Deep Active Learning Explored Across Diverse Label Spaces

by

Hiranmayi Ranganathan

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2018 by the
Graduate Supervisory Committee:

Sethuraman Panchanathan, Chair
Antonia Papandreou-Suppappola
Baoxin Li
Shayok Chakraborty

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

Deep learning architectures have been widely explored in computer vision and have depicted commendable performance in a variety of applications. A fundamental challenge in training deep networks is the requirement of large amounts of labeled training data. While gathering large quantities of unlabeled data is cheap and easy, annotating the data is an expensive process in terms of time, labor and human expertise. Thus, developing algorithms that minimize the human effort in training deep models is of immense practical importance. Active learning algorithms automatically identify salient and exemplar samples from large amounts of unlabeled data and can augment maximal information to supervised learning models, thereby reducing the human annotation effort in training machine learning models. The goal of this dissertation is to fuse ideas from deep learning and active learning and design novel deep active learning algorithms. The proposed learning methodologies explore diverse label spaces to solve different computer vision applications. Three major contributions have emerged from this work; (i) a deep active framework for multi-class image classification, (ii) a deep active model with and without label correlation for multi-label image classification and (iii) a deep active paradigm for regression. Extensive empirical studies on a variety of multi-class, multi-label and regression vision datasets corroborate the potential of the proposed methods for real-world applications. Additional contributions include: (i) a multimodal emotion database consisting of recordings of facial expressions, body gestures, vocal expressions and physiological signals of actors enacting various emotions, (ii) four multimodal deep belief network models and (iii) an in-depth analysis of the effect of transfer of multimodal emotion features between source and target networks on classification accuracy and training time. These related contributions help comprehend the challenges involved in training deep learning models and motivate the main goal of this dissertation.

DEDICATION

Shree Gurubhyo Namaha

This dissertation is dedicated to my dearest amma & appa.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank everyone who has supported me and been a part of my PhD journey. I am very grateful to all the guidance, encouragement and friendship that I have earned during this time.

First, I would like to express my heartfelt gratitude to my mentor and advisor Dr. Sethuraman Panchanathan, who magnanimously gave me the freedom to pursue my research interests from the very beginning. I am deeply indebted to him for helping me convert to part-time status and allowing me continue my PhD journey while I stayed off-campus with family. I cannot thank him enough for believing in me and pushing me to strive for excellence in all my ventures.

I would like to thank Dr. Antonia Papandreou-Suppappola and Dr. Baoxin Li for serving on my committee and providing useful feedback on my research. I would like to convey my sincere gratitude to my mentor and committee member, Dr. Shayok Chakraborty for shaping my PhD dissertation and helping at every step of my research.

It has been an elevating experience working with fellow members of the Center for Cognitive Ubiquitous Computing (CUbiC) at Arizona State University. I would like to thank Vineeth, Sreekar and Troy for the many fruitful discussions and feedback during the initial stages of my graduate study. Many thanks to Rita and Prasanth for their helpful insights during research discussions. My sincere thanks to Hemanth for having supported my research from proof of concept to implementation to completion. I would also like to thank Kathy and Jessica for their prompt help whenever needed. I extend my gratitude to all the faculty and staff at Arizona State University for providing me with all the necessary support during the course of my PhD tenure. Many thanks to all my friends for the wonderful memories made throughout my doctoral career.

To my dear husband Arvind, words fail to express my gratitude. Thank you for being there for me and standing by me through the lows and highs of my PhD. If it wasn't for your support, co-operation and encouragement, this endevour would not have been possible. I appreciate my daughter, Yashasvee for the patience she showed during my thesis writing.

Most importantly, I would like to thank all my family (parents, parents-in-law, Radha, Ravi and Patti) for their continuous encouragement, patience and love through all the years. I would like to dedicate this work to my amma and appa, who have been an unwavering source of inspiration and support in every step of my life. Thank you for motivating me to pursue a doctoral degree in science and engineering. I am what I am today because of your love, care and support.

I thank the Almighty for giving me the strength and patience to work through all these years so that today I can stand proudly with my head held high.

TABLE OF CONTENTS

Page

LIST OF TABLES xii

LIST OF FIGURES xiv

CHAPTER

1 INTRODUCTION 1

1.1 Goals and Motivation 2

1.2 Major Contributions 4

1.3 Additional Contributions 6

1.4 Dissertation Outline 8

1.5 Previously Published Work 12

2 LITERATURE SURVEY 14

2.1 Benchmark Emotion Recognition Datasets 15

2.2 Multimodal Emotion Recognition Models 17

2.3 Transfer of Emotion Features between Deep Models 20

2.4 Multi-class Image Classification using Deep Models 21

2.5 Multi-class Active Learning 23

2.6 Multi-label lmage Classification using Deep Models 24

2.7 Multi-label Active Learning 26

2.8 Deep Models for Regression 27

2.9 Active Learning for Regression 27

2.10 Deep Active Learning 28

3 DEEP LEARNING MODELS 31

3.1 Artificial Neural Networks, (ANNs) 32

3.1.1 The Error Function 34

3.1.2 The Back-Propagation Algorithm 35

CHAPTER Page

3.1.3 Overfitting 39
3.2 Restricted Boltzmann Machines, (RBMs) 40
3.2.1 Training an RBM: Contrastive Divergence 42
3.3 Deep Belief Networks, (DBNs) 43
3.3.1 Greedy Pre-Training in Deep Belief Networks 44
3.3.2 Generating Data from a DBN 47
3.3.3 Stacked RBMs and Deep Belief Networks 49
3.4 Stacked Auto-Associators, (SAs) 50
3.5 Convolutional Neural Networks, (CNNs) 52
3.5.1 Notation 52
3.5.2 Architecture 53
3.5.3 Forward Propagation 54
3.5.4 Stochastic Gradient Descent, (SGD) 55
3.5.5 Back Propagation 56
3.5.6 ReLU Layer 57
3.5.7 Convolution Layer 57
3.5.8 Update Parameters 58
3.5.9 Gradient Computation 59
3.5.10 Pooling layer 60
3.5.11 Fully Connected Layer 61
3.6 Recurrent Neural Networks, (RNNs) 61
3.6.1 Long Short-Term Memory Networks, (LSTMs) 64
3.6.2 LSTM Equations 65
3.6.3 Backpropagation Through Time, (BPTT) 67

CHAPTER Page

3.7 Summary 69
4 DEEP MODELS FOR MULTIMODAL EMOTION RECOGNITION ... 70
4.1 Database for Holistic Emotion Recognition 71
4.2 emoFBVP Database 74
4.2.1 Apparatus and Setup For Data Collection 74
4.2.2 Data Capture Procedure 75
4.2.3 Properties of emoFBVP Database 78
4.2.4 Conclusions 81
4.3 Deep Belief Networks for Emotion Recognition 81
4.3.1 Multimodal Emotion Recognition Model 84
4.3.2 Unsupervised Feature Learning 86
4.3.3 Supervised Feature Selection 87
4.3.4 Feature Extraction - *emoFBVP* Database 88
4.4 Experiments 89
4.4.1 Baseline Model 89
4.4.2 D*emoFV* DBN Models 90
4.4.3 Results for D*emoFV* DBN 90
4.4.4 D*emoBV* DBN Models 93
4.4.5 Results for D*emoBV* DBN 93
4.4.6 D*emoFBV* DBN Models 95
4.4.7 Results for D*emoFBV* DBN 96
4.4.8 D*emoFBVP* DBN Models 98
4.4.9 Results for D*emoFBVP* DBN 98
4.4.10 Results on Standard Emotion Corpora 100

CHAPTER Page

4.4.11 Conclusions 101
4.5 Convolutional Deep Belief Networks for Emotion Recognition 101
4.5.1 Results for CDBN Models 102
4.5.2 Conclusions 103
4.6 Auto-associators for Emotion Recognition 103
4.6.1 Feature Learning Methods 104
4.6.2 Experiments 105
4.6.3 Results 107
4.6.4 Conclusions 108
4.7 Transfer of Emotion-Rich Features between Deep Belief Networks . . 108
4.7.1 emoDBN Models 110
4.7.2 emosource DBN model 110
4.7.3 emotarget and emotarget$_{\text{ft}}$ DBN models 111
4.7.4 Parameter Selection 112
4.8 Experiments and Results 112
4.8.1 Results when emosource is trained on emoFBVP dataset 116
4.8.2 Results when emosource is trained on Mind Reading dataset 117
4.8.3 Results when emosource is trained on MMI dataset 120
4.8.4 Results when emosource is trained on Cohn Kanade dataset . 120
4.8.5 Layer-wise Summary of the Results 120
4.8.6 Conclusions 122
4.9 Summary 123
5 DEEP ACTIVE LEARNING FOR SINGLE-LABEL IMAGE CLASSIFICATION 125

CHAPTER Page

5.1 Active Learning Models 126
5.1.1 Definition 126
5.1.2 Active Learning Scenarios 127
5.1.3 Query Strategies 129
5.1.4 An Example of Active Learning 131
5.2 Deep Active Learning Models 134
5.3 Proposed Framework 135
5.3.1 Cross-entropy Loss for Labeled Data 136
5.3.2 Entropy - Measure of Uncertainty 137
5.3.3 Joint Loss for Active Learning 138
5.3.4 Computing the Gradient 139
5.3.5 Active Learning Network Architecture and Training 140
5.4 Experiments and Results 142
5.4.1 Implementation Details 142
5.4.2 Datasets and Experimental Setup 143
5.4.3 Comparison Baselines 144
5.4.4 Active Learning Performance 146
5.5 Conclusion and Future Work 148
5.6 Summary 149
6 DEEP ACTIVE LEARNING FOR MULTI-LABEL IMAGE CLASSIFICATION 150
6.1 Proposed Framework 152
6.2 Multi-Label Active Learning Without Label Correlation 154
6.2.1 Sigmoid Cross-Entropy Loss for Labeled Data 154

CHAPTER Page

6.2.2 Entropy Loss for Unlabeled Data 155
6.2.3 Joint Objective for Multi-label Active Learning 155
6.2.4 Training and Implementation Details 156
6.3 Multi-Label Deep Active Learning With Label Correlation 157
6.3.1 Loss on Labeled Data 159
6.3.2 Loss on Unlabeled Data 160
6.3.3 Joint Objective for Training 160
6.4 Experiments and Results 163
6.5 Summary 167
7 DEEP ACTIVE LEARNING FOR IMAGE REGRESSION 168
7.1 Related Work 171
7.1.1 Deep Learning for Regression 171
7.1.2 Active Learning for Regression 172
7.1.3 Deep Active Learning for Regression 174
7.2 Proposed Framework 174
7.2.1 Loss on Labeled Data 176
7.2.2 Principle of Expected Model Output Change (EMOC) 176
7.2.3 Loss on Unlabeled Data 178
7.2.4 Novel Joint Objective Function 180
7.2.5 Gradient of Objective Function 181
7.3 Experiments and Results 183
7.3.1 Implementation Details 183
7.3.2 Datasets and Experimental Setup 184
7.3.3 Comparison Baselines and Evaluation Metrics 185

CHAPTER Page

7.3.4 Active Learning Performance 188
7.3.5 Study of the Active Sampling Criterion 190
7.3.6 Visual Illustration of the Selected Samples 194
7.4 Conclusions 196
8 FUTURE DIRECTIONS 198
8.0.1 Multimodal Emotion Recognition 198
8.0.2 Deep Active Learning Models for all Label Spaces 199
9 SUMMARY 202
9.0.1 Summary of Contributions 202
9.0.2 Conference Submissions 204
9.0.3 Workshop Poster Presentations 205
BIBLIOGRAPHY 206
APPENDIX
A DERIVATIVE OF THE JOINT OBJECTIVE FUNCTIONS 219
B PERMISSION STATEMENTS FROM CO-AUTHORS 229

LIST OF TABLES

Table Page

4.1 emoFBVP Emotion Database Properties 73

4.2 Apparatus Used for Data Capture for Multi Modal Emotional Expression 75

4.3 Animation Units and Shape Units from Face Tracking data 79

4.4 Classification Accuracy (%) for D*emoFV* Models 91

4.5 Classification Accuracy (%) for D*emoBV* Models 94

4.6 Classification Accuracy (%) for D*emoFBV* Models 97

4.7 Classification Accuracy (%) for D*emoFBVP* Models 99

4.8 Emotion Recognition Using Facial Expressions 100

4.9 Emotion Recognition Using Vocal Expressions 100

4.10 Emotion Recognition Using Physiological Data 100

4.11 Emotion Recognition Using Multimodal Data 100

4.12 Emotion Recognition Using *emoFBVP* Database 102

4.13 Emotion Recognition Using Cohn Kanade Database 102

4.14 Emotion Recognition Using Mind Reading Database 102

4.15 Emotion Recognition Using DEAP Database 102

4.16 Emotion recognition using MAHNOB-HCI database 102

4.17 Multi-Modal Feature Learning Settings 104

4.18 Emotion Recognition Accuracy on *emoFBVP* Database 107

4.19 emosource DBN Trained on Dataset X 114

4.20 Source: *emoFBVP*, Target: Mind Reading. 115

4.21 Source: *emoFBVP*, Target: MMI 115

4.22 Source: *emoFBVP*, Target: Cohn Kanade 115

4.23 Source: Mind Reading, Target:*emoFBVP* 118

4.24 Source: Mind Reading, Target:MMI. 118

Table Page

4.25 Source: Mind Reading, Target: Cohn Kanade 118
4.26 Source: MMI, Target: emoFBVP 119
4.27 Source: MMI, Target: Mind Reading 119
4.28 Source: MMI, Target: Cohn Kanade 119
4.29 Source: Cohn Kanade, Target: emoFBVP 121
4.30 Source: Cohn Kanade, Target: Mind Reading. 121
4.31 Source: Cohn Kanade, Target:MMI 121
5.1 Query Strategy- Explanation 130
5.2 Active Learning - Example 131
5.3 Labeled Dataset 132
5.4 Unlabeled Dataset 132
5.5 Updated Labeled Dataset 133
5.6 Updated Unlabeled Dataset 133
5.7 Uni-modal Dataset Details. 144
5.8 Multi-modal Dataset Details. 144
7.1 Dataset Details 185
7.2 Label Complexity for $MAE = 9$ 191

LIST OF FIGURES

Figure Page

3.1 Feed Forward Neural Network 33

3.2 Restricted Boltzmann Machine 41

3.3 Deep Belief Network Model. 45

3.4 Recognition Weights Versus Generative Weights 46

3.5 Initialization of Weights 47

3.6 Stacked RBM Architecture 50

3.7 DAA Training Scheme 52

3.8 Recurrent Neural Network 62

3.9 An Unrolled RNN 62

3.10 Short - Term Dependency in RNN 63

3.11 Long - Term Dependency in RNN 64

3.12 LSTM Cell State 65

3.13 LSTM Gate Operation 65

3.14 LSTM Forget Gate 66

3.15 LSTM Input Gate 66

3.16 Updated Cell State 66

3.17 LSTM Output Gate 67

4.1 Equipment Used for Data Capture 76

4.2 Snapshot of a Subject Portraying Emotion, Surprise. 77

4.3 Skeletal Tracking and Joint Hierarchy - 20 Bone Joints Are Tracked. . . . 80

4.4 The RBM Architecture With Visible (V) and Hidden (H) Layers. 83

4.5 Illustration of Proposed D$emoFV$ Models 91

4.6 Illustration of Proposed D$emoBV$ Models: 94

4.7 Illustration of Proposed D$emoFBV$ Models 97

Figure Page

4.8 Illustration of Proposed Models: 99
4.9 RBM Pre-Training Models 105
4.10 Deep Auto-Associator Models. 106
4.11 Proposed emoDBN Models 111
5.1 Active Learning Example Using Toy Dataset 127
5.2 Membership Query Systhesis 128
5.3 Stream-Based Selective Sampling 128
5.4 Pool-Based sampling 129
5.5 Illustration of the Principle of Active Learning 134
5.6 Deep Active Learning Network Architecture. 141
5.7 Active Learning on the Uni-Modal Datasets. 147
5.8 Active Learning on Multimodal Datasets 148
6.1 Architecture of Multi-Label CNN Model Without Label Correlation. 156
6.2 Architecture of Multi-Label CNN-LSTM Model With Label Correlation. 161
6.3 Deep Active Learning on Benchmark Multi-Label Datasets. 162
6.4 Analysis of the Unlabeled Samples Queried for Annotation. 166
7.1 Illustration of the Proposed Deep Active Learning Framework 179
7.2 CNN Architecture for Deep Active Regression 184
7.3 MSE Vs Iteration Number: Synthetic Handwritten Digits 186
7.4 MSE Vs Iteration Number: WIKI Age Estimation 187
7.5 MSE Vs Iteration Number: MNIST Rotation 188
7.6 MSE Vs Iteration Number: BIWI Kinect 189
7.7 MSE Vs Iteration Number: QMUL Multiface 190
7.8 Results after Iteration Number 9 - MSE Vs Digit Class 191

Figure Page

7.9 Results after Iteration Number 9 -Proposed Method 193
7.10 Results after Iteration Number 9 - Random Sampling 194
7.11 Visual Comparison of top 15 EMOC scores -Proposed Method 195
7.12 Visual Comparison of top 15 EMOC scores -Random Sampling 196

Chapter 1

INTRODUCTION

Artificial Intelligence (AI) is one of the largely significant technologies of the current era. Machine learning is a way of achieving AI. It relies on the machine's ability to learn how to solve problems. Training the machine involves feeding large amounts of data to the algorithm and allowing it to adjust itself and improve. The system is then fed with new examples and asked to make predictions. If the training was successful, the system predicts labels with a high level of accuracy. The algorithms that have powered much of this success are referred to as Deep Learning (DL) algorithms. Deep learning is a new area of machine learning research, introduced with the objective of moving machine learning closer to AI. Deep learning architectures are inspired by the structure and function of the human brain. A major advantage of deep learning algorithms over machine learning algorithms is that they make better use of much larger datasets. Learning from more data leads to improved superior predictions thereby contributing to achieving state-of-the-art performance.

In recent years, deep learning has emerged as a dominant machine learning tool for a wide variety of domains. Deep architectures have been widely explored in computer vision and have achieved tremendous improvements in several vision tasks. Deep learning models have replaced the need for hand-crafted features with efficient algorithms for unsupervised and semi-supervised feature learning and have depicted commendable performance in a variety of applications. With the widespread deployment of cheap and inexpensive video cameras, computer vision has become ubiquitous in our society. They form an integral component of self-driving cars, security

and surveillance, assistive technology, medical diagnosis and robotics among myriads of other applications.

A fundamental challenge in training a deep neural network is the requirement of large amounts of labeled training data. The rapid escalation of technology and the widespread emergence of new data capture apparatus has resulted in the generation of humungous amounts of digital data in the modern era. However, while gathering such large quantities of unlabeled data is cheap and easy, annotating the data (with class labels) is an expensive process in terms of time, labor and human expertise. This poses a significant challenge in inducing supervised learning models. The situation is even more serious for deep networks as they require more hand-labeled training data as compared to other classification models. Thus, developing algorithms that minimize the human effort in training deep models is of paramount practical importance.

Active learning (AL) algorithms have gained popularity in reducing the human annotation effort in training machine learning models. Such algorithms automatically identify the salient and exemplar samples from large amounts of unlabeled data that can augment maximal information to the classification models that need to be labeled manually.

1.1 Goals and Motivation

The goal of this dissertation is to fuse ideas from deep learning and active learning and design novel deep active learning algorithms. The proposed learning methodologies explore diverse label spaces to solve a range of computer vision applications like multimodal emotion recognition, single-label and multi-label image classification and

regression. The dissertation has been inspired by some of the key challenges and goals in AI for computer vision. The motivations are highlighted below.

Even though both deep learning and active learning have been extensively studied, research on combining the two is still in a nascent stage. Most of the algorithms treat deep learning and active learning as two independent problems and do not exploit the deep model's ability to learn discriminating sets of features for the given task. A deep model is first learned using a conventional loss function (softmax loss for classification/ $L2$ loss for regression); the active sampling condition is then defined based on the posterior probabilities obtained from the last layer or the distance of a sample from the decision boundary. However, the merit of a deep model lies in its ability to learn a discriminating set of features for a given task; this property has not been leveraged in the existing algorithms combining deep learning and active learning. In this dissertation, we propose novel deep active learning algorithms which are designed to exploit this property and study their performance on a wide variety of computer vision applications. The proposed learning methodologies explore three diverse label spaces. We briefly describe the three label spaces here.

1. **Multi-class Classification:** In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes. (Classifying instances into one of the two classes is called binary classification). For example, classification of a set of images of fruits into oranges, apples, or pears. Multiclass classification makes the assumption that each sample is assigned to one and only one label: a fruit can be either an apple or a pear but not both at the same time.

2. **Multi-label Classification:** In machine learning, multi-label classification is a classification problem where multiple labels may be assigned to each instance. Multi-label classification is a generalization of multiclass classification, which is the single-label problem of categorizing instances into precisely one of more than two classes. In the multi-label problem there is no constraint on how many of the classes the instance can be assigned to. Formally, multi-label classification is the problem of finding a model that maps inputs x to binary vectors y (assigning a value of 0 or 1 for each element (label) in y). This can be thought as predicting properties of a data-point that are not mutually exclusive, such as topics that are relevant for a document. A text might be about any of religion, politics, finance or education at the same time or none of these.

3. **Regression:** In machine learning, regression refers to the problem of finding the best relationship that represents a set of given data. Regression involves estimating or predicting a response where the output variable takes continuous values. Given the following: $f : x \rightarrow y$; if y is real number/continuous, then this is a regression problem.

From the above discussion, we appreciate the need to fuse deep learning and active learning to develop novel deep active learning methodologies. The various label spaces explored are highly diverse in nature and annotating them (with class labels) is an expensive process in terms of time, labor and human expertise. Thus, developing algorithms that minimize the human effort in training deep models is of immense practical importance.

1.2 Major Contributions

The three major contributions of the dissertation as as follows.

1. A novel active learning framework to select the most informative unlabeled samples to train a Deep Belief Network (DBN) is proposed. A loss function specific to the task of active learning is introduced and the model is trained to minimize this loss. Extensive empirical studies on a wide variety of uni-modal and multi-modal vision datasets corroborate the potential of the proposed method for real-world image recognition applications.

2. The feature learning capabilities of deep neural networks is exploited and a novel framework to address the problem of multi-label active learning is proposed. An active sample selection criterion is integrated in the loss function used to train the deep networks. First, a framework without considering the correlation among the multiple labels is proposed using Convolutional Neural Networks (CNNs). Second, the correlations that exist among the multiple labels is modeled using Long Short Term memory (LSTM) cells. Extensive empirical studies on five benchmark multi-label datasets show that the proposed methods outperform state-of-the-art active learning techniques.

3. Ideas from deep learning and active learning are fused and a novel deep active learning paradigm for regression is proposed. The Expected Model Output Change (EMOC) is used as the active selection criterion and integrated with the objective function used to train the deep model. The resulting model optimizes this novel objective function and learns from salient examples that cause maximum change to the current model. Extensive empirical results on benchmark regression datasets demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation.

1.3 Additional Contributions

Through the course of the thesis the following additional contributions were made. Their motivations are detailed here.

Let us consider a social interaction situation where two people are having a conversation. Their interaction typically consists of a combination of verbal and non-verbal communication (like body gestures, facial expressions etc.) cues that help understand each other. Now, if one of the individuals is visually impaired, he/she misses out on all of the non-verbal communication cues during the interaction, making it difficult to comprehend the emotion of the interaction partner. We propose algorithms that use multimodal data like facial expressions, body gestures, audio expressions and physiological signals to recognize human emotions using deep architectures. For this, we created a comprehensive multimodal emotion dataset and made it publicly available to the research community. Using these deep models, one could build assistive devices for visually impaired people to help enrich their social interactions.

One significant challenge while training deep networks is the time taken to train these networks on large datasets. Consider a real-world example where we have a deep model trained on a multimodal emotion dataset. Let us call this model as the source model. The model recognizes emotions with reasonable accuracy and the training time is approximately 10 days. Now, we come across a new emotion dataset. We wish to train a new model (let us call this model as the target model) on the new dataset, but do not have much time available for training. Can we use the emotion-rich features already learned by our source model for training the target model? What is the effect on the classification accuracy and training time when we do so? We present answers to these questions. This makes our study extremely useful in a practical setting.

To the best of our knowledge, this is the first research approach to studying the effect of transfer of emotion features in a layer-by-layer manner in a multimodal setting.

The additional contributions made are as follows:

1. A new multi-modal emotion database (emoFBVP) was created consisting of multi-modal recordings of facial expressions, body gestures, vocal expressions and physiological signals of actors enacting various expressions of emotion. The database consists of audio and video sequences of actors enacting 23 different emotions in three varying intensities of expressions along with facial feature tracking, skeletal tracking and the corresponding physiological data. This is one of the first emotion datasets that has recordings of varying intensities of expresions of emotions in multiple modalities recorded simultaneously. The affective computing community will greatly benefit from the large collection of modalities recorded. The second contribution investigated the use of deep learning architectures - Deep Belief Networks (DBNs) and Convolutional Deep Belief Networks (CDBNs) for multimodal emotion recognition. Four DBN models were proposed and experiments showed that they generated robust multimodal features for emotion recognition. The CDBN model proposed learned salient multimodal features of low intensity expressions of emotions.

2. The effect of transfer of emotion-rich features between source and target networks on classification accuracy and training time in a multimodal setting for vision based emotion recognition is studied. This is the first research effort to study the transfer of emotion features layer-by-layer in a multimodal setting. The $emotarget$ and $emotarget_{ft}$ models proposed were able to successfully repurpose the emotion rich features learned by the $emosource$ model to train

the target models and achieve shorter training times and performance boosts respectively. The results obtained are extremely useful in a practical setting.

These related contributions help comprehend the challenges involved in training deep learning models and motivate the main goal of this dissertation. Due to this reason, we describe these additional contributions in the initial chapters of the thesis before illustrating the major contributions. The outline of the dissertation is given in the following section.

1.4 Dissertation Outline

The dissertation is structured in the following manner.

Chapter 2 has been organized to showcase a literature review for each of the contributions in the thesis. It begins by listing state of the art emotion recognition datasets in literature and motivates the need for a comprehensive multimodal database. It then describes deep models used in literature for emotion recognition using multiple modalities. It talks about the various challenges involved while performing emotion recognition and ways in which current models solve the problems. The chapter then describes the motivation to employ transfer learning in a deep context and outlines models available in literature that perform such a transfer of knowledge. These techniques greatly help with reducing the time taken to train large deep networks. The next section of the chapter gives an exhaustive survey of deep models that perform various computer vision applications like emotion recognition, object recognition, image annotation, digit recognition in the multi-class label space. This is followed by a survey of multi-label deep models and multi-label active learning techniques available in literature. The last section of the chapter enumerates deep models used for regression and popular active learning methods for regression.

Chapter 3 begins with an introduction to deep learning as a part of a larger family of machine learning methods based on learning data representations. It briefly introduces different deep architecture such as artificial neural networks, Restricted Boltzmann machines, deep belief networks, stacked auto-associators, convolutional neural networks and recurrent neural networks. This chapter provides high level descriptions of the deep models, their equations, training methodologies employed along with examples of potential applications. The chapter highlights the difference between discriminative learning and generative learning, error functions employed and explains the back-propagation algorithm used to train the deep models. The chapter also outlines the algorithm for stochastic gradient descent and minibatch gradient descent along with the advantages and disadvantages of using the same. The different parameters of the deep model - learning rate, momentum, rprop, rmsprop and weight decay are presented along with potential methods that can be employed to prevent overfitting. Extensions of these models are used in this dissertation along with active learning to perform different computer vision tasks.

Chapter 4 commences with the motivation for creating a comprehensive multimodal emotion database. The apparatus used, data capture methods and different properties of the *emoFBVP* database are discussed here. An explanation about the proposed deep models for multimodal emotion recognition and methods employed to train the models are also presented. Two baseline models are proposed to help analyze the performance of the proposed models. The experiments conducted to perform multimodal emotion recognition are outlined here. The performance of the proposed DBN models is compared with the baseline models and the results are presented in this chapter as well. The next two sections describe convolutional DBNs and stacked auto-associators for multimodal emotion recognition. The last section of the chapter

describes the *emosource*, *emotarget* and $emotarget_{ft}$ DBN models and how these models learn emotion rich multimodal features through the deep layers. The selection strategy employed while selecting the various parameters of the emoDBN models is also explained here. The experiments and results section investigates the effect of transfer of emotion rich features between source and target DBN networks along with the results. The performance gains observed are two-fold; there is a significant decrease in the time taken to train the networks and the recognition accuracy also shows an increase. The chapter ends highlighting the contributions made with a brief summary.

Chapter 5 opens with a brief overview of active learning methodologies with relevant algorithms and examples. It gives a formal definition and describes active learning scenarios and different query strategies. It explains the advantages of combining active learning and deep learning concepts with a toy dataset example. The next section of this chapter introduces the field of deep active learning - where ideas from deep learning and active learning are combined to learn intelligent models for classification and/ regression. A novel active sampling algorithm to identify the salient and exemplar unlabeled samples to be manually annotated to train DBNs is proposed. To the best of our knowledge, this is the first research effort to incorporate an active learning based criterion in the loss function and train the deep network to optimize the objective. The proposed method is validated on single-label image classification on a variety of benchmark datasets for different applications. Experimental results on a variety of uni-modal and multi-modal datasets from different application domains depict the promise and potential of the method for real-world image recognition applications.

Chapter 6 commences with a definiiton of multi-label classification highlighting the motivation to employ deep active methods for the same. In a multi-label learning setting, the problem is further aggravated as the presence/absence of each class needs to be checked separately to annotate a single unlabeled sample. Often, the number of possible classes is of the order of hundreds, which tremendously increases the labeling burden on the human annotator. Therefore, developing algorithms that reduce human effort in training deep models in a multi-label setting is of paramount practical importance. Active learning algorithms alleviate this problem by selecting the salient and informative samples from vast amounts of unlabeled data. This not only reduces the human effort in training machine learning models, but also produces models with much better generalization capabilities, as they get trained on the salient examples from the underlying data population. In this chapter, we propose a multi-label deep active learning framework that did not model the inherent label correlations and a framework that modeled the relationships between multiple labels. We successfully integrated an entropy based active sampling criterion in the loss function and used this novel joint objective to train the deep models. Our empirical results on benchmark multi-label datasets show that the proposed models outperform state-of-the-art multi-label active learning algorithms, thereby corroborating the potential of our methods for real-world classification problems.

In Chapter 7, ideas from deep learning and active learning are fused and a novel deep active learning paradigm for regression is proposed. The chapter embarks with a brief motivation to develop active learning for deep regression. It explains the principle of Expected Model Output Change (EMOC) and describes how it is modeled as an active selection criterion in the objective function used to train the deep model. The resulting model optimizes this novel objective function and learns from salient exam-

ples that cause maximum change to the current model. The latter part of the chapter details the CNN model used followed by a description of the extensive experiments conducted on benchmark regression datasets. The results obtained demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation.

Chapter 8 features possible future directions for methods proposed in this dissertation. The contributions of this dissertation have shown tremendous promise in using deep active learning techniques in real-world computer vision applications. The results depict the usefulness of the algorithms in reducing human annotation effort in inducing an appropriate classification / regression model. The possibilities of future work are numerous and a few sample directions are presented in this chapter.

Chapter 9 provides a high-level summary of contributions in this dissertation. It continues to enumerate the conference submissions, both published and under-review, inspired from the proposed research. The chapter concludes by listing some workshop presentations made during this dissertation.

Appendix A develops the derivation for the derivative of the joint objective functions for multi-class, multi-label and regression scenarios. Appendix B gives the permission statements from co-authors.

1.5 Previously Published Work

The contents of chapter (4) are based on previously published works, "Multimodal Emotion Recognition Using Deep Learning Architectures", WACV 2016. Chapter (5)

is adapted from published work, " Deep Active Learning for Image Classification", ICIP 2017. Chapter (6) is based on work under review, "Multi-label Deep Active Learning with Label Correlation", submitted to ICIP 2018 and Chapter (7) on Deep Active learning for Regression is adapted from work submitted to ACM MM 2018 (also under review).

Chapter 2

LITERATURE SURVEY

This chapter has been organized to showcase a literature review for each of the contributions in the thesis. It begins by listing state of the art emotion recognition datasets in literature and motivates the need for a comprehensive multimodal database. It then investigates deep models used in literature for emotion recognition using multiple modalities. It talks about the various challenges involved while performing emotion recognition and ways in which current models solve the problems. The chapter then describes the motivation to employ transfer learning in a deep context and outlines models available in literature that perform such a transfer of knowledge. These techniques greatly help with reducing the time taken to train large deep networks. The next section of the chapter gives an exhaustive survey of deep models that perform various computer vision applications like emotion recognition, object recognition, image annotation, digit recognition in the multi-class label space. We briefly motivate the main goal of the dissertation here. This is followed by an assessment of multi-label deep models and multi-label active learning techniques available in literature. Here, we highlight multi-label techniques that consider the correlations that exist between the multiple labels. The last section of the chapter enumerates deep models used for regression and current active learning methods for regression. We discuss the dearth of literature in active learning methods for regression and conceive the need for a general deep active learning paradigm for the same.

2.1 Benchmark Emotion Recognition Datasets

Emotion plays an important role in social interaction, human intelligence and perception. Understanding emotions becomes indispensable for the day-to-day functioning of humans. Perception of human emotions is vital for communication in the social environment. Technologies for processing daily activities like facial expression recognition, understanding speech and language have expanded the interaction modalities between humans and computers. With the growing use of human - computer interactions, emotion recognition technologies provide an opportunity to promote harmonious communication between computers and humans.

To study human emotional experience and expression in more detail and to develop benchmark methods for automatic emotion recognition, researchers are in need of rich sets of data. Recent advances in emotion recognition have motivated the creation of novel databases containing emotional expressions with most databases including audio, video or audio-visual data (Pantic *et al.* (2005); Douglas-Cowie *et al.* (2007); Grimm *et al.* (2008); McKeown *et al.* (2010); Koelstra *et al.* (2012)). Older databases consist of acted or deliberately expressed emotions while recently researchers have shared spontaneous or natural expression databases. Below, we present a discussion of the existing databases, organized by the captured modality. There are many facial expression databases available such as the Cohn-Kanade database (Kanade *et al.* (2000b)), PICS database, JAFFE database, AR database, PIE database and the MMI database. The MMI database is a web-based emotion database of posed and spontaneous facial expressions with both static images and videos in frontal and profile views (Pantic *et al.* (2005); Valstar and Pantic (2010)). The database consists of 61

adults acting basic emotions. This database provides an option to search within the corpus and is easily downloadable. The majority of databases have only static images (except Cohn-Kanade and MMI). The apex of the expression is only available making it difficult to understand the temporal segments of the expression. The data consists of unstructured files and further processing is required before use in automatic facial expression recognition systems.

The Belfast Database (BE) was created by Douglas-Cowie *et al.* (2000) and it includes spontaneous reactions in TV talk shows. It is very rich in both body gestures and facial expressions. The video sequences in this database have a lot of variety in the background and this makes it very challenging for automated recognition systems. The HUMAINE database consists of recordings of three natural reactions and six induced reactions. The database consists of varying number of participants and data in different modalities (Douglas-Cowie *et al.* (2007)) collected in different sites and at different times. Twelve hours of a German talk show were segmented and annotated to form the Vera Am Mittag (VAM) database (Grimm *et al.* (2008)). The database consists of 104 speakers uttering different sentences. The segments were annotated using the valence, activation and dominance framework. The MAHNOB-HCI database has five modalities precisely synchronized - eye gaze data, video, audio, peripheral and central nervous system physiological signals. Here, spontaneous emotional responses to affectively stimulating videos were recorded for 27 participants. Only 9 different emotions were captured for each participant. As spontaneous responses were being captured, the audio sequences captured consist of very few natural utterances and laughter only. Participants are seated during the capture and no body gesture details are available. Facial features are not tracked and head movement data is not available as part of the database.

Affective physiological databases are fewer when compared with audio-visual databases. Healey and Picard (2005) of MIT recorded a physiological dataset of 17 drivers under different stress levels. The electrocardiogram (ECG), Galvanic skin response (GSR) from hands and feet, electromyogram (EMG) from the right trapezius muscles and respiration pattern were recorded. Physiological signals from the peripheral and central nervous system along with face videos were recorded in the Database for Emotion Analysis Using Physiological Signals (DEAP) (Koelstra *et al.* (2012)). The videos were recorded for 22 participants and the EMG, electrooculogram (EOG), blood volume pulse (BVP), skin temperature and GSR were captured.

To contribute to the need for a comprehensive database consisting of multiple modalities, we create the emoFBVP database of multimodal (face, body gesture, voice and physiological signals) recordings of actors enacting various expressions of emotions.The database consists of audio and video sequences of actors displaying three different intensities of expressions of 23 different emotions along with facial feature tracking, skeletal tracking and the corresponding physiological data. We provide details about data capture, apparatus and other properties of the database in Chapter 4.

2.2 Multimodal Emotion Recognition Models

Emotion recognition is the process of predicting high-level affective content from the low-level signal cues. This process is complicated by the inherent multimodality of human emotion expression (e.g., facial and vocal expression). This multimodality is characterized by complex high dimensional and non-linear cross-modal interactions (Taylor *et al.* (2007)). Previous research has demonstrated the benefit of using mul-

timodal data in emotion recognition tasks and has identified various techniques for generating robust multimodal features (Busso *et al.* (2004); Ververidis and Kotropoulos (2008); Vogt and André (2005); Wimmer *et al.* (2008); Pantic *et al.* (2011)). However, although effective, these techniques do not take advantage of the complex nonlinear relationship that exists between the modalities of interest, or alternatively require the use of labeled data. In this dissertation, we apply deep learning techniques to provide robust features for emotion recognition.

Emotion recognition accuracy relies heavily on the ability to generate representative features. However, this is a very challenging problem. In this section, we demonstrate the effectiveness of Deep Belief Networks (DBN) for multimodal emotion feature generation. In this dissertation, we learn multi-layered DBNs that capture the non-linear dependencies of audio-visual and physiological features while reducing the dimensionality of the feature space.

There has been a substantial body of work on feature representation, extraction, and selection methods in the emotion recognition field in the last decade. Our work in this thesis, is motivated by the discovery of methods for learning multiple layers of adaptive features using DBNs (Bengio *et al.* (2009)). Research has demonstrated that deep networks can effectively generate discriminative features that approximate the complex non-linear dependencies between features in the original set. These deep generative models have been applied to speech and language processing, as well as emotion recognition tasks (Morgan (2012); Mohamed *et al.* (2012); Sivaram and Hermansky (2012)). In speech processing, Ngiam *et al.* (2011) proposed and evaluated deep networks to learn audio-visual features from spoken letters. In emotion recognition, Brueckner and Schuller (2012) found that the use of a Restricted Boltzmann

Machine (RBM) prior to a two-layer neural network with fine-tuning could significantly improve classification accuracy in the Interspeech automatic likability classification challenge (Schuller *et al.* (2012)). The work by Stuhlsatz *et al.* (2011) took a different approach for learning acoustic features in speech emotion recognition using Generalized Discriminant Analysis (GerDA) based on Deep Neural Networks (DNNs).

In recent years, there has been a growing interest in the development of technology to recognize an individual's emotional state. There is also an increase in the use of multimodal data (facial expressions, body expressions, vocal expressions and physiological signals) to build such technologies. Each of these modalities have very distinct statistical properties and fusing these modalities helps us learn useful representations of the data. Literature has shown various techniques for generating robust multimodal features (Busso *et al.* (2004); Vogt and André (2005); Pantic *et al.* (2011); Wimmer *et al.* (2008)) for emotion recognition tasks. The high dimensionality of the data, the non-linear interactions across the modalities along with the fact that the way an emotion is expressed varies across people complicate the process of generating emotion specific features (Taylor *et al.* (2007); Anagnostopoulos *et al.* (2015)). Deep architectures and learning techniques have shown to overcome these limitations by capturing complex non-linear feature interactions in multimodal data (Kim *et al.* (2013)).

Previous research has shown that deep architectures effectively generate robust features by exploiting the complex non-linear interactions in the data (Taylor *et al.* (2007)). Deep architectures and learning techniques are very popular in the speech and language processing community (Morgan (2012); Sivaram and Hermansky (2012); Mohamed *et al.* (2012)). Ngiam *et al.* (2011)) report impressive results on audio-

visual speech classification. They use sparse Restricted Boltzmann Machines (RBMs) for cross-modal learning, shared representation learning and multimodal fusion on CUAVE and AVLetters dataset. Srivastava and Salakhutdinov (2012) applied multimodal deep belief networks to learn joint representations that outperformed SVMs. They used multimodal deep Boltzmann machines to learn a generative model of images and text for image retrieval tasks. Kahou *et al.* (2013) used an ensemble of deep learning models to perform emotion recognition from video clips. This was the winning submission to the Emotion Recognition in the Wild Challenge (Dhall *et al.* (2013)). Deep learning has also been applied in many visual recognition studies (Lee *et al.* (2008); Tang and Eliasmith (2010); Lee *et al.* (2011); Sohn *et al.* (2011); Krizhevsky *et al.* (2012)).

Our research is motivated by the above recent approaches in multimodal deep learning. We investigate the use of deep learning architectures - Deep Belief Networks (DBNs) and Convolutional Deep Belief Networks (CDBNs) for multimodal emotion recognition. Four DBN models are proposed and experiments show that they generate robust multimodal features for emotion recognition. A Convolutional Deep Belief Network (CDBN) model is proposed which successfully learns salient multimodal features of low intensity expressions of emotions.

2.3 Transfer of Emotion Features between Deep Models

The introduction of deep architectures has brought significant improvements in many visual recognition tasks. These algorithms come with huge computational costs and finding the best training algorithm that offers the shortest training time is an interesting area of research.

Complex models like the CNNs can overfit the data, especially when the dataset is small. Researchers have resorted to using transfer learning across tasks to overcome this problem. The weights of the deep model are initialized with those of the network trained for related tasks before finetuning them using the target datasets (Girshick (2015); Donahue *et al.* (2014); Krizhevsky *et al.* (2012); Raina *et al.* (2007)). Yosinski *et al.* (2014) experimentally quantified the generality versus specificity of neurons in each layer of a deep CNN. They trained pairs of CNNs on the ImageNet dataset and characterized the layer by layer transition of features from general to specific.

One of the main concerns with using deep architectures for vision tasks is the amount of time required to train the network. Therefore, finding the appropriate training algorithm that gives good performance accuracy with reduced training time becomes very important. In this dissertation, we follow a transfer learning approach and present a study to investigate the effect of transfer of emotion-rich features between source and target networks on classification accuracy and training time.

2.4 Multi-class Image Classification using Deep Models

In recent years, deep learning has emerged as a dominant machine learning tool for a wide variety of domains. Deep architectures have been widely explored in computer vision and have achieved tremendous improvement in several vision tasks including image recognition, object detection, and image segmentation among others. The surge of deep learning started in 2006 when Deep Belief Networks (DBNs) were introduced (Hinton *et al.* (2006)). DBNs and its variants have been shown to depict excellent performance in several applications including visual object recognition,

emotion recognition, speech phone recognition and image denoising.

Computer vision problems like image classification and object detection have traditionally been approached using hand-engineered features like SIFT, HoG, bag-of-visual-words descriptor, followed by learning algorithms like the SVM. The performance of these algorithms was heavily dependent on the features used. To alleviate this issue, deep models were developed which incorporated learning of features from raw images. The Restricted Boltzmann Machines (Hinton (2002)) and the Deep Belief Networks (Hinton and Salakhutdinov (2006)) are some of the early examples of deep models that depicted promising empirical performance. The fundamental idea was to leverage vast amounts of unlabeled data to train the models; the pre-trained models served as a good initialization for supervised tasks such as image classification. There are several different kinds of deep learning architectures (CNNs, RNNs, DBNs to name a few); in this section, we present a brief survey of deep belief networks (DBNs), as we use them to study the performance of our framework - deep active learning for single label image classification.

DBNs are generative deep models and can be effectively constructed by greedily training and stacking multiple RBMs (Hinton and Salakhutdinov (2006)). This is done in two stages - pre-training and fine-tuning. The pre-training stage is unsupervised and has no labels involved and solely relies on the unlabeled data. The weights and biases that are learned are used as starting points for the fine-tuning supervised learning stage. In the fine-tuning stage, typically, a softmax layer is added on top of the stacked RBMs to make the model discriminative. A standard back-propagation is performed with a goal to minimize classification errors given the labeled samples. Tang and Mohamed (2012)proposed a multi-resolution DBN model which learns fea-

tures from a multi-scale representation of images. Liu *et al.* (2011) proposed discriminative DBNs (DDBNs) which integrated the abstraction ability of DBNs and the discriminative ability of backpropagation strategy. DBNs and its variants have been shown to depict excellent performance in several applications including visual object recognition (Salakhutdinov and Larochelle (2010)), emotion recognition (Ranganathan *et al.* (2016a)), speech phone recognition (Dahl *et al.* (2010)) and image denoising (Ranzato *et al.* (2011)).

2.5 Multi-class Active Learning

A fundamental challenge in training a deep neural network is the requirement of large amounts of labeled training data. Thus, developing algorithms to minimize human effort in training deep models is of paramount practical importance. Active learning algorithms automatically identify the salient and exemplar samples from large amounts of unlabeled data and reduce human annotation efforts in inducing a classification model. Active learning is a well-studied problem in machine learning. A comprehensive review of several active learning algorithms developed over the last several years can be found in Settles (2010). In a typical pool-based batch mode active learning (BMAL) setting, the learner is exposed to a pool of unlabeled instances and it iteratively queries batches of samples for annotation. Initial batch mode active learning techniques were largely based on greedy heuristics, such as maximizing the diversity of the selected samples or minimizing their distance from the classification hyperplane (Schohn and Cohn (2000); Brinker (2003a)). More recently, optimization based strategies have been proposed which have been shown to outperform the heuristic approaches. Hoi *et al.* (2006a) used the Fisher information matrix as a measure of model uncertainty and proposed to query the set of points that max-

imally reduced the Fisher information. The same authors also proposed a BMAL scheme based on SVMs where a kernel function was first learned from a mixture of labeled and unlabeled samples, which was then used to identify the informative and diverse examples through a min-max framework (Hoi *et al.* (2008)). Guo and Schuurmans (2007) proposed a discriminative BMAL strategy where the sample selection criterion was based on maximizing the log-likelihoods of the selected samples with respect to their optimistically assigned class labels and minimizing the entropy of the unselected samples in the unlabeled pool. Guo also proposed a batch mode active learning algorithm, independent of the classification model, by maximizing the mutual information between the labeled and unlabeled sets (Guo (2010a)), so that the labeled samples, together with the samples queried for annotation were good representatives of the unselected unlabeled samples. Chakraborty *et al.* (2013) proposed a generalized BMAL scheme, based on Quasi-Newton optimization, and applied it to the face-based biometric recognition problem.

2.6 Multi-label lmage Classification using Deep Models

Multi-label image annotation is one of the most important open problems in computer vision. Unlike existing works that usually use conventional visual features to annotate images, features based on deep learning have shown potential to achieve outstanding performance. Recent years have witnessed an explosive growth of digital images, and most of them are captured by handheld mobile devices. There is an urgent need to develop effective techniques to annotate images with several labels according to the semantic contents, which can be deployed in many applications, such as personal image collection organization and large-scale image retrieval. From the point of view of pattern recognition, the issue of image annotation can be considered as an issue of assigning a set of relevant tags to an image according to the contents

inside it, in which learning good features is a very important task and will significantly improve the overall system performance.

By exploiting deep architectures, deep learning technologies discover hidden structures and effective features from the training data and help improve model performance. We know that Convolutional Neural Networks (CNNs) use convolution and max-pooling as the fundamental operations and are specifically suited for image data. K.Zhaoa *et al.* (2016) proposed a unified framework for Deep Region and Multi-label learning (DRML) using CNNs and showed that their method outperformed alternative techniques. Zhu *et al.* (2017) successfully used CNNs for multi-label pedestrian attribute classification. Huang *et al.* (2013) used deep belief networks for multi-task, multi-label learning and showed state-of-the-art performance on two public image datasets. Zhu *et al.* (2015) proposed a multi-modal deep learning network that optionally integrated multiple deep networks pre-trained with CNNs. Their empirical studies evaluated the performance of the proposed framework for multi-label image annotation and the results validated the effectiveness of their algorithm. Gong *et al.* (2013) proposed the multi-label deep convolutional ranking net to address the multi-label annotation problem. They proposed a model that successfully redesigned the ranking cost layer for multi-label prediction tasks. From the above survey, we note that CNNs have depicted promising performance for multi-label image classification. In this dissertation, we therefore use CNNs as our preferred architecture for the proposed deep active framework for multi-class classification.

2.7 Multi-label Active Learning

In multi-label classification problems, each data sample can be associated with multiple labels simultaneously. A challenging issue for multi-label classification is to identify and model the correlation of multiple labels, to achieve good prediction accuracy. Moreover, manual annotation of a multilabel sample necessitates a human oracle to consider the presence/ absence of every individual label, which is an expensive process in terms of time, labor and human expertise. Thus, developing algorithms that reduce human effort in training a multi-label classifier is of immense practical importance.

While active learning has been extensively studied for the multi-class problem, multi-label active learning is much less explored. Wu *et al.* (2014) proposed a novel example-label based multi-label active learning method. Huang *et al.* (2015) proposed a multi-label active learning method, which queried the relevance ordering of label pairs (by incorporating a selection strategy and a label ranking model) to reduce the labeling burden on the human annotator. Hung and Lin (2011) proposed a multi-label active learning framework with an auxiliary learner, based on the principle of maximum loss reduction with maximum confidence (MMC). Qi *et al.* (2008) proposed a two dimensional active learning framework which queried sample-label pairs for annotation, rather than all the labels of a given sample. Wang *et al.* (2016) used Recurrent Neural Networks (RNNs) in conjunction with CNNs and built a joint embedding space of image and semantic structures. The RNN memorizes long-term label dependencies and the framework exhibits good performance with cross-label correlation implicitly preserved.

2.8 Deep Models for Regression

A large number of regression based deep learning algorithms have been recently proposed. Here, the goal is to predict a set of continuous values as output. Recently, CNNs have been successfully applied for human pose estimation (Li and Chan (2014), Pfister *et al.* (2014), Toshev and Szegedy (2014)) where the regressed values correspond to the positions of the body joints on the image plane. CNNs also effectively predict facial fiducial points (Sun *et al.* (2013)) when applied to facial landmark detection. Szegedy *et al.* (2013) and Jaderberg *et al.* (2016) use deep networks for object and text detection and predict a bounding box for localization. These deep models use the conventional $L2$ loss function for training. Zhang *et al.* (2014) introduced a CNN optimized for landmark detection and attribute classification. They combine the $L2$ loss function with the Softmax classification function to increase robustness to outliers. Wang *et al.* (2014) combine bounding box localization with object segmentation using a similar approach. Gkioxari *et al.* (2014) use a loss function composed of a body pose estimation term and an action detection term. Dosovitskiy *et al.* (2015) and Eigen *et al.* (2014) use multiple $L2$ loss functions for object generation and depth estimation. From the above survey, we see that deep models (specifically CNNs) trained using the $L2$ loss function can be applied effectively for regression tasks.

2.9 Active Learning for Regression

In the literature, work targeting AL for regression is less explored when compared to AL methods developed for classification. Willett *et al.* (2006) theoretically analyzed AL in the context of regression. Population based AL methods were pro-

posed by Sugiyama (2006), where the input data examples are arbitrarily generated in the space. A theoretically optimal AL algorithm was proposed by Sugiyama and Nakajima (2009). This directly minimizes the generalization error by employing an additive regression model. Freund *et al.* (1997) applied a variance-based Query by Committee (QBC) framework to regression. Cohn *et al.* (1996) minimized the output variance to reduce the generalization error. Yu and Kim (2010) provided passive sampling heuristics based on the geometric characteristics of the data. Cai *et al.* (2013) presented a novel data sampling solution which queries the example leading to the largest model change. Most regression-based AL techniques are developed only for sequential mode. Batch Mode Active Learning (BMAL) techniques are very useful in practice and it is highly desirable to derive BMAL methods in the context of regression. Existing BMAL algorithms are derived with classification models (Brinker (2003b), Hoi *et al.* (2006b), Belagiannis *et al.* (2014), Hoi *et al.* (2009), Guo and Schuurmans (2008), Guo (2010b), Chattopadhyay *et al.* (2013), Azimi *et al.* (2012), Chakraborty *et al.* (2015a)) and cannot be directly generalized to regression. Cai *et al.* (2013) extend to BMAL by simulating the sequential mode AL behavior to simultaneously choose a set of examples without re-training. They introduce a novel AL framework for regression called EMCM, which queries the examples maximizing the model change once added to the training data.

2.10 Deep Active Learning

Even though both deep learning and active learning have been extensively studied, research on combining the two is still in a nascent stage. Wang and Shang (2014a) proposed ALDL, an active labeling method for deep learning using DBNs. After training the DBN, it was applied on all the samples in the unlabeled set; three active learning criteria were studied to select a batch of k unlabeled samples: least

confidence, margin sampling and entropy. Stark *et al.* (2015a) presented an active learning algorithm using CNNs for CAPTCHA recognition. An uncertainty sampling approach was proposed for active learning where the difference between the highest and the second highest probabilities for a given unlabeled sample was used to compute its uncertainty. Along similar lines, Zhou *et al.* (2010a) proposed the active deep network (ADN) framework for sentiment classification. Uncertainty sampling was used to select the unlabeled samples for annotation, where the uncertainty of a sample was defined as its distance from the separating hyperplane. Freytag *et al.* (2014) proposed an approach to measure the expected change of model outputs. For each example in the unlabeled set, the expected change of model predictions is calculated and marginalized over the unknown label. The resulting score for each unlabeled example is used for active learning with a broad range of models (including deep models) and learning algorithms. Käding *et al.* (2016) propose a new generalization of the EMOC principle for deep architectures. They also present easy-to-implement approximations that yield efficient techniques for active selection.

All these algorithms treat active learning and deep model training as two independent problems. A deep model is first learned using a conventional loss function (softmax loss or $L2$ loss); the active sampling condition is then defined based on the posterior probabilities obtained from the last layer or the distance of a sample from the decision boundary or the EMOC scores. However, the merit of a deep model lies in its ability to learn a discriminating set of features for a given task; this property has not been leveraged in the existing algorithms combining deep learning and active learning.

In order to address these practical issues, we propose three major contributions

in this Ph.D dissertation:

1. A novel active learning framework to select the most informative unlabeled sample to train a Deep Belief Network (DBN) is proposed. A loss function specific to the task of active learning is introduced and the model is trained to minimize this loss. Extensive empirical studies on a wide variety of unimodal and multimodal vision datasets corroborate the potential of the proposed method for real-world image recognition applications.

2. The feature learning capabilities of deep neural networks is exploited and a novel framework to address the problem of multi-label active learning is proposed. An active sample selection criterion is integrated in the loss function used to train the deep networks. First, a framework without considering the correlation among the multiple labels is proposed using Convolutional Neural Networks (CNNs). Second, the correlations that exist among the multiple labels is modeled using Long Short Term memory (LSTM) cells. Extensive empirical studies on five benchmark multi-label datasets show that the proposed methods outperform state-of-the-art active learning techniques.

3. Ideas from deep learning and active learning are fused and a novel deep active learning paradigm for regression is proposed. The Expected Model Output Change (EMOC) is used as the active selection criterion and integrated with the objective function used to train the deep model. The resulting model optimizes this novel objective function and learns from salient examples that cause maximum change to the current model. Extensive empirical results on benchmark regression datasets demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation.

Chapter 3

DEEP LEARNING MODELS

This chapter first introduces deep learning as part of a larger family of machine learning methods based on learning data representations. It describes different deep architectures like Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). We will employ these models and extensions of these models in later chapters for different computer vision applications.

A Neural network is a biologically-inspired architecture which enables a computer to learn from the data it observes. Deep learning is a powerful learning technique for neural networks. Deep learning is a class of machine learning algorithms that consists of multiple layers of non-linear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. The learning is either supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis). The deep models learn multiple levels of representations that correspond to different levels of abstraction forming a hierarchy of concepts. Modern deep learning models are based on artificial neural networks or latent variables organized layer-wise in deep generative models.

Deep learning exploits a hierarchy where higher level, more abstract concepts are learned from the lower level ones. Deep architectures are often constructed in a greedy layer-by-layer method. Deep learning chooses features that improve performance by disentangling thses abstractions. For supervised learning tasks, deep learning meth-

ods avoid feature engineering, by translating the data into compact intermediate representations similar to principal components and derive layered structures that remove redundancy in representation. Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than labeled data.

In the following sections, we briefly explain popular deep architectures such as deep neural networks, deep belief networks, convolutional neural networks and recurrent neural networks.

3.1 Artificial Neural Networks, (ANNs)

The structure of a neural network is a graph as shown in Figure (3.1). In this context the vertices are called nodes or neurons and edges are called synaptic connections. The synaptic connections have strengths, called weights, which change during the learning process. The networks are structured in layers, according to the connections between nodes. The nodes inside a layer are not connected with each other but are connected to all the nodes in the following layer. The layers can be grouped as follows: the input layer that stores the given data, the hidden layers that define a better representation of the data and the output layer that contains the output, after a pass through the network. The details given in this dissertation about ANNs, Restricted Boltzmann machines and deep belief networks are inspired from descriptions given in (Rosca (2018)).

The input data is transformed into a vector of real numbers and presented to the network. A pass is performed and the neurons get activated, taking values (binary $\{0, 1\}$ or real numbers). These are called states or activations. The output layer can be missing, depending on the task performed by the neural network. The state (or

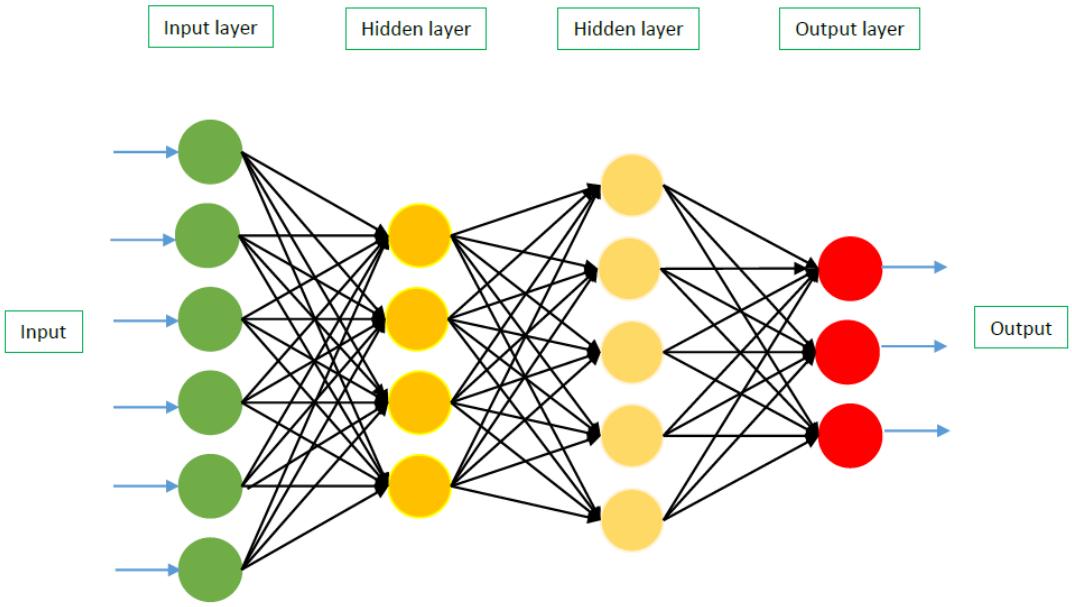


Figure 3.1: Feed Forward Neural Network

activation of a neuron) is typically a real value that depends on the activities in the previous layer and the weights, as follows:

$$y_i^{l+1} = \sigma\left(\sum_j w_{ij} y_j^{(l)} + b_i\right) \tag{3.1}$$

Here, b_i is the bias associated with the unit and σ is the activation function. Most often, the activation functions come from the sigmoid family (such as the logistic sigmoid). The range of the logistic sigmoid $\sigma(x) = \dfrac{1}{(1+e^{-x})}$ is $\{0, 1\}$, making the function particularly suitable when the activations of the neurons represent probabilities. Recent developments have shown that rectifier functions such as $\max\{0, x\}$ perform better for certain kind of tasks (Dahl *et al.* (2013)). Neural networks can be split in different categories, according to the connections between layers: feed forward networks and recurrent networks. In feed forward nets, the connections between units do not form a directed cycle. Recurrent neural nets are characterized by forming a

cycle in the connections between neurons.

3.1.0.1 Discriminative and Generative Learning

The aim of discriminative learning is to find a map from the input data to labels. The labels can be discrete (classification) or continuous (regression). When solving a discrimination problem, the neural network is given a labelled dataset, of the form (x, y), where x is the data instance (represented as a vector of real numbers) and y is the label. The aim of the neural network is to learn a target function f such that $f(x) = y$ and be able to predict the value of the function for unseen data. In general, discriminative models aim to compute the conditional probability of a label (y) given a data instance $(x) : p(y|x)$.

The aim of generative learning is to compute a probability distribution that is very likely to have generated the data. Unlike discriminative models, generative models can be used in an unsupervised setting, in which there are no labels given for the data, making them suitable for applications like clustering and density estimation (chapters 2 and 9 from (Nasrabadi (2007)). Generative models can be applied for classification and regression, as they can model the joint probabilities, $p(x, y)$, between the data x and the labels y, by first computing the marginal probability of the data given a label: $p(x|y)$. Traditionally, neural networks are used as discriminative models.

3.1.1 The Error Function

Correcting the network when it makes mistakes is an important step in the learning process. We need to show the network what it needs to learn and how it needs to improve, so that it can adjust its current understanding about the target function. This requires a measure for the error. The choice of error function is application dependent and has many consequences in the learning process. When presenting a

data instance to the network, we can compare the output it produces (y) with the target output (t). A common choice is to use the square of the $L2$ norm between the two vectors:

$$\mathbb{E}(t, y) = \|t - y\|^2 . \tag{3.2}$$

When computing the error on the entire dataset, the mean square error is used, which is the average error on each individual training cases:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \|y_i - t_i\|^2 \tag{3.3}$$

The root mean square error is also used in common so that the data and error have same units.

$$RMSE = \sqrt{MSE}. \tag{3.4}$$

3.1.2 The Back-Propagation Algorithm

The back-propagation algorithm uses the derivatives of the error function with respect to the weight matrix and the bias term to find a set of parameters that minimizes the value of the error function. The algorithm (refer Algorithm (1)) back propagates the derivatives from the output layer to the layers below, one layer at a time. Once the partial derivatives are computed, an optimization method is employed to find values of the parameters that minimize the error function. The most common optimization technique used is gradient descent, but other algorithms such as conjugate gradient have been successfully employed (LeCun *et al.* (1998b)). For a more comprehensive understanding on back propagation, please refer to (LeCun *et al.* (1998b)).

Multiple passes through the data are required when training a neural network. Each pass is called a training epoch. Inside an epoch, the parameters can be updated at different frequencies:

Algorithm 1 Back propagation learning algorithm (LeCun *et al.* (1998b))

1: Initialize the weights with random values between 0 and 1.
2: **While** not done training **do**
3: **for** d in data **do**
4: Forward Pass
5: Starting from the input layer, use equation (3.1) to do a forward pass through the network, computing the activitions of the neurons at each layer.
6: Backward Pass
7: Compute the gradient of the error with respect to the output layer activations
8: **for** layer in layers **do**
9: Compute the gradient with respect to the linear input of neurons in the layer above
10: Compute the gradient with respect to the parameters of the current layer
11: Compute the gradient with respect to the activations of the current layer
12: Update the parameters

1. Online Weight Update

   In the Online weight update method, we correct the model after each training sample. Because the error function changes after each data sample, the gradients with respect to the parameters fluctuate highly between updates. This results in less stable learning and is not used in practice for this reason.

2. Full-batch Weight Update

   The entire dataset is used to compute the error and the weights are updated using the sum of the gradients obtained from the individual samples. This requires that we go through the entire dataset multiple times to improve the set

of parameters available in the beginning. This method is computation intensive and therefore not used widely.

3. Mini-batch Weight Update

   The above two approaches are combined and we run only a part of the training cases, making the parameters reasonable before continuing. The updates are averaged over multiple cases. This makes the parameters more stable than online learning. When using mini-batch learning it is important to ensure that a mini batch has an equal number of instances of each class the model is trying to learn. This helps reduce fluctuations between the different mini-batch updates.

The back-propagation algorithm describes a way to compute the derivatives of the error function with respect to the weights of the network. These derivatives are then used in conjunction with various optimization algorithms. In the gradient descent algorithm (refer Algoirithm (2)), the weights are updated in the direction of the negative of the gradient. Here ϵ is called the learning rate.

Algorithm 2 Gradient Descent (ϵ, threshold)

1: **while** $|x_n - x_{n-1}| <$ threshold **do**

2: $x_{n+1} = x_n - \epsilon \nabla f(x_n)$

3.1.2.1 Parameters and Techniques

Learning rate

Experiments have shown that the learning rate is a crucial parameter that influences the convergence of training (Schulz *et al.* (2010)). There are many ways to set the learning rate. One method is to try different values in the set $10^{-1}, 10^{-2}, \cdots, 10^{-5}$

and cross validate. The value that yields the best result is chosen. The learning rate is constant throughout training. Another method would be to monitor the error on a validation set. If the error is steadily decreasing, increase the learning rate by a constant factor. If the error is increasing, decrease the learning rate. Towards the end of training, when the error stops decreasing steadily, we further decrease learning rate. This removes the fluctuations in the weights between mini-batches and helps towards keeping a steady set of weights for the final ones.

Momentum

The momentum method is a technique used to improve the speed of learning. Here, the previous values of the weight gradients are used when computing the current update. This ensures that the gradient moves in the same direction as before, thereby speeding up learning. When the current gradient and the previous gradient agree on the direction the weight should move, a bigger step is performed in that direction.

Nesterov Method for Momentum

In the Nesterov momentum method, the parameters according to the direction of the old update are first updated. Then, a forward and a backward pass are performed to compute the gradients. The parameters are updated again using the new computed gradients. The Nesterov momentum can increase the performance of neural networks and tends to perform better than classic momentum (Sutskever *et al.* (2013), (Tieleman and Hinton (2012))).

3.1.3 Overfitting

During training, the network sees regularities in data. It is impossible for a network to distinguish between real regularities that we aim to learn and the accidental regularities occurring in the data. These accidental regularities can potentially make the network not generalize well to unseen testing data. This is an important issue that arises when using machine learning techniques, and it is called overfitting. Trying to fit the training set perfectly will guarantee that the model has learned the accidental regularities in the data and thus will not be able to generalize. Methods that aim to avoid overfitting by imposing a complexity penalty to the model are commonly referred to as regularization techniques. Some regularization techniques are discussed below.

Weight Decay

It has been observed (Krogh and Hertz (1992)) that extreme values (very small or very big) for the parameters of a machine learning model are a symptom of overfitting: the model is trying to perfectly learn the regularities of the data. In order to avoid weights increasing too much, a weight penalty is imposed (Srivastava (2013)).

Early stopping

The idea behind early stopping is to prevent the network from overfitting by stopping training before convergence is achieved. This is done by keeping a validation set on which the error is computed during learning. Once the error stops decreasing on the validation set, training is stopped. This method is highly used to determine when to stop training a model.

Model Averaging

In this method, multiple models are used and their predictions are averaged. Literature has shown that averaging the predictions from multiple models is better than using one single model. This method helps find good models that err on different test cases.

Bagging and boosting

Bagging uses multiple bootstrap datasets to train different classifiers, and at test time averages them in order to obtain a classification result. A bootstrap dataset is obtained by uniformly sampling with repetition from the original dataset. In Boosting, the bootstrap sets are not obtained by increasing the probability of obtaining a sample misclassified by the models trained with the previous bootstrap sets. A comprehensive comparison between boosting, bagging and Bayesian model averaging is offered by Davidson and Fan (2006).

Generative Pre-training

Generative pre-training is another technique that helps in reduction of overfitting. The core idea is to learn the structure of the data, without supervision, and then apply discriminative learning algorithms.

3.2 Restricted Boltzmann Machines, (RBMs)

A Restricted Boltzmann machine is a neural network with two layers of stochastic binary units, with their connections forming an undirected bipartite graph. The layers of the network are called visible and hidden (Figure (3.2)).

RBMs are generative models: the hidden units are latent variables that generate

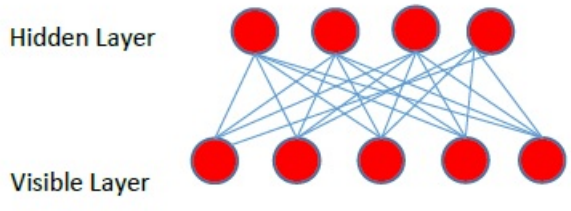


Figure 3.2: Restricted Boltzmann Machine

the observable data (the visible units). These hidden units define a posterior probability distribution on the states of the visible units. The energy function is given below; it emphasizes the structure of the network:

$$\begin{aligned} E(v,h) &= \sum_{i \ visible} a_i v_i - \sum_{i \ hidden} b_i h_i - \sum_{i \ visible, \ j \ hidden} w_{ij} v_i h_j \\ &= a^T v - b^T h - v^T W h \end{aligned} \tag{3.5}$$

At equilibrium the network assigns a probability to each possible state of the network, depending on the energy:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{3.6}$$

In equation (3.6), Z is the normalizing constant, called the partition function:

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3.7}$$

$$p(v) = \sum_{h} p(v,h) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{3.8}$$

The derivative of the log probability of a data instance can be computed as follows:

$$\frac{\partial}{\partial w_{ij}} (\log p(v)) = \langle v_i, h_j \rangle_{data} - \langle v_i, h_j \rangle_{model} \tag{3.9}$$

Equation (3.9) gives an idea for a learning algorithm: use the gradient ascent algorithm with the following weight updates:

$$\nabla w_{ij} = \epsilon \left(\langle v_i, h_j \rangle_{data} - \langle v_i, h_j \rangle_{model} \right) \tag{3.10}$$

In order to be able to use equation (3.10) as part of a training algorithm we need:

- an unbiased sample of $\langle v_i, h_j \rangle_{data}$
- an unbiased sample of $\langle v_i, h_j \rangle_{model}$

Due to the structure of an RBM, the hidden units are conditionally independent given the value of the visible units. This property makes it simple to get an unbiased sample from the $\langle v_i, h_j \rangle_{data}$ distribution, as the hidden units do not depend on each other given the visible unit:

$$p(h_j = 1 \big| v) = \sigma \left(\sum_{i=1}^{N} w_{ij} v_i + b_i \right) \tag{3.11}$$

Similarly, the visible units are conditionally independent given the hidden units:

$$p(v_i = 1 \big| h) = \sigma \left(\sum_{j=1}^{N} w_{ji} h_j + a_i \right) \tag{3.12}$$

Calculating the unbiased sample of $\langle v_i h_j \rangle_{data}$, is called the positive phase of an algorithm that trains an RBM. Calculating an approximation of the unbiased sample of $\langle v_i h_j \rangle_{model}$, is called the negative phase of an algorithm that trains an RBM.

3.2.1 Training an RBM: Contrastive Divergence

Contrastive divergence (CD) (Hinton (2002)) is a training algorithm for RBMs that uses a simple approximation of $\langle v_i h_j \rangle_{model}$. Contrastive divergence is time efficient and gives good results. It starts with a data vector from the training set and uses a step of Gibbs sampling to obtain the states of the hidden units. From the states of the hidden units, visible units are sampled. The process is repeated multiple times, obtaining reconstructions for both the visible and hidden units. An approximation to the unbiased sample of $\langle v_i h_j \rangle_{model}$, is obtained by using the values of the

reconstructions.

$$\nabla w_{ij} = \epsilon \left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstruction} \right) \tag{3.13}$$

$$\nabla a_i = \epsilon \left(\langle v_i \rangle_{data} - \langle v_i \rangle_{reconstruction} \right) \tag{3.14}$$

$$\nabla b_j = \epsilon \left(\langle h_j \rangle_{data} - \langle h_j \rangle_{reconstruction} \right) \tag{3.15}$$

Positive phase in CD

Fix the data vector on the visible units and sample from the hidden units. Use $v_i h_j$ as an unbiased sample of $\langle v_i h_j \rangle_{data}$.

Negative phase in CD

Starting with the data vector on the visible units, perform alternating steps of Gibbs sampling. Use the reconstruction of visible and hidden states as an approximation for the unbiased sample of $\langle v_i h_j \rangle_{reconstruction}$. For mini-batch learning, the samples obtained in the positive and negative phase above are averaged on the entire mini-batch before updating parameters. CD_k denotes the contrastive divergence algorithm with k alternating Gibbs sampling steps performed to obtain the reconstructions. CD_1, is most commonly used, as it is the most time efficient and gives good enough results.

3.3 Deep Belief Networks, (DBNs)

A deep belief network (DBN) is a generative graphical model, composed of multiple layers of latent variables ("hidden units"), with connections between the layers but not between units within each layer. DBNs can be viewed as a composition of simple, unsupervised networks such as restricted Boltzmann machines (RBMs) or autoencoders, where each sub-network's hidden layer serves as the visible layer for the next.

3.3.1 *Greedy Pre-Training in Deep Belief Networks*

Deep belief networks use the principle of greedy layer-wise training to initialize parameters before performing any discriminative or generative fine-tuning. They were discovered by Hinton *et al.* (2006). Like Restricted Boltzmann machines, deep belief networks are probabilistic generative models that use latent variables to learn features from the data. Unlike RBMs, they use multiple layers of hidden units, giving them a more hierarchical structure and allowing them to learn higher level representations (features of features). Deep belief nets were initially introduced using stochastic binary units, but the extensions of RBMs can be stacked together to form DBNs. Multiple hierarchical levels in an image are the pixel level, the stroke level, the edge level and the object level.

In a hierarchical model, we now want to learn the features of these features. We do this by creating another RBM, for which the input are the first set of learned features of the data (the state of the hidden units of the first RBM, when the input is a data vector) (Figure 3.3).

This process can be repeated multiple times, allowing learning of higher and higher layer of features. It can be proven that every time we add another layer, we improve the variational lower bound on the log probability of generating the data. After creating these RBMs, we have to combine them together. The recognition weights are the transpose of the generative weights and will be used for inference. Figure (3.4) exemplifies the difference between recognition and generative weights. DBNs model the joint distribution between the observable data (v) and the latent hidden variables (feature vectors h_k):

$$p(v, h_1, h_2, \cdots, h_n) = p(h_n, h_{n-1}) \prod_{k=0}^{n-2} p(h_k \big| h_{k+1}) \tag{3.16}$$

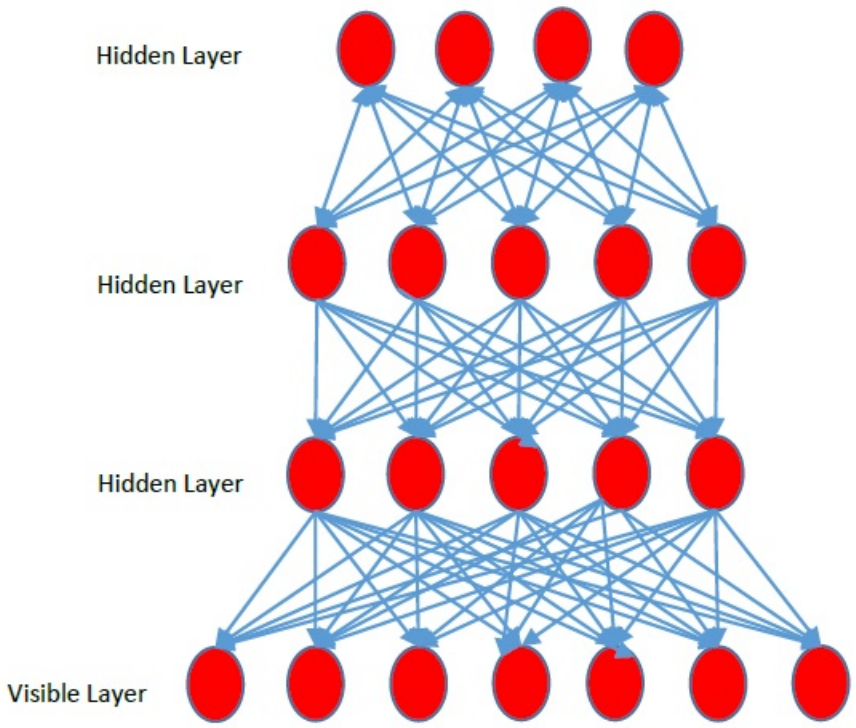


Figure 3.3: Deep Belief Network Model.

As an interesting observation, we note that DBNs are recursive: removing the visible layer of a deep belief net with more than 3 layers will result in another deep belief network, with one layer less.

3.3.1.1 Improving Greedy Pre-training for Network Architectures

In order to stack two RBMs on top of each other, the number of hidden units of the first RBM has to be equal to the number of visible units of the second RBM. This is required in order to propagate the hidden activations of the first RBM as training data for the second RBM. When the number of visible units of the first RBM is equal to the number of hidden units in the second RBM, we can use the same weights. Figure (3.5) shows such an example. If the RBM model is symmetric and the hidden

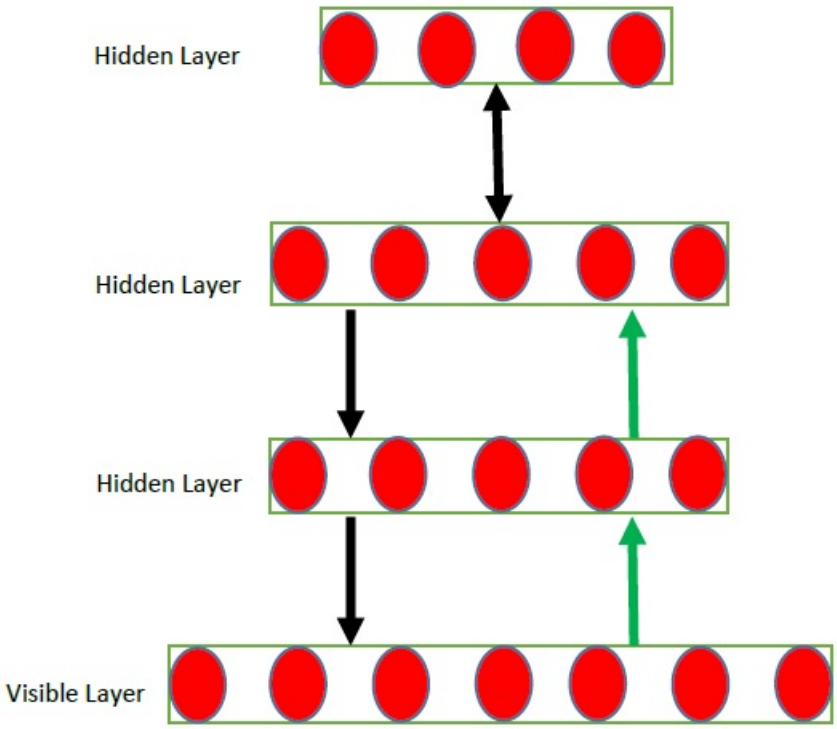


Figure 3.4: Recognition Weights (black arrows) Versus Generative Weights (green arrows)

activations of the first RBM are the input for the second one, the two networks are trying to model similar correlations, so the weights learned by the first one can be used to initialize the weights of the second RBM. This initialization works only if the RBMs are symmetric. In their initial formulation, RBMs are symmetric because both hidden and visible units used the same activation function, namely the logistic sigmoid. Recent work has shown that using Gaussian visible units along with noisy rectified linear units can improve performance of RBMs (Nair and Hinton (2010)). This type of RBM is not symmetric, due to the use of different activation functions. It is also common to scale the input data to the RBM with Gaussian visible units to have zero mean and unit variance so that it does not learn the variance of the visible

units using CD. Here, the input data for the second RBM is not given by the hidden activations of the first RBM, but are given by the scaled hidden activations. Hence the second RBM models different correlations, meaning that we should not initialize the weights of the second RBM to the ones resulted from training the first network, even though the shapes of the two networks allow it.

3.3.2 Generating Data from a DBN

Because DBNs are generative models, we would like to sample from the distributions they define. This is done as follows:(1) Get an equilibrium sample from the top level RBM (by performing alternated Gibbs sampling between the two layers). (2) Starting from the hidden nets of the top layer RBM, use the top down generative weights to perform a pass through the network.

3.3.2.1 Classification using Deep Belief Networks

DBNs can be adapted and used for classification and regression. In order to use deep nets for classification, another layer has to be added on top of the network. Usually this layer is a softmax. To train the network, we first perform the greedy

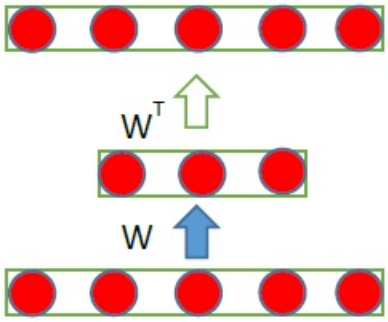


Figure 3.5: Initialization of Weights

pre-training, learning one layer of features at a time. Afterwards, we apply back propagation to the entire network, in order to learn how to discriminate between class labels. This approach eliminates a lot of the problems usually encountered with back propagation. Advantages of using back propagation after greedy pre-training:

- Back-propagation does not have to learn the features of the data. The task has been taken over by the greedy pre-training. This solves the problem of the vanishing gradient: the main aim of back propagation is to learn the weights of the top (discriminative) layer, as the weights of the first layers already have sensitive values. If the gradient is too small to affect the first layers, the impact on learning is not as drastic.

- The algorithm is less likely to get stuck in a bad local minimum of the energy function, due to the sensible initialization of weights.

- Less labelled data is needed. The greedy pre-training does not require labelled data, as it is inherently unsupervised. Labelled data is a scarce resource, as obtaining it involves manual work. Requiring less labelled data is a plus for any algorithm, as it can be given as input bigger datasets.

- Greedy pre-training causes less overfitting than just using standard back propagation, as a lot more information is obtained from the input data (namely the higher-level features which are learned in the first phase of training).

3.3.2.2 Data generation using DBNs

The wake-sleep algorithm described by Hinton *et al.* (1995) can be adapted to DBN, allowing layers to influence each other, after greedy pre-training. The aim of this is to make the network better at data generation. For this algorithm we start

differentiating between the generative and recognition weights of the network. The main steps of the algorithm are:

- Use the recognition weights to do a stochastic bottom-up pass. From the layer activities obtained, adjust the generative weights.
- Do a few iterations of sampling in the top level RBM and adjust its weights using contrastive divergence.
- Use the generative weights to do a top-down pass and use the activities to adjust the reconstruction weights.

3.3.3 Stacked RBMs and Deep Belief Networks

In an RBM, the hidden variables are independent conditionally to the visible variables, but they are not statistically independent. Stacking RBMs aims at learning these dependencies with another RBM. The visible layer of each RBM of the stack is set to the hidden layer of the previous RBM (Figure 3.6). Following the deep learning scheme, the first RBM is trained from the input instances and other RBMs are trained sequentially after that. Stacking RBMs increases a bound on the log-likelihood (Bengio *et al.* (2009)), which supports the expectation to improve the performance of the model by adding layers. A stacked RBMs architecture is a deep generative model. Patterns generated from the top RBM can be propagated back to the input layer using only the conditional probabilities as in a belief network.

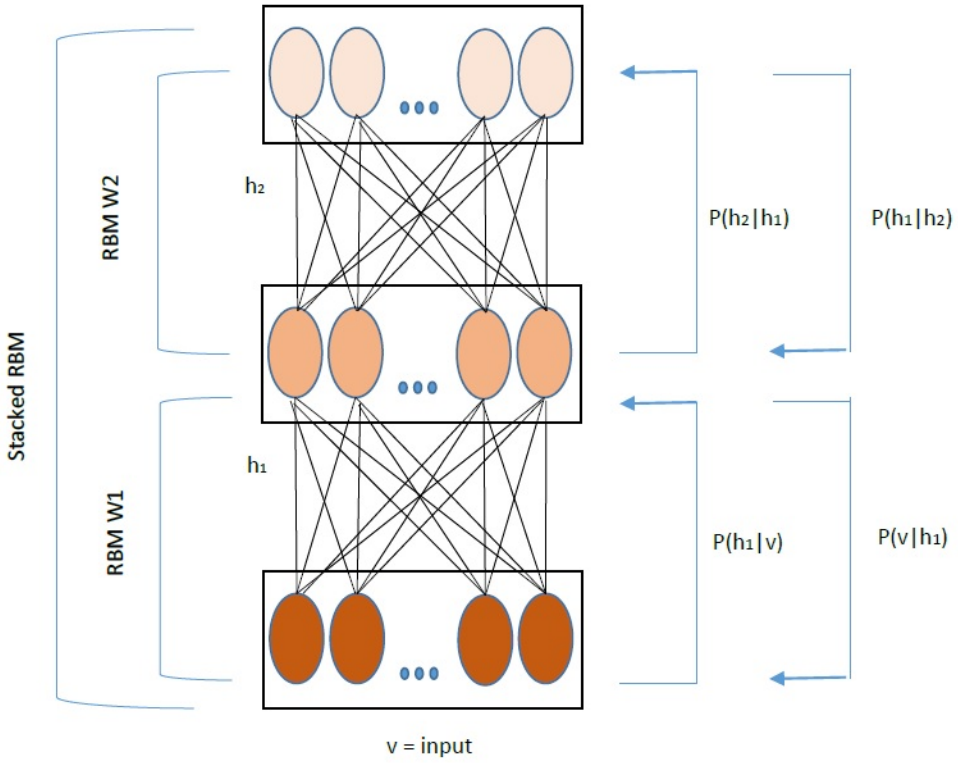


Figure 3.6: Stacked RBM Architecture

3.4 Stacked Auto-Associators, (SAs)

Another model which can be stacked in order to train a deep neural network in a greedy layer-wise manner is the Auto-Associator (AA) (Bourlard and Kamp (1988); Hinton (1990)). An AA is a two-layer neural network. The first layer is the encoding layer and the second is the decoding layer. The number of neurons in the decoding layer is equal to the networks input dimensionality. The goal of an AA is to compute a code y of an input instance x from which x can be recovered with high accuracy.

This models a two-stage approximation to the identity function.

$$f_{dec}\left(f_{enc}(x)\right) = f_{dec}y = \widehat{x} \approx x. \tag{3.17}$$

Here, f_{enc} is a function computed by the encoding layer and f_{dec} is the function computed by the decoding layer. An AA can be trained by applying standard back propagation of error derivatives. Depending on the nature of the input data, the loss function can either be the squared error L_{SE} for continuous values or the cross-entropy L_{CE} for binary vectors:

$$L_{SE}(x, \widehat{x}) = \sum_i \left(\widehat{x}_l - x_i\right)^2 \tag{3.18}$$

$$L_{CE}(x, \widehat{x}) = \sum_i \left[x_i \log \widehat{x}_l + (1 - x_i) \log\left(1 - \widehat{x}_l\right)\right] \tag{3.19}$$

The AA training method approximates the CD method of the RBM (Bengio *et al.* (2009)). Another important fact is that an AA with a nonlinear f_{enc} differs from a PCA as it is able to capture multimodal aspects of the input distribution (Japkowicz *et al.* (2000)). Similar to the parameterization in an RBM, the decoders weight matrix W_{dec} can be set to the transpose of the encoders weight matrix, i.e. $W_{dec} = W_{enc}^T$. In such a case, the AA is said to have tied weights. The advantage of this constraint is to avoid undesirable effects of the training process, such as encoding the identity function, i.e. $f_{enc}(x) = x$. This result is possible when the encoding dimensionality is bigger than the input dimensionality. An interesting variant of the AA is the Denoising Auto-Associator (DAA). A DAA is an AA trained to reconstruct noisy inputs. To achieve this goal, the instance fed to the network is not x but a corrupted version $\widehat{x}$. After training, if the network is able to compute a reconstruction $\widehat{x}$ of x with a small loss, then it is admitted that the network has learned to remove the noise in the data in addition to encode it in a different feature space (Figure 3.7). Finally, a Stacked Auto-Associator (SAA) (Bengio *et al.* (2007); Larochelle *et al.*

(2007); Vincent *et al.* (2008); Poultney *et al.* (2007)) is a deep neural network trained following the deep learning scheme: an unsupervised greedy layer-wise pre-training before a fine-tuning supervised stage (Boureau *et al.* (2008); Mirowski *et al.* (2010)).

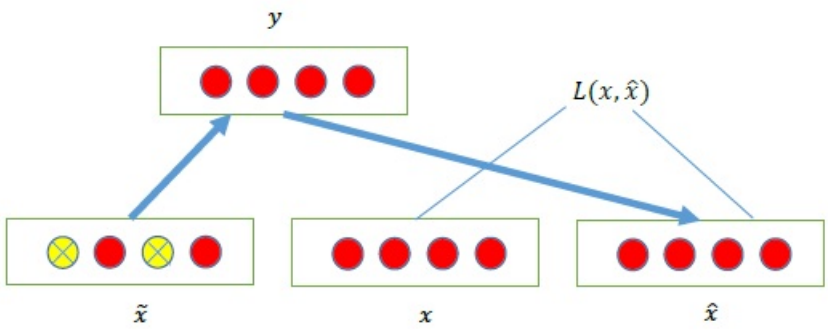


Figure 3.7: DAA Training Scheme

3.5 Convolutional Neural Networks, (CNNs)

The Convolutional Neural Network (CNN) has shown excellent performance in many computer vision and machine learning problems. CNN is useful in a lot of applications, especially in image related tasks. Applications of CNN include image classification, image semantic segmentation, object detection in images, etc. An image is classified into one of the classes based on the identity of its main object, e.g., dog, airplane, bird, etc. We first present the notation and background knowledge. Please refer to Wu (2018) for a detailed mathematical description about CNNs. The theory, figures and derivations about CNNs presented here are inspired from Wu (2018).

3.5.1 Notation

$x \in \mathbb{R}^D$ is a column vector with D elements. We use a capital letter to denote a matrix, e.g., $X \in \mathbb{R}^{H \times D}$ is a matrix with H rows and W columns. The vector x

can also be viewed as a matrix with 1 column and D rows. These concepts can be generalized to higher-order matrices, i.e., tensors. For example, $x \in \mathbb{R}^{H \times W \times D}$ is an order 3 tensor. It contains HWD elements, and each of them can be indexed by an index triplet (i, j, d), with $0 \leq i < H,\ 0 \leq j < W$ and $0 \leq D$. Another way to view an order 3 tensor is to treat it as containing D channels of matrices. Every channel is a matrix with size $H \times W$. An image with H rows and W columns is a tensor with size $H \times W \times 3$: if a color image is stored in the RGB format, it has 3 channels (for R, G and B, respectively), and each channel is a $H \times W$ matrix (second order tensor) that contains the R (or G, or B) values of all pixels. Tensors are essential in CNN. The input, intermediate representation, and parameters in a CNN are all tensors.

3.5.2 Architecture

A CNN usually takes an order 3 tensor as its input, e.g., an image with H rows, W columns, and 3 channels (R, G, B color channels). The input goes through a series of processing steps. Each processing step is called a layer. This could be a convolution layer, a pooling layer, a normalization layer, a fully connected layer or a loss layer.

$$x^1 \longrightarrow \boxed{w^1} \longrightarrow x^2 \longrightarrow \cdots \longrightarrow x^{L-1} \longrightarrow \boxed{w^{L-1}} \longrightarrow x^L \longrightarrow \boxed{w^L} \longrightarrow z \qquad (3.20)$$

The above equation illustrates the forward pass in a CNN. The input is x^1, usually an image (order 3 tensor). It goes through the processing in the first layer, which is the first box. We denote the parameters involved in the first layer's processing collectively as a tensor w^1. The output of the first layer is x^2, which also acts as the input to the second layer processing. This processing proceeds till all layers in the CNN have been finished, which outputs x^L. We add another layer for backward error propagation, a method that learns good parameter values in the CNN. Let's

suppose the problem at hand is an image classification problem with C classes. We output x^L as a C dimensional vector, whose i-th entry encodes the prediction. To make x^L a probability mass function, we can set the processing in the $(L-1)$-th layer as a softmax transformation of x^{L-1}. In other applications, the output x^L may have other forms and interpretations. The last layer is a loss layer. Let us suppose t is the corresponding ground-truth value for the input x^1, then a cost or loss function can be used to measure the discrepancy between the CNN prediction x^L and the target t.

$$z = \frac{1}{2}||t - x^L||^2, \tag{3.21}$$

This squared $L2$ loss can be used in a regression problem. In a classification problem, the cross entropy loss is used. The ground-truth in a classification problem is a categorical variable t. We first convert the categorical variable t to a C dimensional vector t. Now both t and x^L are probability mass functions, and the cross entropy loss measures the distance between them. Hence, we can minimize the cross entropy.

3.5.3 Forward Propagation

Suppose all the parameters of a CNN model $w^1, w^2, \cdots, w^{L-1}$ have been learned, then we can use this model for prediction.

Starting from the input x^1, we make it pass the processing of the first layer (the box with parameters w^1), and get x^2. In turn, x^2 is passed into the second layer, etc. Finally, we achieve $x^L \in \mathbb{R}^C$, which estimates the posterior probabilities of x^1 belonging to the C categories. We can output the CNN prediction as

$$\underset{i}{\text{argmax}} \ x_i^L \tag{3.22}$$

Note that the loss layer is not needed in prediction. It is only useful when we try to learn CNN parameters using a set of training examples.

3.5.4 Stochastic Gradient Descent, (SGD)

The parameters of a CNN model are optimized to minimize the loss z.

Let's suppose one training example x^1 is given for training such parameters. The training process involves running the CNN network in both directions. We first forward propagete to get x^L to achieve a prediction using the current CNN parameters. Instead of outputting a prediction, we compare the prediction with the target t corresponding to x^1. Finally, we achieve a loss z. The loss z is then a supervision signal, guiding how the parameters of the model should be updated. And the SGD way of modifying the parameters is:

$$w^i \longleftarrow w^i - \eta \frac{\partial z}{\partial w^i}. \tag{3.23}$$

In Equation 3.23, the $\longleftarrow$ sign implicitly indicates that the parameters w^i (of the i-layer) are updated from time t to $t+1$. If a time index t is explicitly used, this equation will look like

$$(w^i)^t = (w^i)^{t-1} - \eta \frac{\partial z}{\partial (w^i)^t} \tag{3.24}$$

In Equation 3.24, the partial derivative $\frac{\partial z}{\partial w^i}$ measures the rate of increase of z with respect to the changes in different dimensions of w^i. In order to minimize the loss function, we should update w^i along the opposite direction of the gradient. This updating rule is called the gradient descent.

In every update we only change the parameters by a small proportion of the negative gradient, controlled by η (the learning rate). $\eta > 0$ is usually set to a small number (e.g., $\eta = 0.001$). One update based on x^1 will make the loss smaller for this particular training example if the learning rate is not too large. However, it is possible that it will make the loss of some other training examples larger. Hence, we need to update the parameters using all training examples. When all training examples have

been used to update the parameters, we say one epoch has been processed.

3.5.5 Back Propagation

The last layer's partial derivatives are easy to compute. Because x^L is connected to z directly under the control of parameters w^L, it is easy to compute $\frac{\partial z}{\partial w^L}$. This step is only needed when w^L is not empty. It is also easy to compute $\frac{\partial z}{\partial x^L}$. For every layer, we compute two sets of gradients: the partial derivatives of z with respect to the layer parameters w^i, and that layer's input x^i.

- The term $\frac{\partial z}{\partial w^i}$, can be used to update the current (i-th) layer's parameters;
- The term $\frac{\partial z}{\partial x^i}$ can be used to update parameters backwards, e.g., to the $(i-1)$-th layer.

This layer-by-layer backward updating procedure makes learning a CNN much easier.

Let's take the i-th layer as an example. When we are updating the i-th layer, the back propagation process for the $(i+1)$-th layer must have been finished.

That is, we already computed the terms $\frac{\partial z}{\partial w^{i+1}}$ and $\frac{\partial z}{\partial x^{i+1}}$. Both are stored in memory and ready for use. Next, we compute $\frac{\partial z}{\partial w^i}$ and $\frac{\partial z}{\partial x^i}$. Using the chain rule, we have:

$$\frac{\partial z}{\partial (vec(w^i)^T)} = \frac{\partial z}{\partial (vec(x^{i+1})^T)} \frac{\partial (vec(x^{i+1}))}{\partial (vec(w^i)^T)} \tag{3.25}$$

$$\frac{\partial z}{\partial (vec(x^i)^T)} = \frac{\partial z}{\partial (vec(x^{i+1})^T)} \frac{\partial (vec(x^{i+1}))}{\partial (vec(x^i)^T)} \tag{3.26}$$

Since $\frac{\partial z}{\partial x^{i+1}}$ is already computed and stored in memory, it requires just a matrix reshaping operation (vec) and an additional transpose operation to get $\frac{\partial z}{\partial (vec(x^{i+1})^T)}$, which is the first term in the right hand side (RHS) of both equations. $\frac{\partial (vec(x^{i+1}))}{\partial (vec(w^i)^T)}$

and $\frac{\partial(vec(x^{i+1}))}{\partial(vec(x^i)^T)}$ are much easier to compute than directly computing $\frac{\partial z}{\partial(vec(w^i)^T)}$ and $\frac{\partial z}{\partial(vec(x^i)^T)}$, because x^i is directly related to x^{i+1}, through a function with parameters w^i.

3.5.6 ReLU Layer

A ReLU layer does not change the size of the input, that is, x^l and y share the same size. In fact, the Rectified Linear Unit (hence the name ReLU) can be regarded as a truncation performed individually for every element in the input:

$$y_{i,j,d} = \max\{0, x^l_{i,j,d}\} \tag{3.27}$$

with $0 \leq i < H^l = H^{l+1}, 0 \leq j < W^l = W^{l+1}$, and $0 \leq d < D^l = D^{l+1}$. There is no parameter learning in this layer. Based on Equation (3.27), it is obvious that

$$\frac{dy_{i,j,d}}{dx^l_{i,j,d}} = [\![x^l_{i,j,d} > 0]\!] \tag{3.28}$$

where $[\![.]\!]$ is the indicator function, being 1 if its argument is true, and 0 otherwise. Hence, we have

$$\left[\frac{\partial z}{\partial x^l}\right]_{i,j,d} = \begin{cases} \left[\frac{\partial z}{\partial y}\right]_{i,j,d} & \text{if } x^i_{i,j,d} > 0 \\ 0 \quad \text{otherwise} \end{cases} \tag{3.29}$$

Note that y is an alias for x^{l+1}. The purpose of ReLU is to increase the nonlinearity of the CNN.

3.5.7 Convolution Layer

Let the input in the l-th layer be an order 3 tensor with size $H^l \times W^l \times D^l$. A convolution kernel is also an order 3 tensor with size $H \times W \times D^l$. When we overlap the kernel on top of the input tensor at the spatial location $(0, 0, 0)$, we compute

the products of corresponding elements in all the D^l channels and add the HWD^l products to get the convolution result at this spatial location. Then, we move the kernel from top to bottom and from left to right to complete the convolution.

In a convolution layer, multiple convolution kernels are used. Let us assume D kernels are used and each kernel is of spatial span $H \times W$. The kernels are denoted as f. f is an order 4 tensor in $\mathbb{R}^{H \times W \times D^l \times D}$. Similarly, we use index variables $0 \leq i < H, 0 \leq j < W, 0 \leq d^l < D^l$ and $0 \leq d < D$ to pinpoint a specific element in the kernels. In this section, we consider the simple case when the stride is 1 and no padding is used. Hence, we have x^{l+1} in $\mathbb{R}^{H^{l+1} \times W^{l+1} \times D^{l+1}}$, with $H^{l+1} = H^l - H + 1, W^{l+1} = W^l - W + 1$, and $D^{l+1} = D$. In precise mathematics, the convolution procedure can be expressed as an equation:

$$y_{i^{l+1}, j^{l+1}, d} = \sum_{i=0}^{H} \sum_{j=0}^{W} \sum_{d^l=0}^{D^l} f_{i,j,d^l,d} \times x^l_{i^{l+1}+i, j_{l+1}+j, d^l}. \tag{3.30}$$

One of the major advantages of the convolution layer is that all spatial locations share the same convolution kernel, which greatly reduces the number of parameters needed for a convolution layer.

3.5.8 Update Parameters

As previously mentioned, we need to compute two derivatives: $\dfrac{\partial z}{\partial vec(x^l)}$ and $\dfrac{\partial z}{\partial vec(F)}$, where the first term $\dfrac{\partial z}{\partial vec(x^l)}$ will be used for backward propagation to the previous $(l-1)$-th layer, and the second term will determine how the parameters of the current (l)-th layer will be updated. A friendly reminder is to remember that f, F and w^i refer to the same thing (modulo reshaping of the vector or matrix or tensor). Similarly, we can reshape y into a matrix $Y \in \mathbb{R}^{(H^{l+1}W^{l+1}) \times D}$, then y, Y and x^{l+1} refer to the same object (again modulo reshaping).

The rule for updating the parameters in the l-th layer: the gradient with respect

to the convolution parameters is the product between $\phi(x^l)^T$ and $\frac{\partial z}{\partial Y}$.

$$\frac{\partial z}{\partial F} = \phi(x^l)^T \frac{\partial z}{\partial Y}, \tag{3.31}$$

which is a simple rule to update the parameters in the l-th layer:

3.5.9 Gradient Computation

In the l-th layer, we still need to compute $\frac{\partial z}{\partial vec(x^l)}$. For this purpose, we want to reshape x^l into a matrix $X \in \mathbb{R}^{(H^l W^l) \times D^l}$, and use these two equivalent forms (modulo reshaping) interchangeably. The chain rule states that
$\frac{\partial z}{\partial (vec(x^l)^T)} = \frac{\partial z}{\partial (vec(y)^T)} \frac{\partial vec(y)}{\partial (vec(x^l)^T)}$.

$$\frac{\partial vec(y)}{\partial (vec(x^l)^T)} = \frac{\partial (F \otimes I) vec(\phi(x^l))}{\partial (vec(x^l)^T)} = (F \otimes I) M. \tag{3.32}$$

Thus,

$$\frac{\partial z}{\partial (vec(x^l)^T)} = \frac{\partial z}{\partial (vec(y)^T)} (F \otimes I) M. \tag{3.33}$$

Therefore,

$$\frac{\partial z}{\partial (vec(x^l))} = M^T \left(\frac{\partial z}{\partial Y} F^T \right) \tag{3.34}$$

In order to pinpoint one element in $vec(x^l)$ or one row in M^T , we need an index triplet (i^l, j^l, d^l), with $0 \leq i^l < H^l, 0 \leq j^l < W^l$, and $0 \leq d^l < D^l$. Similarly, to locate a column in M^T or an element in $\frac{\partial z}{\partial Y} F^T$, we need an index pair (p, q), with $0 \leq p < H^{l+1} W^{l+1}$ and $0 \leq q < HWD^l$.

Thus, the (i^l, j^l, d^l)-th entry of $\frac{\partial z}{\partial vec(x^l)}$ equals the multiplication of two vectors: the row in M^T (or the column in M) that is indexed by (i^l, j^l, d^l), and $vec\left(\frac{\partial z}{\partial Y} F^T \right)$

Furthermore, since M^T is an indicator matrix, in the row vector indexed by (i^l, j^l, d^l), only those entries whose index (p, q) satisfies $m(p, q) = (i^l, j^l, d^l)$ have a

value 1, all other entries are 0. Thus, the (i^l, j^l, d^l)-th entry of $\dfrac{\partial z}{\partial (vec(x^l))}$ equals the sum of these corresponding entries in $vec\left(\dfrac{\partial z}{\partial Y} F^T\right)$.

In precise mathematical form, we get the following succinct equation:

$$\left[\frac{\partial z}{\partial X}\right]_{i^l,j^l,d^l} = \sum_{(p,q)\in m^{-1}(i^l,j^l,d^l)} \left[\frac{\partial z}{\partial Y} F^T\right]_{(p,q)} \tag{3.35}$$

3.5.10 Pooling layer

Let $x^l \in \mathbb{R}^{H^l \times W^l \times D^l}$ be the input to the l-th layer, which is now a pooling layer. The pooling operation requires no parameter. The spatial extent of the pooling $(H \times W)$ is specified in the design of the CNN structure. Assume that H divides H^l and W divides W^l and the stride equals the pooling spatial extent, the output of pooling (y or equivalently x^{l+1}) will be an order 3 tensor of size $H^{l+1} \times W^{l+1} \times D^{l+1}$, with

$$H^{l+1} = \frac{H^l}{H}, \quad W^{l+1} = \frac{W^l}{W}, D^{l+1} = D^l \tag{3.36}$$

A pooling layer operates upon x^l channel by channel independently. Within each channel, the matrix with $H^l \times W^l$ elements are divided into $H^{l+1} \times W^{l+1}$ non-overlapping subregions, each subregion being $H \times W$ in size. The pooling operator then maps a subregion into a single number.

Two types of pooling operators are widely used: max pooling and average pooling. In max pooling, the pooling operator maps a subregion to its maximum value, while the average pooling maps a subregion to its average value.

$$\text{max}: \quad y_{i^{l+1},j^{l+1},d} = \max_{0\le i<H,\ 0\le j<W} x^l_{i^{l+1}\times H+i, j^{l+1}\times W+j, d}, \tag{3.37}$$

$$\text{average}: \quad y_{i^{l+1},j^{l+1},d} = \frac{1}{HW} \sum_{0\le i<H,\ 0\le j<W} x^l_{i^{l+1}\times H+i, j^{l+1}\times W+j, d}, \tag{3.38}$$

where $0 \leq i^{l+1} < H^{l+1}, 0 \leq j^{l+1} < W^{l+1}$ and $0 \leq d < D^{l+1} = D^l$

3.5.11 Fully Connected Layer

One benefit of the convolution layer is that convolution is a local operation. The spatial extent of a kernel is often small (e.g., 3×3). One element in x^{l+1} is usually computed using only a small number of elements in its input x^l.

A fully connected layer refers to a layer if the computation of any element in the output x^{l+1} (or y) requires all elements in the input x^l. A fully connected layer is sometimes useful at the end of a deep CNN model. Most often, after many convolution, ReLU and pooling layers, the output of the current layer contains distributed representations for the input image. We want to use all these features in the current layer to build features with stronger capabilities in the next one. A fully connected layer is useful for this purpose.

Let the input of a layer x^l have size $H^l \times W^l \times D^l$. If we use convolution kernels whose size is $H^l \times W^l \times D^l$, then D such kernels form an order 4 tensor in $H^l \times W^l \times D^l \times D$. The output is $y \in \mathbb{R}^D$. It is obvious that to compute any element in y, we need to use all elements in the input x^l. Hence, this layer is a fully connected layer, but can be implemented as a convolution layer. Hence, we do not need to derive learning rules for a fully connected layer separately.

3.6 Recurrent Neural Networks, (RNNs)

A recurrent neural network (RNN) is a class of artificial neural networks where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. There are two types of recurrent networks - a finite impulse and infinite impulse RNN.

A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that cannot be unrolled. Both finite impulse and infinite impulse recurrent networks have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of Long short-term memory (LSTM) and gated recurrent units. The theory, figures and explanation about RNNs and LSTMs presented here are inspired from Colah (2018).

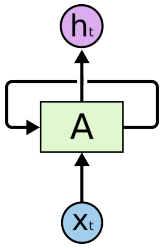


Figure 3.8: Recurrent Neural Network; Source: Colah (2018)

In Figure (3.8), A represents the neural network, x_t is its input and h_t is the output's value. The loop allows information to be passed from one step of the network to the next. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to their successor. Figure (3.9) shows an unrolled network.

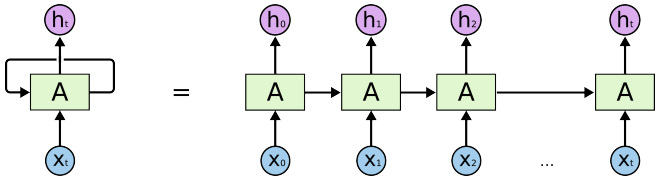


Figure 3.9: An Unrolled RNN; Source: Colah (2018)

This chain-like nature makes RNNs the natural architecture to use for temporal data. In the recent years, RNNs have achieved great success in a variety of problems: speech recognition, language modeling, translation, image captioning.

3.6.0.1 The Issue of Long-Term Dependencies

In a temporal sequence, sometimes, we only look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. When trying to predict the last word in the clouds are in the sky, we do not need any further context. It is pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it is needed is small, RNNs can learn to use the past information. Figure (3.10) shows such a situation.

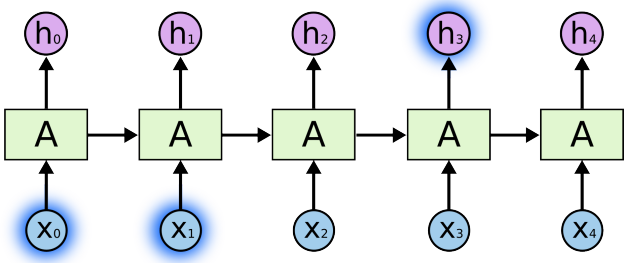


Figure 3.10: Short - Term Dependency in RNN; Source: Colah (2018)

In cases where we need more context, the gap between the relevant information and the point where it is needed, becomes very large. For example, consider trying to predict the last word in the text, I grew up in France I speak fluent French. Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

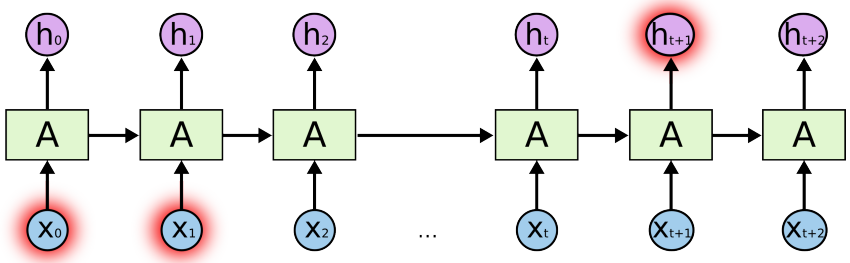


Figure 3.11: Long - Term Dependency in RNN; Source: Colah (2018)

3.6.1 Long Short-Term Memory Networks, (LSTMs)

Long Short-Term Memory networks (LSTMs) are a special kind of RNNs, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997). They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem, as remembering information for long periods of time is built in their architecture.

3.6.1.1 The Core Idea Behind LSTMs

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram in Figure 3.12. The cell state is like a conveyor belt, it runs straight down the entire chain, with some minor linear interactions. The LSTM can add or remove information to the cell state. This is regulated by structures called gates. Gates can optionally let information through. They are composed of a sigmoid neural net layer and a pointwise multiplication operation (refer Figure (3.13)). All the above figures were taken from Colah (2018), please refer to the *url* for more details.

The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through. A value of zero means allow nothing through, while a value of one means allow everything through. An LSTM has three such gates that

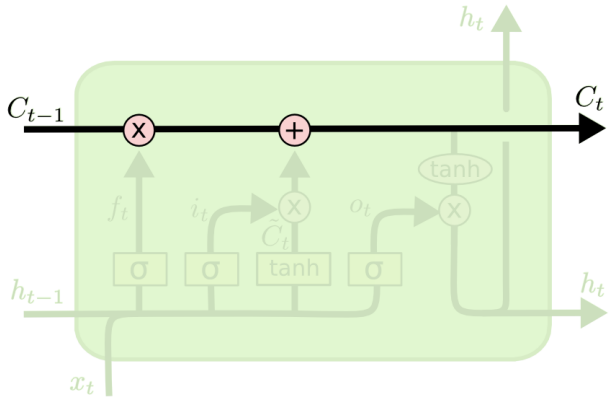


Figure 3.12: LSTM Cell State; Source: Colah (2018)

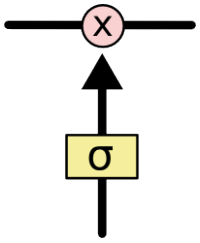


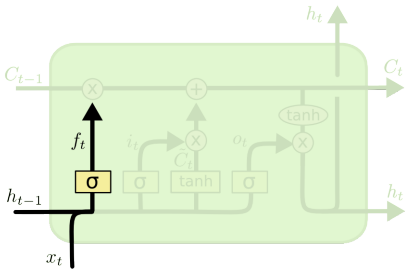
Figure 3.13: LSTM Gate Operation; Source: Colah (2018)

help protect and control the cell state.

3.6.2 LSTM Equations

The LSTM network first decides what information to forget from the current cell state. This decision is made by a sigmoid layer called the *forget gate layer*. It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents *completely retain* while a 0 represents *completely forget*. Figure 3.14 shows the LSTM forget gate along with the equation.

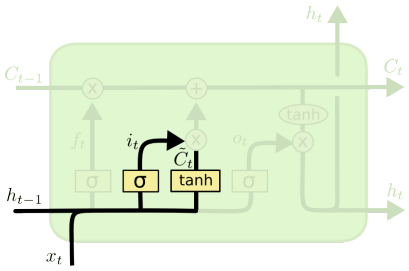
The next step is to decide what new information is to be stored in the cell state. This has two parts. First, a sigmoid layer called the *input gate layer* decides which values need to be updated. Next, a *tanh layer* creates a vector of new candidate values, $\tilde{C}_t$, that are added to the state. In the next step, these are combined to create



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 3.14: LSTM Forget Gate; Source: Colah (2018)

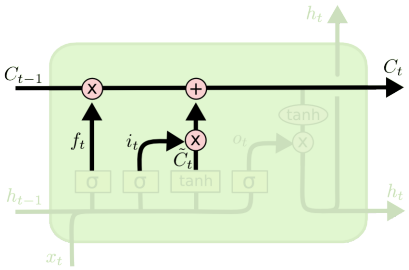
an update to the state as shown in Figure 3.15. The corresponding equations are also given in the figure.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 3.15: LSTM Input Gate; Source: Colah (2018)

Next, the old cell state C_{t-1} is updated into the new cell state C_t. The old state is multiplied by f_t. Then, we add it to $i_t \star \tilde{C}_t$. This is the new candidate value. Figure (3.16) shows this operation along with the equations.

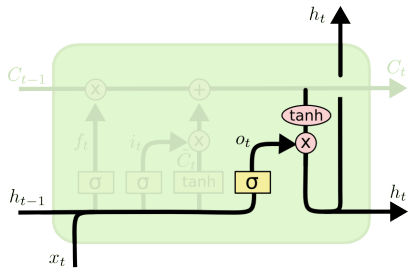


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 3.16: Updated Cell State; Source: Colah (2018)

Finally, the LSTM decides what to output. This output is dependent on the cell state. First, a sigmoid layer is employed which decides what parts of the cell state is

output. Then, the cell state is put through tanh (to push the values to be between 1 and 1) and multiplied by the output of the sigmoid gate (Figure (3.17)).



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 3.17: LSTM Output Gate; Source: Colah (2018)

The LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3.6.3 Backpropagation Through Time, (BPTT)

Backpropagation Through Time, or BPTT, is the application of the Backpropagation training algorithm to recurrent neural network applied to sequence data like a time series. At each time step, a recurrent neural network uses one input and predicts one output. Conceptually, BPTT works by unrolling all input timesteps. Each timestep has one input timestep, one copy of the network, and one output. Errors are then calculated and accumulated for each timestep. The network is rolled back up and the weights are updated. Spatially, each timestep of the unrolled recurrent neural network may be seen as an additional layer given the order dependence of the problem and the internal state from the previous timestep is taken as an input on the subsequent timestep. We can summarize the algorithm as follows:

1. Present a sequence of timesteps of input and output pairs to the network.
2. Unroll the network, then calculate and accumulate errors across each timestep.
3. Roll-up the network and update weights.
4. Repeat.

BPTT can be computationally expensive as the number of timesteps increases. If input sequences are comprised of thousands of timesteps, then this will be the number of derivatives required for a single weight update. This can cause weights to vanish or explode (go to zero or overflow) and make slow learning and model skill noisy.

Truncated Backpropagation Through Time, or TBPTT, is a modified version of the BPTT training algorithm for recurrent neural networks where the sequence is processed one timestep at a time and periodically (k_1 timesteps) the BPTT update is performed back for a fixed number of timesteps (k_2 timesteps). We can summarize the algorithm as follows:

1. Present a sequence of k_1 timesteps of input and output pairs to the network.
2. Unroll the network, then calculate and accumulate errors across k_2 timesteps.
3. Roll-up the network and update weights.
4. Repeat

The TBPTT algorithm requires the consideration of two parameters:

k_1: The number of forward-pass timesteps between updates. Generally, this influences how slow or fast training will be, given how often weight updates are performed.
k_2: The number of timesteps BPTT is applied. Generally, it should be large enough to capture the temporal structure in the problem. Too large a value results in vanishing gradients.

3.7 Summary

This chapter described popular deep learning architectures like Restricted Boltzmann Machines (RBMs), Deep Belief Networks (DBNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). It gave details about the corresponding network architectures and training methodologies along with tips and tricks for their implementation. Extensions of these models will be used in later chapters for different computer vision applications.

Chapter 4

DEEP MODELS FOR MULTIMODAL EMOTION RECOGNITION

Multimodal emotion recognition from images has been a field of intense research for many years. The difficulty of the problem lies in its interdisciplinary nature. Psychology, neuroscience and machine learning concern themselves with how we can teach computers to detect emotions in humans. Emotion recognition can be divided in two main tasks: feature extraction and emotion classification. We tackle the problem by using deep belief networks, a type of neural network that allows both feature detection and classification.

In this chapter, we first present the emoFBVP database of multimodal (face, body gesture, voice and physiological signals) recordings of actors enacting various expressions of emotions. The database consists of audio and video sequences of actors displaying three different intensities of expressions of 23 different emotions along with facial feature tracking, skeletal tracking and the corresponding physiological data. Next, we describe four deep belief network (DBN) models and show that these models generate robust multimodal features for emotion classification in an unsupervised manner. Our experimental results show that the DBN models perform better than the state of the art methods for emotion recognition. Finally, we propose convolutional deep belief network (CDBN) models that learn salient multimodal features of expressions of emotions. Our CDBN models give better recognition accuracies when recognizing low intensity or subtle expressions of emotions when compared to state of the art methods.

We then propose two multi-modal deep auto-associator for learning audio and video emotion data. Our model handles large amounts of unlabeled data effectively

and fuses multiple data modalities to form unified representations that captured features useful for emotion recognition. Our intra-modal audio-only and video-only models are able to effectively capture correlations across shallow representations between multiple modalities.

Finally, the chapter presents a study to investigate the effect of transfer of emotion-rich features between source and target networks on percentage emotion classification accuracy and training time. We make three interesting contributions. First, we propose emosource - a 6-layer DBN trained on multimodal and unimodal emotion corpora for emotion classification. Second, we propose emotarget and emotarget$_{\text{ft}}$ DBN models and study the transfer of features between emosource and these networks in a layer-by-layer manner. Finally, we experimentally show that our emotarget model achieves reasonably comparable classification accuracies to that of emosource with significantly shorter training times when the transferred features are left frozen while our emotarget$_{\text{ft}}$ model achieves a performance boost over emosource model with similar training times as the source network when the entire network is trained on the target dataset. In short, our emotarget and emotarget$_{\text{ft}}$ models successfully repurpose the emotion-rich features learned by the source model to train the target models and achieve shorter training times and performance boosts respectively. This makes our study extremely useful in a practical setting. To the best of our knowledge, this is the first research approach to studying the effect of transfer of emotion features in a layer-by-layer manner in a multimodal setting.

4.1 Database for Holistic Emotion Recognition

To contribute to this need for holistic emotional databases, we present the emoF-BVP database of multimodal (face, body gesture, voice and physiological signals) recordings of participants enacting various expressions of emotions. The Microsoft

Kinect sensor was used for facial feature tracking, skeletal tracking of the body, and recording vocal expressions. The Zephyr BioHarness was used to capture physiological signals and wrist-worn accelerometers were used to capture movement activity. The database consists of 1380 samples of audio and video sequences of people displaying various intensities of expressions of emotions along with facial feature tracking, skeletal tracking and corresponding physiological data. The richness in human emotional expressiveness poses both a technological as well as research challenge. Obtaining multimodal sensor data is a challenge in itself. Different modalities of measurements require different equipment, developed and manufactured by different companies, and different expertise to set up and operate them. Interdisciplinary knowledge and technological solutions to combine measurement data from different sensor equipment are necessary to create a multimodal emotion database. Emotion recognition using facial features is a challenging problem on its own. It is only now that facial expression recognition is reaching commercialization, which makes this the most opportune time to create a publicly available multimodal emotion database. To contribute to this need for multimodal emotional databases, we have recorded natural responses of participants to affective emotion labels using four different modalities- facial expressions, body expressions, vocal expressions and physiological signals. Along with the video and audio sequences, we provide facial feature tracking and skeletal tracking data. The database is freely available to the academic community and is easily accessible through a web-interface http://www.emoFBVP.org . The recordings of all the data are rated through an evaluation form completed by participants immediately after each excerpt of acting emotions. The recordings of this database are synchronized to enable researchers to study simultaneous emotional responses using all the channels. A summary of the emoFBVP database is given in Table (4.1).

It is evident from the discussion in Section (2.1) that there is no single database

Table 4.1: emoFBVP Emotion Database Properties

Number of Subjects	10
Recorded Modalities	Face and body video and audio using Microsoft Kinect. Physiological signals using Zephyr Bio Harness and Acceleration data using wrist worn Accelerometers.
Data	Face Tracking data, Skeletal Tracking data, Heart Rate, Breathing Rate, ECG, R-R interval, Posture, Activity Level, Acceleration
Evaluation	Affective Communication Skill assessment in each modality. Confidence in expressing expressions overall and in each modality. Ease of expression in each modality (Participant's Self Rating)
Number of Emotion Labels	23
Number of Intensities of Expression per Emotion Label	3
Number of Audio, Video Sequences per Subject **Number of Audio, Video Sequences per Standing and Seated Sessions**	69 690
Total Number of Audio, Video Sequences in Database	1380

that has recordings of varying intensities of expression of emotion in multiple modalities recorded simultaneously. The emoFBVP database presented in this dissertation has recordings of facial expressions, body gestures, vocal expressions, physiological signals and activity data along with facial and skeletal tracking data. Ten participants were involved in data capture, and every participant displayed 23 different emotions. Recordings of each emotion were done six times: three in a standing position and three in a seated position when the body gestures and facial expressions were tracked and recorded along with vocal expressions, physiological data and activity respectively. Therefore, the database provides six examples of each of the 23 emotions in varying intensities of expression. The two sessions of recordings (standing and seated) are independent of each other. This makes it possible to use our database for unimodal (using only face, body, physiological signals or activity), bimodal (face & voice, body

& voice, etc.) and multimodal emotion recognition studies. Our database provides information about the affective communication skills of every participant. It also provides evaluation details about the confidence of expression of emotion, intensity of expression of emotion and level of ease of expression of emotion using facial expressions, body gestures and vocal expressions. Our database, therefore, provides both valuable expression data and metadata that will contribute to the ongoing development of emotion recognition algorithms and systems as well as to studies on human emotion and emotion expression. The affective computing community will greatly benefit from the large collection of modalities recorded.

4.2 emoFBVP Database

This section gives details about the apparatus and setup employed for collection of data, the data capture method and other important properties of the database, and details about the metadata that is provided.

4.2.1 Apparatus and Setup For Data Collection

The emoFBVP database consists of responses of participants to affectively stimulating emotion labels. Different modalities of measurement require different equipment. We set up apparatus to record face videos, facial feature tracking, body gesture videos, skeletal tracking, vocal expressions and physiological signals simultaneously. The sensor equipment used to facilitate the recording of the aforementioned modalities includes Microsoft Kinect Sensor, Zephyr BioHarness and wrist-worn accelerometers. Details about each of these sensors are given in Table (4.2).

Figure (4.1) shows a picture of the equipment used along with their component parts labeled. All equipment produced time stamped data to synchronize the data

Table 4.2: Apparatus Used for Data Capture for Multi Modal Emotional Expression

Apparatus	**Modality**	**Software**	Comments
Microsoft Kinect for Windows Sensor	Body gestures Facial, Vocal Expressions	Brekel Kinect Pro Body and Face	• Body gestures: Captured using Skeletal tracking feature • Facial Expressions: Captured using Facial fiducial tracking feature • Vocal Expressions: Captured using Kinect's Microphone array
Zephyr BioHarness	Physiological signals	BioHarness Log Downloader	• Signals Captured: Heart Rate, R-R Interval, Breathing Rate, Posture, Activity Level and Peak Acceleration • FDA approved device, unobtrusive and comfortable
Wrist worn Accelerometers	Activity Level	Custom software for extracting and Storing Acceleration Data	• Measures Acceleration using Triple Axis Accelerometers (LilyPad Accelerometer ADXL 335) • Discreet and unobtrusive

post-recording. The subjects were instructed to include a clap at the start of data capture to help improve data synchronization.

4.2.2 Data Capture Procedure

Recruitment: Participants were recruited after a city-wide call for people who have completed basic coursework in acting/non-verbal communication. They were requested to provide their formal consent to participate by signing a consent form that gave a detailed description of the purpose and data capture procedure of the study.

Figure 4.1: Equipment Used for Data Capture. Left: Microsoft Kinect for Windows Sensor With Labeled Components. Middle: Zephyr BioHarness With Labeled Component Parts. Right: Wrist-Worn Accelerometers. Best Viewed in Color

Participant Information and Assessment: Participants were asked to provide their age range, gender and ethnic background. They answered questions to help assess their affective communication skills. In particular, each participant rated his/her overall skill in expressing emotions, their affective communication skill using facial expressions, body gestures, vocal expressions and how emotionally expressive they were (on a scale of 1 to 5, where 5 was very effective). This information is provided in the database as part of metadata.

Data Capture: Participants were instructed to perform/express emotions using facial expressions, body gestures and vocal expressions naturally. They were specifically instructed not to exaggerate while expressing the emotion to facilitate capture of natural expressions. Their natural responses (using face, body, voice and physiological) were recorded for 23 emotion labels: Happy, Sad, Anger, Disgust, Fear, Surprise, Boredom, Interest, Agreement, Disagreement, Neutral, Pride, Shame, Tri-

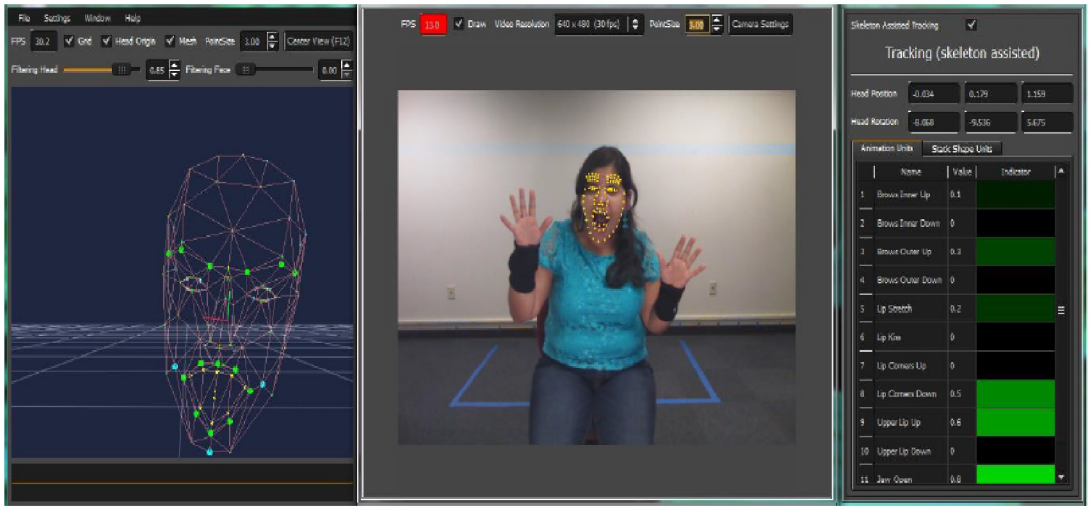


Figure 4.2: Snapshot of a Subject Portraying Emotion, Surprise. Left: 3D Face-Mesh Corresponding to the Emotion. Middle: Brekel Kinect Pro Face Tracking Animation and Shape Units Shown as Yellow Dots Over the SubjectS Face. Right: Tracking Indicator Showing Presence or Absence of Animation Units at Each Instant. Best Viewed in Color, Source: Ranganathan *et al.* (2016a)

umphant, Defeat, Sympathy, Antipathy, Admiration, Concentration, Anxiety, Frustration, Content and Contempt during two sessions. During the first session, participants expressed each of the 23 emotions in three varying intensities of expression in a standing position; their body gestures were recorded and their skeletal representation was tracked. During the second session, participants expressed each of the 23 emotions in three varying intensities of expression in a seated position; their facial expressions were recorded and facial features were tracked. Physiological signals, vocal expressions and acceleration were recorded continuously during both sessions. After recording their responses to each emotion label, they filled out an evaluation form. Here, they provided details about their level of comfort in acting/expressing emotions in each modality. They were asked to rate their confidence level with expressing each emotion, their intensity while expressing each emotion and their ease of

expressing each emotion using facial expressions, body gestures and vocal expressions on a scale of 1 (low) to 5 (high). Participants were given about 2 minutes between expressing different emotions. They used this time to complete the evaluation form and think about their responses to the next emotion label. Further, participants were requested to share their comments and feedback about the data capture process. This information is also made available in the database as part of metadata.

4.2.3 Properties of emoFBVP Database

Face and Voice: We obtained facial expression video sequences and facial tracking data from the Microsoft Kinect for Windows sensor. All video sequences were recorded at the rate of 30 fps with a video resolution of 640×480 pixels. The sequences are of variable length lasting between 600 and 2000 frames. We used Brekel Kinect Pro Face software to record 3D face tracking data obtained from the Kinect sensor. The face tracking data consists of 3D head position and rotation information, 3D coordinates for 11 animation units and 3D coordinates for 11 shape units for each frame of the video. Table (4.3) lists the animation and shape units that were tracked. This data is provided in .txt and.csv formats (.bvh, .daz .pz2 and .fbx formats also available for 3D modeling). Figure (4.2) shows a snapshot of a subject portraying emotion, Surprise along with a 3D face-mesh corresponding to the emotion. The animation and shape units tracked are shown as yellow dots over the subjects face and an indicator shows their presence or absence.

The voice data was recorded using Microsoft Kinect for Windows sensor. The Kinect sensor includes a four-element linear microphone array that captures audio data at 24-bit resolution. This allows accuracy across a wide dynamic range of voice data. The sensor enables high quality audio capture with focus on audio coming from a particular direction with beamforming. The audio sequences are provided in

standard .wav format and are synchronized with the face and body sequences.

Table 4.3: Animation Units and Shape Units from Face Tracking data

Number	**Animation Unit**	**Shape Unit**
1	Brows Inner Up	Head Height
2	Brows Inner Down	Eyebrows Vertical Position
3	Brows Outer Up	Eyes Vertical Position
4	Brows Outer Down	Eyes Width
5	Lip Stretch	Eyes Height
6	Lip Kiss	Eyes Separation Distance
7	Lip Corners Up	Nose Vertical Position
8	Lip Corners Down	Mouth Vertical Position
9	Upper Lip Up	Mouth Width
10	Upper Lip Down	Eyes Vertical Difference
11	Jaw Open	Chin width

Body: We obtained body expression video sequences and skeletal tracking data from the Microsoft Kinect for Windows sensor. All video sequences were recorded at the rate of 30 fps with a video resolution of 640×480 pixels. The sequences are of variable length and synchronized with the face and audio data. We used the Brekel Kinect Pro for Body software to record the skeletal tracking data. The skeletal tracking data provides 3D coordinates of twenty joints of the users body along with 3D coordinates for hand, foot and head rotations for each frame of the video sequence. Figure (4.3) lists the twenty joints that were tracked. The data is available in both .txt and .csv formats (.bvh, .daz .pz2 and .fbx formats also available for 3D modeling).

One of the best ways to validate the authenticity of a new emotion database is

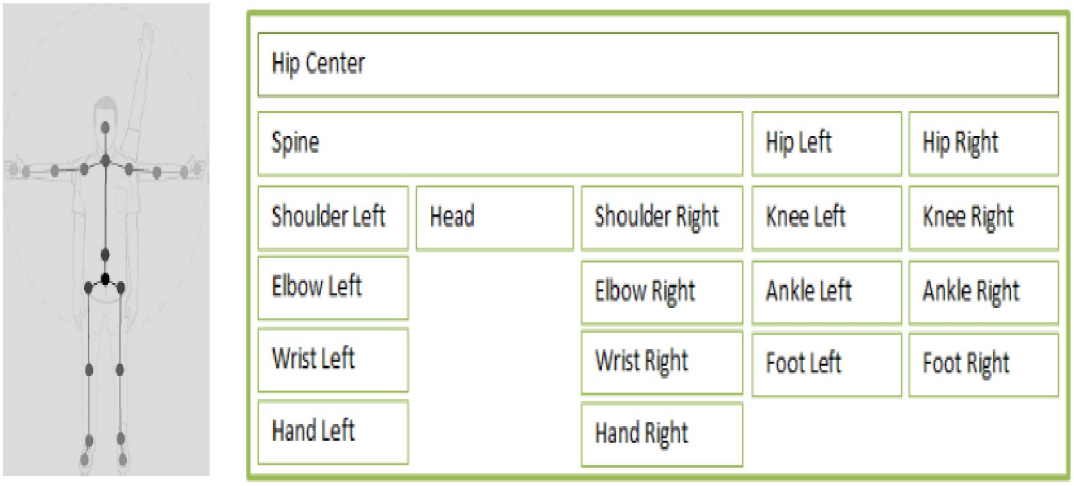


Figure 4.3: Skeletal Tracking and Joint Hierarchy - 20 Bone Joints Are Tracked.

to apply known methods of feature extraction and investigate the performance of state of the art models on the collected data. In order to show the usefulness of our emoFBVP database, we apply extensions of known deep learning techniques for feature learning and investigate the performance accuracies for emotion recognition in unimodal, bimodal and multimodal scenarios. Our database also provides facial feature tracking and skeletal tracking data. We investigate the advantages of adding these features to the deep models using feature selection methods. Kim *et al.* (2013) developed a suite of deep belief network models that showed improvements in emotion classification performance over baselines that do not use deep learning. They perform rigorous experiments to show that deep learning techniques can be used for multimodal emotion recognition. We build extensions of these models, train them on our multimodal data, perform similar experiments and investigate the usefulness of the emoFBVP database for emotion recognition in the following sections.

4.2.4 Conclusions

We presented the emoFBVP database of multimodal recordings of actors enacting various expressions of emotions. This is one of the first emotion datasets that has recordings of varying intensities of expressions of emotions in multiple modalities recorded simultaneously. We strongly believe that the affective computing community will greatly benefit from the large collection of modalities recorded.

4.3 Deep Belief Networks for Emotion Recognition

In statistical machine learning, a major issue is the selection of an appropriate feature space where input instances have desired properties for solving a particular problem. For example, in the context of supervised learning for binary classification, it is often required that the two classes are separable by a hyperplane. In the case where this property is not directly satisfied in the input space, one is given the possibility to map instances into an intermediate feature space where the classes are linearly separable. This intermediate space can either be specified explicitly by hand-coded features, be defined implicitly with a so-called kernel function, or be automatically learned. In both of the first cases, it is the users responsibility to design the feature space. This can incur a huge cost in terms of computational time or expert knowledge, especially with highly dimensional input spaces, such as when dealing with images. As for the third alternative, automatically learning the features with deep architectures, i.e. architectures composed of multiple layers of nonlinear processing, can be considered as a relevant choice. Indeed, some highly nonlinear functions can be represented much more compactly in terms of number of parameters with deep architectures than with shallow ones (e.g. SVM). Unfortunately, training deep architectures is a difficult task and classical methods that have proved effective when applied to shallow architec-

tures are not as efficient when adapted to deep architectures. Adding layers does not necessarily lead to better solutions. For example, the more the number of layers in a neural network, the lesser the impact of the back-propagation on the first layers. The gradient descent then tends to get stuck in local minima or plateaus (Bengio *et al.* (2007)), which is why practitioners have often preferred to limit neural networks to one or two hidden layers. This issue has been solved by introducing an unsupervised layer-wise pre- training of deep architectures (Hinton *et al.* (2006)). More precisely, in a deep learning scheme each layer is treated separately and successively trained in a greedy manner: once the previous layers have been trained, a new layer is trained from the encoding of the input data by the previous layers. Then, a supervised fine-tuning stage of the whole network can be performed.

The deep learning paradigm tackles problems on which shallow architectures (e.g. SVM) are affected by the curse of dimensionality. As part of a two-stage learning scheme, involving multiple layers of non- linear processing, a set of statistically robust features are automatically extracted from the data.

Deep belief networks (DBNs) are probabilistic generative models that stand in contrast to the discriminative nature of traditional neural nets (Bengio *et al.* (2007)). Generative models provide a joint probability distribution over observable data and labels, facilitating the estimation of both $P(Observation|Label)$ as well as $P(Label|Observation)$. DBNs address problems encountered when back-propagation is applied to deeply-layered neural networks; namely:

1. necessity of a substantial labelled data set for training;

2. slow convergence times;

3. inadequate parameter selection techniques that lead to poor local optima (Arel *et al.* (2010)).

DBNs are composed of several layers of Restricted Boltzmann Machines (RBMs), a type of neural network (see Figure 4.4). These networks consist of a single visible layer and single hidden layer, where connections are formed between the layers (Wi, j) (units within a layer are not connected). The hidden units (h) are trained to capture higher-order data correlations that are observed at the visible units (v). An RBM defines a probability distribution p on data vectors v as follows:

$$p(v) = \sum_{h} \frac{e^{-E(v,h)}}{\sum_{u,g} e^{-E(u,g)}}. \tag{4.1}$$

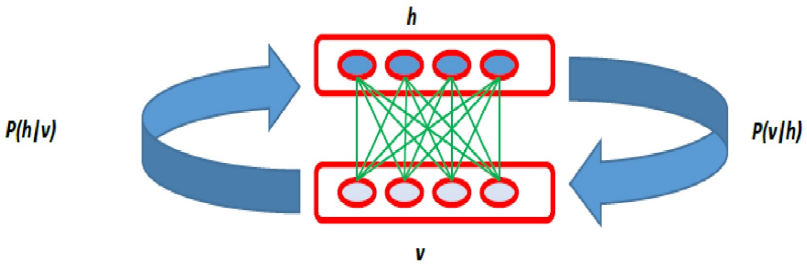


Figure 4.4: The RBM Architecture With Visible (V) and Hidden (H) Layers.

Here, the variable v is the input vector and h corresponds to unobserved features that are hidden units not available in the original dataset (Ghahramani (2004)). A RBM defines a joint probability on both the observed and the unobserved variables which are referred to as visible and hidden units respectively. The distribution is then marginalized over the hidden units to give a distribution over the visible units. The probability distribution is defined by an energy function E, defined over couples (v, h) of binary vectors by:

$$E(v, h) = -\sum_{i} a_i v_i - \sum_{j} b_j h_j - \sum_{i,j} w_{i,j} v_i h_j \tag{4.2}$$

Here, a_i and b_j are the biases associated with v_i and h_j respectively and $w_{i,j}$ are the weights of the pairwise interaction between v_i and h_j. The energy function above is crafted to make the conditional probabilities $p(v|h)$ and $p(h|v)$ tractable.

$$p(v\big|h) = \prod_i p(v_i\big|h) \text{ and } p(h\big|v) = \prod_j p(h_j\big|v)$$

$$\begin{aligned} p(v_i = 1\big|h) &= sigm\left(a_i + \sum_j h_j w_{i,j}\right) \\ p(h_j = 1\big|v) &= sigm\left(b_j + \sum_i v_i w_{i,j}\right) \end{aligned} \tag{4.3}$$

Here $sigm(x) = \frac{1}{1+e^{-x}}$ is the logistic activation function which is a special case of the more general sigmoid function. The above formulation models the visible variables as real valued units and hidden variables as binary units. As it is intractable to compute the gradient of the log-likelihood term, we learn the parameters of the model using contrastive divergence (Hinton (2002)). We have already seen a detailed description of the architecture and equations of the RBM in Chapter 3.

4.3.1 Multimodal Emotion Recognition Model

In this section, we focus on applying deep architectures for multimodal emotion recognition using face, body, voice and physiological signal modalities. We apply extensions of known DBN models for multimodal emotion recognition using the emoF-BVP database and investigate recognition accuracies to validate the utility of the database for emotion recognition tasks. To the best of our knowledge, the use of DBNs for multimodal emotion recognition of data comprising of all the modalities (facial expressions, body gestures, vocal expressions and physiological signals) has not been explored by the affective research community. Recent developments in deep learning techniques exploit the use of single layer building blocks called as Restricted

Boltzmann Machines (RBMs) (Hinton *et al.* (2006)) to build DBNs in an unsupervised manner. DBNs are constructed by greedy layerwise training of stacked RBMs to learn hierarchical representations from the multimodal data (Bengio *et al.* (2007)). RBMs are undirected graphical models that use binary latent variables to represent the input. Like Kim *et al.* (2013), we also use Gaussian RBMs for training the first layer of the network. The visible units of the first layer are real-valued. The deeper layers are trained using Bernoulli-Bernoulli RBMs that employ visible and hidden units that are binary valued.

Here, we present a suite of deep models to investigate audio-visual feature learning for multimodal emotion recognition. Later, we extend the model to learn audio-visual and physiological signal features for emotion recognition. Our baseline is a Support Vector Machine that uses subsets of the original feature space selected using supervised and unsupervised feature selection.

We compare unsupervised feature learning (DBN) and supervised feature selection. We first build an unsupervised two-layer DBN, enforcing multi-modal learning as introduced by Ngiam *et al.* (2011), we call it DemoFV. Here we use facial expression and vocal expressions as features, hence the name DemoFV. We form two other models by adding supervised features to augment DemoFV. For one model, we add features before pre-training the DBN and for the other model, we add the features after DBN pre-training. This helps us compare between feature learning exclusively from emotional salient subset of original features and reduction in learned feature space in a supervised context. We then compare this to the performance of a three-layer 3DemoFV model. We also form DemoBV that uses body gestures and vocal expressions as features, DemoFBV that uses face, body and vocal expressions and DemoFBVP that uses face, body, vocal expressions and physiological signals of emotions.

The results provide important insight into feature learning methods for multimodal emotion data. The results show that the Demo* models outperform the baseline models. Further, the results demonstrate that the three layer 3Demo* models outperform the two-layer Demo* models for low intensity expressions of emotions. This suggests that unsupervised feature learning can be used in lieu of supervised feature selection for emotion data. In addition, the relative performance improvement of the three-layer model for subtle expressions of emotions suggests that these complex feature relationships are particularly important for identifying low intensity emotional cues. This is an important finding given the challenges inherent in and needed for recognizing emotions elicited in realistic scenarios.

4.3.2 Unsupervised Feature Learning

DBNs learn hierarchical representation from data and can be effectively constructed by greedy training and stacking multiple RBMs. RBMs are undirected graphical models that represent the density of input data using binary latent variables.

In this dissertation, we use Gaussian RBMs that employ real-valued visible units for training the first layer of the DBNs. We use Bernoulli-Bernoulli RBMs that employ binary visible and hidden units for training the deeper layers. In a Gaussian RBM, the joint probability distribution and energy function of v and h is as follows:

$$p(v,h) = \frac{1}{Z}e^{-E(v,h)} \tag{4.4}$$

$$E(v,h) = \frac{1}{2\sigma^2}\sum_i v_i^2 - \frac{1}{\sigma^2}\left(\sum_i c_i v_i + \sum_j b_j h_j + \sum_{i,j} v_i W_{i,j} h_j\right) \tag{4.5}$$

where $c \in \mathbb{R}^D$ and $b \in \mathbb{R}^K$ are the biases for visible and hidden units respectively and $W \in \mathbb{R}^{D\times K}$ are weights between visible units and hidden units, σ is a hyperparameter, and Z is a normalization constant. The conditional probability distribu-

tions of the Gaussian RBM are as follows:

$$P\left(h_j = 1 \middle| v\right) = sigmoid\left(\frac{1}{\sigma^2}\left(\sum_i W_{i,j} v_i + b_j\right)\right) \tag{4.6}$$

$$P\left(v_i \middle| h\right) = \mathcal{N}\left(v_i; \sum_j W_{i,j} h_j + c_i, \sigma^2\right) \tag{4.7}$$

The posteriors of the hidden units given visible units form the generated features used in the classification framework. The parameters of the RBM (W, b, c) are learned using contrastive divergence as in (Hinton (2002)). We use sparsity regularization to penalize a deviation of expected activation of the hidden units from a low fixed level p. Given a training set $\{v^{(1)}, v^{(2)}, \cdots, v^{(m)}\}$, we include a regularization penalty as in (Lee *et al.* (2008)).

4.3.3 Supervised Feature Selection

Here, we discuss the feature selection techniques that are used extensively in emotion research including: Information Gain (IG), and Principal Component Analysis (PCA). These techniques are either supervised (forward selection and IG) or use representations based on the linear dependencies between the original features (PCA). IG based feature selection methods are also commonly used in emotion recognition (Polzehl *et al.* (2009); Mower *et al.* (2011)). This method ranks features by calculating the reduction in the entropy of class labels given knowledge of each feature. Both forward selection and IG methods require labeled data during the feature selection process. PCA and its variants (e.g., Principal Feature Analysis, or PFA (Lu *et al.* (2007))) are broadly used in the emotion recognition literature Steidl *et al.* (2005); Metallinou *et al.* (2010); Wöllmer *et al.* (2010)) . PCA finds a linear projection of the base feature set to a new feature space where the new features are uncorrelated. PFA is an extension of PCA. It clusters the data in the PCA space and returns final features closest to the center of each cluster. This results in a feature set that maintains

an approximation of the variance of the original set, while minimizing correlations between features. We use IG for our proposed deep learning feature selection methods, and IG and PFA for the baseline models.

4.3.4 Feature Extraction - emoFBVP Database

The database has recordings of facial expressions, body gestures, vocal expressions, physiological signals and activity data along with facial and skeletal tracking data and intensity of expression of emotions. The *emoFBVP* database allows the study of the relation between simultaneous emotion-related activity and behavior in addition to unimodal, bimodal and multimodal emotion recognition studies. The ground truth of the data was labeled by three evaluators. We only consider utterances with labels from the following set: Angry, Happy, Sad, Disgust, Fear, Surprise and Neutral. We divide the data into three types:

1. Ideal data (complete agreement on the affective state from evaluators),
2. non-ideal data (majority agreement),
3. a combined set of these two data types.

The audio features available include both prosodic and spectral features, such as pitch, energy and mel-frequency filter banks (MFBs). MFBs have been shown to be better discriminative features than mel-frequency cepstral coefficients (MFCCs) in emotion recognition (Busso *et al.* (2007)). The original video features are facial tracking and skeletal tracking points provided by the Brekel Software. The final features are statistical functions of the raw audio-visual and physiological signal features. These include mean, variance, lower and upper quantiles, and quantile range. The features are normalized on a per-person basis to avoid person dependency (Mower *et al.* (2011)).

4.4 Experiments

We pre-train the DBN models (unsupervised) and search for the best hyper-parameters including sparsity parameters and the number of final output nodes. We select our hyper-parameters using cross validation over the training data. We use leave-one-person-out cross validation to ensure that the models are not overtraining to the affective styles of an individual. We fix the number of hidden nodes of the two-layer DBNs, the sigma parameter for the first-layer Gaussian RBMs, and the L2 regularization parameter. We select the best hyper-parameters for each data type: ideal, non-ideal and combined. We use Unweighted Accuracy (UA) for the results (Schuller *et al.* (2012)).

4.4.1 Baseline Model

We propose two baseline models. These models are two SVMs with radial basis function (RBF) kernels. The SVMs do not use features generated via deep learning techniques. We train seven emotion-specific binary SVMs in a self vs. other approach. The final emotion class label is assigned by identifying the model in which the test point is maximally far from the hyperplane similar to Smolensky (1986). The first SVM baseline model uses IG for supervised feature selection (Duch *et al.* (2002)) and the second SVM baseline model uses PFA (Lu *et al.* (2007)) for unsupervised feature selection. IG is applied to each emotion class, resulting in seven sets of emotion-specific features. Each emotion-specific SVM uses the associated emotions specific feature subset. We optimized the baselines using leave-one-subject-out cross validation for each data type (ideal, non-ideal, and combined data).

4.4.2 *DemoFV DBN Models*

We experiment with four different DBN models in order to explore different non-linear dependencies between audio and video features. We also assess the utility of feature selection methods in deep architectures (Figure 5.4). Our basic DBN is a two-layer model and is a building block for the other DBN models. It learns the audio features and video features separately in the first hidden layer. The learned features from the first layer are concatenated and used as the input to the second hidden layer. We call this the DemoFV model (Figure 4.5(a)). The other three DBN models (Figure 4.5(b, c, d)) use this model as their building block. The four models are defined as follows:

1. D*emoFV* is a basic two-layer DBN model.

2. f+D*emoFV* is a two-layer DBN with feature selection prior to the training of DemoFV.

3. D*emoFV+f* is a two-layer DBN with feature selection added post training of DemoFV.

4. 3D*emoFV* is a three-layer DBN that stacks an additional RBM on the second-layer RBM output nodes of DemoFV model.

4.4.3 *Results for DemoFV DBN*

A summary of the emotion classification results can be seen in Table (4.4). All DBN models outperform the baseline models. The two baseline models perform comparably. The DBN models for ideal data achieve accuracies ranging from 82.98% (D*emoFV*) to the maximum of 86.56% (3D*emoFV*). The performance gap between the maximum accuracies of the proposed models and maximum accuracies of the

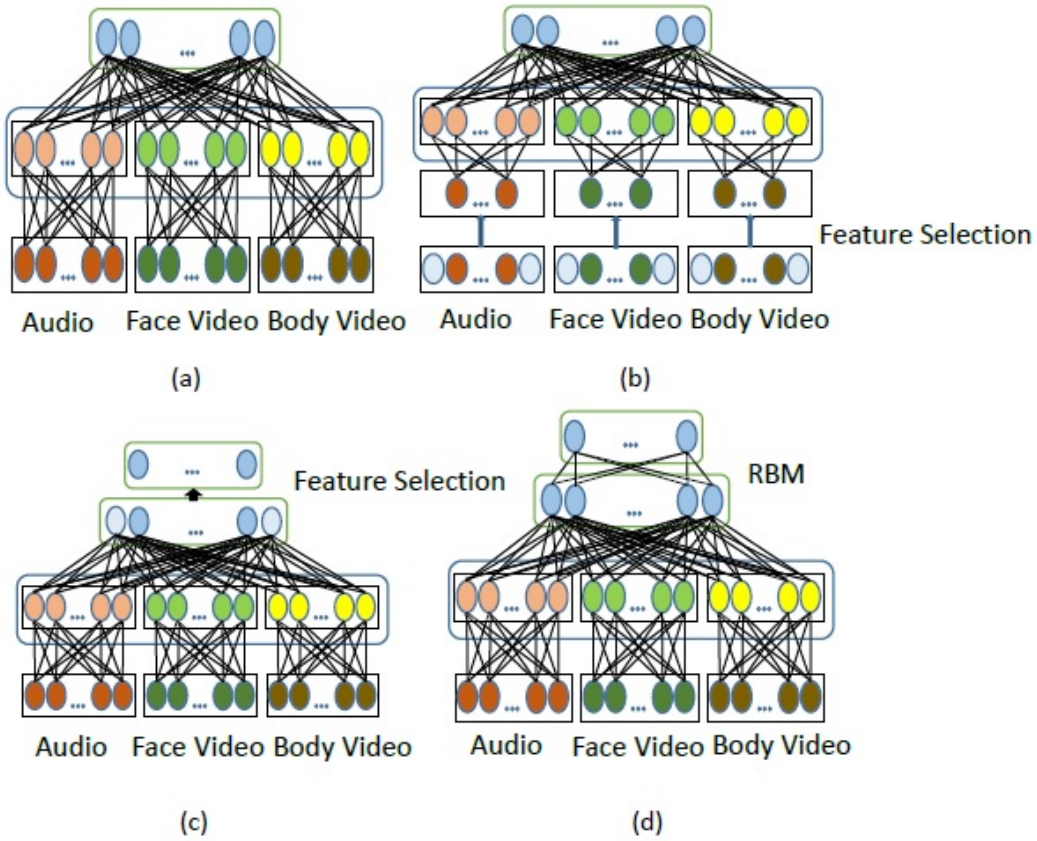


Figure 4.5: Illustration of Proposed D*emoFV* Models: (A) D*emoFV*, (B) F+D*emoFV*, (C) D*emoFV+F*, and (D) 3D*emoFV* , Source: Ranganathan *et al.* (2016a)

Table 4.4: Classification Accuracy (%) for D*emoFV* Models

Data Type	Baseline IG	Baseline PFA	D*emoFV*	D*emoFV+f*	*f*+D*emoFV*	3D*emoFV*
Ideal	86.32	82.33	82.98	84.92	84.56	86.56
Non-Ideal	64.78	64.95	65.52	65.82	65.21	66.41
Combined	75.64	75.83	77.25	78.32	77.78	77.62

baseline models is 0.24%. The IG baseline does better than the PFA baseline by 3.99%. This may suggest that in emotionally clear data, supervised feature selection is preferable to unsupervised feature selection (PFA). The 3D*emoFV* model outperforms unsupervised feature selection (PFA baseline) by 4.23%, highlighting the potential importance of feature learning rather than unsupervised feature reduction for emotionally clear data. The accuracy of the 3D*emoFV* model indicates that unsupervised feature learning can achieve comparable performance to supervised feature selection for emotionally clear data.

The DBN models for the non-ideal data achieve accuracies ranging from 65.21% (*f*+D*emoFV*) to 66.41. (3D*emoFV*). The performance gaps between the maximum accuracies of proposed models and baseline models range from 1.63% to 1.46%. We obtain a slight performance gain when using 3D*emoFV* compared to both *f* +D*emoFV* and D*emoFV+f* for subtle or non-ideal data (0.59% and 1.2% increase, respectively). We know that 3D*emoFV* uses unsupervised feature learning (unlabeled data) while *f* +D*emoFV* model learns a new set of features from a previously identified subset of emotionally salient features and D*emoFV+f* model performs feature selection at the output. Therefore, this proves that we can effectively use unsupervised feature learning for emotion recognition instead of supervised feature selection, even for subtle emotions (non-ideal data).

The DBN models for the combined data achieve accuracies ranging from 77.25% (D*emoFV*) to 78.32% (D*emoFV+f*). The performance gap between the maximum accuracies of proposed models and the IG and PFA baselines are 2.68% and 2.49% respectively.

We observe that even with unsupervised learning methods, DBNs can be used to generate audio-visual features for emotion classification. The comparison of the classification performances between the baseline and the proposed DBN models demon-

strate that it is important to retain complex non-linear feature relationships in emotion classification tasks. The performance gain is strongest when using non-ideal data. This is a very useful result when building automatic emotion recognition systems where discriminating between different intensities of emotions is required.

4.4.4 DemoBV DBN Models

We propose D*emoBV* DBN models similar to the models in Section (4.4.2). We experiment with four different DBN models to explore different non-linear dependencies between audio and video features from body gestures. Our basic DBN is a two-layer model and is a building block for the other DBN models. The four models are shown in (Figure (4.6) (a), (b), (c), (d)) and defined as follows:

1. D*emoBV* is a basic two-layer DBN model.

2. *f*+D*emoBV* is a two-layer DBN with feature selection prior to the training of DemoBV.

3. D*emoBV*+*f* is a two-layer DBN with feature selection added post training of DemoBV.

4. 3D*emoBV* is a three-layer DBN that stacks an additional RBM on the second-layer RBM output nodes of D*emoBV* model.

4.4.5 Results for DemoBV DBN

From Table (4.5), we see that all DBN models outperform the baseline models. The two baseline models perform comparably.

The DBN models for ideal data achieve accuracies ranging from 80.78% (D*emoBV*) to the maximum of 84.99% (3D*emoBV*). The performance gap between the maxi-

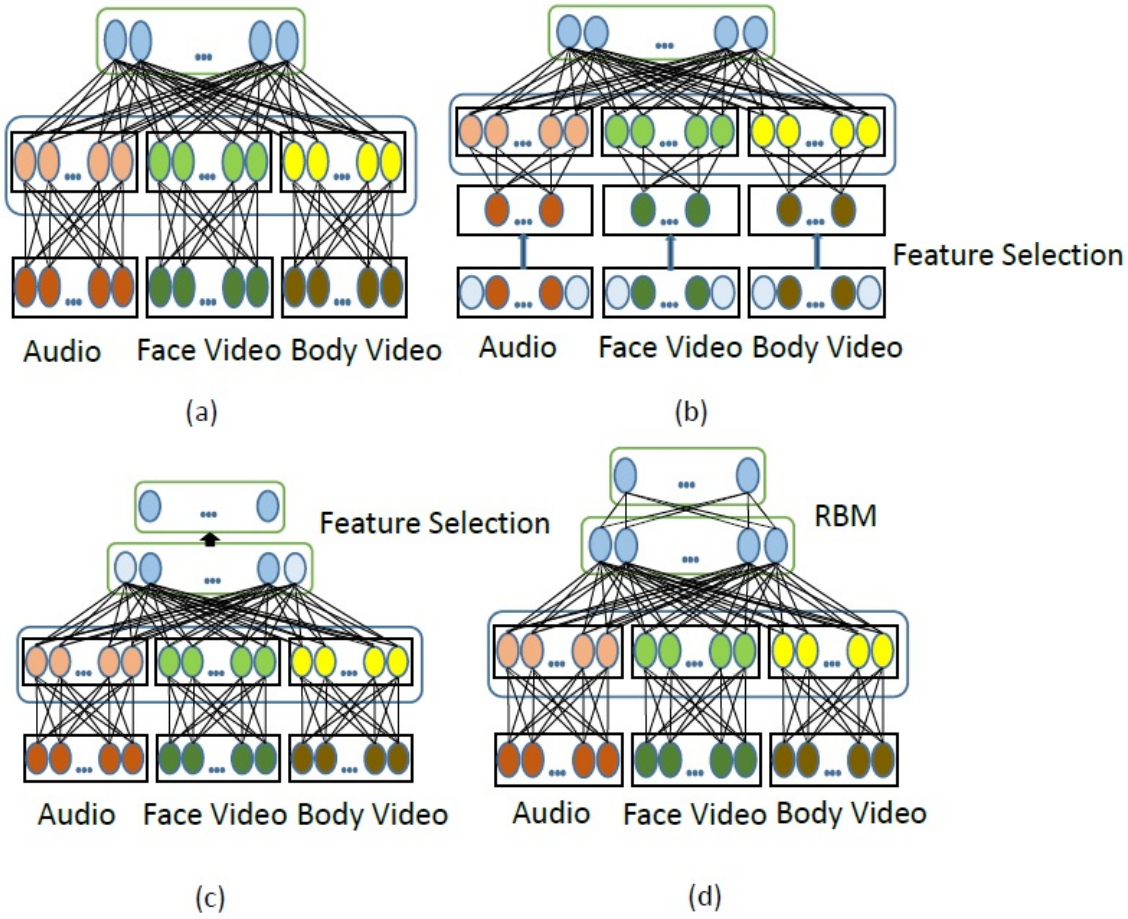


Figure 4.6: Illustration of Proposed D*emoBV* Models: (A) D*emoBV*, (B) *f*+D*emoBV*, (c) D*emoBV*+*f*, and (d) 3D*emoBV*, Source: Ranganathan *et al.* (2016a)

Table 4.5: Classification Accuracy (%) for D*emoBV* Models

Data Type	Baseline IG	Baseline PFA	D*emoFV*	D*emoFV*+*f*	*f*+D*emoFV*	3D*emoFV*
Ideal	84.22	80.25	80.78	82.88	82.46	84.99
Non-Ideal	62.66	62.86	63.67	63.89	63.42	64.64
Combined	73.64	73.83	75.25	76.32	75.78	75.62

mum accuracies of the proposed models and maximum accuracies of the baseline models is 0.77%. The IG baseline outperforms the PFA baseline by 3.97%. This again suggests that in emotionally clear data, supervised feature selection is preferable to unsupervised feature selection. However, it is interesting to note the accuracy of the 3D*emoBV*. This model indicates that unsupervised feature learning can achieve comparable performance to supervised feature selection for emotionally clear data. Further, the 3D*emoBV* outperforms unsupervised feature selection (PFA baseline) by 4.74%, further highlighting the potential importance of feature learning rather than unsupervised feature reduction for emotionally clear data.

The DBN models for the non-ideal data achieve accuracies ranging from 63.42% (*f* +D*emoBV*) to 64.64% (3D*emoBV*). The performance gaps between the maximum accuracies of proposed models and baseline models range from 1.98% to 1.78%. We again obtain a slight performance gain when using 3D*emoBV* compared to both *f* +D*emoBV* and D*emoBV+f* for subtle or non-ideal data (1.22% and 0.75% increase, respectively).

The DBN models for the combined data achieve accuracies ranging from 75.25% (D*emoBV*) to 76.32% (D*emoBV+f*). The performance gap between the maximum accuracies of proposed models and the IG and PFA baselines are 2.68% and 2.49% respectively.

We have obtained very similar results as D*emoFV* DBNs. We further evaluate the performance when using data from facial features from face and body, vocal expressions and physiological signals in the following sections.

4.4.6 DemoFBV DBN Models

We propose DemoFBV DBN models similar to the models in Section 4.4.5 and 4.4.6. We experiment with four different DBN models to explore different non-linear

dependencies between audio and video features of face and body. The four models are shown in (Figure 4.7 (a), (b), (c), (d)) and defined as follows:

1. D*emoFBV* is a basic two-layer DBN model.

2. f+D*emoFBV* is a two-layer DBN with feature selection prior to the training of D*emoFBV*.

3. D*emoFBV+f* is a two-layer DBN with feature selection added post training of D*emoFBV*.

4. 3D*emoFBV* is a three-layer DBN that stacks an additional RBM on the second-layer RBM output nodes of D*emoFBV* model.

4.4.7 Results for DemoFBV DBN

As seen from Table (4.6), all DBN models outperform the baseline models. The two baseline models perform comparably. The DBN models for ideal data achieve accuracies ranging from 83.10% (D*emoFBV*) to the maximum of 86.68% (3D*emoFBV*). The performance gap between the maximum accuracies of the proposed models and maximum accuracies of the baseline models is 0.26%. The IG baseline outperforms the PFA baseline by 3.99%. The 3DemoFBV outperforms unsupervised feature selection (PFA baseline) by 4.25%, further highlighting the potential importance of feature learning rather than unsupervised feature reduction for emotionally clear data. The DBN models for the non-ideal data achieve accuracies ranging from 68.34% (*f* +D*emoFBV*) to 69.54% (3D*emoFBV*). The performance gaps between the maximum accuracies of proposed models and baseline models range from 4.65% to 3.79%.

The DBN models for the combined data achieve accuracies ranging from 77.38% (D*emoFBV*) to 78.45% (D*emoFBV+f*). The performance gap between the maximum

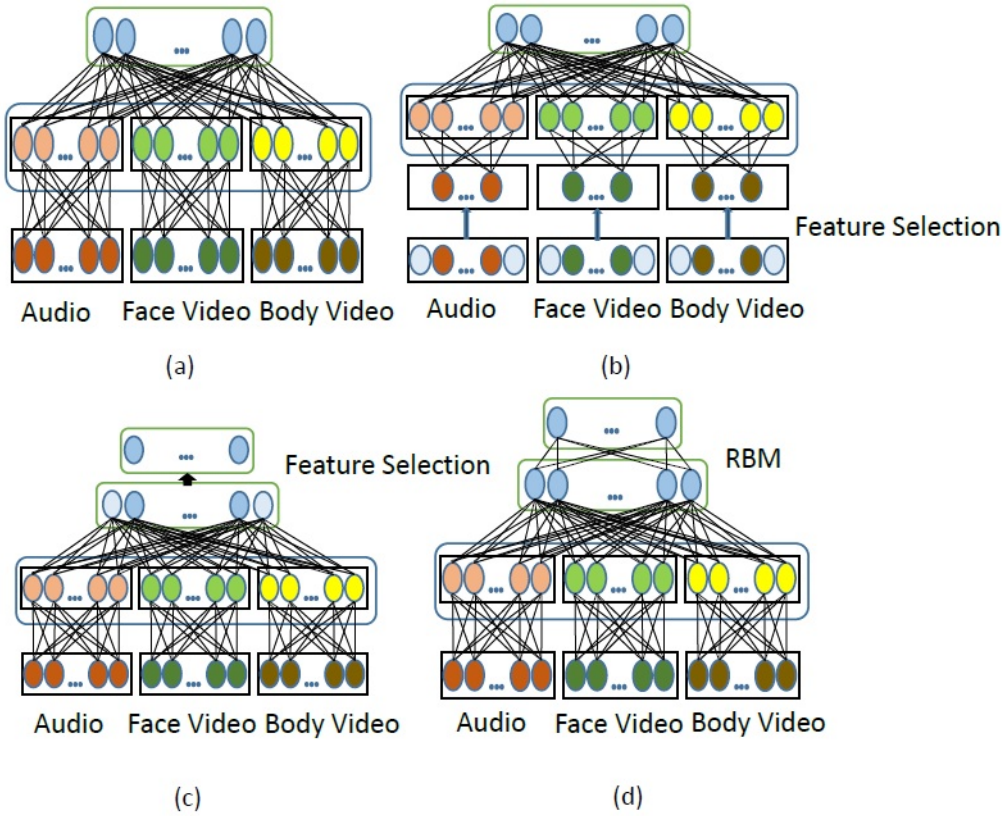


Figure 4.7: Illustration of Proposed D*emoFBV* Models: (A) D*emoFBV*, (B) *f*+D*emoFBV*, (c) D*emoFBV+f*, and (d) 3D*emoFBV*, Source: Ranganathan *et al.* (2016a)

Table 4.6: Classification Accuracy (%) for D*emoFBV* Models

Data Type	Baseline IG	Baseline PFA	D*emoFBV*	D*emoFBV+f*	*f*+D*emoFBV*	3D*emoFBV*
Ideal	86.42	82.43	83.10	84.99	84.68	86.68
Non-Ideal	64.89	65.75	68.66	68.93	68.34	69.54
Combined	75.77	75.90	77.38	78.45	77.89	77.78

accuracies of proposed models and the IG and PFA baselines are 2.68% and 3.45% respectively.

We have obtained very similar results as D*emoFV* and D*emoBV* DBNs. We further evaluate the performance when using data from facial features, vocal expressions and physiological signals in the following sections.

4.4.8 DemoFBVP DBN Models

We experiment with four different DBN models to explore different non-linear dependencies between audio, video features of face and body and physiological features. Our basic DBN is a two-layer model and is a building block for the other DBN models. The four models are shown in (Figure 4.8 (a), (b), (c), (d)) and defined as follows:

1. D*emoFBVP* is a basic two-layer DBN model.

2. f+D*emoFBVP* is a two-layer DBN with feature selection prior to the training of D*emoFBVP*.

3. D*emoFBVP+f* is a two-layer DBN with feature selection added post training of D*emoFBVP*.

4. 3D*emoFBVP* is a three-layer DBN that stacks an additional RBM on the second-layer RBM output nodes of DemoFBVP model.

4.4.9 Results for DemoFBVP DBN

As seen from Table (4.7), all DBN models outperform the baseline models. The two baseline models perform comparably. The DBN models for ideal data achieve accuracies ranging from 86.20% (D*emoFBVP*) to the maximum of 90.10% (3D*emoFBVP*).

.

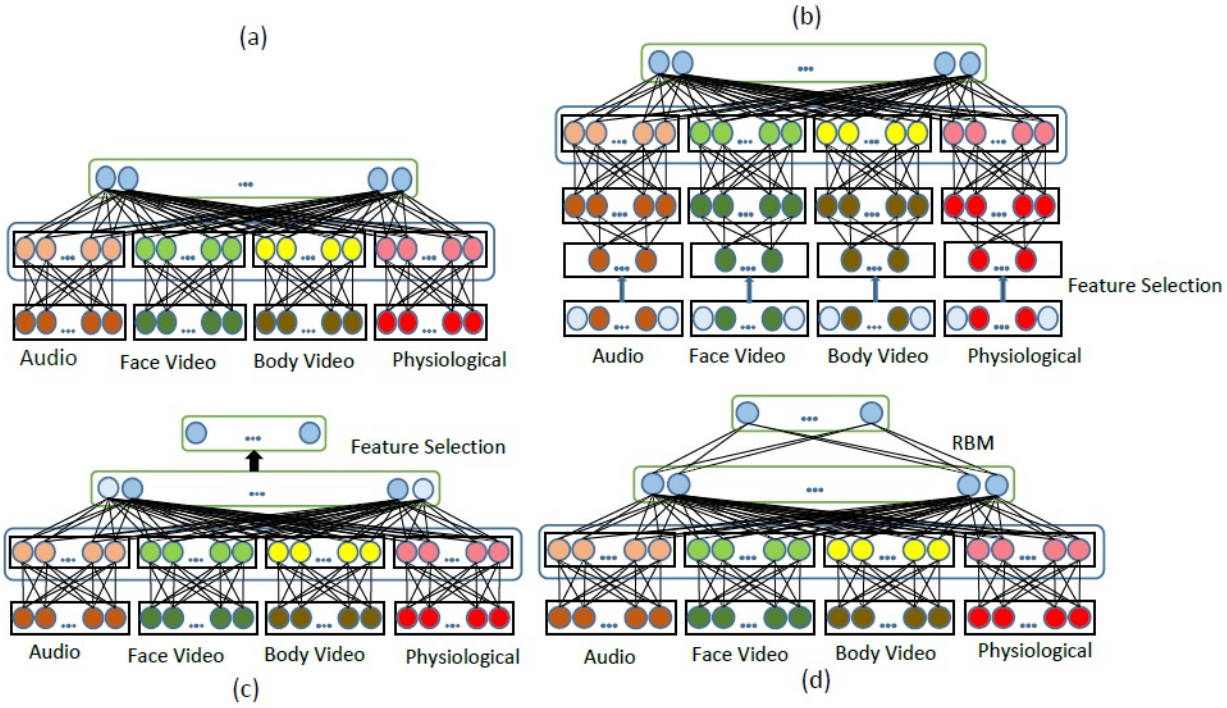


Figure 4.8: Illustration of Proposed Models: (A) D*emoFBVP*, (B) F+D*emoFBVP*, (C) D*emoFBVP*+f, and (D) 3D*emoFBVP*, Source: Ranganathan *et al.* (2016a)

Table 4.7: Classification Accuracy (%) for D*emoFBVP* Models

Data Type	Baseline IG	Baseline PFA	D*emoFBVP*	D*emoFBVP+f*	*f*+D*emoFBVP*	3D*emoFBVP*
Ideal	89.41	85.33	86.20	87.82	87.52	90.10
Non-Ideal	68.89	68.71	71.14	71.84	71.22	73.11
Combined	79.82	79.90	82.28	83.10	82.54	82.40

The performance gap between the maximum accuracies of the proposed models and of the baseline models is 0.69. The DBN models for the non-ideal data achieve accuracies ranging from 71.14% (*f* +D*emoFBVP*) to 73.11% (3D*emoFBVP*). The performance gaps between the maximum accuracies of proposed models and baseline

models range from 4.22% to 4.4%.

The DBN models for the combined data achieve accuracies ranging from 82.28% (D*emoFBVP*) to 83.10% (D*emoFBVP+f*). The performance gap between the maximum accuracies of proposed models and the IG and PFA baselines are 3.28% and 3.2% respectively.

4.4.10 Results on Standard Emotion Corpora

We compare our models to the SVM baseline we explained in earlier sections for each modality. Tables (4.8), (4.9), (4.10) and (4.11) give emotion recognition accuracies while using unimodal (facial, vocal, physiological expressions of emotions) and multimodal DBN models (multimodal expressions of emotions).

Table 4.8: Emotion Recognition Using Facial Expressions

Database	SVM Baseline	D*emoF*	3D*emoF*
Cohn Kanade	95.4 %	95.9 %	**96.3** %

Table 4.9: Emotion Recognition Using Vocal Expressions

Database	SVM Baseline	D*emoV*	3D*emoV*
Mind Reading	90.62%	92.1 %	**92.87** %

Table 4.10: Emotion Recognition Using Physiological Data

Database	SVM Baseline	D*emoP*	3D*emoP*
DEAP	78.6%	78.8 %	**79.2** %

Table 4.11: Emotion Recognition Using Multimodal Data

Database	SVM Baseline	D*emoFBVP*	3D*emoFBVP*
MAHNOB-HCI	52.4%	53.1 %	**54.8** %

To depict generalizability, we use the Cohn Kanade, MindReading, DEAP and MAHNOB-HCI databases to evaluate respective performances. These databases are very popular and are standard datasets used by the affective research community for

emotion recognition. We observe that our deep models perform better than the SVM baselines in both unimodal and multimodal scenarios.

4.4.11 Conclusions

Our results show that we can successfully employ DemoDBN models for the task of multimodal emotion recognition. The proposed DemoDBN models successfully retain complex non-linear feature relationships that exist between the different modalities for ideal, non-ideal and combined data types (as shown by the performance accuracies achieved). Our results highlight the importance of feature learning using deep architectures over unsupervised feature selection for bimodal and multimodal emotion classification using the emoFBVP database of facial expressions, body gestures, vocal expressions and physiological signals. Our experimental results showed that our D*emoDBN* models perform better than the state of the art methods for emotion recognition using popular emotion corpora. This validated the use of our *emoFBVP* database for multimodal emotion recognition studies. The affective computing community will benefit from the collection of modalities recorded.

4.5 Convolutional Deep Belief Networks for Emotion Recognition

In this section, we describe our multimodal Convolutional Deep Belief Network (CDBN) model and investigate their usability to recognize subtle or low intensities of expressions of emotions. Convolutional RBMs are an extension of regular RBMs. These are inspired by convolutional neural nets and rely on convolution and weight sharing. When convolutional RBMs are stacked together, they form convolutional deep belief networks. Convolutional DBNs are solely generative models that are trained in a greedy layer-wise manner. Here, the input is fed into the networks and the features learned by the last layer are fed to a Support Vector Machine (SVM).

In CRBMs, the network's visible layer is a matrix, instead of a vector. This enables the network to understand the spatial proximity of the pixels, leading to more robust feature learning (when compared to regular RBMs).

4.5.1 Results for CDBN Models

We used primary expressions of emotion of the lowest intensity from the *emoF-BVP*, Cohn-Kanade, Mind Reading, DEAP and MAHNOB-HCI databases.

Table 4.12: Emotion Recognition Using *emoFBVP* Database

SVM Baseline	D*emoFBVP*	CD*emoFBVP*	CD*emoFBVP* +ROI
75.67	76.54	81.41	**83.18**

Table 4.13: Emotion Recognition Using Cohn Kanade Database

SVM Baseline	D*emoF*	CD*emoF*	CD*emoF* +ROI
95.4	95.9	96.8	**97.3**

Table 4.14: Emotion Recognition Using Mind Reading Database

SVM Baseline	D*emoV*	CD*emoV*
90.62	92.1	**93.4**

Table 4.15: Emotion Recognition Using DEAP Database

SVM Baseline	D*emoP*	CD*emoP*
78.6	78.8	**79.5**

Table 4.16: Emotion recognition using MAHNOB-HCI database

SVM Baseline	D*emoFBVP*	CD*emoFBVP*	CD*emoFBVP* +ROI
52.4	53.1	57.9	**58.5**

We applied the CD*emoFBVP* model (a model very similar to D*emoFBVP* but formed by stacking convolutional RBMs) to learn the multimodal deep features. We also extracted regions of interest (ROI) in the face (around the eyes, eyebrows and

mouth area) and body images (head, hands and legs) and fed them to the deep CD*emoFBVP*+ROI model. Tables (4.12), (4.13) , (4.14) , (4.15) and (4.16) show percentage emotion recognition accuracies on various emotion corpora. Tables (4.12), (4.13) and (4.16) compare performances of DBN, CDBN and CDBN+ROI models with the SVM baselines. Tables (4.13) and (4.14) compare performances of DBN and CDBN models with SVM baseline models on voice and physiological signal data (there is no ROI in voice and physiological data). Again, to depict generalizability, we show results on standard emotion datasets. We notice that our CDBN+ROI models outperform our CDBN models which in turn perform better than the DBN models and SVM baselines.

4.5.2 Conclusions

In the above section, we showed that convolutional deep belief network (CDBN) models along with region of interest extraction learn salient multimodal features for recognition of low intensity/subtle expressions of emotions.

4.6 Auto-associators for Emotion Recognition

In the following section, we are interested in modeling relationships between audio and video data. We have used audio-visual emotion recognition to validate our methods. We consider three learning settings uni-modal deep learning, intra-modality learning and multimodal fusion, as shown in Table (4.17). A simple linear classifier is used for supervised training and testing to examine the different feature learning models with multimodal data. In the intra-modality feature learning model, data from multiple modalities is available during the feature learning phase. In this model, supervised training and testing phases use data from a single modality. In the multimodal fusion setting, data from all modalities is available at all phases; this represents

the setting considered in most multimodal emotion recognition systems.

In the following sections, we first describe the building blocks of our model. We present different multimodal learning models for successful emotion recognition. We give a brief description about the different datasets used for feature learning and supervised training and testing. Finally, we report our experimental results and conclusions.

Table 4.17: Multi-Modal Feature Learning Settings

	FEATURE LEARNING	**SUPERVISED TRAINING**	**TESTING**
Uni-Modal Deep Learning	Audio Video	Audio Video	Audio Video
Intra-Modality Learning	Audio+ Video	Video Audio	Video Audio
Multi-Modal Fusion	Audio + Video	Audio + Video	Audio + Video

.

4.6.1 Feature Learning Methods

In this section, we describe multimodal feature learning models for the task of emotion recognition. The audio and video input to the model are spectrogram and video frames respectively.

We first describe our uni-modal deep learning model. We use this model as a baseline to compare the results of our multimodal models. This model also acts as a pre-training model for our deep networks. In this model, we train the RBM separately for audio and video data (see Figure 4.9 (a) and (b)). After learning the RBM, the posteriors of the hidden variables given the visible variables acts as a new representation of the data. To pre-train the deep multimodal model, we consider greedy training a RBM over the pre-trained layers for each modality. The posteriors of the first layer hidden variables are used as the training data for the new layer.

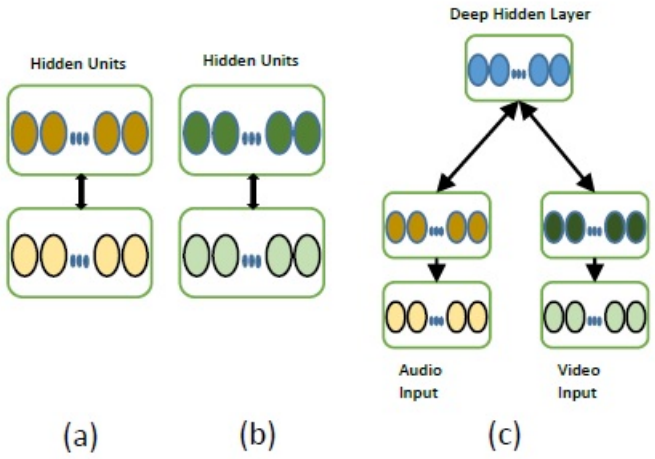


Figure 4.9: RBM Pre-Training Models for (A) Audio, (B) Video (C) Multimodal Models

This new representation of data helps the model learn higher-order correlations across modalities (see Figure 4.9 (c)).

In Intra-modality feature learning setting, both modalities are present during feature learning but only a single modality is used for supervised training and testing. We initialize deep auto-associators (see Figure 4.10 (a)) with multi-modal DBN weights. It is then trained to reconstruct both modalities when given only single modality data and thus able to discover correlations across modalities. In the multimodal fusion setting, multiple modalities are present for supervised training and testing. The deep auto-associator is then trained to reconstruct both modalities when given multimodal data (see Figure 4.10 (b)).

4.6.2 Experiments

We test the deep learning architectures described in the previous sections for multimodal emotion recognition using audio, video and audio-visual data. Unsupervised

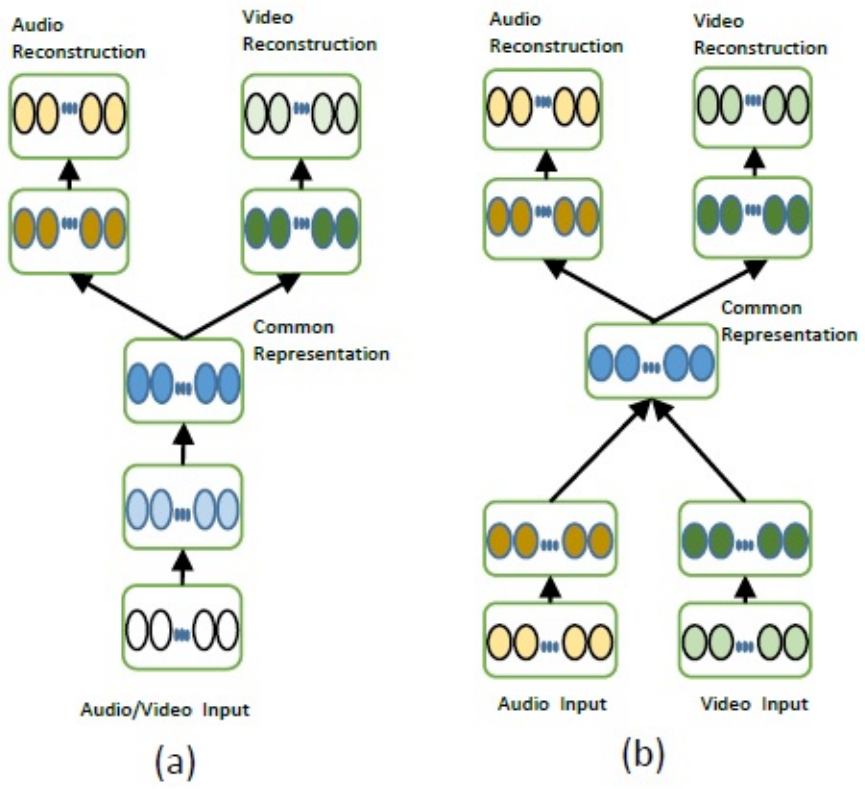


Figure 4.10: Deep Auto-Associator Models. (A)Intra-Modal Audio/ Video-Only Deep Auto- Associator (B) Multimodal Deep Auto-Associator

feature learning requires only unlabeled data; therefore, we combine data from multiple datasets to learn the features. All deep auto-associator models were trained with the available unlabeled audio and video data. The Cohn-Kanade database, MMI database, Haq and Jackson database were used for unsupervised feature learning while the *emoFVBP* database was used for supervised training and testing. Care was taken that no test data was used for unsupervised feature learning. The *emoF-BVP* is a multimodal database capturing facial expressions, body expressions, vocal expressions and physiological data under different emotion labels. We used the face and voice data of the six basic expressions (Happy, Sad, Anger, Disgust, Fear and

Surprise) from this database for supervised training and testing.

4.6.3 Results

Table 4.18 shows recognition accuracies for emotion recognition on the *emoFVBP* database. We see that when audio and video data are both used during the feature learning and classification stages (Multimodal fusion), we obtain close to an 8% increase in accuracy as compared to using only single modalities (Audio RBM, Video RBM). The intra-modal audio-only deep encoder reconstructs audio and video data using only audio signals. Similarly, the intra-modal video-only deep encoder reconstructs both the modalities given only video data. In this case, we are essentially training a modality-specific deep auto-associator network. It is very interesting to note that their recognition accuracies are 93.1% and 94.2% respectively. This shows

Table 4.18: Emotion Recognition Accuracy on *emoFBVP* Database

Deep Learning Model	% Recognition Accuracy
Audio RBM	87.7%
Video RBM	89.2%
Multimodal deep auto-associator	96.8%
Intra-modal audio-only deep auto-associator	93.1%
Intra-modal video-only deep auto associator	94.2%

that the model performs well by learning better single modality features using other additional unlabeled audio and video data. Effectively, the network learns a model that is robust to inputs when a modality is absent. Therefore, we can successfully use these models in emotion recognition systems when only a single modality is present.

In the above section, we proposed two multi-modal deep auto-associator for learning audio and video emotion data. Our multi-modal fusion model gives a recognition accuracy of 96.8%. This shows that our model handles large amounts of unlabeled data effectively and fuses multiple data modalities to form unified representations

that capture features useful for emotion recognition. Our intra-modal audio-only and video-only models give recognition accuracies of 93.1% and 94.2% respectively. This shows that our model effectively captures correlations across shallow representations between modalities.

4.6.4 Conclusions

We proposed two multi-modal deep auto-associator for learning audio and video emotion data. Our model was able to handle large amounts of unlabeled data effectively and fuse multiple data modalities to form unified representations that captured features useful for emotion recognition. Our intra-modal audio-only and video-only models effectively captured correlations across shallow representations between multiple modalities.

4.7 Transfer of Emotion-Rich Features between Deep Belief Networks

The introduction of deep architectures has brought significant improvements in many visual recognition tasks. These algorithms come with huge computational costs and finding the best training algorithm that offers the shortest training time is an interesting area of research. In this section, we follow a transfer learning approach and present a study to investigate the effect of transfer of emotion-rich features between source and target networks on classification accuracy and training time. First, we propose emosource -a 6-layer Deep Belief Network (DBN), trained on popular emotion corpora for multimodal emotion classification. Second, we propose two 6-layer DBNs - emotarget and emotarget$_{\text{ft}}$ and study the transfer of emotion features between source and target networks in a layer-by-layer fashion. Our experimental results reveal that our emotarget model achieves comparable classification accuracy as that of emosource, with reduced training times when the transferred emotion features are not changed

during training on the target dataset. We also show that our emotarget$_{\text{ft}}$ model achieves a performance boost over the emosource model with approximately the same training time when the entire target network is re-trained on the target dataset. To the best of our knowledge, this is the first research effort to study the transfer of emotion features layer-by-layer in a multimodal setting.

One of the main concerns with using deep architectures for vision tasks is the amount of time required to train the network. Therefore, finding the appropriate training algorithm that gives good performance accuracy with reduced training time becomes very important.

Consider a real world example where we have a deep model trained on a multimodal emotion dataset. Let us call this model as the source model. The model recognizes emotions with reasonable accuracy and the training time is approximately 10 days. Now, we come across a new emotion dataset. We wish to train a new model (let us call this model as the target model) on the new dataset, but do not have much time available for training. Can we use the emotion-rich features already learned by our source model for training the target model? What is the effect on the classification accuracy and training time when we do so?

We present answers to these questions in three contributions. First, we apply deep belief networks to solve the problem of multimodal emotion recognition. We train a 6-layer DBN on standard emotion corpora and document the time taken to train the network and the classification accuracy achieved. This model acts as our source network and we call it *emosource*. Second, we propose the *emotarget* DBN model, which is also a 6-layer DBN, and study the transfer of multimodal emotion features between the networks in a layer-by-layer fashion. We document the effect of the transfer on the models emotion classification accuracy and training time when trained on new emotion datasets. Finally, as our third contribution, we show that our

emotarget$_{ft}$ model achieves a performance boost over *emosource* model with similar training time when the entire target network is re-trained on the target dataset.

4.7.1 emoDBN Models

In this section, we introduce our *emosource*, *emotarget* and *emotarget*$_{ft}$ models. In section 4.7.2 and 4.7.3, we explain how our models learn multimodal emotion features through the deep layers. We train our deep belief network models on four popular emotion datasets - emoFBVP database, Mind Reading emotions library, MMI database of facial expressions and the Cohn-Kanade (CK) database for emotion classification tasks.

4.7.2 emosource DBN model

Figure 4.11 (a) gives an illustration of the emosource DBN network. We extend the DBN models proposed in Kim *et al.* (2013) to include multiple modalities of expressions of emotions. The emosource model is a 6-layer DBN that learns multimodal emotion features from facial expressions, body gestures, vocal expressions and physiological signals individually in the first hidden layer. All of these features are concatenated and fed as input to the second hidden layer. Hidden layers 3 to 6 capture higher order non-linear dependencies of the multimodal emotion features. We employ 2000 hidden units in each of the hidden layers in this model. The output of the last layer is fed to an SVM for emotion classification as in Kim *et al.* (2013). Our emosource model is trained on each of the standard emotion datasets. This model acts as the source network for the emotarget and emotarget$_{\text{ft}}$ models.

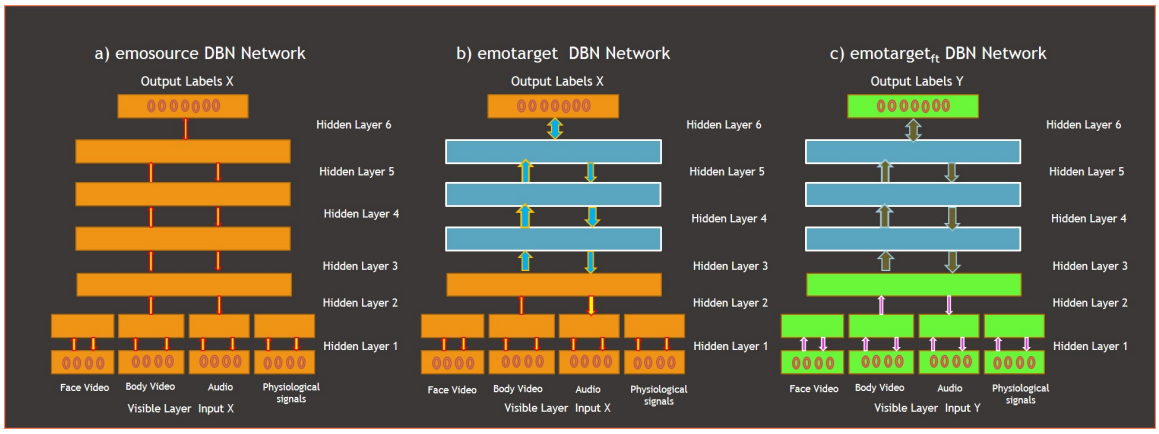


Figure 4.11: Proposed emoDBN Models (Best Viewed in Color), Source: Ranganathan *et al.* (2016b)

4.7.3 emotarget and emotarget$_{ft}$ DBN models

The emotarget model is also a 6-layer DBN that learns multimodal emotion features just like our emosource DBN model. We follow a transfer learning approach (Yosinski *et al.* (2014)) and transfer emotion-rich features from the emosource DBN layer-by-layer to the emotarget DBN model and train the remaining layers of the emotarget DBN on a new emotion dataset leaving the transferred features frozen. Figure 4.11 (b) shows an example to describe the layer-by-layer transfer of features. The features from the first two layers of the emosource model trained on dataset X (in Figure 4.11 (a)) are transferred to the first two layers of the emotarget DBN model. The layers 3, 4 and 5 of the emotarget DBN are re-initialized randomly and trained on a new dataset Y.

In the emotarget$_{\text{ft}}$ model (see Figure 4.11 (c)), the features from the first two layers of the emosource model trained on dataset X (in Figure 4.11 (a)) are transferred to the first two layers of the emotarget$_{\text{ft}}$ DBN model. The entire network is re-trained on the new dataset Y.

4.7.4 Parameter Selection

emoDBN models are trained in a greedy layer-wise manner. In this section, we discuss how we select the parameters of our DBN models. We use the mini-batch learning approach in our models, i.e. learning is done in mini-batches and the parameters are made reasonable before learning more data.

Learning Rate: Learning rate is a crucial parameter that influences the convergence of training. We try out different values in the set $10^{-1}, 10^{-2}, \cdots, 10^{-5}$ and perform cross validation. The learning rate that yields the best results is chosen and kept constant while training each of the stacked RBMs.

L_2 norm constraint: We employ a L_2 norm constraint on the input weights of the hidden layer units during training of our emoDBN models.

Momentum: We increase the momentum parameter linearly from a rate of 0.5 to a maximum value, which we determine using cross validation. We find that the value of momentum and the rate of increase of momentum impacted the classification performance significantly.

Dropout: Two dropout masks, one for the visible units and one for the hidden units are used. The weight matrices of the visible and hidden units are multiplied by the hidden dropout factor (chosen as 0.5) and visible dropout factor (chosen as 0.8) respectively.

Hyperparameters: We select the hyperparameters using cross-validation over the training data.

4.8 Experiments and Results

In this section, we describe our experiments to investigate the effect of transfer of emotion-rich features between source and target DBN networks. In our first ex-

periment, we train our 6-layer emosource model on each of the emotion databases for emotion classification. We document the classification accuracy achieved and the time taken for training. The network acts as a baseline to which we compare the results from our second set of experiments. It is to be noted that our goal here is not to achieve state-of-the-art performance for emotion classification but to present a study to investigate the effect of transfer of emotion features between networks on classification accuracies. Table 4.19 shows the percentage classification accuracies and the training times to train our emosource network on the *emoFBVP*, Mind Reading, MMI and Cohn Kanade databases. All our testing is done on the target datasets.

The emosource model trains on the multimodal data of facial expressions, body gestures, vocal expressions and physiological signal data as explained in Section 5.1. The first row of Table 4.19 gives us the emotion classification accuracy and the time taken to train our emosource network on the *emoFBVP* database. Our 6-layer emosource model achieves a percentage emotion classification accuracy of 81.36%. The model trains on the dataset for 18 hours and 24 minutes. We achieve a percentage classification accuracy of 87.62% on the Mind Reading Emotions library database with a training time of 22 hours and 5 minutes. As the dataset consists of only facial and vocal modalities of expressions of emotions, a bimodal variant of our emosource model with only two modalities in the input layer is used. The MMI database of facial expressions and the Cohn-Kanade database consist of only the face modality of expressions of emotions. Again, we use a unimodal variant of our emosource model to train on these datasets. Our model achieves a percentage classification accuracy of 87.39% and training time of 16 hours and 38 minutes when trained on the MMI database and a percentage classification accuracy of 89.51% and training time of 15 hours and 49 minutes when trained on the Cohn-Kanade database.

In the second set of experiments, we study the transfer of emotion features layer-

Table 4.19: emosource DBN Trained on Dataset X

Dataset	**Accuracy**	**Training Time**
emoFBVP	81.36	18 hrs 24 mins
Mind Reading	87.62	22 hrs 05 mins
MMI	87.39	16 hrs 38 mins
Cohn Kanade	89.51	15 hrs 49 mins

by-layer between our emosource and emotarget models. As explained earlier, the first k layers from the emosource model, which is trained on dataset X, are copied and transferred to the first k layers of the emotarget model and left frozen. The higher layers (layers (k+1) to 6) are initialized randomly and trained on dataset Y. We document the percentage classification accuracy and training time after transferring each layer of emotion features from emosource to emotarget. For this experiment, we choose different pairs of emotion databases to train our emosource and emotarget DBN models. For example, when emosource is trained on *emoFBVP*, we document results when emotarget is trained on Mind Reading, MMI and Cohn Kanade databases. Similarly, when emosource is trained on Mind Reading, we document results when emotarget is trained on *emoFBVP*, MMI and Cohn-Kanade databases. We do this for all combinations of source and target datasets.

In our final set of experiments, we study the transfer of emotion features layer-by-layer between our emosource and emotarget$_{\text{ft}}$ models. Here, the first k layers from the emosource model, which is trained on dataset X, are copied and transferred to the first k layers of the emotarget$_{\text{ft}}$ model. The entire network is re-trained on a new target dataset Y. We document the percentage classification accuracy and training time after transferring each layer of emotion features from emosource to emotarget$_{\text{ft}}$. We follow the same set of rigorous experiments done between emosource and emotarget

models and document results in a similar way.

Table 4.20: Source: *emoFBVP*, Target: Mind Reading.

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	87.59	20:32	87.93	22:28
2	85.61	17:02	89.57	21:43
3	82.18	13:15	90.33	21:36
4	79.34	09:17	93.64	21:47
5	78.72	04:50	94.99	22:07

Table 4.21: Source: *emoFBVP*, Target: MMI

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	87.32	13:36	87.74	16:44
2	84.57	10:53	90.26	16:15
3	81.93	07:49	94.11	16:26
4	77.71	04:06	95.35	16:39
5	76.80	01:52	96.18	16:40

Table 4.22: Source: *emoFBVP*, Target: Cohn Kanade

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	89.42	12:22	89.49	15:30
2	86.99	09:54	92.23	15:46
3	83.24	04:58	93.67	15:28
4	82.86	03:17	95.74	15:56
5	80.66	01:33	97.04	15:32

4.8.1 Results when emosource is trained on emoFBVP dataset

Tables 4.20, 4.21 and 4.22 document the results of percentage classification accuracies and training times achieved when emosource is trained on emoFBVP database and emotarget and emotarget$_{\text{ft}}$ are trained on Mind Reading, MMI and Cohn Kanade databases respectively. The first column of the tables specifies the Layer #, i.e. the layer at which the network is chopped and re-trained. From Table 4.19, we know that a 6-layer emosource DBN trained on the Mind Reading database achieves a percentage classification accuracy of 87.62% and a training time of 22 hours and 5 minutes when trained from scratch. From Table 4.20, we observe that, as we transfer features from successive layers of emosource, the percentage classification accuracy falls from 87.59% (when we transfer features from layer 1) to 78.72% (when we transfer features from first 5 layers) while the time taken to train the emotarget network becomes shorter and shorter (only 4 hours and 50 minutes when features from the first 5 layers are transferred). Thus, our emotarget DBN network is able to achieve comparable classification accuracy to that achieved by emosource model with shorter training time when more and more layers are transferred from source to target networks. This is a very interesting and practically useful result for researchers in the emotion recognition domain. The last two columns of Table 4.20 give us the classification accuracy and training time when our emotarget$_{\text{ft}}$ model is trained on the Mind Reading dataset. Here, we observe that, as we transfer features from successive layers of emosource, the percentage classification accuracy increases from 87.93% (when we transfer features from layer 1) to 94.99% (when we transfer features from first 5 layers) while the time taken to train the emotarget$_{\text{ft}}$ network is similar to the time taken to train emosource on Mind Reading from scratch (approximately 22 hours and 15 minutes). Thus, our emotarget$_{\text{ft}}$ DBN model is able to achieve a boost in

classification accuracy compared to the emosource model with similar training time profiles when more and more layers are transferred from source to target networks. Again, this is a very interesting observation. Tables 4.21 and 4.22 follow the same trend in our observations when we train emotarget and emotarget$_{\text{ft}}$ models on MMI and Cohn Kanade databases.

4.8.2 Results when emosource is trained on Mind Reading dataset

Tables 4.23, 4.24 and 4.25 document the results of percentage classification accuracies and training times achieved when emosource is trained on the Mind Reading database and emotarget and emotarget$_{\text{ft}}$ are trained on emoFBVP, MMI and Cohn Kanade databases respectively. From Table 4.19, we know that a 6-layer emosource DBN trained on the emoFBVP database achieves a percentage classification accuracy of 81.36% and a training time of 18 hours and 24 minutes when trained from scratch. Again, we notice from Table 4.23 that, the percentage classification accuracy falls from 81.24% (when we transfer features from layer 1) to 73.67% (when we transfer features from first 5 layers) while the time taken to train the emotarget network becomes shorter and shorter (only 2 hours and 48 minutes when features from the first 5 layers are transferred). Also, the percentage classification accuracy increases from 82.46% (when we transfer features from layer 1) to 90.15% (when we transfer features from first 5 layers) while the time taken to train the emotarget$_{\text{ft}}$ network is similar to the time taken to train emosource on Mind Reading from scratch (approximately 18 hours and 30 minutes). Tables 4.24 and 4.25 follow the same trend in our observations when we train emotarget and emotarget$_{\text{ft}}$ models on MMI and Cohn Kanade databases.

Table 4.23: Source: Mind Reading, Target:*emoFBVP*

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	81.24	16:48	82.46	18:39
2	78.62	14:23	85.33	18:42
3	77.99	10:59	86.94	18:34
4	74.38	07:46	87.61	18:38
5	73.67	02:48	90.15	18:26

Table 4.24: Source: Mind Reading, Target:MMI.

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	86.99	13:42	87.03	16:39
2	85.02	11:02	90.47	16:48
3	82.40	07:24	92:35	16:30
4	79.64	03:54	95.62	16:44
5	77.77	01:31	96.34	16:40

Table 4.25: Source: Mind Reading, Target: Cohn Kanade

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	80.04	12:34	89.94	15:36
2	86.89	09:23	92.66	15:24
3	83.14	05:14	94.78	15:16
4	81.68	03:48	95.33	15:52
5	80.75	01:57	96.89	15:41

Table 4.26: Source: MMI, Target: emoFBVP

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	80.42	16:31	82.14	18:21
2	77.94	13:59	84.83	18:34
3	74.83	11:14	85.37	18:36
4	72.96	07:15	88.65	18:27
5	72.41	03:20	89.78	18:41

Table 4.27: Source: MMI, Target: Mind Reading

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	86.88	20:06	87.79	22:15
2	85.41	17:01	89.42	22:31
3	82.68	12:48	93.14	22:24
4	79.14	09:34	93.89	22:38
5	78.93	05:10	94.35	22:18

Table 4.28: Source: MMI, Target: Cohn Kanade

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	88.33	11:58	90.14	15:34
2	86.45	09:44	91.33	15:42
3	83.96	04:53	93.47	15:28
4	81.21	03:31	96.63	15:42
5	80.84	01:23	97.08	15.55

4.8.3 Results when emosource is trained on MMI dataset

Tables 4.26, 4.27 and 4.28 document the results of percentage classification accuracies and training times achieved when emosource is trained on the MMI database and emotarget and $\text{emotarget}_{\text{ft}}$ are trained on emoFBVP, Mind Reading and Cohn Kanade databases respectively. We notice similar trends in the classification accuracies and time taken for training as in previous sections.

4.8.4 Results when emosource is trained on Cohn Kanade dataset

Tables 4.29, 4.30 and 4.31 document the results of percentage classification accuracies and training times achieved when emosource is trained on the Cohn Kanade database and emotarget and $\text{emotarget}_{\text{ft}}$ are trained on emoFBVP, Mind Reading and MMI databases respectively. We again notice similar trends in the classification accuracies and time taken for training as in previous sections.

4.8.5 Layer-wise Summary of the Results

In this section, we present a layer-wise summary of the observations from our study.

Layer-wise transfer of features to emotarget DBN

Layer # 1: the emotion classification accuracy is similar to the emosource model. From this, we observe that, for the dataset pairs X and Y used to train emosource and emotarget networks, the emotion features in the first layer of the DBN networks are general. We achieve faster training time when compared to the source network because we are learning only the top 5 layers of the network.

Layer # 2: The emotion classification accuracy shows a slight drop in performance when compared to the emosource model. This is because the transferred features are

Table 4.29: Source: Cohn Kanade, Target: emoFBVP

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	80.86	16:24	82.08	18:15
2	77.38	14:56	84.16	18.23
3	76.04	11:15	87.38	18.34
4	73.83	07:24	88.09	18:21
5	71.78	03:00	90.09	18:20

Table 4.30: Source: Cohn Kanade, Target: Mind Reading.

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	86.34	20:20	87.88	22:03
2	83.42	16:48	88.44	22.11
3	82.14	12:53	92.16	22:10
4	80.11	08:56	93.49	22.16
5	78.35	04:44	94.00	22:21

Table 4.31: Source: Cohn Kanade, Target:MMI

Layer #	emotarget		$\text{emotarget}_{\text{ft}}$	
	Accuracy	Tr.Time	Accuracy	Tr.Time
1	87.12	13:31	88.14	16:48
2	86.42	11:22	89.38	16:42
3	84.64	07:04	92.46	16:37
4	83.18	04:10	94.11	16:30
5	82.91	01:12	96.55	16:29

more specific to dataset X. We achieve faster training time when compared to the source network and when only one layer is transferred because we are learning only the top 4 layers of the network.

Layer # 3, 4, 5: The emotion classification accuracy decreases further as the transferred features become more and more specific to the source dataset X. The training time further decreases as we learn only fewer layers of features at a time.

Layer-wise transfer of features to emotarget$_{ft}$ DBN

Layer # 1: the models show a comparable performance to emosource with respect to classification accuracy and training time. Again, this is attributed to the first layer features being general.

Layer # 2, 3, 4, 5: We observe a significant boost in performance accuracy as we transfer more and more layers from emosource to emotarget$_{\text{ft}}$. This result shows that the transfer of emotion-rich features boosts the generalization performance of the target network. The important point to note here is that we achieve this boost in performance in approximately the same time taken to train the source.

4.8.6 Conclusions

Here, we presented a study to investigate the effect of transfer of emotion-rich features between source and target networks on percentage emotion classification accuracy and training time. We made three interesting contributions in this work. First, we proposed emosource - a 6-layer DBN trained on multimodal and unimodal emotion corpora for emotion classification. Second, we proposed emotarget and emotarget$_{\text{ft}}$ DBN models and studied the transfer of features between emosource and these networks in a layer-by-layer manner. Finally, we experimentally showed that our emotarget model achieved reasonably comparable classification accuracies to

that of emosource with significantly shorter training times when the transferred features are left frozen while our emotarget$_{\text{ft}}$ model achieved a performance boost over emosource model with similar training times as the source network when the entire network is trained on the target dataset. In short, our emotarget and emotarget$_{\text{ft}}$ models were able to successfully re-purpose the emotion-rich features learned by the source model to train the target models and achieve shorter training times and performance boosts respectively. Going back to our real world example, depending on the need for reduced training time or performance boost in accuracy, one could use either of our emotarget or emotarget$_{\text{ft}}$ models. This makes our study extremely useful in a practical setting. To the best of our knowledge, this is the first research approach to studying the effect of transfer of emotion features in a layer-by-layer manner in a multimodal setting.

4.9 Summary

In this chapter, we first presented the emoFBVP database of multimodal (face, body gesture, voice and physiological signals) recordings of actors enacting various expressions of emotions. The database consisted of audio and video sequences of actors displaying three different intensities of expressions of 23 different emotions along with facial feature tracking, skeletal tracking and the corresponding physiological data.

Next, we described four deep belief network (DBN) models and showed that these models generate robust multimodal features for emotion classification in an unsupervised manner. Our experimental results showed that the DBN models perform better than the state of the art methods for emotion recognition. Finally, we proposed convolutional deep belief network (CDBN) models that learn salient multimodal features of expressions of emotions. Our CDBN models furnished better recognition accuracies when recognizing low intensity or subtle expressions of emotions when compared

to state of the art methods.

We also proposed two multi-modal deep auto-associator for learning audio and video emotion data. Our model effectively handled large amounts of unlabeled data and fused multiple data modalities to form unified representations that captured features useful for emotion recognition. Our intra-modal audio-only and video-only models were able to effectively capture correlations across shallow representations between multiple modalities.

The last section in this chapter, presented a study to investigate the effect of transfer of emotion-rich features between source and target networks on percentage emotion classification accuracy and training time. We made three interesting contributions. First, we proposed emosource - a 6-layer DBN trained on multimodal and unimodal emotion corpora for emotion classification. Second, we proposed emotarget and emotarget$_{\text{ft}}$ DBN models and studied the transfer of features between emosource and these networks in a layer-by-layer manner. Finally, we experimentally showed that our emotarget model achieved reasonably comparable classification accuracies to that of emosource with significantly shorter training times when the transferred features are left frozen while our emotarget$_{\text{ft}}$ model achieved a performance boost over emosource model with similar training times as the source network when the entire network is trained on the target dataset. In short, our emotarget and emotarget$_{\text{ft}}$ models successfully re-purposed the emotion-rich features learned by the source model to train the target models and achieve shorter training times and performance boosts respectively. This makes our study extremely useful in a practical setting. To the best of our knowledge, this is the first research approach to studying the effect of transfer of emotion features in a layer-by-layer manner in a multimodal setting.

Chapter 5

DEEP ACTIVE LEARNING FOR SINGLE-LABEL IMAGE CLASSIFICATION

Deep learning algorithms learn a highly discriminating set of features for a given machine learning task and have depicted commendable performance in a variety of applications. One of its major successes has been in computer vision, where it has achieved state-of-the-art performance in object recognition, image segmentation and activity recognition among others. A fundamental challenge in training a deep neural network is the requirement of large amounts of labeled training data. The rapid escalation of technology and the widespread emergence of modern technological equipments has resulted in the generation of humongous amounts of digital data in the modern era. However, while gathering large quantities of unlabeled data is cheap and easy, annotating the data (with class labels) is an expensive process in terms of time, labor and human expertise. This poses a significant challenge in inducing supervised learning models. The situation is even more serious for deep neural network models as they require much more hand-labeled training data as compared to other classification models. Thus, developing algorithms to minimize human effort in training deep models is of paramount practical importance. Active learning algorithms have gained popularity in reducing the human annotation effort in training machine learning models. Such algorithms automatically identify the salient and exemplar samples from large amounts of unlabeled data that can augment maximal information to the classification models and need to be labeled manually.

In this chapter, we begin with a brief overview of active learning methodologies with relevant algorithms and examples. We then describe the field of deep active learning - where we combine ideas from deep learning and active learning to learn in-

telligent models for classification. We then propose a novel active sampling algorithm to identify the salient and exemplar unlabeled samples to be manually annotated to train DBNs. Our method is validated on single-label image classification on a variety of benchmark datasets for different applications.

5.1 Active Learning Models

The main hypothesis in active learning is that when a learning algorithm can choose the data it wants to learn from, it can perform better than traditional methods with substantially less data for training. In many settings, there can be limiting factors that hamper gathering large amounts of labelled data. Let's take the example of studying pancreatic cancer. We wish to predict whether a patient will get pancreatic cancer, however, we only have the opportunity to give a small number of patients further examinations to collect features. In this case, rather than selecting patients at random, we select patients based on certain criteria. An example criteria is, if the patient drinks alcohol and is over 40 years. This criteria does not have to be static but can change depending on results from previous patients. For example, if we find that the model is good at predicting pancreatic cancer for those over 50 years, but struggles to make accurate predictions for those between 40-50 years, we change our selection criteria accordingly. The process of selecting these patients based upon the data we have collected so far is called active learning.

5.1.1 Definition

Let us first examine why active learning works. Looking at the leftmost picture in Figure (5.1) (taken from Settles (2010)), we see two clusters, those colored green and those colored red. This is a classification task and we would like to create a decision boundary that would separate the green and red shapes. However, we can assume

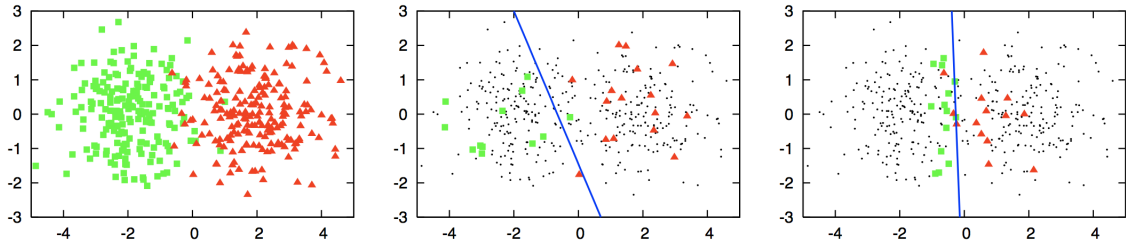


Figure 5.1: Active Learning Example Using Toy Dataset (Best Viewed in Color) (Settles (2010))

that we do not know the labels (red or green) of the data points, but trying to find the label for each of them would be very expensive. As a result, we sample a small subset of points and find those labels and use these labelled data points as training data for a classifier.

In the middle picture, logistic regression is used to classify the shapes by first randomly sampling a small subset of points and labeling them. However, we see that the decision boundary created using logistic regression (the blue line) is sub-optimal. This line is clearly skewed away from the red data points and into the green shapes area. This means that there will be many green data points that will be labelled incorrectly as red. This skew is due to the poor selection of data points for labelling.

In the right-most picture, logistic regression is used again, but this time, a small subset of points is selected using an active learning query method. This new decision boundary is significantly better as it better separates both colors. This improvement comes from selecting superior data points so that the classifier was able to create a very good decision boundary.

5.1.2 Active Learning Scenarios

In active learning, there are typically three scenarios or settings in which the learner will query the labels of instances. The three main scenarios that have been considered in literature are:

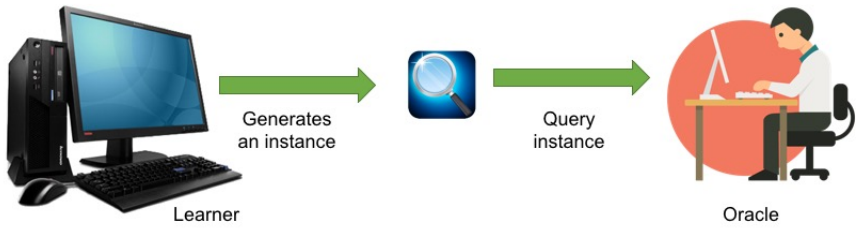


Figure 5.2: Membership Query Systhesis

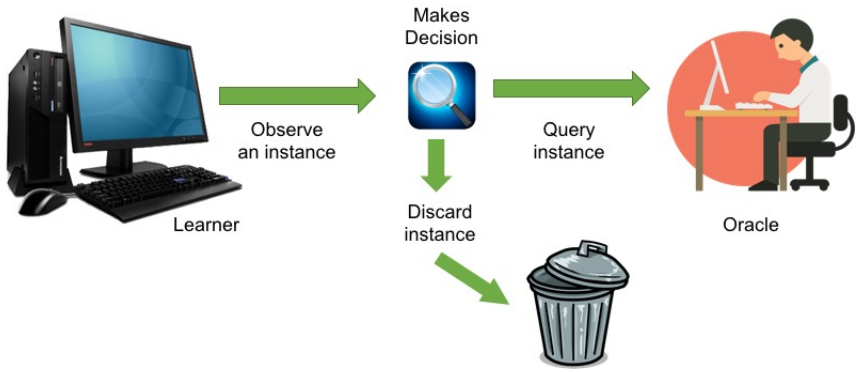


Figure 5.3: Stream-Based Selective Sampling

- Membership Query Synthesis: this means that the learner generates an instance from some underlying natural distribution (Refer Figure (5.2)). For example, if the data is pictures of digits, the learner would create an image that is similar to a digit and this created image is sent to the oracle to label.

- Stream-Based Selective Sampling: in this setting, we make the assumption that getting an unlabeled instance is free. Based on this assumption, we select each unlabeled instance one at a time and allow the learner to determine whether it wants to query the label of the instance or reject it based on its informativeness (Refer Figure (5.3)). To determine informativeness of the the instance, we use a query strategy. For example, we would select one image from the set of unlabeled images, determine whether it needs to be labelled or discarded, and then repeat with the next image.

- Pool-Based sampling: this setting assumes that there is a large pool of unlabeled

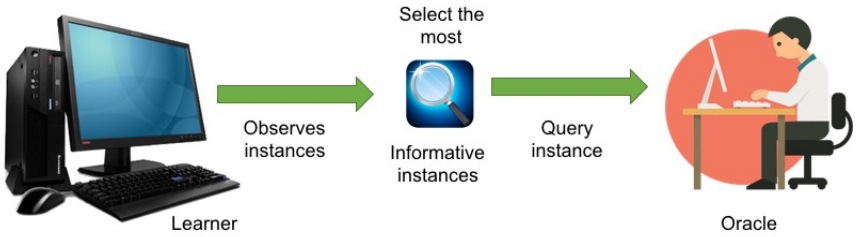


Figure 5.4: Pool-Based sampling

data, Instances are then drawn from the pool according to some informativeness measure. This measure is applied to all instances in the pool (or some subset if the pool is very large) and then the most informative instance(s) are selected (Refer Figure (5.4)). This is the most common scenario in the active learning community. For example, all the unlabeled images of digits are ranked and then the best (most informative) instance(s) are selected and their labels requested.

5.1.3 Query Strategies

The core difference between an active and a passive learner is the ability to query instances based upon past queries and the labels from those queries. All active learning scenarios require some sort of informativeness measure of the unlabeled instances. In this section, we explain three popular approaches for querying instances under the common topic called uncertainty sampling due to its use of probabilities (for more query strategies and in-depth information on active learning, refer to Settles (2010)).

We will use Table (5.1) to explain the query strategies. This table shows two data points and the probabilities that each point has each label. The probability d_1 has label A, B and C is 0.9, 0.09 and 0.01 respectively and probability that d_2 has label A, B and C is 0.2, 0.5 and 0.3.

1. **Least Confidence (LC)**: in this strategy, the learner selects the instance for

Table 5.1: Query Strategy- Explanation

Instances	Label A	Label B	Label C
d_1	0.9	0.09	0.01
d_2	0.2	0.5	0.3

which it has the least confidence in its most likely label. From Table (5.1), the learner is pretty confident about the label for d_1, since it thinks it should be labelled A with probability 0.9, however, it is less sure about the label of d_2 since its probabilities are more spread and it thinks that it should be labelled B with a probability of only 0.5. Thus, using least confidence, the learner would select d_2 to query it's actual label.

2. **Margin Sampling**: the shortcoming of the LC strategy, is that it only takes into consideration the most probable label and disregards the other label probabilities. The margin sampling strategy seeks to overcome this disadvantage by selecting the instance that has the smallest difference between the first and second most probable labels. Looking at d_1, the difference between its first and second most probable labels is 0.81 and for d_2 it is 0.2. Hence, the learner will select d_2 again.

3. **Entropy Sampling**: in order to utilize all the possible label probabilities, you use a popular measure called entropy. The entropy formula is applied to each instance and the instance with the largest value is queried. Using our example, d_1 has a value of 0.155 while d_2's value is 0.44 and so the learner will select d_2 once again.

5.1.4 An Example of Active Learning

In the previous section, we have seen the different components that make up active learning. This section puts all the components together with a simple example.

1. **Gathering Data**:

   The dataset is representative of the true distribution of the data. In reality, it becomes impossible to have a totally representative sample due to limitations such as time and availability.

   Consider an example detailed in Table (5.2) with 5 data points. Feature A and Feature B represent some features that a data point might have. It is important to note that the data we gather is unlabeled.

Table 5.2: Active Learning - Example

Instances	Feature A	Feature B
d_1	10	0
d_2	4	9
d_3	8	5
d_4	3	3
d_5	5	5

2. **Split into Labeled and Unlabeled Datasets**:

   We then split the data into a very small dataset which we will label and a large unlabeled dataset. There is no set number or percentage of the unlabeled data that is typically used.

   Usually, researchers do not use an oracle or an expert to label these instances. Typically, the dataset is fully labeled and we use a small amount as labeled data

and use the rest as unlabeled data. Whenever the learner selects an instance to query the oracle with, we look up the label for the instance.

Continuing with the example, we select two instances for labeled data, d_1 and d_3. The possible labels in this case are $'Y'$ and $'N'$ (Refer Table (5.3) and Table (5.4)).

Table 5.3: Labeled Dataset

Instances	Feature A	Feature B	Label
d_1	10	0	Y
d_3	8	5	N

Table 5.4: Unlabeled Dataset

Instances	Feature A	Feature B
d_2	4	9
d_4	3	3
d_5	5	5

3. **Training the Model**:

   After splitting the data, we use the labeled data to train the learner. Leaners that give a probabilistic response to whether an instance has a particular label are typically used, to enable the use of these probabilities for querying.

   In the example, we can use any classifier to train on the two labelled instances.

4. **Choosing Unlabeled Instances**:

   After training is complete, we select an instance or instances to query. We would use one of the active learning scenarios (Membership Query Synthesis, Stream-Based Selective Sampling or Pool-Based sampling) and the query strategy.

Let us use pool-based sampling with a batch size of 2 for our example. This means that, at each iteration, we will select two instances from the unlabeled dataset and then add these instances to the labelled dataset. Let us use least confidence to select instances. The learner selects d_2 and d_4 whose queried labels are 'Y' and 'N', respectively (refer Tables 5.5 and 5.6).

Table 5.5: Updated Labeled Dataset

Instances	Feature A	Feature B	Label
d_1	10	0	Y
d_3	8	5	N
d_2	4	9	Y
d_4	3	3	N

Table 5.6: Updated Unlabeled Dataset

Instances	Feature A	Feature B
d_5	5	5

5. **Stopping Criterion**:

   Now, we repeat steps 2 and 3 until some stopping criteria. That means that, we re-train our learner using the updated labeled dataset and then select further unlabeled data to query.

   In our example, we will stop at one iteration and are finished with the active learning algorithm. We use a separate test dataset to evaluate our learner and record its performance. This way, we see how the performance on the test set improved or stagnated with added labelled data.

Now that we have seen an overview of how active learning works, we will move on to describe an overview of deep active learning.

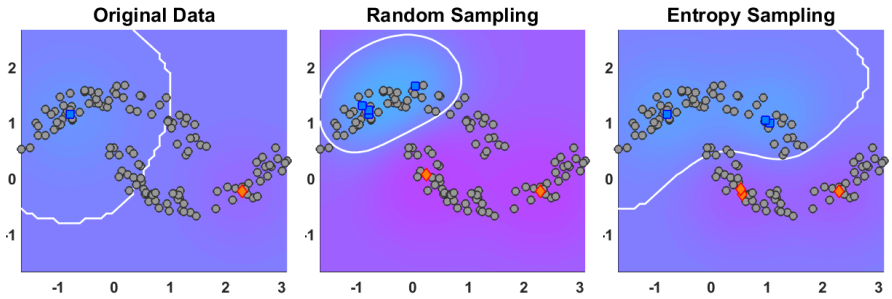


Figure 5.5: Illustration of the Principle of Active Learning With a Toy Two-Moon Dataset. Two Class Problem With Labeled Data in red and blue and Unlabeled Data in gray. SVM-Based Classification Results (Left) With Given Labeled Data, (Center) After Randomly Selecting Points and Getting Their Labels and (Right) After Labeling the Most Uncertain (Informative) Points Based on Entropy. Best Viewed in Color.

5.2 Deep Active Learning Models

Deep learning algorithms learn a highly discriminating set of features for a given machine learning task and have depicted commendable performance in a variety of applications. One of its major successes has been in computer vision, where it has achieved state-of-the-art performance in object recognition, image segmentation and activity recognition among others.

Active learning algorithms have gained popularity in reducing the human annotation effort in training machine learning models. Such algorithms automatically identify the salient and exemplar samples from large amounts of unlabeled data that can augment maximal information to the classification models and need to be labeled manually.

A visual illustration of active sampling is depicted in Figure 5.5. The figure on the left shows the original two-moon dataset with one labeled sample (blue and red) from each class. The unlabeled samples are marked in gray. The middle figure depicts the decision boundary (the white curve) corresponding to random sampling of two unlabeled data points and the figure on the right shows the decision boundary corresponding to uncertainty based active learning. Evidently, active learning produces a

much more discriminating decision boundary with very few labeled examples.

Existing algorithms treat active learning and deep model training as two independent problems. A deep model is first learned using a conventional loss function (softmax loss); the active sampling condition is then defined based on the posterior probabilities obtained from the last layer or the distance of a sample from the decision boundary. However, the merit of a deep model lies in its ability to learn a discriminating set of features for a given task; this property has not been leveraged in the existing algorithms combining deep learning and active learning. The core idea of this research presented in this dissertation is to leverage the feature learning capability of deep models to identify the most informative unlabeled samples for active learning.

In this work, we propose a novel active sampling algorithm to identify the salient and exemplar unlabeled samples to be manually annotated to train a deep belief network model. We introduce a novel loss function which consists of the conventional softmax loss and an entropy loss (to guide the active learning process) and train the deep model to optimize this joint objective. To the best of our knowledge, this is the first attempt to incorporate an uncertainty based criterion to train a deep belief network so that it can appropriately identify the most informative unlabeled samples for manual annotation. Although validated on the image classification task in this paper, the proposed algorithm is generic and can be used in any application where the salient and exemplar unlabeled samples need to be identified to train a deep learning model.

5.3 Proposed Framework

The core idea of this research is to leverage the feature learning capability of deep models to identify the most informative unlabeled samples for active learning. While existing algorithms first train a deep model using the softmax loss and then use the

posterior probabilities from the last network layer to derive sample selection criteria, we attempt to integrate the active sample selection criterion in the loss function and train the network to optimize the function. The feature representations learnt by the network are then specially tailored to the active learning task and enables it to better identify the samples that can augment maximal information to the model. From the above survey, it is clear that uncertainty sampling is the most widely used strategy for active sample selection. Entropy is a well-accepted measure to quantify the uncertainty of a classification model. We therefore append an entropy based term to the conventional softmax loss term and train the network to optimize the joint loss function.

Formally, let $X = \{x_1, x_2, \ldots, x_n\}$ be the training set containing n samples. The subset of labeled samples is represented as $X_l = \{x_1, x_2, \ldots, x_{n_l}\}$. The corresponding labels for X_l are denoted by $Y_l = \{y_1, y_2, \ldots, y_{n_l}\}$. Let the subset of unlabeled data points be, $X_u = \{x_{n_l+1}, x_{n_l+2}, \ldots, x_{n_l+n_u}\}$. $X = X_l \cup X_u$, is the union of the disjoint subsets X_l and X_u. Therefore, $n = n_l + n_u$. The goal is to estimate a classifier function $f(x)$, using the labeled data $\mathscr{D} = \{X_l, Y_l\}$. Here, $f(x_i),\ \forall i \in [1, n_l]$ is the conditional probability that the classifier assigns x_i to label y_i. The classifier function $f(.)$ is then applied on the unlabeled data $f(x_i),\ \forall i \in [n_l + 1, n]$, to predict the label $\hat{y}_i$. The accuracy of the classifier is tested by comparing the predicted labels of the unlabeled data with the ground truth labels for the unlabeled data.

5.3.1 Cross-entropy Loss for Labeled Data

Since the classifier $f(.)$ is implemented using a deep belief network, we use the standard cross-entropy loss to estimate the empirical classification error $E(\mathscr{D}; f)$,

which is given by,

$$\underset{f \in \mathscr{F}}{\operatorname{argmin}} E(\mathscr{D}; f) = \frac{1}{n_l} \sum_{i=1}^{n_l} L(f(x_i), y_i), \tag{5.1}$$

where the cross-entropy loss is given by,

$$L(f(x_i), y_i) = -\sum_{j=1}^{C} 1\{y_i = j\} \log f_j(x_i), \ \forall i \in [1, n_l]. \tag{5.2}$$

Here, C is the total number of label categories and $1\{.\}$ is the indicator function. $f_j(x_i) = e^{h_{ij}^N} / \sum_{j'} e^{h_{ij'}^N}$ is the softmax function defined on the activation h_{ij}^N, where h_{ij}^N is j-th component of the i-th data point in the N-th (final) layer of the network. The softmax function ensures $f(x_i) = [f_1(x_i), f_2(x_i), \ldots, f_C(x_i)]^\top$ is a probability vector with $f_j(x_i)$ being the probability that data point x_i is assigned to category C. During the process of active learning the labeled dataset is repeatedly augmented with newly obtained labeled data taken from the unlabeled dataset. The classifier $f(.)$ is in turn updated by retraining with the updated labeled dataset.

5.3.2 Entropy - Measure of Uncertainty

In the active learning setting, we have an oracle who provides labels for the unlabeled data at a fixed cost for every unlabeled data point. From the unlabeled set X_u, the oracle is given a batch of points B to be labeled. The labeled batch B is then combined with the labeled set X_l along with the corresponding labels (added to Y_l) that are provided by the oracle, i.e. $X_l \to X_l \cup B$. These data points are removed from the unlabeled set X_u in order to ensure X_l and X_u are disjoint, i.e. $X_u \to X_u \backslash B$. A new and improved classifier is estimated using the augmented labeled sets $\{X_l, Y_l\}$. This procedure is repeated until we run out of budget to get labeled data from the oracle. The challenge in active learning is to identify the most informative set of unlabeled samples to be labeled by the oracle. Intuitive reasoning leads us to select

data points that have been classified with highest uncertainty by the existing classifier. The classification of a data point is said to be uncertain if the data point can be assigned to more than one category with nearly equal probability. Given the probability of label assignment for a data point, entropy (from Information Theory) can be used to obtain a measure of uncertainty regarding its label assignment. The set B can therefore be chosen by selecting the data points with the largest uncertainty. Entropy based measures have been applied previously to select a set B with the most informative (highest uncertainty) data points in order to estimate an active learning based classifier (Chakraborty *et al.* (2015b)). The entropy can be expressed in terms of assigned label probabilities for the unlabeled data as,

$$H(f(x_i)) = -\sum_{j=1}^{C} f_j(x_i)\log f_j(x_i),\ \forall i \in [n_l+1, n] \tag{5.3}$$

where $f_j(x_i)$ is the probability of assigning x_i to category j. We define a probability vector for x_i as $p_i \coloneqq [f_1(x_i), f_2(x_i), \ldots, f_C(x_i)]^\top$. In standard active learning settings, a classifier is first trained on the labeled data and used to obtain the predictions for the unlabeled data. Entropy is then applied to obtain the uncertainty of such a classifier prediction. In this two-step approach, the unlabeled data does not play a role in training the classifier. We propose an active learning model where we combine the entropy measure along with the cross-entropy loss during training. We discuss the benefits of this joint loss in the following section.

5.3.3 Joint Loss for Active Learning

In our active learning DBN model, we treat the entropy measure as a loss and attempt to reduce the uncertainty of classification. The DBN is trained by combining both the labeled and unlabeled data with the aim to obtain least entropy on the unlabeled data and also least cross-entropy on the labeled data. We see the following

advantages to incorporating entropy minimization along with standard cross-entropy loss when training an active learning model: (*i*) all the available data is used to train the active learning model which results in a much more robust and adaptive model when compared to training with only the labeled data; (*ii*) training with unlabeled data eventually results in learning an effective classifier for the unlabeled data and (*iii*) the network trains itself to reduce entropy along with estimating a classifier (by reducing cross-entropy). In a single step, the network restructures its weights while minimizing cross-entropy (for labeled data) and minimizing entropy (for unlabeled data). The joint loss ensures that the data points with the largest entropy that are selected to form B, are the most uncertain and informative unlabeled data points with respect to the classifier $f(.)$. Over successive iterations, the positive effects of joint training with the labeled and unlabeled data get enhanced. Based on this intuitive reasoning, we combine the cross-entropy loss in Equation (5.2) and the entropy in Equation (5.3) to formulate a classifier with a joint loss that is given by,

$$\begin{aligned}\underset{f\in\mathscr{F}}{\operatorname{argmin}}\, E(\mathscr{D}; f) =& \frac{1}{n_l}\sum_{i=1}^{n_l} L(f(x_i), y_i) \\ &+ \frac{\lambda}{n_u}\sum_{i=n_l+1}^{n} H(f(x_i)). \end{aligned} \tag{5.4}$$

where λ controls the relative importance of the entropy loss.

5.3.4 Computing the Gradient

We use the standard backpropagation algorithm to learn the weights of the DBN. The output of the N-th layer of the network (before the loss) for a data point x_i, is given by the vector h_i^N. We define $p_{ij} \coloneqq f_j(x_i) = e^{h_{ij}^N} / \sum_{j'} e^{h_{ij'}^N}$, the probability that

data point x_i belongs to class j. The loss in terms of probabilities is given by,

$$E(X_l, X_u, Y_l) = -\frac{1}{n_l}\sum_{i=1}^{n_l}\sum_{j=1}^{C} 1\{y_i = j\}\log p_{ij} - \frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{C} p_{ij}\log p_{ij}. \tag{5.5}$$

We outline the derivative of the loss $E(.)$ with respect to h^N_{pq}, which is the q-th component of the p-th data point in the output of the N-th layer as,

$$\frac{\partial E}{\partial h^N_{pq}} = \begin{cases} \frac{1}{n_l}p_{pq} - 1\{y_p = q\}, & p \in [1, n_l] \\ \frac{\lambda}{n_u}p_{pq}\Big(\sum_j^C p_{pj}h^N_{pj} - h^N_{pq}\Big), & p \in [n_l + 1, n]. \end{cases} \tag{5.6}$$

During the training procedure, the derivative $\partial E/\partial h^N$ is back - propagated through the network in order to update the weights of the network [1].

5.3.5 Active Learning Network Architecture and Training

The network architecture and the method employed for training the network are discussed here along with the **algorithm pseudocode**. Figure 7.2 illustrates the network architecture of our Deep Active Learning model. Our model is a three layer deep belief network constructed by greedily training and stacking RBMs. Since the number of data points $n = n_l + n_u$, is usually very large, we use mini-batch based gradient descent to train the network. We present the network with a mini-batch of n' data points which consists of n'_l labeled data points (green circles) and n'_u unlabeled data points (red circles), i.e. $n' = n'_l + n'_u$ with $n'_l \leq n_l$ and $n'_u \leq n_u$. The cross-entropy loss is computed over the labeled data in the mini-batch and entropy loss is computed over the unlabeled data in the mini-batch. The negative gradient of the joint loss function with respect to the mini-batch is back - propagated in order

[1]The derivation of the gradient computations in detailed in Appendix A

to train the DBN. When the network has seen all the data points in the training set (labeled and unlabeled), we consider it as one epoch. We repeat the training procedure over multiple epochs until convergence and consider this as one training iteration t of the active learning algorithm. At the end of every iteration t, we sample the most informative batch of unlabeled data points (highest entropy data points using Equation 5.3) to form B. We obtain the labels for B using an oracle and update the labeled and unlabeled datasets, as discussed earlier. We iterate until we run out of unlabeled data to be labeled or we run out of budget to get them labeled. For ease of representation and evaluation we fix the number of these iterations as T. The pseudo-code of our algorithm is presented in Algorithm 3.

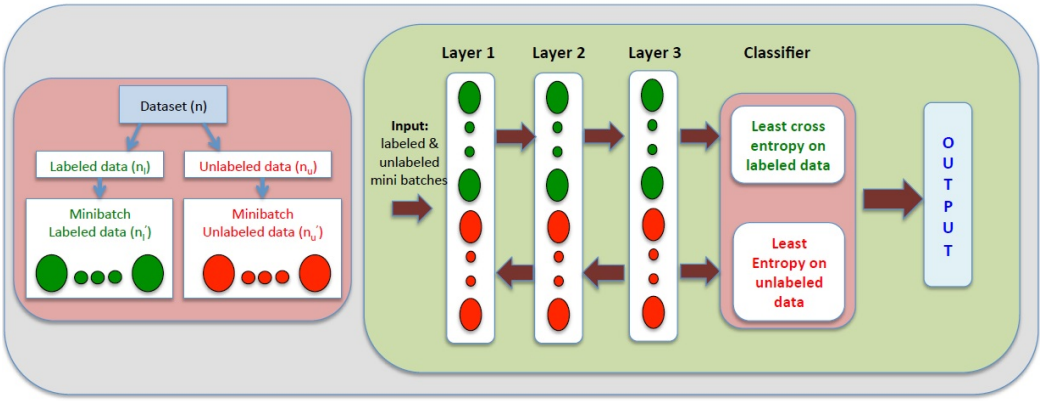


Figure 5.6: Deep Active Learning Network Architecture. Best Viewed in Color, Source: Ranganathan *et al.* (2016c)

Algorithm 3 The proposed Deep Active Learning Algorithm

Input: The labeled set X_l, class labels Y_l, unlabeled set X_u, weight parameter λ, batch size k, maximum number of iterations T

1: **for** $t = 1, 2, \ldots T$ **do**

2: Compute the derivative of the loss function, using Equation 6

3: Train the deep model to obtain the network weights

4: Compute the entropy of each unlabeled sample, using Equation 3

5: Select a batch B containing k unlabeled samples from X_u furnishing the highest entropy

6: Update $X_l \leftarrow X_l \cup B$; $X_u \leftarrow X_u \backslash B$

5.4 Experiments and Results

The following sections detail the implementation specifics of the proposed model, baseline models for comparison and evaluation metrics for validation.

5.4.1 Implementation Details

Our DBN model consists of three hidden layers, with 500 units in each hidden layer. Depending on the dataset we are experimenting with, the number of nodes in the input and the final layers are set to the data dimensions and number of categories of the the dataset respectively. In the unsupervised learning stage, we used a learning rate of 0.05 and number of epochs as 100. We employed minibatch ($n' = 100$), momentum (0.5 for first 5 epochs and 0.9 later) and weight decay (0.0002) techniques for accelerating the learning process and for preventing overfitting respectively. The DBN learning parameters for the supervised finetuning stage are as follows: learning rate = 0.05, number of epochs = 50 and minibatch size = 100 (with 50 labeled and 50 unlabeled data samples). The same deep architecture and parameters were used for

all the competing methods for fair comparison. The weight parameter λ was selected as 1 based on preliminary experiments.

5.4.2 Datasets and Experimental Setup

We studied the performance of the algorithm on a variety of uni-modal and multi-modal datasets from different application domains. These are detailed below:

Uni-modal datasets: We used four uni-modal datasets in our experiments: (*i*) the **VidTIMIT** face recognition dataset (Sanderson (2008) consisting of video and corresponding audio recordings of subjects reciting short sentences; (*ii*) the **Cohn-Kanade (CK)** AU-Coded Expression Database (Kanade *et al.* (2000a)), which is widely used for research in automatic facial image analysis, synthesis and for perceptual studies; (*iii*) the **MNIST** database of handwritten digits (LeCun *et al.* (1998a)); and (*iv*) the **CIFAR 10** dataset (Krizhevsky (2009)) for object recognition, which contains images from 10 different classes of objects.

Multi-modal datasets: We also validated the performance of our algorithm on two multi-modal datasets: (*i*) **emoFBVP**, which contains 23 different emotions enacted by 10 professional actors together with corresponding recordings of Face, Body gesture, Voice (captured using the Microsoft Kinect sensor) and Physiological data (captured using the wearable Zephyr BioHarness) (Ranganathan *et al.* (2016a)) and (*ii*) **MindReading**, which is a bi-modal dataset to study human emotions from audio and vocal cues (El-Kaliouby and Robinson (2004)). For the *emoFBVP* dataset, we also studied the performance of our algorithm on subsets of modalities (face and voice, face, body and voice).

Our objective was to test the performance of the proposed active sampling framework for deep learning and not to outperform the best accuracy results on these datasets; so, we did not follow the precise train/test splits given for many of these

datasets. Each dataset was divided into an initial training set, an unlabeled set and a test set. For a given batch size k, each algorithm selected k instances from the unlabeled pool to be labeled in each iteration. After each iteration, the selected points were removed from the unlabeled set, appended to the training set and the performance was evaluated on the test set. The goal was to study the improvement in performance on the test set with increasing sizes of the training set. The experiments were run for 20 iterations. This setup is similar to previous work by Wang and Shang (2014a). The dataset details are summarized in Tables 5.7 and 5.8.

Dataset	**Training**	**Unlabeled**	**Testing**	**Batch Size**
VidTIMIT	500	20000	8000	100
CK	500	10000	5000	100
MNIST	1000	50000	10000	200
CIFAR	1000	45000	10000	200

Table 5.7: Uni-modal Dataset Details.

Dataset	**Training**	**Unlabeled**	**Testing**	**Batch Size**
emoFBVP	400	20000	10000	80
MindReading	1000	70000	10000	200

Table 5.8: Multi-modal Dataset Details.

5.4.3 Comparison Baselines

Uni-modal datasets: We used the following algorithms as baselines for comparison: (*i*) **Random Sampling**, which selects a batch of unlabeled samples at random from the unlabeled pool; (*ii*) **Active Labeling with Least Confidence (AL-LC)** (Wang and Shang (2014a)) which selects the samples with the smallest of the maximum activations as follows:

$$x_i^{LC} = \underset{x_i}{\operatorname{argmin}} \max_j p_{ij} \tag{5.7}$$

where x_i is the input vector and $p_{ij} = e^{h_{ij}^N} / \sum_{j'} e^{h_{ij'}^N}$, is the probability that data point x_i belongs to class j. h_{ij}^N is the activation of the unit j for x_i in the final layer N before the loss layer; (*iii*) **Active Labeling with Margin Sampling (AL-MS)** (Wang and Shang (2014a)) which selects the samples with the smallest separation between the top two class predictions:

$$x_i^{MS} = \underset{x_i}{\operatorname{argmin}}(p_{i,j'} - p_{i,j''}) \tag{5.8}$$

where j' and j'' are the first and second most probable class labels for sample x_i predicted by the DBN; and (*iv*) **Active Labeling with Entropy (AL-Entropy)** (Wang and Shang (2014a)), which selects the unlabeled samples with the largest class prediction information entropy:

$$x_i^{Entropy} = \underset{x_i}{\operatorname{argmax}} - \sum_j p_{ij} \log p_{ij} \tag{5.9}$$

Multi-modal datasets: Multi-modal learning has gained importance in recent years where each data sample is characterized by multiple features representing distinct statistical properties. Multi-modal active learning has not been studied in the context of deep learning. Existing algorithms query unlabeled samples based on the principle of mutual disagreement. Muslea *et al.* (2006) proposed the *Multi-view (MV)* algorithm in which a separate classification model was trained for each modality (view). A set of *Contention Points* was identified from the unlabeled set, where at least two models produced different predictions; samples were queried from this set using three selection strategies: (*i*) **MV-Naive**, where samples were selected at random from the set of contention points; (*ii*) **MV-Aggressive**, where we select a batch of samples on which the least confident of the models makes the most confident prediction; (*iii*) **MV-Conservative**, where we select a batch of contention points on which the confidence of the predictions made by the models is as close as possible. Cebron and Berthold (2010) proposed a multi-modal active learning algorithm

called *Parallel Universes (PU)* which followed the general Multi-view Active Learning framework but incorporated sample diversity along with uncertainty (entropy) in batch selection. We compared our algorithm against all these methods, together with Random Sampling.

5.4.4 Active Learning Performance

Uni-modal datasets: The results on the uni-modal datasets are depicted in Figure 5.7. In each graph, the x-axis denotes the iteration number and the y-axis denotes the accuracy on the test set. The proposed framework outperforms Random Sampling on all the datasets; the accuracy increases at a faster rate with increasing size of the labeled set. Our algorithm therefore identifies the salient and exemplar instances for manual annotation and attains a given level of performance with much reduced human labeling effort. The AL-LC, AL-MS and AL-Entropy methods depict better performance than Random Sampling, but are not as good as our method. This corroborates the merit of incorporating an entropy based term in the loss function to train the deep belief network and learning the features accordingly, so that the unlabeled samples selected for annotation are maximally informative. The results unanimously lead to the conclusion that our algorithm depicts the best performance consistently across all the datasets.

Multi-modal datasets: The results on the multi-modal emoFBVP and MindReading datasets are depicted in Figure 5.8 (a) and 5.8 (b). We note that Random Sampling depicts comparable performance as the MV-Aggressive and MV-Conservative methods. Thus, a simple method like random selection can sometimes depict good performance. The PU algorithm combining uncertainty and diversity depicts better performance than the Multi-view active learning algorithms. Our method demonstrates the best performance on both datasets; at any given iteration, it attains the

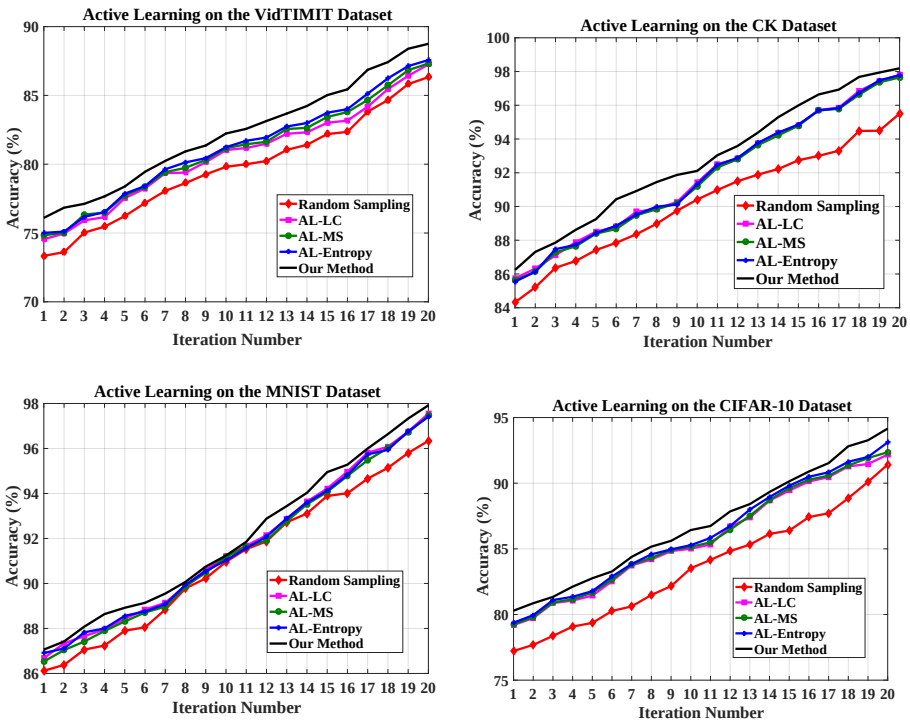


Figure 5.7: Active Learning on the Uni-Modal Datasets. Best Viewed in Color, Source: Ranganathan *et al.* (2016c)

highest accuracy on the test set. Thus, a deep belief network trained to minimize the cross-entropy loss on the labeled data together with the entropy loss on the unlabeled data succeeds in selecting the exemplar unlabeled samples for manual annotation in both uni-modal and multi-modal settings and achieves a given level of accuracy with the least amount of human effort.

We also studied the performance of our framework on different subsets of modalities of the *emoFBVP dataset*. Figure 5.8 (c) and Figure 5.8 (d) depict the results when using only the face and voice modalities and only the face, body and voice modalities respectively. The results depict a similar trend, further corroborating the generalizibility of our framework. A two-sided paired *t-test* at the significance level of $\alpha < 0.05$ reveals that the improvement in performance achieved by our method is statistically significant for all the datasets.

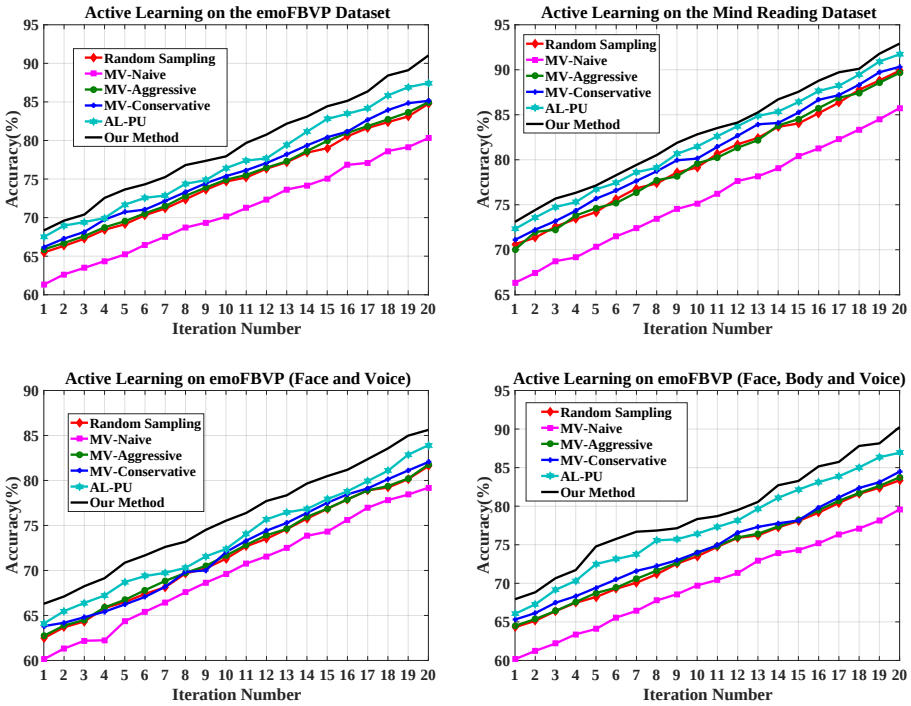


Figure 5.8: Active Learning on Multimodal Datasets. (A) and (B) Active Learning on the Multi-Modal Datasets *emoFBVP* and MindReading. (C) and (D) Active Learning on Subsets of Modalities of the emoFBVP Multi-Modal Dataset. Best Viewed in Color, Source: Ranganathan *et al.* (2016c)

5.5 Conclusion and Future Work

In this chapter, we proposed a novel algorithm to actively sample unlabeled instances that are most promising in training a deep belief network model. We introduced a loss function based on softmax and entropy losses and trained the deep model to optimize the loss function. The network therefore gets specifically trained for the active learning task and depicts much better performance than a network trained only on the conventional softmax loss. To the best of our knowledge, this is the first research effort to incorporate an active learning based criterion in the loss function and train the deep network to optimize the objective. Our experimental results on a variety of uni-modal and multi-modal datasets from different application domains depict the promise and potential of the method for real-world image

recognition applications.

As part of future research, we plan to extensively validate the performance of the framework on other computer vision applications such as image segmentation, image search and retrieval, clustering and object detection among others. We also intend to study the performance of the network trained to optimize other loss functions specific to active learning. For instance, some active learning algorithms include a diversity based criterion (besides uncertainty), to ensure that the samples selected for annotation also have a high diversity among them (Chakraborty *et al.* (2015b)); representativeness based algorithms explored in Shen *et al.* (2004), enforce the selected samples to be good representatives of the unselected unlabeled samples. We plan to integrate these criteria in the loss function and study their effects on the results. We would also like to modify our algorithm to work on video datasets, where the temporal structure provides useful information that are less obvious in static images.

5.6 Summary

This chapter began with a brief overview of active learning methodologies in literature and then continued to introduce the field of deep active learning - where ideas from deep learning and active learning are combined to learn intelligent models for classification and/ regression. We then proposed a novel active sampling algorithm to identify the salient and exemplar unlabeled samples to be mannually annotated to train DBNs. The proposed method was validated on single-label image classification on a variety of benchmark datasets for different applications and shown to outperform current state-of-the-art techniques.

Chapter 6

DEEP ACTIVE LEARNING FOR MULTI-LABEL IMAGE CLASSIFICATION

Multi-label classification is a generalization of conventional classification problems, where each data sample can have multiple labels. Deep learning algorithms learn a highly discriminating set of features for a given machine learning task and have depicted commendable performance in a variety of applications, including multi-label learning. However, training a deep model necessitates a large amount of labeled training data, which is an expensive process in terms of time, labor and human expertise. The problem is further compounded in a multi-label learning application, as the human oracle needs to consider the presence/absence of every label to annotate a single data sample. Active learning algorithms automatically identify the salient and exemplar samples from large amounts of unlabeled data and tremendously reduce human annotation effort in inducing a machine learning model. In this chapter, we propose a novel active learning framework to select the most informative unlabeled samples to train a deep model for multi-label classification. We introduce a novel loss function specific to the task of multi-label active learning and train the model to optimize this loss. To the best of our knowledge, this is the first research effort to develop a deep learning framework for multi-label active learning. Our experiments on three challenging, real-world multi-label datasets show that our method outperforms the state-of-the-art multi-label active learning algorithms, corroborating its potential for real world image classification applications.

In a conventional classification problem, each data sample is assumed to belong to a single class. However, many applications necessitate a more generalized setting,

where each data sample can belong to multiple classes simultaneously. This is referred as the *multi-label classification* setting. For instance, classifying a natural scene image is a multi-label problem, as a single image can have multiple classes (like sky, sunset, mountains and trees) associated with it.

A fundamental challenge in training a deep neural network is the requirement of large amounts of labeled training data. While gathering large quantities of unlabeled data is cheap and easy, annotating the data (with class labels) entails significant human effort. In a multi-label learning setting, the problem is further aggravated as the presence/absence of each class needs to be checked separately to annotate a single unlabeled sample. Often, the number of possible classes is of the order of hundreds, which tremendously increases the labeling burden on the human annotator. Therefore, developing algorithms that reduce human effort in training deep models in a multi-label setting is of paramount practical importance. Active learning algorithms alleviate this problem by selecting the salient and informative samples from vast amounts of unlabeled data. This not only reduces the human effort in training machine learning models, but also produces models with much better generalization capabilities, as they get trained on the salient examples from the underlying data population. Specifically, batch mode active learning (BMAL) algorithms, where a batch of unlabeled samples are simultaneously selected for manual annotation, have been widely used in computer vision applications with encouraging empirical results.

In our work, we leverage the feature learning capabilities of deep neural networks and propose a novel framework to address the challenging problem of multi-label active learning. We integrate an active sample selection criterion in the loss function and train the deep network to optimize the function. First, we propose a framework

without considering the correlation among the multiple labels using CNNs. Second, we model the correlations that exist among the multiple labels using Long Short Term Memory networks (LSTMs).

6.1 Proposed Framework

We propose to exploit the feature learning capabilities of deep networks to identify the most informative unlabeled samples for multi-label active learning. We do this by appending an active sampling criterion to the objective function and train the deep network to optimize the function. In our framework, we formulate a loss function which captures the active sampling criterion and train the deep network to optimize that loss. The feature representations learnt by the network are then specially tailored to the active learning task and enable it to better identify the samples that can augment maximal information to the model. Formally, let $X^L = \{x_1, x_2, \cdots, x_{n_l}\}$ denote the labeled training set with n_l samples and $Y^L = \{y_1, y_2, \cdots, y_{n_l}\}$ denote the corresponding multi-labels with $y_i \in \{0,1\}^{1\times M}$ where M is the number of unique labels in the dataset. The learner is also presented with an unlabeled set $X^U = \{x_{n_l+1}, x_{n_l+2}, \cdots, x_{n_l+n_u}\}$. Let $X = X^L \cup X^U$ denote the union of the disjoint subsets X^L and X^U and $N = n_l + n_u$. The objective is to select a batch B with k unlabeled samples for manual annotation (k being the batch size) such that the modified learner, trained on $X^L \cup B$, has maximal generalization capability.

In a multi-label active learning setting, we have an oracle who provides labels at a fixed cost for every label in the unlabeled data. From the unlabeled set X^U, the oracle is given a batch of points B to be labeled. The labeled batch B is then added to the labeled set X^L along with the corresponding labels Y^L; i.e. $X^L \rightarrow X^L \cup B$.

These data points are removed from the unlabeled set X^U to ensure that X^L and X^U are disjoint, i.e. $X^U \rightarrow X^U \backslash B$. A new and improved multi-label classifier is estimated using the augmented labeled sets $\{X^L, Y^L\}$. This procedure is repeated for a fixed number of iterations, T.

The challenge in multi-label active learning is to identify the most informative set of unlabeled samples to be labeled by the oracle while considering the correlation between the multiple labels. Intuitive reasoning leads us to select data points that have been classified with highest uncertainty by the existing multi-label classifier. Entropy (from Information Theory) is a widely used measure of uncertainty regarding label assignment. The set B can therefore be chosen by selecting the data points with the largest uncertainty. In standard active learning settings, a classifier is first trained on the multi-labeled data and used to obtain the predictions for the unlabeled data. Entropy is then applied to obtain the uncertainty of such a classifier prediction. In this two-step approach, the unlabeled data does not play a role in training the multi-label classifier. In our deep active learning model, we append the entropy measure to the conventional loss function and train the network with both labeled and unlabeled data using a novel joint training objective.

Our intuition behind training multi-label active learning models using the joint objective function is based on the following advantages: (i) all the available data is used to train the model. This results in a much more robust and adaptive model when compared to training with only the labeled data; (ii) training with unlabeled data eventually results in learning an effective multi-label classifier for unlabeled data and (iii) the deep network trains itself to reduce entropy along with estimating a robust multi-label classifier (by reducing cross-entropy). This joint training objective ensures

that the data points with the largest entropy that are selected to form B, are the most uncertain and informative unlabeled data points with respect to the classifier. Over successive iterations, the positive effects of the joint training with labeled and unlabeled data get enhanced.

6.2 Multi-Label Active Learning Without Label Correlation

In this framework, the multiple labels of a data sample are assumed to be independent of each other. We propose a method to solve the problem of multi-label active learning using CNNs without considering label correlation. We introduce a novel joint objective function that integrates an active selection criterion to the conventional loss function, and train the CNN to optimize this function.

6.2.1 Sigmoid Cross-Entropy Loss for Labeled Data

Let the ground truth of all the n_l labeled samples be $Y^L \in \{0,1\}^{n_l \times M}$ and $y_{i,j}$ indicates the $(i,j)^{th}$ element of Y^L . We have the predictions $\widehat{Y} \in \mathbb{R}^{n_l \times M}$ and $\widehat{y}_{i,j}$ indicates the $(i,j)^{th}$ element of $\widehat{Y}$. The loss over the labeled data was designed as the multi-label sigmoid cross-entropy loss (K.Zhaoa *et al.* (2016)) given by:

$$\begin{aligned}\mathcal{C}(Y^L, \widehat{Y}) = -\frac{1}{n_l}\sum_{n=1}^{n_l}\sum_{m=1}^{M}[y_{nm}\log(\widehat{y}_{nm}) \\ + (1-y_{nm})\log(1-\widehat{y}_{nm})].\end{aligned} \tag{6.1}$$

6.2.2 Entropy Loss for Unlabeled Data

Uncertainty sampling is the most widely used strategy for active sample selection and entropy is a well-accepted measure to quantify the uncertainty of a classification model. We therefore append an entropy based term to the loss function, which quantifies the level of uncertainty of the network on an unlabeled sample. The entropy of an unlabeled sample in the multi-label setting is defined as in Clare and King (2001):

$$\mathcal{H}(\widehat{Y}) = -\frac{1}{n_u}\sum_{n=n_l+1}^{n_l+n_u}\sum_{m=1}^{M}[\widehat{y}_{nm}\log(\widehat{y}_{nm}) + (1-\widehat{y}_{nm})\log(1-\widehat{y}_{nm})]. \tag{6.2}$$

6.2.3 Joint Objective for Multi-label Active Learning

The CNN is trained by combining both the labeled and unlabeled data with the objective of obtaining a robust classifier for labeled data and reducing the uncertainty of classification on the unlabeled data. The joint objective function for multi-label deep multi-label active learning is thus given by:

$$\mathcal{L}(X^L, X^u, Y^L) = \mathcal{C}(Y^L, \widehat{Y}) + \lambda\mathcal{H}(\widehat{Y}) \tag{6.3}$$

where $\mathcal{C}(Y^L, \widehat{Y})$ and $\mathcal{H}(\widehat{Y})$ are as in equations (6.1) and (6.2) respectively. Here $\lambda \geq 0$ controls the relative importance of the two terms. The function is optimized using the mini-batch gradient descent algorithm to train the network [1].

Figure 6.1 illustrates the network architecture of the CNN used for multi-label deep active learning without label correlation. The CNN model employed consists of eight layers. The input images are re-sized to 256×256. The weight parameter λ in

[1]The gradient computation is detailed in Appendix B

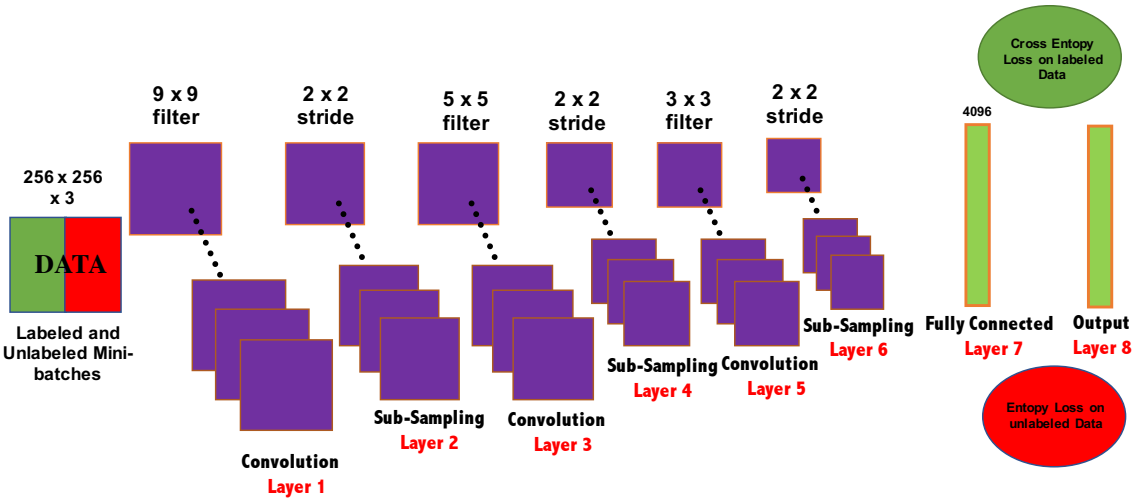


Figure 6.1: Architecture of Multi-Label CNN Model Without Label Correlation. Best Viewed in Color.

equation (6.3) was selected to be 1 giving equal weight to both terms in the equation. We compute the entropy of each unlabeled sample in X^U using equation (6.2) and select k samples furnishing the maximum entropies to form batch B. We update the labeled and the unlabeled data sets and repeat for T iterations.

6.2.4 Training and Implementation Details

The training procedure is repeated over multiple epochs until convergence, and this is considered as one training iteration t. At the end of t, the model is applied on all the unlabeled samples and those furnishing the highest entropies are selected to form a set B. The joint loss ensures that the data points with the largest entropy that are selected to form B, are the most uncertain and informative unlabeled data points with respect to the classifier. This is a much more informed way of actively selecting unlabeled samples for annotation, since the active learning criterion is embedded in the loss function used to train the network. These samples are appended to the labeled set and a new classifier is estimated with the updated labeled and unlabeled

data. We repeat this for a fixed number of iterations, T.

The CNN model employed consists of eight layers. Before feeding inputs to the first layer of the CNN, the images in the datasets were resized to 256×256. The first convolution layer has a filter size of 9×9 and 96 feature maps, the second convolution layer has a filter size of 5×5 and 256 feature maps and the third convolution layer has a filter size of 3×3 and 256 feature maps. All the sub-sampling layers have a stride of 2×2. The fully connected layer has an output size of 4096. The fully connected layer combines these features and feeds them to the classifier. The weight parameter λ was selected to be 1 based on preliminary experiments.

The multi-label deep active CNN model treats the labels independently and does not exploit the inherent dependencies among the multiple labels. Research has shown that multi-label datasets exhibit strong label correlations (Xue *et al.* (2011)). For instance, the labels *sea* and *boat* usually appear together, while the labels *trees* and *desert* almost never co-occur. We now describe a multi-label active learning framework that models label correlations using RNN models, which can potentially improve the learning performance.

6.3 Multi-Label Deep Active Learning With Label Correlation

Here, we propose a multi-label deep active learning framework for image classification by effectively learning the higher order dependencies among the multiple labels. We use CNNs coupled with Long-Short Term Memory (LSTM) cells to learn a joint low-dimensional image-label embedding to model the semantic relevance among images and labels. In this model, the CNN is used to produce high level representation

of the image and the LSTM models the label dependencies. We incorporate an active sampling criterion in the objective function used to train the model. This model is inspired from Wang *et al.*'s CNN-RNN model (Wang *et al.* (2016)).

Let the predicted label at time step t be $\widehat{y}(t)$. The LSTM model predicts multiple labels by finding the prediction path that maximizes the *a priori* probability. The probability of a prediction path is obtained as the product of the *a priori* probability of each label given the previous labels in the prediction path.

In every image, we represent each of the labels as a one-hot vector y_k which is all zeros and a one in the k^{th} spot. We then obtain the label embedding $embed_k$ by multiplying y_k with an embedding matrix ($embed_{matrix}$).

$$embed_k = embed_{matrix}.y_k. \tag{6.4}$$

The LSTM takes $embed_k(t)$; the label embedding at time step t; as input and produces a hidden state $h(t)$ as output using the standard LSTM equations. The output of the recurrent layer and the image representation are projected into the same low-dimensional space as the label embedding.

$$x(t) = RELU(W_I.I + W_h.h(t)) \tag{6.5}$$

Here, $RELU$ is the rectified linear unit activation function, I is the CNN image representation and W_I, W_h are the projection matrices for the image and hidden state respectively. The number of columns of W_I and W_h are the same as the label embedding matrix $embed_{matrix}$. The label scores are computed by multiplying the transpose of $embed_{matrix}$ and $x(t)$.

$$s(t) = embed_{matrix}{}^{T}.x(t). \tag{6.6}$$

The predicted probability is computed using Softmax normalization on the scores.

$$p(t) = Softmax(s(t)). \tag{6.7}$$

A prediction path is a sequence of labels $(l_1, l_2, l_3, \cdots, l_N)$, where the probability of each label l_t can be computed with the information of the image I and the previously predicted labels $l_{t-1}, l_{t-2}, l_{t-3}, \cdots, l_1$. The LSTM model predicts multiple labels by finding the prediction path that maximizes the a-priori probability. The probability of a prediction path is obtained as the product of the a-priori probability of each label given the previous labels in the prediction path.

$$\begin{aligned} l_1, l_2, l_3, \cdots, l_k &= \underset{l_1, l_2, \cdots, l_k}{\operatorname{argmax}} P(l_1, l_2, l_3, \cdots, l_k \Big| I) \\ &= \underset{l_1, l_2, \cdots, l_k}{\operatorname{argmax}} \prod_{j=1}^{k} P(l_j \Big| I, l_1, l_2, \cdots, l_{j-1}) \\ &= \underset{l_1, l_2, \cdots, l_k}{\operatorname{argmax}} P(l_1 \Big| I) * P(l_2 \Big| I, l_1) * \cdots * P(l_k \Big| I, l_1, l_2, \cdots, l_{k-1}) \end{aligned} \tag{6.8}$$

The LSTM predicts labels in order of decreasing frequency. The cross-entropy loss function punishes the model not only if it predicts a label that does not apply to the image, but also when it predicts a true label in the wrong order of the sequence.

6.3.1 Loss on Labeled Data

First, we decompose the multi-label prediction as an ordered prediction path. A prediction path is a sequence of labels $(y_1, y_2, \cdots, y_M)$, where the probability of each label y_m, at time step t, can be computed with the information of the image I and

the previously predicted labels. We train our model with a combination of labeled and unlabeled data. Let y_l be the vector of ground truth labels for an image x_l sorted from the most frequent label to the least frequent label in the dataset. During training, at time step t, we feed $y_l(t-1)$ as input, and the loss the model incurs is the cross-entropy loss between the prediction $\widehat{y}_l(t)$ and the next true label $y_l(t)$ as:

$$\mathscr{C}_{x_l}(y_l(t), \widehat{y}_l(t)) = -\, y_l(t) \log(\widehat{y}_l(t)) \tag{6.9}$$

This formulation has edge cases at the first time step and last time step. To begin, we feed the LSTM with the embedding of a special START label and force the LSTM to predict a special END label at the end of the sequence.

6.3.2 Loss on Unlabeled Data

When training using unlabeled data, we begin by feeding the network with the START label and employ a beam search approach to predict the set of labels for an image x_u. The loss on unlabeled data is the entropy loss on the predicted label $\widehat{y}_u(t)$ given by:

$$\mathscr{H}_{x_u}(\widehat{y}_u(t)) = -\, \hat{y}_u(t) \log(\widehat{y}_u(t)) \tag{6.10}$$

6.3.3 Joint Objective for Training

The proposed joint objective function for training is given by:

$$\mathscr{L}_{x_l, x_u}(t) = \mathscr{C}_{x_l}(y_l(t), \widehat{y}_l(t)) + \alpha \mathscr{H}_{x_u}(\widehat{y}_u(t)) \tag{6.11}$$

Here $\alpha \geq 0$ controls the relative importance of the entropy loss. We accumulate the joint loss at each time step and estimate the average loss for every data point to obtain the total loss the model incurs. Back-propagation through time algorithm is used to update the weights of the network. In order to obtain the most informative unlabeled

samples to form batch B, we forward propagate X^U and compute the average entropy loss on every unlabeled sample. We then select k samples furnishing the highest average entropies to form batch B. We update the labeled and the unlabeled data sets as explained in Section 2 and repeat for T iterations.

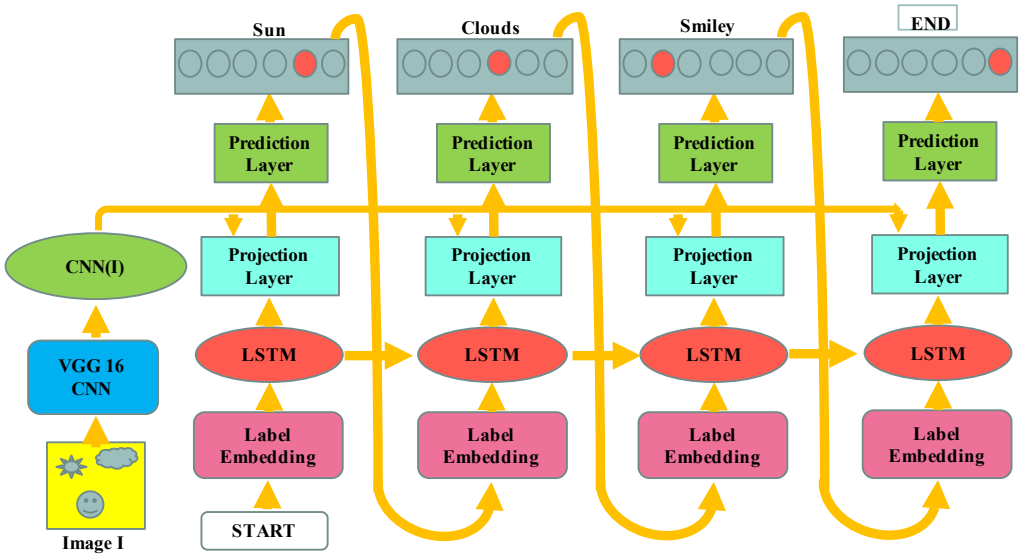


Figure 6.2: Architecture of Multi-Label CNN-LSTM Model With Label Correlation.. Best Viewed in Color.

Figure 6.2 describes the architecture of the deep active CNN-LSTM model. The network is first fed with the START label and it first predicts Sun, this is then fed as input to the second step which predicts Clouds, which when fed to the next step predicts Smiley, and when that is fed to the fourth step, we sample the final END label, stopping the process. For the CNN module, we use the 16- layer $VGG16$ network pretrained on the ImageNet 2012 dataset. To keep training simple, we do not finetune the CNN. The dimensions of the label embedding layer was 64 and that of the LSTM recurrent layer was 512. We used RMSprop optimization with a minibatch size of 100 to avoid gradient vanishing/exploding issues. The value of momentum used to optimize the model is 0.9. We employed dropout regularization and hyperparameter

tuning to obtain the values for learning rate and weight decay. We also employed dropout regularization to avoid overfitting the data. We kept the CNN part of our architecture unchanged for simplicity.

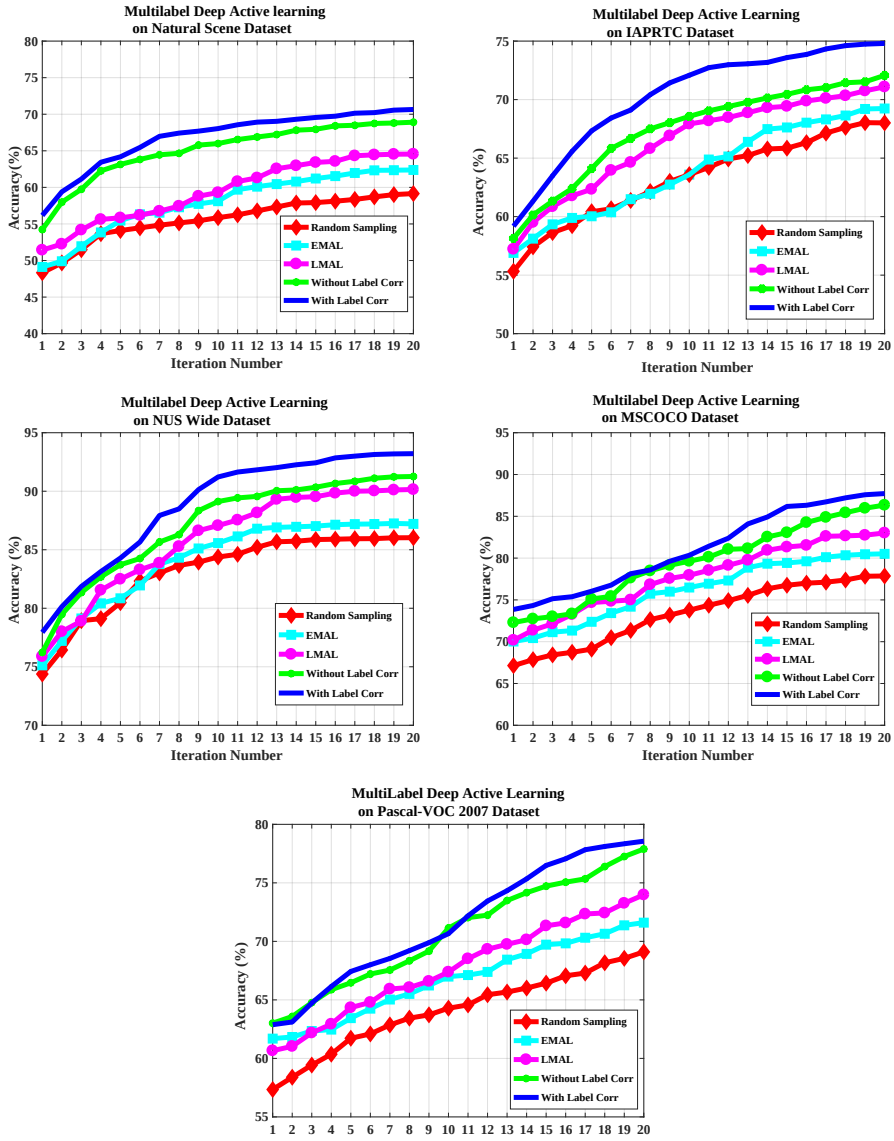


Figure 6.3: Deep Active Learning on Benchmark Multi-Label Datasets. Best Viewed in Color.

6.4 Experiments and Results

We used five benchmark multi-label datasets to evaluate our framework: the Natural Scene (Zhang and Zhou (2007)), IAPR-TC-12 (Wang *et al.* (2010)), NUS-WIDE-OBJECT(Chua *et al.* (2009)), MSCOCO (Lin *et al.* (2014)) and PASCAL-VOC 2007 (Everingham *et al.* (2010)). We used the example based active learning (*EMAL*) algorithm and example-label based active learning (*LMAL*) algorithms (Wu *et al.* (2014)) as baselines for comparison, as these frameworks were recently shown to outperform several multi-label active learning algorithms. We also used *Random Sampling* as a comparison baseline.

Our objective was to test the performance of the proposed framework and not to outperform the best accuracy results on these datasets. The datasets were divided into an initial training set, an unlabeled set and a test set. Each algorithm (baseline and proposed) selected k instances from the unlabeled pool to be labeled in each iteration. After each iteration, the selected points were removed from the unlabeled set and appended to the training set. The performances of all the algorithms were evaluated on the test set. The objective was to study the improvement in performance of the algorithms on the test set with each iteration. Our experiments were run for $T = 20$ iterations. The *EMAL* and *LMAL* algorithms were not proposed in the context of deep learning. However, to facilitate fair comparison with our method, we train the CNN described in Section 3 using the standard loss function and then apply the active selection criterion on the unlabeled data.

The performance of the proposed algorithm is depicted in Figure 6.3. In each figure, the x-axis specifies the iteration number and y-axis denotes the accuracy on the test set. We observe that *Random Sampling* depicts the least performance on all

the five datasets. The *LMAL* method does better than *EMAL*, which in turn does better than *Random Sampling*. Our deep active CNN model without label correlation performs better than all the baselines. This shows the usefulness of integrating an active sample selection criterion in the loss function to train a deep network. Our deep active model with label correlation consistently depicts the best performance across all datasets. This further depicts the merit of incorporating label correlations in multi-label active learning using deep learning architectures.

Figure 6.4 presents an analysis of the unlabeled samples queried for annotation. In Figure 6.4(a), we compare the 200 samples that are chosen to form batch B after iteration number 12 on the NUS-WIDE-OBJECT dataset using the proposed methods and *Random Sampling*. Figure 6.4(b) and 6.4(c) show the heatmap of the corresponding label co-occurence matrix and frequency of label occurence in the dataset. We know from (Wang *et al.* (2016)) that the CNN-LSTM model trained using the conventional cross entropy loss easily predicted labels with high co-occurence and labels that occurred frequently in the data.

From Figure 6.4 , we see that our deep active model with label correlation attempts at selecting two types of samples - those that exhibit low dependency on other labels and those that occur less frequently in the data. It selects fewer samples of labels that co-occur and labels that occur frequently.

For example, from Figure 6.4 (a), we see that the tall violet bars correspond to the labels "Bear", "Book", "Computer", "Flags" and "Zebra". From Figure 6.4 (b), we see that the above labels have low co-occurence and, from Figure 6.4 (c), are the least frequently occurring labels. The short violet bars in Figure 6.4 (a) correspond to

labels “Boats”, “Sun”, “Trees”, and “Vehicle”. These labels have high co-occurence (Figure 6.4 (b)) and occur very frequently (Figure 6.4 (c)). Our CNN model without label correlation also concentrates on selecting less frequent samples (refer to the green bars in Figure 6.4 (a) and the corresponding label frequencies in Figure 6.4 (c)). This corroborates the significance of the proposed joint training objective and the effectiveness of the active sampling criterion employed.

The samples selected by *Random Sampling* do not bear any specific trend to the label/sample distribution in the dataset, as seen from the yellow bars in Figure 6.4 (a). This shows that the proposed deep active models are able to successfully augment the training set with under-represented samples and labels, thereby contributing to the improvements in the performance of the models.

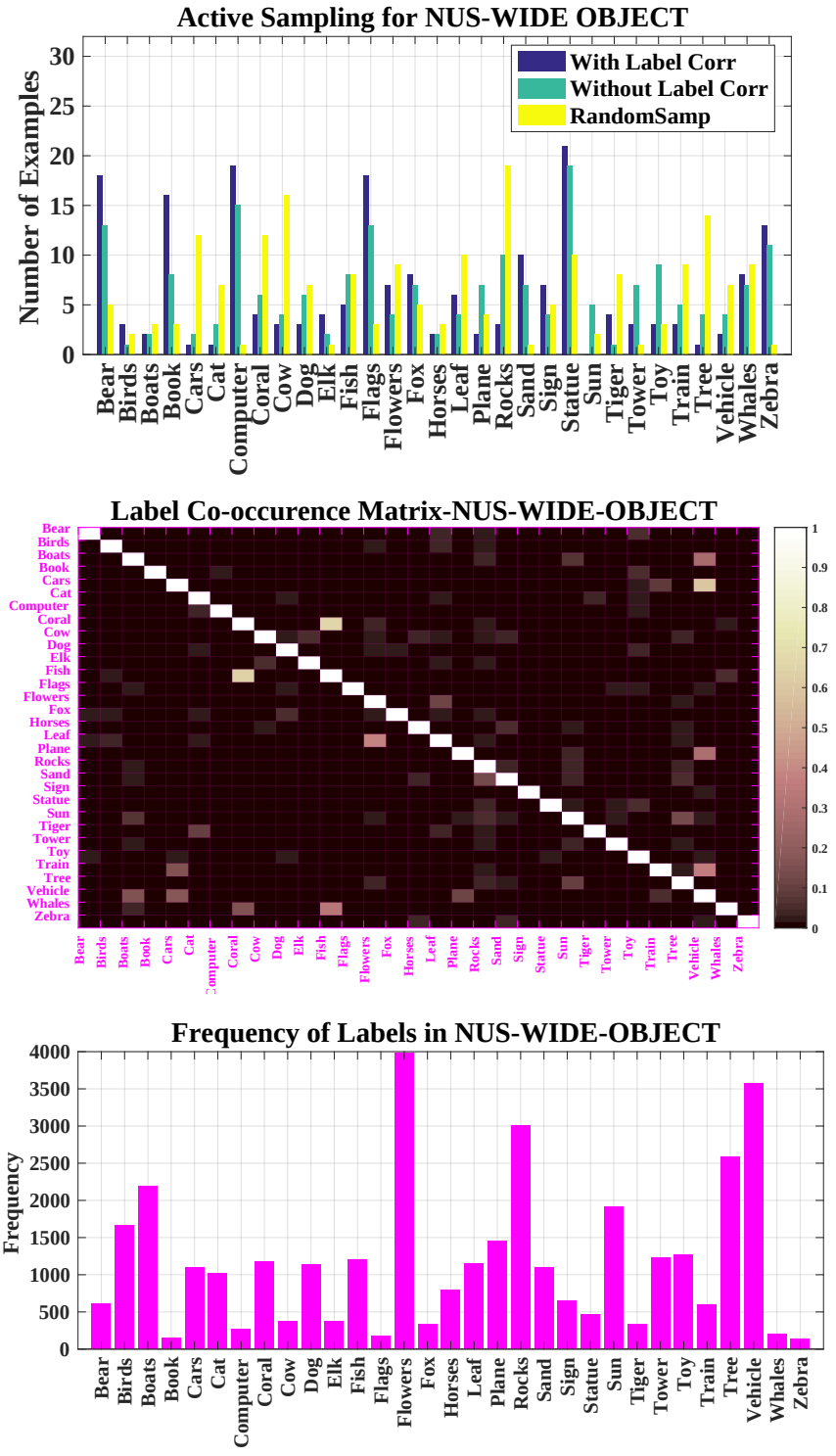


Figure 6.4: Analysis of the Unlabeled Samples Queried for Annotation. Best Viewed in Color.

6.5 Summary

In this chapter, we proposed a novel approach to solve the problem of multi-label deep active learning for image classification. To the best of our knowledge, this is the first research effort to exploit the feature learning capabilities of CNN and LSTM architectures for active learning in the multi-label setting. We proposed two deep learning based frameworks, one which does not consider label correlation and another model which does, to address the challenging problem of multi-label active learning. First, we proposed a framework to solve this problem without considering the correlation among the multiple labels using CNNs. Second, we modeled the correlations that exist among the multiple labels using CNN-LSTMs in an active learning setting.

We proposed a multi-label deep active learning framework that did not model the inherent label correlations and a framework that modeled the relationships between the multiple labels. We successfully integrated an entropy based active sampling criterion in the loss function and used this novel joint objective to train the deep models. Our empirical results on benchmark multi-label datasets showed that the proposed models outperformed state-of-the-art multi-label active learning algorithms, thereby corroborating the potential of our methods for real-world classification problems.

Chapter 7

DEEP ACTIVE LEARNING FOR IMAGE REGRESSION

Image regression is the problem of predicting continuous values such as angles or distances by analyzing image data. Regression-based computer vision applications require large amounts of manually annotated training data, which is expensive to acquire. Active Learning (AL) algorithms are known to maximize the performance of a learning model using as few labeled training examples as possible. These algorithms automatically identify the informative samples from large amounts of unlabeled data and significantly reduce human annotation efforts in inducing a learning model. Additionally, deep models (especially Convolutional Neural Networks (CNNs)) have effectively contributed to improve the performance in a variety of regression applications. In this paper, we exploit the feature learning capabilities of deep neural networks and propose a novel paradigm to address the problem of active learning for regression. We use Expected Model Output Change (EMOC) as the active selection criterion and integrate it within the objective function used to train the deep active model. The resulting model optimizes this novel objective function and learns from salient examples that cause maximum change to the current model. Extensive empirical results on benchmark regression datasets demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation.

In regression-based computer vision applications, each data sample is associated with a set of continuous values. Image regression is the task of predicting these continuous quantities by studying image data. A fundamental challenge in training models for image regression is the requirement of large amounts of labeled training data.

The rapid escalation of technology and the widespread emergence of technological equipment has resulted in the generation of humongous amounts of digital data in the modern era. However, while gathering large quantities of unlabeled data is cheap and easy, annotating the data is an expensive process in terms of time, labor and human expertise. Thus, developing algorithms to minimize human effort in training models for image regression is of immense practical importance. Passive learning randomly selects training examples according to the underlying data distribution and sends them to the oracle for manual annotations. This process is cost intensive with respect to both time and labor; forcing a trade-off between data quality and algorithm performance. Also, not all examples chosen by passive learning contribute constructively to the training process. Active learning algorithms have shown to successfully reduce the cost of data annotation by effectively selecting the most informative samples that maximize the accuracy of the model when labeled and added to the training set. In the recent years, AL has been applied in many machine learning applications (Campigotto *et al.* (2014); de Fortuny and Martens (2015); Zliobaite *et al.* (2014)).

While AL has been extensively studied for classification, AL for regression is much less explored (Lewis and Gale (1994); Settles and Craven (2008); Tong and Koller (2001)). The output for regression is a continuous value (and not posterior probabilities), therefore margin-based sampling strategies are not applicable. Also, distance-based sampling methods are unsuitable as there is no concept of distance in regression tasks. Existing work on AL for regression can be found in (Burbidge *et al.* (2007); Willett *et al.* (2006); Sugiyama and Nakajima (2009); Sugiyama (2006); Yu and Kim (2010); Cohn *et al.* (1996)). Contributing to the need for a general AL framework for regression, Cai *et al.* proposed a model that maximizes the Expected Model Output Change (EMOC) (Cai *et al.* (2013)). The proposed criterion considers

the capacity of unlabeled examples to change the current model in order to tackle the problem of AL for regression. Inspired by the Stochastic Gradient Descent (SGD) update rule, where the model parameters are updated repeatedly using the gradient of the loss with respect each training example, the gradient of the error with respect to a candidate example is used to estimate the model changes. In 2016, Kading *et al.* proposed a generalization of the EMOC principle for deep neural network architectures and applied it for image annotation.

Deep learning algorithms have recently emerged as a dominant machine learning tool to learn representative features for classification and regression tasks (LeCun *et al.* (2015)). Architectures such as the CNNs, RNNs, etc. have created a paradigm shift in multimedia computing applications. DL has been widely explored in computer vision and has achieved tremendous improvements in several vision tasks including image recognition (Krizhevsky *et al.* (2012)), object detection (Girshick *et al.* (2014)), multimodal emotion recognition (Ranganathan *et al.* (2016a,b)) and image segmentation (Liu *et al.* (2015)) among others. Besides classification, CNNs have also been effectively trained for regression tasks such as human pose estimation (Li and Chan (2014); Toshev and Szegedy (2014)), object detection (Szegedy *et al.* (2013)), facial landmark detection (Sun *et al.* (2013)) and depth prediction (Eigen *et al.* (2014)).

In this chapter, we exploit the virtue of deep networks to learn rich sets of features and fuse ideas of DL and AL to propose a novel deep active paradigm for regression. We use Expected Model Output Change (EMOC) as the active selection criterion and integrate it within the objective function used to train the deep active model. The resulting model optimizes this novel objective and learns from salient examples that cause maximum change to the current model. Extensive empirical results on bench-

mark regression datasets demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation. Research in deep active learning is still in a nascent stage (Ranganathan *et al.* (2016c); Stark *et al.* (2015b); Wang and Shang (2014b); Zhou *et al.* (2010b)). To the best of our knowledge, this is the first research effort to develop a paradigm that combines DL and AL for regression by formulating a novel objective function.

The rest of the chapter is organized as follows: we present a survey of related techniques in section 7.1; section 7.2 details the principle of EMOC, the proposed deep active paradigm and the CNN model used are presented in section 7.3 and 7.4 respectively. The experiments and results are analysed in section 7.5. Finally, we present our conclusions in section 7.7.

7.1 Related Work

In this section, we present a brief survey of existing work in deep learning and active learning for regression.

7.1.1 Deep Learning for Regression

A large number of regression based deep learning algorithms have been recently proposed. Here, the goal is to predict a set of continuous values as output. Recently, CNNs have been successfully applied for human pose estimation, where the regressed values correspond to the positions of the body joints on the image plane (Li and Chan (2014); Pfister *et al.* (2014); Toshev and Szegedy (2014)). Sun *et al.* use CNNs effectively to predict the facial fiducial points in facial landmark detection (Sun *et al.*

(2013)). Szegedy and Jaderberg *et al.* use deep networks for object and text detection and predict a bounding box for localization (Szegedy *et al.* (2013); Jaderberg *et al.* (2016)). The above deep models use the conventional $L2$ loss function for training. Zhang *et al.* introduced a CNN optimized for landmark detection and attribute classification (Zhang *et al.* (2014)). They combine the standard $L2$ loss function with the Softmax classification function to increase robustness to outliers. Wang *et al.* combine bounding box localization with object segmentation using a similar approach (Wang *et al.* (2014)). Gkioxari *et al.* use a loss function composed of a body pose estimation term and an action detection term (Gkioxari *et al.* (2014)). Dosovitskiy and Eigen *et al.* use multiple $L2$ loss functions for object generation and depth estimation (Dosovitskiy *et al.* (2015); Eigen *et al.* (2014)). From the above survey, we see that deep models (specifically CNNs) trained using the $L2$ loss function can be applied effectively for regression tasks. Therefore, we use CNNs as our preferred deep model in this work.

7.1.2 Active Learning for Regression

In literature, work targeting AL for regression is less explored when compared to AL methods developed for classification. Here, we summarize some of them.

Willett *et al.* (2006) provide a theoretical analysis of AL in the context of regression . Population based AL methods were proposed by Sugiyama (2006) using the weighted least-squares learning where they predict the conditional expectation of the generalization error given the input training points. A theoretically optimal AL algorithm was proposed by Sugiyama and Nakajima (2009). This directly minimizes the generalization error by employing an additive regression model. Freund *et al.* (1997)

applied a variance-based Query-by-Committee (QBC) framework to regression. Cohn *et al.* (1996) minimized the output variance to reduce the generalization error. Yu and Kim (2010) provided passive sampling heuristics based on the geometric characteristics of the data . Freytag *et al.* (2014) proposed an approach to measure the expected change of model outputs. For each example in the unlabeled set, the expected change of model predictions is calculated and marginalized over the unknown label. The resulting score for each unlabeled example is used for active learning with a broad range of models and learning algorithms. Wenben Cai Cai *et al.* (2013) presented a novel data sampling solution which queries the example leading to the largest model change.

Most regression-based AL techniques are developed only for sequential mode. Batch Mode Active Learning (BMAL) techniques are very useful in practice and it is highly desirable to derive BMAL methods in the context of regression. Existing BMAL algorithms are derived with classification models and cannot be directly generalized to regression (Azimi *et al.* (2012); Belagiannis *et al.* (2014); Brinker (2003b); Chakraborty *et al.* (2015a); Chattopadhyay *et al.* (2013); Guo and Schuurmans (2008); Guo (2010b); Hoi *et al.* (2006b, 2009)). Cai *et al.* (2013) extend sequential mode AL to BMAL by simulating the sequential mode AL behavior to simultaneously choose a set of examples without re-training. They introduce a novel AL framework for regression called Expected Model Change Maximization (EMCM), which queries the examples maximizing the model change once added to the training data. Käding *et al.* (2016) proposed a new generalization of the EMOC principle for deep architectures. Their algorithm actively selected relevant batches of unlabeled examples for image annotation. Active learning and deep model training were treated as two independent problems. A deep model was first learned using a conventional loss function. Batches of unlabeled examples that lead to a significant model output change,

based on the EMOC scores, were selected for annotation. Although Käding *et al.* (2016) presented easy-to-implement approximations that yielded efficient techniques for active selection, the deep model employed was not used to its complete ability. The merit of a deep model lies in its ability to learn rich sets of features for a given task; this property was not effectively leveraged.

7.1.3 Deep Active Learning for Regression

Even though deep learning and active learning for regression have been studied individually, research on combining the two is unexplored. In this paper, we exploit the feature learning capabilities of deep networks and propose a novel framework to address the problem of active learning for regression. We use Expected Model Output Change (EMOC) as the active selection criterion and integrate it within the objective function used to train the deep model. Since the active learning criterion is embedded within the loss function, the network gets specifically trained for the task of active learning and can potentially depict better performance than a network trained merely using a conventional L_2 loss. Furthermore, our technique considers changes in all the parameterized layers of the deep network and implicitly combines them into a single criterion for active sample selection. This is in contrast to existing methods that only make use of the current output of a deep neural network for querying unlabeled samples. We now describe our framework.

7.2 Proposed Framework

In our active learning setting, we are given labeled and unlabeled sets of samples (images) for training. In addition, there is also a test set of images for evaluation

purposes. In each iteration of our algorithm, a batch of unlabeled samples is selected and given to the human oracle for annotation (labeling). These labeled samples are removed from the unlabeled set and appended to the labeled set. An active learning model is iteratively trained using the updated labeled and unlabeled datasets. The trained model is evaluated using the test set. This procedure is repeated until we run out of budget to get labeled data from the oracle. The challenge in active learning is to identify the most informative set of unlabeled samples to be labeled by the oracle.

The core idea of this research is to leverage the feature learning capabilities of deep neural network models to identify the most informative unlabeled samples for active learning. We attempt to integrate an active sample selection criterion in the objective function and train the network to minimize this objective. The features learned by the network are then specially tailored to the active learning task. This enables the model to better identify the samples that can augment maximal information to the model. We use the EMOC criterion to quantify the utility of an unlabeled sample in our active learning framework. We achieve this by adding an EMOC based loss term to the conventional regression objective and train the network to optimize this joint objective function.

Formally, let $g(x;\phi)$ be the output of a neural network where parameters ϕ are the parameters of the network and x is the input image. In this work, we focus on layered deep models, $g(x_i;\phi) = g_l(\cdots(g_2(g_1(x_i;\phi_1);\phi_2)\cdots);\phi_l)$. Here, $\phi = (\phi_1, \cdots, \phi_l)$ denotes the parameters of the deep model and l is the total number of layers in the deep model. The training data consists of both labeled and unlabeled samples (images). Let the set of labeled samples be represented as $X^L = \{x_1, x_2, \ldots, x_{n_l}\}$. The corresponding labels for X^L are denoted by $Y^L = \{y_1, y_2, \ldots, y_{n_l}\}$ represent-

ing continuous real values; $y_i \in \mathbb{R}$. Let the set of unlabeled samples be $X^U = \{x_{n_l+1}, x_{n_l+2}, \cdots, x_{n_l+n_u}\}$. Let $X = X^L \cup X^U$ denote the union of the disjoint subsets X^L and X^U and $n = n_l + n_u$. The goal of active learning is to select a batch B containing k unlabeled samples for manual annotation such that the modified learner trained on labeled set $X^L \cup B$ and unlabeled set $X^U \backslash B$ has maximum generalization capability. We now formulate a novel loss function to train the deep CNN for active learning.

7.2.1 Loss on Labeled Data

We use the conventional L_2 loss for regression on the labeled data. Consider a subset of labeled samples $X^l = \{x_1, x_2, \cdots, x_{n'_l}\}$ and their corresponding labels $Y^l = \{y_1, y_2, \cdots, y_{n'_l}\}$. Let $\hat{Y}^l = [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{n'_l}] = [g(x_1; \phi), g(x_2; \phi), \ldots, g(x_{n'_l}; \phi)]$, be the predictions of the deep neural network on the subset of labeled data. The prediction loss is given by,

$$\mathcal{L}(\phi; X^l, Y^l) = \frac{1}{n'_l} \sum_{i=1}^{n'_l} (y_i - \hat{y}_i)^2 \,. \tag{7.1}$$

Minimizing this loss iteratively over different subsets of the training data ensures that the trained model makes predictions that are consistent with the training data. During the process of active learning the labeled dataset is repeatedly augmented with newly obtained labeled data taken from the unlabeled dataset. The model $g(.; \phi)$, is in turn updated by retraining with the updated labeled set.

7.2.2 Principle of Expected Model Output Change (EMOC)

The EMOC criterion provides a principled way to quantify the importance of a sample by measuring the difference of model outputs when trained with and without

a particular data sample:

$$\triangle g(x') = \mathbb{E}_{y'|x'}\mathbb{E}_x \big|\big| g(x;\phi') - g(x;\phi) \big|\big|_1. \tag{7.2}$$

In Equation (7.2), $||g(x;\phi')-g(x;\phi)||_1$ computes the L_1 norm of the difference between the outputs of the models. Here, ϕ' denotes the parameters of the model obtained by additionally training with unlabeled example x'. In order to estimate ϕ' we need to know the label of x'. We assume y' is the label for x'. In general, the first expectation operation is used to marginalize over y' in the above equation to get the expected model change. The expectation $\mathbb{E}_x$ is estimated by computing the empirical mean across the dataset and the expectation $\mathbb{E}_{y'|x'}$ is based on the output of the updated model $g(.;\phi')$ for all possible values of y' given x'.

A direct implementation of the EMOC principle would require training a model from scratch for each example x' in the dataset, making it very computation intensive. Therefore, development of efficient techniques that approximate the change in model output $\triangle g(.)$, is required. Freytag *et al.* derived a closed form expression for $\triangle g(x')$ focusing on Gaussian process regression Freytag *et al.* (2014). Käding *et al.* used the stochastic gradient approximation with a single sample to estimate model parameter updates Käding *et al.* (2016). This approximation is given in Equation (7.3), where the gradient of the objective with respect to a candidate example (x', y') is used to estimate the model changes:

$$(\phi' - \phi) \approx \eta \nabla_\phi \mathcal{L}(\phi; (x', y')), \tag{7.3}$$

where, $\eta > 0$ is some constant. The difference $||g(x;\phi')$ - $g(x;\phi)||_1$ can be approximated using the first-order Taylor series approximation as:

$$\big|\big| g(x;\phi') - g(x;\phi) \big|\big|_1 \approx \big|\big| \nabla_\phi g(x;\phi)^\top (\phi' - \phi) \big|\big|_1. \tag{7.4}$$

We substitute Equation (7.3) in Equation (7.4) to get:

$$||g(x;\phi') - g(x;\phi)||_1 \approx \eta \big|\big|\nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi;(x',y'))\big|\big|_1. \tag{7.5}$$

Since marginalizing over all possible values for y' is impractical, Käding *et al.* proposed an approximation which considers only the most likely label $\bar{y}'$, (as the label for unlabeled sample x'), inferred by the model g Käding *et al.* (2016). It is therefore assumed that all examples in a given unlabeled set X' have label $\bar{y}'$. With this simplifying approximation the EMOC score for each unlabeled set X' is given by:

$$\triangle g(X') = \sum_{x' \in X'} \mathbb{E}_x \big|\big|\nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi;(x',\bar{y}'))\big|\big|_1. \tag{7.6}$$

7.2.3 Loss on Unlabeled Data

In our active learning framework, we leverage the principle of EMOC in developing the loss function to train the deep network. We train the regression network such that all the unlabeled samples have low EMOC scores with the trained model, i.e., no unlabeled sample can drastically affect the model parameters. We see the following benefits to incorporating the unlabeled data when training the network: (i) the CNN network is trained to extract features from both the labeled and unlabeled images, making it more robust compared to a network that only trains with labeled images, (ii) the EMOC loss from unlabeled data acts like a regularizer preventing over-fitting and improving the generalization capabilities of the network, (iii) since the network minimizes the EMOC loss, this helps in selecting the most relevant samples to form the batch B. This enables the network to converge to the optimal $\phi*$ with fewer labeled samples. On the basis of these arguments, we append a term in the loss function which enforces all the unlabeled samples to have low EMOC scores.

Let $X^u = \{x'_1, x'_2, \cdots, x'_{n'_u}\}$, denote a subset of unlabeled samples. We do not have

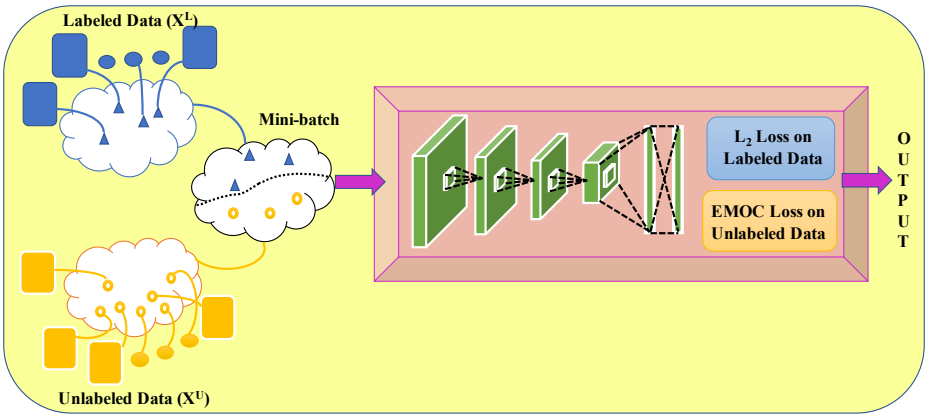


Figure 7.1: Illustration of the Proposed Deep Active Learning Framework. The deep model is trained with a combination of labeled and unlabeled mini-batches to minimize the L_2 loss over labeled data and EMOC loss over unlabeled data. A batch B of unlabeled points furnishing maximum EMOC scores is annotated and added to the labeled set. Best viewed in color.

the labels for X^u. Estimating the model change in Equation (7.2), by marginalizing over all possible labels for X^u is computation-intensive and impractical. We therefore approximate the labels of X^u to be the mean of the labels inferred by model g, along the lines of Käding *et al.* (2016). We forward propagate $X^u = \{x'_1, x'_2, \cdots, x'_{n'_u}\}$ through the network $g(.;\phi)$ and obtain the predictions $\hat{Y}^u = \{\hat{y}'_1, \hat{y}'_2, \cdots, \hat{y}'_{n'_u}\}$. Let $\bar{y}' = 1/n'_u \sum_{i=1}^{n'_u} \hat{y}'_i$ be the mean of $\hat{Y}^u$. The unlabeled samples and their approximated labels are: $\{(x'_1, \bar{y}'), (x'_2, \bar{y}'), \cdots, (x'_{n'_u}, \bar{y}')\}$. We formulate Equation (7.6) as a loss on unlabeled data. Therefore, in our framework, the loss over a subset of unlabeled data is given by:

$$\mathcal{U}(\phi; X^u) = \sum_{x' \in X^u} \mathbb{E}_x \big|\big| \nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')) \big|\big|_1. \tag{7.7}$$

Here, $\mathcal{L}(\phi; (x', \bar{y}')) = (\bar{y}' - \hat{y}')^2$, where x' is an unlabeled sample, $\bar{y}'$ is its approximate label, $\hat{y}'$ is the output of the network and $\mathbb{E}_x$ is the empirical mean over the dataset. Minimizing this loss ensures that the features are learned in such a way that all the unlabeled samples have low EMOC scores on the trained model.

7.2.4 Novel Joint Objective Function

The deep model is trained using both labeled and unlabeled data with the objective of minimizing the L_2 loss on labeled data and the EMOC loss on the unlabeled data. The joint loss ensures that the network can accurately predict the labels of the labeled training data while the unlabeled samples have minimal effect on the trained model parameters; i.e., the model depicts good performance on the unlabeled data. Over successive iterations, the positive effects of this joint training with labeled and unlabeled data get enhanced. Our novel joint objective function over a batch of labeled and unlabeled data is given by:

$$\mathcal{J}(\phi, X^l, Y^l, X^u) = \mathcal{L}(\phi; X^l, Y^l) + \lambda \mathcal{U}(\phi; X^u). \tag{7.8}$$

Here, $\lambda \geq 0$ controls the relative importance of the two terms. The objective function in Equation (7.8) is minimized over multiple batches of labeled and unlabeled data using mini-batch gradient descent. In order to train our network, we compute $\nabla_\phi \mathcal{J}$ and use back-propagation to update the network parameters (the next section gives the expression for the gradient $\nabla_\phi \mathcal{J}$). Once the network is trained, the unlabeled examples with the largest EMOC scores are selected to form the batch B. These samples are annotated by a human expert and the resulting labeled batch is appended to the labeled set X^L. Since the network is trained to minimize the EMOC score of the unlabeled samples, the unlabeled samples furnishing the highest EMOC scores after model training are the most informative data points with respect to the current model. They are hence queried for labels.

7.2.5 Gradient of Objective Function

We provide a high level overview of the gradient computation of the objective function in this section. Please refer to the supplemental file for the complete derivation.

The gradient of the joint objective function for deep active regression is given by:

$$\nabla_\phi \mathcal{J}(\phi, X^l, Y^l, X^u) = \nabla_\phi \mathcal{L}(\phi; X^l) + \lambda \nabla_\phi \mathcal{U}(\phi; X^u), \tag{7.9}$$

where the gradients for individual layers are:

$$\frac{\partial \mathcal{L}(\phi; X^l)}{\partial \phi_j} = \frac{-2}{n'_l} \sum_{i=1}^{n'_l} \left[(y_i - \hat{y}_i) \frac{\partial g(x_i; \phi)}{\partial \phi_j} \right], \ \forall j \in \{1, 2, \cdots, l\}, \tag{7.10}$$

and

$$\begin{aligned} \frac{\partial \mathcal{U}(\phi; X^u)}{\partial \phi_j} &= \sum_{x' \in X^u} \mathbb{E}_x \frac{\partial}{\partial \phi_j} \left|\left| \nabla_\phi g(x; \phi)^\top \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')) \right|\right|_1 \\ &= \sum_{x' \in X^u} \mathbb{E}_x \ (Q_j), \qquad \forall j \in \{1, 2, \cdots, l\}. \end{aligned} \tag{7.11}$$

In equation (A.31), Q_j stands for,

$$Q_j = \begin{cases} +(Q_{j_1} + Q_{j_2}) & \text{if} \quad \nabla_\phi g(x; \phi)^\top \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')) \geq 0 \\ -(Q_{j_1} + Q_{j_2}) & \text{if} \quad \nabla_\phi g(x; \phi)^\top \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')) < 0, \end{cases} \tag{7.12}$$

with

$$Q_{j_1} = \frac{\partial}{\partial \phi_j} \left(\nabla_\phi g(x; \phi)^\top \right) \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')), \tag{7.13}$$

and

$$Q_{j_2} = \nabla_\phi g(x; \phi)^\top \frac{\partial}{\partial \phi_j} \left(\nabla_\phi \mathcal{L}(\phi; (x', \bar{y}') \right). \tag{7.14}$$

We compute $Q_{j_1} + Q_{j_2}$ for a fixed $x' \in X^u$ and for all $x \in X^l$. Then we compute the expected value $\mathbb{E}_x(Q_j)$. We do this for every $x' \in X^u$ and then compute

Algorithm 4 The proposed Deep Active Paradigm for Regression

Input: The labeled set X^L, labels Y^L, unlabeled set X^U, weight parameter λ, batch size k, maximum number of iterations T

1: **for** $t = 1, 2, \ldots T$ **do**

2: Compute the derivative of the joint objective function in Equation (7.8)

3: Train the deep model to obtain the network weights

4: Compute the EMOC score of each unlabeled sample, using Equation (**??**)

5: Select a batch B containing k unlabeled samples from X_u furnishing the highest EMOC scores

6: Update $X^L \rightarrow X^L \cup B$; $X^U \rightarrow X^U \backslash B$

$\sum_{x' \in X^u} \mathbb{E}_x(Q_j),\ \forall j \in \{1, 2, \ldots, l\}$ to get $\nabla_\phi \mathcal{U}(\phi; X^u)$.

Figure (7.1) shows a graphical illustration of the proposed framework. We present the network with a mini-batch of n' data points consisting of n'_l labeled points and n'_u unlabeled points; $n' = n'_l + n'_u, (n'_l \leq n_l, n'_u \leq n_u)$. The L_2 loss is computed over the labeled data in the mini-batch and the EMOC loss is computed over the unlabeled data in the mini-batch. The negative gradient of the joint objective function with respect to the mini-batch is back-propagated to train the CNN. The weight parameter λ was selected to be 1 giving equal weightage to both the terms. When the network has seen all the data points in the training set (both labeled and unlabeled), we consider it as one epoch. We repeat the training procedure over multiple epochs until convergence and consider this one training iteration t of the active learning algorithm. At the end of every iteration t, we sample the most informative batch of unlabeled data samples (samples furnishing highest EMOC score from Equation (7.6)) to form B. We obtain the labels for B using an oracle and update the labeled and unlabeled

datasets as discussed earlier. We iterate until we run out of unlabeled data points to be labeled or run out of budget to get them labeled. For implementation purposes, we fix the maximum number of iterations as T. The pseudo-code of the proposed algorithm is given in Algorithm 4.

We note that an outlier in the dataset also furnishes a high EMOC score. Therefore, a significant change to the current model output does not always lead to better generalization performance. However, when the model has been altered by an outlier, the joint training objective along with the EMOC sampling criterion selects an informative set of examples in the next iteration that instantly alleviates the adverse effect of the outlier. In general, the number of outliers is low compared to the number of samples in the training set. Hence, it is reasonable to assume that the proposed framework will result in good generalization performance when salient examples are added to the labeled set over time.

7.3 Experiments and Results

7.3.1 Implementation Details

Figure (7.2) illustrates the network architecture of the CNN used for deep active regression. As seen in Figure (7.2), the size of the input image in the input layer of the CNN is 128×128 pixels. The convolution layer of our CNN model performs convolution operations with a kernel size of 3×3 pixels to acquire feature maps of the input information. The dimension of the first convolution layer is $128 \times 128 \times 32$ which denotes a feature size of 128×128 pixels and 32 different convolution kernels. All convolution layers are connected to RELU activation functions and max-pooling layers.

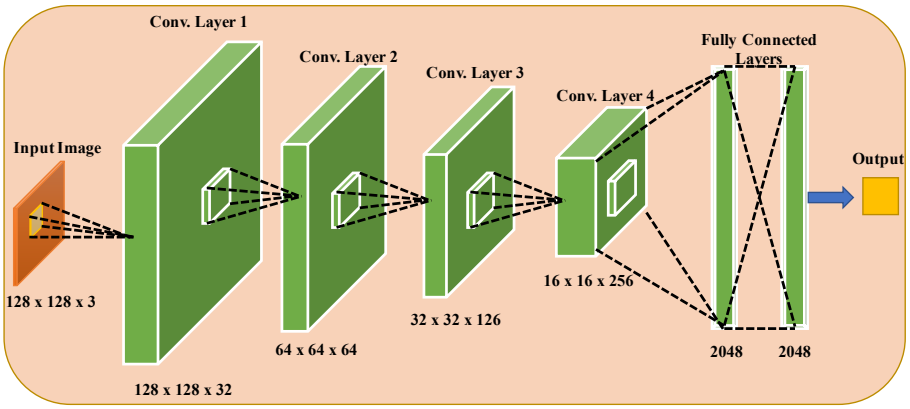


Figure 7.2: CNN Architecture for Deep Active Regression

The dimensions of the second, third and fourth convolution layers are $64 \times 64 \times 64$, $32 \times 32 \times 128$ and $16 \times 16 \times 256$ respectively. The dimensions of each fully connected layer is 2048. The activation function of the output layer is a linear function so as to obtain a continuous value output. The network is trained by minimizing the joint loss function given in Equation (7.8) using mini-batch gradient descent with an initial learning rate of 0.01. The implementations were performed in Matlab R2017b on a machine running a NVIDIA 1080Ti GPU with 11 GB memory.

7.3.2 Datasets and Experimental Setup

We used five benchmark regression datasets to evaluate our deep active framework; (1) *Synthetic hand-written digit dataset* (Zhu *et al.* (2003)), (2) *Rotated MNIST Digits* (Laptev *et al.* (2016)), (3) *WIKI Age Estimation Dataset* (Rothe *et al.* (2015)), (4) *BIWI Kinect Dataset* (Baltrušaitis *et al.* (2012)) and the (5) *QMUL Multiview Face Dataset* (Sherrah and Gong (2001)). These datasets represent different application domains (head pose, age and handwritten digit recognition) and have been widely used for testing regression models.

Our objective was to test the performance of the proposed active sampling frame-

work for deep learning and not to outperform the best accuracy results on these datasets; so, we did not follow the precise train/test splits given for these datasets. We split the dataset into three disjoint parts to construct the initial labeled set X^L, unlabeled set X^U and the test set $\mathcal{T}$. Each algorithm (baseline and proposed) selected k instances from the unlabeled pool to be labeled in each iteration. After each iteration, the selected samples were removed from the unlabeled set, appended to the training set and the performance was evaluated on the test set. The goal was to study the improvement in performance on the test set with increasing sizes of the training set. The experiments were run for 15 iterations. The dataset details are summarized in Table 1.

Dataset Name	**Labeled** (X^L)	**Unlabeled** (X^U)	**Test Set** ($\mathcal{T}$)	**Batch Size** (k)
Synthetic Handwritten Digits	500	4500	1000	200
WiKI Age Estimation	20000	30000	10000	400
MNIST Rotation	15000	25000	5000	400
BIWI Kinect	4000	6000	4000	400
QMUL Multiview	800	4200	1000	200

Table 7.1: Dataset Details

7.3.3 Comparison Baselines and Evaluation Metrics

To study the performance of our proposed framework, we compared our method against four state-of-the-art regression-based active learning algorithms:

1. **Käding *et al.* (2016)**: In this method a CNN, described in Section 7.3.1, is trained using the L_2 loss function given in Equation (7.1). Unlabeled samples

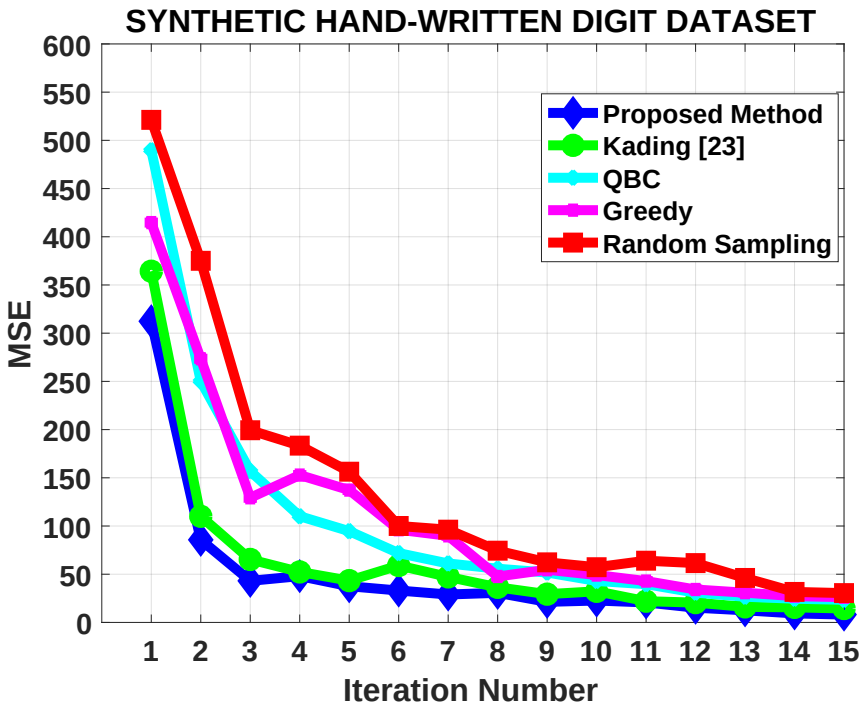


Figure 7.3: MSE Vs Iteration Number: Synthetic Handwritten Digits

with the largest EMOC are selected to form a batch. Note that, this is a two-step process, where a CNN is first trained using a conventional loss function and the EMOC criterion is then applied for active sampling. In contrast, our framework integrates the EMOC criterion in the loss function to train the network.

2. **Greedy**: This model selects unlabeled examples having the largest minimum distance from labeled data Yu and Kim (2010).

3. **Query-by-Committee (QBC)**: This model selects data points that have the largest variance among the committee's predictions Burbidge *et al.* (2007).

4. **Random Sampling**: This method selects a batch of samples at random from the unlabeled pool.

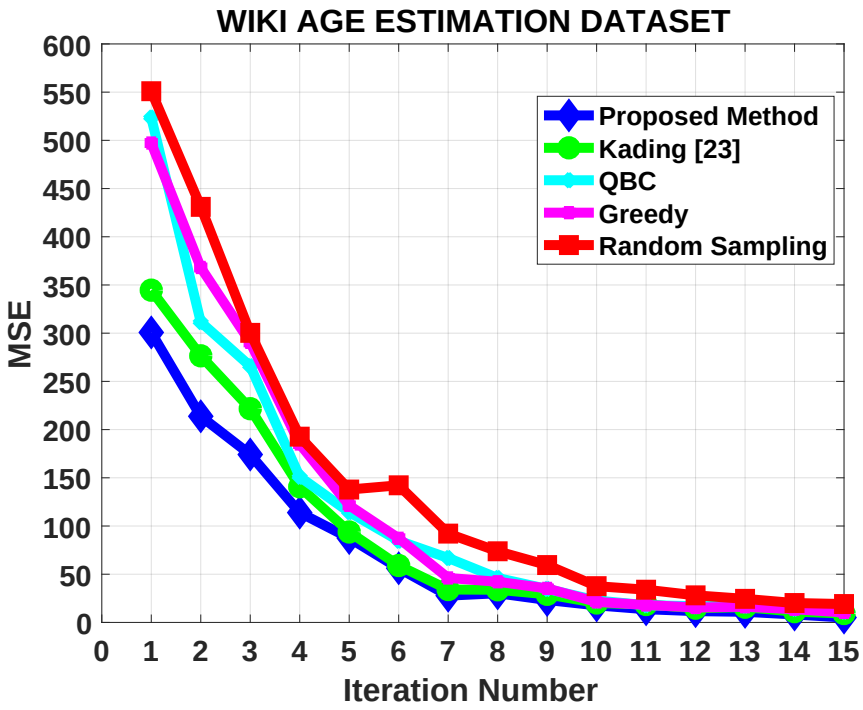


Figure 7.4: MSE Vs Iteration Number: WIKI Age Estimation

The *QBC* and *Greedy* active learning strategies were not proposed in the context of deep learning. However, for fair comparison, we trained the CNN described in Section 7.3.1 using the standard L_2 loss function and then applied the active selection criterion.

For evaluation, we used two popular error-based metrics, Mean Squared Error (*MSE*) and Mean Absolute Error (*MAE*), to study the performance of each method on the test set:

$$MSE = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}(y_i - g(x_i))^2 \tag{7.15}$$

$$MAE = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}|y_i - g(x_i)| \tag{7.16}$$

Here, $|\mathcal{T}|$ denotes the size of the test set; y_i and $g(x_i)$ are the ground truth and the

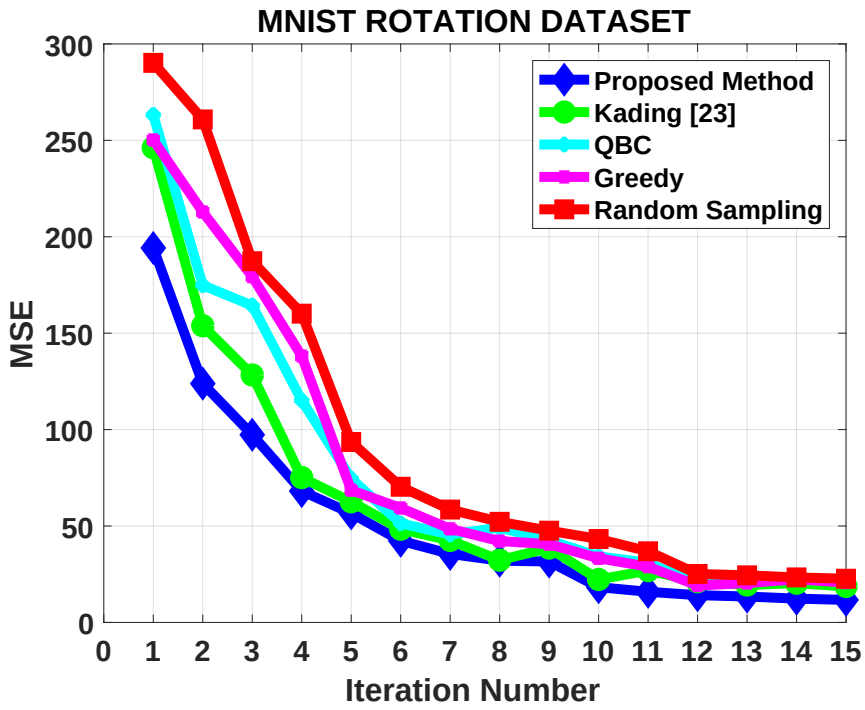


Figure 7.5: MSE Vs Iteration Number: MNIST Rotation

predictions of test sample x_i.

7.3.4 Active Learning Performance

The performance of the five active learning algorithms on the benchmark datasets are presented in Figures (7.3, 7.4, 7.5, 7.6 and 7.7). The x-axis represents the total number of iterations and the y-axis denotes the *MSE* values. In general, we see that, the *MSE* values decrease when the number of training points increases for all five algorithms. This is in accordance with the insight that the performance of the model increases with increase in labeled data. Our proposed deep active framework consistently depicts the best performance across all datasets; at any given iteration number, it has the least error among all the methods. This shows that the proposed framework can appropriately identify the most informative unlabeled samples for manual annotation and can attain a given performance level with the least hu-

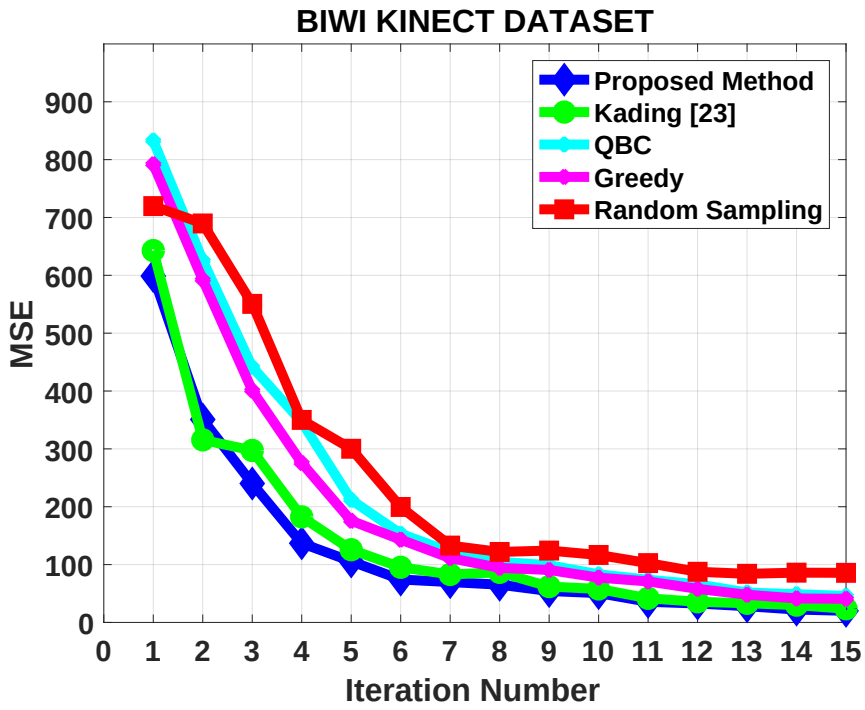


Figure 7.6: MSE Vs Iteration Number: BIWI Kinect

man effort. The performance of the Käding *et al.* (2016) baseline is better than the other three baselines, but is not as good as our method. This corroborates the fact that training a deep network to minimize a joint loss function containing the EMOC criterion depicts better performance than the two-step process of training a network to minimize the L_2 loss and then selecting samples based on EMOC. The *QBC* and *Greedy* algorithms outperform *Random Sampling*, but perform poorly compared to the Käding *et al.* (2016) baseline.

The *MAE* active learning curves for all the five datasets depict similar trends as the *MSE* curves. For the sake of brevity, we report the label complexity values in Table 7.2 (corresponding to $MAE = 9$). Each entry in the table denotes the number of unlabeled samples that had to be annotated to achieve an *MAE* value of 9. The results follow a similar pattern as in Figure **??**; the proposed method requires the

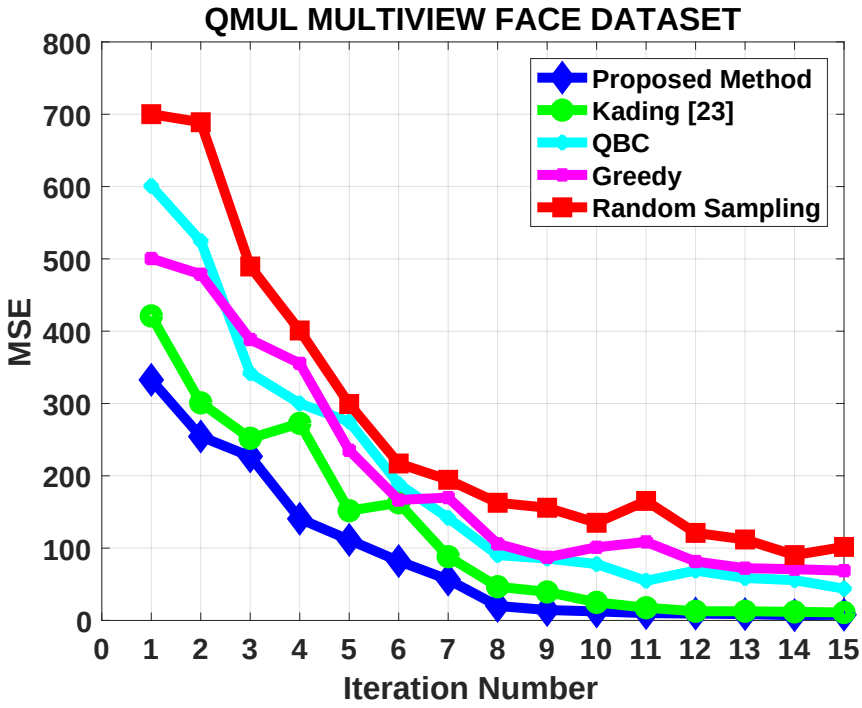


Figure 7.7: MSE Vs Iteration Number: QMUL Multiface

least amount of labeled samples (and consequently, the least human effort) to attain the given level of performance, for all the datasets. For the QMUL dataset, *Random Sampling* does not attain an *MAE* of 9 even after 15 iterations. The results unanimously lead to the conclusion that the proposed method consistently depicts the best performance over all the baseline algorithms, across all datasets.

7.3.5 Study of the Active Sampling Criterion

In order to further evaluate the active sampling criterion employed in our framework, we performed the following two experiments. We conducted these experiments on the MNIST Rotation dataset.

Experiment 1

Dataset Name	Proposed Method	Käding [23]	Greedy	QBC	Random Sampling
Synthetic Handwritten Digits	400	600	1000	1000	1200
WiKI Age Estimation	1200	1600	2000	2400	2800
MNIST Rotation	1200	1600	2400	2000	2800
BIWI Kinect	2000	2400	4000	3200	5200
QMUL Multiview	1000	1400	1600	2000	-

Table 7.2: Label Complexity for $MAE = 9$. The proposed framework requires the least amount of labeled data to reach a given performance level ($MAE = 9$) compared to all the baseline methods. For the QMUL dataset, *Random Sampling* does not attain an MAE of 9 even after 15 iterations.

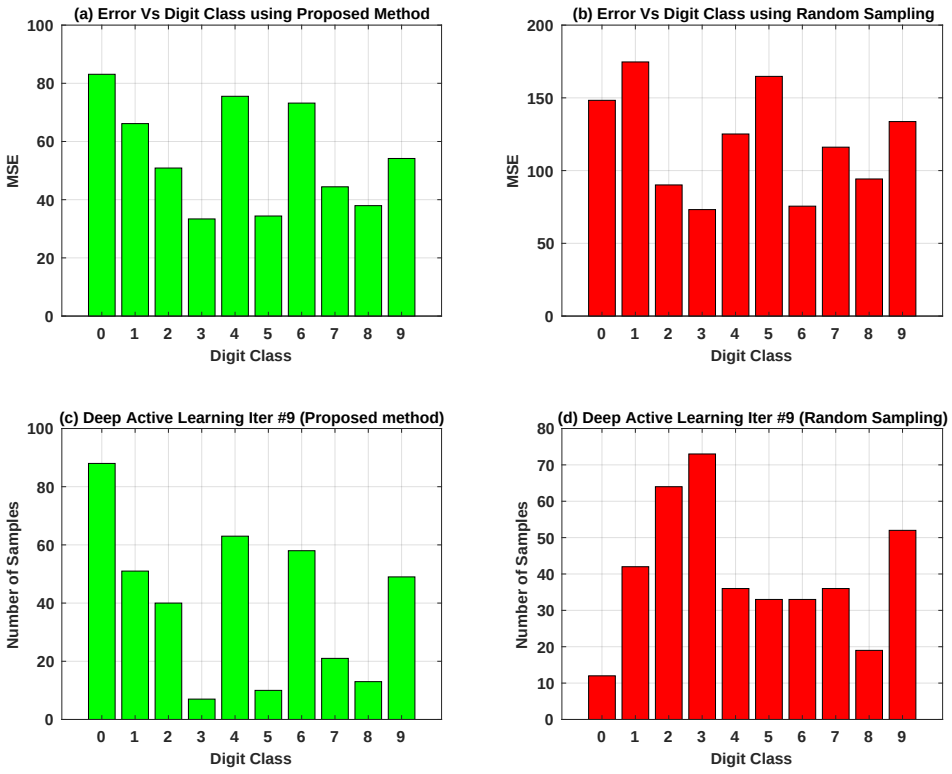


Figure 7.8: Results after Iteration Number 9. (a) MSE Vs Digit Class using Proposed method, (b) MSE Vs Digit Class using Random sampling, (c) Number of samples of each digit (0 - 9) selected using proposed method and (d) Number of samples of each digit (0 - 9) selected using Random sampling. Best viewed in color.

We selected 200 samples of each digit $(0-9)$ at random from the test set $(200 \times 10 = 2000$ samples). Figure (7.8(a)) shows the performance of the proposed model (after iteration number 9) per digit class on the selected 2000 samples. The x-axis corresponds to the digit class and y-axis shows the *MSE*. We perform similar experiments using the *Random Sampling* method. The performance of *Random Sampling* per digit class after iteration number 9 is shown in Figure 7.8(b). We then plot the number of samples of each digit picked to form batch B after iteration number 9 using the proposed method and *Random Sampling*. The results are shown in Figure 7.8(c) and 7.8(d) respectively.

Observations: From Figure 7.8(a), we see that, the top four digits furnishing the maximum errors are digits 0,4,6 and 1 when using the proposed model. Similarly from Figure 7.8(b), we observe that the top four digits furnishing the maximum error are 1,5,0 and 9 when using *Random Sampling*. From Figure 7.8(c), we observe that, 65% of the 400 samples selected to form batch B by the proposed method, belonged to digits 0,4,6 and 1 (the digits furnishing the maximum error using the model after iteration number 9). This shows that our proposed model intelligently selects samples to augment the training set, which can maximally reduce the generalization error. On the other hand, when using *Random Sampling* (Figure 7.8(d)), we notice that only 34.75% of the 400 samples selected to form batch B belonged to the four classes furnishing maximum error. This shows that there is no correlation between the number of samples selected for a digit to its corresponding error when using *Random Sampling* accounting for its poor performance.

Experiment 2

In this experiment, we further look into the rotation angle of the four digits furnishing maximal errors from the previous experiment. Figure 7.9(a) shows the performance

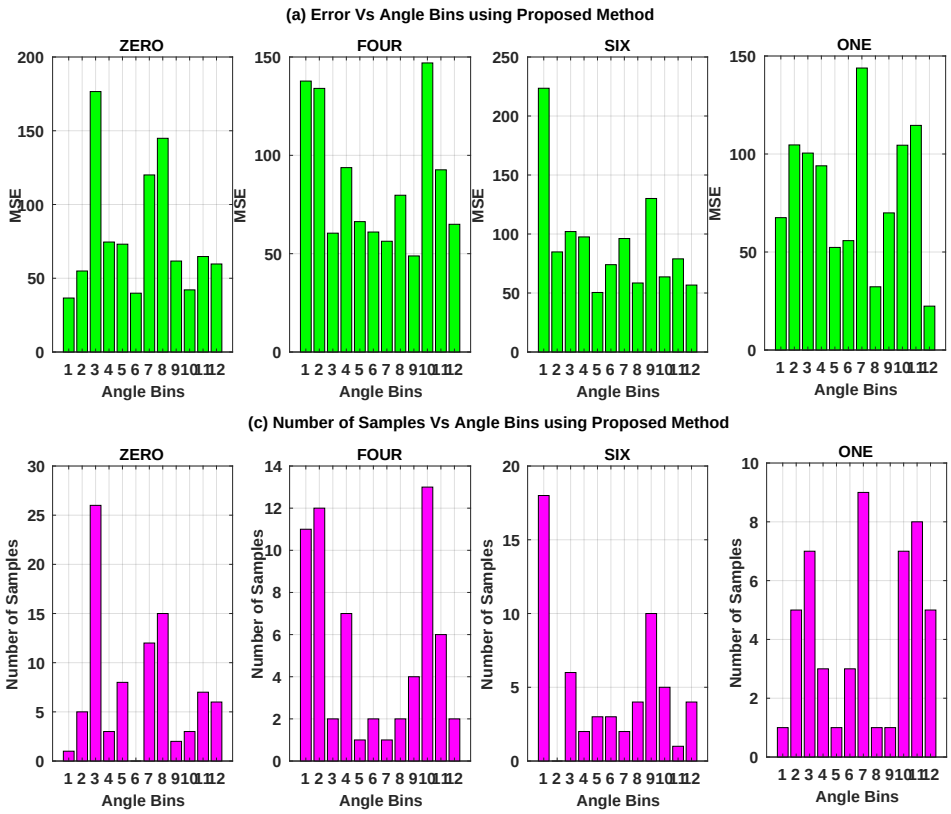


Figure 7.9: Results after Iteration Number 9. (a) MSE Vs Rotation Angle Bins - Proposed method,(c) Number of samples selected in each anglular bin - Proposed method. Best viewed in color.

of the proposed model per angular bin after iteration 9 (for the four digits 0,4,6 and 1 furnishing the maximal errors). We split the range of the predicted angles into 12 different bins ($\textbf{Bin 1} : (-60°to-50°), \textbf{Bin 2} : (-49°to-40°), \ldots, \textbf{Bin 12} : (+50°to+60°)$). In Figure 7.9(c), we plot the number of samples picked after iteration number 9 in each angular bin. Figures 7.10(b) and 7.10(d) show similar plots using the *Random Sampling* method (for the four digits 1,5,0 and 9 furnishing the maximal errors).

Observations: From Figures 7.9(a) and 7.9(c), when using the proposed method, we see that there is a direct correlation between the angular bins showing high error and the number of samples chosen in those angular bins. We see no such relations when

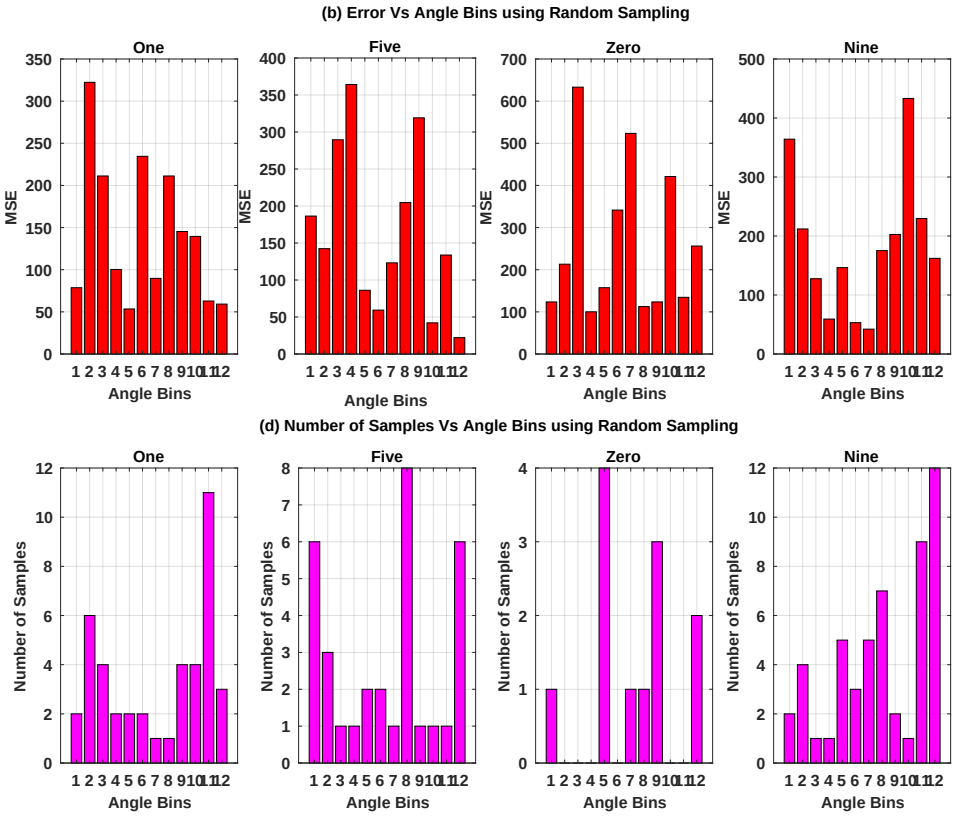


Figure 7.10: Results after Iteration Number 9. (b) MSE Vs Rotation Angle Bins - Random Sampling and (d) Number of samples selected in each angular bin - Random sampling. Best viewed in color.

using *Random Sampling* in Figures 7.10(b) and 7.10(d). This further corroborates the usefulness of the active sampling criterion used to train the deep CNN in our framework.

7.3.6 Visual Illustration of the Selected Samples

Figure 7.11 and 7.12 presents a visual illustration of the top 15 unlabeled samples, from each digit class, selected for manual annotation (furnishing the maximum EMOC scores) by our method and *Random Sampling* after iteration number 12. It is evident that the proposed method captures a wide range of informative samples across all digits while *Random Sampling* captures much less variation. Thus, the proposed

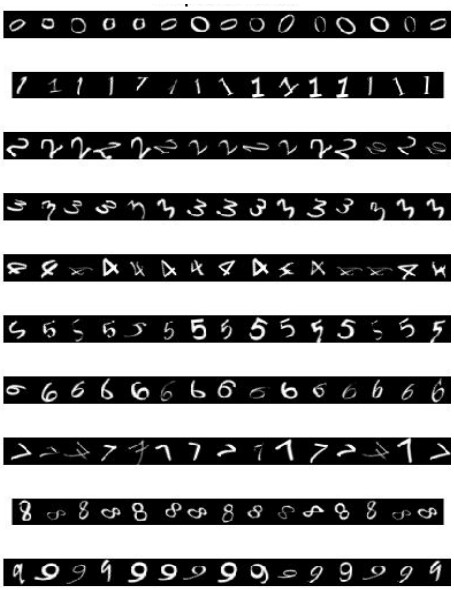


Figure 7.11: Visual Comparison of top 15 EMOC scores furnished by digits 0-9 using Proposed method after iteration number 12.

method augments useful knowledge to the model, which accounts for its improved performance.

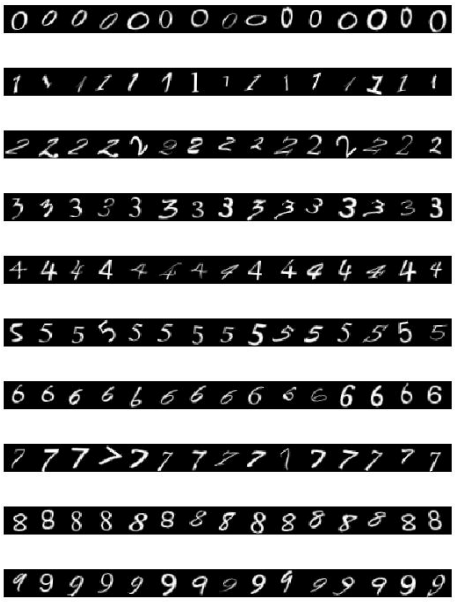


Figure 7.12: Visual Comparison of top 15 EMOC scores furnished by digits 0-9 using Random Sampling after iteration number 12.

7.4 Conclusions

In this paper, we proposed a novel deep active learning framework for regression applications. We used the Expected Model Output Change (EMOC) as the active selection criterion and integrated it within the objective function used to train the deep CNN. The resulting model optimized this novel objective function and learned from salient examples that caused the maximum change to the current model. Extensive empirical results on benchmark regression datasets demonstrated the effectiveness of the proposed framework in selecting the most informative samples for learning and annotation. Our in-depth analysis of the proposed active sampling criterion further corroborated the efficacy of our algorithm. To the best of our knowledge, this is the first research effort to leverage the feature learning capabilities of deep CNNs to develop a novel active learning algorithm for regression applications.

As part of future work, we plan to conduct extensive large-scale experiments to study the performance of our framework. We also intend to study the performance of

our deep active learning framework on multimedia applications involving other types of label spaces, such as multi-label and fuzzy-label classification.

Chapter 8

FUTURE DIRECTIONS

This chapter proposes some directions for future research in multimodal emotion recognition and deep active learning for computer vision.

8.0.1 Multimodal Emotion Recognition

In the current era, human - computer interaction (HCI) interface undoubtedly plays an important role in our daily life. Automated analysis and recognition of human emotion has attracted increasing attention from the researchers in multidisciplinary research fields. Hand-engineering task-specific features is often difficult and time consuming. This difficulty is more pronounced with multimodal data as the features have to relate multiple data sources. In this dissertation, we showed how deep learning can be applied to this challenging task for discovering multimodal features. Although a number of promising studies have been proposed and successfully applied to emotion recognition, there are some important issues, outlined in the following that can become potential future directions of research.

- Traditional emotion recognition systems work using laboratory controlled data. A new direction to explore the performance of emotion recognition methods that work in real-world conditions will be more useful. A comprehensive and accessible database covering various social signals such as laughs, smiles, depression, agreement, disagreement, etc. is desirable to help better understand different affective behaviors.

- Designing better data fusion methods considering various model properties,

temporal expression and asynchrony would lead to improved performance of multimodal emotion recognition systems.

- Another possible direction will be to explore the expression styles from different users. Here, importance is given not only to the intensity of expression but also to the manner in which the emotion is expressed along with personality trait. This is essential for effective emotion recognition.

- It is insufficient to build a general emotion recognition system that performs equally well for every user. In contrast, it is desirable to use personal computers/devices to build person-centric emotion recognition systems. Developing model adaptation methods using small-sized adaptation datasets for person-centric emotion recognition should be considered in future.

- Existing emotion recognition methods explore variations in spontaneous emotion expressions, including head pose variations, speaking-influenced facial expression and partial facial occlusion in facial emotion recognition. Further investigations on these effects are essential for achieving robust emotion recognition for real-life applications.

8.0.2 Deep Active Learning Models for all Label Spaces

The field of deep active learning combines ideas of deep learning and active learning. Even though both deep learning and active learning have been extensively studied, research on combining the two is still in inception. These are either treated as two independent problems like in existing literature or as one problem as proposed in this dissertation. This section enumerates some important issues that can become potential future directions of research.

- As mentioned in Wang and Hua (2011), we need to give importance to the human annotation behavior. Active learning is an interactive approach that involves two parts - human and computer. But most of the existing research efforts have been dedicated to computation algorithms, such as sample selection strategies and learning models, whereas the human part receives relatively less attention. However, human also plays a very important role in active learning. As part of future work, the human annotation behavior could be analyzed. This analysis will help benefit active learning-based multimedia annotation and retrieval.

- A study of the performance of the network trained to optimize other loss functions specific to active learning will be an interesting direction of research. Some active learning algorithms include a diversity based criterion (besides uncertainty), to ensure that the samples selected for annotation also have a high diversity among them (Chakraborty *et al.* (2015b)); representativeness based algorithms explored in Shen *et al.* (2004), enforce the selected samples to be good representatives of the unselected unlabeled samples. These criteria could be integrated in the loss function and their effects on the results be studied.

- The proposed algorithms could be modified to work on video datasets, where the temporal structure provides useful information that are less obvious in static images and thereby help improve the performance of the deep active models.

- Another direction for future work will be to extensively validate the performance of the framework on other computer vision applications such as image segmentation, image search and retrieval and object detection among others.

- Identifying an appropriate stopping criterion for active learning is still an open

problem and is a promising direction for future research.

The following chapter provides a summary of the contributions made in this dissertation.

Chapter 9

SUMMARY

This chapter enumerates the summary of all the contributions made in this dissertation. It also lists the conference submissions that are published in peer reviewed conference proceedings along with submissions that are currently under review. The chapter finishes with a record of the poster presentations made at different workshops during my PhD tenure.

9.0.1 Summary of Contributions

1. A new multimodal emotion database (emoFBVP) was created consisting of multimodal recordings of facial expressions, body gestures, vocal expressions and physiological signals of actors enacting various expressions of emotion. The affective computing community will greatly benefit from the large collection of modalities recorded.

2. The second contribution investigated is the use of deep learning architectures - Deep Belief Networks (DBNs) and Convolutional Deep Belief Networks (CDBNs) for multimodal emotion recognition. Four DBN models were proposed and experiments showed that they generated robust multimodal features for emotion recognition. The CDBN model proposed learned salient multimodal features of low intensity expressions of emotions.

3. The effect of transfer of emotion-rich features between source and target networks on classification accuracy and training time in a multimodal setting for vision based emotion recognition is studied. This is the first research effort to

study the transfer of emotion features layer-by-layer in a multimodal setting. The models proposed were able to successfully re-purpose the emotion rich features learned by the source model to train the target models and achieve shorter training times and performance boosts respectively. The results obtained are extremely useful in a practical setting.

4. A novel active learning framework to select the most informative unlabeled sample to train a Deep Belief Network (DBN) is proposed. A loss function specific to the task of active learning is introduced and the model is trained to minimize this loss. Extensive empirical studies on a wide variety of unimodal and multimodal vision datasets corroborate the potential of the proposed method for real-world image recognition applications.

5. The feature learning capabilities of deep neural networks is exploited and a novel framework to address the problem of multi-label active learning is proposed. An active sample selection criterion is integrated in the loss function used to train the deep networks. First, a framework without considering the correlation among the multiple labels is proposed using Convolutional Neural Networks (CNNs). Second, the correlations that exist among the multiple labels are modeled using Long Short Term memory (LSTM) cells. Extensive empirical studies on five benchmark multi-label datasets show that the proposed methods outperform state-of-the-art active learning techniques.

6. Ideas from deep learning and active learning are fused and a novel deep active learning paradigm for regression is proposed. The Expected Model Output Change (EMOC) is used as the active selection criterion and integrated with the objective function used to train the deep model. The resulting model optimizes this novel objective function and learns from salient examples that cause max-

imum change to the current model. Extensive empirical results on benchmark regression datasets demonstrate the effectiveness of the proposed paradigm in choosing the most informative samples for learning and annotation.

9.0.2 Conference Submissions

1. Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. Multimodal Emotion Recognition Using Deep Learning Architectures. *In IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

2. Hiranmayi Ranganathan, Shayok Chakraborty, and Sethuraman Panchanathan. Transfer of Multimodal Emotion Features in Deep Belief Networks. *In 50th Asilomar Conference on Signals, Systems and Computers, 2016 , pages 449 - 453. IEEE*, 2016.

3. Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Deep Active Learning for Image Classification. *In Proc. IEEE International Conference on Image Processing (ICIP)*, 2017.

4. Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Multi-label Deep Active Learning with Label Correlation. *In Proc. IEEE International Conference on Image Processing (ICIP)*, 2018. (Under Review)

5. Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. Deep Active Learning for Regression. *In Proc. of the ACM international conference on multimedia*, 2018. (Under Review)

9.0.3 Workshop Poster Presentations

1. Deep Architectures for Multimodal Emotion Recognition, Hiranmayi Ranganathan, Women in Computer Vision CVPR Workshop (WiCV), 2016.

2. Deep Active models for Image Classification, Hiranmayi Ranganathan, Women in Computer Vision CVPR Workshop (WiCV), 2017.

3. Deep Active models for Single label and Multi label Image Classification, Hiranmayi Ranganathan, Women in Machine Learning (WiML), 2017 co-located with NIPS 2017.

4. A Novel Deep Active Paradigm for Regression, Hiranmayi Ranganathan, Women in Computer Vision CVPR Workshop (WiCV), 2018 (Under Review).

BIBLIOGRAPHY

Anagnostopoulos, C.-N., T. Iliou and I. Giannoukos, "Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011", Artificial Intelligence Review **43**, 2, 155–177 (2015).

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

Azimi, J., A. Fern, X. Zhang-Fern, G. Borradaile and B. Heeringa, "Batch active learning via coordinated matching", arXiv preprint arXiv:1206.6458 (2012).

Baltrušaitis, T., P. Robinson and L.-P. Morency, "3d constrained local model for rigid and non-rigid facial tracking", in "Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on", pp. 2610–2617 (IEEE, 2012).

Belagiannis, V., S. Amin, M. Andriluka, B. Schiele, N. Navab and S. Ilic, "3d pictorial structures for multiple human pose estimation", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 1669–1676 (2014).

Bengio, Y., P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training among of deep networks", in "Advances in neural information processing systems", pp. 153–160 (2007).

Bengio, Y. *et al.*, "Learning deep architectures for ai", Foundations and trends® in Machine Learning **2**, 1, 1–127 (2009).

Boureau, Y.-l., Y. L. Cun *et al.*, "Sparse feature learning for deep belief networks", in "Advances in neural information processing systems", pp. 1185–1192 (2008).

Bourlard, H. and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition", Biological cybernetics **59**, 4-5, 291–294 (1988).

Brinker, K., "Incorporating diversity in active learning with support vector machines", in "International Conference on Machine Learning (ICML)", (2003a).

Brinker, K., "Incorporating diversity in active learning with support vector machines", in "Proceedings of the 20th international conference on machine learning (ICML-03)", pp. 59–66 (2003b).

Brueckner, R. and B. Schuller, "Likability classification–a not so deep neural network approach", in "Proceedings INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association", (2012).

Burbidge, R., J. J. Rowland and R. D. King, "Active learning for regression based on query by committee", in "International Conference on Intelligent Data Engineering and Automated Learning", pp. 209–218 (Springer, 2007).

Busso, C., Z. Deng, S. Yildirim, M. Bulut, C. M. Lee, A. Kazemzadeh, S. Lee, U. Neumann and S. Narayanan, "Analysis of emotion recognition using facial expressions, speech and multimodal information", in "Proceedings of the 6th international conference on Multimodal interfaces", pp. 205–211 (ACM, 2004).

Busso, C., S. Lee and S. S. Narayanan, "Using neutral speech models for emotional speech analysis", in "Eighth Annual Conference of the International Speech Communication Association", (2007).

Cai, W., Y. Zhang and J. Zhou, "Maximizing expected model change for active learning in regression", in "Data Mining (ICDM), 2013 IEEE 13th International Conference on", pp. 51–60 (IEEE, 2013).

Campigotto, P., A. Passerini and R. Battiti, "Active learning of pareto fronts", IEEE transactions on neural networks and learning systems **25**, 3, 506–519 (2014).

Cebron, N. and M. Berthold, "Active learning in parallel universes", in "ACM International Conference on Information and Knowledge Management (CIKM)", (2010).

Chakraborty, S., V. Balasubramanian and S. Panchanathan, "Generalized batch mode active learning for face-based biometric recognition", in "Pattern Recognition Journal", (2013).

Chakraborty, S., V. Balasubramanian and S. Panchanathan, "Adaptive batch mode active learning", IEEE transactions on neural networks and learning systems **26**, 8, 1747–1760 (2015a).

Chakraborty, S., V. Balasubramanian, Q. Sun, S. Panchanathan and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds", IEEE transactions on pattern analysis and machine intelligence **37**, 10, 1945–1958 (2015b).

Chattopadhyay, R., Z. Wang, W. Fan, I. Davidson, S. Panchanathan and J. Ye, "Batch mode active sampling based on marginal probability distribution matching", ACM Transactions on Knowledge Discovery from Data (TKDD) **7**, 3, 13 (2013).

Chua, T.-S., J. Tang, R. Hong, H. Li, Z. Luo and Y. Zheng, "Nus-wide: a real-world web image database from national university of singapore", in "Proceedings of the ACM international conference on image and video retrieval", p. 48 (ACM, 2009).

Clare, A. and R. King, "Knowledge discovery in multi-label phenotype data", Principles of data mining and knowledge discovery pp. 42–53 (2001).

Cohn, D. A., Z. Ghahramani and M. I. Jordan, "Active learning with statistical models", Journal of artificial intelligence research (1996).

Colah, *Understanding LSTM Networks*, URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` (2018 (accessed March 16, 2018)).

Dahl, G., M. Ranzato, A. Mohamed and G. Hinton, "Phone recognition with the mean-covariance restricted boltzmann machine", in "Advances in Neural Information Processing Systems (NIPS)", (2010).

Dahl, G. E., T. N. Sainath and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout", in "Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on", pp. 8609–8613 (IEEE, 2013).

Davidson, I. and W. Fan, "When efficient model averaging out-performs boosting and bagging", in "European Conference on Principles of Data Mining and Knowledge Discovery", pp. 478–486 (Springer, 2006).

de Fortuny, E. J. and D. Martens, "Active learning-based pedagogical rule extraction", IEEE transactions on neural networks and learning systems **26**, 11, 2664–2677 (2015).

Dhall, A., R. Goecke, J. Joshi, M. Wagner and T. Gedeon, "Emotion recognition in the wild challenge 2013", in "Proceedings of the 15th ACM on International conference on multimodal interaction", pp. 509–516 (ACM, 2013).

Donahue, J., Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition", in "International conference on machine learning", pp. 647–655 (2014).

Dosovitskiy, A., J. T. Springenberg and T. Brox, "Learning to generate chairs with convolutional neural networks", in "Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on", pp. 1538–1546 (IEEE, 2015).

Douglas-Cowie, E., R. Cowie and M. Schröder, "A new emotion database: considerations, sources and scope", in "ISCA tutorial and research workshop (ITRW) on speech and emotion", (2000).

Douglas-Cowie, E., R. Cowie, I. Sneddon, C. Cox, O. Lowry, M. Mcrorie, J.-C. Martin, L. Devillers, S. Abrilian, A. Batliner *et al.*, "The humaine database: addressing the collection and annotation of naturalistic and induced emotional data", in "International conference on affective computing and intelligent interaction", pp. 488–500 (Springer, 2007).

Duch, W., J. Biesiada, T. Winiarski, K. Grudzinski, K. Grabczewski *et al.*, "Feature selection based on information theory filters", in "In Proceedings of the International Conference on Neural Networks and Soft Computing (ICNNSC 2002), Advances in Soft Computing", (Citeseer, 2002).

Eigen, D., C. Puhrsch and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network", in "Advances in neural information processing systems", pp. 2366–2374 (2014).

El-Kaliouby, R. and P. Robinson, "Mind reading machines: Automated inference of cognitive mental states from video", in "IEEE International Conference on Systems, Man and Cybernetics", (2004).

Everingham, M., L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge", International journal of computer vision **88**, 2, 303–338 (2010).

Freund, Y., H. S. Seung, E. Shamir and N. Tishby, "Selective sampling using the query by committee algorithm", Machine learning **28**, 2-3, 133–168 (1997).

Freytag, A., E. Rodner and J. Denzler, "Selecting influential examples: Active learning with expected model output changes", in "European Conference on Computer Vision", pp. 562–577 (Springer, 2014).

Ghahramani, Z., "Unsupervised learning", in "Advanced lectures on machine learning", pp. 72–112 (Springer, 2004).

Girshick, R., "Fast r-cnn", arXiv preprint arXiv:1504.08083 (2015).

Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 580–587 (2014).

Gkioxari, G., B. Hariharan, R. Girshick and J. Malik, "R-cnns for pose estimation and action detection", arXiv preprint arXiv:1406.5212 (2014).

Gong, Y., Y. Jia, T. Leung, A. Toshev and S. Ioffe, "Deep convolutional ranking for multilabel image annotation", arXiv preprint arXiv:1312.4894 (2013).

Grimm, M., K. Kroschel and S. Narayanan, "The vera am mittag german audio-visual emotional speech database", in "Multimedia and Expo, 2008 IEEE International Conference on", pp. 865–868 (IEEE, 2008).

Guo, Y., "Active instance sampling via matrix partition", in "Advances of Neural Information Processing Systems (NIPS)", (2010a).

Guo, Y., "Active instance sampling via matrix partition", in "Advances in Neural Information Processing Systems", pp. 802–810 (2010b).

Guo, Y. and D. Schuurmans, "Discriminative batch mode active learning", in "Advances of Neural Information Processing Systems (NIPS)", (2007).

Guo, Y. and D. Schuurmans, "Discriminative batch mode active learning", in "Advances in neural information processing systems", pp. 593–600 (2008).

Healey, J. A. and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors", IEEE Transactions on intelligent transportation systems **6**, 2, 156–166 (2005).

Hinton, G. E., "Connectionist learning procedures", in "Machine Learning, Volume III", pp. 555–610 (Elsevier, 1990).

Hinton, G. E., "Training products of experts by minimizing contrastive divergence", Neural computation **14**, 8, 1771–1800 (2002).

Hinton, G. E., P. Dayan, B. J. Frey and R. M. Neal, "The" wake-sleep" algorithm for unsupervised neural networks", Science **268**, 5214, 1158–1161 (1995).

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

Hinton, G. E. and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", science **313**, 5786, 504–507 (2006).

Hochreiter, S. and J. Schmidhuber, "Lstm can solve hard long time lag problems", in "Advances in neural information processing systems", pp. 473–479 (1997).

Hoi, S., R. Jin, J. Zhu and M. Lyu, "Batch mode active learning and its application to medical image classification", in "International Conference on Machine Learning (ICML)", (2006a).

Hoi, S., R. Jin, J. Zhu and M. Lyu, "Semi-supervised SVM batch mode active learning for image retrieval", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2008).

Hoi, S. C., R. Jin and M. R. Lyu, "Large-scale text categorization by batch mode active learning", in "Proceedings of the 15th international conference on World Wide Web", pp. 633–642 (ACM, 2006b).

Hoi, S. C., R. Jin and M. R. Lyu, "Batch mode active learning with applications to text categorization and image retrieval", IEEE Transactions on knowledge and data engineering **21**, 9, 1233–1248 (2009).

Huang, S.-J., S. Chen and Z.-H. Zhou, "Multi-label active learning: Query type matters.", in "IJCAI", pp. 946–952 (2015).

Huang, Y., W. Wang, L. Wang and T. Tan, "Multi-task deep neural network for multi-label learning", in "Image Processing (ICIP), 2013 20th IEEE International Conference on", pp. 2897–2900 (IEEE, 2013).

Hung, C.-W. and H.-T. Lin, "Multi-label active learning with auxiliary learner", in "Asian conference on machine learning", pp. 315–332 (2011).

Jaderberg, M., K. Simonyan, A. Vedaldi and A. Zisserman, "Reading text in the wild with convolutional neural networks", International Journal of Computer Vision **116**, 1, 1–20 (2016).

Japkowicz, N., S. J. Hanson and M. A. Gluck, "Nonlinear autoassociation is not equivalent to pca", Neural computation **12**, 3, 531–545 (2000).

Käding, C., E. Rodner, A. Freytag and J. Denzler, "Active and continuous exploration with deep neural networks and expected model output changes", arXiv preprint arXiv:1612.06129 (2016).

Kahou, S. E., C. Pal, X. Bouthillier, P. Froumenty, Ç. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R. C. Ferrari *et al.*, "Combining modality specific deep neural networks for emotion recognition in video", in "Proceedings of the 15th ACM on International conference on multimodal interaction", pp. 543–550 (ACM, 2013).

Kanade, T., J. Cohn and Y. Tian, "Comprehensive database for facial expression analysis", in "IEEE International Conference on Automatic Face and Gesture Recognition", (2000a).

Kanade, T., J. F. Cohn and Y. Tian, "Comprehensive database for facial expression analysis", in "Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000. Proceedings.", pp. 46–53 (IEEE, 2000b).

Kim, Y., H. Lee and E. M. Provost, "Deep learning for robust feature generation in audiovisual emotion recognition", in "Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on", pp. 3687–3691 (IEEE, 2013).

Koelstra, S., C. Muhl, M. Soleymani, J.-S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt and I. Patras, "Deap: A database for emotion analysis; using physiological signals", IEEE Transactions on Affective Computing **3**, 1, 18–31 (2012).

Krizhevsky, A., "Learning multiple layers of features from tiny images", in "Technical Report", (2009).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in "Advances in neural information processing systems", pp. 1097–1105 (2012).

Krogh, A. and J. A. Hertz, "A simple weight decay can improve generalization", in "Advances in neural information processing systems", pp. 950–957 (1992).

K.Zhaoa, W.S.Chu and H.Zhang, "Deep region and multilabel learning for facial action unit detection", in "CVPR", (2016).

Laptev, D., N. Savinov, J. M. Buhmann and M. Pollefeys, "Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 289–297 (2016).

Larochelle, H., D. Erhan, A. Courville, J. Bergstra and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation", in "Proceedings of the 24th international conference on Machine learning", pp. 473–480 (ACM, 2007).

LeCun, Y., Y. Bengio and G. Hinton, "Deep learning", nature **521**, 7553, 436 (2015).

LeCun, Y., L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition", in "Proceedings of IEEE", (1998a).

LeCun, Y., L. Bottou, G. Orr and K. Muller, "Efficient backprop in neural networks: Tricks of the trade (orr, g. and müller, k., eds.)[j]", Lecture Notes in Computer Science **1524** (1998b).

Lee, H., C. Ekanadham and A. Y. Ng, "Sparse deep belief net model for visual area v2", in "Advances in neural information processing systems", pp. 873–880 (2008).

Lee, H., R. Grosse, R. Ranganath and A. Y. Ng, "Unsupervised learning of hierarchical representations with convolutional deep belief networks", Communications of the ACM **54**, 10, 95–103 (2011).

Lewis, D. D. and W. A. Gale, "A sequential algorithm for training text classifiers", in "Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval", pp. 3–12 (Springer-Verlag New York, Inc., 1994).

Li, S. and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network", in "Asian Conference on Computer Vision", pp. 332–347 (Springer, 2014).

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context", in "European conference on computer vision", pp. 740–755 (Springer, 2014).

Liu, Y., S. Zhou and Q. Chen, "Discriminative deep belief networks for visual data classification", in "Pattern Recognition", (2011).

Liu, Z., X. Li, P. Luo, C.-C. Loy and X. Tang, "Semantic image segmentation via deep parsing network", in "Computer Vision (ICCV), 2015 IEEE International Conference on", pp. 1377–1385 (IEEE, 2015).

Lu, Y., I. Cohen, X. S. Zhou and Q. Tian, "Feature selection using principal feature analysis", in "Proceedings of the 15th ACM international conference on Multimedia", pp. 301–304 (ACM, 2007).

McKeown, G., M. F. Valstar, R. Cowie and M. Pantic, "The semaine corpus of emotionally coloured character interactions", in "Multimedia and Expo (ICME), 2010 IEEE International Conference on", pp. 1079–1084 (IEEE, 2010).

Metallinou, A., C. Busso, S. Lee and S. Narayanan, "Visual emotion recognition using compact facial representations and viseme information", in "Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on", pp. 2474–2477 (IEEE, 2010).

Mirowski, P., M. Ranzato and Y. LeCun, "Dynamic auto-encoders for semantic indexing", in "Proceedings of the NIPS 2010 Workshop on Deep Learning", pp. 1–9 (2010).

Mohamed, A.-r., G. E. Dahl and G. Hinton, "Acoustic modeling using deep belief networks", IEEE Transactions on Audio, Speech, and Language Processing **20**, 1, 14–22 (2012).

Morgan, N., "Deep and wide: Multiple layers in automatic speech recognition", IEEE Transactions on Audio, Speech, and Language Processing **20**, 1, 7–13 (2012).

Mower, E., M. J. Mataric and S. Narayanan, "A framework for automatic human emotion classification using emotion profiles", IEEE Transactions on Audio, Speech, and Language Processing **19**, 5, 1057–1070 (2011).

Muslea, I., S. Minton and C. Knoblock, "Active learning with multiple views", in "Journal of Artificial Intelligence Research", (2006).

Nair, V. and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in "Proceedings of the 27th international conference on machine learning (ICML-10)", pp. 807–814 (2010).

Nasrabadi, N. M., "Pattern recognition and machine learning", Journal of electronic imaging **16**, 4, 049901 (2007).

Ngiam, J., A. Khosla, M. Kim, J. Nam, H. Lee and A. Y. Ng, "Multimodal deep learning", in "Proceedings of the 28th international conference on machine learning (ICML-11)", pp. 689–696 (2011).

Pantic, M., G. Caridakis, E. André, J. Kim, K. Karpouzis and S. Kollias, "Multimodal emotion recognition from low-level cues", in "Emotion-Oriented Systems", pp. 115–132 (Springer, 2011).

Pantic, M., M. Valstar, R. Rademaker and L. Maat, "Web-based database for facial expression analysis", in "Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on", pp. 5–pp (IEEE, 2005).

Pfister, T., K. Simonyan, J. Charles and A. Zisserman, "Deep convolutional neural networks for efficient pose estimation in gesture videos", in "Asian Conference on Computer Vision", pp. 538–552 (Springer, 2014).

Polzehl, T., S. Sundaram, H. Ketabdar, M. Wagner and F. Metze, "Emotion classification in children's speech using fusion of acoustic and linguistic features", in "Tenth Annual Conference of the International Speech Communication Association", (2009).

Poultney, C., S. Chopra, Y. L. Cun *et al.*, "Efficient learning of sparse representations with an energy-based model", in "Advances in neural information processing systems", pp. 1137–1144 (2007).

Qi, G.-J., X.-S. Hua, Y. Rui, J. Tang and H.-J. Zhang, "Two-dimensional active learning for image classification", in "Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on", pp. 1–8 (IEEE, 2008).

Raina, R., A. Battle, H. Lee, B. Packer and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data", in "Proceedings of the 24th international conference on Machine learning", pp. 759–766 (ACM, 2007).

Ranganathan, H., S. Chakraborty and S. Panchanathan, "Multimodal emotion recognition using deep learning architectures", in "IEEE Winter Conference on Applications of Computer Vision (WACV)", (2016a).

Ranganathan, H., S. Chakraborty and S. Panchanathan, "Transfer of multimodal emotion features in deep belief networks", in "Signals, Systems and Computers, 2016 50th Asilomar Conference on", pp. 449–453 (IEEE, 2016b).

Ranganathan, H., H. Venkateswara, S. Chakraborty and S. Panchanathan, "Deep active learning for image classification", in "International Conference on Image Processing (ICIP 2017)", (2016c).

Ranzato, M., J. Susskind, V. Mnih and G. Hinton, "On deep generative models with applications to recognition", in "IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", (2011).

Rosca, M., *Networks with Emotions*, URL `https://www.doc.ic.ac.uk/teaching/distinguished-projects/2014/mrosca.pdf` (2018 (accessed March 9, 2018)).

Rothe, R., R. Timofte and L. Van Gool, "Dex: Deep expectation of apparent age from a single image", in "Proceedings of the IEEE International Conference on Computer Vision Workshops", pp. 10–15 (2015).

Salakhutdinov, R. and H. Larochelle, "Efficient learning of deep boltzmann machines", in "Artificial Intelligence and Statistics Conference (AISTATS)", (2010).

Sanderson, C., "Biometric person recognition: Face, speech and fusion", in "VDM Verlag", (2008).

Schohn, G. and D. Cohn, "Less is more: Active learning with support vector machines", in "Proceedings of the International Conference on Machine Learning (ICML)", (2000).

Schuller, B., S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. v. Son, F. Weninger, F. Eyben, T. Bocklet *et al.*, "The interspeech 2012 speaker trait challenge", in "Thirteenth Annual Conference of the International Speech Communication Association", (2012).

Schulz, H., A. Müller and S. Behnke, "Investigating convergence of restricted boltzmann machine learning", in "NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning", (2010).

Settles, B., "Active learning literature survey", in "Technical Report 1648, University of Wisconsin-Madison", (2010).

Settles, B. and M. Craven, "An analysis of active learning strategies for sequence labeling tasks", in "Proceedings of the conference on empirical methods in natural language processing", pp. 1070–1079 (Association for Computational Linguistics, 2008).

Shen, D., J. Zhang, J. Su, G. Zhou and C.-L. Tan, "Multi-criteria-based active learning for named entity recognition", in "Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics", p. 589 (Association for Computational Linguistics, 2004).

Sherrah, J. and S. Gong, "Fusion of perceptual cues for robust tracking of head pose and position", Pattern Recognition **34**, 8, 1565–1572 (2001).

Sivaram, G. S. and H. Hermansky, "Sparse multilayer perceptron for phoneme recognition", IEEE Transactions on Audio, Speech, and Language Processing **20**, 1, 23–29 (2012).

Smolensky, P., "Information processing in dynamical systems: Foundations of harmony theory", Tech. rep., COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE (1986).

Sohn, K., D. Y. Jung, H. Lee and A. O. Hero, "Efficient learning of sparse, distributed, convolutional feature representations for object recognition", in "Computer Vision (ICCV), 2011 IEEE International Conference on", pp. 2643–2650 (IEEE, 2011).

Srivastava, N., "Improving neural networks with dropout", University of Toronto **182** (2013).

Srivastava, N. and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines", in "Advances in neural information processing systems", pp. 2222–2230 (2012).

Stark, F., C. Hazirbas, R. Triebel and D. Cremers, "Captcha recognition with active deep learning", in "Workshop on New Challenges in Neural Computation", (2015a).

Stark, F., C. Hazırbas, R. Triebel and D. Cremers, "Captcha recognition with active deep learning", in "Workshop New Challenges in Neural Computation 2015", p. 94 (Citeseer, 2015b).

Steidl, S., M. Levit, A. Batliner, E. Noth and H. Niemann, "" of all things the measure is man" automatic classification of emotions and inter-labeler consistency [speech-based emotion recognition]", in "Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on", vol. 1, pp. I–317 (IEEE, 2005).

Stuhlsatz, A., C. Meyer, F. Eyben, T. Zielke, G. Meier and B. Schuller, "Deep neural networks for acoustic emotion recognition: raising the benchmarks", in "Acoustics, speech and signal processing (ICASSP), 2011 IEEE international conference on", pp. 5688–5691 (IEEE, 2011).

Sugiyama, M., "Active learning in approximately linear regression based on conditional expectation of generalization error", Journal of Machine Learning Research **7**, Jan, 141–166 (2006).

Sugiyama, M. and S. Nakajima, "Pool-based active learning in approximate linear regression", Machine Learning **75**, 3, 249–274 (2009).

Sun, Y., X. Wang and X. Tang, "Deep convolutional network cascade for facial point detection", in "Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on", pp. 3476–3483 (IEEE, 2013).

Sutskever, I., J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning", in "International conference on machine learning", pp. 1139–1147 (2013).

Szegedy, C., A. Toshev and D. Erhan, "Deep neural networks for object detection", in "Advances in neural information processing systems", pp. 2553–2561 (2013).

Tang, Y. and C. Eliasmith, "Deep networks for robust visual recognition", in "Proceedings of the 27th International Conference on Machine Learning (ICML-10)", pp. 1055–1062 (Citeseer, 2010).

Tang, Y. and A. Mohamed, "Multiresolution deep belief networks", in "International Conference on Artificial Intelligence and Statistics (AISTATS)", (2012).

Taylor, G. W., G. E. Hinton and S. T. Roweis, "Modeling human motion using binary latent variables", in "Advances in neural information processing systems", pp. 1345–1352 (2007).

Tieleman, T. and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", COURSERA: Neural networks for machine learning **4**, 2, 26–31 (2012).

Tong, S. and D. Koller, "Support vector machine active learning with applications to text classification", Journal of machine learning research **2**, Nov, 45–66 (2001).

Toshev, A. and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 1653–1660 (2014).

Valstar, M. and M. Pantic, "Induced disgust, happiness and surprise: an addition to the mmi facial expression database", in "Proc. 3rd Intern. Workshop on EMOTION (satellite of LREC): Corpora for Research on Emotion and Affect", p. 65 (2010).

Ververidis, D. and C. Kotropoulos, "Fast and accurate sequential floating forward feature selection with the bayes classifier applied to speech emotion recognition", signal processing **88**, 12, 2956–2970 (2008).

Vincent, P., H. Larochelle, Y. Bengio and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders", in "Proceedings of the 25th international conference on Machine learning", pp. 1096–1103 (ACM, 2008).

Vogt, T. and E. André, "Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition", in "Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on", pp. 474–477 (IEEE, 2005).

Wang, D. and Y. Shang, "A new active labeling method for deep learning", in "International Joint Conference on Neural Networks (IJCNN)", (2014a).

Wang, D. and Y. Shang, "A new active labeling method for deep learning", in "Neural Networks (IJCNN), 2014 International Joint Conference on", pp. 112–119 (IEEE, 2014b).

Wang, J., J. Yang, K. Yu, F. Lv, T. Huang and Y. Gong, "Locality-constrained linear coding for image classification", in "Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on", pp. 3360–3367 (IEEE, 2010).

Wang, J., Y. Yang, J. Mao, Z. Huang, C. Huang and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2285–2294 (2016).

Wang, M. and X.-S. Hua, "Active learning in multimedia annotation and retrieval: A survey", ACM Transactions on Intelligent Systems and Technology (TIST) **2**, 2, 10 (2011).

Wang, X., L. Zhang, L. Lin, Z. Liang and W. Zuo, "Deep joint task learning for generic object extraction", in "Advances in Neural Information Processing Systems", pp. 523–531 (2014).

Willett, R., R. Nowak and R. M. Castro, "Faster rates in regression via active learning", in "Advances in Neural Information Processing Systems", pp. 179–186 (2006).

Wimmer, M., B. Schuller, D. Arsic, B. Radig and G. Rigoll, "Low-level fusion of audio and video feature for multi-modal emotion recognition", in "Proc. 3rd Int. Conf. on Computer Vision Theory and Applications VISAPP, Funchal, Madeira, Portugal", pp. 145–151 (2008).

Wöllmer, M., A. Metallinou, F. Eyben, B. Schuller and S. Narayanan, "Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling", in "Proc. INTERSPEECH 2010, Makuhari, Japan", pp. 2362–2365 (2010).

Wu, J., *Introduction to Convolutional Neural Networks*, URL `https://cs.nju.edu.cn/wujx/paper/CNN.pdf` (2018 (accessed March 9, 2018)).

Wu, J., V. S. Sheng, J. Zhang, P. Zhao and Z. Cui, "Multi-label active learning for image classification", in "Image Processing (ICIP), 2014 IEEE International Conference on", pp. 5227–5231 (IEEE, 2014).

Xue, X., W. Zhang, J. Zhang, B. Wu, J. Fan and Y. Lu, "Correlative multi-label multi-instance image annotation", in "Computer Vision (ICCV), 2011 IEEE International Conference on", pp. 651–658 (IEEE, 2011).

Yosinski, J., J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?", in "Advances in neural information processing systems", pp. 3320–3328 (2014).

Yu, H. and S. Kim, "Passive sampling for regression", in "Data Mining (ICDM), 2010 IEEE 10th International Conference on", pp. 1151–1156 (IEEE, 2010).

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

Zhang, Z., P. Luo, C. C. Loy and X. Tang, "Facial landmark detection by deep multi-task learning", in "European Conference on Computer Vision", pp. 94–108 (Springer, 2014).

Zhou, S., Q. Chen and X. Wang, "Active deep networks for semi-supervised sentiment classification", in "International Conference on Computational Linguistics", (2010a).

Zhou, S., Q. Chen and X. Wang, "Active deep networks for semi-supervised sentiment classification", in "Proceedings of the 23rd International Conference on Computational Linguistics: Posters", pp. 1515–1523 (Association for Computational Linguistics, 2010b).

Zhu, J., S. Liao, Z. Lei and S. Z. Li, "Multi-label convolutional neural network based pedestrian attribute classification", Image and Vision Computing **58**, 224–229 (2017).

Zhu, S., Z. Shi, C. Sun and S. Shen, "Deep neural network based image annotation", Pattern Recognition Letters **65**, 103–108 (2015).

Zhu, X., J. Lafferty and Z. Ghahramani, "Combining active learning and semi-supervised learning using gaussian fields and harmonic functions", in "ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining", vol. 3 (2003).

Zliobaite, I., A. Bifet, B. Pfahringer and G. Holmes, "Active learning with drifting streaming data", IEEE transactions on neural networks and learning systems **25**, 1, 27–39 (2014).

APPENDIX A

DERIVATIVE OF THE JOINT OBJECTIVE FUNCTIONS

A.0.0.1 Derivation of Multi-class Joint Loss Function

In this section the partial derivative of Equation (5.5) for the backpropagation algorithm is outlined.

We use the standard backpropagation algorithm to learn the weights of the DBN. The output of the N-th layer of the network (before the loss) for a data point x_i, is given by the vector h_i^N. We define $p_{ij} \coloneqq f_j(x_i) = e^{h_{ij}^N} / \sum_{j'} e^{h_{ij'}^N}$, the probability that data point x_i belongs to class j. The loss in terms of probabilities is given by,

$$E(X_l, X_u, Y_l) = -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \log p_{ij} - \frac{\lambda}{n_u} \sum_{i=n_l+1}^{n} \sum_{j=1}^{C} p_{ij} \log p_{ij}. \tag{A.1}$$

During implementation, p_{ij} is expressed as $p_{ij} \coloneqq e^{(h_{ij}^N - m_i)} / \sum_{j'} e^{(h_{ij'}^N - m_i)}$ where, m_i is the maximum of $h_{ij'}^N$ over j'. It is meant to ensure that $e^{h^N{}_{ij}}$ does not go out of bounds and it does not change the value of p_{ij}. We represent h_{pq}^N as h_{pq} for ease of notation.

A.0.0.2 Gradient of cross entropy loss

The cross-entropy over a batch of n_l data points is represented by L.

$$\begin{aligned}
L =& -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \log p_{ij} \\
=& -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \log \frac{e^{h_{ij}}}{\sum_{j'} e^{h_{ij'}}} \\
=& -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \log \frac{e^{(h_{ij} - m_i)}}{\sum_{j'} e^{(h_{ij'} - m_i)}} \\
=& -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \big((h_{ij} - m_i) - \log \sum_{j'} e^{(h_{ij'} - m_i)} \big)
\end{aligned} \tag{A.2}$$

The partial derivative $\frac{\partial L}{\partial h_{pq}}$ is the gradient of L with respect to h_{pq}, which is the q-th component of the p-th data point in the output of the N-th layer.

$$\begin{aligned}
\frac{\partial L}{\partial h_{pq}} =& \frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{C} 1\{y_i = j\} \big(- I\{^{i=p,}_{j=q}\} + \frac{e^{(h_{iq} - m_i)}}{\sum_{j'} e^{(h_{ij'} - m_i)}} I\{i = p\} \big) \\
& I\{\text{condition}\} \text{ is 1 when the condition is true, else it is 0} \qquad \text{(A.3)} \\
=& \frac{1}{n_l} \big(- 1\{y_p = q\} + p_{pq} \big) \qquad \text{(A.4)}
\end{aligned}$$

A.0.0.3 Gradient of entropy loss

The entropy over a batch of n_u data points is represented by H.

$$\begin{aligned}
H =& -\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{C} p_{ij}\log p_{ij} \\
=& -\frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{C}\frac{e^{h_{ij}}}{\sum_{j'}e^{h_{ij'}}}\log\frac{e^{h_{ij}}}{\sum_{j'}e^{h_{ij'}}} \\
=& \frac{\lambda}{n_u}\sum_{i=n_l+1}^{n}\sum_{j=1}^{C} -\frac{e^{(h_{ij}-m_i)}}{\sum_{j'}e^{(h_{ij'}-m_i)}}(h_{ij}-m_i) + \frac{e^{(h_{ij}-m_i)}}{\sum_{j'}e^{(h_{ij'}-m_i)}}\log\sum_{j'}e^{(h_{ij'}-m_i)}
\end{aligned} \tag{A.5}$$

The partial derivative $\frac{\partial H}{\partial h_{pq}}$ is the gradient of H with respect to h_{pq}, which is the q-th component of the p-th data point in the output of the N-th layer. We will drop the summation outside and introduce it later.

$$\begin{aligned}
\frac{\partial H}{\partial h_{pq}} =& -\frac{\sum_{j'}e^{(h_{ij'}-m_i)}\big(e^{(h_{ij}-m_i)}(h_{ij}-m_i)+e^{(h_{ij}-m_i)}\big)I\{^{i=p,}_{j=q}\}}{\sum_{j'}\left(e^{(h_{ij'}-m_i)}\right)^2} \\
&+\frac{e^{(h_{ij}-m_i)}(h_{ij}-m_i)e^{(h_{iq}-m_i)}I\{i=p\}}{\sum_{j'}\left(e^{(h_{ij'}-m_i)}\right)^2} \\
&+\frac{\sum_{j'}e^{(h_{ij'}-m_i)}\big(e^{(h_{ij}-m_i)}\log\sum_{j'}e^{(h_{ij'}-m_i)}I\{^{i=p,}_{j=q}\}+\frac{e^{(h_{ij}-m_i)}}{\sum_{j'}e^{(h_{ij'}-m_i)}}e^{(h_{iq}-m_i)}I\{i=p\}\big)}{\sum_{j'}\left(e^{(h_{ij'}-m_i)}\right)^2} \\
&-\frac{e^{(h_{ij}-m_i)}\log\sum_{j'}e^{(h_{ij'}-m_i)}e^{(h_{iq}-m_i)}I\{i=p\}}{\sum_{j'}\left(e^{(h_{ij'}-m_i)}\right)^2}
\end{aligned} \tag{A.6}$$

$$I\{\text{condition}\} \text{ is 1 when the condition is true, else it is 0} \tag{A.7}$$

$$\begin{aligned}
=& -p_{ij}(h_{ij}-m_i)I\{^{i=p,}_{j=q}\} - p_{ij}I\{^{i=p,}_{j=q}\} + p_{ij}(h_{ij}-m_i)p_{iq}I\{i=p\} \\
&+p_{ij}\log\sum_{j'}e^{(h_{ij'}-m_i)}I\{^{i=p,}_{j=q}\} + p_{ij}p_{iq}I\{i=p\} - p_{ij}\log\sum_{j'}e^{(h_{ij'}-m_i)}p_{iq}I\{i=p\}
\end{aligned} \tag{A.8}$$

Reintroducing the summation;

$$
\begin{aligned}
&\tfrac{\lambda}{n_u}\textstyle\sum_{i=n_l+1}^{n}\sum_{j=1}^{C} \\
=&\frac{\lambda}{n_u}\Bigg[-p_{pq}(h_{pq}-m_p)-p_{pq}+p_{pq}\sum_{j=1}^{C}p_{pj}(h_{pj}-m_p)+p_{pq}\log\sum_{j'}e^{(h_{pj'}-m_p)} \\
&+p_{pj}p_{pq}-\log\sum_{j'}e^{(h_{pj'}-m_p)}p_{pq}\sum_{j=1}^{C}p_{pj}\Bigg] \qquad \text{(A.9)} \\
&\text{since } \textstyle\sum_{j=1}^{C}p_{pj}=1 \\
=&\frac{\lambda}{n_u}\Bigg[-p_{pq}h_{pq}-p_{pq}m_p+p_{pq}\Big(\sum_{j=1}^{C}p_{pj}h_{pj}-\sum_{j=1}^{C}p_{pj}m_p\Big)\Bigg] \qquad \text{(A.10)} \\
=&\frac{\lambda}{n_u}\Bigg[-p_{pq}h_{pq}+p_{pq}\sum_{j=1}^{C}p_{pj}h_{pj}\Bigg] \qquad \text{(A.11)} \\
=&\frac{\lambda}{n_u}p_{pq}\Bigg[\sum_{j=1}^{C}p_{pj}h_{pj}-h_{pq}\Bigg] \qquad \text{(A.12)}
\end{aligned}
$$

A.0.0.4 Overall gradient

The overall gradient is the sum of the cross-entropy gradient and the entropy gradient. We outline the derivative of the loss $E(.)$ with respect to h_{pq}^N, which is the q-th component of the p-th data point in the output of the N-th layer as,

$$
\frac{\partial E}{\partial h_{pq}^N}=\begin{cases}\frac{1}{n_l}\Big(p_{pq}-1\{y_p=q\}\Big), & p\in[1,\ldots,n_l] \\ \frac{\lambda}{n_u}p_{pq}\Big(\sum_j^C p_{pj}h_{pj}^N-h_{pq}^N\Big), & p\in[n_l+1,\ldots,n].\end{cases} \qquad \text{(A.13)}
$$

During the training procedure, the derivative $\partial E/\partial h^N$ is backpropagated through the network in order to update the weights of the network.

A.0.0.5 Derivation of Multi-label Joint Loss Function

The joint objective function for deep multi-label active learning for image classification without label correlation is given by:

$$\mathcal{L}(X^L, X^u, Y^L) = \mathcal{C}(Y^L, \widehat{Y}) + \lambda \mathcal{H}(\widehat{Y}) \tag{A.14}$$

where $\mathcal{C}(Y^L, \widehat{Y})$ and $\mathcal{H}(\widehat{Y})$ are as in equations (6.1) and (6.2) respectively. λ is a constant that controls the importance of the entropy loss.

$$\begin{aligned}\mathcal{C}(Y^L, \widehat{Y}) = &- \frac{1}{n_l} \sum_{n=1}^{n_l} \sum_{m=1}^{M} [y_{nm} \log(\widehat{y}_{nm}) \\ &+ (1 - y_{nm}) \log(1 - \widehat{y}_{nm})].\end{aligned} \tag{A.15}$$

$$\begin{aligned}\mathcal{H}(\widehat{Y}) = &- \frac{1}{n_u} \sum_{n=n_l+1}^{n_l+n_u} \sum_{m=1}^{M} [\widehat{y}_{nm} \log(\widehat{y}_{nm}) \\ &+ (1 - \widehat{y}_{nm}) \log(1 - \widehat{y}_{nm})].\end{aligned} \tag{A.16}$$

The Sigmoid function is applied on the activation h^N_{nm}, where h^N_{nm} is the n^{th} component of the m^{th} data point in the output of the N^{th} layer. Therefore we have,

$$\widehat{y}_{nm} = \left(\frac{1}{1 + e^{-h^N_{nm}}} \right) \tag{A.17}$$

We compute the derivative of the joint objective function $\mathcal{L}(.)$ with respect to h^N_{pq}, which is the q^{th} component of the p^{th} data point in the output of the N^{th} layer. We observe that,

$$\begin{aligned}&\frac{\partial}{\partial h^N_{pq}} \Big\{ y_{nm} \log(\widehat{y}_{nm}) + (1 - y_{nm}) \log(1 - \widehat{y}_{nm}) \Big\} \\ &\quad = 0, \text{ for } (n, m) \neq (p, q)\end{aligned} \tag{A.18}$$

and

$$\begin{aligned}&\frac{\partial}{\partial h^N_{pq}} \Big\{ \widehat{y}_{nm} \log(\widehat{y}_{nm}) + (1 - \widehat{y}_{nm}) \log(1 - \widehat{y}_{nm}) \Big\} \\ &\quad = 0, \text{ for} (n, m) \neq (p, q)\end{aligned} \tag{A.19}$$

First, let us compute,

$$\begin{aligned}\frac{\partial \log(\widehat{y}_{pq})}{\partial h^N_{pq}} &= \frac{\partial}{\partial h^N_{pq}} \Big[- \log(1 + e^{-h^N_{pq}}) \Big] \\ &= \frac{e^{-h^N_{pq}}}{1 + e^{-h^N_{pq}}} = e^{-h^N_{pq}} \widehat{y}_{pq} = 1 - \widehat{y}_{pq}\end{aligned} \tag{A.20}$$

$$\begin{aligned}\frac{\partial \log\left(1-\widehat{y}_{pq}\right)}{\partial h_{pq}^{N}} =& \frac{\partial}{\partial h_{pq}^{N}}\left[\log e^{-h_{pq}^{N}} - \log(1+e^{-h_{pq}^{N}})\right] \\ =& -1 + \frac{e^{-h_{pq}^{N}}}{1+e^{-h_{pq}^{N}}} \\ =& -\left(1 - e^{-h_{pq}^{N}}\widehat{y}_{pq}\right) = -\widehat{y}_{pq} \end{aligned} \tag{A.21}$$

$$\frac{\partial \widehat{y}_{pq}}{\partial h_{pq}^{N}} = -\, e^{-h_{pq}^{N}}\widehat{y}_{pq}^{2} = \widehat{y}_{pq}(1-\widehat{y}_{pq}) \tag{A.22}$$

A.0.0.6 Gradient of Sigmoid Cross Entropy Loss

Substituting from equation $(A.20)$ and $(A.21)$, we get

$$\begin{aligned}\frac{\partial \mathcal{C}}{\partial h_{pq}^{N}} =& -\frac{1}{n_l}\left[\sum_{n=1}^{n_l}\sum_{m=1}^{M}\frac{\partial}{\partial h_{pq}^{N}}\Big\{ y_{nm}\log\left(\widehat{y}_{nm}\right) \right. \\ & \left. + (1-y_{nm})\log\left(1-\widehat{y}_{nm}\right)\Big\}\right] \\ =& -\frac{1}{n_l}\left[\frac{\partial}{\partial h_{pq}^{N}}\Big\{ y_{pq}\log\left(\widehat{y}_{pq}\right) \right. \\ & \left. + (1-y_{pq})\log\left(1-\widehat{y}_{pq}\right)\Big\}\right] \\ =& -\frac{1}{n_l}\left[y_{pq}(1-\widehat{y}_{pq}) + (1-y_{pq})(-\widehat{y}_{pq})\right] \\ =& -\frac{1}{n_l}(y_{pq}-\widehat{y}_{pq}) \quad \text{for} \quad 1\le p\le n_l,\ 1\le q\le M, \end{aligned} \tag{A.23}$$

A.0.0.7 Gradient of the Multilabel Entropy Loss

Similarly, using equations $(A.19), (A.20), (A.21)$ and $(A.22)$ we get,

$$\begin{aligned}
\frac{\partial \mathcal{H}(\widehat{Y})}{\partial h_{pq}^N} =& -\frac{1}{n_u}\Bigg[\sum_{n=n_l+1}^{n_l+n_u}\sum_{m=1}^{M}\frac{\partial}{\partial h_{pq}^N}\Big\{\widehat{y}_{nm}\log\left(\widehat{y}_{nm}\right) \\
&+ (1-\widehat{y}_{nm})\log\left(1-\widehat{y}_{nm}\right)\Big\}\Bigg] \\
=& -\frac{1}{n_u}\Bigg[\frac{\partial}{\partial h_{pq}^N}\Big\{\widehat{y}_{pq}\log\left(\widehat{y}_{pq}\right) \\
&+ (1-\widehat{y}_{pq})\log\left(1-\widehat{y}_{pq}\right)\Big\}\Bigg] \\
=& -\frac{1}{n_u}\Big[\widehat{y}_{pq}(1-\widehat{y}_{pq}) + \widehat{y}_{pq}(1-\widehat{y}_{pq})\log\widehat{y}_{pq} \\
&+ (1-\widehat{y}_{pq})(-\widehat{y}_{pq}) - \widehat{y}_{pq}(1-\widehat{y}_{pq})\log\left(1-\widehat{y}_{pq}\right)\Big] \\
=& -\frac{1}{n_u}\left[\widehat{y}_{pq}(1-\widehat{y}_{pq})\log\left(\frac{\widehat{y}_{pq}}{1-\widehat{y}_{pq}}\right)\right] \\
&\text{for } n_l+1 \le p \le n_l+n_u,\ 1 \le q \le M,
\end{aligned} \tag{A.24}$$

A.0.0.8 Overall Gradient

Summarizing equations $(A.23)$ and $(A.24)$, we get the derivative to the joint loss function $\mathcal{L}(.)$ with respect to h_{pq}^N as,

$$\frac{\partial \mathcal{L}}{\partial h_{pq}^N} = \begin{cases} -\frac{1}{n_l}[y_{pq}-\widehat{y}_{pq}], \\ \qquad\qquad 1 \le p \le n_l,\ 1 \le q \le M \\ -\frac{\lambda}{n_u}\left[\widehat{y}_{pq}(1-\widehat{y}_{pq})\log\left(\frac{\widehat{y}_{pq}}{1-\widehat{y}_{pq}}\right)\right], \\ \qquad n_l+1 \le p \le n_l+n_u,\ 1 \le q \le M \end{cases} \tag{A.25}$$

A.0.0.9 Derivation of Joint Loss Function for Regression

The gradient of the joint objective function for deep active regression is given by:

$$\nabla_\phi \mathcal{J}(\phi, X^l, Y^l, X^u) = \nabla_\phi \mathcal{L}(\phi; X^l, Y^l) + \lambda \nabla_\phi \mathcal{U}(\phi; X^u) \tag{A.26}$$

To keep the notation simple, let

$$\nabla_\phi \mathcal{J} = \nabla_\phi \mathcal{J}_1 + \lambda \nabla_\phi \mathcal{J}_2 \tag{A.27}$$

where

$$\mathcal{J}_1 = \mathcal{L}(\phi; X^l, Y^l) \text{ and } \mathcal{J}_2 = \mathcal{U}(\phi; X^u),$$

A.0.0.10 Gradient of Standard L2 Loss

$$\nabla_\phi \mathcal{J}_1 = \frac{\partial \mathcal{J}_1}{\partial \phi} = \left\{ \frac{\partial \mathcal{J}_1}{\partial \phi_1}, \frac{\partial \mathcal{J}_1}{\partial \phi_2}, \cdots, \frac{\partial \mathcal{J}_1}{\partial \phi_l} \right\} \tag{A.28}$$

Here,

$$\begin{aligned}
\frac{\partial \mathcal{J}_1}{\partial \phi_1} &= -\frac{2}{n'_l} \sum_{i=1}^{n'_l} (y_i - \widehat{y}_i) \frac{\partial g(x_i; \phi)}{\partial \phi_1} \\
\frac{\partial \mathcal{J}_1}{\partial \phi_2} &= -\frac{2}{n'_l} \sum_{i=1}^{n'_l} (y_i - \widehat{y}_i) \frac{\partial g(x_i; \phi)}{\partial \phi_2} \\
\cdots &= \cdots \\
\cdots &= \cdots \\
\cdots &= \cdots \\
\frac{\partial \mathcal{J}_1}{\partial \phi_l} &= -\frac{2}{n'_l} \sum_{i=1}^{n'_l} (y_i - \widehat{y}_i) \frac{\partial g(x_i; \phi)}{\partial \phi_l}
\end{aligned} \tag{A.29}$$

A.0.0.11 Gradient of EMOC Loss

We have

$$\nabla_\phi \mathcal{J}_2 = \frac{\partial \mathcal{J}_2}{\partial \phi} = \left\{ \frac{\partial \mathcal{J}_2}{\partial \phi_1}, \frac{\partial \mathcal{J}_2}{\partial \phi_2}, \cdots, \frac{\partial \mathcal{J}_2}{\partial \phi_l} \right\} \tag{A.30}$$

We observe that

$$\begin{aligned}
\frac{\partial \mathcal{J}_2}{\partial \phi_j} &= \sum_{x' \in X^u} \mathbb{E}_x \frac{\partial}{\partial \phi_j} \left|\left| \nabla_\phi g(x; \phi)^\top \nabla_\phi \mathcal{L}(\phi; (x', \bar{y}')) \right|\right|_1 \\
&= \sum_{x' \in X^u} \mathbb{E}_x \left(Q_j\right), \qquad \forall j \in \{1, 2, \cdots, l\}.
\end{aligned} \tag{A.31}$$

where

$$Q_j = \frac{\partial}{\partial \phi_j} \left|\left| \nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi;(x',\bar{y}')) \right|\right|_1 \tag{A.32}$$

The product $\nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi;(x',\bar{y}'))$ in equation $(A.32)$ is a scalar product that yields a scalar $z = z\,(x, x', \phi, \bar{y}')$. When $\nabla_\phi g(x;\phi)$ and $\nabla_\phi \mathcal{L}(\phi;(x',\bar{y}'))$ are non-zero, $||z||$ is zero if and only if the two gradient vectors are orthogonal. In such a case, we may change $\bar{y}'$ accordingly, so that it is different from zero. We have:

$$\frac{\partial ||z||}{\partial \phi} = \begin{cases} \frac{\partial z}{\partial \phi} & \text{if } \ z > 0 \\ -\frac{\partial z}{\partial \phi} & \text{if } \ z < 0 \end{cases} \tag{A.33}$$

Hence, when $z > 0$

$$Q_j = \frac{\partial}{\partial \phi_j} \left[\nabla_\phi g(x;\phi)^\top \nabla_\phi \mathcal{L}(\phi;(x',\bar{y}')) \right] = Q_{j_1} + Q_{j_2} \tag{A.34}$$

where

$$Q_{j_1} = \frac{\partial}{\partial \phi_j} \left(\nabla_\phi g(x;\phi)^\top \right) \nabla_\phi \mathcal{L}(\phi;(x',\bar{y}')) \tag{A.35}$$

and

$$Q_{j_2} = \nabla_\phi g(x;\phi)^\top \frac{\partial}{\partial \phi_j} \left(\nabla_\phi \mathcal{L}(\phi;(x',\bar{y}')) \right) \tag{A.36}$$

First we compute Q_{j_1} as the inner product of

$$\frac{\partial}{\partial \phi_j} \left(\frac{\partial g}{\partial \phi_1}, \frac{\partial g}{\partial \phi_2}, \cdots, \frac{\partial g}{\partial \phi_l} \right)$$

and

$$\left[(-2)(\bar{y}' - g(x';\phi)) \nabla_\phi g(x';\phi) \right].$$

Note: We are computing Q_{j_1} for a single $x' \in X^u$.
Next we compute Q_{j_2} as the inner product of
$\left(\frac{\partial g}{\partial \phi_1}, \frac{\partial g}{\partial \phi_2}, \cdots, \frac{\partial g}{\partial \phi_l} \right)$ and

$$(-2) \left[(\bar{y}' - g(x';\phi)) \frac{\partial}{\partial \phi_j} \nabla_\phi g(x';\phi) - \left(\frac{\partial g}{\partial \phi_j} \right) \nabla_\phi g(x';\phi) \right].$$

We have used the product rule for differentiation in computing the derivative $\frac{\partial}{\partial \phi_i} \left(\nabla_\phi \mathcal{L}(\phi;(x',\bar{y}')) \right)$ as

$$(-2) \left[(\bar{y}' - g(x';\phi)) \frac{\partial}{\partial \phi_j} \nabla_\phi g(x';\phi) - \left(\frac{\partial g}{\partial \phi_j} \right) \nabla_\phi g(x';\phi) \right].$$

Here,

$$\frac{\partial}{\partial \phi_j} \left(\nabla_\phi g(x';\phi) \right) = \left(\frac{\partial^2 g}{\partial \phi_j \partial \phi_1}, \frac{\partial^2 g}{\partial \phi_j \partial \phi_2}, ..., \frac{\partial^2 g}{\partial \phi_j \partial \phi_l} \right) \tag{A.37}$$

and

$$\frac{\partial g}{\partial \phi_j}(\nabla_\phi g(x';\phi)) = \frac{\partial g}{\partial \phi_j}\left(\frac{\partial g}{\partial \phi_1}, \frac{\partial g}{\partial \phi_2}, \cdots, \frac{\partial g}{\partial \phi_l}\right) \tag{A.38}$$

Similarly when $z < 0$, we have:

$$Q_j = -(Q_{j_1} + Q_{j_2}) \tag{A.39}$$

We compute $Q_{j_1} + Q_{j_2}$ for a fixed $x' \in X^u$ and for all $x \in X^l$. Then we compute the expected value $\mathbb{E}_x(Q_j)$. We do this for every $x' \in X^u$ and then compute $\sum_{x' \in X^u} \mathbb{E}_x(Q_j),\ \forall j \in \{1, 2, \ldots, l\}$ to get $\nabla_\phi \mathcal{J}_2$.

A.0.0.12 Overall Gradient

The overall gradient of the joint objective function for deep active regression is given by:

$$\nabla_\phi \mathcal{J} = \nabla_\phi \mathcal{J}_1 + \nabla_\phi \mathcal{J}_2.$$

APPENDIX B

PERMISSION STATEMENTS FROM CO-AUTHORS

Permission for including co-authored material in this dissertation was obtained from co-authors, Prof. Sethuraman Panchanathan, Dr. Shayok Chakraborty and Dr. Hemanth Venkateswara.