ and bias $\mathbf{b}$. We have now decoupled the common hidden layer for each view (see step-2 in Figure 5.5).
3. Add a new common hidden layer which takes its input from the hidden layers created at step 2. We now have a CorrNet which is one layer deeper (see step-3 in Figure 5.5).
4. Train the new Deep CorrNet on the same data.
5. Repeat steps 2, 3 and 4, for as many hidden layers as required.

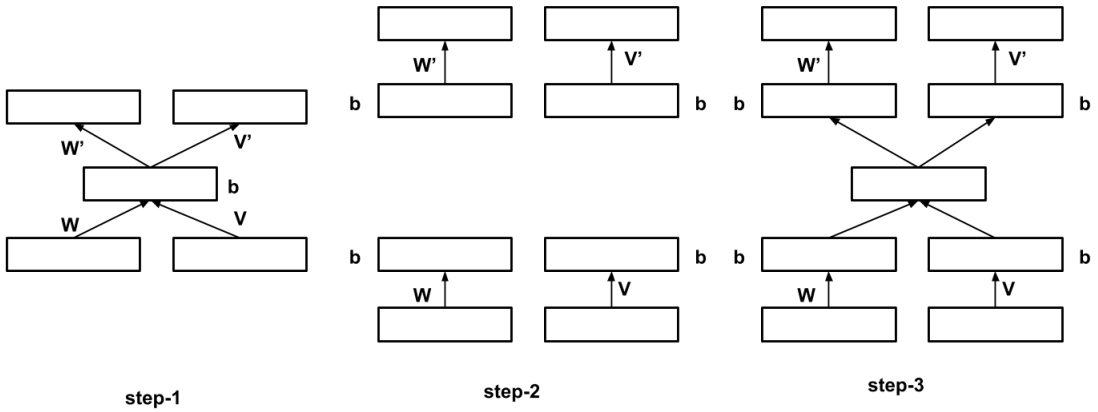

Figure 5.5: Stacking CorrNet to create Deep Correlartional Neural Network.

We would like to point out that we could have followed the procedure described in Ngiam *et al.* (2011). However, we chose not to use the deep training procedure described in Ngiam *et al.* (2011) since the objective function used by them during pre-training and training is different. Specifically, during pre-training the objective is to minimize self reconstruction error whereas during training the objective is to minimize

both self and cross reconstruction error. In contrast, in the stacking procedure outlined above, the objectives during training and pre-training are aligned.

5.5 Experiments using Deep Correlational Neural Network

In this section, we evaluate the performance of the deep extension of CorrNet. Having already compared with MAE in the previous section, we focus our evaluation here on a comparison with DCCA (Andrew *et al.*, 2013). All the models were trained using 10000 images from the MNIST training dataset and we computed the sum correlation and transfer learning accuracy for each of these models. For transfer learning, we use the linear SVM implementation provided by (Pedregosa *et al.*, 2011) for all our experiments and do 5-fold cross validation using 10000 test images from MNIST data. We report results for two settings (i) Left to Right (training on left view, testing on right view) and (ii) Right to Left (training on right view, testing on left view). These results are summarized in Table 5.3. In this Table, model-x-y means a model with x units in the first hidden layer and y units in second hidden layer. For example, CorrNet-500-300-50 is a Deep CorrNet with three hidden layers containing 500, 300 and 50 units respectively. The third layer containing 50 units is used as the common representation.

Model	Sum Correlation	Left to Right	Right to Left
CorrNet-500-50	**47.21**	77.68	77.95
DCCA-500-50	33.00	66.41	64.65
CorrNet-500-300-50	45.634	**80.46**	**80.47**
DCCA-500-500-50	33.77	70.06	72.43

Table 5.3: Comparison of sum correlation and transfer learning performance of different deep models

Both the Deep CorrNets (CorrNet-500-50 and CorrNet-500-300-50) clearly perform better than the corresponding DCCA. We notice that for both the transfer learning tasks, the 3-layered CorrNet (CorrNet-500-300-50) performs better than the 2-layered CorrNet (CorrNet-500-50) but the sum correlation of the 2-layered CorrNet is better than that of the 3-layered CorrNet.

5.6 Summary

In this chapter, we proposed two ways to make CorrNets deep. The first way learns view specific deep represenations and use them to train a shallow CorrNet. We call such models quasi-deep models. The second way actually learns a deep CorrNet. Both models have their own advantages. Quasi-deep models can be used when there is abundant single view data and fewer two view data. Deep CorrNets are preferable when there is abundant two view data.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This thesis considers the problem of learning common representation for multiple views which has several applications in transfer learning, multi-view learning and multi modal learning. To solve this problem, we propose Correlational Neural Networks, a class of Neural Networks that can learn common representation for two views. CorrNets can be considered as the representation learning tool for multi-view data in the similar way how Auto-encoders are representation learning tool for single view data. Experimental results prove that CorrNet performs better than several other state-of-the-art CRL algorithms like CCA, KCCA, MAE, and DCCA. We also notice that shallow CorrNet itself is more powerful than deep models like DCCA.

CorrNet has been applied for several cross language learning tasks. We considered transfer learning tasks like Cross Language Document Classification (CLDC), and Cross Language Sentiment Analysis (CLSA) and equivalence tasks like matching equivalent items across views (transliteration equivalence). In all these tasks, CorrNet performs significantly better than the previous state-of-the-art methods. Applying CorrNet to several lanaguage pairs like English/German, English/French, English/Spanish, English/Hindi, we also demonstrated the language independent nature of the model. The thesis also outlines two ways to make CorrNet deep, with each approach having its own advantages and disadvantages.

This thesis introduces a new class of CRL algorithms which we believe, has a huge potential for future work. We briefly mention them below.

Multiple views: Even though we claim that CorrNets are easily extensible to multiple views, this thesis does not talk about extending CorrNet to more than two views. One way to extend CorrNets to multiple views has been studied in (Rongali *et al.*, 2015). However, an even more elegant way of extending CorrNet would be to have encoder and decoder for each view and pair them up dynamically based on the training data.

Correlational RBMs: This thesis describes an autoencoder version of Correlational Neural Network. However, the approach is more general. The goal is to learn common

hidden representation such that the views are correlated in the common subspace. We can also design Correlational RBMs where we have two RBMs (one for each view) and train them in such a way that their hidden units are correlated.

Sparse Correlational Neural Networks: Sparse CCA has been widely applied in several biological applications. CorrNet can also be extended to Sparse CorrNets which can be applied in all such biological applications. Making CorrNet sparse can be easily achieved by adding L1 regularization for encoder matrices. However, we do not know if that would hinder the performance of the model, which needs further study.

Multimodal Learning: Multimodal learning, or learning from multiple modalities like text, image, and video is gaining significant attention because of various potential applications. CCA is a competitive approach in Multimodal Learning as well. We can also apply CorrNets to learn common representation for multiple modalities. Such a representation can be used for multimodal information retrieval or predicting missing modalities.

Better pretraining procedure: In this thesis, we have proposed a pretraining procedure for Deep CorrNet which has similar objective as the training procedure. We claimed that this resulted in better performance when compared to models where pretraining and training are not aligned. However, this pretraining is costly since the number of correlation terms in the objective function will be very high during the pretraining stage. One interesting future work is to design relatively cheaper pretraining methods with similar performance.

In conclusion, the thesis proposes a potential and promising new architecture, CorrNet, for common representation learning, and the strength of the model was demonstrated by application on several cross language tasks.

REFERENCES

1. **Akaho, S.**, A kernel method for canonical correlation analysis. *In In Proceedings of the International Meeting of the Psychometric Society (IMPS2001)*. Springer-Verlag, 2001.

2. **Al-Onaizan, Y.** and **K. Papineni**, Distortion models for statistical machine translation. *In Proceedings of ACL*, ACL-44. Association for Computational Linguistics, Morristown, NJ, USA, 2006. URL `http://dx.doi.org/10.3115/1220175.1220242`.

3. **Andrew, G.**, **R. Arora**, **K. Livescu**, and **J. Bilmes**, Deep canonical correlation analysis. *In International Conference on Machine Learning (ICML)*. Atlanta, Georgia, 2013.

4. **Arora, R.** and **K. Livescu**, Kernel CCA for multi-view learning of acoustic features using articulatory measurements. *In 2012 Symposium on Machine Learning in Speech and Language Processing, MLSLP 2012, Portland, Oregon, USA, September 14, 2012*. 2012.

5. **Bengio, Y.**, **A. Courville**, and **P. Vincent**, Unsupervised feature learning and deep learning: A review and new perspectives. *In IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Special Issue*. 2013.

6. **Bengio, Y.**, **P. Lamblin**, **D. Popovici**, and **H. Larochelle**, Greedy layer-wise training of deep networks. *In* **B. Schölkopf**, **J. Platt**, and **T. Hoffman** (eds.), *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

7. **Bergstra, J.**, **O. Breuleux**, **F. Bastien**, **P. Lamblin**, **R. Pascanu**, **G. Desjardins**, **J. Turian**, **D. WardeFarley**, and **Y. Bengio** (2010). Theano: a cpu and gpu math expression compiler. *In Proceedings of the Python for Scientific Computing Conference (SciPy).*.

8. **Bird Steven, E. L.** and **E. Klein**, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

9. **Blitzer, J.**, **M. Dredze**, and **F. Pereira**, Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *In Association for Computational Linguistics*. Prague, Czech Republic, 2007.

10. **Chandar A P, S.**, **M. M. Khapra**, **B. Ravindran**, **V. C. Raykar**, and **A. Saha**, Multilingual deep learning. *In NIPS Deep Learning Workshop*. 2013.

11. **Chandar A P, S.**, **S. Lauly**, **H. Larochelle**, **M. Khapra**, **B. Ravindran**, **V. C. Raykar**, and **A. Saha**, An autoencoder approach to learning bilingual word representations. *In* **Z. Ghahramani**, **M. Welling**, **C. Cortes**, **N. Lawrence**, and **K. Weinberger** (eds.), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

12. **Chang, C.-C.** and **C.-J. Lin** (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

13. **Collobert, R.**, **K. Kavukcuoglu**, and **C. Farabet** (2011*a*). A matlab-like environment for machine learning. *In BigLearn, NIPS Workshop, number EPFL-CONF-192376*.

14. **Collobert, R.**, **J. Weston**, **L. Bottou**, **M. Karlen**, **K. Kavukcuoglu**, and **P. Kuksa** (2011*b*). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, **12**, 2493–2537.

15. **Das, D.** and **S. Petrov**, Unsupervised part-of-speech tagging with bilingual graph-based projections. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, 2011.

16. **Dauphin, Y.**, **X. Glorot**, and **Y. Bengio**, Large-Scale Learning of Embeddings with Reconstruction Sampling. *In Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*. Omnipress, 2011.

17. **Dauphin, Y. N.**, **R. Pascanu**, **C. Gulcehre**, **K. Cho**, **S. Ganguli**, and **Y. Bengio**, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *In* **Z. Ghahramani**, **M. Welling**, **C. Cortes**, **N. Lawrence**, and **K. Weinberger** (eds.), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

18. **Dhillon, P. S.**, **D. Foster**, and **L. Ungar**, Multi-view learning of word embeddings via cca. *In Advances in Neural Information Processing Systems (NIPS)*, volume 24. 2011.

19. **Duchi, J.**, **E. Hazan**, and **Y. Singer** (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**, 2121–2159. ISSN 1532-4435.

20. **Fisher, R.** (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, 179–188.

21. **Gao, J.**, **X. He**, **W.-t. Yih**, and **L. Deng**, Learning continuous phrase representations for translation modeling. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, 2014.

22. **Hannun, A.**, **C. Case**, **J. Casper**, **B. Catanzaro**, **G. Diamos**, **E. Elsen**, **R. Prenger**, **S. Satheesh**, **S. Sengupta**, **A. Coates**, and **A. Y. Ng**, Deep speech: Scaling up end-to-end speech recognition. *In Arxiv*. 2014.

23. **Hermann, K. M.** and **P. Blunsom**, Multilingual Distributed Representations without Word Alignment. *In Proceedings of International Conference on Learning Representations (ICLR)*. 2014*a*.

24. **Hermann, K. M.** and **P. Blunsom**, Multilingual models for compositional distributed semantics. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. 2014*b*.

25. **Hinton, G. E.** (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, **14**(8), 171–180.

26. **Hinton, G. E.**, **S. Osindero**, and **Y. W. Teh** (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**(7), 1527–1554.

27. **Hornik, K.** (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**(2), 251–257.

28. **Hotelling, H.** (1936). Relations between two sets of variates. *Biometrika*, **28**, 321 – 377.

29. **Hsieh, W.** (2000). Nonlinear canonical correlation analysis by neural networks. *Neural Networks*, **13**(10), 1095 – 1105.

30. **Ioffe, S.** and **C. Szegedy** (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, **abs/1502.03167**. URL `http://arxiv.org/abs/1502.03167`.

31. **King, D. E.** (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, **10**, 1755–1758.

32. **Klementiev, A.**, **I. Titov**, and **B. Bhattarai**, Inducing Crosslingual Distributed Representations of Words. *In Proceedings of the International Conference on Computational Linguistics (COLING)*. 2012.

33. **Koehn, P.**, Europarl: A parallel corpus for statistical machine translation. *In MT Summit*. 2005.

34. **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton**, Imagenet classification with deep convolutional neural networks. *In NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada*. 2012.

35. **Kumaran, A.**, **M. M. Khapra**, and **H. Li**, Report of news 2010 transliteration mining shared task. *In Proceedings of the 2010 Named Entities Workshop*. Uppsala, Sweden, 2010.

36. **Li, H.**, **A. Kumaran**, **M. Zhang**, and **V. Pervouvhine**, Whitepaper of news 2009 machine transliteration shared task. *In Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*. Suntec, Singapore, 2009.

37. **Liu, B.**, *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

38. **Lu, Y.** and **D. P. Foster**, large scale canonical correlation analysis with iterative least squares. *In* **Z. Ghahramani**, **M. Welling**, **C. Cortes**, **N. Lawrence**, and **K. Weinberger** (eds.), *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014.

39. **Luo, Y.**, **D. Tao**, **Y. Wen**, **K. Ramamohanarao**, and **C. Xu**, Tensor canonical correlation analysis for multi-view dimension reduction. *In In Arxiv*. 2015.

40. **Martens, J.**, Deep learning via hessian-free optimization. *In Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. 2010.

41. **Mihalcea, R.**, **C. Banea**, and **J. Wiebe**, Learning multilingual subjective language via cross-lingual projections. *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, 2007. URL `http://www.aclweb.org/anthology/P07-1123`.

42. **Mika, S.**, **G. Ratsch**, **J. Weston**, **B. Scholkopf**, and **K. R. Mullers** (1999). Fisher discriminant analysis with kernels. *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, 41–48. URL `http://dx.doi.org/10.1109/nnsp.1999.788121`.

43. **Mikolov, T.**, **Q. Le**, and **I. Sutskever** (2013). Exploiting Similarities among Languages for Machine Translation. Technical report, arXiv.

44. **Minsky, M. L.** and **S. A. Papert** (1969). Perceptrons. *Cambridge, MA: MIT Press.*.

45. **Nesterov, Y.** (1983). A method of solving a convex programming problem with convergence rate o(1/sqr(k)). *Soviet Mathematics Doklady*, **27**, 372–376.

46. **Ngiam, J.**, **A. Khosla**, **M. Kim**, **J. Nam**, **H. Lee**, and **A. Y. Ng**, Multimodal deep learning. *In International Conference on Machine Learning (ICML)*. Bellevue, USA, 2011.

47. **Padó, S.** and **M. Lapata** (2009). Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research (JAIR)*, **36**, 307–340.

48. **Pearson, K.** (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, **2**, 559–572.

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

50. **Polyak, B.** (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, **4**(5), 1–17.

51. **Rongali, S.**, **S. Chandar A P**, and **B. Ravindran**, From multiple views to single view : A neural network approach. *In In the Proceedings of the Second ACM-IKDD Conference on Data Sciences (IKDD CoDS) 2015*. 2015.

52. **Rosenblatt, F.** (1957). The perceptron–a perceiving and recognizing automaton. *Report 85-460-1, Cornell Aeronautical Laboratory.*.

53. **Rumelhart, D. E.**, **G. E. Hinton**, and **R. J. Williams** (1986). Learning representations by backpropagating errors. *Nature*, **323**.

54. **Scholkopf, B.**, **A. Smola**, and **K.-R. Müller** (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10**(5), 1299–1319.

55. **Smolensky, P.** (1986). Information processing in dynamical systems: Foundations of harmony theory. *In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing, volume 1, chapter 6*, 194–281.

56. **Socher, R.**, **J. Bauer**, **C. D. Manning**, and **N. Andrew Y.**, Parsing with compositional vector grammars. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 2013.

57. **Tenenhaus, A.** and **M. Tenenhaus** (2011). Regularized generalized canonical correlation analysis. *Psychometrika*, **76**(2), 257–284.

58. **Tieleman, T.** and **G. Hinton** (2012). Lecture 6.5- rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

59. **Toutanova, K.**, **D. Klein**, **C. D. Manning**, and **Y. Singer**, Feature-rich part-of-speech tagging with a cyclic dependency network. *In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03. 2003.

60. **Turing, A. M.**, Computing machinery and intelligence. *In Mind*, volume 49. 1950.

61. **Udupa, R.** and **M. M. Khapra**, Transliteration equivalence using canonical correlation analysis. *In Proceedings of the 32nd European Conference on IR Research*. 2010.

62. **Wan, X.**, Co-training for cross-lingual sentiment classification. *In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore, 2009.

63. **Yarowsky, D.** and **G. Ngai**, Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. *In Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Pittsburgh, Pennsylvania, 2001.

64. **Zeiler, M. D.**, Adadelta: An adaptive learning rate method. *In in Arxiv*. 2012.

65. **Zhang, X.** and **Y. LeCun**, Text understanding from scratch. *In Arxiv*. 2015.

66. **Zou, W. Y.**, **R. Socher**, **D. Cer**, and **C. D. Manning**, Bilingual Word Embeddings for Phrase-Based Machine Translation. *In Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*. 2013.

LIST OF PAPERS BASED ON THESIS

1. Sarath Chandar A P, Mitesh M Khapra, Hugo Larochelle, Balaraman Ravindran, *"Correlational Neural Networks"*. Submitted to Neural Computation. In Arxiv `http://arxiv.org/abs/1504.07225`.

2. Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, Amrita Saha, *"An Autoencoder Approach to Learning Bilingual Word Representations"*. In the proceedings of the Neural Information Processing Systems 27, 2014.

3. Sarath Chandar A P, Mitesh M Khapra, Ravindran B, Vikas Raykar, Amrita Saha, *"Multilingual Deep Learning"*. In Deep Learning Workshop at NIPS 2013.

4. Subendhu Rongali, Sarath Chandar A P, Ravindran B, *"From Multiple Views to Single View : A Neural Network Approach"*. In the Proceedings of the Second ACM-IKDD Conference on Data Sciences (IKDD CoDS) 2015.

CURRICULUM VITAE

Name	Sarath Chandar A P
Date of Birth	28 January 1991
E-Mail	apsarathchandar@gmail.com
Website	www.sarathchandar.in
Permanent Address	52, Bharathiar Street, Batlagundu, Dindigul Dt., Tamilnadu - 624202.
Education	M.S. by Research (CSE), IIT Madras B.E (CSE), Sri Venkateswara College of Engineering (2012)

GENERAL TEST COMITTEE

Chairman Dr. Sukhendu Das
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras.

Guide Dr. Balaraman Ravindran
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras.

Members Dr. John Augustine
Department of Computer Science and Engineering,
Indian Institute of Technology, Madras.

Dr. Andrew Thangaraj
Department of Electrical Engineering,
Indian Institute of Technology, Madras.