ABSTRACT

Title of dissertation: TOWARDS BUILDING GENERALIZABLE SPEECH EMOTION RECOGNITION MODELS

Saurabh Sahu, Doctor of Philosophy, 2019

Dissertation directed by: Professor Carol Espy-Wilson
Department of Electrical and Computer Engineering

Detecting the mental state of a person has implications in psychiatry, medicine, psychology and human-computer interaction systems among others. It includes (but is not limited to) a wide variety of problems such as emotion detection, valence-affect-dominance states prediction, mood detection and detection of clinical depression. In this thesis we focus primarily on emotion recognition. Like any recognition system, building an emotion recognition model consists of the following two steps:

1. Extraction of meaningful features that would help in classification

2. Development of an appropriate classifier

Speech data being non-invasive and the ease with which it can be collected has made it a popular choice to extract features from. However, an ideal system designed should be agnostic to speaker and channel effects. While feature normalization schemes can counter these problems to some extent, we still see a drastic drop in performance when the training and test data-sets are unmatched. In this dissertation we explore some novel ways towards building models that are more robust to speaker and domain differences.

Training discriminative classifiers involves learning a conditional distribution $p(y_i|\mathbf{x_i})$, given a set of feature vectors $\mathbf{x_i}$ and the corresponding labels $y_i, i = 1..N$. For a classifier to be generalizable and not overfit to training data, the resulting conditional distribution $p(y_i|\mathbf{x_i})$ is desired to be smoothly varying over the inputs $\mathbf{x_i}$. Adversarial training procedures enforce this smoothness using manifold regularization techniques. Manifold regularization makes the model's output distribution more robust to local perturbation added to a datapoint $\mathbf{x_i}$. In the first part of the dissertation, we investigate two training procedures: (i) adversarial training where we determine the perturbation direction based on the given labels for the training data and, (ii) virtual adversarial training where we determine the perturbation direction based only on the output distribution of the training data. We demonstrate the efficacy of adversarial training procedures by performing a k-fold cross validation experiment on the Interactive Emotional Dyadic Motion Capture (IEMOCAP) and a cross-corpus performance analysis on three separate corpora. We compare their performances to that of a model utilizing other regularization schemes such as L1/L2 and graph based manifold regularization scheme. Results show improvement over a purely supervised approach, as well as better generalization capability to cross-corpus settings.

Our second approach leverages multi-modal learning and automated speech recognition (ASR) systems toward improving the generalizability of an emotion recognition model that requires only speech as input. Previous studies have shown that emotion recognition models using only acoustic features do not perform satisfactorily in detecting valence level. Text analysis has been shown to be helpful for sentiment classification. We compared classification accuracies obtained from an audio-only model, a text-only model and a multimodal system leveraging both by performing a cross-validation analysis on IEMOCAP dataset. Confusion matrices show it's the valence level detection thats being improved by incorporating textual information. In the second stage of experiments, we used three ASR application programming interfaces (APIs) to get the transcriptions. We compare the

performances of multi-modal systems using the ASR transcriptions with each other and with that of one using ground truth transcription. This is followed by a cross-corpus study.

In the third part of the study we investigate the generalizability of generative adversarial networks (GANs) based models. GANs have gained a lot of attention from machine learning community due to their ability to learn and mimic an input data distribution. GANs consist of a discriminator and a generator working in tandem playing a min-max game to learn a target underlying data distribution; when fed with data-points sampled from a simpler distribution (like uniform or Gaussian distribution). Once trained, they allow synthetic generation of examples sampled from the target distribution. We investigate the applicability of GANs to get lower dimensional representations from the higher dimensional feature vectors pertinent for emotion recognition. We also investigate their ability to generate synthetic higher dimensional feature vectors using points sampled from a lower dimensional prior. Specifically, we investigate two set ups: (i) when the lower dimensional prior from which synthetic feature vectors are generated is pre-defined, (ii) when the lower dimensional prior is learned from training data. We define the metrics used to measure and analyze the performance of these generative models in different train/test conditions. We perform cross validation analyses followed by a cross-corpus study.

Finally we make an attempt towards understanding the relation between two different sub-problems encompassed under mental state detection namely depression detection and emotion recognition. We propose approaches that can be investigated to build better depression detection models by leveraging our ability to recognize emotions accurately.

TOWARDS BUILDING GENERALIZABLE SPEECH EMOTION RECOGNITION MODELS

by

Saurabh Sahu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Carol Espy-Wilson, Chair/Advisor
Professor Shihab Shamma
Professor Jonathan Simon
Professor Behtash Babadi
Professor William Idsardi (Dean's Representative)

Acknowledgments

First and foremost I would like to express my gratitude towards my advisor Dr. Carol Espy-Wilson. This work wouldn't have been possible without her encouragement, motivation and intuitive inputs. I have learned a lot about speech signal processing as well as about research in general along with presentation and writing skills that has helped me in writing this thesis. I would like to thank my parents and siblings and their families for their constant support and providing me with the confidence in times of need. I would also like to thank Dr. Vikramjit Mitra and Dr. Rahul Gupta for their ideas, discussions and contributions to my thesis. I am grateful to my peers at Speech Communication Lab at UMD Dr. Ganesh Sivaraman, Nadee Seneviratne, Vasudha Kowtha, Rahil Parikh, Nirat Saini and my colleagues from one of the internships Dr. Elizabeth Shriberg and Ben Reeves for interesting discussions and valuable suggestions towards improving my dissertation. I am indebted to my PhD dissertation talk panel members - Dr. Shihab Shamma, Dr. Jonathan Simon, Dr. Behtash Babadi and Dr. William Idsardi for taking time out of their busy schedule and agreeing to be in my panel. I would also like to thank Ravi uncle and his family and my friends who have supported me in times of need and treated me like their own in this foreign land. Finally, I am thankful for Almighty's blessings without which this wouldn't have been possible.

Table of Contents

Chapter 1: Introduction

Human interactions via speech is the most common and efficient way of interaction that occurs on a daily basis. Moreover, their non-invasive nature has also resulted in speech features being popular for various tasks such as emotion recognition. Human listeners have the ability to pick up on the emotions of the speaker they are listening to and they can assess the speaker's mental state. This is an ability that machines lack. Even though we have made great progress in the field of speech recognition, speech emotion recognition systems are still not that accurate. Furthermore, such models that have been trained with few amounts of data, don't perform well when evaluated under unseen test conditions. If we want to achieve our goal of building a realistic human-computer interaction system, it is imperative that the machines understand the emotional state of a person and generalize their performance across unseen speakers and environments so that the conversation is natural.

Speech emotion recognition systems can help us extract useful semantics from speech thereby improving the performance of speech recognition systems (El Ayadi et al. [30], Nicholson et al. [88]). Apart from that, automatic recognition of a speaker's emotional/mental state from speech can help us build systems that can play a crucial role in early diagnosis of psychiatric diseases (France et al. [36]). It can be used in designing web tutorials whose response depend on the users emotion and can also be useful for pilot stress management if implemented in cockpits of aircrafts or even in cars or trucks (Schuller et al. [107], Hansen et al. [52]). Speech emotion recognition has also found applications in call centers (Jin et al. [61]).

The classification of emotions has been researched from two fundamental viewpoints: one, that emotions are discrete and fundamentally different constructs; or two, that emotions can be characterized on a dimensional basis in groupings. A typical set of emotion classes may contain contain up to 300 emotional states (El Ayadi et al. [30], Schubiger [104], O'Connor [90]). Obviously, classifying such a large number of emotions is very difficult and impractical. Hence, to counter this problem, researchers came up with the idea that emotions, like colors can be decomposed into primary components (El Ayadi et al.[30]). The primary emotions are Anger, Disgust, Fear, Joy, Sadness, and Surprise. They are called the archetypal or categorical emotions (El Ayadi et al. [30], Cowie et al. [25]). Moreover, there has also been research on classifying emotions by scoring them along certain pre-defined dimensions. There are several models having their own definition of dimensions that reflect the affect state of an individual. Valence-Arousal-Dominance model has been used predominantly in studies (Valstar et al. [120], Grimm et al. [45]). Each of these dimensions can take a continuous value from low to high. We can quantify them as lying in the set $[-1, 1]$. The valence dimension measures how pleasurable an emotion is. For instance, 'anger' and 'fear' are low valence emotions while 'happy' is a high valence emotion. The arousal scale measures the intensity of an emotion. For example, both 'anger' and 'rage' are low valence emotions but 'rage' has a higher arousal state. Similarly, we can say 'boredom' is a low arousal emotion along with being a low valence emotion. The dominance scale represents the dominant/controlling nature of emotions. While both anger and fear are low valence emotions, 'anger' has a higher dominance value than 'fear'. In this thesis we focus on categorical emotion classification.

Speech emotion recognition is a quite challenging task for several reasons. One of the main reasons is that it is unclear as to which speech features are most powerful in distinguishing between emotions (El Ayadi et al. [30]). Further, the acoustic variability introduced by different sentences, speakers, speaking styles, speaking rates and record-

ing conditions adds another obstacle because these properties directly affect most of the common extracted speech features such as pitch, energy contours, Mel frequency cepstral coefficients (MFCC) etc. Thus, designing speech recognition systems typically requires extraction of a considerably large dimensionality of features to reliably capture the emotional traits, followed by training of a machine learning system ideally with a huge amount of data so that it performs well in unseen conditions.

As this problem has a lot of practical applications it is not surprising that research in the field of speech emotion recognition has been going on for quite some time (Williams and Stevens [124]). The degree of naturalness of a dataset is an important factor to be considered while designing a speech emotion recognizer (El Ayadi et al. [30]). A list of common databases used for speech emotion/depression research has been listed in (El Ayadi et al. [30], Ververidis and Kotropoulos [122]). An important concern is whether we should focus on real world emotions or acted emotions. Acted ones tend to be more exaggerated but acoustic correlates found with acted emotions do not contradict those found with real ones (Williams and Stevens [124]) . Some acted corpora play out real life scenarios to elicit realistic emotions (Busso et al. [13]). Below we describe some features and classification schemes that have been used in various studies.

1.1 Features

While using features, researchers always ponder on the question of global vs local features. Global features are one per utterance and hence its convenient for cross validation. However, we lose the temporal information contained in speech. It could also be unreasonable to try and train a complex classifier like SVM or HMM with global features since the number of training vectors might not be enough (El Ayadi et al. [30]).

1.1.1 Continuous features

Prosodic continuous speech features such as frequency and energy have been investigated in a lot of studies involving mental state detection (El Ayadi et al. [30], Cummins et al. [26]). Continuous features are related to F0, energy, articulation rate and spectral information in various regions. They can be grouped into the following categories:

- pitch-related features
- formants features
- energy related features
- timing features
- articulation rate features

Continuous speech features have been heavily used in speech emotion recognition. For example, Banse et al. examined vocal cues for 14 emotion categories [7]. Prosodic features like F0, energy, etc have been used in various studies like Cowie et al. [25], Williams and Stevens [124], Murray and Arnott [85], Oster and Risberg [91]. While these features are quite reliable and used predominantly for emotion detection, there have been contradictory reports for some of them. To get a global value of these features, researchers use functionals like mean, median, range, standard deviation, quartile ranges etc (Eyben et al [35]).

1.1.2 Voice quality features

Another class of features commonly used is voice quality features. Voice quality and perceived emotion in particular full blown emotions that make people take direct actions are strongly related (Cowie et al. [25]). The same study categorizes the acoustic correlates related to voice quality into following categories

- voice level (amplitude, energy)
- voice pitch
- temporal structures

Voice quality features can be obtained from glottal signals which can be obtained from speech by filtering out the effects of the vocal tract. Since only voiced signals are generated form voiced signals, it is imperative that we do a voiced segment detection before using this method. However, non-uniform vocal fold behavior, presence of noise, formant ripples are some big problems that researchers have to overcome while inverse filtering. Researchers have used the method mentioned in (El Ayadi et al. [30]) for voice quality feature extraction. Pitch and the first four formant frequencies and bandwidth are estimated from the speech signal. The effect of the vocal tract is mitigated by subtracting the vocal tract's influence from harmonic amplitudes. Source features which estimate the air flow from the lungs through glottis are an effective way to capture voice qualities (Cummins et al. [26]). Voice qualities features can quantify irregular phonation relating to laryngeal qualities such as breathiness, creakiness or harshness (Klatt and Klatt [66], Gobl et al. [40]). Other source features include jitter, shimmer and harmonic to noise ratio (HNR). However, like prosodic features there is disagreement between researchers on how to associate vocal quality features with emotions (El Ayadi et al. [30]). For example, according to Scherer [103], tense voice is associated with anger,joy and fear and lax voice is associated with sadness. On the other hand, Murray and Arnott [85] were of the opinion that breathy voice is associated with both anger and happiness while sadness is associated with a resonant voice quality.

1.1.3 Spectral features

Spectral features characterize the spectrum of speech i.e. the frequency domain representation of a speech signal at a particular time instant in some high dimensional space.

Studies have indicated that different emotions affect the sub-band energy distribution in speech (El Ayadi et al. [30]. Some depression papers have reported a relative shift in energies from lower to higher sub-bands while others have focused on sub-band energy variability (Cummins et al. [26]). Cepstral features which are obtained from spectral features have been used extensively for emotion recognition. While Bou-Ghazale and Hansen [12] report that cepstral features like MFCC, linear predictor cepstral coefficients (LPCC) outperform linear based features like LPC, Nwe et al. [89] report that linear based feature like the log frequency power coefficient worked better than MFCC and LPCC for their experiments. Features like power spectral density, linear prediction coefficients, Mel frequency cepstral coefficients, spectral centroid, spectral flux are some examples of spectral features. One major drawback of using these kind of features is that spectral features contain both linguistic and paralinguistic information. They also contain speaker dependent information and hence MFCCs has been widely used in speech and speaker recognition. Ideally we would like our features for this task to be independent of any linguistic information. So features of this sort might hinder speaker emotion recognition systems. Clearly there are some relationships among the feature types described above. For example, spectral variables relate to voice quality, and the pitch contours relate to the patterns arising from different tones. But links are rarely made in the literature (El Ayadi et al. [30].

1.1.4 Teager energy operator

Teager energy operator based features characterize the non-linear airflow in the vocal system (Teager and Teager [116]). Under stressful conditions, the muscle tension of the speaker affects the air flow in the vocal system producing the sound (El Ayadi et al. [30]. Therefore, nonlinear speech features could be useful for detecting emotions or a speaker's mental state.

1.2 Classifiers for emotion recognition

Classifiers like hidden Markov models (HMM), Gaussian mixture models (GMM), artificial neural networks (ANN) and Support vector machines (SVM) have been used for the task of emotion recognition (El Ayadi et al. [30]). Furthermore, k-nearest neighbor, fuzzy classifiers, decision trees and systems where multiple classifiers are combined have been employed by researchers in the past.

1.2.1 Hidden Markov Models

HMMs have been widely used for automatic speech recognition (ASR). However, HMMs used for emotion recognition are usually fully connected unlike the HMMs used for ASR which are left to right. Also the output of HMM states for emotion recognition are feature values extracted from larger time units spanning one or multiple words since it doesn't make sense to label smaller units like a phoneme with an emotion category. While designing an HMM system, some design criteria are number of optimal states, type of observations (discrete or continuous) and the optimal number of observation symbols/optimum number of Gaussians. Nwe et al. [89] showed that a four state HMM model does better than humans in recognizing emotion for Burmese and Mandarin databases. However, we can't generalize the results until a more comprehensive study is done.

1.2.2 Gaussian Mixture Models

GMMs are probabilistic models used to model multi-modal distributions. Determining the number of Gaussians is an important design problem and methods such as classification error with respect to a cross-validation set, minimum description length, Akaike information criterion, kurtosis based goodness of fit measures and greedy expectation-

maximization have been employed for this task (El Ayadi et al. [30]) .GMMs are trained using global features and thus they don't have the ability to model the temporal dynamics of training data. In order to do that GMMs were employed with vector auto-regressive process resulting in Gaussian mixture vector auto-regressive models. El Ayadi et al. [31] showed that GMM performed better than methods like HMM, k-nearest neighbors and ANN.

1.2.3 Artificial neural networks

ANNs have a better ability to model non-linear mappings as compared to HMMs and GMMs. They also perform better than GMM or HMM when the number of training examples is low. The design criteria for ANN include the number of hidden layers, number of neurons in each hidden layer and the activation function. The performance of an ANN depends heavily on these parameters. Hence, researchers have tried using the aggregate decision of multiple ANN architectures. The ANN can be one-class-in-one network classifier where we have one neural network for each emotion giving binary output values indicating the presence/absence of the emotion. The final decision is based on the output of all the neural networks. In contrast we can also have all-emotion-in-one neural network architectures by having a softmax output layer. ANNs have performed less well compared to GMM/HMM and how good they perform is thought to be dependent on the corpus used for the study (El Ayadi et al. [30]). However, due to more computing power being available these days, Deep Neural Networks (DNNs) have become quite popular. DNNs are simply ANNs with multiple hidden layers. Stuhlsatz et al. [113] report some impressive accuracies using a DNN on 9 corpora using Generalized Discriminant Analysis features to do a binary classification between positive and negative arousal and positive and negative valence states.

1.2.4 Support Vector Machines

SVMs use kernels to map the non-linear feature space to a high dimensional space where they are linearly separable. SVMs can be used for multi-class classification by training one SVM per emotion to give a binary decision and then combining the decisions from all the SVMs. The design criteria include the type of kernel and the cost parameter C (Chang and Lin [18]). A table summarizing the performances of these classifiers is presented in (El Ayadi et al. [30]). HMM and GMM are the more popular choice of classifiers for emotion recognition. HMM has the ability to model the transitions between states thereby capturing the temporal dynamics of how features change. But they need proper initialization schemes for their training and parameter estimation. Models like ANN and SVM have been employed widely as well because of their ease of implementation. Even though the training time for an SVM is larger compared to a GMM or an HMM we used it for our classification purposes because unlike an HMM/GMM we don't need to initialize any parameters.

1.3 Objectives of this study

Performance of speech emotion recognition classifiers deteriorate when evaluated on a dataset different than the training set. Below we show the mean class-wise accuracies when we train and evaluate a neural network based classifier on two datasets named IEMOCAP (Busso et al. [13]) and MSP-IMPROV (Busso et al. [16]). We consider four way classification between emotions angry, sad, neutral and angry. As can be seen in Table 1.1 cross-corpus accuracies are always lower than in-domain speaker independent cross-validation accuracies. The model trained on MSP-IMPROV under-performs than that of the model trained on IEMOCAP probably because of its lower size. At the same time we can see that

Table 1.1: Mean class-wise accuracies obtained for in-domain speaker independent cross-validation and cross-corpus evaluation

$\rightarrow$ *Training* $\downarrow$ *Evaluation*	IEMOCAP	MSP-IMPROV
IEMOCAP	**58.15**	47.04
MSP-IMPROV	43.43	**49.56**

the cross-corpus differences are more evident when using a model trained on IEMOCAP.

The objective of this thesis is to investigate the generalizability of speech emotion recognition systems trained on limited amount of data. Towards that end we use existing feature extraction schemes to train classifiers and carry out in-domain speaker independent cross-validation studies which gives us an idea how well our models perform on unseen speakers. Additionally, we perform cross-corpus experiments to determine how different recording conditions/label space/annotation and data collection procedures affect the models. We investigate the generalizability of discriminative and generative models. Discriminative models learn a conditional distribution $p(y_i|\mathbf{x_i})$, given a set of feature vectors $\mathbf{x_i}$ and the corresponding labels $y_i, i = 1..N$. In other words given the training data-points they aim to learn the hard/soft boundary between classes. Generative models on the other hand model the distribution of the classes. They aim to learn the joint distribution $p(\mathbf{x_i}, y_i)$. Once the joint distribution has been computed, it can be used to evaluate the $p(y|\mathbf{x})$ in order to classify a new data-point $\mathbf{x}$. Also by sampling a point from the joint distribution it is possible to generate synthetic examples of data-points $\mathbf{x}$. If our goal is to build classification models and we have enough labeled data available for training, then discriminative models are the way to go. However, the annotation process can be expensive and time consuming. Generative models can be helpful in such cases when limited labeled training data are available since they can also exploit vast amount of unlabeled data. However, in often cases the generalization performance of generative models is found to be poorer than discriminative models due to differences between the model and the true distribution of

the data (Bernardo et al [11]) which is what we aim to investigate in one of the chapters. Specific contributions of this thesis are as follows:

1. Investigating adversarial/manifold learning based regularization schemes and comparing them with L1/L2 regularization that have been known to prevent overfitting in discriminative models (chapter 3).

2. Leveraging the generalizability of automatic speech recognition systems to get the transcripts from audio files and using them to build multi-view speech emotion recognition models (chapter 4). We also study the effect of using attention mechanisms in discriminative models that uses frame-wise features for speech emotion recognition.

3. Studying the capability of generative adversarial networks (GANs) to learn the probability distribution of the feature vectors used for speech emotion recognition (chapter 5). We perform experiments to determine how well these models can encode higher dimensional feature vectors to lower dimensions and then use the trained GAN based models to generate synthetic higher dimensional feature vectors. The synthetic feature vectors can potentially be used to train models in low resource conditions. To our knowledge, this is the first such study comparing the GAN based models with regards to their encoding ability and comparing the quality of the generated synthetic feature vectors for speech emotion recognition.

4. Finally we talk about future directions. We show some results on how a speech emotion recognition model trained on acted datasets performs on real world samples. We also propose some experiments that can potentially improve mental heath detection using information gained from emotion recognition models (chapter 6).

Chapter 2: Literature Survey

A generalizable speech emotion recognition system should perform well under various conditions. Shami and Verhelst [109] showed that aggregating data from different emotion datasets to train a model can improve the performance as compared to when the model is trained on just one dataset. More recently, Zhang et al [133] performed a cross-corpus binary arousal and valence level classification across six databases to explore the effectiveness of unsupervised learning across six emotion databases. The databases they used corresponded to different languages such as German, English and Danish. They used the Opensmile toolkit [35] to extract a 6552 dimensional feature set that consists of 39 functionals of 56 low level descriptors and their delta and delta-delta coefficients. This high dimensional feature vector was used to train a SVM with linear kernel. They performed a 6-way leave one corpus out cross-validation experiments and reported the mean class-wise accuracies. They experimented with three different normalization techniques namely mean subtraction, min-max normalization and mean-variance normalization performed before and after aggregating the datasets. They reported higher accuracies with mean centering for valence level detection while mean-variance normalization worked the best for arousal level detection. Also normalizing the features before aggregating was found to be more beneficial. Their next step was to investigate a unsupervised adaptation technique. They considered three settings (i) they used only three out of five datasets for training resulting in ten training set permutations for each of the six test sets. (ii) they trained a model on three datasets and used it to predict the labels for remaining two training sets. They then consid-

ered these predicted labels as ground truth labels and used all five sets for training a model which was evaluated on the test set. This was the unsupervised framework where availability of unlabeled data was assumed (iii) they used data and ground truth labels from all five sets for training. As expected the performance of unsupervised method lied somewhere in between case (i) and case (iii). The absolute improvement in the mean accuracy over six cross-validation splits was 0.4% for arousal detection and 0.8% for valence detection.

Abdelwahan and Busso [2] proposed feature selection based domain adaptation technique for a 4-class (angry, sad, neutral and happy) speech emotion recognition problem. Domain adaptation is closely related to transfer learning in which our aim is to learn and fit a model on a source data distribution that performs 'well' on a different but related target data-distribution. The authors' goal here was to improve the performance of a SVM classification technique trained on IEMOCAP dataset [13] and tested on MSP-IMPROV [16]. In most cases as in this paper, different emotion datasets are considered to have different (due to speaker, channel, recording conditions, label space) but related (since all of them have emotionally colored utterances) distributions. They extracted 6373 dimensional feature sets using Opensmile which was reduced to 3000 dimensions by selecting features that correlate with class label of the training set but not with each other. They also train an ensemble of SVM classifiers with each targeted towards classifying a particular emotion in particular maximizing the f2-score obtained for that class. They then try three different methods to select data from target set to label and use them for feature selection for each of the SVM classifiers in the ensemble. They data selection methods include (i) vote entropy where they select samples in target set which have the highest disagreement over the ensemble of classifiers (ii) uncertainty sampling where the samples closest to the decision margin of a SVM trained on source data are considered. Note that in this case only one SVM classifier is used (iii) random sampling. Once the labeled set from target data is obtained, feature selection was done sequentially by adding features one at a time to the selection set for each

of the classifiers. The selection is done on the basis of highest f2-score obtained when the SVM trained on source data is evaluated on the labeled target set. They showed that vote entropy works the best when target training set is small whereas random sampling works better for larger sample size. They also showed that implementing this feature selection technique more often than not performs better than a baseline model trained on the features selected using forward feature selection technique.

Abdelwahab and Busso [1] proposed using supervised model adaptation techniques to obtain better cross-corpus performances. It can also be viewed as a domain adaptation technique where the classifier trained on a source data is adapted to perform better on a target data. They used two different English databases for training (source data) and a smaller French database for evaluation (target data) of their classifiers across high/low arousal and valence detection. To reduce the effect of data-differences they mean-variance normalized the features obtained for utterances from different datasets using their corresponding dataset specific statistics. To mitigate the differences in their label spaces they mean-variance normalized the scores obtained from different datasets individually. Samples with normalized value below -0.3 were considered negative or low arousal/valence while those with normalized value above 0.3 were considered positive or high arousal/valence. Baseline classifiers were SVM trained only on one of the two source datasets. They explored two adaptation techniques (i) adaptive SVM which tries to transform the decision boundary of the already trained SVM classifier such that it classifies the labeled target data correctly without changing the decision boundary by a large amount. (ii) incremental SVM where along with newer target domain training data, a portion of the older source data was used for training. Specifically, only those datapoints from source domain were retained that corresponded to the support vectors of the already trained SVM. This process was repeated iteratively. They also considered using different amounts of target data to adapt the SVM parameters. They found that using 35% of target training set for adaptation is as good as

using all of the target training set. Furthermore, speaker diversity also didn't seem to matter much while selecting the data used for adaptation from target training set. Both adaptation schemes were found to perform similarly out-performing a non-adapted model.

Sanchez et al. [102] also applied domain adaptation techniques for improving cross-corpus accuracies. They used two datasets in their experiments namely a 911 call dataset and LDC dataset (Liberman et al. [74]) for a binary classification task between neutral and fear classes. Their metric was f-measure score obtained for fearful calls. The 911 call had 95 calls and hence they reported the average 95 fold leave one call out cross-validation f-scores. The calls were broken down into segments after prosodic features were extracted. They were used for training and evaluation of a SVM classifier. They compared training only using the 911 data and found that it always performs similar or worse than a model trained using 911+LDC. They tried another approach where they trained a classifier only on LDC and used the prediction probabilities obtained for 911 as additional features along with the prosodic features. Then they trained a model using this appended feature set obtained for 911. This model was found to perform better than both the above methods. They also employed a method called nuisance attribute projection (NAP) to compensate for the differences in channel and utterance length duration between the two datasets. The goal of this method was to find a projection matrix P to project the original features to a space more robust to nuisances arising from domain differences. While the f-measure obtained using a model trained only on 911 dataset was found to be 64.1% the best performance was achieved by training a model on NAP 911+LDC features with the f-measure being 64.8%. In [112], Song et al. used a similar projection matrix based approach to enhance the cross-corpus performance of speech emotion recognition models. They proposed a joint framework to learn a common projection matrix W to project the feature representations X onto label space or embedding space Y for the source and target datasets. Moreover, they also attempt to minimize the differences in Y that appear due to differences in X belonging

to different datasets. Hence, they add a maximum mean discrepancy (MMD) regularization term that tries to minimize the difference between the mean embeddings of features in source and target domains. At the same time, they also implement feature selection by minimizing the $l_{2,1}$ norm of W. Generally, we can say that whichever row of W has the least l_2 norm, that corresponding dimension of the feature vector has the least contribution towards generating the embeddings. Additionally the also add a graph based regularization term that minimizes the distance between the embeddings if the corresponding features lie are close to each other in the feature space. The resulting objective function does not have a closed form solution due to the presence of $l_{2,1}$ norm and so an iterative algorithm is implemented to get the matrix W and the resulting label space embeddings for target dataset. Compared to a baseline linear classifier that directly learns the matrix W from source without any of the regularization terms, this method showed an absolute improvement of 17-18% in cross-corpus recognition accuracies and other popular projection matrix based methods. Liu et al. [76] followed a very similar projection based matrix approach to get the embeddings in label space except they did not have the graph based regularization term in their objective function. The two studies used the same databases for their cross-corpus evaluations and the better accuracies obtained using Song et al's method shows the worth of the graph based regularization term.

Now we focus on methods that instead of projection matrices, use auto-encoders (Baldi [6]) to learn common feature representations for different data sets. Deng et al. [28] explored a supervised sparse auto-encoder based feature learning scheme. Their task was valence level detection (high/low) in cross-corpus training scenarios. While the target dataset was kept fixed, they used five different databases as source dataset. There were variations between target and source datasets due to participant ages, languages and recording conditions. They trained a single hidden layer auto-encoder to minimize the reconstruction error when the feature vectors were fed to them as input. Moreover, they enforced a sparsity

constraint penalizing when the expected activations of the hidden layer exceeds a low fixed level. Hence, these auto-encoders were termed as sparse auto-encoders. If the hidden layer dimension is less than that of input layer dimension then the auto-encoder learns a sparse low dimensional representation of the input feature vectors. The transfer feature learning procedure starts with randomly choosing a certain number of instances from the target set belonging to a particular class and training a class specific sparse auto-encoder to minimize its reconstruction error. Then instances belonging to the same class are selected from the source database and reconstructed using the trained auto-encoder. It is this reconstructed vectors that are now used as features to train a SVM classifier. The authors experimented with different numbers of instances chosen from target set. They showed that even with only 50 target instances, on an average the classifiers trained on reconstructed source feature sets exceed the performance of classifiers trained on actual source feature sets by 9% across the different source datasets. Note that this was a supervised feature transfer learning method because class information from target dataset was utilized to train the sparse auto-encoders. Deng et al. [27] extended it to an unsupervised setting as well. They used a single layer denoising auto-encoder (DAE). While training denoising auto-encoders, the input feature vectors are first corrupted by either adding Gaussian noise or masking certain dimensions. Then the corrupted version is fed to auto-encoder while the output's reconstruction error is minimized with respect to the clean input. Since the auto-encoder in this case is made to reconstruct clean input from its noisy version, it learns more robust and pertinent representations of the input. To avoid over-fitting a weight decay regularization term was also added. The authors also implemented an adaptive DAE (A-DAE). The first step towards training an adaptive DAE is to train a DAE to reconstruct the features in target set. Once the weights of this target-DAE are learned, a new DAE is trained on source set but with a modified objective function. The objective function now not only reduces the reconstruction error for feature vectors belonging to source set, but it also forces the parameters

of this new DAE to be closer in values to the parameters of the target-DAE. The importance of these two terms in the objective function can be controlled by hyper-parameter tuning. Finally the source and target datasets are encoded using the adaptive DAE and are used to train and evaluate SVM classifiers. The authors showed that the A-DAE approach performs better than DAE and some other popular feature transformation methods. Mao et al. [80] implemented a shared hidden layer auto-encoder (SHLA) based scheme for domain adaptation. A SHLA is an auto-encoder with a common input and hidden layer but with separate output layers for source and target dataset. It is trained to reconstruct data from both source and target domains. While the hidden to output layer weights are updated based on the reconstruction errors obtained on the corresponding dataset, the input to hidden layer weight matrix is updated in both cases. The authors considered a binary classification task between high and low valence levels with different databases used for training and testing. A feed-forward neural network classifier with one hidden layer was implemented for this purpose. Similar to a SHLA, there were two output layers one for the source training set and one for target training set. Since it was binary classification problem, each output layer had two neurons corresponding to high and low valence classes. The input to hidden layer weights of the classifier were initialized with the input to hidden layer weight matrix of the SHLA. The hidden layer to output layer weights were initialized in such a way that the priors between the related classes are shared i.e. the weight vectors from hidden layer to 'high valence' neurons for both the source and target domains followed a Gaussian distribution with identical mean vectors and covariance matrices. Similarly, the weight vectors from hidden layer to 'low valence' neurons for both the source and target domains followed a Gaussian distribution with identical mean vectors and covariance matrices but different from that of 'high valence' neurons. So, we can say that priors between related classes cross different datasets are shared. The authors showed that in general the shared prior technique led to better accuracies compared to when the priors weren't shared.

Abdelwahab and Busso [3] trained what they call as domain adversarial neural network (DANN) to improve the cross-corpus recognition accuracies of speech emotion recognition models. Their goal was to learn a common representation between the samples from source and target domain. They train a neural network classifier with two different output layers - one for emotion classification and one for domain classification. The two tasks share some portion of the hidden layers which is used to get the aforementioned common representation for samples from both domains. The loss function includes a term that reduces the prediction loss obtained for source data for which we have ground truth labels available. It includes another term that updates the shared hidden layer parameters by reversing the sign of gradients obtained from domain classification error term. We can leverage both source and unlabeled target data to get the domain classification loss. Hence its an unsupervised domain adaptation method. Both loss terms compete against each other in an adversarial manner. The gradient reversal attempts to make the features similar across domains, so that feature transformation information learned from source domain is retained for target domain. The t-distributed stochastic neighbour embedding (TSNE) plots show that the feature transformations obtained from the last layer of the shared hidden layers from a trained DANN for source and target domains are more indistinguishable compared to that obtained from a baseline vanilla deep neural network (DNN). They also showed better performance of DANN models for arousal, valence and dominance value prediction compared to a vanilla DNN. The relative improvements over the baseline models are 22.8% for arousal, 33.4% for valence and 15.5% for dominance. Since, they formulated it as a regression problem, the metric used was concordance correlation coefficient (CCC) between ground truth and predicted values. CCC is defined so that it combines the idea of two more well known metrics - root mean square error (RMSE) and Pearsons correlation coefficient.

Neumann and Vu [86] investigated the importance of pre-training on source dataset followed by fine-tuning on a few training samples from target set. They attempted a binary

arousal and valence level classification using two datasets namely IEMOCAP (English) and RECOLA (French). The features used for this experiment was 26 dimensional logMel filterbank coefficients extracted at a frame rate of 10ms for 7.5 second long utterances. They applied a 1D convolution over time followed by a max-pooling layer, output from which was used to compute attention weights over time. The attention coefficients were then used to compute a weighted sum of the information obtained from different parts of the input. It was concatenated with the feature maps obtained after max-pooling layer and was fed to softmax layer to get the classification prediction probabilities. They consider 4 training scenarios : (i) within corpus speaker independent cross-validation (ii) multi-lingual cross-validation by combining the data in IEMOCAP and RECOLA and using them for training and validation (iii) cross-lingual (CL) where they trained on one corpus and tested on another (iv) CL followed by fine-tuning on a smaller target training set (CL+FT). The mean class-wise accuracies for both arousal and valence detection showed an increase in CL+FT as compared to CL showing leveraging information from few target training samples can be beneficial in cross-corpus experiments. As we have seen from previous studies this was expected. The in-domain cross-validation studies in general performed better than CL and CL+FT showing cross-domain differences can still matter even with fine-tuning. Kim et al. [62] used a different transfer learning approach namely multi-task learning where the knowledge gained from auxiliary tasks was used for improving the performance on the main task. Their main task was emotion detection while the auxiliary tasks were gender detection and naturalness (acted or natural) detection for samples in database. They used six corpora in their experiments. Features such as F0, MFCC coefficients, voicing probability were extracted after normalizing the gain of the utterance. They first performed in-domain cross-validation studies on each dataset separately using long short-term memory (LSTM) and DNN architectures both implemented under (i) a single task learning (STL) framework where they recognized only emotions (ii) a MTL framework with shared hidden layers but

different output layers for each of the tasks. While DNN-MTL did not provide any improvement over DNN-STL, LSTM-MTL improved the average of mean class-wise accuracies over the six cross-validation experiments by 1.7%. They then performed cross-corpus experiments using utterances from five of the corpora for training and one for testing. The MTL framework showed more significant improvement compared to STL framework in his scenario. They achieved an average absolute gain of 7.4% and 5.4% when comparing the MTL with STL frameworks in case of DNN and LSTM respectively. They concluded that MTL is potentially more effective for larger corpora. Furthermore, the TSNE of high level features showed better clustering according to emotions in case of MTL than STL frameworks.

As mentioned above most of the research done to improve cross-corpus generalizability of speech emotion recognition models involve domain adaptation techniques that try to learn a data representation that is nuisance free across source and target datasets. The model adaptation methods we discussed used SVM classifier with some of them leveraging the labels of the target domain data. In this thesis, we focus our effort on training discriminative neural network based classifiers without leveraging any information from target domain data. Furthermore, none of the past works have explored the generalization of generative methods. One of the aims of this thesis is to try and close that gap.

Chapter 3: Smoothing model predictions using adversarial examples

3.1 Introduction

A speech emotion recognition system design involves extracting cues from speech and depicting them as feature representations. This is followed by training a classification algorithm using existing supervised/semi-supervised methods (Busso et al. [14], Koelstra et al. [68]). We consider a setting with N training examples $\{\mathbf{x_i}, y_i\}, i = 1, .., N$, where $\mathbf{x_i}$ is the obtained feature representation for example i and y_i is the corresponding label. Let $\mathbf{x}$ and y denote the random variables of which $\mathbf{x_i}$ and y_i are instances. A typical supervised learning approach involves modeling the probability $p(y|\mathbf{x})$ using a chosen functional form (e.g. neural networks or a support vector machine classifier). For the chosen model (trained on finite training data) to generalize well to unseen data, the probability $p(y|\mathbf{x})$ is desired to have certain properties (Chapelle et al. [19]). One such property is the smoothness of the distribution $p(y|\mathbf{x})$ which states that if two points $\mathbf{x_i}$ and $\mathbf{x_j}$ are close to each other in the feature space (based on some distance metric), then so should be their corresponding model outputs $p(y_i|\mathbf{x_i})$ and $p(y_j|\mathbf{x_j})$. The underlying idea is for the classifier to be generalizable and not overfit to training data. Enforcing this smoothness can be particularly useful for low resource tasks such as emotion recognition, where collecting a large number of labeled data instances may not always be possible.

Szegedy et al. [114] showed that neural network models are vulnerable to something called adversarial examples. These are examples only slightly different from the training

examples but the trained model fails in recognizing their class correctly. Clearly, this is a roadblock towards building generalizable models. Goodfellow et al. [44] suggested an improved method called adversarial training (AT) in which the perturbations are added along a specific adversarial direction. The adversarial direction for a certain training data point is the direction along which the label probability of the model for that data point is most sensitive. Miyato et al. [**?**] proposed an extension of adversarial training, termed Virtual Adversarial Training (VAT) wherein determining the adversarial direction doesn't depend on the availability of labels. Both of these methods are implemented by adding an extra regularization term to the vanilla loss term. We refer to the training methods proposed by Goodfellow et al. [44] and Miyato et al. [**?**] as adversarial training procedures, and investigate their applicability for improving the performance of emotion recognition systems.

Similarly, manifold regularization methods impose this smoothness by modifying the optimization objective (Belkin et al. [10]). Manifold regularization exploits the distribution $p(\mathbf{x})$ as available through a set of labeled/unlabeled points to better estimate $p(y|\mathbf{x})$, thereby leveraging the concept of manifold learning to enforce model smoothness. In the past, researchers have investigated manifold learning methods for speech-based emotion recognition. Most of these methods attempt to learn the manifold by reducing the dimensionality of the input feature space and subsequently feeding them to a classifier. For example, Kim et al. [64] and Ping et al. [94] employed isometric feature embedding for deriving the manifolds and then used Gaussian Mixture Models as classifiers. You et al. [129] employed Lipschitz embedding for non-linear manifold learning in an unsupervised way followed by using support vector machines for classification. Qian et al. [96] applied a supervised manifold learning method by considering the difference between feature subsets of different classes and reported improvement in recognition accuracy. However, none of them have investigated manifold regularization techniques that jointly optimize a manifold

regularization loss along with supervised classification loss. In particular, jointly optimizing the two losses has shown promise with deep neural networks (DNNs) for improving ASR (Tomar and Rose [117]) and sentiment classification (Zhou et al. [134]). Researchers have proposed several manifold regularization techniques, starting from Belkin et al. [10] and Geng et al. [37]. These methods make use of available labeled/unlabeled data points for regularization for better performance of classification models.

In this paper we compare the performance of AT and VAT to that of a baseline DNN model for emotion recognition. After training the model using the aforementioned procedures, we evaluate its performance under two settings: (i) Running a cross validation experiment on a single corpora (ii) Doing a cross corpora study. Under the single corpora setting, we aim to understand the impact of adversarial training on system performance under matched conditions. In the cross corpora setting, we train the model on a single dataset and evaluate performance on three separate unseen datasets. We also compare the adversarial training procedures with other regularization schemes. Along with the widely known L1/L2, we also investigate the effect of the graph based manifold regularization scheme discussed in Tomar and Rose [117]. We hypothesize that since manifold regularization imposes smoothness constraint on the model's outputs for the data points that are in the neighborhood of each other, it can also make the model more robust to noise arising due to difference in data distributions. In the following sections, we provide a background of the adversarial training procedures (AT and VAT) and other regularization schemes that we have employed. This is followed by a detailed explanation of the experiments after which we present our conclusions.

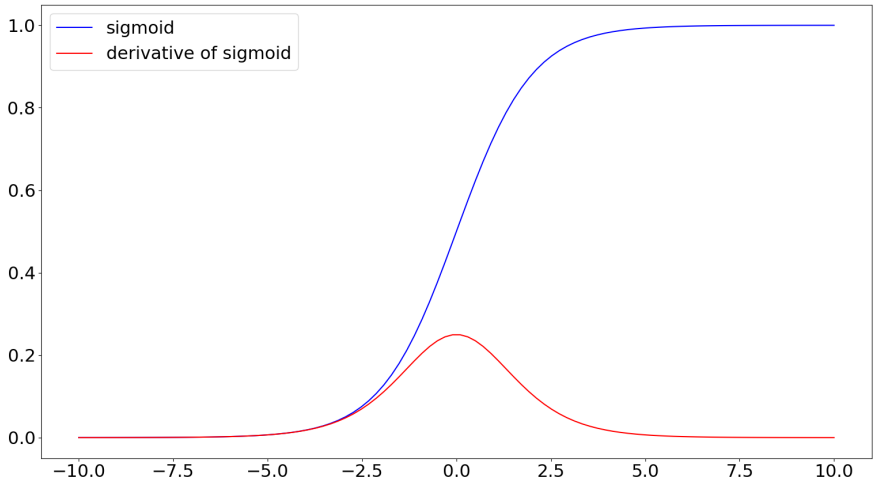

Figure 3.1: Figure representing the sigmoid activation function and its derivative

3.2 Understanding adversarial examples

Even though deep neural networks are highly non-linear functions, Goodfellow et al. [44] suggest that the existence of adversarial examples can be explained by considering simple linear models. They argue that complex neural network models such as LSTMs or feed-forward networks with activation functions such as sigmoid, ReLUs (Rectified Linear Units) or maxout (Goodfellow et al. [43]) are kept in their linear region of operation for efficient and easier optimization. For example, the derivative of sigmoid function gets saturated in its non-linear region, thereby not providing enough gradients for the network to learn as shown in the figure 3.1. However it might come as a surprise that these models have very different outputs for a small deviation in the input. But that is usually the case when we work in high dimensional spaces. Consider a linear model with weights $\mathbf{w}$ and input $\mathbf{x}$. Let $\tilde{\mathbf{x}} = \mathbf{x} + \eta$ be the adversarial input with η being the adversarial error term added to input. Assume that $\|\eta\|_\infty \leq \varepsilon$ where ε is a small quantity. So, the activation of the

linear model for the perturbed input is given by $\mathbf{w}^T\tilde{\mathbf{x}} = \mathbf{w}^T\mathbf{x} + \mathbf{w}^T\eta$. Assuming $\mathbf{w} \in \mathbb{R}^n$ and the average absolute value of the elements in $\mathbf{w}$ is m, $\mathbf{w}^T\eta \leq \varepsilon mn$. Hence, even though the input changes by a small amount ε the output of the model changes by an amount of the order of εmn which can be a substantial quantity when working in higher dimensions.

One way of mitigating the effect of adversarial examples is to find such examples and train the model with them. Let us consider a binary classification problem with ground truth labels $y = \{-1, 1\}$. We train a model that estimates the probability for a given datapoint $\mathbf{x}$ using the equation : $P(y=1) = \sigma(\mathbf{w}^T\mathbf{x} + b)$, where σ is the sigmoid function. To estimate the parameters $\mathbf{w}$, loss function $L(\mathbf{x}, y)$ is minimized over the training data. Since the log function is monotonic, formally $L(\mathbf{x}, y)$ can be defined as:

$$
\begin{aligned}
L(\mathbf{x}, y) &= \begin{cases} -log(\sigma(\mathbf{w}^T\mathbf{x} + b)), & \text{if } y = 1 \\ -log(1 - \sigma(\mathbf{w}^T\mathbf{x} + b)), & \text{if } y = -1 \end{cases} && (3.1) \\
&= \begin{cases} log(1 + e^{-(\mathbf{w}^T\mathbf{x} + b)}), & \text{if } y = 1 \\ log(1 + e^{(\mathbf{w}^T\mathbf{x} + b)}), & \text{if } y = -1 \end{cases} && (3.2) \\
&= log(1 + e^{-y(\mathbf{w}^T\mathbf{x} + b)}) && (3.3) \\
&= T(-y(\mathbf{w}^T\mathbf{x} + b)) && (3.4)
\end{aligned}
$$

where, $T(z) = log(1 + e^z)$. To derive the perturbation term η constrained by $\|\eta\|_\infty \leq \varepsilon$ for the above loss function, we can follow the fast gradient sign method as mentioned in [44]. Basically it states:

$$
\begin{aligned}
\eta &= \varepsilon sign(\nabla_{\mathbf{x}} L(\mathbf{x}, y)) && (3.5) \\
&= \varepsilon sign\left(-\left(\frac{e^{-y(\mathbf{w}^T\mathbf{x} + b)}}{1 + e^{-y(\mathbf{w}^T\mathbf{x} + b)}} \right) y\mathbf{w} \right) && (3.6) \\
&= -\varepsilon sign(y) sign(\mathbf{w}) && (3.7)
\end{aligned}
$$

Since, $y \cdot sign(y) = |y| = 1$ and $(\mathbf{w}^T sign(\mathbf{w})) = \|\mathbf{w}\|_1$, the loss term $L(\tilde{\mathbf{x}}, y)$ for the perturbed input $\tilde{\mathbf{x}} = \mathbf{x} + \eta$ can be written as:

$$
\begin{aligned}
L(\tilde{\mathbf{x}}, y) &= T(-y(\mathbf{w}^T \tilde{\mathbf{x}} + b)) && (3.8)\\
&= T(-y(\mathbf{w}^T \mathbf{x} + b) - y(\mathbf{w}^T \eta)) && (3.9)\\
&= T(-y(\mathbf{w}^T \mathbf{x} + b) + \varepsilon(y \cdot sign(y))(\mathbf{w}^T sign(\mathbf{w}))) && (3.10)\\
&= T(-y(\mathbf{w}^T \mathbf{x} + b) + \varepsilon\|\mathbf{w}\|_1) && (3.11)
\end{aligned}
$$

It can be observed that to some extent minimizing the loss function for adversarial inputs is akin to L1 regularization. However, it is less stringent than L1 regularization penalty because in this case as the model starts making confident predictions, $y(\mathbf{w}^T \mathbf{x} + b)$ attains a high value thereby making the function $T(z)$ enter its saturation region. This in turn does not provide enough gradients for $\mathbf{w}$ to change as can be seen in Figure 3.2. Hence the effect of term $\varepsilon\|\mathbf{w}\|_1$ disappears in this case unlike vanilla L1 regularization.

An alternative way to counter the effect of adversarial examples is via adding a regularization term to the overall loss function as has been done in Goodfellow et al. [44] and Miyato et al. [**?**]. This is the approach that we have followed for our task of speech emotion recognition. In the following section, we explain the loss functions for a baseline neural network without any regularization and the loss terms obtained after employing the various regularization techniques.

3.3 Delving into loss functions

Let $\{\mathbf{x_i}, y_i\}, i = 1, .., N$ be the set of N labeled data points that will be used to train a neural network model. We represent the parameters of the neural network by θ. Then the

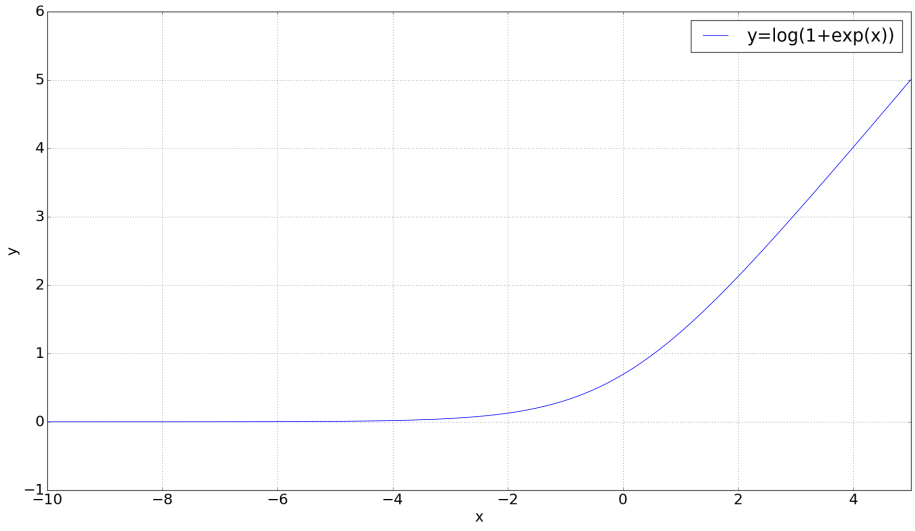

Figure 3.2: $T(x) = log(1 + e^x)$. Note how the graph saturates when $x \rightarrow -\infty$

output for the point $\mathbf{x_i}$ is given by $\theta(\mathbf{x_i})$. $\theta(\mathbf{x_i})$ is a vector of probabilities that the neural network assigns to each class in the label space spanned by y. It is computed using softmax activation. A loss function is defined based on the neural network outputs and the one hot-vectors $\mathbf{y_i}$ corresponding to labels y_i, as shown below. $V(\theta(\mathbf{x_i}), \mathbf{y_i})$ is the loss for the data point $\mathbf{x_i}$ and ground truth label $\mathbf{y_i}$. Choices for the loss function include cross-entropy (usually for classification tasks), mean squared error (usually for regression tasks) or the hinge loss. We choose cross-entropy as the loss function for our baseline neural network.

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} V(\theta(\mathbf{x_i}), \mathbf{y_i}) \tag{3.12}$$

With a small number of training instances N, its hard to generalize the performance of a model trained solely on the loss above. The trained model performs well on training set but not so much on unseen data suggesting overfitting. Studies have used L1 or L2 regularizers on neural network parameters or dropout to prevent overfitting on the training

set (Tripathi and Jadeja [118]). Manifold learning and smoothing is another way to prevent overfitting and build models that generalize better. Along with the methods mentioned above, we also implemented a graph based manifold regularization technique that leverages the smoothness assumption that was mentioned in Section 3.1 Another approach is to add a regularization term that penalizes large differences in model outputs when a small perturbation is added to a data point. This mitigates the effect of adversarial examples on a trained model. We determine the perturbation based on two existing methods: (i) adversarial training and, (ii) virtual adversarial training. A brief overview of these methods are given below.

3.3.1 L1/L2 regularization

L1/L2 regularization techniques belong to a larger class of parameter norm based penalties which have been used for regularization for quite some time (Goodfellow et al. [41]). Denoting training data points as $\mathbf{x_i}$, ground truth labels as y_i and the parameters of the model as Θ, the modified loss function consists of the supervised loss function V and a regularization term $f(\Theta)$.

$$\mathscr{L}' = \frac{1}{N}\sum_{i=1}^{N} V(\theta(\mathbf{x_i}), \mathbf{y_i}) + \alpha f(\Theta) \tag{3.13}$$

The hyperparameter α controls the weight given to the regularization term. For a neural network based model, usually only the weights (denoted by $\mathbf{W}$) are modified and the biases (denoted by b) are unaffected by the regularization term. Weights involve the dynamics between two variables while biases control a single variable. Hence, fitting biases accurately requires less data than fitting weights and hence we would not gain much by regularizing the biases.

For L2 regularization $f(\Theta)$ is chosen as squared L2 norm. L2 norm for a vector is

defined as the square root of the sum of squares of its components. So, the loss function becomes

$$\mathscr{L}' = \frac{1}{N}\sum_{i=1}^{N} V(\theta(\mathbf{x_i}), \mathbf{y_i}) + \alpha\|\mathbf{W}\|_2^2 \tag{3.14}$$

The weights $\mathbf{W}$ are updated by taking the derivative of the loss function with respect to the weight vector and implementing stochastic gradient descent. L2 regularization doesn't have much affect on the components of the weight vector that have a larger impact on the objective function. On the contrary, the components of the weight vector which do not affect the gradient undergo decay much faster. This leads to mitigating any training noise induced along those components and prevent overfitting.

For L1 regularization $f(\Theta)$ is chosen as L1 norm. L1 norm for a vector is defined as the sum of the absolute value of its components. So, the loss function becomes

$$\mathscr{L}' = \frac{1}{N}\sum_{i=1}^{N} V(\theta(\mathbf{x_i}), \mathbf{y_i}) + \alpha\|\mathbf{W}\|_1 \tag{3.15}$$

Unlike L2 regularization, L1 regularization makes the model more more generalizable by inducing sparsity in the parameter space. A technique called 'least Absolute shrinkage and selection operator' or LASSO leverages this property and removes less important feature's coefficients to zero thereby doing feature selection. While L2 regularization is the same as Bayesian MAP (Maximum a posteriori) inference with Gaussian prior on weights, L1 regularization is the same as MAP with Laplacian prior on weights.

3.3.2 Adversarial training

Adversarial training (Goodfellow et al. [44], Miyato et al. [**?**]) modifies the loss function in such a way that it penalizes large deviations in model outputs when small perturbations are added to the training data points $\mathbf{x_i}$. A perturbation vector $\mathbf{r_i}^a$ is determined

for every datapoint $\mathbf{x_i}$ followed by optimizing the modified loss $\mathscr{L}_{\text{adv}}$ to train a neural network, as shown in Equation 3.16. D is a non-negative function that quantifies the distance between the predictions $\theta(\mathbf{x_i} + \mathbf{r_i}^a)$ and targets $\mathbf{y_i}$. D is usually chosen to be cross entropy or Kullback-Leibler divergence. α is a tunable hyper-parameter, determining the trade-off between $\mathscr{L}$ and the adversarial loss.

$$\mathscr{L}_{\text{adv}} = \mathscr{L} + \alpha \times \frac{1}{N} \sum_{i=1}^{N} D(\mathbf{y_i}, \theta(\mathbf{x_i} + \mathbf{r_i}^a)) \tag{3.16}$$

The perturbation $\mathbf{r_i}^a$ is determined based on Equation 3.17. The hyper-parameter ε determines the search neighborhood for $\mathbf{r_i}^a$.

$$\mathbf{r_i}^a = \arg \max_{\mathbf{r}:\|\mathbf{r}\| \leq \varepsilon} D(\mathbf{y_i}, \theta(\mathbf{x_i} + \mathbf{r})) \tag{3.17}$$

Considering $||r||$ to be the Euclidean norm, $\mathbf{r_i}^a$ in Equation 3.17 can be approximated as shown below.

$$\mathbf{r_{adv}} \approx \varepsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2}, \textit{where } \mathbf{g} = \nabla_{\mathbf{x_i}} D(\mathbf{y_i}, \theta(\mathbf{x_i})) \tag{3.18}$$

If $||r||$ is considered to be the infinity norm, then $\mathbf{r_i}^a$ is computed using Equation 3.5. The gradient term in both the equations is obtained by differentiating the baseline loss function with respect to the input. It can be easily computed during back-propagation. It can be observed that the regularization term added to the baseline loss function in adversarial training depends on the ground truth labels. Hence, it can be considered as a supervised regularization scheme unlike the other methods discussed here. We note that this optimization has two hyper-parameters to tune, α and ε. We investigate the impact of these hyper-parameters on the model performance in one of our experiments.

3.3.3 Virtual Adversarial training

Virtual adversarial training (Miyato et al. [**?**]) also modifies the loss function in such a way that it penalizes large deviations in model outputs for small perturbations in the input. A perturbation vector $\mathbf{r_i}^v$ is determined for every datapoint $\mathbf{x_i}$ followed by optimizing the modified loss $\mathscr{L}_{\text{adv}}$ to train a neural network, as shown in Equation 3.19.

$$\mathscr{L}_{\text{adv}} = \mathscr{L} + \alpha \times \frac{1}{N} \sum_{i=1}^{N} D(\theta(\mathbf{x_i}), \theta(\mathbf{x_i} + \mathbf{r_i}^v)) \tag{3.19}$$

The perturbation $\mathbf{r_i}^v$ is determined based on Equation 3.20. The hyper-parameter ε determines the search neighborhood for $\mathbf{r_i}^v$.

$$\mathbf{r_i}^v = \arg\max_{\mathbf{r}:\|\mathbf{r}\|\leq\varepsilon} D(\theta(\mathbf{x_i}), \theta(\mathbf{x_i} + \mathbf{r})) = \arg\max_{\mathbf{r}:\|\mathbf{r}\|\leq\varepsilon} \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta}) \tag{3.20}$$

where $\hat{\theta}$ is the current estimate of model parameters. Assuming D is KL divergence, we can observe that $\mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta}) = 0$ for $r = 0$. Hence we can't find an expression for $\mathbf{r_i}^v$ as we did for $\mathbf{r_i}^a$ using Equation 3.18. Since the minimum value a KL divergence can attain is 0, the derivative $\nabla_r \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta}) = 0$ at $r = 0$. Using these along with Taylor's approximation we get:

$$\begin{aligned} \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta}) &\approx \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta})|_{r=0} + \mathbf{r}^T \nabla_r \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta})|_{r=0} + \mathbf{r}^T H(\mathbf{x_i}, \hat{\theta}) \mathbf{r} && (3.21) \\ &= \mathbf{r}^T H(\mathbf{x_i}, \hat{\theta}) \mathbf{r} && (3.22) \end{aligned}$$

where $H(\mathbf{x_i}, \hat{\theta}) = \nabla \nabla_r \mathbb{D}(\mathbf{x_i}, \mathbf{r}, \hat{\theta})|_{r=0}$. Assuming we have a symmetric Hessian matrix $H(\mathbf{x_i}, \hat{\theta})$ (which would be the case if $\mathbb{D}$ is twice differentiable at $r = 0$) would imply its unit length eigenvectors $\mathbf{e_i}$ (associated with the i-th biggest eigenvalue λ_i) are orthogonal.

Therefore, any unit vector $\mathbf{r}$ can be expressed as sum of these basis vectors i.e.

$$\mathbf{r} = \sum_{i=1}^{K} \alpha_i \mathbf{e_i} \quad such\ that \quad \sum_{i=1}^{K} \alpha_i^2 = 1 \tag{3.23}$$

Hence, for $\|\mathbf{r}\| = 1$

$$\begin{aligned} \mathbf{r}^T H(\mathbf{x_i}, \hat{\theta})\mathbf{r} &= \sum_{i=1}^{K} \alpha_i^2 \mathbf{e_i}^T H(\mathbf{x_i}, \hat{\theta})\mathbf{e_i} && (3.24) \\ &= \sum_{i=1}^{K} \alpha_i^2 \lambda_i \leq \sum_{i=1}^{K} \alpha_i^2 \lambda_1 = \lambda_1 && (3.25) \end{aligned}$$

And the maximum is obtained when $\mathbf{r}$ is the dominant eigenvector $\mathbf{e_1}$. The perturbation term can therefore be computed as:

$$\mathbf{r_i}^v = \arg \max_{\mathbf{r}:\|\mathbf{r}\| \leq \varepsilon} \mathbf{r}^T H(\mathbf{x_i}, \hat{\theta})\mathbf{r} = \varepsilon \cdot \mathbf{e}_1(\mathbf{x_i}, \hat{\theta}) \tag{3.26}$$

The dominant eigenvector for $H(\mathbf{x_i}, \hat{\theta})$ can be computed by initializing a randomly sampled unit vector $\tilde{\mathbf{r}}_0$ and using the power iteration method.

$$\tilde{\mathbf{r}}_{m+1} = \frac{H\tilde{\mathbf{r}}_m}{\|H\tilde{\mathbf{r}}_m\|} \tag{3.27}$$

The power iteration method will converge as long as the random initialization isn't orthogonal to the dominant eigenvector $\mathbf{e_1}$ and the rate of convergence would depend on the ratios $\frac{\lambda_k}{\lambda_1}$ for $k \neq 1$. Expressing $\tilde{\mathbf{r}}_0$ as mentioned in Equation 3.23 we can work out the convergence

of power iteration method.

$$\begin{aligned}
\mathbf{r_1} &= H\tilde{\mathbf{r}}_0 && (3.28)\\
&= H(\sum_{i=1}^{K} \alpha_i \mathbf{e_i}) && (3.29)\\
&= \sum_{i=1}^{K} \alpha_i H\mathbf{e_i} && (3.30)\\
&= \sum_{i=1}^{K} \alpha_i \lambda_i \mathbf{e_i} && (3.31)
\end{aligned}$$

Pre-multiplying H to both sides of the above equation m times, where m is large we get,

$$\begin{aligned}
\mathbf{r_m} &= \sum_{i=1}^{K} \alpha_i {\lambda_i}^m \mathbf{e_i} && (3.32)\\
&= {\lambda_1}^m [\alpha_1 \mathbf{e_1} + \sum_{i=2}^{K} \alpha_i \Big(\frac{\lambda_k}{\lambda_1}\Big)^m \mathbf{e_i}] && (3.33)\\
&\approx {\lambda_1}^m \alpha_1 \mathbf{e_1} && (3.34)
\end{aligned}$$

Equation 3.34 follows from the fact that $|\lambda_1| < |\lambda_2| < ... < |\lambda_k|$ and therefore the ratios $\rightarrow 0$ as $m \rightarrow 0$. Hence, power iteration method converges to a vector lying along the dominant eigenvector direction. The number of iterations m is a hyper-parameter for VAT. We didn't see any major differences in the model's performance for different values of m and so it was fixed at $m = 1$. In VAT, it can be seen that the adversarial perturbation term depends on the model parameters θ. While updating the model parameters using backpropagation, we do not take into account the gradient flow from the perturbation term. Further details about the algorithm to compute $\mathbf{r_i}^v$ can be obtained from (Miyato et al. [**?**]).

Equation 3.20 is very similar to Equation 3.17 except instead of ground truth labels $\mathbf{y_i}$ we use the "virtual" labels $\theta(\mathbf{x_i})$ which are probabilistic estimates obtained from a neural network model. Since the regularization loss term is independent of ground truth labels,

it can be used in semi-supervised training scenarios where the first term $\mathscr{L}$ is computed using labeled data and the second term is computed using both labeled and unlabeled data.

3.3.4 Graph based manifold regularization

We also try a graph based manifold regularization scheme that penalizes the model for producing very different outputs for input datapoints within a certain neighborhood. However unlike using KL divergence as in case of VAT, we consider Euclidean distance in this case. Manifold regularization was proposed by Belkin et al. [10] and similar to above methods we add a regularization penalty term to the cross entropy loss function. Let us consider training datapoints $\mathbf{x_i}(i = 1,..,N)$, with corresponding labels y_i. For a choice of Reproducible Kernel Hilbert Space (RHKS) $\mathscr{H}_k$ and a loss function V, they optimize equation 3.35 to yield a classifier function f^* belonging to the space $\mathscr{H}_k$. In the equation, $V(\mathbf{x_i}, y_i, f)$ is considered to be cross-entropy loss function as ours is a classification problem. $||f||_I^2$ is a regularization term modifying the parameters of the classifier depending on the distribution of the set of given data points in the training set (please refer to Section 2 in Belkin et al. [10] for more details). γ_I is the hyper-parameter controlling the trade-off between the losses in the equation 3.35.

$$f^* = \arg\min_{f \in \mathscr{H}_k} \frac{1}{N} \sum_{i=1}^{N} V(\mathbf{x_i}, y_i, f) + \gamma_I ||f||_I^2 \tag{3.35}$$

$||f||_I^2$ is computed as shown in equation 3.36.

$$||f||_I^2 = \sum_{i=1}^{N} \sum_{\substack{\mathbf{x_i}^u \in \\ \text{Neighborhood of } \mathbf{x_i}}} \frac{||f(\mathbf{x_i}) - f(\mathbf{x_i}^u)||_2^2}{||\mathbf{x_i} - \mathbf{x_i}^u||_2} \tag{3.36}$$

The loss minimizes the Euclidean distance between the outputs for labeled instance $\mathbf{x_i}$:$f(\mathbf{x_i})$ and a set of data-points in the neighborhood of $\mathbf{x_i}$:$f(\mathbf{x_i}^u)$. Neighborhood $\mathbf{x_i}^u$ for

any point $\mathbf{x_i}$ is defined as the set of points lying within a L2 norm ball centered at $\mathbf{x_i}$. The distance between the outputs is inversely weighted by the distance between $\mathbf{x_i}$ and $\mathbf{x_i}^u$, so that the loss function weights the distance $||f(\mathbf{x_i}) - f(\mathbf{x_i}^u)||_2^2$ more when $\mathbf{x_i}^u$ is closer to $\mathbf{x_i}$ when considering Euclidean distance. For fast computation, we compute the loss term in Equation 3.35 iteratively, updating the weights based on cross-entropy loss first and then updating them based on the regularization loss. A similar approach was followed in Gupta et al. [48] where the authors explored semi-supervised learning on twitter sentiment dataset using doc2vec features. Since, the regularization term is independent of ground truth labels, it can be used in a semi-supervised setting like VAT.

3.4 Comparison of various generalization schemes

We perform experimental investigations under two settings: (i) a single corpora setting using a cross validation setup and, (ii) a cross corpora setting involving training on one corpus and testing on the other. In the single corpora setting, we aim to test improvements in the generalized performance of the model under matched dataset conditions. However, in the case of cross-corpora evaluation, representations for emotional utterances tend to be dissimilar due to factors such as differences in data collection protocol and noise conditions. Through cross-corpora evaluation, we aim to investigate if manifold regularization can yield models robust to the corpus specific variations.

3.4.1 Single corpora setting

We use the Interactive Emotional Dyadic Motion Capture (IEMOCAP) dataset (Busso et al. [13]) for our single corpora evaluation. The dataset consists of five sessions of scripted and improvised interactions between two actors acting out real world situations. No two sessions have the same set of actors, enabling us to do a speaker independent leave-one-

session-out five-fold cross validation. The database comes with the dyadic conversation segmented into utterances which are on an average about 5 seconds in duration. The utterances were then labeled by three annotators for emotion labels such as happy, sad, angry, excitement and neutral. We only use utterances for which we obtain a majority vote regarding the ground truth label. Following the work of Kim and Provost [65], we combine the utterances in the happy and excited classes to get a "combined happy" class for our experiments. This was done to obtain a more balanced dataset, given there are only a small number of "happy" class instances. For our classification experiments we focused on a set of 5531 utterances shared amongst four emotional labels: neutral (1708), angry (1103), sad (1084), and happy (1636). Overall, this amounts to approximately 7 hours of data.

3.4.2 Cross corpus evaluation

We use a set of four datasets for the cross corpora evaluation. We train a DNN on the IEMOCAP dataset to identify four classes of emotion, followed by predictions on these datasets.

Surrey Audio Visual Expressed Emotion (SAVEE) database: Surrey Audio-Visual Expressed Emotion (SAVEE) database (Haq et al. [53]) has recordings of four male speakers reciting IEEE sentences in seven different emotions. For the purpose of our evaluation, we only select the subset of utterances belonging to one of the four target emotions, as predicted by the model trained on the IEMOCAP dataset. The dataset consists of 60 utterances each belonging to the angry, sad, happy classes and 120 neutral utterances. We acknowledge that transfer of models across corpora spanning different label spaces is a challenge. By selecting a subset of utterances in our experiments, we simulate a study that assumes that the two datasets span the same label space.

Electromagnetic Articulography (EMA) database: Electromagnetic Articulography (EMA) database (Lee et al. [72]) contains a set of 680 utterances spoken in four different target emotions, such as anger, happiness, sadness and neutrality. Speakers are native speakers of American English: two females and one male. Note that the label space spanned by this dataset is equivalent to the one spanned by utterances in the training set.

Linguistic Data Consortium's (LDC) emotional prosody dataset: This database (Liberman et al. [74]) was developed by LDC and contains the recordings of professional actors reading a series of semantically neutral utterances (dates and numbers) spanning fourteen distinct emotional categories. We select a subset of 714 utterances from the dataset that span the four emotion labels as modeled using training on the IEMOCAP dataset.

MSP-IMPROV dataset: MSP-IMPROV (Busso et al. [16]) has actors participating in dyadic conversations across six sessions and like IEMOCAP they also have been segmented into utterances. But unlike IEMOCAP, it also includes a set of pre-defined 20 target sentences that are spoken with different emotions depending on the context of conversation. There are 7798 utterances belonging to the same four emotion classes. The class distribution is unbalanced with the number of utterances belonging to happy/neutral class more than three times that of angry/sad.

We note that there are several dissimilarities between the IEMOCAP dataset and the datasets used in the cross corpora study. Whereas the speakers in EMA and LDC have an American accent, SAVEE has speakers having a British accent. Unlike IEMOCAP, these databases aren't dyadic conversations. While EMA and SAVEE have speakers speaking different sentences emulating different emotions, in the LDC database we have speakers reading out numbers while emulating different emotions. While MSP-IMPROV is more similar to IMEOCAP than others in terms of how it was collected, the data distribution in both is very different. We next discuss the features extracted on these datasets.

3.4.3 Features

We use the openSMILE toolkit to extract 1582 dimensional feature vector (Eyben et al. [35]). This feature set consists of various functionals computed for spectral, prosody and energy based features. The same feature set has also been used in several previous works including the INTERSPEECH Paralinguistic Challenges (Schuller et al. [108]). Similar sets of spectral, prosodic and energy based features have shown considerable success in emotion classification and affect tracking (Gupta et al. [47]). However an increased feature count leads to the "curse of dimensionality", a problem that manifold learning and smoothing can mitigate.

3.4.4 Experimental setup

We use a DNN as our classification model, such that the output layer consists of four nodes (each corresponding to an emotion), with softmax activation function. The DNN has three hidden layers with the number of neurons in each layer set to 64. The objective function $V(\theta(\mathbf{x_i}), \mathbf{y_i})$ is chosen to be the cross entropy loss in our experiments (Goodfellow et al. [41]). We performed L1/L2 regularizations and compared their performance with the other regularization techniques mentioned above. While performing AT, we chose the D function to be the cross entropy between $\mathbf{y_i}$ and $\theta(\mathbf{x_i} + \mathbf{r_i^a})$, while in the case of VAT, D is set to be the cross entropy between $\theta(\mathbf{x_i})$ and $\theta(\mathbf{x_i} + \mathbf{r_i^v})$. Miyato et al. [**?**] considered two different distance functions D for VAT training: (i) Kullback-Leibler divergence between $\theta(\mathbf{x_i})$ and $\theta(\mathbf{x_i} + \mathbf{r_i^v})$. (ii) cross entropy between $\theta(\mathbf{x_i})$ and $\theta(\mathbf{x_i} + \mathbf{r_i^v})$. We also experimented with the Kullback-Leibler divergence as the distance function D, without observing significant differences in the model performances. We also replaced the adversarial error terms $\mathbf{r_i^a}$ and $\mathbf{r_i^v}$ with a random error term and analyzed if adding perturbations along targeted directions rather than random has any advantage. While performing graph based regularization, we

consider the neighborhood of each point $\mathbf{x_i}$ to be its two nearest neighbors.

We implemented the models in Keras (Chollet et al. [22]) with a Tensorflow backend and performed optimization using stochastic gradient descent (Zhang [132]). Our evaluation metric is Unweighted Accuracies (UWA) which has been used previously in emotion classification tasks (Sahu et al. [101]). Since the distribution of emotion classes are unbalanced in the datasets of interest, the UWA metric assigns equal weight to each emotion class during evaluation. Next, we present further details regarding the single corpus and cross corpus evaluation.

Results: Single corpus setting

We perform a leave one session out cross validation experiment on IEMOCAP. Through this experiment, we aim to understand the impact of the hyper-parameters, mainly the impact of ε and α for adversarial training procedures on the model performance. We first study the impact of regularization factor α as mentioned in Equations 3.14 and 3.15. The plots are shown in Figure 3.3. It was observed that with regularizations, increasing the weight of the regularization term to a higher value decreases model performance. This makes sense because we still want the cross entropy error term $V(\theta(\mathbf{x_i}), \mathbf{y_i})$ to dictate the training of the DNN and not the regularization terms. The optimum performance was obtained for $\alpha = 0.005$ for L1 and $\alpha = 0.05$ for L2 regularizations.

For adversarial training procedures, in order to study their impact individually, we perform evaluation by perturbing one of the two parameters, while keeping the other constant. By altering ε, we aim to understand the impact of smoothing radius around the data-points on the model performance and perturbing α impacts the weight of the adversarial loss on the overall optimization. The plots comparing the UWA of baseline DNN with that of DNN with adversarial training procedures for different values of hyper-parameters is shown in

Figure 3.4. It is evident that DNN trained with adversarial training procedures perform better than the baseline DNN. First, the value of α was kept fixed at 2 and ϵ was varied. For DNN trained with AT regularization term, the model shows a higher performance for lower value of ϵ peaking at $\epsilon = 0.5$. As we increase the value of ϵ, the model's performance starts deteriorating. This is expected since ϵ defines the neighborhood around an input feature vector over which the conditional distribution $p(y|\mathbf{x})$ is smoothed. Increasing the radius of this neighborhood forces our model to learn smoother functions that cannot capture the complexity of the conditional distribution function $p(y|\mathbf{x})$ thereby decreasing its performance on the validation set. For lower values of ϵ, AT outperforms VAT which may be due to the fact that AT is a supervised learning scheme where we use actual labels to find the adversarial direction. However, for higher values of ϵ, the trend reverses which leads us to believe that for larger values of search radius we are better off smoothening the output of the perturbed input with respect to the output of the actual input rather than the label. Similar observations can be made when we compare targeted perturbations versus random perturbation term added to input. For lower values of ϵ, targeted adversarial training procedures are better while for higher values, adding a perturbation in random direction performs better than both the adversarial training procedures; all of which perform worse than baseline DNN with no regularization. Changing the weight α while keeping ϵ fixed at 0.5 did not seem to affect the accuracies of AT very much. For VAT however, increasing the weight of the VAT loss parameter in the loss function decreases the performance of the system. These experiments showed that with the right value of hyper-parameters, using adversarial training procedures that add perturbation along a targeted direction perform better than adding perturbations along a random direction. It was observed that for $\alpha = 2$ and $\epsilon = 0.5$ performance of AT was the best. We also implemented the graph based manifold regularization scheme. The results obtained using the above hyper-parameters for various regularization schemes are mentioned in Table 4.1. The hyper-parameters so

Table 3.1: Unweigthed accuracies obtained for different regularization schemes from the cross-validation experiment

Model	**UWA**
Baseline DNN	58.15
L1 regularization	59.02
L2 regularization	59.21
AT	59.54
VAT	58.17
Graph based manifold regularization	58.35
L2 + AT	**60.33**

obtained have been tuned via the cross-validation scheme. It was seen that AT performs the best compared to other regularization schemes. Implementing it along with L2 regularization further seemed to improve the results by a relative amount of approximately 4%. It is observed that AT performs better than the other regularization schemes where the ground truth label is not taken into account.

We further analyze the posterior probability distribution of the labels given the feature vectors (expressed by $p(y|\mathbf{x})$) by projecting and visualizing the four dimensional output vector $\theta(\mathbf{x_i})$ using t-Stochastic Neighbor Embedding (t-SNE) approach proposed by Maaten and Hinton [78]. t-SNE is a dimension reduction technique that clusters similar vector values together. The four dimensional output is projected to a two dimensional space using t-SNE and plotted in Figure 3.5 for one of the cross-validation sets. The results shown are with the hyperparameter values ε and α values fixed at 0.5 and 2, respectively. We observe that compared to baseline DNN, neural networks trained with adversarial training procedures are better able to distinguish the 'happy' samples. While for the baseline DNN most of the pink 'happy' samples overlap with blue 'neutral' samples, for the other two regularized models especially the one trained with AT, we see a few clusters formed more or less entirely of the pink samples. Analyzing the confusion matrix suggests that this leads to less confusion between utterances belonging to other classes with 'happy' class.

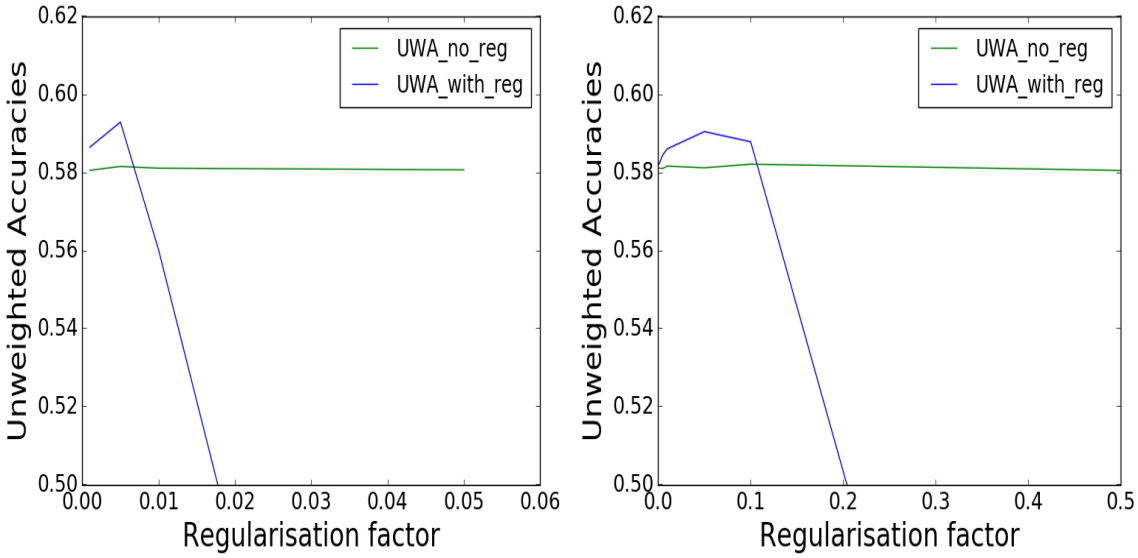

Figure 3.3: Unweighted accuracies vs the hyper-parameter α for baseline DNN (green) and DNN with L1 regularization on left (blue) and with L2 regularization on right (blue).

Table 3.2: Cross-corpus accuracies (%) obtained using baseline DNN and DNN models trained with different regularization schemes. The training was performed using IEMOCAP in all cases.

Test Dataset	Baseline DNN	DNN with L2	DNN with AT	DNN with L2+AT
MSP-IMPROV	43.43	43.57	45.22	**45.37**
SAVEE	47.29	52.29	**53.13**	52.5
EMA	57.77	58.32	**64.51**	64.1
LDC	43.66	43.28	45.64	**45.97**

Results: Cross corpus evaluation

Since the adversarial training procedure make the model robust to small perturbations to the input training points, we hypothesize that the regularized models are also robust to variation across datasets due to dissimilar noise conditions. Hence a model trained on an external corpus can achieve better performance on a dataset of interest. To verify this, we did a cross corpus analysis where the whole of IEMOCAP dataset was used for training and a different corpus was used for testing. We extract the openSMILE features for the four external corpora, followed by mean-variance normalization using in-corpus statistics. We compare the UWA for three datasets as shown in Table 3.2 and show the superior perfor-

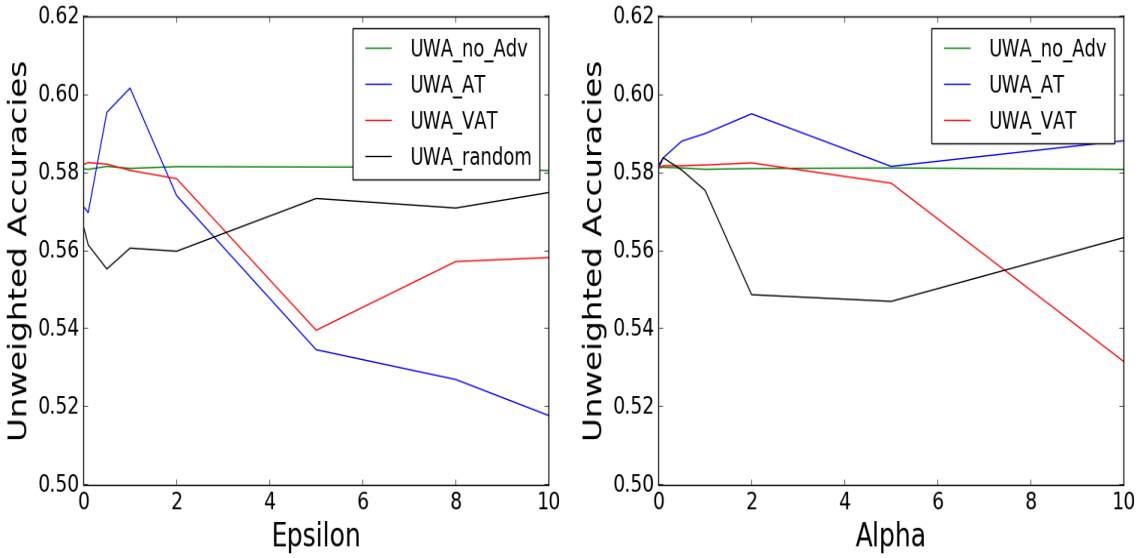

Figure 3.4: Unweighted accuracies vs the hyper-parameters ϵ (left) and α (right)

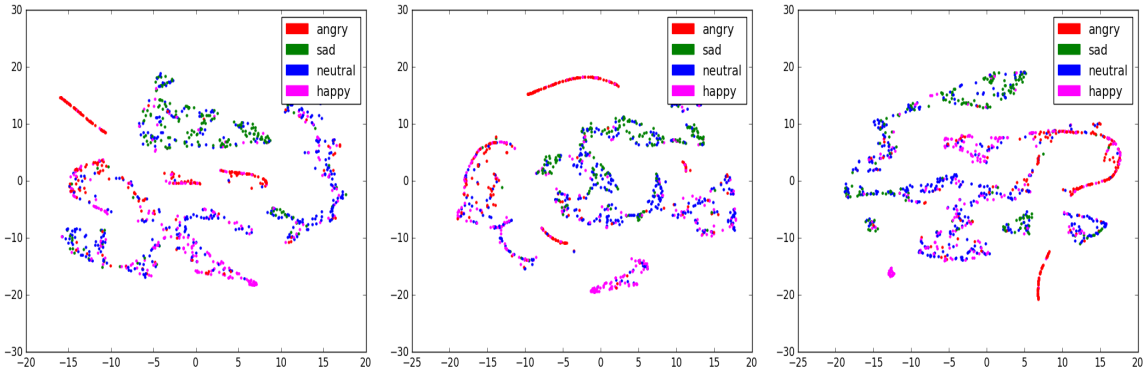

Figure 3.5: TSNE plots comparing the output of the baseline DNN model (left), DNN trained with AT (center) and DNN trained with VAT (right)

mance of models trained with regularization procedures than baseline DNN. This indicates that the adversarial procedures increase model robustness to cross-corpus differences. We also note that the IEMOCAP trained models perform better on EMA compared to the other three datasets. This can be explained by domain variabilities. While SAVEE has British accented speech, in LDC the actors are reading out just numbers instead of actual English sentences. EMA being an American English corpus where participants are reading out sentences, comes closest to IEMOCAP which has actors having conversations in English.

The worst performance on MSP-IMPROV can probably be explained by the fact that this dataset is more realistic compared to SAVEE and EMA that are more extreme in terms of the emotional dialogs. This observation suggests that despite better model generalization across datasets, data specific characteristics still play a part in determining the model performance.

3.5 Conclusion and future work

In this chapter, we show the effectiveness of adversarial training procedures for emotion classification using a DNN model. The regularization schemes enforce the smoothness of the output probabilities $p(y|\mathbf{x})$, a case particularly applicable to low resource tasks such as emotion classification. We perform two sets of evaluation, a single corpus evaluation on the IEMOCAP dataset and four evaluations using a cross-corpus setup. In both the cases, we observe an improvement in the classification performance using adversarial training. Regularization methods such as VAT and graph based manifold learning scheme that do not leverage the ground truth labels do not show significant improvement probably because of the less amount of data available to us while training. We perform further investigation to understand the impact of the model hyper-parameters on the model performance and analyze the model outputs using t-SNE projections of the model outputs.

In the future, we aim to conduct further investigations using the adversarial loss. In particular, the VAT training procedure and the graph based manifold learning scheme can be used for semi-supervised training schemes. This can be performed using an in-domain/external source for unlabeled data. Another interesting area to further investigate would be to study the effects of one regularization scheme on another when multiple regularization schemes are implemented together. As can be seen from the cross-corpus results, using L2 regularization with AT doesn't always give an improvement. We also aim to

investigate other distance metrics D and its impact on the performance. Another pertinent problem is making the cross-corpus study compatible to different output label spaces across the datasets. Finally, one can also test the adversarial methods to other low resource problems.

Chapter 4: Multi-modal learning for Speech Emotion Recognition : An Analysis and comparison of ASR outputs with ground truth transcriptions

4.1 Introduction

Speech is the most common and efficient way of interaction that occurs on a daily basis and its non-invasive nature has also resulted in speech features being popular for various tasks one of them being emotion recognition. It has applications in several fields including building intelligent voice-assistants, psychiatry, analysis of human interaction and other behavioral studies (El Ayadi et al. [30]). Affect recognition or emotion recognition is a well-researched field and the results demonstrate that using speech features does a better job at predicting arousal levels (intensity) than valence (pleasantness) level of the utterance. Valstar et al. [119] employed a support vector machine based regressor and found that the metric concordance correlation coefficient (CCC) is higher for predicting arousal levels than valence. They managed to improve the valence prediction task using information from other modalities such as video and physiological signals. The work by Yang and Hirschberg [127] shows similar results on a couple of databases after extracting features from raw waveform and spectrogram using a convolutional neural network and passing them through a neural network based regressor to get the predicted arousal and valence scores. Li and Akagi [73] employed a fuzzy inference based system and their results show

a lower mean absolute error and a higher CCC in predicting arousal than valence across three different languages. From the results shown by Lotfian and Busso [77] it can be observed that the same is still true even after employing curriculum learning. The work by Kim et al. [63] compared different neural network based systems in classifying between angry, sad, neutral and happy and it was observed that all of them struggled in classifying the 'happy' samples correctly.

These results indicate that audio-based systems can be improved in predicting valence levels by leveraging information from other modalities. Since our aim was to build an emotion recognition model that only uses speech as input and modern state of the art ASR models can generate good transcriptions, we looked at previous works using audio and text features. Metallinou et al. [81] combined audio, video and phoneme level transcripts for multi-modal emotion classification and showed an improvement as compared to a uni-modal classifier. Zadeh et al. [130] used word level acoustic, vision and text features to implement an attention architecture that captures cross-modal dynamics. In the work by Hazarika et al. [54], similar features were input to a deep neural framework that was implemented to capture the dynamics between speakers in a dyadic conversation. However, none of these works have provided an insight of why multi-modal learning helps for emotion classification and what is the contribution from each modality. Furthermore, all of them have used ground truth text transcription for their experiments which can be time-consuming and expensive to obtain. Schuller et al. [105, 106] trained an ASR model on the dataset at hand and used the spoken words along with acoustic features for emotion recognition. However, due to unavailability of ground truth transcriptions they were unable to compare how much is the loss in performance when they use ASR transcriptions instead of ground truth. In this paper we analyze the performance of a multi-modal system employing audio and text features, with the hypothesis that while audio features help us with detecting arousal levels, the text features help us with valence prediction. We also devel-

oped a system that uses transcriptions obtained from different ASR models and compare its performance with that of a system that uses only audio features and a multi-modal system using ground truth transcriptions. In the following sections, we provide a background of the datasets used for our cross-validation and cross-corpus study. Before we carry out the experiments analyzing audio and text modality we explore a couple of different attention mechanisms (Chorowski et al. [23]) in neural network architectures with audio features as input for emotion classification. We wanted to see if they could improve accuracies when we are training with limited data. We then carry out the multi-modal experiments where we use both audio and textual features and show our results and analysis. Finally we present our conclusions and future directions.

4.2 Methodology

In this section we explain the databases, feature sets and classifiers used for our experiments. We then talk about the different ASR models employed to get the transcriptions to be used instead of ground truth transcriptions.

4.2.1 Datasets

IEMOCAP

We use the Interactive Emotional Dyadic Motion Capture (IEMOCAP) dataset (Busso el al. [13]) as one of the datasets in our experiments. The dataset consists of five sessions. In each session, two actors act out scenarios which are either scripted or improvised. No two sessions have the same actor participating in them. This enabled us to perform a five fold leave-one-session out cross-validation analysis on IEMOCAP. The conversations have been segmented into utterances which are then labeled by three annotators for emotions

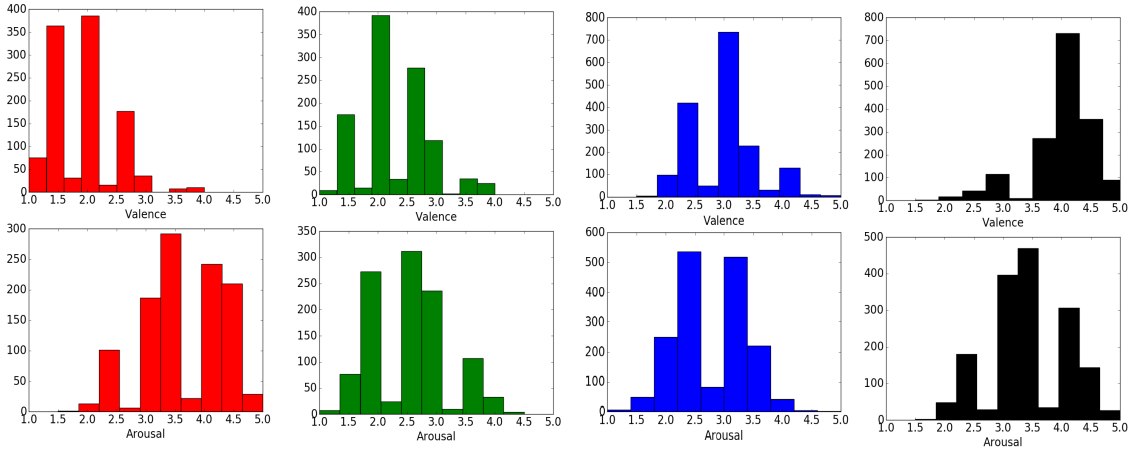

Figure 4.1: Distribution of valence (top) and arousal (bottom) values for utterances in IEMOCAP belonging to classes angry (red), sad (green), neutral (blue) and happy (black) classes

such as happy, sad, angry, excitement and, neutral. Manual transcriptions provided with the dataset are considered as ground truth transcriptions. For our experiments, we only use utterances for which we could obtain a majority vote and assign that as the ground truth label. We used approximately 7 hours of data from the dataset which amounts to 5530 utterances : neutral (1708), angry (1103), sad (1083), and happy (1636). Apart from annotating for categorical emotions, the utterances were also rated on a scale of 1-5 in terms of their arousal and valence; 1 being low arousal/valence and 5 being high arousal/valence. In Figure 4.1 we show a class-wise distribution of arousal and valence values. It can be seen that while 'anger' is low valence and high arousal, 'sad' is low both in terms of valence and arousal. 'Neutral' is more or less symmetrical along the mean for arousal and valence. The emotion 'happy' is high in valence and and has more percentage of utterances with higher arousal than 'neutral' and 'sad' but lesser than that of 'angry'. Following the observations in the work by Neumann and Vu [87] we set the length of utterances as 7.5 seconds. Shorter utterances were pre-padded with zeros while longer ones were clipped.

MSP-IMPROV

MSP-IMPROV (Busso et al. [16]) has actors participating in dyadic conversations across six sessions and like IEMOCAP they also have been segmented into utterances. But unlike IEMOCAP, it also includes a set of pre-defined 20 target sentences that are spoken with different emotions depending on the context of conversation. There are 7798 utterances belonging to the same four emotion classes : neutral (3477), angry (792), sad (885), and happy (2644). The class distribution is unbalanced with the number of utterances belonging to happy/neutral class more than three times that of angry/sad. We didn't have ground truth transcriptions available for this dataset. We used MSP-IMPROV to perform a cross-corpus study where we used it as a test set while IEMOCAP was used as training set.

4.2.2 Feature extraction

We extracted two sets of features for the speech based model and compared their performances. The first set was the Extended Geneva Minimalistic Acoustic Parameter Set (EGeMAPS) extracted using the openSMILE toolkit (Eyben et al. [34]). It is a 23 dimensional feature set consisting of prosodic features like pitch, loudness, jitter, shimmer and spectral parameters. These features were computed for every 20 ms window with a 10 ms overlap. To reduce the computation time, we took the expectation of every ten such consecutive frames so that we have a smoother feature summary vector every 100ms which was then fed to the LSTM. A similar approach was employed in the work by Zadeh et al. [130] to get word level acoustic features from frame level features. The second feature set was computed using the toolkit pyAudioAnalysis (Giannakopoulos [39]). This feature set was also used by Chernykh et al. [21] for speech emotion recognition. The motivation behind using such a feature set is the expectation that it would be more helpful towards building a speaker agnostic emotion recognition model since they don't include prosodic

features. Speaker-based normalization was applied to reduce speaker specific effects using only the neutral speech as proposed by Busso et al [15]. Real world emotion recognition systems usually have access to such samples, so its a fair assumption to make that they can normalize the utterances from test speakers (Le and Provost [69]).

We used 100 dimensional Glove embeddings (Pennington et al. [92]) to initialize the embedding layer of the text based neural network model. The embeddings are computed by essentially factorizing the logarithm of a word-word co-occurrence count matrix obtained from a 2014 dump of Wikipedia (glove.6B). The embedding layer was then fine-tuned for the task at hand by backpropagating the error values obtained from the output layer.

4.2.3 Classification models

We used recurrent cells to compute a sequence of high-level representations from the time-series of feature vectors capturing their contextual information as has been done by Lee and Tashev [71] and Huang and Narayanan [59]. For the audio modality, we had two long short-term memory (LSTM) layers with 256 and 128 hidden units, respectively, followed by a dense layer of 64 neurons with rectified linear unit (ReLU) activation which was connected to the output layer consisting of four neurons with softmax activation. Our text based model had a similar architecture except there was an embedding layer that matched the words with their corresponding Glove vector which was input to the first LSTM layer. For our multi-modal system, the summary feature vectors obtained from the second LSTM layer of the audio modality and the text based model were concatenated to form a 256 dimensional vector. This was followed by a ReLU activated dense layer with 64 neurons and finally the output layer. A recurrent dropout probability of 0.3 was applied to all the recurrent layers in all the models. The hyper-parameters such as the number of recurrent/dense layers, number of recurrent units, batch size, dropout probability etc. were decided based

on the cross-validation study done on IEMOCAP. Since the audio features were computed every 100 ms for a 7.5 second segment, we have 75 time-steps for audio modality. For text-based LSTM, we had 40 time-steps meaning the transcriptions were limited to 40 words if they were longer than that, otherwise zero-padding was applied.

4.2.4 ASR models employed

Our next experiments involved running two free ASR applications to generate the transcriptions and using them in our experiments instead of ground truth transcriptions. This automatically generated transcription enables us to have an emotion recognition model that only requires speech as its input so that we can do away with the manual transcription of the utterances. We used the codes from Zhang's github repository [131] to get the transcriptions by implementing the models from Wit.ai (a Facebook company) and Google. We note that the ASR engines from Google and Wit.ai were not able to generate transcriptions for all the utterances mainly due to troubles with communicating with the API's server. For IEMOCAP, Google and Wit.ai could transcribe 89.9% and 78.3% of the samples, respectively. For MSP-IMPROV, the percentage of utterances for which we could obtain transcriptions using APIs from Google and Wit.ai are 90.55% and 60.24%, respectively. Below we show a few ground truth (GT) annotations and their ASR transcriptions.

1. GT: *You're going to fill out a form on your desk*
   Google: *fill out a form on your desk*
   WIT.ai: *out a form on your desk*

2. GT: *you have to tell me*
   Google: *you have to tell me*
   WIT.ai: *you have to tell me*

3. GT: *Really you don't work for anybody it's just you*

   Google: *really you don't work for anybody it's just you*

   WIT.ai: *really don't work for anybody is up*

4.3 Exploring attention mechanisms

In IEMOCAP and MSP-IMPROV the annotations for emotion are given at an utterance level. However all the frames of an utterance might not be relevant at determining the emotion that it should belong to. Hence, we might need to weigh the frame/window level features appropriately based on their importance. This is what the attention mechanism does. Our goal here was to see if attention mechanisms help us with our experiments where we have limited data. Weighing the frame-wise features might be a good idea but at the same time computing the attention weights leads to an increase in number of parameters which might lead to over-fitting in such cases. We considered only the audio modality for these experiments. The baseline model was as described in section 4.2.3. We incorporated our attention mechanisms between the first and second layer of LSTMs i.e. the sequential output of the first layer of LSTM was weighed by the attention weights before being fed to the second layer of LSTM. Let's denote the output for time-step t obtained from the first LSTM layer as $\mathbf{h}(\mathbf{t}) \in \mathbb{R}^n$, where n is the number of hidden units in the LSTM layer. If we denote the attention weight for time t by $\alpha(t) \in \mathbb{R}$, then the output of the attention mechanism is denoted by $\mathbf{c}(\mathbf{t}) = \alpha(t)\mathbf{h}(\mathbf{t})$ We investigated two ways to compute the attention weights as described below:

1. att_1: a scalar attention weight $\alpha(t) \in \mathbb{R}$ computed based on the root mean square energy (RMSE) of the audio signal. For a segment of an audio signal, its RMSE is defines as the square root of the sum of squares of the samples occurring in that segment. Let $e(t) \in \mathbb{R}$ be the RMSE value obtained for the t-th window of an audio

signal. To make sure it has the same temporal resolution as the time-series of feature vectors being fed to LSTMs, the RMSE was computed for a window of 200 ms with a frame interval of 100 ms. The attention weight for time t was then computed using the following equation:

$$\alpha(t) = \frac{e(t)}{\sum_{t'=1}^{T} e(t')} \tag{4.1}$$

So, $\mathbf{h}(\mathbf{t})$ from segments with lower RMSE will have lower contribution towards making the final decision than the ones with higher RMSE. Note that employing this attention mechanism doesn't lead to any increase in the number of trainable parameter

2. att_2: a scalar attention weight $\alpha_t \in \mathbb{R}$ computed from the output of the first LSTM layer $\mathbf{h}(\mathbf{t})$. This is similar to the way the attention weights were computed using equation 9 in Huang and Narayanan [59] and also in Neumann and Vu [87]. We define a weight layer $\mathbf{w} \in \mathbb{R}^n$ which was shared across the time-steps. The attention weight for time t was then computed using the following equation:

$$\alpha(t) = \frac{\mathbf{w}^T \mathbf{h}(\mathbf{t})}{\sum_{t'=1}^{T} \mathbf{w}^T \mathbf{h}(\mathbf{t}')} \tag{4.2}$$

Note that in this case the number of trainable parameters increase by a value of n.

In Figure4.2, we plot the raw waveform and the attention waveforms obtained using the two mechanisms along with the energy waveform obtained from $\mathbf{h}(\mathbf{t})$. Note that as $\mathbf{h}(\mathbf{t}) \in \mathbb{R}^n$, we simply squared and summed its different dimensions to get the energy waveform for $\mathbf{h}(\mathbf{t})$. We can see that while the profile of attention time-series att_1 computed from RMSE $e(t)$ follows the intensity of signal, att_2 follows the envelope of the energy waveform obtained from $\mathbf{h}(\mathbf{t})$. This is expected behavior given how they have been computed. So, to get the best of the both worlds our final attention att was considered to be a weighted sum

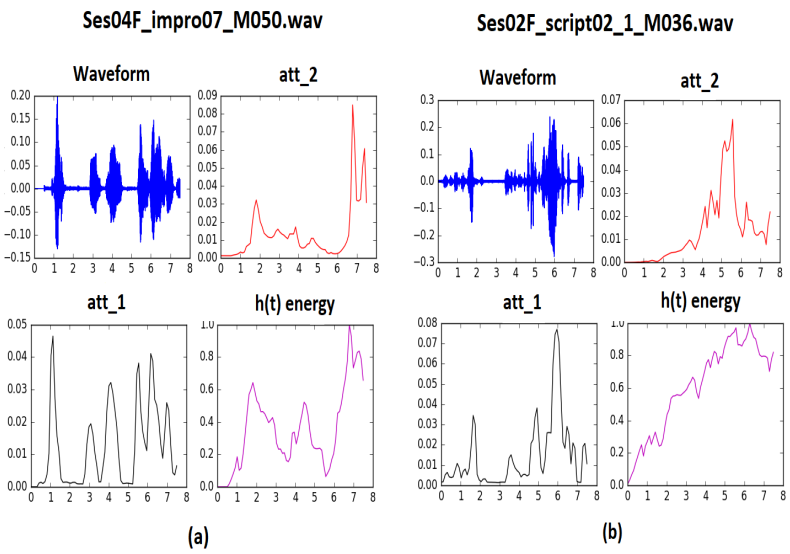

Figure 4.2: Plots of raw waveform (top left), att_1 (bottom left), att_2 (top right), energy of h(t) (bottom right) for two audio files. Note that while att_1 follows the envelope of the raw waveform, att_2 follows the envelope of energy waveform of h(t)

of att_1 and att_2 i.e.

$$att = (1-\beta)att_1 + \beta att_2 \tag{4.3}$$

We varied the parameter $\beta = [0, 0.2, 0.8, 1]$ to see the effect of weighting the two attention mechanisms differently. From Figure4.3 we observe that the mean class-wise accuracy obtained over the five cross-validation splits of IEMOCAP doesn't vary significantly from baseline. This is similar to the results obtained in Huang and Narayanan [59] and also in Neumann and Vu [87] where implementing attention mechanisms didn't lead to any significant changes in accuracies from the model without attention mechanism. At the same time we can see that it decreases as we increase the value of β. Note that as β increases, contribution of att_2 increases. We believe it could be because of limited data and any advantage gained by implementing the attention mechanism is overshadowed by over-fitting. This means the attention profile so computed is specific to the training set and does not generalize well to unseen validation sets. It would be an interesting experiment to

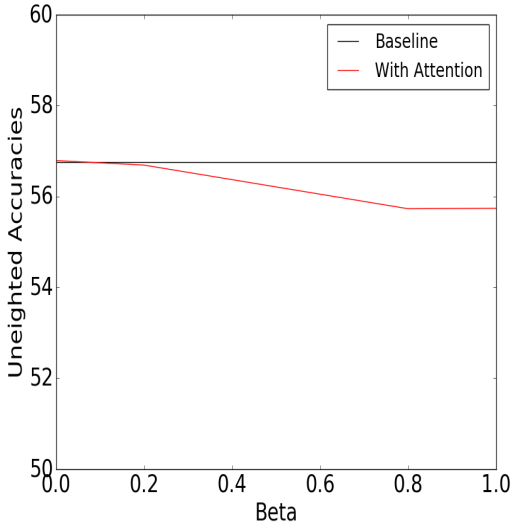

Figure 4.3: Unweighted accuracies vs beta. As we increase the parameter 'beta' accuracy decreases which is possibly because of overfitting

see their effects on emotion recognition on a larger dataset.

4.4 Multi-modal experiments

Here we show the results and our analysis for the experiments performed. Our metric would be un-weighted accuracy (UWA) which is the average of class-wise accuracies. Since our datasets are not perfectly balanced, we believe it would be a better metric to use than the overall accuracy or weighted accuracy. The results shown have been averaged across four runs with different random seeds.

4.4.1 Comparing audio and text modalities

Our initial set of experiments were carried out to show the worth of multi-modal systems. We compared the two different audio feature sets EGeMAPS and the ones obtained using pyAudioAnalysis but didn't notice a big difference in the accuracies. We believe that since the feature sets have undergone speaker based normalization prior to being fed to

the neural network model, we are getting rid of speaker specific characteristics and hence the speaker-specific prosodic features used in EGeMAPS don't deteriorate the performance of the audio-only model.We chose to use the pyAudio feature set for further experiments. Next we investigated the performance of a text-based system and a multi-modal system. It can be seen from Table 4.1 that both of those models perform better than an audio-only model. To verify our assumption that the audio modality is better for detecting arousal while the text modality is better at detecting valence, we provide the confusion matrices in Figure 4.4. It can be seen that the audio-based model (left) performs better than the text-based model (center) in detecting 'anger' which is a high arousal emotion. From the first row of the matrices, we can also see that the text based model is more likely to confuse the 'angry' and 'sad' classes than is the audio based model. This is because both anger and sadness are low valence emotions but they differ in their arousal level, thereby making it easier for the audio modality to distinguish between the two. Both the models perform similarly when it comes to identifying the 'neutral' speech samples. This is probably because the 'neutral' class lies somewhere in the middle of the arousal and valence axes and not at one of the extremes. Hence neither of the modalities end up having any advantage over the other. However, text based models do a much better job in identifying the 'happy' samples than the audio based model. While the audio based model classifies 26% of 'happy' samples as angry, our text based model does a better job at distinguishing between the two classes. It further strengthens our hypotheses that text based models are better than audio based models in distinguishing between high and low valence utterances. While anger is a low valence emotion, happiness is high valence. Combining the two modalities we see that class-wise accuracies either improve or remain almost the same for all of the classes. Accuracies for 'sad' and 'neutral' obtained using multi-modal system are better than that of uni-modal systems indicating that speech and text features supplement each other while identifying samples from these two classes.

To further identify which modality helps with classification of what emotions, we implemented an attention based multi-modal fusion for emotion recognition as has been done by Hori et al [57] for video description. However, our implementation was a simpler version of their implementation. One reason for a simpler implementation was because we had lesser amount of data-points to work with and hence a simple implementation with lesser number of parameters would prevent over-fitting. Another reason for this was because of the way these problems are formulated. For a video description model, input to a recurrent network are video samples and their output would be sentences which is a sequence of words. This architecture is known as many-to-many recurrent model because both input and output are sequential. In our case the input to LSTM models is a time-series consisting of audio/text features for an utterance whereas the output is just one emotion label per utterance. These models are known as many-to-one recurrent models. Implementing an attention model for a many-to-one recurrent model is more simplified than many-to-many recurrent models as we don't have to consider the effect of a sequential output on computing the attention weights. But the core idea still remains the same. We want to implement an attention mechanism which assigns relevance weights to the summary feature vectors obtained from each modality. The summary feature vectors are then multiplied by the corresponding relevance weights and added before being fed to the following layers. Please note that in the multi-modal system explained in section 4.2.3 the two summary vectors were being concatenated instead of being added. While Hori et al. reported an improvement in performance from multimodal fusion, we didn't notice any significant change in our metric UWA. However it did give us some important insights as to how the audio and text modalities work towards classifying the four emotions. We now describe our implementation to compute the relevance weights for the two modalities. Let the summary vector obtained for the time series of audio features after passing them through LSTM layers be denoted by $\mathbf{a} \in \mathbb{R}^m$ and the summary vector obtained for the sequential text features be

$\mathbf{t} \in \mathbb{R}^m$ where m is the number of units in the final LSTM layer (assuming its the same for both audio and text modalities as in our case). We define a weight layer $\mathbf{w} \in \mathbb{R}^m$ which was shared across the two modalities. The relevance weights γ and τ for audio and text modalities respectively were computed using the following equations:

$$\gamma = \frac{\mathbf{w}^T \mathbf{a}}{\mathbf{w}^T \mathbf{a} + \mathbf{w}^T \mathbf{t}} \tag{4.4}$$

$$\tau = 1 - \gamma = \frac{\mathbf{w}^T \mathbf{t}}{\mathbf{w}^T \mathbf{a} + \mathbf{w}^T \mathbf{t}} \tag{4.5}$$

Hence, if for an utterance $\gamma < 0.5$, it implies $\tau > 0.5$ and so text modality was weighed more in classifying that particular utterance. In Figure 4.5 we plot the histograms depicting the number of utterances having a certain value of audio relevance γ as computed from the multi-modal attention mechanism. We observe that while for 'angry' and 'sad' classes $\gamma > 0.5$ for most utterances, its the other way around for 'neutral' and 'happy'. This means that for classifying most of the 'angry' and 'sad' utterances audio modality was found to be more relevant while for most 'neutral' and 'happy' utterances text modality was more helpful. Referring to Figure 4.1 gives us an insight as to why this could have been the case. We note that the histogram is obtained for the training set of one of the cross-validation splits so as to make sure that most of the utterances were assigned to the correct class although the accuracy wasn't 100% because the training was stopped once the validation error started rising. We see that while most of 'angry' utterances have high arousal, most of 'sad' utterances have low arousal but they have similar valence distribution. Hence, text features being better at detecting valence would get confused to distinguish between the two while audio features can easily make that distinction. At the same time both 'neutral and 'happy' classes have a good amount of their utterances with neither high nor low arousal values (concentrated around that 3-3.5 region on the x-axis) and hence audio modality can be helpful in classifying 'angry' or 'sad' from these two classes. Similarly, 'neutral' is

Table 4.1: UWA obtained from 5-fold cross-validation on IEMOCAP. Ground truth text transcriptions are used here.

Model	pyAudio	Egemaps	Glove	pyAudio + Glove
UWA	56.94	56.85	61.89	**68.18**

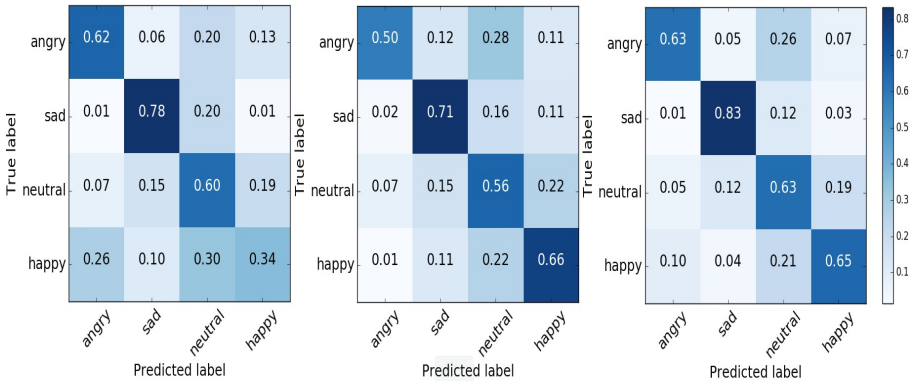

Figure 4.4: Confusion matrices for one of the validation splits showing the class-wise accuracies of audio-only (left), text-only (center) and multi-modal (right) systems. Numbers shown are in percentages

the only class with most utterances having neither high nor low valence unlike the other three classes which leads to text features being more helpful in classifying the 'neutral' utterances. Similarly,'happy' is the only class with utterances having higher valence than the rest of the classes so text features play a greater role in identifying them. However, we also see audio modality being assigned higher relevance weight for some of the 'happy' utterances. This is probably because some 'happy' utterances also have high arousal values causing them to be confused with 'angry' as seen from the multi-modal confusion matrix in Figure 4.4. Even though that matrix was computed for one of the validation sets, the same confusion must also be occurring in training set. This experiment further verified our claim that audio helps with arousal level classification while text helps with valence level classification.

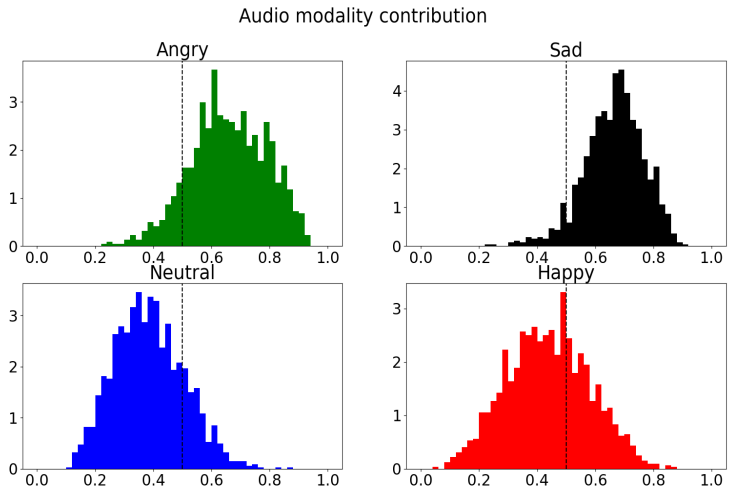

Figure 4.5: Histogram plots showing the relevance weights assigned to audio modality (γ) for different utterances belonging to different classes namely angry (green), sad (black), neutral (blue) and happy (red). The vertical black line in each plot shows x=0.5 i.e. for samples lying along that line there is equal contribution from audio and video. Note that for most angry and sad utterances, audio modality contributes more towards classifying them while for neutral and happy utterances its the other way round.

4.4.2 ASR model output vs ground truth transcriptions used for multi-modal classification

Having performed the multi-modal experiments on ground truth transcriptions, we now created a pipeline where we only used audio data as input. We used the ASR transcriptions generated from audio in the multi-modal system instead of ground truth transcriptions. Since, the different APIs were able to transcribe different numbers of utterances, we ran the experiments comparing the models with a different train/test file-list for each API. This resulted in different accuracies even when only audio features were used or when they were used along with text features obtained from ground-truth transcriptions. Figure 4.6(a) shows the comparison between the cross-validation UWAs obtained from the audio only model, the multi-modal system using ground truth text and the multi-modal system using the API's transcription. It indicates that the model trained on ground truth transcriptions

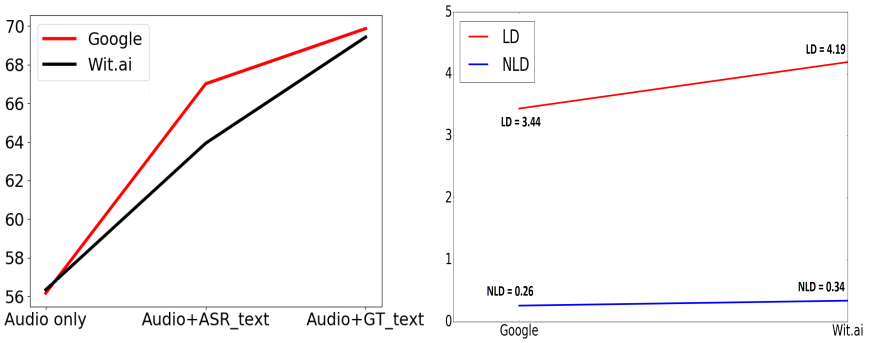

Figure 4.6: (a) Left figure compares the performance of an audio-only model with multi-modal systems using ground truth text or the ASR transcriptions for the different ASR systems (b) In the right we show the performance of the different ASR modules used in our experiment on IEMOCAP. Lower is better

perform better than the ASR transcriptions as expected. We get a relative loss of 4% and 5.3% in accuracy compared to ground truth transcriptions when using Google's and Wit.ai's ASR engine, respectively. To compare the quality of the transcriptions generated, we computed the word error rate (WER) by measuring the Levenshtein distance (Heeringa [55]) (LD) between the generated transcriptions and the ground truth ones for each IEMOCAP utterance and then averaging it over the entire dataset. Levenshtein distance between two sentences measures the minimum number of insertions/deletions/substitutions of words required to convert one sentence to another. In general, longer utterances are more likely to have a higher LD when compared to shorter utterances because there are more words where the ASR model can make an error in transcribing. Since the different API's transcribed different numbers of utterances, this measure could provide us with a skewed idea about the performance of APIs. Hence, we also computed a normalized Levenshtein distance (NLD) where we divide LD by the number of words in the ground truth transcription. Figure 4.6 compares the performance of the two ASR APIs in terms of those two metrics. We see that the difference is less stark in case of NLD, however both the metrics show similar trends. The lower drop in UWA compared to ground truth transcriptions was obtained

using Google's system. This can be explained by its lower word error rate as obtained for the IEMOCAP dataset. Google's and Wit.ai's APIs have probably been trained on a large amount of data so that the deep learning models used for ASR in both the APIs were more generalizable giving us satisfactory performance on an unseen dataset. Wit.ai's API seems to perform worse than Google's API in terms of UWA, but we should also keep in mind that we are using different subsets of the dataset to evaluate the models. Also we are using less data to train the pipeline using Wit.ai's transcriptions (as explained in section 4.2.4) which could also be one of the reasons for its worse performance. Having looked at the WER of the two APIs, we now compare the average confusion matrix obtained over five cross-validation sets for multi-modal systems using ground truth transcription vs ASR outputs in Figure 4.7. It can be observed that class-wise accuracies are higher when ground truth transcriptions are used as expected. Comparing models using Google API's output with that of using ground truth transcriptions, the absolute increase in percentage of 'happy' samples being classified as 'angry' and 'neutral' is more than that of the 'sad' class. This could be because the arousal value distribution of 'happy' utterances is more similar to 'angry' and 'neutral' than that of 'sad' utterances (from Figure 4.1). When using Wit.ai API's output instead of the ground truth transcriptions we see more 'angry' samples being miss-classified as 'happy' probably for a similar reason. The same is true when there are more 'sad' samples being miss-classified as 'neutral' and 'happy' when Wit.ai is used to transcribe. These observations show that using worse quality transcriptions leads to more confusion between classes with similar arousal values which points to the fact that audio features contribute to the classification to a greater extent in such cases.

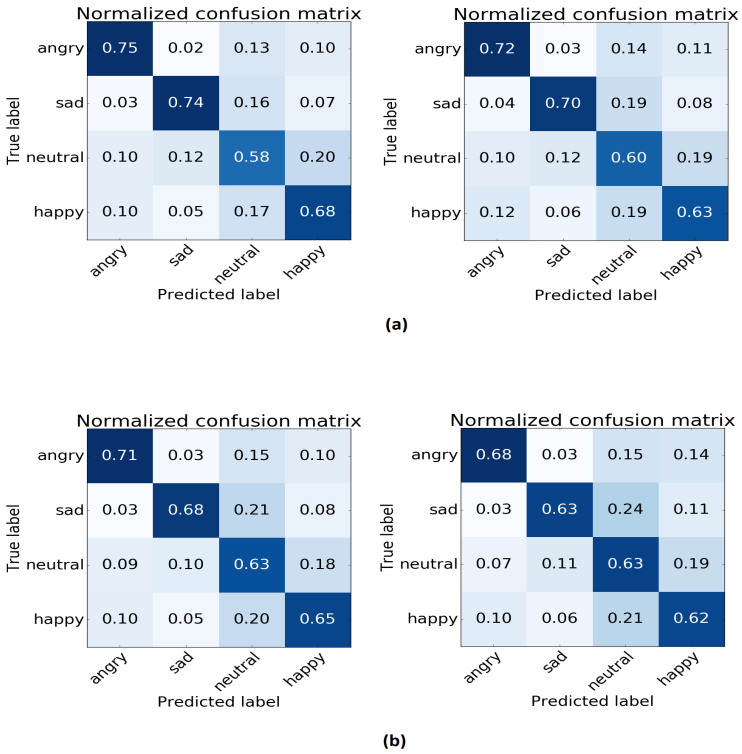

Figure 4.7: Confusion matrix obtained from multi-modal systems using ground truth transcriptions (left) and ASR transcriptions (right) using (a) Google's and (b) Wit.ai's API

4.4.3 Cross-corpus analysis

To verify the generizability of our model, we did a cross-corpus analysis where we trained our model using IEMOCAP and tested it on MSP-IMPROV. We preferred IEMOCAP for training because it is more balanced. We have compared the performance between an audio-only system and a multi-modal system using the generated transcriptions. The tokenizer used in these experiments were generated from the IEMOCAP dataset. Doing so would allow us to capture the cross-domain difference in their vocabulary. Utterances in MSP-IMPROV for which we could not find any of the words in the tokenizer were not used in the experiment. We see a similar trend where using ASR transcriptions along with audio results in a better emotion recognition model. The Google based system gives a relative improvement of 9.8% and using Wit.ai's ASR API results in a relative improvement of

Table 4.2: Cross corpus results with IEMOCAP as training set and MSP-IMPROV as test set.

Model	**Google**	**Wit.ai**
Audio only	35.93	38.06
Audio + ASR output	**39.45**	**40.08**

5.2% compared to an audio-only model. However, the improvements weren't as much as observed in cross-validation experiments, possibly due to cross-domain differences in the vocabulary of IEMOCAP and MSP-IMPROV.

4.5 Conclusion

Our experiments demonstrate that acoustic features help in detecting level of arousal whereas the text based model helps in detecting valence level. Combining information from both to build a multi-modal system seems to increase the class-wise accuracies. When using ASR transcriptions instead of ground truth ones, audio features seem to contribute more towards deciding which class an utterance should belong to. Deep learning based ASR models trained on thousands of hours of data (Prabhavalkar et al. [95]) improves their generalizability thereby giving us meaningful transcriptions for unseen datasets which we can leverage to get higher cross-corpus accuracies. Hence, we can take advantage of the genralizability of ASR models to improve the generalizability of emotion classification models. In the future we plan to investigate the utility of articulatory features by incorporating them in our models. We also aim to explore various word embeddings other than Glove or sub-word embeddings which are better at handling out of domain vocabulary words. We also plan to look at ways we can get word embeddings specific for an emotion recognition/sentiment classification task (Tang et al. [115]). It would also be interesting to explore text features obtained using dictionaries used specifically for an emotion recognition/sentiment classification tasks. Additionally, we plan to explore novel ways to combine the information from the audio and text modes in the multi-modal learning framework.

Chapter 5: Generative models to capture the underlying distribution of feature vectors

5.1 Introduction

Emotion recognition is a fairly widely researched topic. Some of the previous works done by Williams and Stevens [124] and by Banziger and Schere [8] include use of F0 contours. In their survey paper, El Ayadi et al. [30] mention several features such as formant features, energy related features, timing features, articulation features, TEO features, voice quality features and spectral features useful for emotion recognition . Researchers have also investigated various machine learning algorithms such as Hidden Markov Models (Lin and Wei [75]), Gaussian Mixture Models (GMM) (Hu et al [58]), Artificial Neural Networks (Singh et al [111]), Support Vector Machines (SVM) (Ververidis and Kotropoulos [122]) and binary decision trees (Lee et al [70]) for emotion classification. Recently, researchers have also proposed several deep learning based approaches for emotion recognition (Huang and Narayanan [59]). Stuhlsatz et al. [113] reported accuracies using a Deep Neural Network on 9 corpora using Generalized Discriminant Analysis features to do a binary classification between positive and negative arousal and positive and negative valence states. Xia and Liu [125] implemented a denoising auto-encoder for emotion recognition. They captured the neutral and emotional information by mapping the input to two hidden representations, and later using an SVM model for further classification. Ghosh et al. [38] used denoising auto-encoders and showed that the bottleneck layer representations are

highly discriminative of activation intensity and at distinguishing negative versus positive valence.

A typical setup in several of these studies involves using a large dimensionality of features and using a machine learning algorithm to learn class boundaries in the corresponding feature space. This design renders a joint feature analysis in the high dimensional space rather difficult. Methods such as principal component decomposition (PCA) and linear discriminant analysis (LDA) have been known to compress high dimensional feature into lower dimensions. PCA aims to de-correlate the features by finding the axes with maximum variance where the data is most spread, and then projecting the original feature vectors onto those dimensions. LDA projects data-points onto axes so as to minimize the within class covariance of the projected data-points but maximize the between class co-variances. More details about these methods can be found in the book "Pattern classification" by Duda et al [29]. Auto-encoders (Baldi [6]) have also been used for similar tasks. The input high dimensional feature vector is passed as input to a stack of neural network layers. The initial part of an auto-encoder is known as encoder. It consists of a series of hidden layers with the number of neurons in them decreasing from one layer to another. The final layer of encoder called the bottleneck layer, has the same number of neurons as the dimension of the compressed space. Assuming the encoding function is denoted by E, for an input $\mathbf{x}$ the output of an encoder can be denoted by $E(\mathbf{x})$. The second part of auto-encoder following the encoder is called a decoder which renders the compressed representation back to the original dimension through a series of neural network layers. Denoting the decoding function by D, the final transform that the auto-encoder applies on an input $\mathbf{x}$ is given by $D(E(\mathbf{x}))$. The weights of the neural network layers are then updated by backpropagating errors from a loss function which intends to make $D(E(\mathbf{x}))$ as similar to $\mathbf{x}$ as possible either by minimizing the mean square error between them or using a cross entropy loss function. PCA, LDA and auto-encoders have been investigated as dimensionality reduction

techniques for speech emotion recognition (You et al [128] and Cibau et al [24]). However the drawback of these methods is that even in the lower dimensions, it is hard to see structures or clusters being formed by the feature vectors belonging to same class. We train a model that leverages label information to cluster the input vectors and then apply the compressed representation on a test/validation set. We believe this type of a framework can be useful in deciding the worth of features being used for a particular task. For example, if the compressed features from both training and test sets cluster well according to the categories they belong to, it means the features are well suited for the classification task. If on the other hand it doesn't cluster well for test set, it means they are not. We compare this method with other traditional methods both qualitatively and quantitatively and investigate their generalizability.

The next part of this chapter is focused on generative models. As mentioned before, discriminative models learn a conditional distribution $p(y_i|\mathbf{x_i})$, given a set of feature vectors $\mathbf{x_i}$ and the corresponding labels $y_i, i = 1..N$. Generative models on the other hand model the distribution of the classes. They aim to learn the joint distribution $p(\mathbf{x_i}, y_i)$. We implement generative adversarial networks (GANs, proposed by Goodfellow et al [42]) based models to learn the distribution of feature vectors used for speech emotion recognition. We define metrics on how to compare the different models and investigate their generalizability and applicability.

5.2 Generative adversarial networks

A generative adversarial network consists of two components: a generator, G and a discriminator, D. Given a random sample $\mathbf{z}$ from a random probability distribution $p_\mathbf{z}$, the generator is responsible for generating a fake data-point $G(\mathbf{z})$. The discriminator attempts to classify real samples $\mathbf{x}$ (drawn from a distribution p_{data}) against the one generated by

the generator. Probability distribution $p_{\mathbf{z}}$ is usually considered to be of lower dimensional and simpler than the data distribution p_{data}. Popular choices include Gaussian or a uniform distribution. The objective of training a GAN is to obtain a generator that can mimic real data such that the discriminator is incapable of differentiating between real and fake samples. GAN is trained using the following optimization on the GAN loss $V(D,G)$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{5.1}$$

In the equation above, $D(\mathbf{x})$ and $D(G(\mathbf{z}))$ are the probabilities that $\mathbf{x}$ and $G(\mathbf{z})$ are inferred to be real sample by the discriminator. Note that in the optimization in equation 5.1, the generator attempts to fool the discriminator as it tries to minimize $V(D,G)$. During GAN training, optimization of the loss function is achieved by updating the parameters of the discriminator and generator networks in an iterative way. We minimize the discriminator and generator losses as defined below and track them separately. Note that for discriminator loss, y is 1 if input is $\mathbf{x}$ and 0 if input is $G(\mathbf{z})$.

$$\begin{aligned} &\textbf{Disc. loss: } -y\log(D(\mathbf{x})) - (1-y)\log(1 - D(G(\mathbf{z}))) \\ &\textbf{Gen. loss: } -\log(D(G(\mathbf{z}))), \text{where } \mathbf{x} \sim p_{\text{data}}, \mathbf{z} \sim p_{\mathbf{z}} \end{aligned} \tag{5.2}$$

Figure 5.1 provides a block diagram of a GAN architecture. While a lot of work has been done exploring the applicability of GANs for vision tasks, there are only a few such works that has explored their utility for speech emotion recognition in recent years. In [51], Han et al propose adding an extra GAN based adversarial loss term along with the usual categorical cross entropy loss term to predict emotions from speech. Eskimez et al [32] investigate unsupervised feature learning using various GAN and auto-encoder based architectures for speech emotion recognition. In the next sections we discuss the variations of GAN architectures we have used followed by our experiments and results.

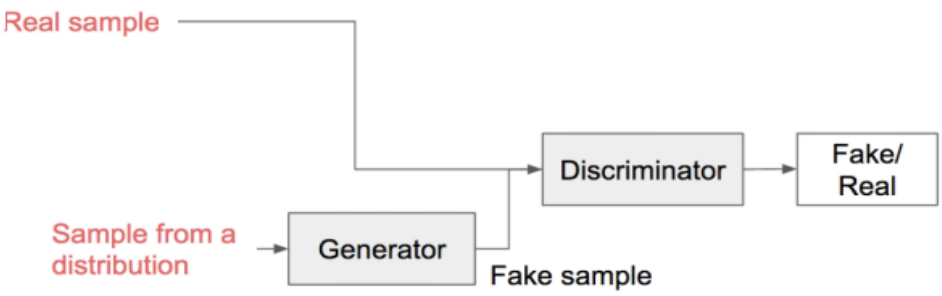

Figure 5.1: Block representation of a GAN architecture. A GAN requires access to real samples from a dataset and samples from a probability density.

5.2.1 Adversarial auto-encoders

Adversarial auto-encoders (AAE) proposed by Makhzani et al [79] have been shown to perform quite well in digit recognition and face recognition tasks. We use adversarial auto-encoders for emotion recognition in this paper motivated by their performances on other tasks for feature compression as well as data generation from random noise samples. Speech emotion recognition involves working with high dimensional features which can render a joint feature analysis in a high dimensional space rather difficult. Adversarial auto-encoders address this issue by encoding a high dimensional feature vector onto a code vector, which can be further enforced to follow a pre-defined probability distribution function. This has been termed as mapping space distribution (MSD) in Figure 5.2. To the best of our knowledge, this is the first such application of adversarial auto-encoders to the domain of emotion recognition. We borrow a specific setup of adversarial auto-encoders with adversarial regularization to incorporate class label information as has been shown in Figure 5.2. An adversarial auto-encoder broadly consists of two major components: a generator and a discriminator. In Figure 5.2, we show the generator at the top, which given a sample $\mathbf{x}$ from the real data (e.g. pixels from an image, features from a speech sample) learns a code vector for the data sample. We model an auto-encoder for this purpose, where

the model learns to reconstruct $\mathbf{x}$ through a bottleneck layer. We represent the reconstruction for $\mathbf{x}$ as $\mathbf{x}'$ in Figure 5.2. The discriminator (in the bottom half of Figure 5.2) obtains the code vectors encoded by the auto-encoder as well as samples from MSD, and learns to discriminate the encoded real samples from the MSD samples. The generator and the discriminator operate against each other, where the discriminator attempts to accurately classify real samples against MSD samples and the generator produces code vectors resembling MSD samples to confuse the discriminator (so that the discriminator is not able to distinguish real from synthetic inputs). They further proposed tricks such as, in a setting where the samples x belong to different classes, the MSD is a mixture of Probability Distribution Functions with as many components as the number of classes. In our case it was chosen to be a 2-dimensional 4 component (since our task is a 4-way classification) Gaussian mixture with same co-variance matrices and their mean vectors orthogonal to each other. The orthogonal means ensure that the different mixture components are maximally separated. Furthermore, to enforce each component of the mixture PDF to correspond to a class, the authors regularized the hidden code vector generation by providing a one-hot encoding for the classes to the discriminator.

Our model is trained while the following two adversarial losses converge: (i) cross-entropy is minimized for code vectors to be classified as MSD samples (implying encoder is able to generate code vectors resembling MSD), and (ii) cross-entropy is minimized so that the discriminator is able to classify between encoded samples and samples from MSD. More specifically, while adversarial losses converge the parameters of the adversarial auto-encoder are updated in the following iterative way:

- Weights of the auto-encoder (both encoder and decoder) are updated based on a reconstruction loss function. We chose this function to be Mean Squared Error (MSE) between the inputs x and the reconstruction x'.

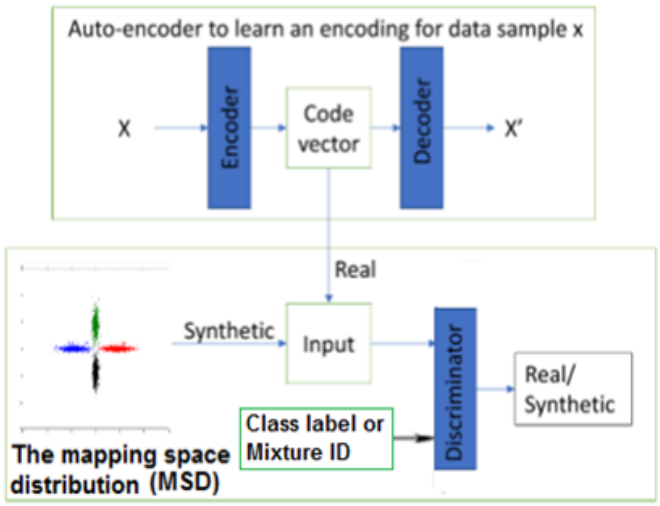

Figure 5.2: A summarization of the adversarial auto-encoders. The generator at the top creates code vectors. The discriminator learns to classify the code vectors generated from real data from the synthetic samples. Label information is provided to discriminator so that samples from a particular class are mapped to a specific mixture component of MSD

- The data is transformed by the encoder and we sample an equal number of samples from MSD $p(z)$. Weights of the discriminator are updated to minimize cross-entropy to classify between encoded samples and samples from MSD.

- We then freeze the discriminator weights. The weights of encoder are updated based on its ability to fool the discriminator (equivalently minimizing the cross-entropy for real samples to be labeled as MSD samples).

Once trained, we can use the encoder to get compressed representations of higher dimensional feature vectors. At the same time we can sample points from MSD, pass it through decoder and generate synthetic feature vectors. The performances of an adversarial auto-encoder in this regard are discussed later in this chapter.

5.2.2 Data generating GAN

The main purpose of implementing an adversarial auto-encoder was to generate meaningful lower dimensional representations from higher dimensional features. The synthetic samples generated from the decoder of the auto-encoder was a useful by-product. While we mapped the encoded samples to match a mapping space distribution with maximally separated mixture components until adversarial losses converge, the only updates done to the decoder (that generates synthetic samples given samples from MSD) were based on the auto-encoder reconstruction error. A question that comes to mind is how realistic the synthetic samples would be if we trained a GAN based model where a discriminator differentiates between synthetic and real samples until adversarial losses converge. We discuss such an implementation in this section. The architecture is similar to what has been shown in Figure 5.1 with a few modifications. The real samples consisted of the high dimensional feature vectors obtained from real data. $p_{\mathbf{z}}$ was considered to be same as the pdf of MSD in case of an adversarial auto-encoder i.e. a 2-dimensional 4 component Gaussian mixture with orthogonal means and same co-variance matrix to ensure the mixture components are maximally separated. Since, we consider our dataset to have four classes, we train the GAN in such a way that each component of the mixture pdf when passed through the generator, generates a synthetic sample belonging to a specific class. To ensure correspondence between the mixture components and classes, we provide the discriminator with an additional input of one-hot label vector in the same way we did in case of adversarial auto-encoder's discriminator. For real data-points the one-hot label vector depicted its class whereas in case of synthetic samples it depicted the mixture index of the 4 component Gaussian mixture from which it was generated. The generator in our data generating GAN had the same architecture as the decoder of our adversarial auto-encoder set up. To make sure the adversarial errors converge, we had to incorporate some tricks. The changes incorporated were

mainly to improve the generator: (i) Initializing the generator's weight with the decoder's weight of a trained adversarial auto-encoder (ii) keeping the generator's learning rate higher than the discriminator (0.01 vs 0.001 respectively) and, (iii) training the generator for two iterations for every iteration of discriminator training. The effects of these methods has been discussed in more detail in Sahu et al [100]. We call this architecture dGAN_1.

One thing to note in this architecture is that $p_{\mathbf{z}}$ was considered to be a mixture pdf based on the number of classes we have. However, in most GAN applications it is considered to be a simpler pdf like a normal or a uniform distribution. We modified our above GAN architecture so that $p_{\mathbf{z}}$ was now a normal distribution. We also modified it so that now it was a conditional GAN (Mirza et al [82]) where along with providing the generator with a sample $\mathbf{z}$ from $p_{\mathbf{z}}$ we also provide it the class label that we want the generator's output to belong to. This was required as we did not have different mixture components to generate data from for different classes. In addition we added an extra term to the GAN loss function that maximizes the mutual information between the class label provided at the generator input and the discriminator output. Since, implementing the actual mutual information loss was intractable, we implement an approximation using variational technique as has been discussed in Chen et al [20]. We note that only the parameters of the generator are modified based on this mutual information loss function. We call this architecture dGAN_2.

While adversarial losses converge the parameters of the dGANs are updated in the following iterative way:

- Points are sampled from $p_{\mathbf{z}}$ and fed to generator to generate synthetic samples (along with class labels in case of dGAN_2). Weights of the discriminator are updated to minimize cross-entropy to classify between synthetic samples and real samples.

- We then freeze the discriminator weights. The weights of generator are updated based on its ability to fool the discriminator (equivalently minimizing the cross-

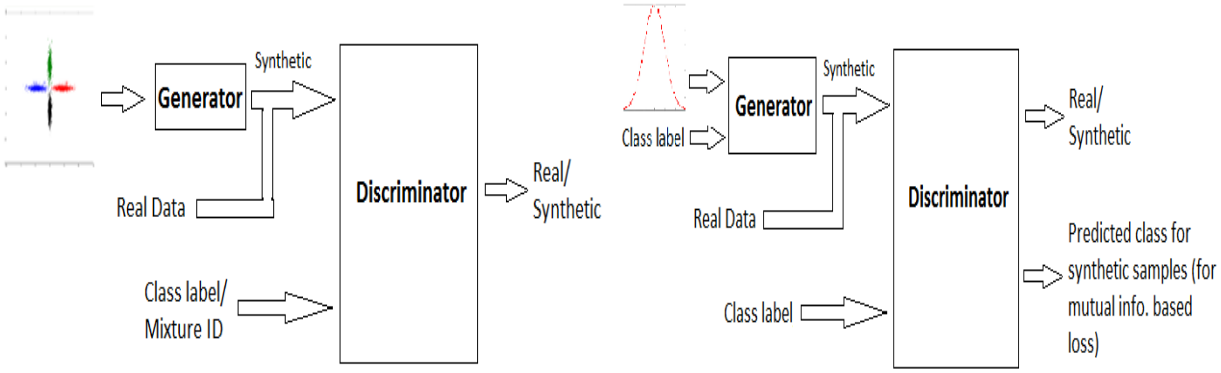

Figure 5.3: Architectures for dGAN_1 (left) and dGAN_2 (right). Note that in case of dGAN_2 the discriminator has a second output layer which predicts the class of the synthetic samples generated. Mutual info. based loss is added while optimization so that the predictions are as close to the class label being provided to the generator.

entropy for synthetic samples to be labeled as real samples). In case of dGAN_2, an additional loss term based on mutual information is also considered to update generator's weights.

Once trained, we can sample points from $p_{\mathbf{z}}$ and feed it to the generator (along with the class labels in case of dGAN_2) to generate synthetic samples.

5.2.3 Adversarial auto-encoder with data generating GAN

The next set of GAN architectures we implemented was to get the best of both worlds by combining adversarial auto-encoders with data generating GANs discussed in last sections. Since, the generator of the data generating GANs in our case had the same architecture as the decoder of adversarial auto-encoders, we simply combined the the two and trained them jointly as shown in Figure5.4. Since the adversarial auto-encoder and the data generating GANs are trained simultaneously, there was no need to initialize with the decoder's weight of a trained adversarial auto-encoder as was done in case of dGAN_1. Also instead of five, the generator was trained for three iterations for every iteration of discriminator training. We call the two architectures as AAE_dGAN_1 and AAE_dGAN_2 corresponding to the

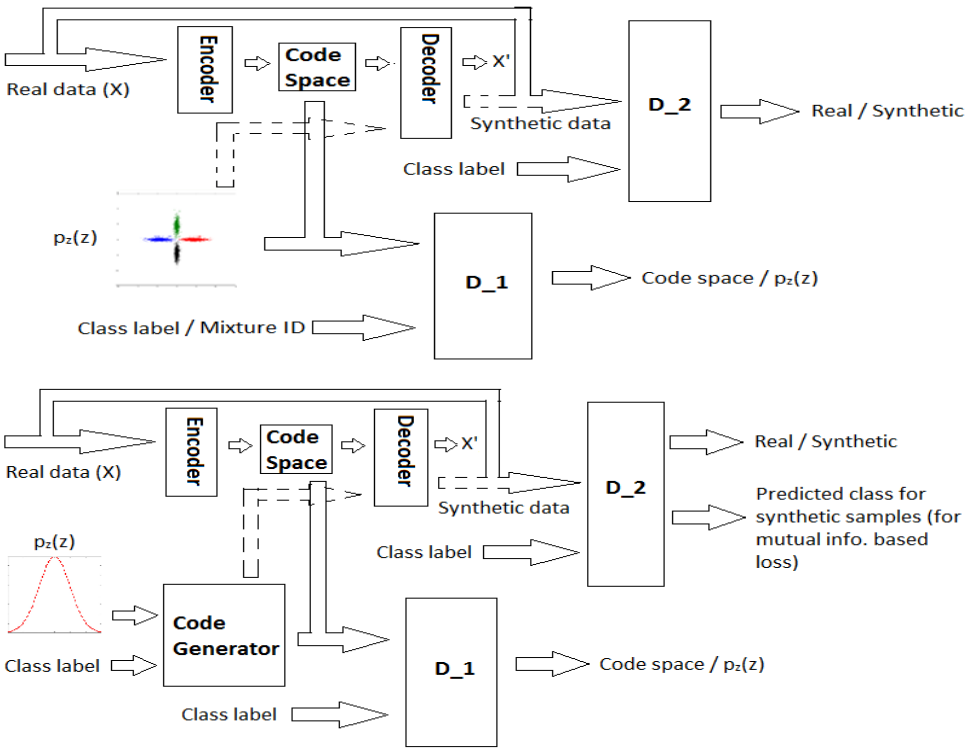

Figure 5.4: Architectures for AAE_dGAN_1 (top) and AAE_dGAN_2 (bottom). Note that there are two discriminators now, one to learn the encoding space and one to generate data samples. While in AAE_dGAN_1, the encoding space is pre-defined to be a mixture of 4 maximally separated Gaussians, in case of AAE_dGAN_2 it is being learned from the training data provided using a code generator block

data generating GANs they have been derived from. Note that AAE_dGAN_2 has been implemented by Wang et al. for computer vision tasks [123].

While adversarial losses converge the parameters of the AAE_dGANs are updated in an iterative way alternating between an AAE training phase and a dGAN training phase. In the AAE training phase:

- Weights of the auto-encoder (both encoder and decoder) are updated based on a reconstruction loss function. We chose this function to be Mean Squared Error (MSE) between the inputs x and the reconstruction x'.

- The data is transformed by the encoder and we sample an equal number of samples

from $p_{\mathbf{z}}$. In case of AAE_dGAN_2, the sampled points are also passed through the code generator: block CG. Weights of the discriminator (D_1 in pictures) are updated to minimize cross-entropy to classify between encoded samples and samples obtained/derived from $p_{\mathbf{z}}$.

- We then freeze the discriminator (D_1) weights. The weights of encoder are updated based on its ability to fool the discriminator (equivalently minimizing the cross-entropy for real samples to be labeled as MSD samples).

In the dGAN training phase:

- Points are sampled from $p_{\mathbf{z}}$ and fed to decoder (in case of AAE_dGAN_1) or to CG + decoder along with a class label (in case of AAE_dGAN_2). Weights of the discriminator (D_2 in pictures) are updated to minimize cross-entropy to classify between synthetic samples and real samples.

- We then freeze the discriminator (D_2) weights. The weights of decoder (in case of AAE_dGAN_1) or to CG + decoder (in case of AAE_dGAN_2) are updated based on its ability to fool the discriminator (equivalently minimizing the cross-entropy for synthetic samples to be labeled as real samples). In case of dGAN_2, an additional loss term based on mutual information is also considered to update the weights.

Once trained, we can use the encoder to get compressed representations of higher dimensional feature vectors. At the same time we can sample points from $p_{\mathbf{z}}$, pass it through decoder (in case of AAE_dGAN_1) or through CG + decoder along with a class label(in case of AAE_dGAN_2) to get synthetic feature vectors.

One thing to note is that unlike in case of AAE_dGAN_1 where the coding space was specified to be a mixture of 4 maximally separated Gaussians, in case of AAE_dGAN_2

the code generator block (CG) learns the coding space from the training data provided. We next evaluate these models' coding and synthetic data generating capabilities.

5.3 Comparison of various models' performance

In this section we discuss the performances of the various various GAN based models. The adversarial auto-encoder and the derived models (AAE_dGAN_1 and AAE_dGAN_2) have an encoder that can project the higher dimensional features onto a lower dimensional code space. At the same time they can also be used to generate synthetic features by sampling points from a prior $p_{\mathbf{z}}$ and passing it through the decoder. The other two GAN based models dGAN_1 and dGGAN_2 have only the capability to generate synthetic data points when we feed their generator with points sampled from a prior $p_{\mathbf{z}}$. After training the models on utterances with emotions, we conduct two specific experiments: (i) judging the encoder's capability to project higher dimensional feature vectors onto lower dimensions in AAE based models, (ii) judging the GAN based models' capabilities to generate synthetic data. We compare them with each other and with a traditional approach like fitting a GMM.

5.3.1 Features

We use the openSMILE toolkit to extract a set of 1582 dimensional feature vector [35]. This feature set consists of various functionals computed for spectral, prosody and energy based features. Same feature set has also been used in several previous works including the INTERSPEECH Paralinguistic Challenges [108]. Similar sets of spectral, prosodic and energy based features has shown considerable success in emotion classification and affect tracking [47].

5.3.2 Projecting higher dimensional points onto lower dimensions

In this section we look at the encoders' performance to project higher dimensional feature vectors onto lower dimensions. The goal of this experiment is to quantify the loss in discriminability after compressing the original feature to a smaller feature subspace. We compare the different AAE based models as well as using more traditional methods for feature compression like PCA and LDA. We also implement a vanilla auto-encoder and compare its compression quality with others. The encoder of the AAE based models implemented have 1582 neurons in their input layer corresponding to the feature dimension. It is followed by three hidden layers with 1000, 500 and 100 neurons respectively before the samples are fed to a bottleneck layer with K neurons corresponding to the dimension of the code space. This is followed by a decoder with 100, 500 and 1000 neurons in three hidden layers followed by an output layer of 1582 neurons. The code generator in case of AAE_dGAN_2 had an input layer of 24 neurons which was fed samples from a 20 dimensional zero mean unit variance Gaussian distribution. The remaining 4 neurons were used to provide the class info using a four dimensional one-hot vector, each neuron corresponding to an emotion. This was followed by a hidden layer with 40 neurons followed by an output layer with K neurons corresponding to the dimension of the code space. The hidden layers in all these blocks had regularized linear (ReLU) activation while the bottleneck and output layers in all these blocks had linear activation. K was fixed at 2 for AAE and AAE_dGAN_1 where the prior $p_{\mathbf{z}}$ was explicitly defined as mixture of four maximally separated Gaussians, each mixture corresponding to an emotion. In case of AAE_dGAN_2 where the code space is learned from data using a code generator block, we swept the value of K among 2, 8, 64 and 256 to find the dimension of the code space that achieves the best separation of the compressed feature vectors. We also experimented with training the encoder and decoder of AAE_dGAN_2 with and without L2 regularization. We discuss the

effect of regularization below. The discriminator in AAE and the discriminators D_1 in the AAE_dGANs all had the same architecture. The input layer has $K+4$ neurons, and was fed with K dimensional samples from either the bottleneck layer of auto-encoder or the samples derived from prior $p_{\mathbf{z}}$ along with class label/mixture index using a four dimensional one hot vector. This was followed by three hidden layers with 1000,500 and 100 neurons with ReLU activation and finally an output layer with one neuron with sigmoid activation indicating whether the sample fed to the discriminator was from bottleneck layer or whether it was a sample derived from $p_{\mathbf{z}}$. The discriminators D_2 in AAE_dGANs has the same architecture as D_1 except the input layer which takes in high dimensional feature sets as input. There is also an auxiliary output layer with four neurons to predict the class of synthetic samples which is used to compute the mutual information based loss. The vanilla auto-encoder implemented for comparison purposes had the same architecture as the architecture of the auto-encoder blocks in the AAE based models.

Single corpora setting

We use IEMOCAP dataset (Busso et al [13]) to run a leave-one-session out five fold cross validation analysis. Since each session had independent speakers we ensure there is no speaker overlap between training and validation sets. Since the individual feature values lie over a wide range of values we do mean-variance normalization of both training and validation sets. We use training data statistics to normalize the validation set.

For a specific cross-validation iteration, we train an AAE based model with the training set. Then we obtain the lower dimensional representations of the raw features for both training and validation set using the trained encoder. Note that the AAE based models aren't being provided any label information while getting the lower dimensional representations. In Figure 5.5, we look at the two adversarial cross-entropy losses namely the

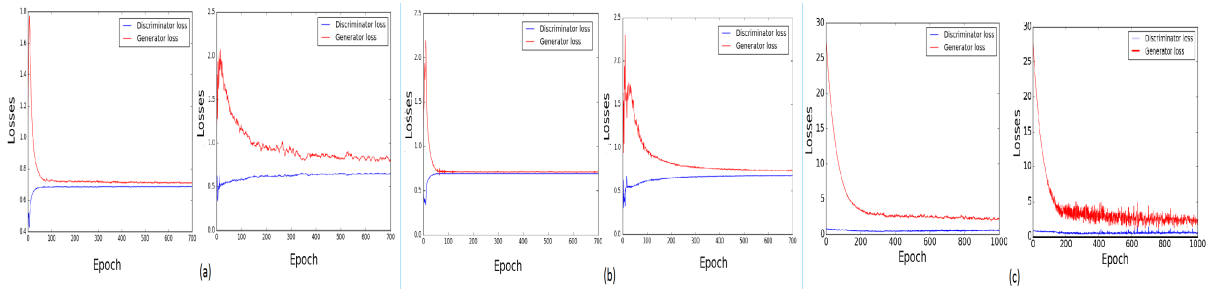

Figure 5.5: Discriminator's (blue) and Generator's (red) loss curves for training (left) and one of the validation sets (right) for (a) AAE (b) AAE_dGAN_1 (c) AAE_dGAN_2

discriminator and generator losses for the three AAE based models. We plot these errors per epoch on the training and the validation set during one specific cross-validation set for all three AAE based models. We observe that the adversarial losses converge indicating that the discriminator's ability to discriminate is countered by generator's ability to confuse it. This trend is observed for both, training and validation sets, indicating that the learnt parameters generalize well to data unseen during model training. We also observe that the error seems to be converging the best for AAE_dGAN_1 indicating their superior coding ability. After we train the AAE based models, we use the encoder in auto-encoder to compute the code vectors for the training set as well as the validation set. We then train an SVM classifier on the openSMILE features as well as lower dimensional representation of these features as obtained using the following techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), an auto-encoder and finally the code vector representations learned using the AAE based models. We learn and obtain these lower dimensional representations (PCA, LDA, auto-encoder and AAE based) of the openSMILE features on the training set, which are then used to train the SVM model. The SVM parameters (box-constraint and kernel) are tuned using an inner-cross validation on the training set. Since our goal here is dimension reduction, we keep the maximum dimension of these representations during our experiments to be 256. We use Unweighted Accuracy (UWA)

Table 5.1: Cross-validation accuracies when the raw opensmile features or its compressed representations are fed to SVM. Metric used was UWA or mean class-wise accuracies

Features	Dimension	UWA
Raw opensmile features	1582	**59.46**
PCA on raw features	2	45.76
	8	53.62
	64	**57.61**
	256	56.49
LDA on raw features	2	50.22
	3	53.42
Vanilla auto-encoder	2	46.78
	8	56.02
	64	57.36
	256	**57.41**
AAE with 4-mixture Gaussian prior	2	**57.01**
AAE_dGAN_1 with 4-mixture Gaussian prior	2	**57.78**
AAE_dGAN_2	2	33.89
	8	50.59
	64	56.14
	256	**57.06**

as our evaluation metric which is basically the average of class-wise accuracies. This is especially helpful in cases where all the training set doesn't have equal number fo samples from all classes. We list the results of the classification experiment in Table 5.1.

From the results, we observe that the performances of SVMs trained on the openSMILE features and the code vectors are fairly close especially in case of AAE and AAE_dGAN_1 where we compress the feature sets from 1582 to just 2 dimensions. This indicates that the compressed code vectors capture the differences between the emotional labels in the openSMILE feature space to a fairly high degree. We do not observe as high a performance from any of the other feature compression techniques when we limit their dimensionality to 2. AAE_dGAN_1 slightly outperforms AAE which further confirms our observation from Figure 5.5 that it has better coding capability. This is probably because while in case of AAE, the parameters of the decoder are updated based only on reconstruction error,

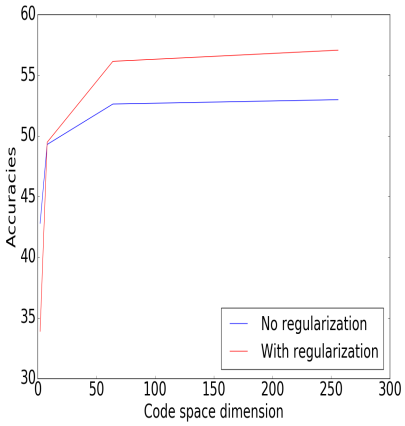

Figure 5.6: Effect of regularization on coding capability of AAE_dGAN_2

in case of AAE_dGAN_1 they are also updated based on the adversarial losses so as to make their output resemble real features. Updating the parameters of encoder then based on reconstruction error of the auto-encoder, makes it better at compressing the real samples onto K dimensions because they can probably generate more realistic samples when passed through the decoder in case of AAE_dGAN_1. We will verify this claim in a later section. Methods such a PCA and compressing using vanilla auto-encoder achieve similar performance when we increase the dimension of the compressed feature vectors. Same is the case with AAE_dGAN_2. The superior performance of AAE and AAE_dGAN_1 as compared to AAE_dGAN_2 can be explained by the fact that while in the former cases we explicitly define the code space $p_{\mathbf{z}}$ to be maximally separated Gaussians, in the latter there is no such explicit definition and learning the code space is data driven. Note that the results for AAE_dGAN_2 shown in Table 5.1 are when we use L2 regularization on encoder and decoder parameters while training them. Figure 5.6 shows the accuracies when regularization is applied vs when its not and we can see clearly that regularization helps. Although the results havent been reported, we note that regularization did not provide us with any benefits in the other two AAE based models. This is probably because without regularization, the code space learned while training in AAE_dGAN_2 is too specific to the

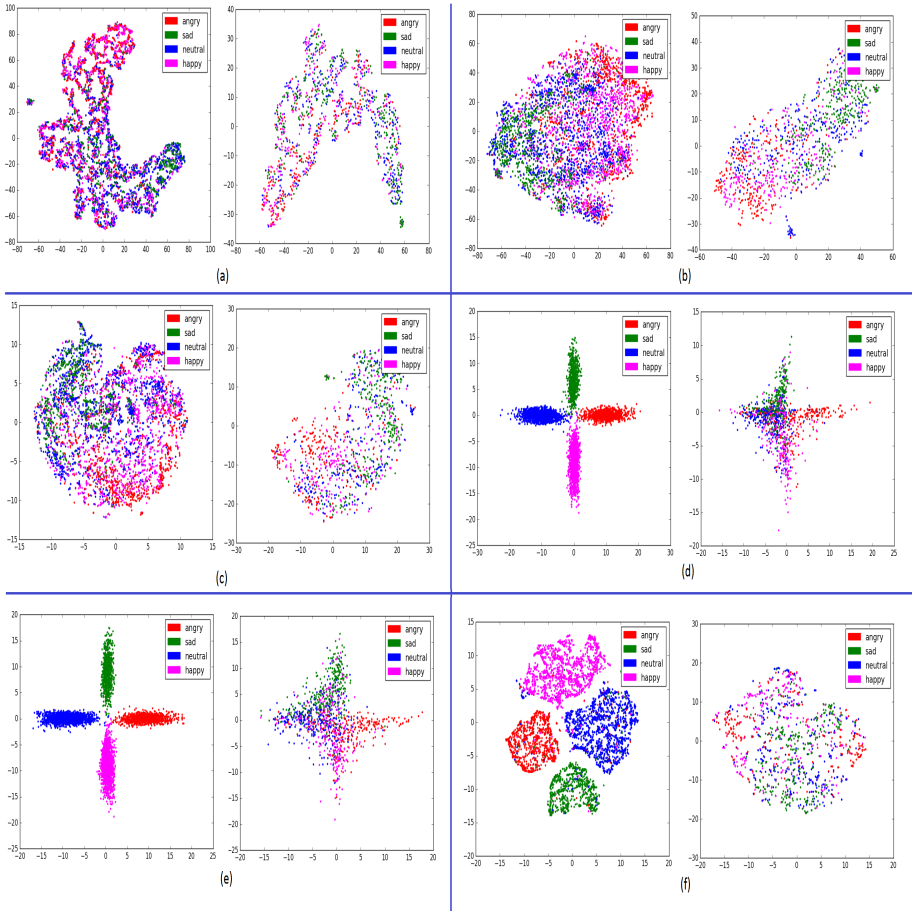

Figure 5.7: TSNE/scatter plots for samples in training (left) and validation (right) set for one of corss-validation IEMOCAP splits (a) 1582-D raw Opensmile features (b) their 64-D PCA encodings (c) 256-D encodings obtained from a vanilla auto-encoder (d) 2-D encodings obtained from AAE (e) 2-D encodings obtained from AAE_dGAN_1 (f) 256-D encodings obtained from AAE_dGAN_2. Note that the 2-D encoding from AAE and AAE_dGAN_1 resemble the matching space distribution which is a mixture of four 2-D Gaussians with orthogonal mean vectors.

training set and fails to generalize well for an unseen validation set. We don't face this issue with the other two AAE based models because the mapping space distribution has been pre-defined to be a mixture of four maximally separated Gaussians. Figure 5.7 we show the scatter/TSNE plots of clusterings obtained from models shown in bold in Table 5.1 for one of the training and validation sets. We observe that AAE based models show almost perfect clustering for training set unlike the non-AAE based methods. This is because of providing label information to the discriminator explicitly while training them. The scatter

plot of 2-dimensional encodings show that the validation set samples are also fairly separable in case of AAE and AAE_dGAN_1. Figure 5.7 provides a sense of the separability of emotion labels by plotting the scatter/TSNE plots of the compressed feature space of the 1582-dimensional openSMILE features. The classification experiments in Table 5.1 quantify this separability.

This low dimensional representation retaining the discriminability across classes provides a powerful tool for analysis in a low dimensional subspace, which is otherwise not possible with a large feature dimensionality. The low dimensional representation could be used for applications such as clustering as well as an "experimentation by observation", as a low dimensional code vector (in particular 2-D) allows plotting the emotion utterances and analyzing them. We also note the fact that the auto-encoder allows reconstructing the features from these code vectors. Therefore, a recovery of the actual utterance representations is also possible, which is otherwise more lossy in other dimension reduction techniques.

Cross corpus setting

Having studied the ability of GAN based architectures to encode higher dimensional features onto a lower dimensional space in a single corpora setting, we now move to performing cross-corpus evaluations. The objective of this experiment is to investigate if a GAN based model can produce meaningful lower dimensional encodings for an external corpus. We use IEMOCAP for training and MSP-IMPROV [16] as our testing set. MSP-IMPROV, like IEMOCAP, also has actors participating in dyadic conversations which has then been segmented into utterances and annotated by evaluators. There are 7798 utterances in total spanned across the same four emotion classes. However, the distribution across classes was highly unbalanced with the number of utterances belonging to happy/neutral class more than three times the number of angry/sad samples. This prompted us to use it

Table 5.2: Cross-corpus accuracies obtained on MSP-IMPROV. Training of SVM is done using either raw opensmile features or encoded version of higher dimensional raw features extracted from utterances in IEMOCAP

	Raw opensmile	PCA	Vanilla auto-encoder	AAE	AAE_dGAN_1	AAE_dGAN_2
Dimension	1582	64	256	2	2	256
UWA	45.14	44.29	44.53	41.51	41.83	44.88

as a test set rather than training set. We only select a few of the models for cross-corpus comparison, specifically the ones which performed better than others and whose results are mentioned in bold in Table 5.1. From results in Table 5.2 we can make similar observations that using AAE and AAE_dGAN_1 we can reduce the dimensionality significantly without much loss in accuracy. However the loss in accuracy in cross-corpus case is more compared to what we saw in cross-validation cases due to domain differences. Unlike cross-validation experiments, higher dimensional encodings obtained from a vanilla auto-encoder or AAE_dGAN_2 or by using PCA seem to be a better representation than the 2D encodings obtained from AAE and AAE_dGAN_1. In Figure 5.8 we show the scatter plots of the encoded features obtained from AAE and AAE_dGAN_1. It shows the some amount of separability is still maintained when features are compressed onto two dimensions. However the scatter plot for test set is highly populated with blue and magenta samples because of higher number of neutral and happy samples in MSP-IMPROV.

5.3.3 Generative capability of GAN models

We next examine the possibility of synthetically creating samples representative of utterances with emotions using the GAN based models we have trained. We generate the synthetic vectors using the models discussed above i.e. AAE, dGAN_1, dGAN_2, AAE_dGAN_1 and AAE_dGAN_2. For AAE, dGAN_1 and AAE_dGAN_1 we randomly sample points from prior $p_{\mathbf{z}}$ which is a mixture of four Gaussians with orthogonal means.

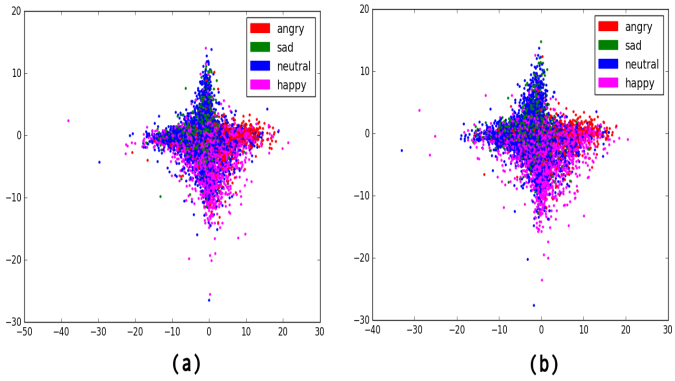

Figure 5.8: Scatter plots for 2D encodings obtained for MSP-IMPROV utterances for (a) AAE (b) AAE_dGAN_1. Models are trained using IEMOCAP.

For dGAN_2 and AAE_dGAN_2 the points are sampled from a zero mean uni-variate 20 dimensional Gaussian distribution. Once the points are sampled they are passed through the trained decoder of the auto-encoder in case of AAE and AAE_dGAN_1 and through the trained generator in case of dGAN_1 and dGAN_2. For AAE_dGAN_2 it is passed through the code generator block followed by feeding its output to the trained decoder. Note that in case of dGAN_2 and AAE_dGAN_2 the generator/code generator is also given the label as input along with samples from the Gaussian prior. Please refer to Figures 5.2, 5.3, 5.4 for a visualization of the work flow of how the synthetic feature vectors were generated. This synthetically generated vector thus is an openSMILE-like feature vector obtained by passing a randomly sampled 2-dimensional code vector through the decoder/generator of the GAN based models (and not directly obtained from an utterance from the database). Note that in case of AAE, dGAN_1 and AAE_dGAN_1 each GMM component was enforced to pertain to a specific emotion label through discriminator regularization using the one hot label vector. The labels for the synthetically generated samples is assigned to be the same as the GMM component label used to sample the code vector. In case of dGAN_2 and AAE_dGAN_2, the synthetic vectors are assigned the label which was fed to the generator/code generator block along with samples from the prior. Note that while in case

of AAE the decoder's parameter is only updated based on reconstruction error, in case of dGANs the generator's parameter are updated based on the adversarial losses. In case of AAE_dGANs we use both reconstruction error and adversarial losses to update the decoder parameters. In Figure 5.9 we show the reconstruction error curves for AAE, AAE_dGAN_1 and AAE_dGAN_2 and adversarial losses for dGANs and AAE_dGANs for training and validation set belonging to one of the cross-validation experiments. We observe that while the reconstruction errors decreases, the adversarial error converges indicating that the discriminators ability to discriminate between real and synthetic data points is countered by generators ability to confuse it. This trend is observed for both, training and validation sets, indicating that the learned parameters generalize well to data unseen during model training.

While the convergence of loss functions are a helpful tool to judge the training of GANs, we can't judge the quality of synthetic samples being generated. To this end we perform two experiments explained below:

- Using real samples as training set and synthetic data as test set. The objective of this experiment is to assess the similarity between real and synthetic data by using a model trained on real data to classify synthetic data. This would give us an idea about the quality of the synthetic data. However, it may so happen that all or majority of the generated samples belong to the same or only a few of the class (called mode collapse Arjovsky et al. [5]) or even if we get samples from all modes/classes there is not much variance within samples belonging to the same class i.e. they are not a good representation of the class to which they belong because it will be under represented.

- Using synthetic samples as training set and real data as test set. This experiment will give us an idea which model produces samples that are good representations of all the classes. This measure would reflect the diversity of synthetic data. In other words, the classifier trained using synthetic samples form a GAN model that's liable

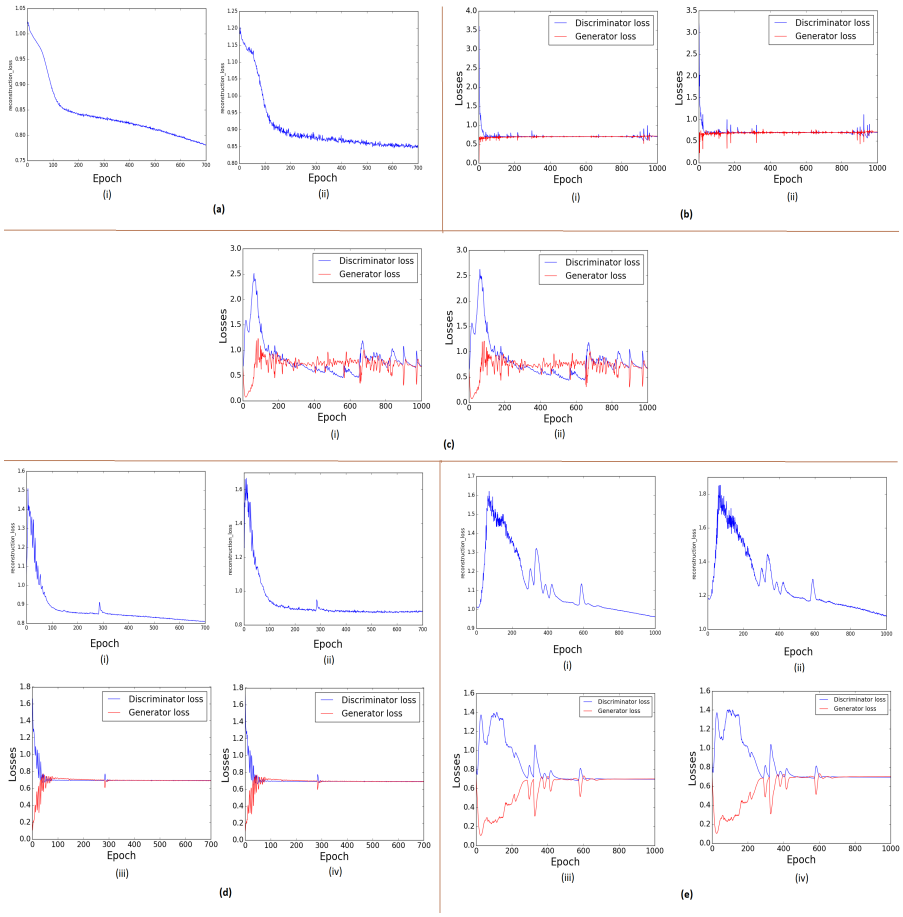

Figure 5.9: Reconstruction or adversarial errors (discriminator's (blue) and generator's (red) errors) for one of the cross-validation splits (a) AAE (b) dGAN_1 (c) dGAN_2 (d) AAE_dGAN_1 (e) AAE_dGAN_2. a(i), b(i), c(i), d(i,iii), e(i,iii) belong to training set while a(ii), b(ii), c(ii), d(ii,iv), e(ii,iv) belong to validation set

to mode collapse would perform poorly. Note that the test set in this case should have real data samples from all classes.

Single corpora setting

We perform the above experiments on IEMOCAP in a speaker independent cross-validation setting. For each cross-validation split, lets call the training set used to train the GAN model as set-1 and the validation set as set-2. The real samples mentioned in above three steps can either come from set-1 or set-2. We can use synthetic data points as

training set and use wither set-1 or set-2 as test set. Similarly, we can use set-1 or set-2 as training set and synthetic samples as test set. Finally we can append set-1 with synthetic data points to train a classifier and evaluate it on set-2. The purpose of using set-1 for experiments is to judge how the synthetic samples compare to the real samples that were used to train the GAN generating them. With set-2 we intend to determine how the synthetic samples compare to real samples obtained from utterances involving new speakers. We choose SVM as classifier and as before the SVM parameters (box-constraint and kernel) are tuned using an inner-cross validation on the training set. The metric used is UWA. We report the average UWA over the five cross-validation splits. We generate 6000 synthetic data-points which is almost the same number of data-points in the entire IEMOCAP set that we consider for our experiments. We show the results for the different train and test conditions in Table 5.3. Since, it is a 4-way classification chance accuracy should be $\frac{1}{4} = 25\%$. We note that the accuracies obtained using the dGAN models are either equal to or very close to that number which indicates that the data generated using those models are as good as sampling random points from 1582 dimensional space. The results are however more interesting for AAE based models which shows the importance of auto-encoders in the GAN based models. It can be seen that results obtained for set-1 are better than that for set-2. This is expected as set-1 was used to train the GAN models and hence the generated data should be more like set-1 than set-2 which has independent speakers than set-1. It can be observed that using synthetic data produced by AAE_dGAN_1 gives us better results than that produced by AAE which is probably because how the decoder is trained. While in case of AAE decoder parameters are updated based only on reconstruction loss, in case of AAE_dGAN_1 the parameters are updated based on an additional adversarial loss that determines how close it is to real data. We hypothesize that this extra update is what leads to better synthetic sample generation by AAE_dGAN_1 that also generalizes better for unseen speakers. Another interesting thing to note is the characteristic of syn-

Table 5.3: Cross-validation accuracies (%) obtained using different combinations of data-sets for training and evaluating a classifier. Set-1 refers to the training set of a cross-validation split used to train the GAN model while set-2 refers to the validation set.

	Train : Set-1 Test : Synthetic	Train : Synthetic Test : Set-1	Train : Set-2 Test : Synthetic	Train : Synthetic Test : Set-2
AAE	85.60	48.58	72.52	45.89
AAE_dGAN_1	**88.41**	49.91	**74.75**	46.96
AAE_dGAN_2	55.20	**52.24**	45.63	**51.58**
dGAN_1	25.26	25	25	25.28
dGAN_2	25	25	25	25

thetic data generated using AAE_dGAN_2. When used as test set, the classifiers trained on real data are unable to classify them as good as they classify samples generated from AAE and AAE_dGAN_1. However, training a classifier on synthetic data obtained from AAE_dGAN_2 performs better at classifying real data-points than a classifier trained on samples generated from AAE and AAE_dGAN_1. This indicates that data generated using the model AAE_dGAN_2 has more diverse samples than data generated using AAE and AAE_dGAN_1. The difference lies in the prior $p_{\mathbf{z}}$ from which points are sampled to be fed into the decoder to generate synthetic samples. While in case of AAE and AAE_dGAN_1 it is pre-defined to be a mixture of four Gaussians, in case of AAE_dGAN_2 the GAN model learns it during training. The coding space of AAE_dGAN_2 also has more dimensions (256) than AAE and AAE_dGAN_1 (2). This provides the decoder with a wider range of input samples which probably leads to diverse synthetic data-points.

Lower dimensional visualizations of synthetic data

To further analyze the synthetic data samples, we plot the t-SNE embeddings of real data and data generated from AAE, AAE_dGAN_1 and AAE_dGAN_2. From Figure **??** we observe that while majority of the synthetic data lies in the space spanned by the real data, we also see some samples in the space not spanned by the real data. We believe

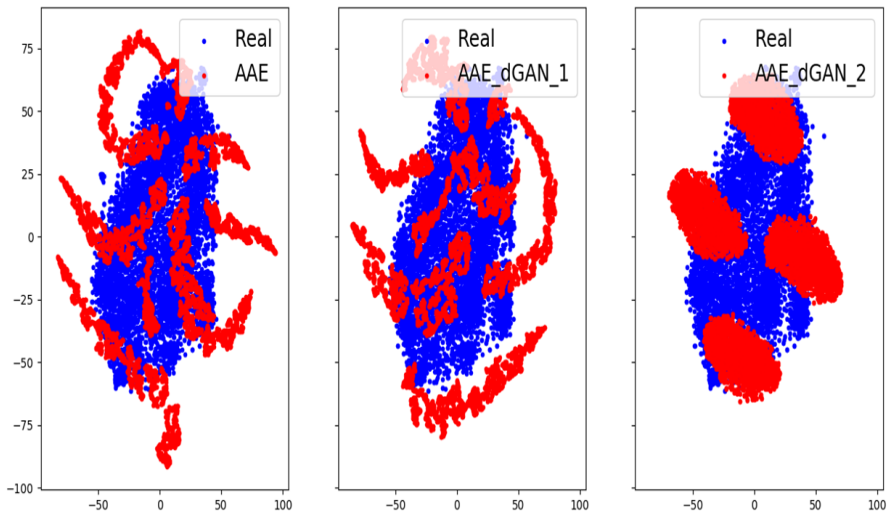

Figure 5.10: Comparison of t-SNE embeddings of real data with synthetic data generated using the three AAE based GAN models for one of the cross-validation splits of IMEOCAP.

such samples when used to train a classifier along with real data, will supplement the performance of the classifier by providing it with extra information. We look at this aspect briefly when we do cross-corpus experiments. We observe that the data generated by AAE models with $p_{\mathbf{z}}$ as mixture of four Gaussians has similar t-SNE plots and quite different from the data generated by AAE_dGAN_2 where $p_{\mathbf{z}}$ is Gaussian. We note that synthetic data generated using our models also come with their corresponding labels. In Figure **??** we observe the class-wise clustering of the synthetic samples with each color representing one class. We note that the clustering of samples obtained from AAE and AAE_dGAN_1 is quite different from that of obtained from AAE_dGAN_2. This is probably because of the way we are enforcing the condition that generated samples should fall into four distinct classes. While in case of AAE and AAE_dGAN_1 it is enforced by selecting a mixture of Gaussian prior with orthogonal means, in case of AAE_dGAN_2 it is enforced by maximizing the mutual information between generated data and the labels being fed to the code generator block which generates those samples. The distinct clusters seen in case of AAE_dGAN_2 suggest that it produces samples that have higher between-class discrim-

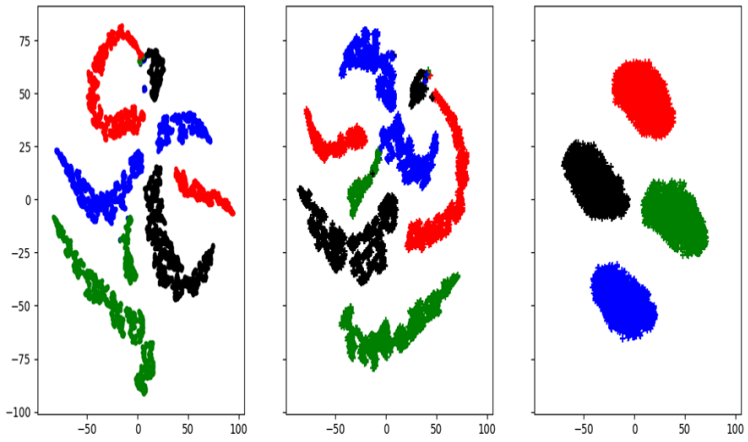

Figure 5.11: Class-wise clustering of the synthetic data generated using AAE (left), AAE_dGAN_1 (center) and AAE_dGAN_2 (right). Each of the four color represents a specific class

inability to AAE and AAE_dGAN_1. On the other hand in case of AAE_dGAN_2, the embeddings of the synthetic samples belonging to the same class are concentrated over a small circular region as compared to the embeddings of samples generated from the other two models suggesting lower within-class variability in case of AAE_dGAN_2. We also pass the synthetic samples through the trained encoder of AAE_dGAN_1. We re-iterate that the encoder of AAE_dGAN_1 was trained to project the higher dimensional feature vectors onto two dimensions with the coded space resembling mixture of four Gaussian with orthogonal means. The scatter plot for various models are shown in Figure 5.12. We see a significant amount of overlap across classes for data-points obtained from dGAN models which explains their substandard performance. Looking at the 'angry' (red) and 'sad' (green) synthetic samples obtained from AAE and AAE_dGAN_1, we observe that their variance is only along one dimension unlike the 'angry' and 'sad' samples obtained from AAE_dGAN_2. The 'happy' (magenta) samples also seem to have more variance in case of AAE_dGAN_2 compared to the other two models. The 'neutral' (blue) samples seem to be overlapping to a greater extent with 'happy' and 'sad' samples in case of AAE

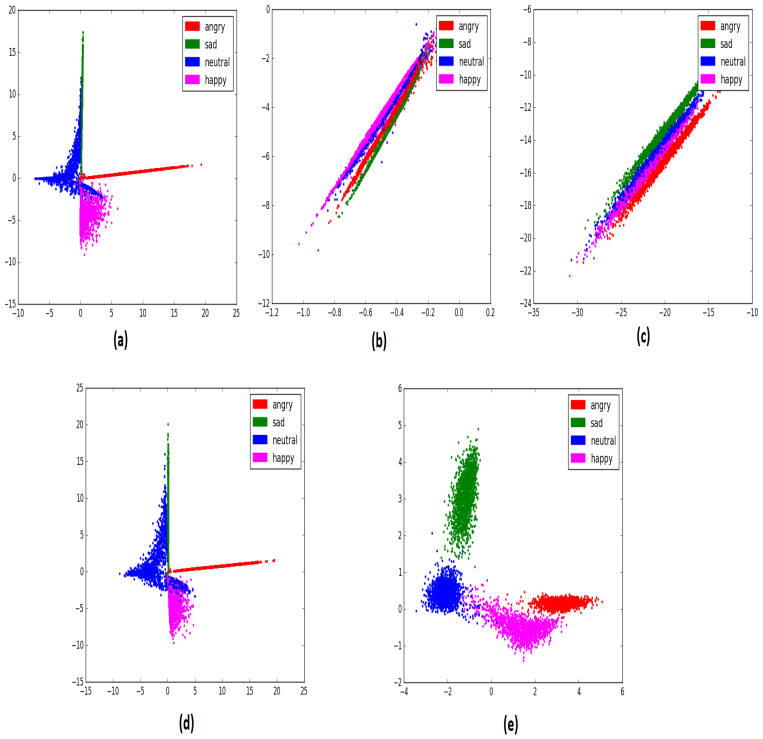

Figure 5.12: Scatter plot for the encoded points obtained for synthetic data generated using (a) AAE (b) dGAN_1 (c) dGAN_2 (d) AAE_dGAN_1 (e) AAE_dGAN_2

and AAE_dGAN_1 than AAE_dGAN_2. Hence, we hypothesize that a classifier trained with the more diverse synthetic samples obtained from AAE_dGAN_2 is better at classifying real data points than the other two AAE based models.

Cross corpus setting

Having studied the convergence of GAN architectures and evaluating the quality of synthetically generated samples produced by them in a single corpora setting, we now move to performing cross-corpus evaluations. The objective of this experiment is to investigate how well the synthetically generated samples generalize for classification tasks on an external corpus (as opposed to being applicable for only in-domain tasks). We generate the synthetic samples from GAN models trained using the entire IEMOCAP dataset. Since, synthetic

data obtained from trained dGAN models didn't give us better than chance accuracy in cross-validation experiments, for cross-corpus experiments we only train AAE based models. As before, we conduct two experiments (Table 5.4). First, use the synthetic dataset as training set and MSP-IMPROV as test set. This was followed with using MSP-IMPROV to train a classifier and evaluating it on synthetic data. We observe that evaluating a classifier that has been trained on MSP-IMPROV to classify the synthetic sets shows higher accuracies when the synthetic samples are generated using AAE_dGAN_1 and AAE_dGAN_2 that the AAE based models where decoder is receiving an extra adversarial error to update its parameters produces more generalizable samples. On the other hand, evaluating different classifiers which has been trained using synthetic samples generated from different AAE based GAN models perform almost similarly in classifying samples from MSP-IMPROV. This is probably because of the differences in distributions of utterances belonging to different classes in MSP-IMPROV and the synthetic data sets. While the synthetic data is balanced with respect to all the classes, MSP-IMPROV has more happy and neutral samples compared to angry and sad. Hence, the classifiers trained on different synthetic sets make similar mistakes when it comes to classifying the evaluation set leading to similar mean class-wise accuracies.

Finally we investigate the feasibility if using the synthetic feature vetors along with real data in low resource conditions. To start with, for our baseline classifier we train a SVM on IEMOCAP data and tested it on MSP-IMPROV. Then we added 600 synthetic data samples generated from the AAE based models and look for any improvement in accuracy. As Table 5.5 shows we indeed see a minor improvement in accuracy when synthetic data is used along with real data. Further experiments are needed to determine the applicability of synthetic data for training models in low resource conditions

Table 5.4: Cross-corpus accuracies obtained on MSP-IMPROV. Synthetic data is generated from GAN based models trained on IEMOCAP

	Train : MSP-IMPROV Test : Synthetic	Train : Synthetic Test : MSP-IMPROV
AAE	42.07	38.3
AAE_dGAN_1	**51.93**	38.61
AAE_dGAN_2	49.53	37.72

Table 5.5: Training using IEMOCAP with and without synthetic data and test on MSP-IMPROV. Note that we see aminor improvement in accuracy when synthetic data is used along with IEMOCAP. Each column represents the model from which synthetic data was generated.

Baseline	AAE	AAE_dGAN_1	AAE_dGAN_2
31.01	**32.08**	**32.52**	**32.32**

5.4 Conclusion and future work

Automatic emotion recognition is a problem of wide interest with implications on understanding human behavior and interaction. A typical emotion recognition system design involves use of high dimensional features on a curated dataset. We implemented GAN based models which can encode the higher dimensional features onto a lower dimensional space. At the same time, they are also generative models that can provide us with synthetic feature vectors. We establish that the code vectors learnt by the adversarial auto-encoder can be obtained in a low dimensional subspace without losing much class discriminability in the higher dimensional feature space. Having a pre-defined code space $p_{\mathbf{z}}$ with maximally separated components seem to encode the higher dimensional features more efficiently than if we try to learn the encoding space from data. We also observe that synthetically generated samples from these models do seem to retain relevant class information. Additionally from our experiments we found that updating the decoder parameters with adversarial error along with reconstruction error seems to generate "better" synthetic samples. It is encouraging to see these results given limited datasets that we have exper-

imented with (IEMOCAP has around only 7 hours of training data). With more data it is expected that GANs will be able to learn a more generalized distribution/manifold where the openSMILE feature vectors lie. The experiments show that a generator's job to estimate a more complex PDF from a simpler PDF is more complex than a discriminator's job which is to distinguish between fake and real samples. Hence, we had to incorporate tricks like updating the generator more times for a single update of discriminator or keeping the learning rate of generator more than that of a discriminator. All our GAN models enforce the conditio that the generated data should belong to four distinct classes. We observe that enforcing that clustering by implementing an infoGAN framework rather than specifying a Gaussian mixture prior with orthogonal means leads to generation of samples with inter-class variability but low within class variability. Cross corpus results showing an improvement in accuracy when real data is appended with synthetic data suggests that the synthetic samples could be generalizable across datasets with different priors. This opens up the possibility of using them in low resource conditions.

In future, we plan to investigate auto-encoder architectures that can be fed frame level features instead of utterance level features. We believe temporal dynamics of feature contours can lead to better classification results. It will also be interesting to study the effects of weighting the two updates that the decoder in AAE_dGAN_1 and AAE_dGAN_2 receives (one from reconstruction error of the auto-encoder and one from the adversarial error determining how well it can fool a discriminator into believing its output is are real datapoints). We can also investigate trying different loss functions to train the GAN based models. Wasserstein GAN [5] has been very popular in recent years. It uses earth-mover distance instead of Jensen-Shannon divergence (used by traditional GANs, Arjovsky et al. [4]) to learn a complex PDF from a simpler one and has been shown to be better at handling the mode collapse problem. Another interesting avenue could be to explore the usage of synthetic data generated as additional training samples in low resource settings. A more

detailed study needs to be done to determine their applicability in low resource conditions. For example, we can vary the amount of synthetic data in the training set and see how it affects accuracy. While too less of synthetic data probably wouldn't give us any advantage, using too much of synthetic data might not be ideal because we want the model to be trained mostly on real data. Finally, the GAN based architecture can also be used in analysis of other behavioral traits such as engagement [46] jointly with the emotional states.

Chapter 6: Future directions

This dissertation investigates the ways in which we can enhance the generalizability of speech emotion recognition models so that they perform better on evaluating the emotions of unseen speakers and cross-corpus tasks. We start off with discussing ways to improve discriminative models and then move on to talk about generative models. Our first approach to boost the performance of discriminative models using only audio features was based on regularization approaches and manifold learning techniques. We found that adversarial examples based manifold learning techniques can lead to more generalizable models. Furthermore, supervised approach led to better results than semi-supervised approaches. At the same semi-supervised approaches where the adversarial examples were found without using the label information did not lead to significant improvements. This could be because of the limited training data available to us (approximately 7 hours as compared to hundreds or thousands of hours used to build models used for commercial applications). However, audio features can only help so much. The reason behind this was because while audio features can help detect the arousal or intensity level of a speaker, they fail to do well in detecting the valence or pleasantness of an utterance. Since, the words used by a speaker can tell us about their sentiments, we leveraged the text transcriptions and used them to build a multi-modal system utilizing both audio and text features for emotion recognition. We observed that while audio features helped with arousal detection, text features indeed helped with valence detection with the multi-modal system out-performing both these uni-modal systems. Since, our end goal was to come up with a pipeline that utilizes only

speech as input, we used existing deep learning based ASR API's trained on thousands of hours of data to get the transcriptions. We then compared the performances of systems using audio features along with ASR transcriptions with that of using audio features with ground truth transcriptions. While we see a deterioration in performance as expected, using ASR transcriptions along with audio features as opposed to using only audio features gave us an absolute improvement of around 10% in our within-corpus cross-validation study. Moreover, we also saw improvements in cross-corpus study showing that these models are indeed generalizable across cross-corpus differences appearing due to recording conditions, label space, speakers and annotators. Having investigated ways to improve the generlizability of discriminative models, our next step was to focus on generative models. Generative models could be helpful in understanding the underlying process that generated the data. We can set the model parameters so that we can generate synthetic data that can potentially be used to train models in low resource conditions. We used auto-encoder and GAN based architectures to do so. Not only did we measure the quality of synthetic data generated, we also investigated their ability to encode higher dimensional feature vectors onto lower dimensions. The quality of these encodings was judged by how distinguishable they are as compared to the original higher dimensional raw feature vectors. We compared different architectures with different loss functions to generate synthetic feature vectors from samples obtained from lower dimensional priors. While one of them only used the reconstruction error as seen in auto-encoders, couple of them used only GAN based adversarial errors. Finally, we implemented architectures using both of them. Additionally we also investigated the effect of having a pre-defined prior vs having a data-driven prior from which synthetic samples are generated. We also defined metrics and visualizations to judge the quality of synthetic feature vectors generated. We found that the quality of the synthetic vectors obtained from architectures using only reconstruction error was better than those using only adversarial error. But it was best for the architectures whose parame-

ters were updated using both these errors showing that they were complimentary. From our within-corpus speaker independent cross-validation experiments we concluded that using a data-driven prior learned from training data produces more generalizable samples. However, with cross-corpus experiments it did not matter as much. The error functions used to update the parameters seemed to matter more in this case. We next list out some potential future directions we can extend this work to.

6.1 Appending low volume datasets with adversarial counterparts

More the amount of training data, better is the performance of deep learning algorithms. To that end, researchers in computer vision community resort to data augmentation techniques such as cropping, rotating and flipping input images etc (Wang and Perez [93]). In speech emotion recognition where limited data-sets are usually an issue, data augmentation techniques can be handy. One method could be augmenting the training data set with their respective adversarial counterparts. As mentioned in chapter 3, we can find the perturbation vector $\mathbf{r_i}^a$ for every data-point $\mathbf{x_i}$ using the equations 3.17 and 3.18. We can then add the perturbation vector to the actual data-point to get its adversarial counterpart. The ground truth label for the adversarial counterpart can be considered to be same as that of the actual datapoint following the smoothness assumption which states that the resulting conditional distribution $p(y_i|\mathbf{x_i})$ is desired to be smoothly varying over the inputs $\mathbf{x_i}$ (Huang et al. [60]). In chapter 3, instead of appending the dataset, we enforced this assumption by regularizing the loss function of the neural network. It would be interesting to see how appending a dataset with its adversarial counterparts instead of loss function regularization affects the training and generalization of the model. We can also compare the effect of augmenting the dataset by adding random perturbations to actual data-points rather than finding the adversarial perturbation. We think data augmentation using adversarial counterparts would

be more beneficial than augmentation by adding random noise vectors. We saw a similar trend in Figure 3.4 with the model performing better when regularization was carried out by adding adversarial perturbation as opposed to random perturbation to the input datapoint.

Another interesting experiment could be to generate such adversarial examples using a GAN based framework as proposed by Xiao et al. [126] and augment the training dataset using such examples. The purpose of [126] was however not data augmentation but to generate adversarial examples of high perceptual quality that a traditional classifier trained on real unperturbed images would fail to classify. They compare their method with other methods that generate such adversarial examples. In their approach, instead of computing the perturbation vector $\mathbf{r_i}^a$ for every data-point $\mathbf{x_i}$ explicitly, they train a GAN to do so. While the generator generates the perturbation given an input image, the discriminator tries to distinguish between the input image and the perturbed image obtained by adding the perturbation to the input. Training this GAN architecture would make the perturbed images look like real images. At the same time, they also include another term to their loss function that encourages these perturbed images to be miss-classified (not being classified as input image's class) by a trained classifier. This can be done by maximizing the distance between the prediction of the classifier and the ground truth. Note that the parameters of the trained classifier aren't updated while back-propagation. Only the parameters of the generator is updated during this process. This would encourage the generated images to be an adversary of the actual image. They trained their network using image datasets and claim once the generator is trained, it can generate perturbations efficiently for any image. They show perturbed images which look exactly like the images they have been generated from which is miss-classified by a classifier trained only un-augmented dataset. It would be interesting to investigate the effect of such a data augmentation technique for speech emotion recognition.

6.2 Emotion recognition on real world datasets

The experiments presented in this thesis till now has been on acted datasets. While it is easier to collect it would be more realistic to train and evaluate our models on real word datasets. We have done a pilot study on a couple of call center conversations provided to us by Hughes telecommunications that involved affect detection based on audio and sentiment analysis based on text to analyze how the calls went. For audio based affect level detection, we trained two convolutional neural network based architectures, one for valence and another for arousal detection using 3 second speech samples from two acted corpora available to us namely, IEMOCAP and MSP-IMPROV. The arousal and valence levels of the resulting utterances have been annotated by various annotators on a Likert scale of 1-5, with their average being considered as the ground truth rating. For our experiments, speech samples with arousal/valence rating < 2.5 were labelled as low arousal/valence and anything with arousal/valence rating > 2.5 as high arousal/valence. In total we have approximately 7 hours of speech that was used for training the models and performing a binary classification between high/low arousal or high/low valence with the two soft-max activated neurons in the final layer of the model giving us the probability of a speech sample belonging to each of the two classes . We performed a speaker independent cross-validation (i.e. there is no overlap of speakers between training and validation set) to fix the hyper-parameters of the network. The cross-validation accuracy obtained for arousal detection is around 85%, whereas for valence it is only around 63%. These results are consistent with the findings in chapter 4 where we have shown that models trained using audio data work better for arousal level detection than for valence level detection.

The first step in analyzing the Hughes data consisted of speaker diarization which was done by hand for the purpose of this study. Second, the data for the customers and the representatives was divided into 3 second chunks that were then passed through the model

separately. We extract Mel filterbank based spectrograms for the utterances with a frame interval of 0.5 seconds. This data is fed to our pre-trained models to obtain the probability of the utterance being high arousal/high valence. Since, there were no ground truth annotations of the call center conversations for their arousal and valence levels, our analysis are based on our own perceptions of affect and sentiment. The representatives in the conversations were always calm. An example is shown in Figure 6.1 where the spectrogram and the arousal plot are shown for a 3 second segment of speech from the representative in one of the phone calls. Note that there appears to be a shift in the data plot vs the audio because each point in the former is plotted at the center of the 3 second window. That is, a data point at 1.5 seconds shows the value for a window ranging from 0-3 seconds. As can be seen, the probability of being aroused never exceeds 0.2, indicating that the representative was calm while speaking, a result we have verified by listening to the utterances. As in the example above, the representatives in both of the calls provided are perceived to be calm by us throughout the call and the arousal values we get confirm this impression except in 6 out of 1087 3-second turns when the probability of being aroused exceeded 0.5 (for these cases, the values always lie between 0.55 to 0.77). This results gives an accuracy of 99.4% in detecting low arousal calls belonging to the representatives. Being able to detect that the representatives are calm based on the audio is especially important. The speech of the representative in one of the calls was not recognized correctly by Google's ASR API for most of the call, presumably because of his accent. It is likely the case that many representatives will have an accent which may make textual analysis ineffective.

In Figure 6.2, we show an example from the customer in the 25 minute phone call who was sometimes annoyed. For this particular segment, she is annoyed about her internet bill increasing and we can see the arousal detector evaluating it to be a high arousal utterance towards the end of the utterance. Listening to the turn confirms the fact that her voice indeed becomes louder and annoyed towards the end of the conversation. The transcription

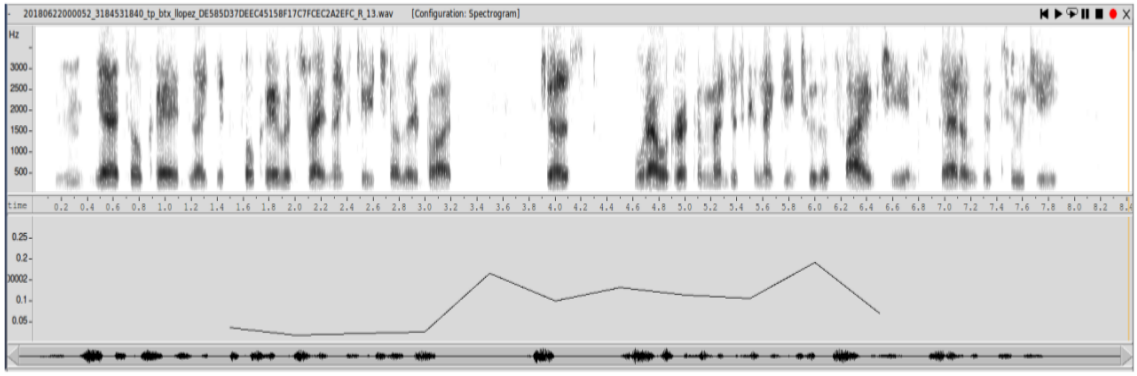

Figure 6.1: Spectrogram and plot of arousal probability of a call center representatives utterance. The higher the value, the more aroused is the speaker. Its fairly low values indicate the person remained calm during the call

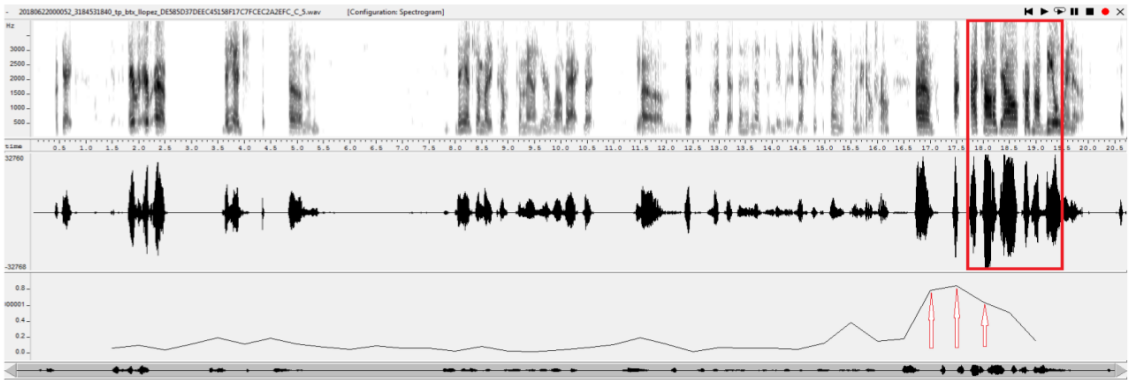

Figure 6.2: Customer's response where the arousal tends to rise towards the end of her sentence. The arousal values are marked with an arrow at the bottom for the boxed region

of the audio with the more aroused portion written in bold is: "Okay (stammers) they had somebody come out here (stammers) that was just a thing (unclear), **I called yall coz my bill went up**". Altogether, the customer in the 25-minute call had 28 conversation sides that were at least 3 seconds long. Out of the 28 utterances, 14 were judged by us as being low arousal, while we felt the remaining had some portions that we would consider high arousal. Of the 14 low arousal calls, our model predicted 12 of them correctly with it showing low arousal probabilities throughout the utterance. For one of the other two calls, it is the case that while the customer does not appear to be aroused, she was clearly bothered and it is this part of the utterance where the arousal detector had high values. Of the 14

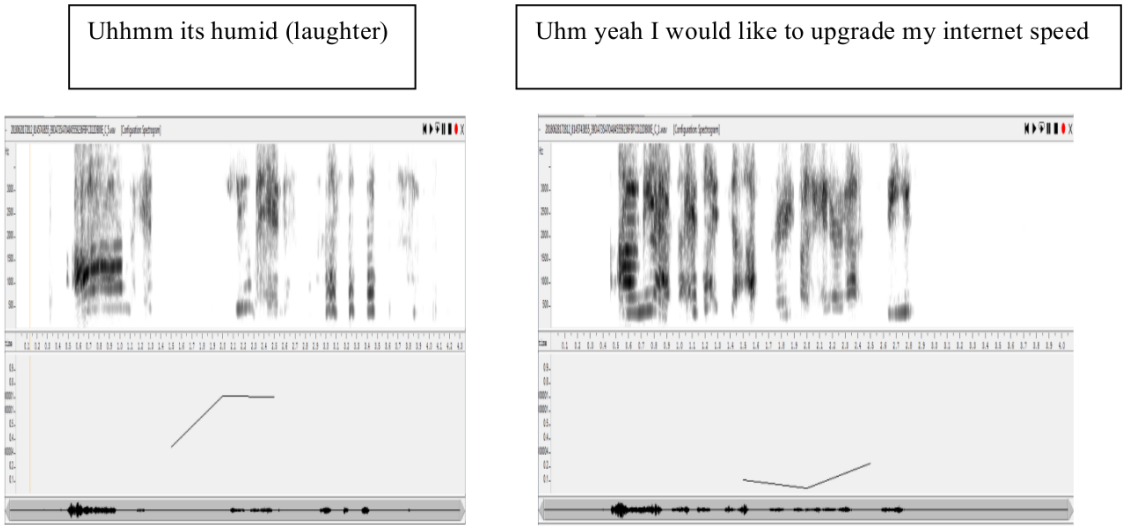

Figure 6.3: Valence probability plot for two utterances for when the customer laughs (left) vs when she doesn't (right). Higher value shows the person is more pleasant. Note that there is an increase in valence in case of laughter.

conversation sides that had certain high arousal parts within them, our model failed to find any high arousal regions in four of them. In all of these four utterances, the relevant aroused portion was only about 3 seconds long which means the algorithm had little data to predict the arousal values. For the second phone call, the arousal detection of half of 6 conversation sides for the customer confirm our perception. The arousal detector gives high values in the beginning of the call when she introduces herself and during an utterance when she laughs. On the other hand, the arousal value declines when she faintly says "okay thank you". However, in the remaining three conversation sides, the model mistakenly detects high arousal during portions of the utterances. The performance of valence detector wasnt as good as we would like. However, there was an interesting result that we show in Figure 6.3 where we contrast two situations, one where the valence profile rises and one where the valence profile stays low. In the one to the left, the valence profile rises when she starts to laugh. In the one on the right, she is making a statement of why she has made the call. Transcriptions are provided in the boxes. Overall we felt the arousal detector performed

reasonably well especially given it was trained on acted and improvised speech. Thus, we feel it would be significantly more accurate if it is trained on actual call center conversations. This should also be the case for the valence detector and we believe it is worth exploring whether using matched data would improve its performance to a level where it will be useful. Furthermore, we also obtained ASR transcriptions from Google's API and used it for sentiment analysis of the conversations. We used term frequency-inverse document frequency (Ramos [97]) also called tf-idf that depicts the importance of a word in a corpus as the value of it is increased proportionally to the number of occurrences of that word in the document (in our case, one side of the conversation), but will be offset by how many documents contains that particular word. For our purpose, a document is one side of a telephone conversation. Then we used the SentiWordNet corpus (Esuli and Sebastiani [33]) in Pythons Natural Language Toolkit to evaluate the sentiment of a particular word in the given context by assigning them a positivity or negativity score. The overall positive and negative score for a response was computed by weighing each of its constituent word's sentiment score by their tf-idf score and adding them. The sentiment analysis metric was computed by taking the difference between the positive and negative scores of the response. If the metric is less than 0, the response is classified as having a negative sentiment, and as having a positive sentiment otherwise and the magnitude gives an idea of the strength of that overall sentiment. Based on these values, we concluded that the customer in one of the calls had a negative sentiment while the representative in that conversation had a slightly positive sentiment. From listening, we feel that the representative handled the situation well. For the other call, both the customer and the representative had positive sentiment scores indicating the call went through without a hiccup which was indeed the case.

Our next step in this study is to use in-domain call center conversations to train the affect recognition models. We have been provided with 75 phone conversations which we had diarized using pyAudioAnalysis [39] and chunked into 5 second segments. Currently we

are in the process of getting annotations that we can use as ground truth labels to train the emotion recognition models. Apart from getting the annotations for arousal and valence, we are also asking the annotators to categorize the utterances into one of the four classes relevant to the task at hand namely angry, calm, frustrated and pleasant. Moreover we have also asked the annotators if they can hear just one speaker or they can hear multiple speakers in that 5 second segment which can be used to evaluate the diarization system.

6.3 Connecting depression detection and emotion recognition

In [99], we talk about an Average Magnitude Difference Function (AMDF) based feature that quantifies the voice quality features such as jitter (change in pitch across pitch periods), shimmer (change in amplitude across pitch periods) and breathiness (amount of aperiodicity in voiced regions) that showed promise for the task of depression detection. We believe these features capture the psychomotor retardation present in a depressed persons speech. It was found that in general, depressed voice has more jitter, shimmer and sounds, more breathy. The dataset we used was the Mundt database [84] which include speech data collected from 35 physician-referred patients undergoing treatment for depression. The patients were assessed weekly once over a period of 6 weeks. We used both the sustained vowel sounds (four vowels a, i, ae, u held for 4-5 seconds each) and the free flowing utterances where the patients talk about their emotional state, physical state and their ability to function for our study. The decision whether a person was depressed or not was determined by using the Hamilton Depression (HAM-D) score [50] which ranges between 0-26, with higher score implying more severe depression. Sessions with scores greater than 17 were considered to be ones where the patient was depressed and sessions with score less than 7 were considered to be ones where the patient was not depressed, while the session with intermediate scores were considered ones where the subjects mental

state was ambiguous and so their data was not included in the study as has also been done by Helfer et al. [56]. We only used the data from six patients who underwent a change in their depressive state which was reflected by a decrease in their HAM-D scores. Since, we didnt have a lot of data we build a speaker dependent classifier i.e. the training and test set had speaker overlap. We used support vector machines (SVM) as classifiers. We used voiced frames from the free flowing utterances for training the SVMs and reported an utterance wise classification accuracy of 77.8%. At the same time, it was also found that the AMDF based features can be useful for the task of emotion classification by Ko and Espy-Wilson [67]. Using the emotion database from USC that also contains electromagnetic articulography measurements [72] it was shown that using the AMDF based features together with Mel Frequency Cepstrum Coefficients(MFCC) and pitch related features led to a better performance in comparison to the openSMILE toolkits emobase feature set. It not only improved the overall accuracy by 3.3% but also greatly reduced the feature space by 72%. Caruana [17] defines two tasks to be similar if they use the same features to make a decision. Based on the results we got from the above two experiments we can conclude that the task of emotion recognition and depression detection are similar because the same AMDF features helped us with both the classification tasks.

In another study conducted by Gupta et al. [49], it was found that incorporating depression severity as a parameter in Deep Neural Networks (DNNs) by altering the activation functions using the depression score show improvements in arousal and valence prediction compared to a DNN with a vanilla *tanh* activation function. We performed experiments on affect prediction using the Audio-Visual Depressive language Corpus [120], which involves subjects with varying degree of depression. The dataset consists of both free flowing utterances where the participant is answering a question and read speech where the person is reading a pre-defined script. Each video is continuously rated for three affective dimensions of valence, arousal and dominance at a frame rate of 30 Frames Per Second by a set of 3-5

annotators. The final ground truth affect ratings are computed as the frame-wise mean over the annotator ratings for a given session. The subjects in the sessions also complete the standardized self-assessment based Beck Depression Inventory-II (BDI-II) questionnaire [9]. The score ranges between 0-63, with a higher score implying more severe depression. Our results on both the sessions show that using depression severity can improve arousal and valence prediction, thereby suggesting a link between the two tasks.

Since, the tasks of emotion recognition and depression detection have similar characteristics we hypothesize that we can use the information from one task to help with the task of the other. Our goal is to use a model's knowledge in recognizing emotions/affect state to better estimate the presence/severity of depression. One such approach could be to use transfer learning. The core idea of transfer learning is utilizing the knowledge learned from solving one problem to solve a different but related problem. While the concept of transfer learning can be useful to build more generalizable models, it can also assist us with related tasks. For example, Razavian et al. [110] reported state of the art results when they used a convolutional neural network (CNN) trained on Imagenet dataset for object classification to classify bird images from Caltech-UCSD Birds (CUB) 200-2011 dataset. Note that birds formed one of the categories in the Imagenet dataset. Pre-training has also applications in audio music classification (Van den Oord et al. [121]) and for natural language processing tasks (Mou et al. [83]). Now we describe a typical transfer learning set up. We have a dataset A and a task for which it was collected T_A. We have another dataset B and corresponding task T_B. We aim to use the knowledge that the model has gained while learning T_A to perform T_B. The steps involved in pre-training are:

1. We have a neural network model M initialized randomly.

2. Pre-training: We have a large dataset A on which we train M for task T_A.

3. We have the dataset of interest B and we are interested in Task T_B. Instead of initializing the weights randomly and training on B, we take the model M already trained on A as our

initialization.

4. Fine-tuning: We fine-tune the weights by training on B. The fine-tuning can be done for all/some of the hidden layers.

Since, we believe emotion recognition and depression detection are related, we can pre-train a network to recognize emotions/affect and then fine-tune the weights of the layers on the depression dataset. So dataset A is a dataset which has the emotion/affect labels available and task T_A for which the model M is trained is emotion/affect recognition. B is the dataset of interest which in our case is the depression database and hence task T_B is depression detection. One potential roadblock we could face in such an approach is the size of pre-training dataset. Usually pre-training dataset A is large compared to dataset of interest B. Researchers working in object detection problems have access to huge amounts of data. For example, Imagenet has 1.2 million images in it. But it is difficult to get such a huge dataset for the task of emotion recognition/depression detection. For, example, IEMOCAP database that we have used for our experiments has around 10 hours of data (around 10,000 utterances) if we consider all 15 emotional categories. It would be interesting to explore how good of a pre-trained model we can achieve from 10 hours of emotional data. We could combine several emotion recognition datasets to mitigate this problem. We can pre-train using all of them together or using them sequentially one at a time. Apart from pre-training we can also explore multi-task learning (MTL) frameworks that leverages the use of auxiliary tasks to improve the performance of a model for a target task. One such neural network based MTL framework could be where we train the same neural network for emotion classification and depression detection simultaneously with a different output layer for each of these tasks while the other layers are shared. Ruder [98] discusses some MTL framework that could be worth exploring.

Bibliography

[1] Mohammed Abdelwahab and Carlos Busso. Supervised domain adaptation for emotion recognition from speech. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5058–5062. IEEE, 2015.

[2] Mohammed Abdelwahab and Carlos Busso. Ensemble feature selection for domain adaptation in speech emotion recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5000–5004. IEEE, 2017.

[3] Mohammed Abdelwahab and Carlos Busso. Domain adversarial for acoustic emotion recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(12):2423–2435, 2018.

[4] Martín Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *CoRR*, abs/1701.04862, 2017.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[6] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. *ICML unsupervised and transfer learning*, 27(37-50):1, 2012.

[7] Rainer Banse and Klaus R Scherer. Acoustic profiles in vocal emotion expression. *Journal of personality and social psychology*, 70(3):614, 1996.

[8] Tanja Bänziger and Klaus R Scherer. The role of intonation in emotional expressions. *Speech communication*, 46(3):252–267, 2005.

[9] Aaron T Beck, Robert A Steer, and Gregory K Brown. Beck depression inventory-ii. *San Antonio*, 78(2):490–498, 1996.

[10] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[11] JM Bernardo, MJ Bayarri, JO Berger, AP Dawid, D Heckerman, AFM Smith, and M West. Generative or discriminative? getting the best of both worlds. *Bayesian statistics*, 8(3):3–24, 2007.

[12] Sahar E Bou-Ghazale and John HL Hansen. A comparative study of traditional and newly proposed features for recognition of speech under stress. *IEEE Transactions on speech and audio processing*, 8(4):429–442, 2000.

[13] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335, 2008.

[14] Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 205–211. ACM, 2004.

[15] Carlos Busso, Angeliki Metallinou, and Shrikanth S Narayanan. Iterative feature normalization for emotional speech detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5692–5695. IEEE, 2011.

[16] Carlos Busso, Srinivas Parthasarathy, Alec Burmania, Mohammed AbdelWahab, Najmeh Sadoughi, and Emily Mower Provost. Msp-improv: An acted corpus of dyadic interactions to study emotion perception. *IEEE Transactions on Affective Computing*, (1):67–80, 2017.

[17] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[18] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[19] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[20] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[21] Vladimir Chernykh, Grigoriy Sterling, and Pavel Prihodko. Emotion recognition from speech with recurrent neural networks. *arXiv preprint arXiv:1701.08071*, 2017.

[22] François Chollet et al. Keras, 2015.

[23] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585, 2015.

[24] Neri E Cibau, Enrique M Albornoz, and Hugo L Rufiner. Speech emotion recognition using a deep autoencoder. *Anales de la XV Reunion de Procesamiento de la Informacion y Control*, 16:934–939, 2013.

[25] Roddy Cowie, Ellen Douglas-Cowie, Nicolas Tsapatsoulis, George Votsis, Stefanos Kollias, Winfried Fellenz, and John G Taylor. Emotion recognition in human-computer interaction. *IEEE Signal processing magazine*, 18(1):32–80, 2001.

[26] Nicholas Cummins, Stefan Scherer, Jarek Krajewski, Sebastian Schnieder, Julien Epps, and Thomas F Quatieri. A review of depression and suicide risk assessment using speech analysis. *Speech Communication*, 71:10–49, 2015.

[27] Jun Deng, Zixing Zhang, Florian Eyben, and Björn Schuller. Autoencoder-based unsupervised domain adaptation for speech emotion recognition. *IEEE Signal Processing Letters*, 21(9):1068–1072, 2014.

[28] Jun Deng, Zixing Zhang, Erik Marchi, and Björn Schuller. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 511–516. IEEE, 2013.

[29] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[30] Moataz El Ayadi, Mohamed S Kamel, and Fakhri Karray. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*, 44(3):572–587, 2011.

[31] Moataz MH El Ayadi, Mohamed S Kamel, and Fakhri Karray. Speech emotion recognition using gaussian mixture vector autoregressive models. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–957. IEEE, 2007.

[32] Sefik Emre Eskimez, Zhiyao Duan, and Wendi Heinzelman. Unsupervised learning approach to feature analysis for automatic speech emotion recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5099–5103. IEEE, 2018.

[33] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, volume 6, pages 417–422. Citeseer, 2006.

[34] Florian Eyben, Klaus R Scherer, Björn W Schuller, Johan Sundberg, Elisabeth André, Carlos Busso, Laurence Y Devillers, Julien Epps, Petri Laukka, Shrikanth S Narayanan, et al. The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing. *IEEE Transactions on Affective Computing*, 7(2):190–202, 2016.

[35] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 835–838. ACM, 2013.

[36] Daniel Joseph France, Richard G Shiavi, Stephen Silverman, Marilyn Silverman, and M Wilkes. Acoustical properties of speech as indicators of depression and suicidal risk. *IEEE transactions on Biomedical Engineering*, 47(7):829–837, 2000.

[37] Bo Geng, Dacheng Tao, Chao Xu, Linjun Yang, and Xian-Sheng Hua. Ensemble manifold regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1227–1233, 2012.

[38] Sayan Ghosh, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. Learning representations of affect from speech. *arXiv preprint arXiv:1511.04747, International Conference on Learning Repre-sentations (ICLR) 2016 Workshop*, 2015.

[39] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12):e0144610, 2015.

[40] Christer Gobl, Ailbhe Ni, et al. The role of voice quality in communicating emotion, mood and attitude. *Speech communication*, 40(1):189–212, 2003.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[43] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1319–1327. PMLR, 17–19 Jun 2013.

[44] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572, ICLR 2015*, 2014.

[45] Michael Grimm, Kristian Kroschel, and Shrikanth Narayanan. The vera am mittag german audio-visual emotional speech database. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 865–868. IEEE, 2008.

[46] Rahul Gupta, Daniel Bone, Sungbok Lee, and Shrikanth Narayanan. Analysis of engagement behavior in children during dyadic interactions using prosodic cues. *Computer Speech & Language*, 37:47–66, 2016.

[47] Rahul Gupta, Nikolaos Malandrakis, Bo Xiao, Tanaya Guha, Maarten Van Segbroeck, Matthew Black, Alexandros Potamianos, and Shrikanth Narayanan. Multimodal prediction of affective dimensions and depression in human-computer interactions. In *Proceedings of the 4th International Workshop on Audio/Visual Emotion Challenge*, pages 33–40. ACM, 2014.

[48] Rahul Gupta, Saurabh Sahu, Carol Espy-Wilson, and Shrikanth Narayanan. Semi-supervised and transfer learning approaches for low resource sentiment classification. *arXiv preprint arXiv:1806.02863*, ICASSP 2018.

[49] Rahul Gupta, Saurabh Sahu, Carol Y Espy-Wilson, and Shrikanth S Narayanan. An affect prediction approach through depression severity parameter incorporation in neural networks. In *INTERSPEECH*, pages 3122–3126, 2017.

[50] Max Hamilton. A rating scale for depression. *Journal of Neurology, Neurosurgery & Psychiatry*, 23(1):56–62, 1960.

[51] Jing Han, Zixing Zhang, Zhao Ren, Fabien Ringeval, and Björn Schuller. Towards conditional adversarial training for predicting emotions from speech. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6822–6826. IEEE, 2018.

[52] John HL Hansen and Douglas A Cairns. Icarus: Source generator based real-time recognition of speech in noisy stressful and lombard effect environments. *Speech Communication*, 16(4):391–422, 1995.

[53] Sanaul Haq, Philip JB Jackson, and J Edge. Speaker-dependent audio-visual emotion recognition. In *AVSP*, pages 53–58, 2009.

[54] Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. Conversational memory network for emotion recognition in dyadic dialogue videos. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2122–2132, 2018.

[55] Wilbert Jan Heeringa. *Measuring dialect pronunciation differences using Levenshtein distance*. PhD thesis, Citeseer, 2004.

[56] Brian S Helfer, Thomas F Quatieri, James R Williamson, Daryush D Mehta, Rachelle Horwitz, and Bea Yu. Classification of depression state based on articulatory precision. In *Interspeech*, pages 2172–2176, 2013.

[57] Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John R Hershey, Tim K Marks, and Kazuhiko Sumi. Attention-based multimodal fusion for video description. In *Proceedings of the IEEE international conference on computer vision*, pages 4193–4202, 2017.

[58] Hao Hu, Ming-Xing Xu, and Wei Wu. Gmm supervector based svm with spectral features for speech emotion recognition. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–413. IEEE, 2007.

[59] Che-Wei Huang and Shrikanth S Narayanan. Attention assisted discovery of sub-utterance structure in speech emotion recognition. In *INTERSPEECH*, pages 1387–1391, 2016.

[60] Gao Huang, Shiji Song, Jatinder ND Gupta, and Cheng Wu. Semi-supervised and unsupervised extreme learning machines. *IEEE transactions on cybernetics*, 44(12):2405–2417, 2014.

[61] Hai Jin, Laurence T Yang, and Jeffrey J-P Tsai. *Ubiquitous Intelligence and Computing: Third International Conference, UIC 2006, Wuhan, China, September 3-6, 2006, Proceedings*, volume 4159. Springer, 2006.

[62] Jaebok Kim, Gwenn Englebienne, Khiet P Truong, and Vanessa Evers. Towards speech emotion recognition" in the wild" using aggregated corpora and deep multi-task learning. *arXiv preprint arXiv:1708.03920, Interspeech*, 2017.

[63] Jaebok Kim, Khiet P Truong, Gwenn Englebienne, and Vanessa Evers. Learning spectro-temporal features with 3d cnns for speech emotion recognition. In *Affective Computing and Intelligent Interaction (ACII), 2017 Seventh International Conference on*, pages 383–388. IEEE, 2017.

[64] Jangwon Kim, Sungbok Lee, and Shrikanth S Narayanan. An exploratory study of manifolds of emotional speech. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5142–5145. IEEE, 2010.

[65] Yelin Kim and Emily Mower Provost. Emotion recognition during speech using dynamics of multiple regions of the face. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(1s):25, 2015.

[66] Dennis H Klatt and Laura C Klatt. Analysis, synthesis, and perception of voice quality variations among female and male talkers. *the Journal of the Acoustical Society of America*, 87(2):820–857, 1990.

[67] Yi-Chun Ko. A study of feature sets for emotion recognition from speech signals. Master's thesis, 2015.

[68] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. Deap: A database for emotion analysis; using physiological signals. *IEEE Transactions on Affective Computing*, 3(1):18–31, 2012.

[69] Duc Le and Emily Mower Provost. Emotion recognition from spontaneous speech using hidden markov models with deep belief networks. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 216–221. IEEE, 2013.

[70] Chi-Chun Lee, Emily Mower, Carlos Busso, Sungbok Lee, and Shrikanth Narayanan. Emotion recognition using a hierarchical binary decision tree approach. *Speech Communication*, 53(9):1162–1171, 2011.

[71] Jinkyu Lee and Ivan Tashev. High-level feature representation using recurrent neural network for speech emotion recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[72] Sungbok Lee, Serdar Yildirim, Abe Kazemzadeh, and Shrikanth Narayanan. An articulatory study of emotional speech production. In *Interspeech*, pages 497–500, 2005.

[73] Xingfeng Li and Masato Akagi. A three-layer emotion perception model for valence and arousal-based detection from multilingual speech. *Proc. Interspeech 2018*, pages 3643–3647, 2018.

[74] Mark Liberman. Emotional prosody speech and transcripts. *http://www. ldc. upenn. edu/Catalog/CatalogEntry. jsp? catalogId= LDC2002S28*, 2002.

[75] Yi-Lin Lin and Gang Wei. Speech emotion recognition based on hmm and svm. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 8, pages 4898–4901. IEEE, 2005.

[76] Na Liu, Yuan Zong, Baofeng Zhang, Li Liu, Jie Chen, Guoying Zhao, and Junchao Zhu. Unsupervised cross-corpus speech emotion recognition using domain-adaptive subspace learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5144–5148. IEEE, 2018.

[77] Reza Lotfian and Carlos Busso. Curriculum learning for speech emotion recognition from crowdsourced labels. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):815–826, 2019.

[78] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[79] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644, ICLR 2016*.

[80] Qirong Mao, Wentao Xue, Qiru Rao, Feifei Zhang, and Yongzhao Zhan. Domain adaptation for speech emotion recognition by sharing priors between related source and target classes. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2608–2612. IEEE, 2016.

[81] Angeliki Metallinou, Sungbok Lee, and Shrikanth Narayanan. Decision level combination of multiple modalities for recognition and analysis of emotional expression. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2462–2465. IEEE, 2010.

[82] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[83] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics*, pages 479–489, 2016.

[84] James C Mundt, Peter J Snyder, Michael S Cannizzaro, Kara Chappie, and Dayna S Geralts. Voice acoustic measures of depression severity and treatment response collected via interactive voice response (ivr) technology. *Journal of neurolinguistics*, 20(1):50–64, 2007.

[85] Iain R Murray and John L Arnott. Toward the simulation of emotion in synthetic speech: A review of the literature on human vocal emotion. *The Journal of the Acoustical Society of America*, 93(2):1097–1108, 1993.

[86] Michael Neumann et al. Cross-lingual and multilingual speech emotion recognition on english and french. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5769–5773. IEEE, 2018.

[87] Michael Neumann and Ngoc Thang Vu. Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech. *arXiv preprint arXiv:1706.00612, Interspeech*, 2017.

[88] Joy Nicholson, Kaxuhiko Takahashi, and Ryohei Nakatsu. Emotion recognition in speech using neural networks. In *Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on*, volume 2, pages 495–501. IEEE, 1999.

[89] Tin Lay Nwe, Say Wei Foo, and Liyanage C De Silva. Speech emotion recognition using hidden markov models. *Speech communication*, 41(4):603–623, 2003.

[90] Joseph D O'Connor. Intonation of colloquial english. 1984.

[91] AM Oster and Arne Risberg. The identification of the mood of a speaker by hearing impaired listeners. *SLT-Quarterly Progress Status Report*, 4:79–90, 1986.

[92] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[93] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[94] Fan Ping, Jiang Dongmei, Wang Fengna, Ravyse Ilse, and Sahli Hichem. Manifold analysis for subject independent dynamic emotion recognition in video sequences. In *Image and Graphics, 2009. ICIG'09. Fifth International Conference on*, pages 896–901. IEEE, 2009.

[95] Rohit Prabhavalkar, Tara N Sainath, Bo Li, Kanishka Rao, and Navdeep Jaitly. An analysis of attention in sequence-to-sequence models,. In *Proc. of Interspeech*, 2017.

[96] Yu Qian, Li Ying, and Jia Pingping. Speech emotion recognition using supervised manifold learning based on all-class and pairwise-class feature extraction. In *Conference Anthology, IEEE*, pages 1–5. IEEE, 2013.

[97] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

[98] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[99] S Sahu and C. Y. Espy-Wilson. Speech features for depression detection. In *Interspeech*, pages 1928–1932, 2016.

[100] Saurabh Sahu, Rahul Gupta, and Carol Espy-Wilson. On enhancing speech emotion recognition using generative adversarial networks. *arXiv preprint arXiv:1806.06626, Interspeech*, 2018.

[101] Saurabh Sahu, Rahul Gupta, Ganesh Sivaraman, Wael AbdAlmageed, and Carol Espy-Wilson. Adversarial auto-encoders for speech based emotion recognition. In *Interspeech*, pages 1243–1247, 2017.

[102] Michelle Hewlett Sanchez, Gokhan Tur, Luciana Ferrer, and Dilek Hakkani-Tür. Domain adaptation and compensation for emotion detection. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[103] Klaus R Scherer. Vocal affect expression: a review and a model for future research. *Psychological bulletin*, 99(2):143, 1986.

[104] Maria Schubiger. *English intonation, its form and function*. M. Niemeyer Verlag, 1958.

[105] Bjorn Schuller, Anton Batliner, Stefan Steidl, and Dino Seppi. Emotion recognition from speech: putting asr in the loop. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 4585–4588. IEEE, 2009.

[106] Björn Schuller, Ronald Müller, Manfred Lang, and Gerhard Rigoll. Speaker independent emotion recognition by early fusion of acoustic and linguistic features within ensembles. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[107] Björn Schuller, Gerhard Rigoll, and Manfred Lang. Speech emotion recognition combining acoustic features and linguistic information in a hybrid support vector machine-belief network architecture. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–577. IEEE, 2004.

[108] Björn W Schuller, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian A Müller, Shrikanth S Narayanan, et al. The interspeech 2010 paralinguistic challenge. In *Interspeech*, volume 2010, pages 2795–2798, 2010.

[109] Mohammad Shami and Werner Verhelst. Automatic classification of expressiveness in speech: a multi-corpus study. In *Speaker classification II*, pages 43–56. Springer, 2007.

[110] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[111] Mandeep Singh, Mr Mooninder Singh, and Nikhil Singhal. Ann based emotion recognition. *Emotion*, (1):56–60, 2013.

[112] Peng Song, Wenming Zheng, Shifeng Ou, Yun Jin, Wenming Ma, and Yanwei Yu. Joint transfer subspace learning and feature selection for cross-corpus speech emotion recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[113] André Stuhlsatz, Christine Meyer, Florian Eyben, Thomas Zielke, Günter Meier, and Björn Schuller. Deep neural networks for acoustic emotion recognition: raising

the benchmarks. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5688–5691. IEEE, 2011.

[114] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199, ICLR*, 2014.

[115] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565, 2014.

[116] HM Teager and SM Teager. Evidence for nonlinear sound production mechanisms in the vocal tract. In *Speech production and speech modelling*, pages 241–261. Springer, 1990.

[117] Vikrant Singh Tomar and Richard C Rose. Graph based manifold regularized deep neural networks for automatic speech recognition. *arXiv preprint arXiv:1606.05925*, 2016.

[118] Nishtha Tripathi and Avani Jadeja. A survey of regularization methods for deep neural network. *International Journal of Computer Science and Mobile Computing, IJCSMC*, 3(11):429–436, 2014.

[119] Michel Valstar, Jonathan Gratch, Björn Schuller, Fabien Ringeval, Denis Lalanne, Mercedes Torres Torres, Stefan Scherer, Giota Stratou, Roddy Cowie, and Maja Pantic. Avec 2016: Depression, mood, and emotion recognition workshop and challenge. In *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge*, pages 3–10. ACM, 2016.

[120] Michel Valstar, Björn Schuller, Kirsty Smith, Timur Almaev, Florian Eyben, Jarek Krajewski, Roddy Cowie, and Maja Pantic. Avec 2014: 3d dimensional affect and depression recognition challenge. In *Proceedings of the 4th International Workshop on Audio/Visual Emotion Challenge*, pages 3–10. ACM, 2014.

[121] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pre-training for audio-based music classification. In *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[122] Dimitrios Ververidis and Constantine Kotropoulos. Emotional speech recognition: Resources, features, and methods. *Speech communication*, 48(9):1162–1181, 2006.

[123] Hui-Po Wang, Wei-Jan Ko, and Wen-Hsiao Peng. Learning priors for adversarial autoencoders. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1388–1396. IEEE, 2018.

[124] Carl E Williams and Kenneth N Stevens. Emotions and speech: Some acoustical correlates. *The Journal of the Acoustical Society of America*, 52(4B):1238–1250, 1972.

[125] Rui Xia and Yang Liu. *Using denoising autoencoder for emotion recognition*, pages 2886–2889. International Speech and Communication Association, 2013.

[126] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[127] Zixiaofan Yang and Julia Hirschberg. Predicting arousal and valence from waveforms and spectrograms using deep neural networks. *Proc. Interspeech 2018*, pages 3092–3096, 2018.

[128] Mingyu You, Chun Chen, Jiajun Bu, Jia Liu, and Jianhua Tao. Emotion recognition from noisy speech. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1653–1656. IEEE, 2006.

[129] Mingyu You, Chun Chen, Jiajun Bu, Jia Liu, and Jianhua Tao. Manifolds based emotion recognition in speech. *Computational Linguistics and Chinese Language Processing*, 12(1):49–64, 2007.

[130] Amir Zadeh, Paul Pu Liang, Soujanya Poria, Prateek Vij, Erik Cambria, and Louis-Philippe Morency. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[131] A Zhang. Speech recognition (version 3.8). `https://github.com/Uberi/speech_recognition#readme`, 2017.

[132] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

[133] Zixing Zhang, Felix Weninger, Martin Wöllmer, and Björn Schuller. Unsupervised learning in cross-corpus acoustic emotion recognition. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 523–528. IEEE, 2011.

[134] Shusen Zhou, Qingcai Chen, and Xiaolong Wang. Active deep networks for semi-supervised sentiment classification. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1515–1523. Association for Computational Linguistics, 2010.