THE UNIVERSITY OF AIZU

DOCTORAL THESIS

Multimodal Information Fusion based on Deep Learning

Author:
Jianguo YU

Supervisor:
Prof. Konstantin Markov

A dissertation submitted in partial fulfilment of the requirements for the degree of Doctor of computer science and engineering

in the

Human Interface Laboratory
Graduate Department of Computer and Information Systems

August 26, 2019

Copyright by Yu Jianguo

All Rights Reserved

The thesis titled

Multimodal Information Fusion based on Deep Learning

by

Jianguo Yu

is reviewed and approved by:

Chief Referee
Professor
Konstantin Markov

Konstantin Markov マルコフ

Professor
Masahide Sugiyama

M Sugiyama

Professor
Qiangfu Zhao

Qiangfu Zhao Aug. 19, 2019

Professor
Ian Wilson

Ian Wilson ウィルソン

The University of Aizu

2019

THE UNIVERSITY OF AIZU

Abstract

Field of Study Applied Information Technologies
Graduate Department of Computer and Information Systems

Doctor

Multimodal Information Fusion based on Deep Learning

by Jianguo YU

To survive in this complex world, we have to constantly obtain information about events happening around us. A modality is a particular form of a signal, from which we can extract information about an event and the information about the same event can have many modalities. This thesis is related to deep learning-based multimodal learning approaches.

Chapter 1 describes the motivations and applications of multimodal learning; What challenges we are facing as well as the basic approaches and related studies.

Chapter 2 presents the most commonly used representations for different modalities, i.e. acoustic information, articulatory information, video information, image information, and text information as well as their preprocessing and feature extractions approaches.

Chapter 3 presents the basic knowledge about deep neural networks, i.e. feedforward neural networks, recurrent neural networks (LSTM and GRU implementations), convolutional neural networks, autoencoders, correlational neural networks, and word embedding.

Chapter 4 first presents the basic knowledge of ASR systems, i.e. the GMM-HMM system, and the DNN-HMM hybrid speech recognition system, then gives a brief review of related studies of acoustic and articulatory information integration. After that, the details of our proposed methods are presented, i.e. acoustic RNN training using generalized distillation and joint inversion training followed by the experiments, results, and analysis.

Chapter 5 presents the basic knowledge of personality recognition; The Big Five personality traits, the details of our low-level and high-level feature extractions, multi-stage training strategy, the formulation of our objective functions and neural network settings. Then, the experiments, results, and analysis will be reported.

Chapter 6 summaries the overall principles of multimodal information fusion, the advantages, and the disadvantages. Then, the directions for future research will also be discussed.

Acknowledgements

After I finished my master program, I was planning to find a job, but my supervisor encouraged me to continue the doctoral program and I am very glad that I've taken his advice because, during the process of pursuing a doctoral degree, I finally found what I want to do. I would like to express my deepest sense of gratitude to my supervisor. Without his continuous advice and support, I couldn't have made it.

I would like to express my very sincere gratitude to Prof. Masahide Sugiyama who helped me a lot with daily life in Japan.

I am also very thankful to Daria Vazhenina who has been a good senior and friend.

Finally, I take this opportunity to express my profound gratitude to my parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In order to survive in this complex world, we have to constantly obtain information about events happening around us. A modality is a particular form of signal, from which we can extract information about an event and the information about the same event can have many modalities. Human evolved many sensory systems that interpret different forms of signals (we see objects, hear sounds, feel textures, smell odors, taste flavors) in order to deal with complicated situations, some of which are listed below.

The situation in which some modalities are not available
For example, we know visual information travels very fast, but there is not enough light for us to see things clearly at night, so we have to rely on our auditory system to avoid dangers and even generate vague images from the sound we hear.

The situation in which a single modality is not sufficient
For example, when neither visual information nor acoustic information is sufficient for making a decision, our brains have the ability to extract information complementary to each other from both modalities, which refers to *multimodal*.

The situation in which a modality can help to learn another modality
For example, sometimes we have knowledge in forms of visual information and we can learn knowledge in forms of acoustic information with the help of the learned visual knowledge.

In order to make progress in Artificial Intelligence, we need to enable machines to interpret and reason about multimodal information. The goal of *Multimodal Learning* is to build machines that can process and relate information from multiple modalities.

The rest of this chapter is organized as follows. Section 1.1 describes the motivations and applications of multimodal learning. Section 1.2 talks about the challenges we are facing. Section 1.3 briefly introduces the basic approaches and related studies. Section 1.4 summarizes the main contributions of this research. Finally, Section 1.5 will outline the structure of this thesis.

1.1 Applications of Multimodal Learning

Multimodal learning attracts many researchers nowadays because the advances in algorithms, data, and computational power enabled multimodal learning to have a wide range of applications. The overview of three typical applications is provided here.

Audio-Visual Speech Recognition
One of the typical multimodal applications is audio-visual speech recognition (AVSR) [1], which was motivated by the McGurk effect [2] that visual information can affect the results of speech perception. For example, when human subjects heard the syllable /ba-ba/ while watching the lips of a person saying /ga-ga/, what they perceived was /da-da/. This drives many researchers to introduce additional visual information into their automatic speech recognition systems to improve the robustness of the system. However, in many scenarios, to use the ASR systems conveniently, people need to recognize the speech only from the sound signal, which makes the visual information inaccessible during recognition and limits the systems to only use visual information for the training propose.

Medical Diagnosis
Multimodal learning plays an indispensable role in medical applications. Since medical detections usually involve measurements in the form of different modalities such as X-ray, functional MRI (fMRI), CT, ultrasound, positron emission tomography (PET), etc, each of which has its strengths and shortcomings. The integrated interpretation for diagnosis of these modalities requires highly trained human experts. Therefore, the ability of learning from multimodal information will be a key solution to this problem, which has prompted more and more multimodal learning-based medical analysis research in recent years such as tissue and organ segmentation [3], multimodal medical image retrieval [4], multimodal medical image registration, and computer-aided diagnosis [5] and more information can be found in [6] [7].

Human Behavior Modeling
Modeling, analysis, and synthesis of human behavior have become less difficult nowadays because of the vast amounts of data generated by users on the internet such as images, videos, comments, and views that hold rich information about the users themselves. Utilizing this multimodal information can help us to design better human-human and human-machine interactions. According to Vinciarelli and Mohammadi [8], any technology involving understanding, prediction, and synthesis of human behavior is likely to benefit from personality computing approaches. Personality analysis from multimodal social information provides a better understanding of our life choices, well-being, and many other social behaviors, which enables many important practical applications, such as products, jobs, or services recommendation [9] [10], word polarity disambiguation, crime forecasting [11], etc.

1.2 Challenges of Multimodal Learning

However, researchers are facing many challenges when building machines that benefit from multimodal information. Some of the typical challenges are listed below:

Representation
The first challenge is how to represent multimodal data in a way that exploits the complementarity and redundancy of multiple modalities and filters the conflicting ones. Conventionally, techniques like principal component analysis (PCA), independent components analysis, and canonical correlation analysis (CCA) can be applied to achieve these objectives. However, the well-studied representation of one task is not necessarily suitable for other tasks, which requires repeated efforts spent on feature selection. Therefore, the common representation learning framework is badly needed.

Compensatory Mechanism
The second challenge involves dealing with missing modalities. For example, the auditory system can be viewed as an alternative when the visual information is lacking, it will lose this purpose if machines have to require both information for decision making. Therefore, researchers need to come up with systems that are robust enough to compensate for missing modalities during inference. A generative model that maps signals from one modality to another is often involved to accomplish this objective, which, however, leads to another machine learning challenges due to the nonlinear and complex relationship between two spaces of modalities.

Lack of Parallel Data
The third challenge involves the training of the multimodal system. It will only be a computational power problem if the parallel data of all modalities are present. However, in practice, such parallel datasets are extremely rare. The data of multiple modalities often come from different datasets with different numbers of samples. How can we organize the non-parallel datasets to train a multimodal system for predicting of a specific task and how can we use the knowledge that is already learned from other tasks to build the desired multimodal system, which refers to *transfer learning*, is also included in the research area of multimodal learning.

Multimodal Fusion
The last challenge is how to combine information from multiple modalities to perform a better prediction. Because the multiple modalities usually come with different dimensionalities and sampling rates and adding or reducing one particular modality will completely change the distribution of input space, which consequentially affects the choice of the machine learning algorithms and architectures.

1.3 Methods and Related Studies

Techniques for multimodal learning have long been investigated by the research community [12], [13]. Traditionally, there are two approaches for combining the signals of multiple sensors.

1.3.1 Feature Fusion

One approach is called *early fusion* or *feature-level fusion*. Feature-level fusion involves how to integrate the multiple modalities into a single feature vector, before being used as input to a machine learning algorithm. For example, the simplest form of early fusion involves concatenation of different multimodal features, as illustrated in Fig.1.1.

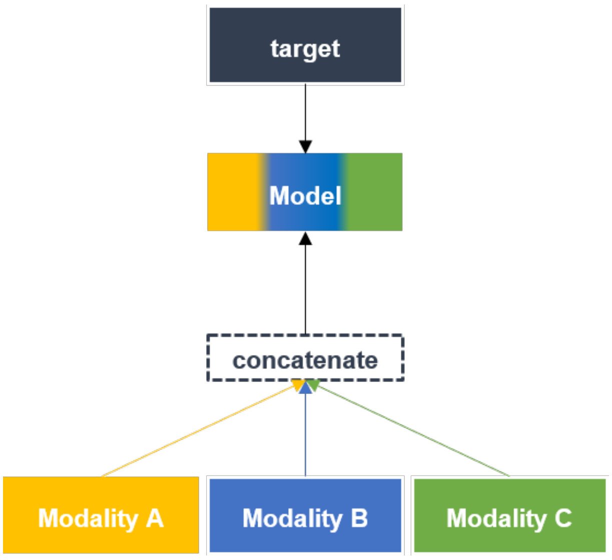


FIGURE 1.1: *An illustration of early fusion for multimodal learning.*

Most feature-level fusion models make the assumption that there is conditional independence between different modalities, which sometimes may not be true in practice, as multiple modalities tend to be highly correlated. But [14] also argues that different streams contain information that is correlated to another stream only at a high level, which supports the output of each modality to be processed independently of the others. However, feature level fusion still has some disadvantages:

Asynchronicity
Feature-fusion could be quite challenging if the data to be fused is raw due to the mismatched sampling rate and forms of representation. For example, a raw video signal is typically a 4D tensor with a sampling rate of 5-30 frames/second while a speech signal is typically a 3D tensor with a much higher sampling rate. Usually, people apply feature extraction and selection to get higher-level representations before the feature fusion.

Redundancy
It is easy to contain redundant information when applying feature level

fusion. Typically, dimensionality reduction techniques like PCA, autoencoders are applied to remove these redundancies in the input space.

1.3.2 Model Fusion

Model fusion also combines different modalities, but instead of concatenating the features of different modalities, it uses the additional modalities to change the structure or parameters or features of the main modality, as illustrated in Fig.1.2. Since it doesn't treat information from different modalities as features, it does not require the additional observations during inference.

For example, in [15], a hybrid Bayesian network/HMM acoustic model incorporates the articulatory data. A relatively new way to incorporate knowledge into neural networks is the so called *Distillation Training*, where an additional loss function with soft targets is also being minimized during training. In [16], Hinton et al. have shown that soft targets from complex models can transfer knowledge to small models that are easy to deploy.

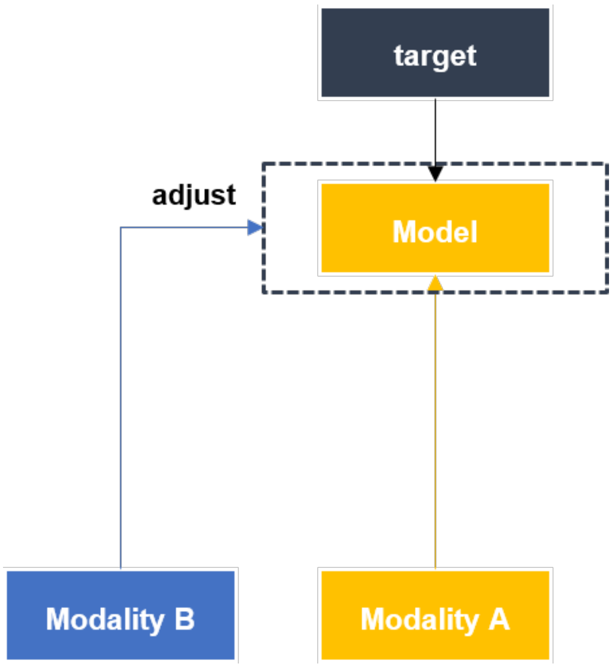


FIGURE 1.2: *An illustration of model fusion for multimodal learning.*

1.3.3 Decision Fusion

The other is called *late fusion* or *decision-level* fusion, which refers to the aggregation of decisions from multiple classifiers, each trained on separate modalities, as illustrated in Fig.1.3.

Decision-level fusion was popular within the machine-learning community in the early- to mid-2000s, people favor this fusion approach is because it doesn't have the shortcoming of asynchronicity like early fusion, therefore, is easier to implement and it doesn't depend on the representations of different modalities.

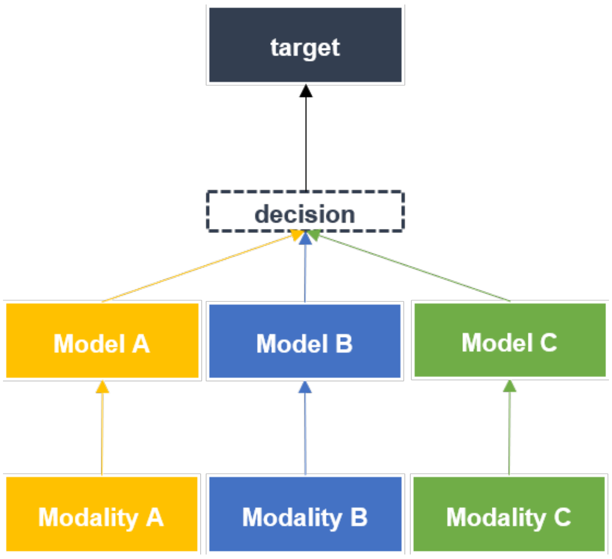


FIGURE 1.3: *An illustration of late fusion for multimodal learning.*

There are various ways to determine how the decisions from different modalities are combined to a single one, which could be max-fusion, averaged-fusion, Bayes' rule-based, etc.

Although decision-level fusion is easier to implement, it is not necessarily better than feature level fusion. Because the classifiers (or regressors) in decision-level fusion are relatively rigid, it is likely to miss information important for the final decision during the processing of each modality and the performance of feature level fusion heavily depends on the task.

1.3.4 Structure Fusion

The flexibility of deep neural networks (DNNs) offers an alternative approach for implementing multimodal learning task, which refers to *structure fusion*.

Because the neural network (structure fusion) approach can fuse features from different modalities at any layer, as illustrated in Fig.1.4, it could have both advantages of early fusion and late fusion if the balance is well adjusted.

Neural networks transform raw inputs to higher-level representations by mapping the input through consecutive layers, which has many advantages compared to early or late fusion:

Representation learning
Unlike early fusion where exports need to design handcrafted features or apply feature selection algorithms for each modality, neural networks can automatically learn a suitable representation for the final objective. Some specific operation can be inserted as a layer to filter out conflicting information.

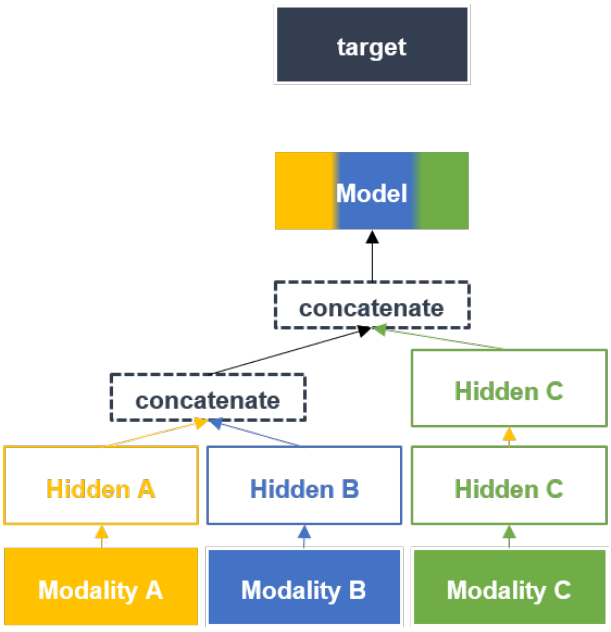


FIGURE 1.4: *An illustration of structure fusion for multimodal learning.*

Tensor rank adjustment
The basic form of data that flows in a neural network is tensor. The mismatch caused by different data form can be simply overcome by transforming the tensor from a modality into lower-rank tensor if necessary. For example, the 4D tensor of a raw video signal can finally be transformed into a single vector that can be concatenated with other feature vectors, which implicitly skips the problem of different sampling rates.

Hybrid structure
Usually, a discriminative model will be trained for prediction and sometimes a generative model will be adopted to compensate for missing modalities during inference. The flexibility of deep neural networks allows us to implicitly or explicitly combine the two types of models into a hybrid model for better performance.

Transfer learning
Deep neural networks also allow us to deal with the problem of non-parallel data because the learned parameters in the hidden layers can be shared between certain tasks. For example, one can simply take the first several layers trained on ImageNet dataset to his neural network then fix the parameters of these layers and finetune other parameters if he doesn't have enough data or he can retrain the whole network with a better initialization, as illustrated in Fig.1.5.

The work in [17] empirically showed that implementing a gradual fusion strategy, by first fusing highly correlated modalities, to less correlated ones in a progressive manner (e.g., visual modality first, then motion capture followed by audio), produced state-of-art results for communicative gesture recognition.

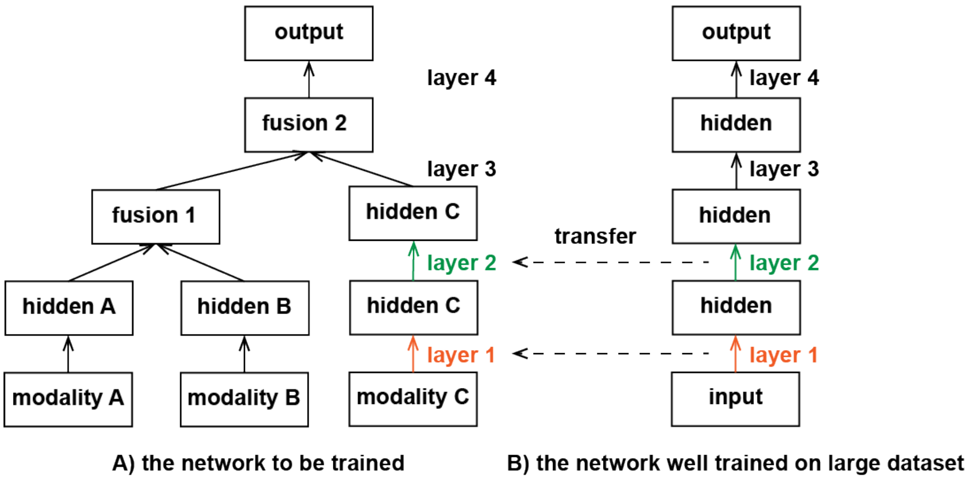


FIGURE 1.5: *An illustration of transfer learning. B) is the neural network well trained on a large dataset and A) is the neural network to be trained. The first two layers for modality C in A) are transfered from B), which can either be fixed or fine-tuned using new data.*

1.3.5 Regularization

The way to train a neural network is to minimize a loss function. In order to reduce overfitting and improve generalization, one or more regularization strategies are employed, often as an additional term added to the loss function. It could also be beneficial to separate the learning for each modality and apply regularization constraint based on other modalities. Therefore, the formulation of cost functions and regularizers is an important consideration for the structural design of neural networks. For example, [18] proposed a cost function that minimized the variation of information between modalities, whose intuition behind this formulation is that learning to maximize the amount of information that one data modality has about the others would allow models to generate information about the missing data modality.

The employing of regularizations can also help us to adjust the common goal that we need the model to achieve. For example, [19] incorporated a temporal term that grows exponentially in time into their cost function for a multimodal recurrent neural network (RNN) in order to encourage the model to fix mistakes as early as it can.

Although deep neural networks indeed offer much more flexibility of implementing multimodal learning and allow us to use a much easier way to handle challenges aforementioned. The architecture of a neural network needs to be carefully designed in terms of how, when the outputs of different modalities should be fused, which could be very time consuming and expensive (in terms of computational cost). The optimal structure and formulation of the cost function are more challenging to search and transfer for new tasks.

1.4 Contributions of this Research

Machine learning is typical task-dependent and there is no best structure and algorithm for all problems. Therefore, we investigated two different problems and found some common principles: 1. When automatically extracting features for different modalities, it is really helpful for the extractors to know the final objective, which can be done by introducing the final loss function to the extractors. 2. The typical way to merge different modalities using deep learning is to put them into a big single network. However, because this big network has many parts whose speeds of convergences are different, for example, the parts related to the audio modality may have already converged, but the parts related to the video modality are still training, it is hard for this single network to find optimal parameters for all modalities. One way to tackle this problem is to set the parameters of particular modalities untrainable and force the other parts to map the desired modalities to a good latent space first. In other words, constrain the freedom of deep neural network's automatic feature extraction.

Here we would like to outline the main contributions of the researches in this thesis: The new approaches to utilize multimodal learning for Automatic Speech Recognition (ASR) systems and Automatic Personality Recognition (APR) systems, which were developed, studied and evaluated during the last three years.

Multimodal automatic speech recognition

In order to make the systems more reliable and robust, researchers have been trying to utilize *articulatory information*. However, incorporating such information is challenging since it is impractical to obtain observations of articulators movements in real-life speech recognition scenarios. This constraint requires ASR systems to utilize articulatory data for training only, i.e. to be able to recognize without them.

The traditional approach to incorporate articulatory information is *feature based*, also called *articulatory-to-acoustic inversion*, where the missing articulatory features are generated from the acoustic signal during recognition. This, however, is not a simple task since the mapping between acoustic and articulatory data spaces is non-linear and not unique.

In order to skip this expensive and difficult articulatory inversion, we started with the idea of *transfer learning* and *regularization* in structure fusion method mentioned above and adopted *Generalized Distillation* framework [20] with the neural network structure of Long short term memory (LSTM) [21], in which soft targets are transferred from a teacher model trained using both articulatory and acoustic features to a student model using only acoustic features and developed a system that achieved 21.9% PER (phoneme error rate) reduction and introduces no extra cost during recognition, hence can be applied in the real situation.

But the flexibility of neural networks also allows us to implicitly embed a generative model into the network and form a hybrid network that is better than the two models trained separately. Therefore, we followed this idea, developed another novel hybrid network which is also able to recognize without articulatory information and achieved the best result of a 25.3% PER reduction at the expense of increasing the size of the neural network [22].

Multimodal automatic speech recognition

Because more and more social contents generated by users become available, automatic personality recognition has garnered much interest recently. However, conventional approaches depend heavily on the data representation which often is based on hard-coded prior knowledge. Because deep learning approaches can learn suitable representation automatically, we implemented several deep learning algorithms including fully-connected neural networks (FC), convolutional neural networks (CNN) and recurrent neural networks (RNN) in our personality recognition system and evaluated it on the task from the "Workshop on Computational Personality Recognition (Shared Task)" [23], in which the facebook status updates are used to predict the author's Big-Five personality traits (Openness to Experience, Conscientiousness, Extraversion, Agreeableness, and Neuroticism) [24]. Our experiments showed that CNN with average pooling is better than both the RNN and FC, which achieved the best results 60.0±6.5%.

After the primary investigation of APR task, we moved to a more challenge database which contains HD Youtube videos and developed a system for automatic apparent personality recognition (AAPR) that takes as inputs any combination of three modalities (text, audio, video) from a video clip and outputs 5 personality traits (the first impression of other people on this vlogger) along with an interview variable that recommends whether a job candidate should be invited for an interview. The system of the winner on this dataset processed different modalities separately, then combine them. But these features may not be complementary to each other. However, if we train them in a single network, the convergence speeds of these modalities are also different, it doesn't work well either. To address this problem, we propose an attention network with multiple training stages.

1.5 Thesis Outline

In this thesis, we are presenting an investigation of multimodal information fusion based on deep learning. ASR and APR systems are proposed, discussed and evaluated.

Chapter 2 presents the most commonly used representations for different modalities, i.e. acoustic information, articulatory information, video information, image information, and text information as well as their preprocessing and feature extractions.

Chapter 3 presents the basic knowledge about deep neural networks, i.e. feedforward neural networks, recurrent neural networks (LSTM and GRU implementations), convolutional neural networks, autoencoders, correlational neural networks, and word embedding.

Chapter 4 first presents the basic knowledge of ASR systems, i.e. the GMM-HMM system, and the DNN-HMM hybrid speech recognition system and gives a brief review of related studies of acoustic and articulatory information integration. The details of our proposed methods are presented, i.e. *acoustic RNN training using generalized distillation* and *joint inversion training* followed by the experiments, results, and analysis.

Chapter 5 presents the basic knowledge of personality recognition, the

Big Five personality traits, the details of our low-level and high-level feature extractions, multistage training strategy with attention, the formulation of our objective functions and neural network settings. Then, the experiments, results, and analysis will be reported.

Chapter 6 summaries the overall principles of multimodal information fusion, the advantages,and the disadvantages. Then, the directions for future research will also be discussed.

Chapter 2

Representation of Modalities

In order for machines to make sense of multiple modalities, we need to quantize and process the data of each modality into a suitable representation. Hence, this chapter presents the basic common representation of acoustic, articulatory, image, video, and text information.

The rest of this chapter is organized as follows. Section 2.1 describes the spectrum presentation of acoustic modality; The Mel Frequency Cepstral Coefficients (MFCCs) and the standard pipeline of feature extraction, Section 2.2 talks about what is articulatory information and how to represent it.Section 2.3 briefly introduces the basic knowledge about image and video information in computer and the spatial and temporal representations of them. Finally, Section 2.4 is about the text information and distributed representation.

2.1 Acoustic Modality

The acoustic signal is easy to produce and receive without any extra devices, therefore spoken languages have been the most used way of communication in our daily life. From the Acoustic signal, we can get a lot of information about the messages the speakers want to send and even the speakers themselves. In order for a system to obtain information from it, we need to extract proper features recognizable to the system as the representation of the acoustic signal.

2.1.1 Signal Preprocessing

In order to simplify the mathematical analysis, the acoustic signals are usually converted from the time domain to the frequency domain. Before that, several steps are needed to pre-process the signal, as illustrated in Fig.2.1.

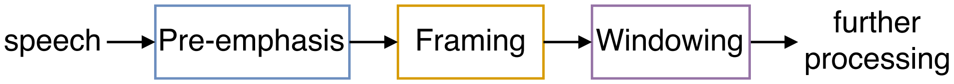


FIGURE 2.1: *Pre-processing steps*

1. **Pre-emphasis**: is a high-pass filter, that boosts the high-frequency energy.

2. **Framing**: divides speech signal into small overlapping chunks. Length is typically $20 - 25ms$ and Shift is typically $10ms$ or about 50% of the frame.

3. **Windowing**: Each frame is multiplied by a window function to minimize spectral discontinuities at the frame start/end. The most common used window is *Hamming window*.

2.1.2 Mel Frequency Cepstral Coefficient

One of the most popular and reliable features for speech recognition is *Mel Frequency Cepstral Coefficents* [25], whose process of extraction is shown in Fig.2.2.

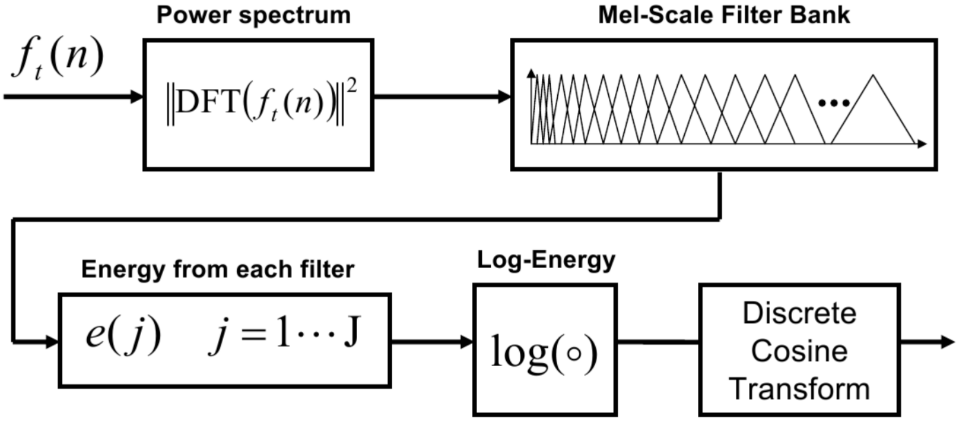


FIGURE 2.2: *MFCC block diagram*

1. **Power Spectrum**: Apply *Discrete Fourier Transform* to windowed signal $f_t(n)$ to transform the signal from time-domain to frequency-domain and take of power of it, because in frequency-domain, the signal can be analysed more consistently and easily.

2. **Mel-Scale Filter Bank**: This approximates the human auditory system's response

3. **Discrete Cosine Transform (DCT)**: After log-energy, DCT is applied to map the signal back from the frequency domain into the time domain. Instead of Inverse Discrete Fourier Transform (IDFT) use DCT because log power spectrum is real and symmetric. Usually people take the first 12 Coefficients.

4. **Frame Energy**: If we append energy computed from windowed frame, the dimensions become 13.

5. **Velocity and Acceleration**: To introduce the information of temporal changes, velocity and acceleration are often appended, resulting a 39 dimensional vector.

2.2 Articulatory Information

Due to the inherent characteristics of acoustic signals under the noise environment, Automatic Speech Recognition systems suffer from acoustic variabilities posed by background noises, reverberations, and recording device, etc. This motivated researchers to use additional signals as complementary of acoustic signals for modeling.

A simplified form of visual information of speech is articulatory information, which is the movements of lips, jaw, and other speech organs when we are speaking, as shown in Fig.2.3.

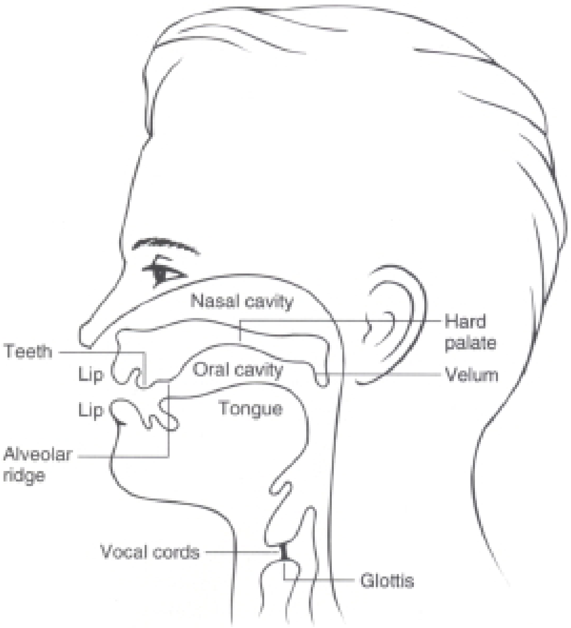


FIGURE 2.3: *Articulatory information*

Articulatory information is especially useful for modeling the coarticulation, which is a speech production effect which results in an assumption where some sounds have influences on others (e.g. "p" sound in the word " speed" becomes "b" sound due to "s" sound). The traditional way to model this phenomenon is to introduce bi-phone or tri-phone (several phones tied together as one unit) into acoustic models, but this approach assumes that coarticulation effects only impact neighboring phones which is not always true, as, in practice, there are many instances where coarticulation effects extend beyond the immediate neighbors [26] [27].

While articulatory information can model coarticulation effectively since it doesn't rely on the "non-overlapping phonetic units" assumption. Here are some advantages of using articulatory information:

1. Articulatory factors are much less affected by environmental conditions.

2. It can make systems more robust.

3. It can preserve information that tends to be lost during the extraction of speech acoustic features.

4. It can also help model coarticulation in a more systematic way rather than using tri- or quin-phone acoustic models.

Articulatory Information is essentially images through time. The representation of it can be images, outlines, points. For example, the articulatory representation in the MOCHA-TIMIT database captured using Electromagnetic articulography (EMA) is the trajectories of points and the one in the USC-TIMIT database obtained using Real-time magnetic resonance imaging (rtMRI) is the video of the vocal tract during speaking.

Trajectories of Articulators
Although images naturally contain the maximum information. We can also reduce the size of data by just looking at the critical points since most of the information is carried by the critical articulators (tongue, lips, and jaw and so on) as shown in Fig.4.11. Unfortunately, such absolute measure can be inconsistent and may suffer from variability.

Tract Variables and Articulatory Gestures
Tract variables and articulatory Gesture [28] representation are relative measures and hence should be invariant. Articulatory phonology treats each word as a constellation of vocal tract constriction actions, called gestures (roughly 1 to 3 gestures for each of the phones in a phonetic transcription). Although it can model the coarticulation [29] simply and elegantly, there is no existing database of it. Therefore, it's usually synthetic data or data estimated from other databases, such as the procedure described in [30].

2.3 Visual Information

The most common signals to which we access are visual information. People post a large number of images and videos on the internet every day. It is very useful for machines to be able to interpret information from the visual signals.

2.3.1 Image representation

Images in computer
A gray image (with only one color representing the lightness) in computers is typically a matrix of pixels. If it is an 8-bit image, the values in the matrix are integers in the range [0, 255], where 0 is darkest and 255 is lightest. An RGB image (with three colors representing red, green, blue intensities) is a 3D tensor with an extra dimension representing the number of channels (also called *depth*), as shown in Fig.2.4.

Spatial features of images
The presentation of an image could be a flatten vector, for example, an MNIST image with a size of 28 x 28 can be represented as a 784-dimensional vector. However, this will lose the spatial information of an image, since the computer does not know the neighboring relationships of pixels. Therefore, the common way is to represent an image as 3D tensor with a shape of (width, height, depth).

2.3.2 Video representation

Videos in computer
The video is essentially a sequence of images varying in time and is represented as a 4D tensor with a shape of (timestep, width, height, channel), as

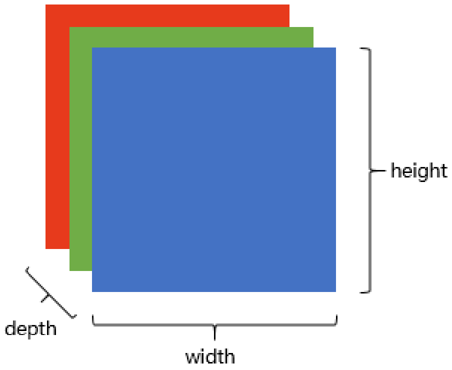


FIGURE 2.4: *the representation of 8-bit RGB images in computer*

shown in Fig.2.5.. The commonly used frame rate of videos on the internet is 25-30 frames per second (f/s), which means the uncompressed video information could be very big for processing, hence will often be downsampled to 10-15 f/s to reduce the size. In addition to spatial information, we also need to consider how to capture temporal information of a video.

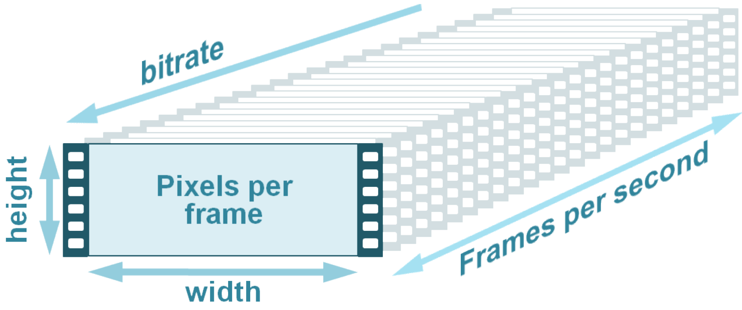


FIGURE 2.5: *the representation of a video in computer*

Spatial and temporal features of videos

There are three common ways to extract spatial and temporal features using neural networks. When using 3D convolutional neural networks, the spatial features and temporal features of a video are captured together. When using a convolutional-recurrent neural network, the convolutional layers are believed to be able to extract spatial features, which will be feed into the recurrent layers for temporal feature extraction.

In 2014, the authors [31] proposed to use two separate networks - one for spatial features (pre-trained), one for temporal features, therefore is called Two Stream Network, in which images in the temporal network are replaced by in stacked optical flow vectors, as shown in Fig.2.6. Though Two Stream Network improved the performance of the single-stream method by explicitly capturing local temporal movement, there were still a few drawbacks and many papers have been published intending to improve these basic 3 ways mentioned above, which can be summarized in Fig.2.7.

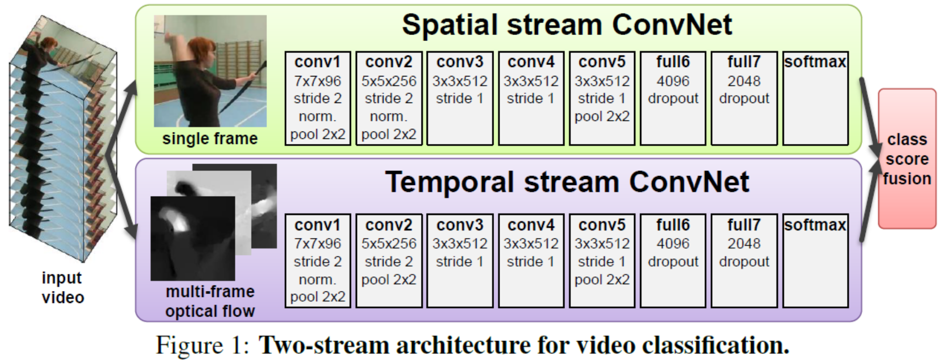


FIGURE 2.6: *Two-stream architecture for video classification [31].*

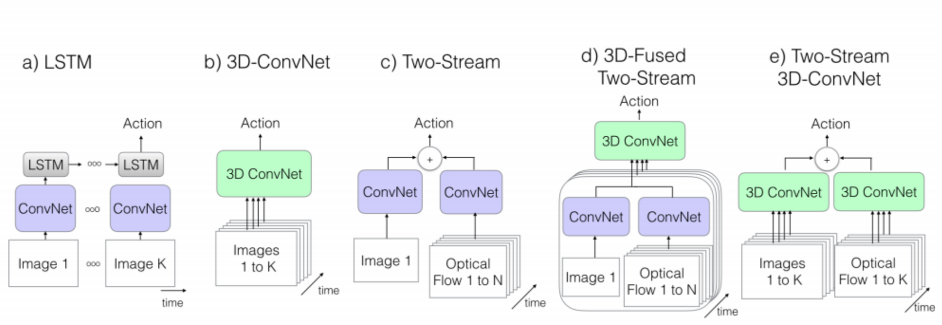


FIGURE 2.7: *Video architectures summarization, where K stands for the total number of frames in a video, whereas N stands for a subset of neighboring frames of the video [32].*

2.4 Text Information

Dislike other types of information, text information is typically represented as a sequence of word vectors, each of which represents a single word. There are two types of word representations in deep learning.

One hot representation

If there are 10000 different words in the vocabulary, one hot encoding assigns each of these words a unique index and a word will a 10000-dimensional vector with only the assigned index being 1, the others are 0's, as shown in Fig.2.8. However, this means all words are independent, which is not sure in real situations. For example, we know "girl" and "woman" should have a relationship that is similar to the one between "boy" and "man". If we can take into account of this property, we can reduce the number of examples needed to train all the words, which leads us to the distributed representation of words.

Distributed representation

The one hot representation of words is high dimensional, sparse, and orthogonal. However, this can be fixed by learning a mapping f which

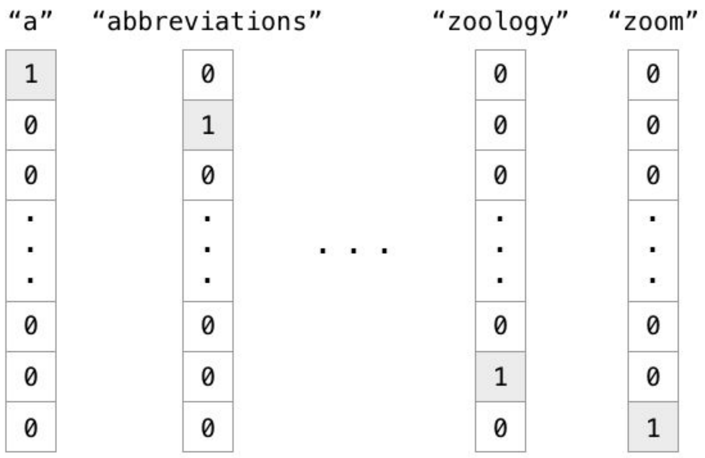


FIGURE 2.8: *one hot encoding for words.*

transforms the one hot vector space into a low-dimensional space, as shown in Fig.2.9, in a way that similar words have similar values, for examples, the distance between "girl" and "woman" are close to the distance between "boy" and "man", as shown in Fig.2.10.

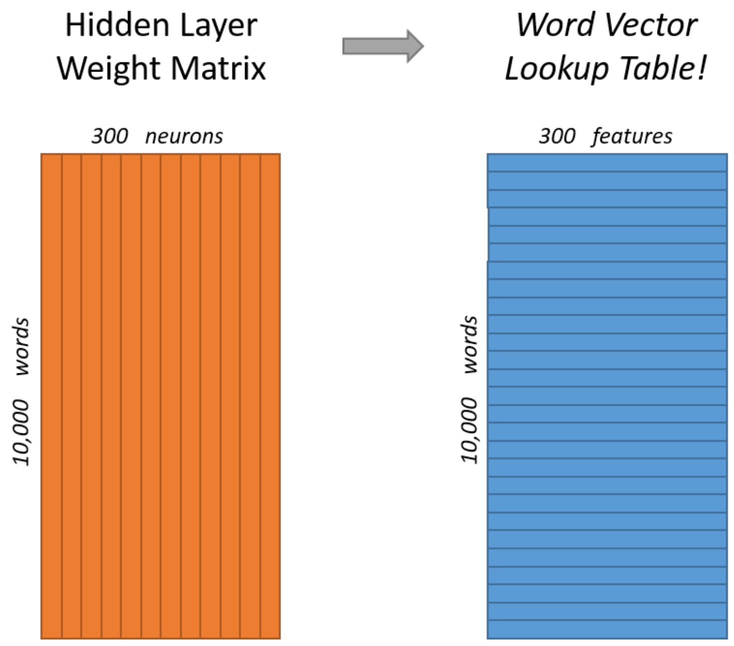


FIGURE 2.9: *The goal of word embedding just to learn this hidden layer which is just a lookup table.*

The distributed representation is often achieved by learning a embedding matrix using a method called word embedding, which will be introduced in next chapter. After converting one hot representation of a sentence into its distributed representation, the dimension of every word vector is reduced, as shown in Fig.2.11.

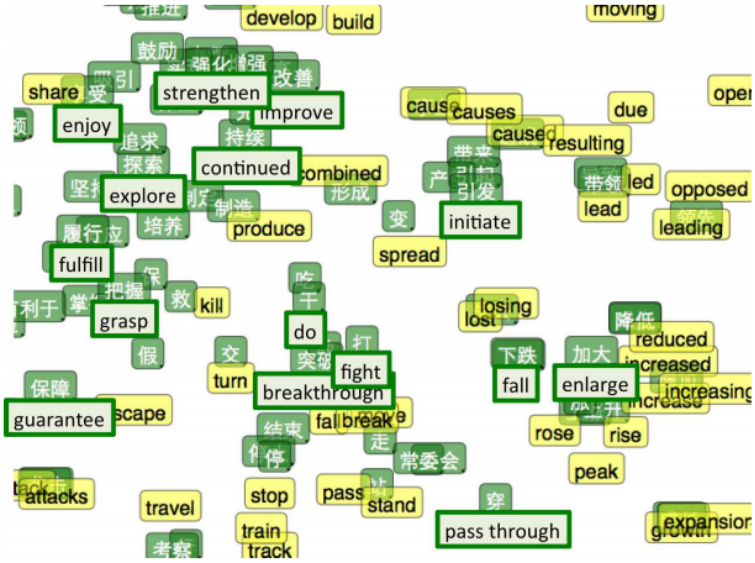


FIGURE 2.10: *t-SNE visualization of the bilingual word embedding. Green is Chinese, Yellow is English.*[33]

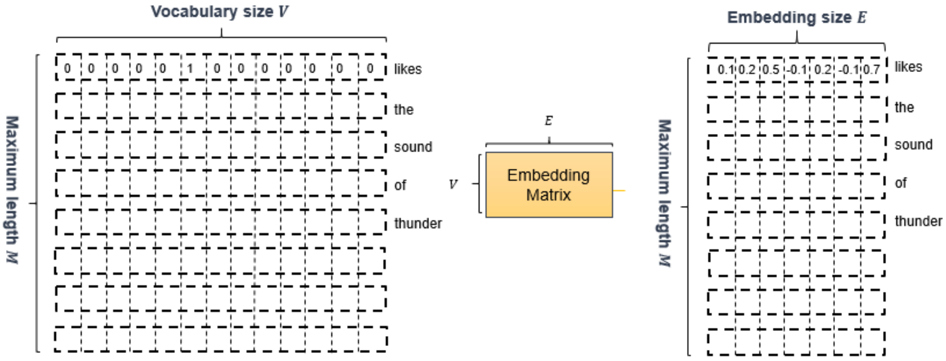


FIGURE 2.11: *The left side is a matrix of a sentence and the right side is the matrix of the sentence after word embedding.*

Chapter 3

Deep Neural Networks

Deep Neural Networks (DNNs) have recently achieved breakthrough results in almost every machine-learning task. They are a set of machine-learning algorithms inspired by how the brain works. Unlike most traditional machine-learning algorithms, deep Neural networks perform automatic feature extraction without human intervention [34]. However, it is still not fully understood why deep learning works so well.

The rest of this chapter is organized as follows. Section 3.1 describes Feed-Forward DNNs and the basic knowledge about how a DNN is formed and trained. Section 3.2 talks about recurrent DNN; The motivation of using this type of layer and its implementations (LSTMs and GRUs) and variants. Section 3.3 Convolutional DNN; What is the different between a FNN and a CNN and how to control the operation the a convolutional layer. Section 3.4 is about autoencoder and what it is used for. Section 3.5 presents how distributed representations of words can be learned by word embedding.

3.1 Feed-Forward DNN

Two perspectives to understand the behavior of DNNs are shared here.

Perspective of linearly separability
A toy DNN shown in Fig.3.1 It is challenging to understand the behavior of deep neural networks in general, but it's much easier to see what's happening when the data is passed through a single layer: A mapping from input space to output space. It is basally **linear transformation followed by an activation function**, whose mathematical description is Eq.(3.1), where $\vec{x}$ is the input, W is weights matrix, $\vec{b}$ is bias, $a()$ is activation function and $\vec{y}$ is the output of this layer.

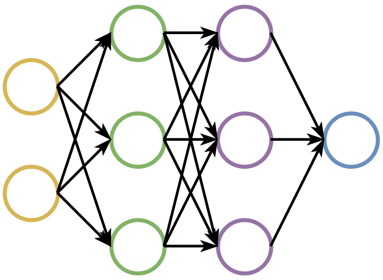


FIGURE 3.1: *simple DNN*

$$\vec{y} = a(W \cdot \vec{x} + \vec{b}) \tag{3.1}$$

If we break down the equation, the single layer actually transforms the data and create a new representation by the following 5 space operations.

1. Change the dimensionality of the input space.
2. Rotate the input space.
3. Scale the input space.
4. Translate the input space.
5. "bend" the input space.

The first 3 operations are done by $W \cdot \vec{x}$, the 4th one is provided by $\vec{b}$, and the 5th operation is done by $a()$ which gives nonlinearity to the layer.

Take the example of the simple classification case shown as Fig.3.2. It's impossible to separate the two curves with a line.

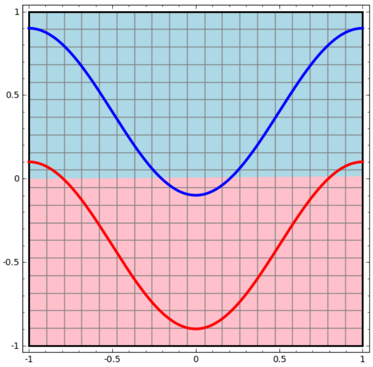


FIGURE 3.2: *simple case [35]*

If we increase the number of layers, the 5 operations can be applied again and again so that we can finally find a hyperplane that separates the two curves in the final representation. Fig.3.2 shows the final representation after these space operations of several layers.

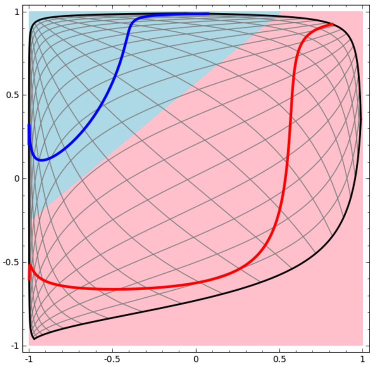


FIGURE 3.3: *After operations of several layers [35]*

From this perspective, the parameters of DNN indicate operations applied to the input space and the learning process of DNN is to make the input space linearly separable with the interactions of 5 space-operations of several layers *More about what DNNs do:* [35].

Physical perspective
Another perspective is about how a matter forms. We know that matters are composed of molecules which are again composed of atoms. Each layer of DNN can represent a level of matter (such as atom level or molecule level). If there are a certain amount of different atoms, they can form various molecules and eventually form a car. We can understand DNN with the similar concept, then the parameters of DNN indicate how, let's say, a car is composed of atoms. Fig.3.4 demonstrates that with the different combinations of pixels, we can get various edges, with different combinations of edges we get noses or ears and eventually we get all kinds of faces. Once we know how a certain picture forms from pixels, we can tell whether a combination of new pixels is that picture.

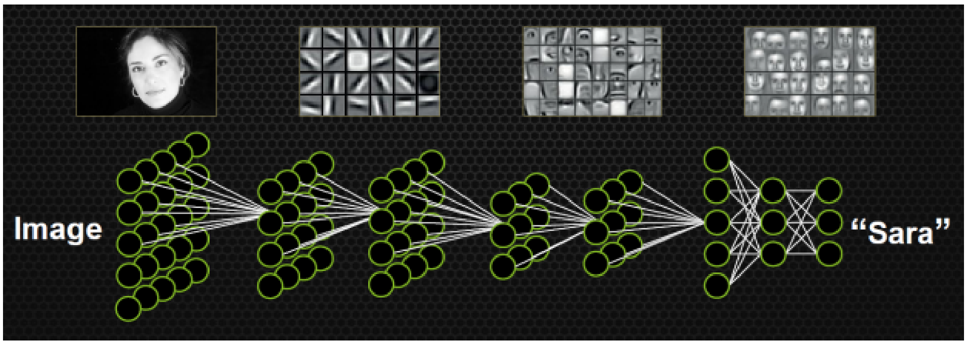


FIGURE 3.4: *How a picture of face is composed of pixels*

But why this simple idea of "DEEP" neural networks works so well in reality? Many researchers argued that the success of deep learning could depend not only on mathematics but also on physics [36].

The well-known universal approximation theorem [37] guarantees that a neural network with only one hidden layer can approximate arbitrary functions, provided enough number of hidden nodes, However, this does not touch upon the algorithmic learnability of those parameters and how many different examples are needed to achieve this approximation. If the number of examples needed for learning a function using the shallow neural network is the same as the one needed when using a lookup table, it will lose the meaning of learning. However, this could be improved by "DEEP" neural networks by assuming that instants (examples) share the same hidden representations in a series of hidden layers and these hidden representations can be reused by all examples including unseen ones so that the models can learn better with some amount of training data. Therefore, the success of Neural Networks could hinge on the ability to disentangle the factors of variations and learn those factors independently, which allows models to represent an exponentially large number of variations by learning a linearly large number of factors.[38].

The learning process of DNN is to transform the input space to a linearly separable space with the interactions of 5 space-operations of several layers. But how do we train it? First, some data pre-processings are needed to convert the signals into the ones recognizable to computer.

3.1.1 Data Pre-Processing

Like other machine learning methods, pre-processing is needed before training a deep neural network.

Mean Subtraction
Mean subtraction is the most common way which can make it easy for the network to converge. It's just subtracting the mean across every individual feature in the data and it can center the cloud of data around the origin along every dimension.

Normalization
Normalization refers to normalizing the data dimensions so that they are of approximately the same scale. There are two common ways of achieving this:

1. Divide each dimension by its standard deviation, once it has been zero-centered.

2. Normalize each dimension so that the min and max along the dimension is -1 and 1

Notes that the outputs should also correspond the activation function of the output layer. For example, if we use **sigmoid** activation function, then the range of outputs should be $(0, 1)$.

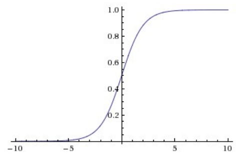
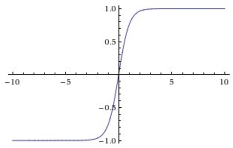
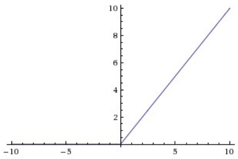
One-Hot Vector
If the task is classification, then instead outputting a most likely class, we also want to know the probabilities being other classes. Then, we need to convert our labels to one-hot vectors of size **number of classes**. For example, the class with index 12 would be the vector of all 0's and a 1 at position 12.

3.1.2 Non-Linearities

The Non-linearity of the network is given by activation function. Commonly used activation functions are **Sigmoid**, **Tanh**, **ReLU**. Their details are shown in Table.1 3.1 which one to use depends on the task. For example, the gates in a Recurrent Neural Network are a Neural Network with output layer of Sigmoid, because this network controls how much information can flow. *Also see how to choose an activation function:* [39]

Generally, ReLU is much faster than the other two. It is argued that this is due to its linear, non-saturating form. Compared to tanh/sigmoid that involves expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero. Unfortunately, ReLU has the "dying ReLU" problem. For example, a large gradient flowing through a ReLU neuron could cause the weights to update in such a way that the neuron will never activate on any datapoint again. If this happens, then the gradient flowing through the unit will forever be zero from that point on. By setting learning rate smaller and proper Initializing of weights can somewhat help to avoid it.

TABLE 3.1: *Details of activations*

Name	Plot	Equation	Range
sigmoid		$1/(1+e^{-x})$	$(0,1)$
tanh		$2/(1+e^{-2x})-1$	$(-1,1)$
ReLU		$max(0,x)$	$[0,\infty)$

Another very important activation function is the **softmax** function, which is usually used as the activation function of the output layer for classification tasks. It is a generalization of the logistic(sigmoid) function that gives the probabilities being classes and makes the sum of them equal to 1. The equation of softmax is given by Eq.(3.2) where K is the total number of neurons in the layer. This activation function gets applied row-wise (the sum of the row is 1 and each single value is in $[0,1]$).

$$\phi(x)_j = \frac{e^{x_j}}{\sum\limits_{k=1}^{K} e^{x_k}} \tag{3.2}$$

3.1.3 Loss Functions

We train a network by minimizing the error the current network makes. Therefore, first, we need to define the error. The function that measures the error is called the **loss function**.

Classification
For the multi-class classification tasks, crossentropy and softmax output activation function come with the pair. It's given by Eq.(3.3)

$$L_i = -\sum_j t_{i,j} \log(p_{i,j}) \tag{3.3}$$

Regression
For the regression tasks, we hope the predictions and targets are as close as possible, therefore, the loss function can be any type of distance between them, like **squared error(SE)**, which is given by given by Eq.(3.4)

$$L = |t-p|^2 \tag{3.4}$$

3.1.4 Updates

Once we have the loss function, we can update the parameters of DNN by minimizing the error got from loss function.

Backpropagation

Backpropagation is the game changer that makes training deep models computationally trainable. The earliest deep-learning-like algorithms that had multiple layers of non-linear features can be traced back to Ivakhnenko and Lapa in 1965 and the earliest convolutional networks were used by Fukushima in 1979. However, backpropagation was lacking at this point. Although it was derived already in the early 1960s but in an inefficient and incomplete form. Backpropagation can make training with gradient descent as much as ten million times faster. Backpropagation is essentially the chain rule that starts backward from the error. It's just a technique for calculating derivatives quickly. The real difference it made is that it enables us to train a model within a week than many years.

Gradient Descent

The method used in conjunction with Backpropagation for finding the minimum of loss function is Gradient descent. Generates update expressions of Eq.(3.5). It takes steps proportional to the negative of the gradient as , because we want to minimize the loss function.

$$param = param - learningrate * gradient \tag{3.5}$$

Depending on the **Size of examples** in each iteration, the name will also change:

1. **Stochastic Gradient Descent (SGD)**: one example from training set in each iteration.

2. **Mini-batch gradient descent**: m examples from training set in each iteration and the gradient will be averaged over m examples.

Unlike vanilla Gradient descent that runs through **all** samples in the training set to do a single update for a parameter in a particular iteration, SGD often converges much faster. Note that sometimes people use the term SGD even when referring to mini-batch gradient descent. The size of the mini-batch is a hyperparameter but it is not very common to cross-validate it. It is usually based on memory constraints. We use powers of 2 in practice because many vectorized operation implementations work faster when their inputs are sized in powers of 2. The smaller size tends to give more generalization because it will not fit the training set too well in each iteration.

Gradient Descent also has a problem. Because it is a first-order optimization algorithm that finds a local minimum of a function, it can get stuck in local minima and fail to reach the global minima as shown in Fig.3.5.

Learning Rate

The learning rate determines to what extent the parameters will update (also called step size). A learning rate of 0 will not learn anything. The learning rate is also a hyperparameter. Fig.3.6 shows how the learning rate

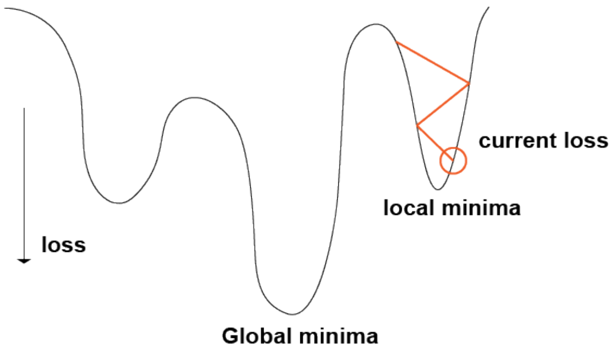


FIGURE 3.5: *get stuck in local minima*

affects the decay of loss function. Higher learning rates will decay the loss faster, but they get stuck at worse values of loss (green line). This is because there is too much "energy" in the optimization and the parameters are bouncing around chaotically, unable to settle in a nice spot in the optimization landscape.

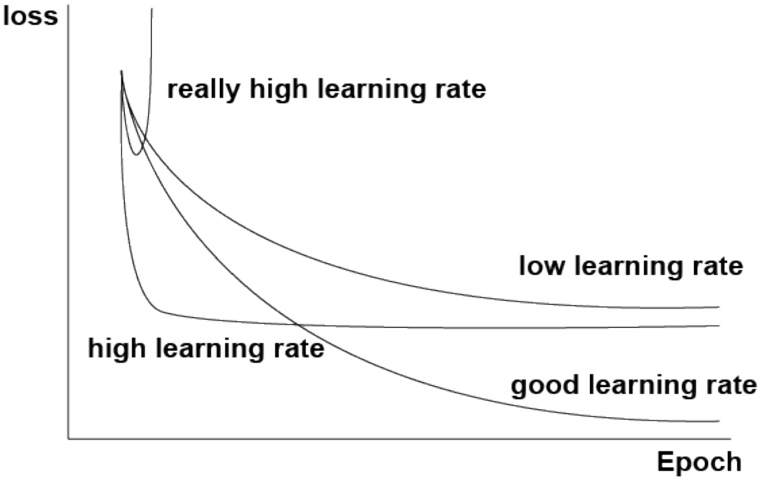


FIGURE 3.6: *How the learning rate affects the decay of loss function*

Therefore, it's useful to decay the learning rate during training. However, this can be tricky because decay it slowly and you'll be wasting computation bouncing around chaotically with little improvement for a long time. But decay it too aggressively and the system will cool too quickly, unable to reach the best position it can.

Momentum

Another technique that can help the network out of local minima is the use of a momentum term. Momentum simply adds a fraction m of the previous weight update to the current one. When the gradient keeps pointing in the same direction, this will increase the size of the steps taken towards the minimum. It is therefore often necessary to reduce the global learning rate when using a lot of momentum (*momentum* close to 1). If you combine a high learning rate with a lot of $momentum$, you will rush past the minimum with huge steps. This is different from the GD update shown above, where the gradient directly integrates the position. Instead, the physics view suggests an update in which the gradient only directly influences the velocity,Eq.(3.6, 3.7) which in turn has an effect on the position. The parameter momentum is usually set to values such as $[0.5, 0.9, 0.95, 0.99]$ and velocity is initialized as 0.

$$velocity = momentum * velocity - learningrate * gradient \tag{3.6}$$

$$param = param + velocity \tag{3.7}$$

Nesterov Momentum

Nesterov Momentum is a slightly different version of the momentum update has recently been gaining popularity. Fig.3.6 shows the difference from the momentum update. It generates update in a way described as Eq.(3.8, 3.9). Also see [40] for further reading.

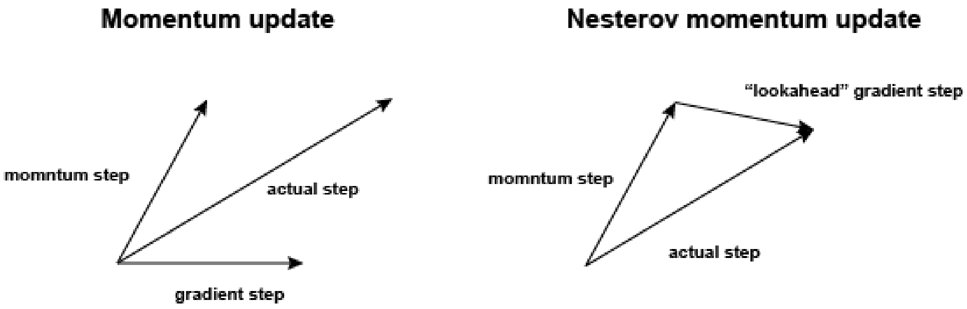


FIGURE 3.7: *With Nesterov momentum we evaluate the gradient at this "looked-ahead" position.*

$$velocity = momentum * velocity - learningrate * gradient \tag{3.8}$$

$$param = param + momentum * velocity - learningrate * gradient \tag{3.9}$$

Adaptive Updates

A bunch of update methods that adapt the learning rate and use momentum exist now. New update methods are also coming to us. Some of them are *Adagrad, Adadelta, RMSprop, Adam [41], Hessian-free [42]*. Which optimizer to use? In practice **Adam** is currently recommended as the default algorithm to use, and often works slightly better than **RMSProp**. However, it is often also worth trying **SGD+Nesterov Momentum** as an alternative. A good blog that is worth reading: [43]

Initialization

Before we can begin to train the network we have to initialize its parameters. One thing we should not do is to set all the initial weights to zero. Because if every neuron in the network computes the same output, then they will also all compute the same gradients during backpropagation and undergo the exact same parameter updates. In other words, there is no source of asymmetry between neurons if their weights are initialized to be the same. The right way to do it instead is to initialize weights randomly with small values. There are many ways to do this as well. The recommended heuristic is to initialize each neuron's weight vector as Eq.(3.10, where n is the dimension of the input to current layer.

$$w = \frac{N(0, \mu)}{\sqrt{n}} \tag{3.10}$$

Overfitting
Deep neural networks with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks, which means the system not only learns the pattern of training set, but also the noise of it and such case, the system can only give good predictions for the training set, but worse for test set. Fig 3.8 demonstrates underfitting and overfitting, where degree 1 is underfitting and degree 15 is overfitting.

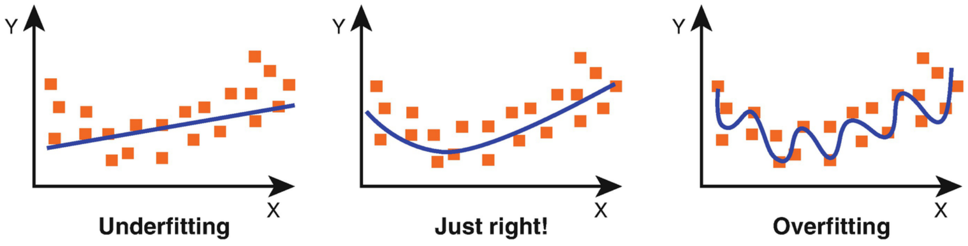


FIGURE 3.8: *Underfitting and Overfitting [44]. The dots are samples and the lines are predictions.*

Train/Validation Losses
A common way to see whether the model actually overfits is to split an additional set called validation set and calculate the accuracy for validation set as well. The gap between the training and validation accuracy indicates the amount of overfitting as shown in Fig 3.9

Add Noise To Training Set
Since overfitting means the model fits the training set too well and give worse results for test data, one way to reduce this bad effect is to add noise to the training set.

L2 regularization
Another common method is L2 regularization. If a deep neural network has a large number of parameters it's hard to guarantee that all parameters(nodes) have contribution to predictions. L2 regularization has the appealing property of encouraging the network to use all of its inputs a little rather than using only some of its inputs a lot. We do it by adding a penalty $1/2\lambda w^2$ to the loss function, where λ is the strength and adding $1/2$ is just for easy derivative calculation. Using the L2 regularization ultimately means that every weight is decayed linearly to zero. Because of

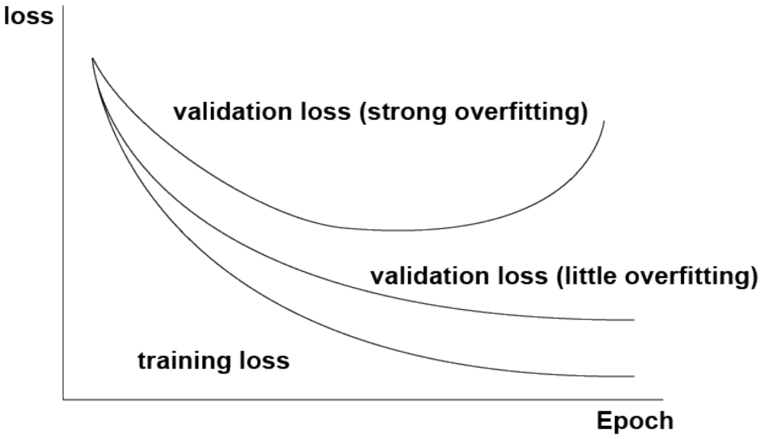


FIGURE 3.9: *The amount of overfitting can be noticed by analysis train/loss accuracies*

this phenomenon, L2 regularization is also commonly referred to as *weight decay*.

Dropout

Dropout is an extremely effective, simple and recently introduced regularization technique by Srivastava et al. in[45]. The key idea is to randomly drop units (along with their connections) from the neural network during training as shown in Fig 3.10. This prevents units from co-adapting too much. While training, dropout is implemented by only keeping a neuron active with some probability p (a hyperparameter), or setting it to zero otherwise.

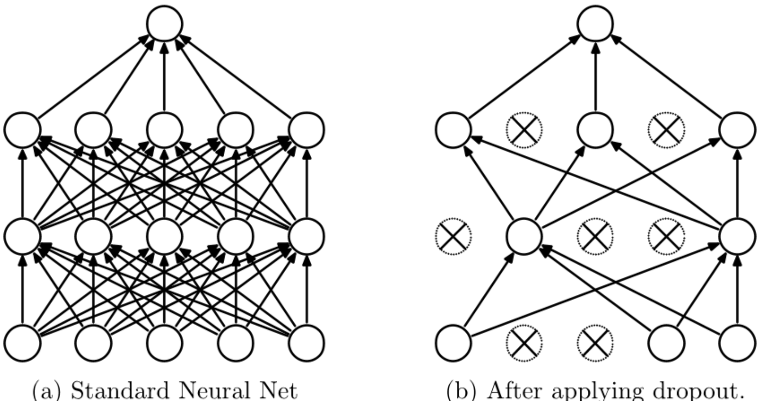


FIGURE 3.10: *Note: Only apply dropout during training*

Early Stopping

Another straightforward way is Early Stopping, which just simply stops training once some bad conditions happens. Fig 3.11. For example, stop the training if the validation loss function increases for 3 epochs.

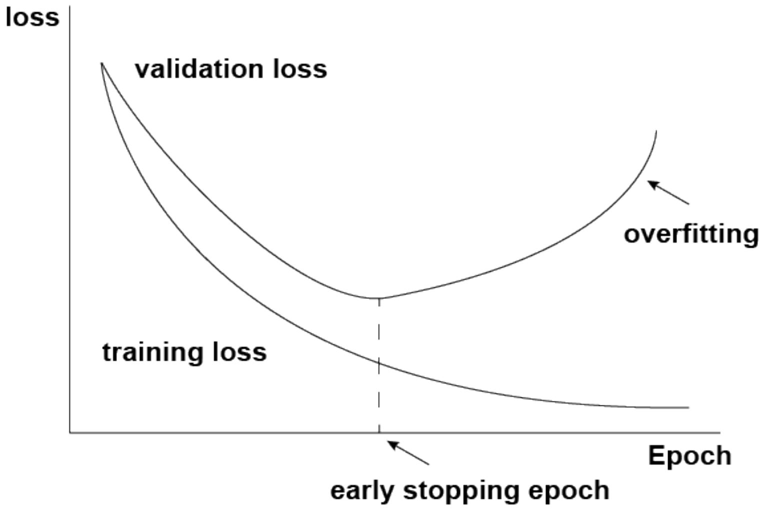


FIGURE 3.11: *Early Stopping*

3.2 Recurrent DNN

However, FNNs are general architecture for all tasks and do not have any preferred structure. Without enough training data, deeper FNNs easily tend to take into account those factors that only exist in the training set, resulting in overfitting. There are two basic types of neural networks addressing this problem by introducing parameter-sharing mechanism: recurrent networks (RNNs) and convolutional neural networks (CNNs).

3.2.1 Temporal Sharing

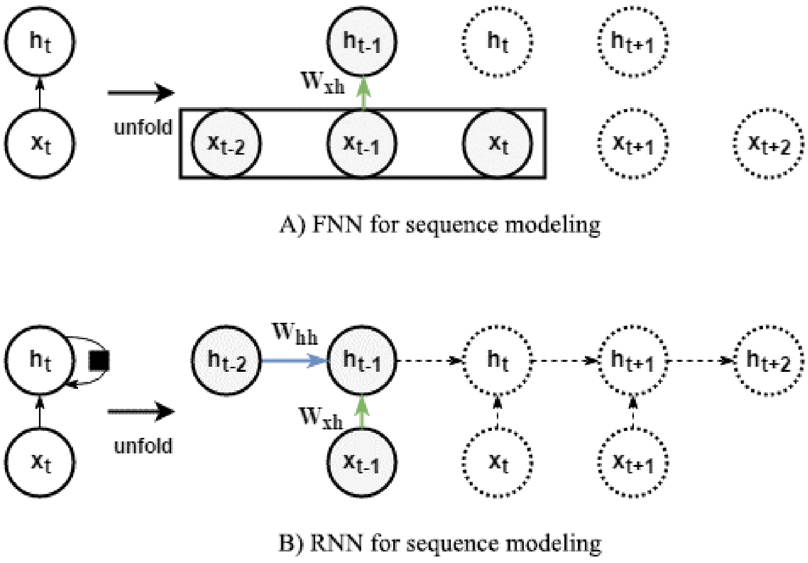


FIGURE 3.12: *FNN vs RNN for sequence modeling.*

Suppose that we want to model temporal information of a speech. When using FNNs, we will choose a window size and concatenate all the vectors inside the window to take into account the past and future information. It needs a lot of different training examples to train such a FNN and will result in a very long vector even though many dimensions are redundant. However, if we assume that each output is calculated by the same function $f(x_t; W_{hh}, Wxh)$, in other words, the parameters W_{hh}, Wxh are shared at every timestep, the same amount of training examples can be used to train a better model, as shown in Fig.3.12.

Recurrent neural networks are networks with loops allowing information to persist. Consider what happens if we unroll the loop: Fig.3.13.

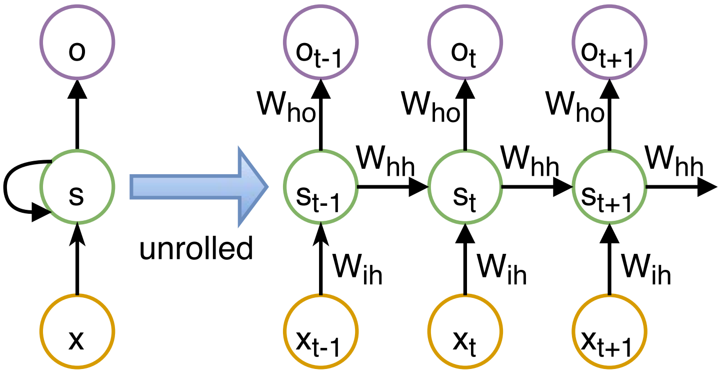


FIGURE 3.13: *An unrolled recurrent neural network*

The hidden state and output mathematical descriptions are Eq.(3.11) and Eq.(3.12). Compared to Feed-Forward Network, Recurrent Neural Network has "memory" ($\vec{s_t}$) which contains information about what happened in the previous time steps. The output at step $\vec{o_t}$ is calculated solely based on the memory at time t.

$$\vec{s_t} = tanh(W_{ih} \cdot \vec{x} + W_{hh} \cdot \vec{s_{t-1}} + \vec{b}) \tag{3.11}$$

$$\vec{o_t} = a(W_{ho} \cdot \vec{s_t} + \vec{b}) \tag{3.12}$$

From the equations we can also see that even the RNN has different structure, the basic linear transformation followed by an activation function remains. In other words, the basic idea behind every Neural Network is mappings between spaces.

With this structure, RNN can further allow us to map spaces between sequences of vectors, or in the most general case shown in Fig.3.14.

Backpropagation Through Time

Recurrent networks rely on an extension of backpropagation called backpropagation through time (BPTT). As shown in Fig.3.15, to calculate gradient for x_3 we also need to apply chain rule to backpropagate gradients all the way to $t = 0$.

Vanishing and Exploding Gradients

The idea of Vanilla recurrent network is beautiful, but it has problems in

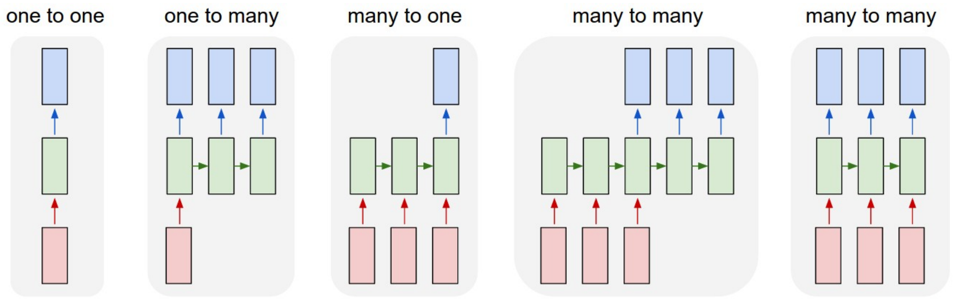


FIGURE 3.14: *Types of mapping*

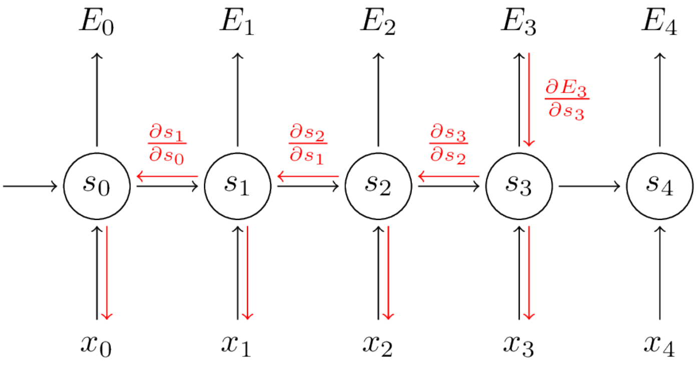

E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

FIGURE 3.15: *Backpropagation Through Time*

practice: *Vanishing and Exploding Gradients*. The vanishing gradient problem was originally discovered by Sepp Hochreiter in 1991. The gradient for E_3 in Fig.3.15 is calculated as: Eq.3.13, from which we can see that there are two factors that affect the magnitude of gradients: the weights and the derivatives of their activation functions. If either of these factors is smaller than 1.0, the other gradients in previous layers will be driven towards 0. Thus, with small values in the matrix and multiple matrix multiplications the gradient values are shrinking exponentially fast, eventually vanishing completely after a few time steps. The information at those steps doesn't contribute to what you are learning. The fact is that you end up not learning long-range dependencies. Conversely, if the weights in this matrix are large it can lead to a situation where the gradient signal is so large that it can cause learning to diverge. This is often referred to as *exploding gradients*. For detailed explanation[46].

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_k} \right) \frac{\partial E_3}{\partial W} \tag{3.13}$$

On the other hand, the derivatives of activation functions like *Tanh/sigmoid* are smaller than 1.0 for all inputs except 0 as shown in Fig.3.16.

Vanishing gradients also happen in deep Feedforward Neural Networks. It's just that RNNs tend to be very deep (as deep as the timesteps), which

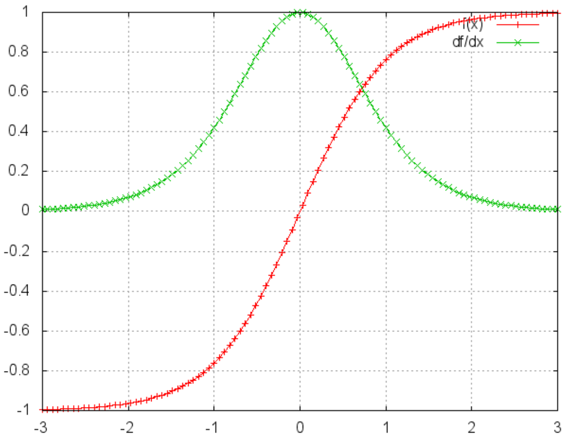


FIGURE 3.16: *Derivative of tanh*

makes the problem a lot more common.

One way to reduce the effect of vanishing gradients is proper initialization of the Weight matrix. A more preferred solution is to use **ReLU**, since the ReLU derivative is a constant of either 0 or 1. But the most popular way is to use *Long Short-Term Memory (LSTM)* or *Gated Recurrent Unit (GRU)*.

Bidirectional RNNs

Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, if we predict a missing word in a sentence like "How ( ) you been?", we consider both "How" and "you been?". Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs(such as the sum of outputs of both directions at a particular timestep) Fig.3.17.

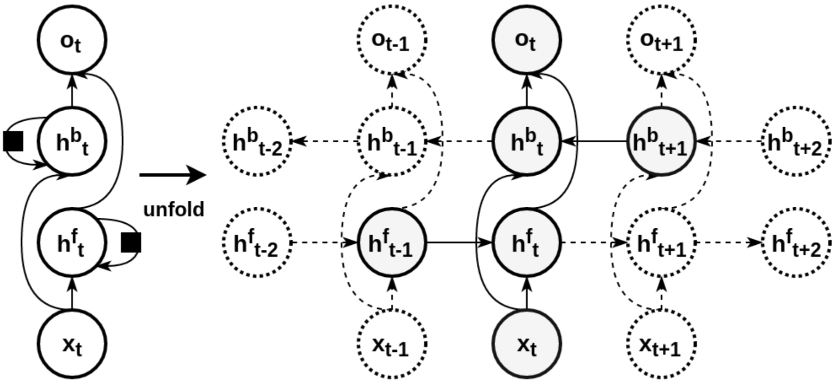


FIGURE 3.17: *Bidirectional RNN*

Deep RNNs

We can also stack multi-layers in RNN architecture, which gives us higher learning capacity, but we also need a lot of training data and the model is easy to overfit. Fig 3.18.

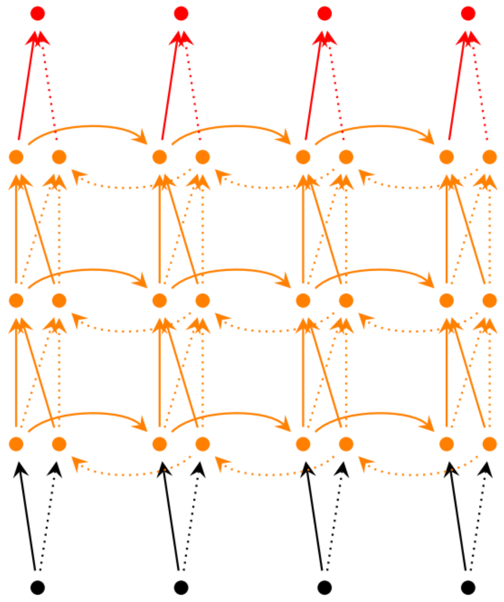


FIGURE 3.18: *Deep Bidirectional RNN*

3.2.2 Long Short-Term Memory

LSTMs are explicitly designed to avoid the long-term dependency problem. LSTM model introduces a new structure called a *memory cell*. A memory cell is composed of four main elements that serve to modulate the interactions between the memory cell itself and its environment:

1. **input gate**: One allows incoming signal to alter the state of the memory cell or block it, whose equation is Eq.3.14

$$i_t = sigmoid(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + b_i) \tag{3.14}$$

2. **forget gate**: One allows the state of the memory cell to have an effect on other neurons or prevent it, whose equation is Eq.3.15

$$f_t = sigmoid(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + b_f) \tag{3.15}$$

3. **output gate**: One modulates the memory cell's self-recurrent connection, allowing the cell to remember or forget its previous state, as needed, whose equation is Eq.3.16

$$o_t = sigmoid(W_{xo} \cdot x_t + W_{ho} \cdot h_{t-1} + b_o) \tag{3.16}$$

4. **self-recurrent**: One has a weight of 1.0 and ensures that, barring any outside interference, the state of a memory cell can remain constant from one timestep to another. The current cell is calculated by using all the gates and previous cell together as Eq.3.17. $\odot$ is elementwise multiplication and the $i_t \odot tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c)$ is the candidate

state value filtered by input gate i_t.

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c) \tag{3.17}$$

i, f, o are just vanilla neural networks with sigmoid function. They have the same equation but different parameters corresponding physical meanings. LSTM cell can also be thought as a combination of filters learned by several neural networks that all take current input x_t and previous hidden stateh_{t-1} as inputs. Finally, the current hidden state is calculated as Eq.3.18, where the output activation function $tanh$ can be changed to ReLU or else. Fig.3.19 Illustrates how a LSTM cell combats vanishing gradients through a gating mechanism.

$$h_t = o_t \odot tanh(c_{t-1}) \tag{3.18}$$

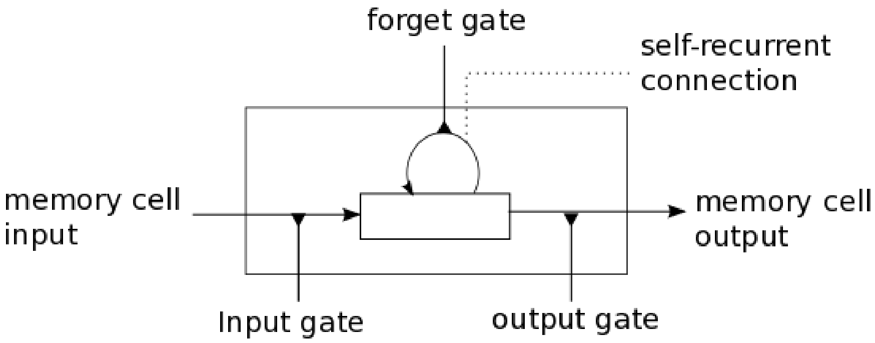


FIGURE 3.19: *Illustration of an LSTM memory cell*

Several excellent articles on LSTMs [47][48].

3.2.3 Gated Recurrent Unit

Another popular gating mechanism based method is Gated Recurrent Unit, introduced by[49]. The idea behind a GRU layer is quite similar to that of a LSTM layer. The structure is illustrated in Fig.3.20. It only has two gates, the reset gate and an update gate that combines the forget and input gates. It also merges the cell state and hidden state. The resulting model is lighter than standard LSTM models.

1. **reset gate**: One determines how to combine the new input with the previous memory, whose equation is Eq.3.19

$$r_t = sigmoid(W_{xr} \cdot x_t + W_{hr} \cdot h_{t-1} + b_r) \tag{3.19}$$

2. **forget gate**: One defines how much of the previous memory to keep around, whose equation is Eq.3.20

$$u_t = sigmoid(W_{xu} \cdot x_t + W_{hu} \cdot h_{t-1} + b_u) \tag{3.20}$$

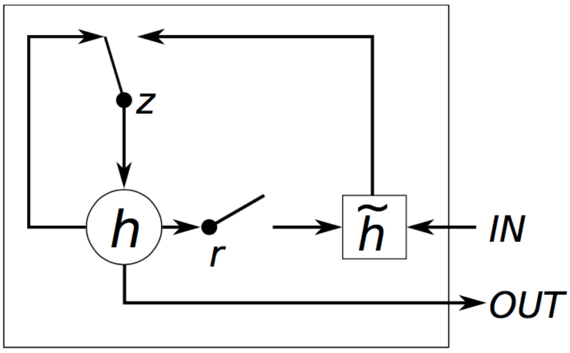


FIGURE 3.20: *GRU Gating*

3. **self-recurrent**: The current cell is calculatedas Eq.3.21.

$$c_t = tanh(W_{xc} \cdot x_t + r_t \odot (W_{hc} \cdot h_{t-1}) + b_c) \tag{3.21}$$

4. **hidden state**: The current hidden state is calculatedas Eq.3.22.

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot c_t \tag{3.22}$$

Which to use
According to empirical evaluations in [50] and [51], in many tasks both architectures perform similarly. But GRUs have fewer parameters and may train a bit faster or need less data to generalize. On the other hand, if you have enough data, the greater expressive power of LSTMs may lead to better results.

3.2.4 Training

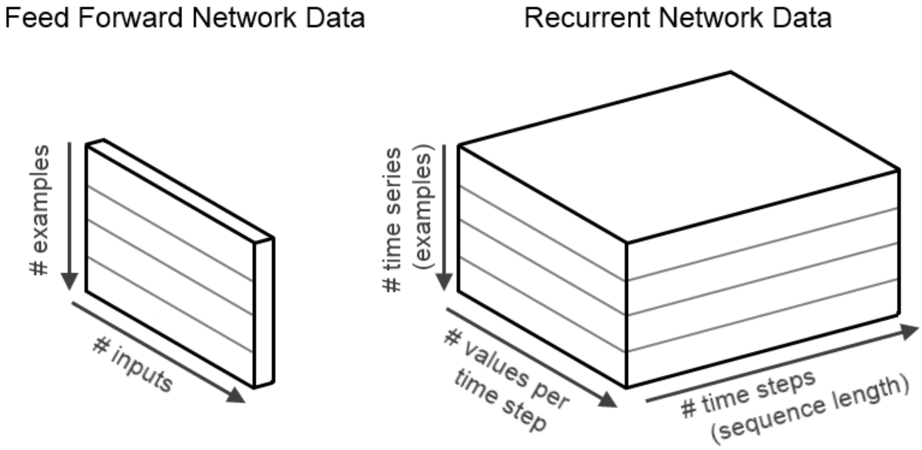


FIGURE 3.21: *RNN training inputs*

Because the RNNs also look at the previous information, therefore the ways we process the data for training and loss function are a little different.

Data preprocessing
We need to introduce anther dimension *timestep* for RNNs training. The input of each example is on longer a vector $\vec{x}$, but a sequence of vectors, which makes it matrix whose first dimension is timestep. If we use Mini-batch gradient descent to train the model, then the input data will become a 3D tensor with a shape of **[batch size, timestep, vector dimension]** as illustrated in Fig.3.21.

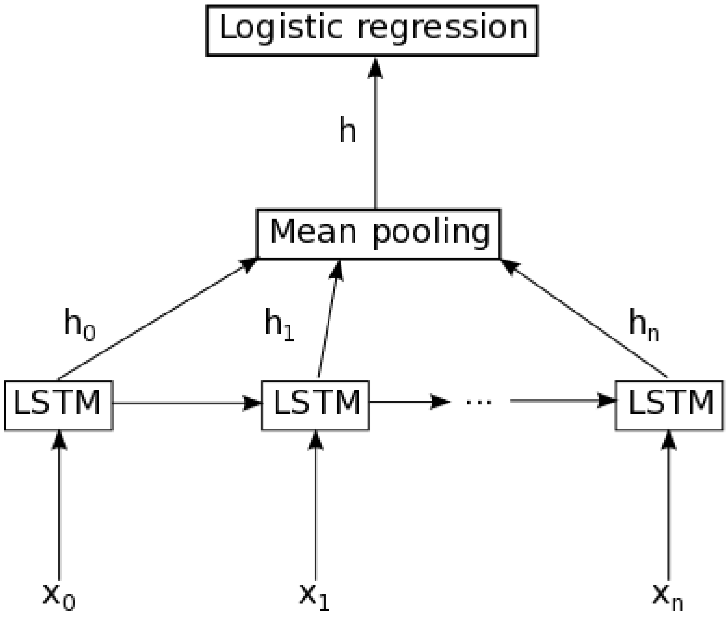


FIGURE 3.22: *The average loss over all timesteps is the finial loss*

Loss function
Depending on the type of mapping shown in Fig.3.14, the loss function also varies. If it's many-to-one mapping, the network will just minimize loss for one step. If it's many-to-one mapping such as language model, then the finial loss can be the average over all timesteps as Fig.3.22.

3.2.5 RNN Architectures

As mentioned in Fig.3.14, Feed Forward Neural Network can do the one-to-one mapping and Recurrent Neural Network extend it to many-to-many or many-to-one mapping, both of which can be used in Distillation and Articulatory Inversion tasks and lead to different RNN architectures. I listed 3 architectures I used in my experiments below.

Many-To-Many
Many-To-Many mapping is commonly used in language model task and

can also be used in here. The two sequences to be mapped in my experiments have the same length and the architecture is shown in Fig.3.23. The final loss is the average over all timesteps.

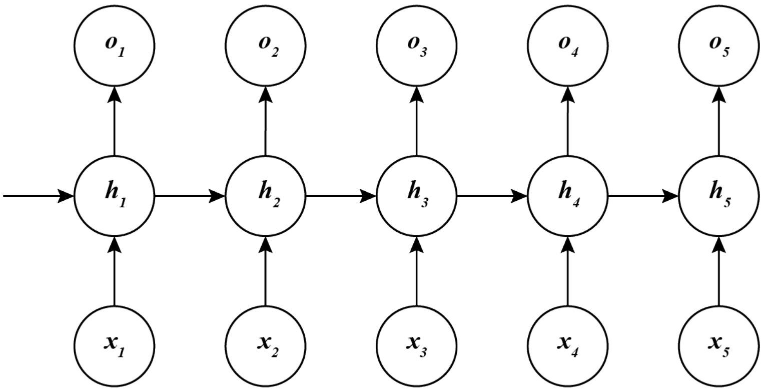


FIGURE 3.23: *Many-To-Many*

Many-To-One

Depending on which state the model predicts, the architecture also changes a bit. The idea of this one is to use historical information to predict the last state. The architecture is shown in Fig.3.24 and the final loss is just the loss of last state.

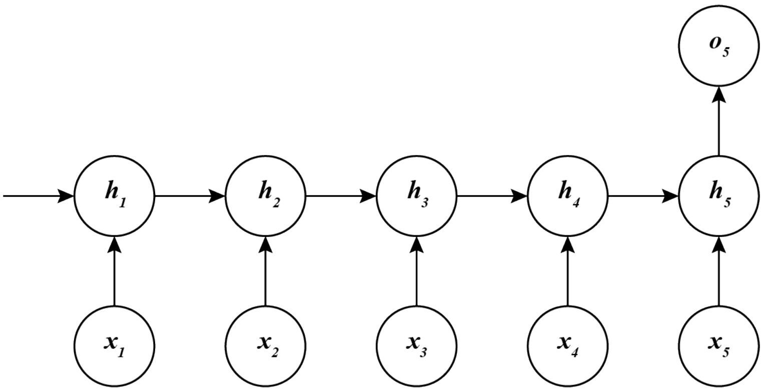


FIGURE 3.24: *A certain mount of historical states to current state*

3.3 Convolutional DNN

Another neural network introducing parameter-sharing mechanism is convolutional neural networks. Consider the situation in Fig.3.25 where we want to detect whether an "L" shape exists. When using FNN shown as Fig.3.25 A), there are $16 \times n$ parameters (n is the number of channels) to be learned. A "L" shape appearing at left-down corner and a "L" shape appearing at top-right corner are different for FNN, since they are detected with different parameters. In order to train a good enough model, we need a lot of examples with "L" at different locations, But why do we have to learn the same thing many times?

3.3.1 Spatial Sharing

The property that we want a neural network to have is called *translation invariance*, which can be achieved by using a "smaller" network at different local many times. As shown in Fig.3.25 B), the assumption that a function for detecting "L" can be shared at different locations is true, we only need to train a small neural network with $4 \times n$parameters, which reduces the amount of the training example required.

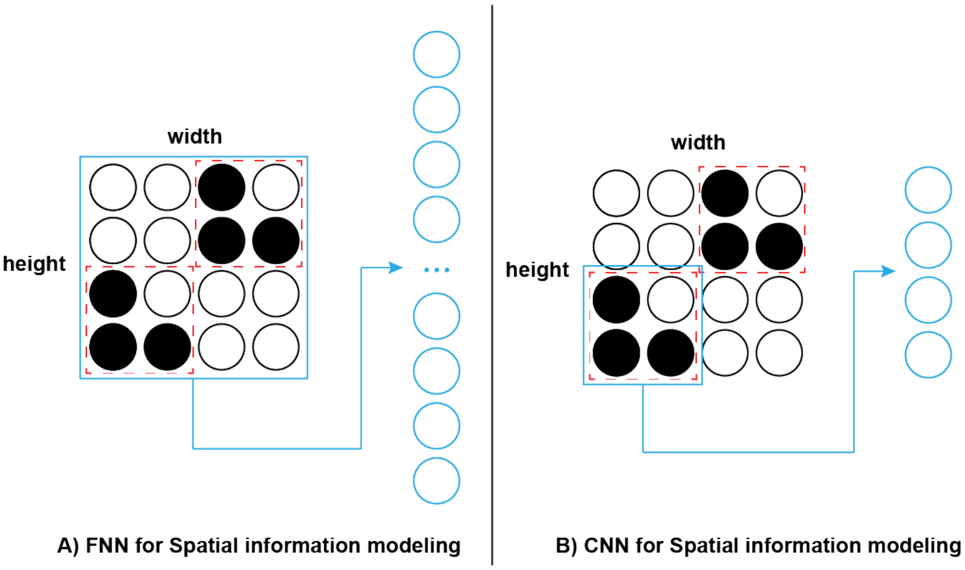

width
height
...
A) FNN for Spatial information modeling
width
height
B) CNN for Spatial information modeling

FIGURE 3.25: *FNN vs CNN for Spatial information modeling. Each circle represents a pixel, which is a vector with n dimensions, n is the number of channels (e.g. n is 3 for an RBG image with 3 channels). Blue box is locating the dimension of the input to the neural network and the red box points out the shape to be detected.*

3.3.2 Reception Field

Now we can choose the size and the shape of the *reception field* or *kernel size*, which is the partial area of the image the "smaller" network (CNN) can see at each glance. If we set the kernel size of a CNN as the whole image, then it is equivalent to a FNN.

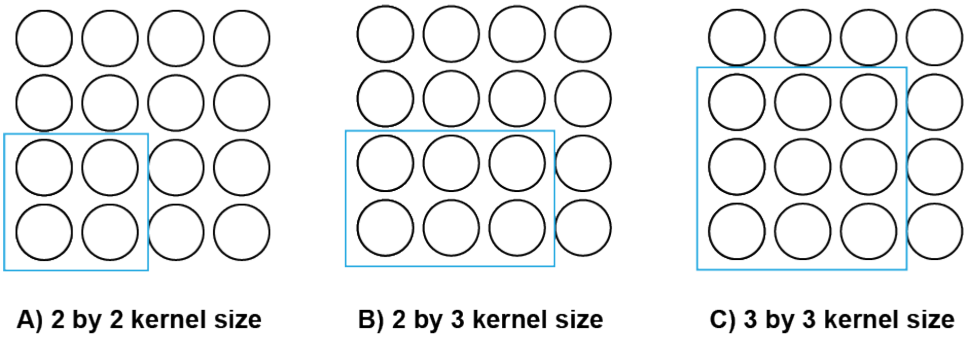


FIGURE 3.26: *reception fields with different size*

n by m reception field
The standard size of a reception field is (n by m), for example, the kernel sizes of (2 by 2), (2 by 3) or (3 by 3) shown in Fig.3.26. The output dimension of the "smaller" network is determined by *"the number of filters"*. Unlike FNN, a 2D-CNN each time actually transforms a vector $x \in R^{n \times m \times d}$ inside the input tensor into the $y \in R^k$, where k is the number of filters, d is the number of channels, and (n, m) is the kernel size. A CNN keep doing the transformation until it have seen the whole image and keep the structure of all the outputs to form a new tensor.

1 by 1 reception field
The special type of n by m size is 1 by 1. Although it does not take into account the spatial information, it can be used to adjust the dimensions of each pixel, as shown in Fig.3.27.

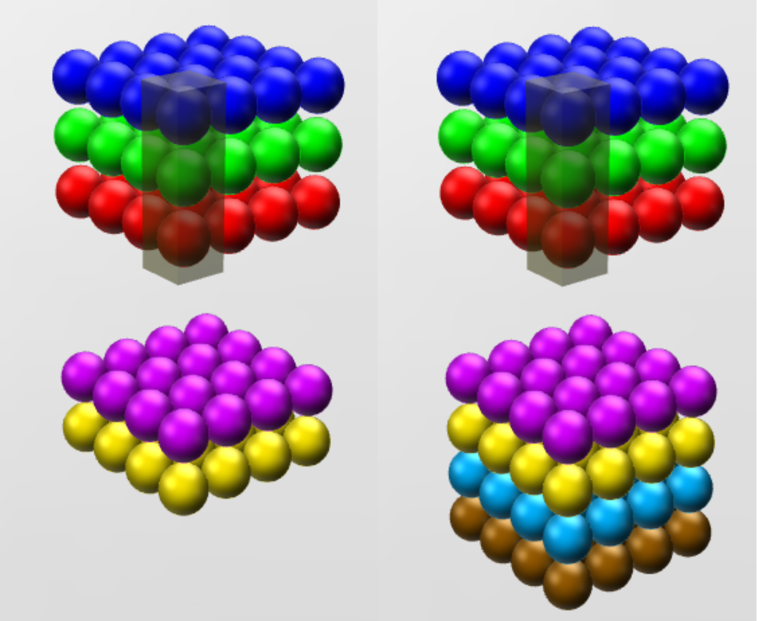


FIGURE 3.27: *1 by 1 reception field. Each pixel of the original image is represented as a 3-dimensional vector, which can be adjusted to a 2-dimensional or a 4 dimensional vector using 1 by 1 reception field*

Dilated reception field
The reception field has not necessarily to be a rectangle, for example, it can be a rectangle with holes (also called *Dilated*), as shown in Fig.3.28. This type of convolutional layer was proposed in [52] which intend to reduce the redundant parameters in different layers. When using multiple successive convolutional layers, dilated reception field can still support exponential expansion of the receptive field without loss of resolution or coverage. (a) F1 is produced from F0 by a 1-dilated convolution; each element in F1 has a receptive field of 3 by 3. (b) F2 is produced from F1 by a 2-dilated convolution; each element in F2 has a receptive field of 7 by 7. (c) F3 is produced from F2 by a 4-dilated convolution; each element in F3 has a receptive field of 15 by 15. The number of parameters associated with each

layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

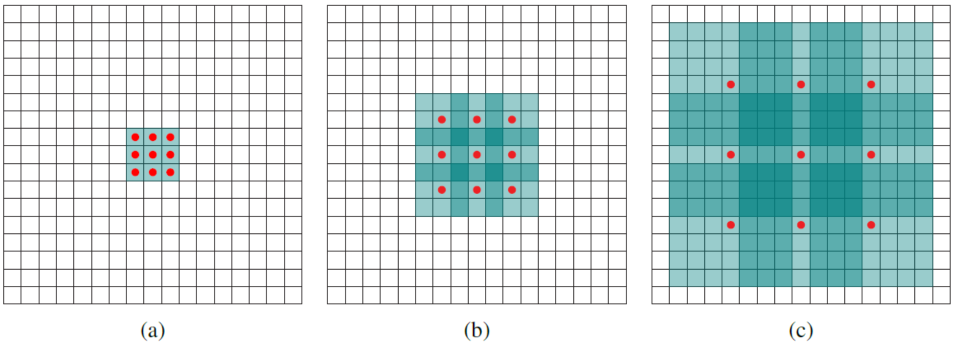


FIGURE 3.28: *reception field with holes [52]*

Graph reception field
However, all the structures of reception fields mentioned above are *Euclidean Structure*. CNNs with this reception field don't perform well on datasets coming in the form of graphs or networks: social networks, knowledge graphs, protein-interaction networks. In the last couple of years, a number of papers re-visited this problem of generalizing neural networks called graph convolutional network, as shown in Fig.3.29, to work on arbitrarily structured graphs, such as works in [53] [54].

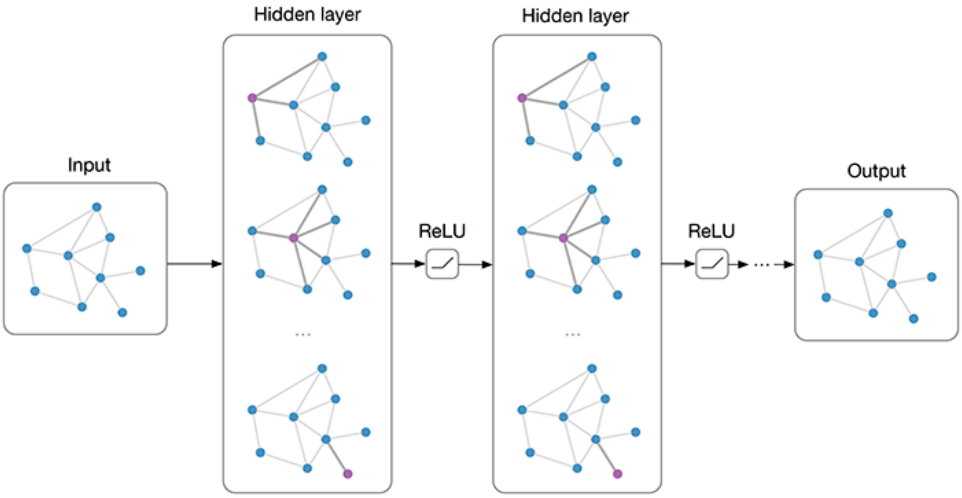


FIGURE 3.29: *Multi-layer Graph Convolutional Network (GCN) with first-order filters [55].*

3.3.3 Stride and Padding

The next thing we need to consider is how to slide this small network so that it can see all the pixels in an image. The number of pixels by which the reception field slide is called *stride*. For example, the blue box in Fig.3.30 represents the first step and red box represents the second step, if the stride is 2, then it moves to the second step by two pixels. We keep this operation until it slide the whole image.

However, this will get a output with a size smaller than the input. Zero padding can be used to control the size of output as shown by Fig.3.30 C).

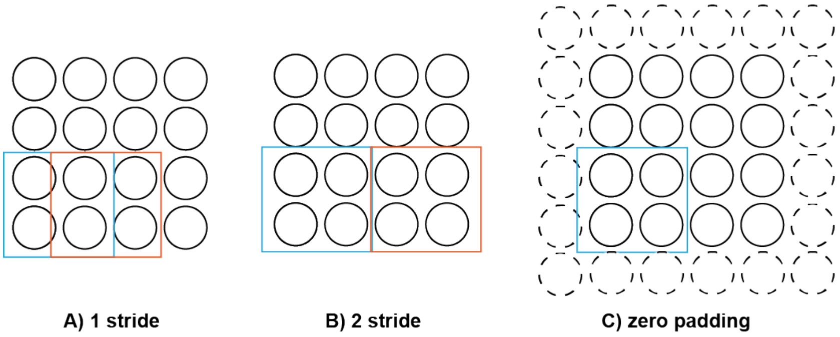


FIGURE 3.30: *Examples of different strides. The dashed circles are the zero padded pixels*

3.3.4 Convolutional Dimension

Depending on the rank of the input tensor, we can allow the convolutional layer to slide along 1 or 2 or 3 dimenson, resulting in 1D or 2D or 3D convolutional layer, as shown in Fig.3.31. 1D convolutional layer is usually used for sequence modeling, since it is essentially windowing. For example, a 1D convolutional layer with kernel size 2 is essentially a bigram model capable of learning weights automatically.

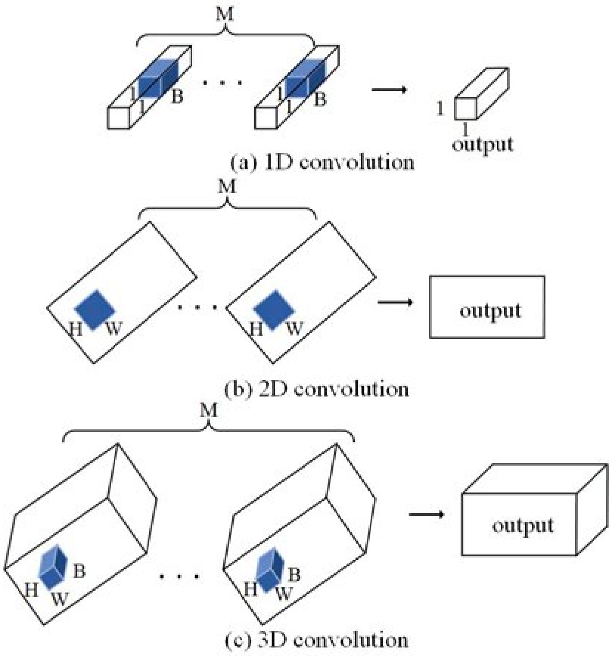
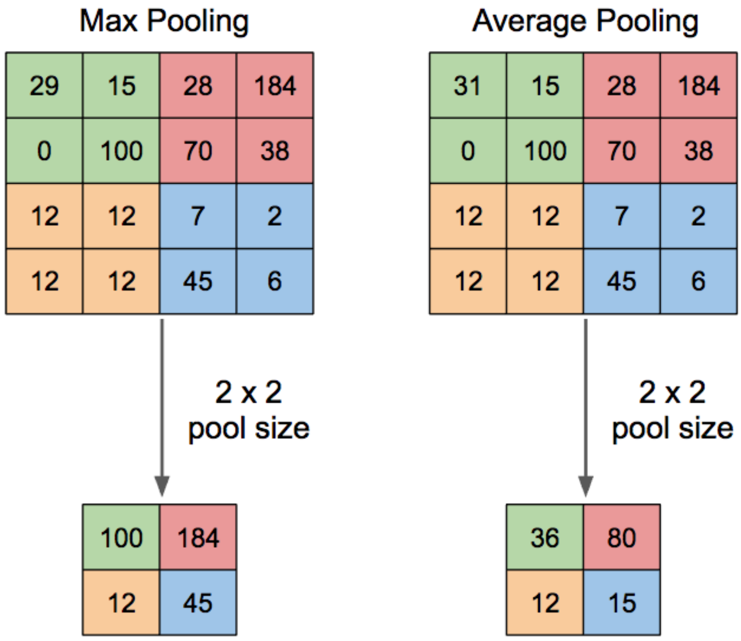


FIGURE 3.31: *Illustration of 1D, 2D, 3D convolution. B, H, W represent the size of kernel along spectral and spatial dimension respectively. M is the number of feature maps [56]*

3.3.5 Pooling Layer

The output of a convolutional layer is still a tensor with the same rank as the input tensor, We can use pooling layer to down-sample or up-sample tensors, as shown in Fig.3.32.



29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

100	184
12	45

31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

36	80
12	15

FIGURE 3.32: *Illustration of average and max pooling layers*

3.4 Auto-Encoder

Usually the input and output of a neural network are different, i.e. the input is the observation and the output is the prediction or target while there is a speical type of neural network called *autoencoder* whose input and the output are basically same, as shown in Fig.3.33. The aim of an autoencoder is to reconstruct the input. Although this structure is quite simple, but the hidden representation (aslo called *"bottleneck"*) between the encoder and the decoder has many good properties. Autoencoders can be used for dimension reduction and its compressed representation of the input data will be used for other machine learning tasks. An autoencoders can be used as a generative model and .

3.5 Word Embedding

Currently, word embedding has become a standard component for the DNN based natural language processing. It converts the one-hot representation of the word to a distributed representation [57], which has many benefits and allows to map words with similar meaning to similar values: the learning of one word can indirectly help the learning of the other words with similar meaning. This is especially helpful for tasks with small training data.

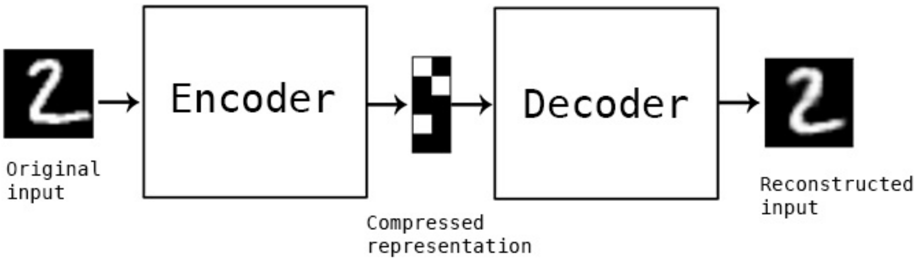


FIGURE 3.33: *Illustration of an autoencoder. The encoder and decoder could be several layers of any type of networks*

Word embedding achieves the by learning a projection matrix converting words into distributed representation, either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram, as shown in Fig.3.34.

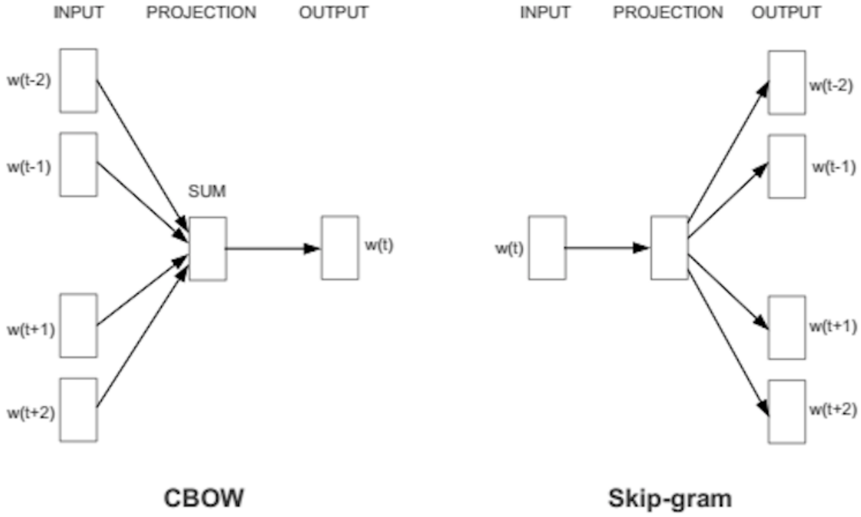


FIGURE 3.34: *Illustration of CBOW and skip-gram*

After learning a word embedding with CBOW or skip-gram, the distribution representation transformed from original input are more "meaningful" in terms of describing the relationship between words. The vectors obtained by subtracting two related words sometimes express a meaningful concept such as gender or verb tense, as shown in Fig.3.35 and Fig.3.36

There are some popular methods that are used for implementing word embedding.

Google's Word2vec

Word2vec is a method of computing vector representations of words introduced by a team of researchers at Google led by Tomas Mikolov [59]. The algorithm has been subsequently analysed and explained by other researchers. Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms and it is typically a shallow network with only one hidden layer that takes as its input a large

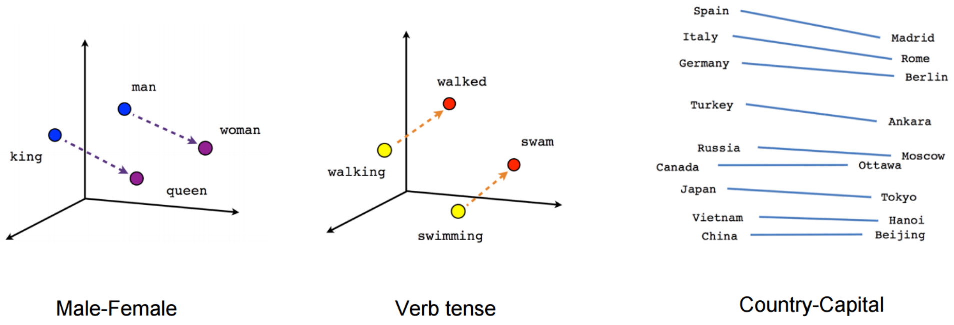


FIGURE 3.35: *Visualize Word Vectors*

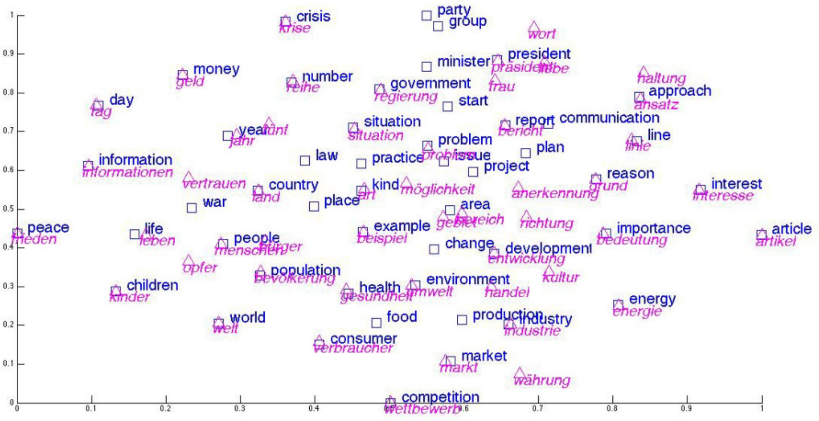


FIGURE 3.36: *After embedding, the English word and German word with the same meaning have similar vectors [58]*

corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space, as shown in Fig.3.37.

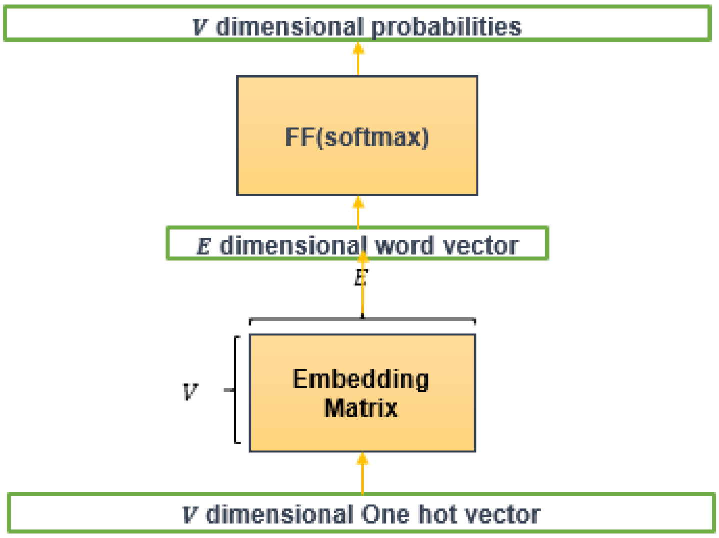

V dimensional probabilities
FF(softmax)
E dimensional word vector
E
V
Embedding Matrix
V dimensional One hot vector

FIGURE 3.37: *Word2vec architecture*

However, the problem of is neural network is that there are so many parameters to update, for example, if the vocabulary is 10000 and word dimension is 300, then it will need 10000×300 parameters to be updated. The authors of Word2Vec addressed these issues in [60].

There are three innovations in this second paper:

1. Treating common word pairs or phrases as single "words" in their model.
2. Subsampling frequent words to decrease the number of training examples.
3. Modifying the optimization objective with a technique they called "Negative Sampling", which causes each training sample to update only a small percentage of the model's weights and is not only reduced the compute burden of the training process, but also improved the quality of their resulting word vectors.

FastText

FastText is an extension to Word2Vec proposed by Facebook in 2016 [61]. Instead of feeding individual words into the Neural Network, FastText breaks words into several sub-words. The word embedding vector for apple will be the sum of all these sub-words and rare words can be properly represented since it is highly likely that some of their sub-words also appears in other words.

Chapter 4

Articulatory and Acoustic information integration

The Automatic Speech Recognition (ASR) technology as an ideal substitute for traditional user interfaces has been an important research topic since the 70s. Because the voice user interface is the most appropriate way to manipulate machines for human and has an inherent advantage in the situations where our eyes or hands are fully occupied on other things and is the only choice for pocket-sized devices such as Google Glass, Watch Phones which tend to be too small to hold buttons or touchscreen for user input.

Nowadays ASR technology is becoming good enough to enable many exciting applications, yet current ASR systems still suffer from acoustic variabilities such as background noises, speakers, accents, recording conditions, etc. In order to make the systems more reliable and robust, researchers have been trying to utilize additional information such as articulatory organs (lips, tongue, velum, etc.) movements, which is more suitable to model the coarticulation effects [62]. The modalities we are going to fuse for this task are illustrated in Fig.4.1.

The rest of this chapter is organized as follows. Section 4.1 describes the basic knowledge of automatic speech recognition system; The different between conventional GMM-HMM-based acoustic modeling and DNN-HMM-based acoustic modeling as well as the decoding and evaluation of them. Section 4.2 talks about the motivation of integration articulatory information, feature based vs model based integration and their related studies. Section 4.3 presents the proposed joint inversion method as well as its training and testing procedure. Section 4.4 presents generalized distillation as well as its training and testing procedure. Section 4.5 reports the experiments we did; The database, settings, results as well as some discussions. Finally, section 4.6 will be a summary of this chapter.

4.1 Background

A typical structure of ASR systems is illustrated in Fig.4.2.

Speech signal first will be converted into a sequence of fixed size acoustic vectors $o = o_1, o_2, ...o_n$ and the decoder tries to find the most probable sequence of words $w = w_1, w_2, ...w_m$ given the sequence of acoustic vectors. Eq.4.1.

$$\hat{w} = \arg\max_{w} P(w|o) \tag{4.1}$$

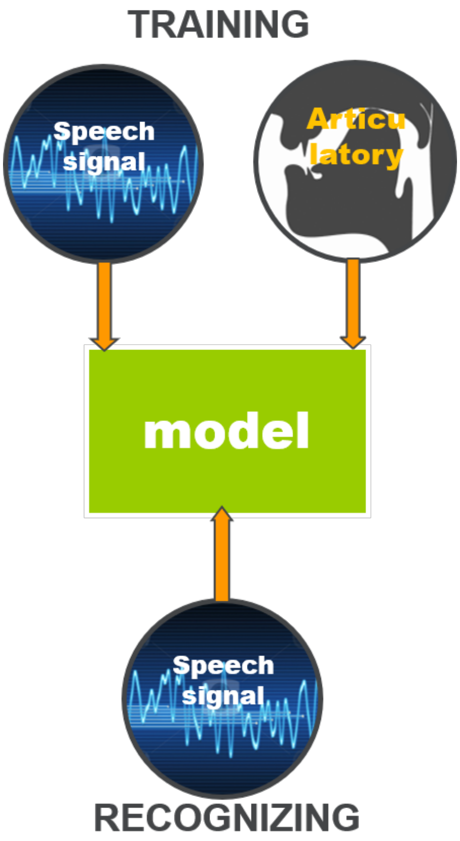


FIGURE 4.1: *The diagram of a system using articulatory + acoustic modalities. This task should use articulatory information only during training and needs to recognize speech without articulatory observations.*

However, $\arg\max_w P(w|o)$ is difficult to find directly. Here, Bayes' Rule is applied to transform Eq.4.1 into the equivalent Eq.4.2.

$$\hat{w} = \arg\max_{w} P(o|w)P(w) \tag{4.2}$$

Then, the likelihood $P(o|w)$ is found by *an acoustic model* and the prior probability $P(w)$ is found by *a language model*.

The basic unit of speech represented in a standard acoustic model is called *phoneme* (For example, the word "bad" is composed of three phones /b/ /ae/ /d/). For any word w, its acoustic model is the combination of several phoneme models, which is defined in *a pronunciation dictionary*.

4.1.1 Language Modelling

The prior probability $P(w)$ of a sequence of words $w = w_1, w_2, ...w_K$ is given by Eq.4.3, where N is typically in the range $2-4$. This is called *N-gram Language Model*. It gives the probability of a word based on previous

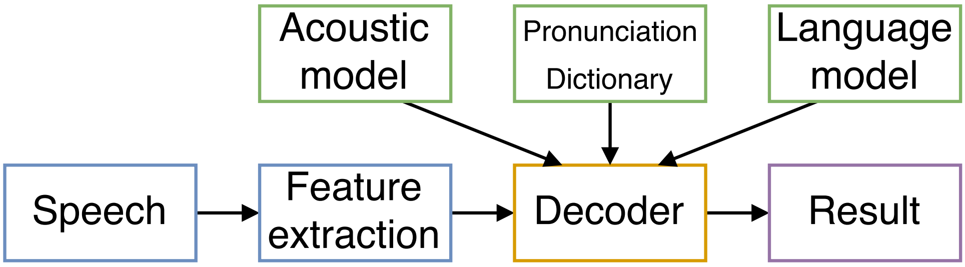


FIGURE 4.2: *Block diagram of a typical Automatic Speech Recognition System*

$N - 1$ words.

$$P(w) = \prod_{k=1}^{K} P(w_k | w_{k-1}, w_{k-2}, ..., w_{k-N+1}) \tag{4.3}$$

The N-gram probabilities are estimated from training texts by counting N-gram occurrences to form maximum likelihood (ML) parameter estimates. ML estimates uni-gram, bi-gram, tri-gram respectively: Eq.4.4,Eq.4.5,Eq.4.6, where $C()$ the occurrences of word or word sequence in the training texts.

$$P(w_n) \approx \frac{C(w_k)}{C(ALL)} \tag{4.4}$$

$$P(w_n | w_{n-1}) \approx \frac{C(w_{k-1}, w_k)}{C(w_k)} \tag{4.5}$$

$$P(w_n | w_{n-2}, w_{n-1}) \approx \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-1}, w_k)} \tag{4.6}$$

However, in practice there never be enough data to estimate all possible probabilities and the probability of unseen word estimated in this way will become 0. to avoid zero probability events, probability estimates are smoothed. Several smoothing methods are listed below:

1. **Back-off**: If an N-gram doesn't exist, back-off to smaller order [63].
2. **Laplace smoothing**: Assume that N-grams occur one more time than they actually do.
3. **Model interpolation**: Interpolate with lower order N-grams.
4. **Good-Turing discounting**: Redistribute probability mass from "seen" events to "unseen" events by discounting counts.
5. **Class N-gram models**: Cluster the similar words together.

4.1.2 Acoustic Modelling

Modern speech recognition systems are mainly based on Hidden Markov Models for finding the likelihood $P(o|w)$.

Hidden Markov model

The speech that a ASR system detects is generated from the internal physical changes in the human body. The internal physical changes are hidden, but the resulting sounds are observable. Hidden Markov Model (HMM) [64] is used to estimate the sequence of internal physical changes by knowing the sequence of resulting sounds.

An HMM model in ASR task is characterized by the following 5 components (2 sets of states and 3 sets of probabilities):

1. **Hidden States**: The internal hidden physical changes in human body.
2. **Observable States**: The resulting sounds generated by the internal physical changes.
3. **Initial State Probabilities**: A vector π of the initial state probabilities at time $t = 1$.
4. **Transition Probabilities**: matrix A storing every hidden state probability given the previous hidden state.
5. **Emission Probabilities**: matrix B storing every probability of observing a particular observable state given that the hidden model is in a particular hidden state.

The hidden state of the model depends only upon the previous n hidden states of the model. and each probability in A and B are time independent.

A left-to-right HMM for acoustic modelling looks like Fig.4.3.

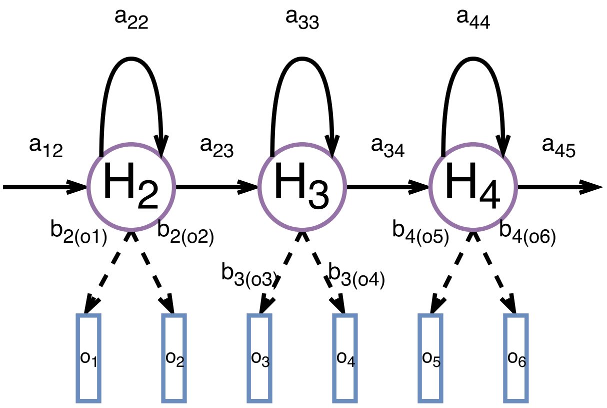


FIGURE 4.3: *HMM*

Using left-to-right topology is because speech is time-evolving, non-stationary signal. Paths from current state to previous state are not existing in ASR task neither. The joint probability that o is generated by an HMM model M moving through the hidden state sequence w is calculated as the product of transition probabilities and emission probabilities, which is Eq.4.7.

$$P(o, w|M) = a_{12}b_{2(o_1)}a_{22}b_{2(o_1)}a_{23}b_{3(o_3)}\cdots \tag{4.7}$$

Given w is hidden, the likelihood of $P(o|M)$ is computed by summing all possible state sequences $w = w(1), w(2), ..., w(T)$, that is Eq.4.8 and can be approximated by only considering the most likely state sequence, that is Eq.4.9.

$$P(o|M) = \sum_{w} a_{w(0)} a_{w(1)} \prod_{t=1}^{T} b_{w(t)(o_t)} a_{w(t)} a_{w(t+1)} \tag{4.8}$$

$$P(o|M) = \max_{w} \left\{ a_{w(0)} a_{w(1)} \prod_{t=1}^{T} b_{w(t)(o_t)} a_{w(t)} a_{w(t+1)}) \right\} \tag{4.9}$$

M_i is a Hidden Markov Model of word i. Since, $P(o|w_i) = P(o|M_i)$, the $P(o|w_i)$ can be found by computing $P(o|M_i)$.

The HMM is trained by using *Baum–Welch algorithm*[65].

GMM-HMM-based

In GMM-HMM-based Modelling, the $b_{w(t)(o_t)}$ is found by a Gaussian Mixture Model (GMM) [66] that computes probability density function (pdf) over a continuous space of input feature vectors.

DNN-HMM-based

Traditional ASR systems represent speech as a sequence of non-overlapping phonetic units while implicitly assuming that speech can be decomposed into disjoint acoustic segments, which limits the acoustic model's ability to properly learn the underlying variations in spontaneous or conversational speech. Such systems suffer from acoustic variabilities posed by the variations in different speaking styles, talkers, contexts.

But these variations have been well handled by DNNs. In the so-called DNN-HMM hybrid systems [67], [68], big performance boosts were achieved by replacing GMMs by a feedforward Neural Network (FNN) which takes a window of several frames as input and produces posterior probabilities over HMM states as illustrated in Fig.4.4.

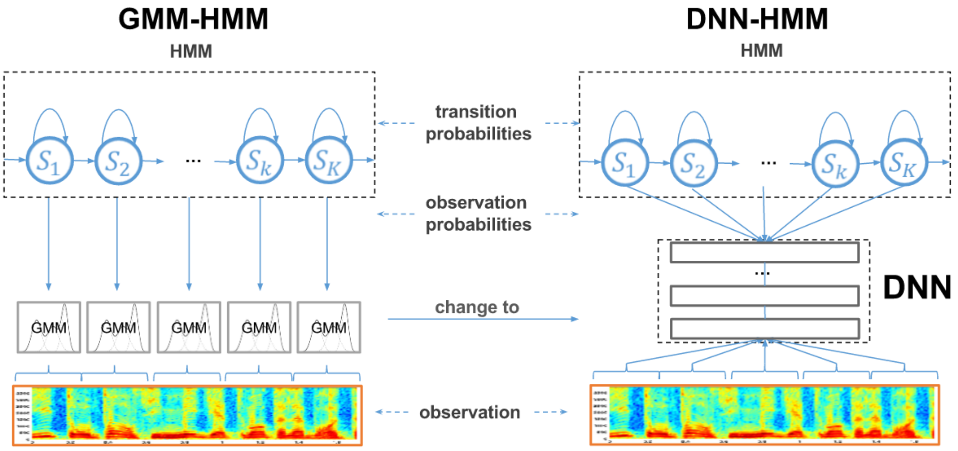


FIGURE 4.4: GMM-HMM vs DNN-HMM acoustic models.

In DNN-HMM hybrid systems, the $b_{w(t)(o_t)}$ is replaced with $P(o|h)$ found by a Deep Neural Network calculated using Eq.4.10, where h_t is a particular hidden state of a Hidden Markov Model of a word and o_t is observable vector.

$$P(o_t|h_t) = \frac{P(h_t|o_t)P(o_t)}{P(h_t)} \tag{4.10}$$

The prior probability of hidden state $P(h_t)$ is calculated from (occurrences of) the training set, and $p(o_t)$ can be assigned a constant since the observation feature vectors are regarded as independent of each other.

The DNN-HMM Training Procedure is listed below:

1. Train a standard GMM-HMM based recognizer.
2. Use force alignment to get the corresponding target label h_t of hidden state in the recognizer for every vector o_t as the DNN training data.
3. Count occurrences of hidden states in the training set to compute the prior probability of every hidden state $P(h_t)$.
4. Train a DNN that maps spaces o_t –> h_t (classification task of o_t).
5. With $P(h_t|o_t)$ got from the trained DNN model and $P(h_t$ compute $P(o_t|h_t)$ using Eq.4.10.
6. Use Eq.4.9 and Eq.4.2 to get the recognition results.

Phoneme Modelling

If using phoneme modelling, then every HMM represents a phoneme instead of a word.

4.1.3 Decoding

In decoding, the acoustic score and language model score are combined together to determine the most likely word sequence given observation sequence. However, there is often a significant mismatch between the dynamic range of the two scores. The dynamic range of the acoustic likelihood can be excessively high, which makes the effect of language model relatively small. Therefore, the language scores are often scaled, then the fundamental formula of speech recognition can be modified as:Eq.4.11 and the solution is given by the *Viterbi algorithm*[69].

$$\hat{w} = \arg\max_{w} P(o|w)P(w)^s \tag{4.11}$$

where s is language model scale (lm scale) factor.

4.1.4 System Evaluation

The metrics for speech recognizers are word/phone error rate (WER/PER) and the accuracy, that tells how the results recognized by the system differs from the orthographic transcripts. The word/phone error rate and accuracy are defined as follows: Eq.4.12 and Eq.4.13

$$WER/PER = \frac{S+D+I}{N} \tag{4.12}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions and N is the total number of words in the reference.

$$ACC = 1 - WER/PER = \frac{N - (S + D + I)}{N} = \frac{H - I}{N} \tag{4.13}$$

where H is $N - (S + D)$, the number of correctly recognized words.

4.2 Related Studies of Articulatory Information Integration

Although the ability of DNNs for modeling complex mappings addresses variations in different speaking styles, talkers, contexts and DNN-HMM based ASR systems perform fairly well for clearly articulated speech in "controlled" conditions, they still suffer from background noises, differences in recording devices etc. This motivated many researchers to incorporate the articulatory information into ASR systems.

A lot of prior studies like [62], [15], [70], [71], [72] have proved that articulatory information can improve the ASR performance and increase the robustness against noise contamination and speaker variation. However, incorporating such information is challenging since it is impractical to obtain observations of articulators movements in real-life speech recognition scenarios. This constraint requires ASR systems that are able to handle missing articulatory information during recognition and utilize articulatory information only during the training phase.

4.2.1 Feature based integration

One approach to incorporate the articulatory information is to utilize it at the feature level, which I call *feature based* approach.

The most straightforward and widely used implementation of this approach is the articulatory-to-acoustic inversion, where the missing articulatory features are generated from the acoustic signal. This, however, is not a simple task since the mapping between acoustic and articulatory data spaces is non-linear and not unique [73]. Various machine-learning methods haven been applied to model this mapping, for example, Hidden Markov Model (HMM) [74], Gaussian Mixture Model (GMM) [75], and Mixture Density Networks (MDN) [76]. The Canonical Correlation Analysis (CCA) used in [77] and its deep learning extension DCCA [78] are also the feature based approaches where transformations of the acoustic features are learned such that they become maximally correlated with the articulatory data. Since the Deep Neural Networks have become the new state-of-the-art tool in a wide range of application domains, multiple studies [79] [80] [81] have shown that DNNs' ability to learn highly non-linear and complex functions results in better prediction of articulatory trajectories from acoustic speech data. Several Deep Autoencoder (DAE) architectures for articulatory inversion are also investigated in [82].

In the conventional feature based approach (Standard Inversion), the articulatory inversion model and the acoustic model are trained separately.

Articulatory features are first generated using the inversion model and then combined with the acoustic features for acoustic model training. The same procedure is applied during recognition. We present the training and testing procedures of standard inversion using deep recurrent nerual network below.

Training Procedure
Given the training data $\{(x_i, a_i)\}_{i=1}^n$, we would like to learn a mapping f_{INV} from the acoustic space to the articulatory space by minimizing a mean squared error (MSE) loss function L with some form of regularization

$$f_{INV} = \arg \min_{f_{inv} \in \mathcal{F}_{INV}} \frac{1}{n} \sum_{i=1}^{n} L(f_{inv}(x_i), a_i) + \Omega(||f_{inv}||) \tag{4.14}$$

$$L(f_{inv}(x_i), a_i) = \frac{1}{q} \sum_{j=1}^{q} (f_{inv}(x_i)_j - a_{ij})^2. \tag{4.15}$$

Here, $x_i \in \mathcal{R}^p$ and $a_i \in \mathcal{R}^q$ are the acoustic and articulatory feature vectors, F_{INV} is a space of mapping functions from $\mathcal{R}^p$ to $\mathcal{R}^q$ and Ω is L2 norm regularizer.

The acoustic model f_{AC} maps concatenated feature vectors into HMM state probabilities through a softmax function and is trained by minimizing the categorical cross entropy loss function H:

$$f_{AC} = \arg \min_{f_{ac} \in \mathcal{F}_{AC}} \frac{1}{n} \sum_{i=1}^{n} H(\sigma(f_{ac}(x_i^*)), y_i) + \Omega(||f_{ac}||) \tag{4.16}$$

$$H(y_i, \sigma(f_{ac}(x_i^*))) = -\sum_{j=1}^{c} y_{ij} \log \sigma(f_{ac}(x_i^*)_j) \tag{4.17}$$

where, $x_i^* = concat(x_i, f_{INV}(x_i)) \in \mathcal{R}^{p+q}$ is the concatenated vector of acoustic and reconstructed articulatory features, $y_i \in \Delta^c$ is a vector representing target HMM states, Δ^c is the c-dimensional space of probability vectors, F_{AC} is a class of functions from $\mathcal{R}^d$ to $\mathcal{R}^c$, $\sigma : \mathcal{R}^c \to \Delta^c$ is the softmax function.

Testing Procedure
During testing, the inversion model is used to generate the articulatory features which are concatenated with the acoustic features and used as input to the acoustic model. The overall diagram is shown in Fig.4.5.

4.2.2 Model based integration

Another way to integrate articulatory information in the ASR systems is the *model based* approach, where the articulatory data are used to adjust the parameters and optionally the structure of the acoustic model in a way that does not require articulatory observations during testingas shown in Fig.4.6.

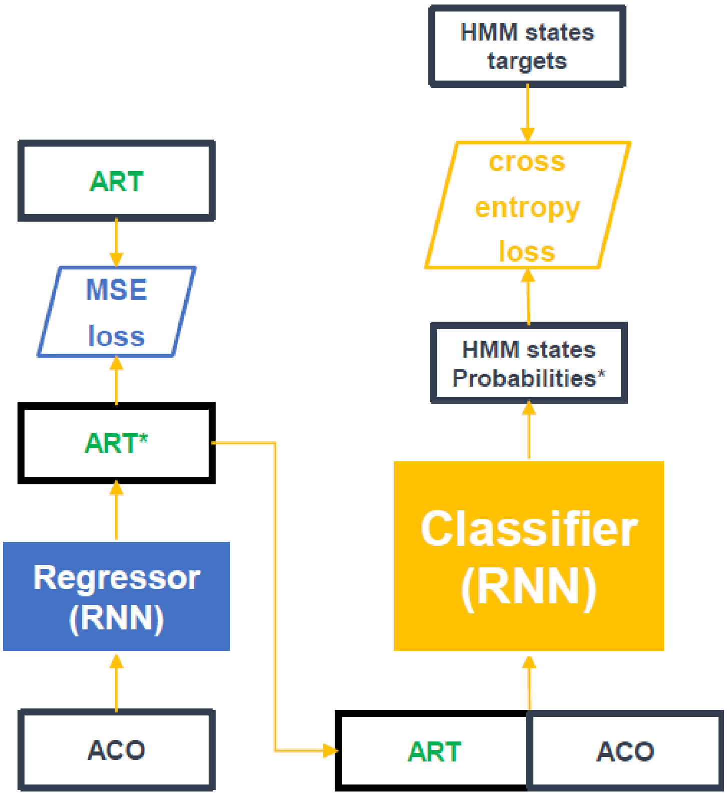


FIGURE 4.5: Block diagram of the standard inversion and acoustic model DNN training.

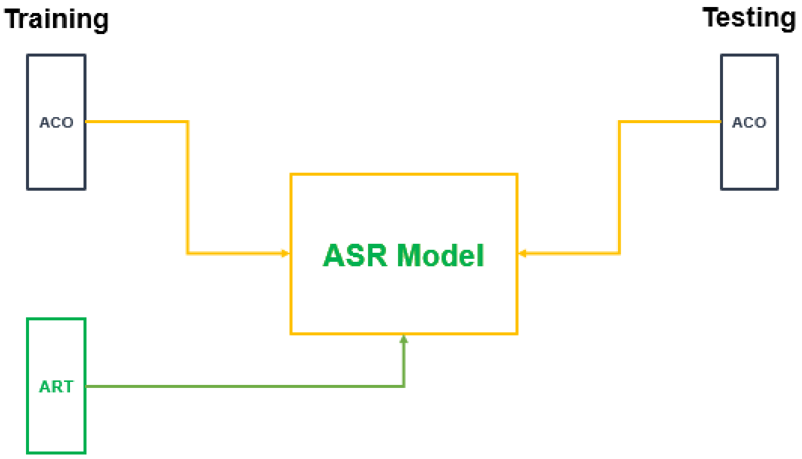


FIGURE 4.6: Block diagram of the model based articulatory information integration.

Obviously, in this case no articulatory inversion is necessary. In [15], a hybrid Bayesian network/HMM acoustic model incorporates the articulatory data.

A relatively new way to incorporate knowledge into neural networks is the so called *Distillation Training*, where an additional loss function with soft targets is also being minimized during training. In [16], Hinton et al. have shown that soft targets from complex models can transfer knowledge to small models that are easy to deploy.

Recently, the learning using privileged information [83] and the distillation methods have been combined into a *Generalized Distillation* framework [20] which utilizes the strengths of both methods. Knowledge is transferred through soft targets from a "teacher" model trained with additional features to a "student" model with no access to those features. In our previous work [84], we applied Generalized Distillation in a feedforward DNN-HMM system. The results showed that soft targets can transfer knowledge from the teacher trained with both articulatory and acoustic data to the student model learned from acoustic data only. In [85], the effectiveness of RNN models pretrained with soft targets was also investigated and compared with the distillation method. Both approaches lead to models that have higher generalization abilities.

4.3 Joint Inversion

The recognition result of standard inversion depends on the outputs of the inversion model. However, the inversion model is only trained to minimize its MSE loss with respect to the articulatory vector and does not have any knowledge about how its outputs will be used later. It would be helpful to tell the inversion model what the final goal is. Thus, we train the inversion model and the acoustic model as a single network so that the parameters of the two models are trained to minimize the final objective jointly.

The joint training procedure and the network structure are illustrated in Fig.4.7. The acoustic vector x_i is passed through the inversion DNN f_{inv} to calculate the MSE loss with respect to the articulatory vector a_i using Eq.(4.15). In addition, x_i concatenated with the inversion DNN output $f_{inv}(x_i)$ (which is expected to have a physical meaning of articulatory feature) and the vector $concat(x_i, f_{inv}(x_i))$ (denoted as x_i^*) is passed through the acoustic model DNN f_{ac} followed by a softmax output function σ to calculate the categorical cross entropy loss with respect to HMM state labels y_i using Eq.(4.17).

4.3.1 Training Procedure

During the training the entire network is trained by minimizing the weighted average of the two loss functions controlled by $\lambda \in [0, 1)$ using Eq.(4.18)

$$\begin{aligned} f_{INV}, f_{AC} \quad = \quad & \arg \min_{f_{inv} \in \mathcal{F}_{INV}, f_{ac} \in \mathcal{F}_{AC}} \frac{1}{n} \sum_{i=1}^{n} [(1-\lambda) \\ & H(\sigma(f_{ac}(x_i^*)), y_i) + \lambda L(f_{inv}(x_i), a_i) \\ & +\Omega(||f_{ac}|| + ||f_{inv}||)] \end{aligned} \tag{4.18}$$

We have to note that the weight λ which controls the contribution of each loss function in the weights update cannot be set to 1, because this will eliminate the HMM states as targets and destroy the acoustic model. On the other hand, $\lambda = 0$ means that articulatory targets are eliminated and no articulatory information is integrated.

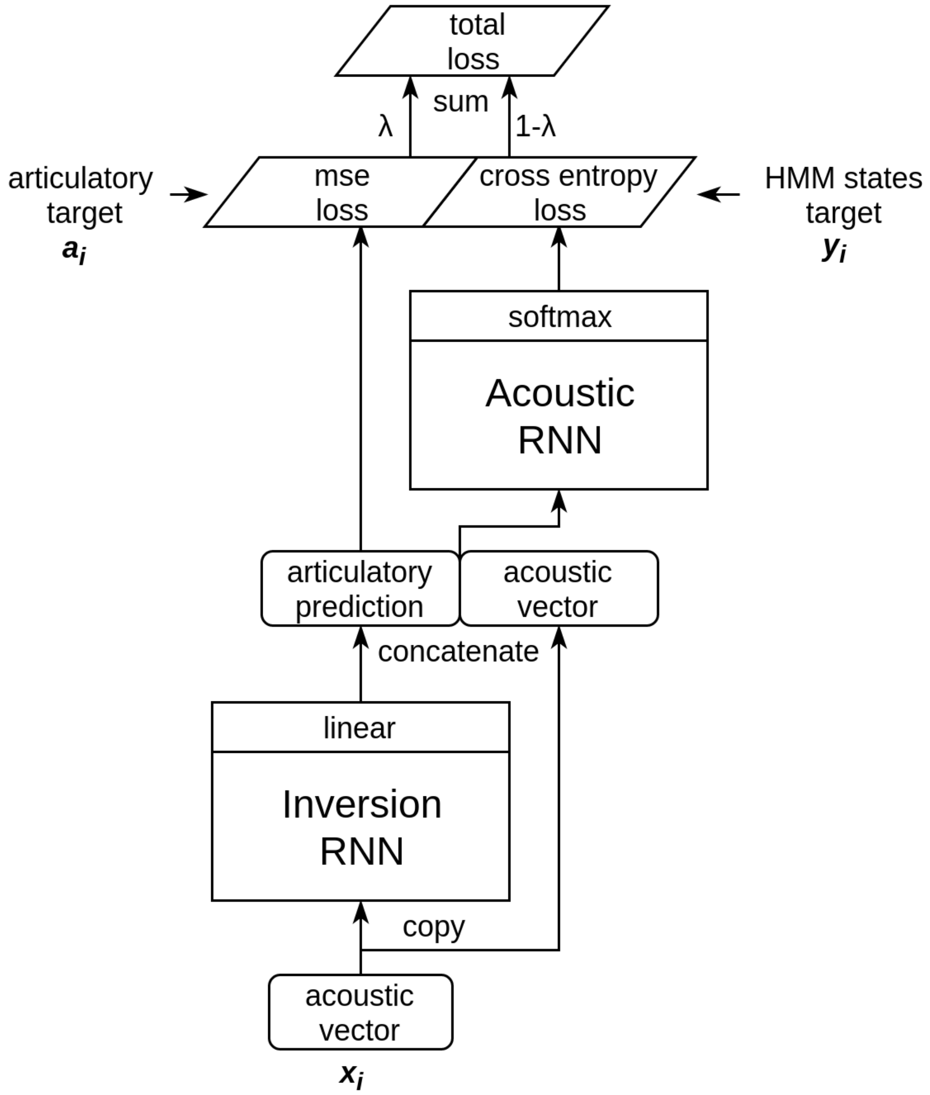


FIGURE 4.7: Block diagram of the joint articulatory inversion and acoustic model DNN training.

4.3.2 Testing Procedure

During testing, the HMM state probabilities obtained from the joint model are fed to the HMM decoder as shown in Fig.4.8. The only input data in this case are the acoustic features as in any articulatory inversion based system.

Joint Inversion allows the ideas of hybrid network and regularization of multimodal learning, though it is very simple and straightforward, its performance is very good, as we will see in the experiment section.

4.4 Generalized Distillation

Generalized distillation method has been proposed in [20] to combine two techniques - Hinton's distillation [16] and Vapnik's privileged information [83] which enables machines to learn from other machines. In this framework, an "intelligent teacher" is incorporated into machine learning and

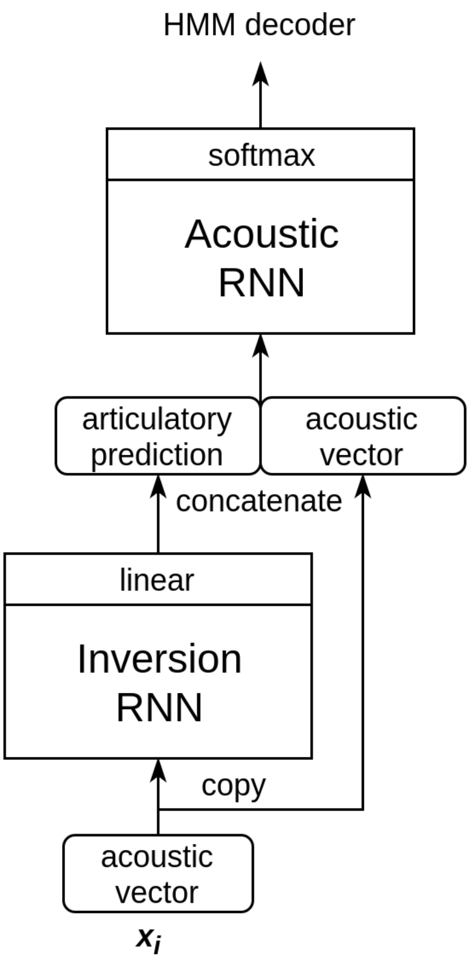


FIGURE 4.8: Block diagram of the joint articulatory inversion testing.

the training data is formed by a collection of triplets

$$(x_1, x_1^*, y_1), \ldots, (x_n, x_n^*, y_n) \sim P^n(x, x^*, y),$$

where (x_i, y_i) is a feature-label pair and x_i^* represents a privileged information about x_i provided by an intelligent teacher and is supposed to have higher discriminating power than x_i itself. The teacher is assumed to develop a language that effectively communicates information to help the student come up with better representation and to enable to it learn characteristics about the decision boundary which are not contained in the student training data. In our task, combined acoustic and articulatory feature vectors are regarded as privileged information source, $x_i^* = \text{concat}(x_i, a_i)$.

4.4.1 Teacher Training Procedure

The training procedure is as follows:

1. Learn teacher $f^T \in \mathcal{F}^T$ using $\{(x_i^*, y_i)\}_{i=1}^n$.

$$f^T = \arg \min_{f^t \in \mathcal{F}^T} \frac{1}{n} \sum_{i=1}^{n} l(y_i, \sigma(f^t(x_i^*))) + \Omega(||f^t||) \tag{4.19}$$

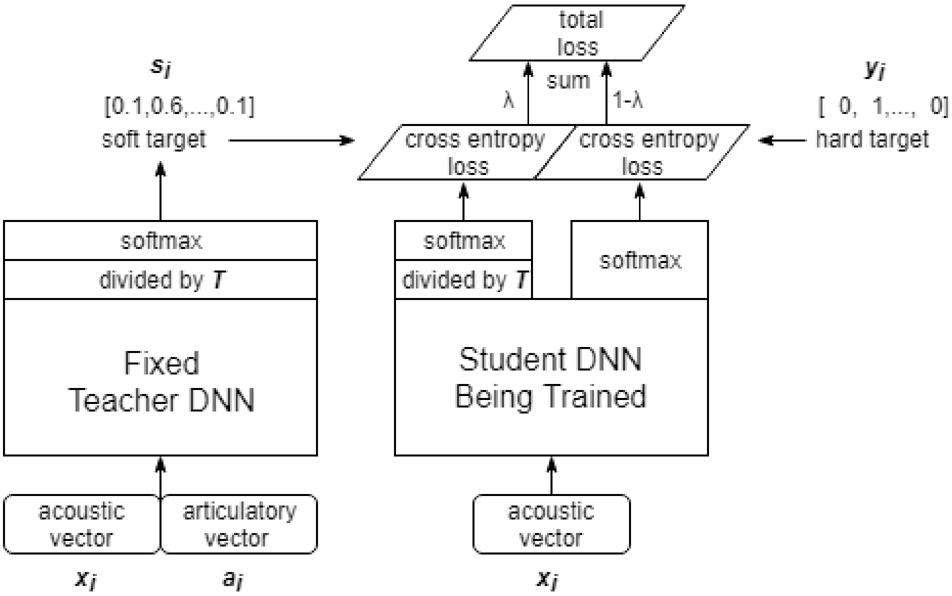


FIGURE 4.9: Student training block diagram. In contrast to hard targets y_i, soft targets s_i provide information about between class relations.

where, $x_i^* \in \mathcal{R}^d$, d is the total dimension of acoustic and articulatory features, $y_i \in \Delta^c$, $\mathcal{F}^T$ is a class of functions from $\mathcal{R}^d$ to $\mathcal{R}^c$, $\sigma : \mathcal{R}^c \to \Delta^c$ is a softmax function, l is a loss function (in our case, it is the categorical cross entropy from Eq.(4.17)). The paramters of teacher model are then fixed.

2. Compute teacher soft labels $\{s_i\}_{i=1}^n$ using temperature parameter T, where $s_i = \sigma(f^T(x_i^*)/T) \in \Delta^c$. T is normally set to 1. We use a higher value for T to soften the probability distribution over classes.

3. Learn student $f^S \in \mathcal{F}^S$ from Eq.(4.20) using $\{(x_i, y_i, s_i)\}_{i=1}^n$ and imitation parameter $\lambda \in [0, 1]$. Because the magnitudes of the gradients produced by the soft targets scale as $1/T^2$, multiplying the second loss by T^2 is necessary [16].

$$\begin{aligned} f^S &= \arg\min_{f^s \in \mathcal{F}^S} \frac{1}{n} \sum_{i=1}^{n} [(1-\lambda) l(y_i, \sigma(f^s(x_i))) \\ &+ \quad T^2 \lambda l(s_i, \sigma(f^s(x_i)/T))] \end{aligned} \tag{4.20}$$

4.4.2 Student Training Procedure

The student DNN training procedure is illustrated in Fig.4.9. The outputs of the teacher DNN softened by the temperature parameter T are used as soft targets s_i and together with the hard targets y_i act as arguments of the student DNN loss function as in Eq.(4.20). The input training data for the student DNN consists of acoustic features only. The corresponding concatenated acoustic and articulatory features, are given to the teacher DNN input in order to calculate the soft targets. However, only the student DNN parameters are updated during this procedure.

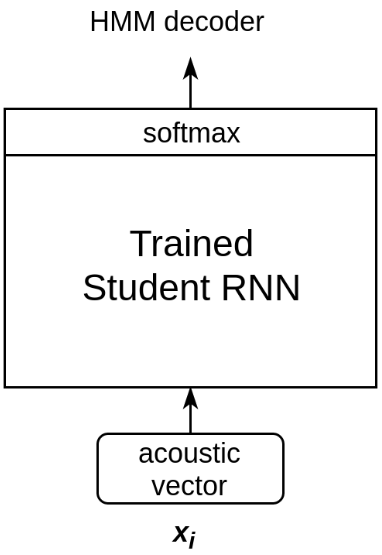


FIGURE 4.10: Testing with student DNN. No extra cost is required during the test.

4.4.3 Testing Procedure

During testing, only the student DNN is used and the state probability predictions from the "hard" output, i.e. the output that was compared with the hard targets during training, are fed to the HMM decoder as shown in Fig.4.10. Student DNN model trained using this method does not need articulatory feature nor extra computational resources during testing and is as fast as the standard DNN acoustic model.

4.5 Experiments

4.5.1 Database Description

We experimented with the University of Wisconsin X-ray microbeam database (XRMB) [86] which consists of simultaneously recorded acoustic and articulatory measurements from 47 American English speakers (22 males, 25 females). Each speaker's recordings comprise at most 118 tasks whose type can be number sequence, TIMIT sentences, isolated word sequence, paragraph as well as non-speech oral motor. The order and content of records was the same for all speakers.

Task list

Task list is shown in Table.4.1. Only normal speed sentences and number sequence tasks were used in the experiments for now.

1. Word type: seven words selected randomly from the full list of citation words.
2. Sentence type: three different examples selected from the sentence list.
3. Long passages of connected speech: were partitioned(the last and first sentences of each contiguous subdivision were repeated).

TABLE 4.1: *Task List.*

Record type	Number
word: standard	40
word-like: vcv, cvc, vseq, v	4
sentence: normal	36
sentence: fast and slow	6
sentence: clear	1
sentence: emphasis	2
paragraph	6
swallow	10
diadochokinetic	3
counting (1-20)	1
number sequences	5
oral gym: wag and protrude	4

Data Forms

There are three forms of data in XRMB database:

1. Orthographic transcripts of the spoken utterance.
2. Digitized waveforms of the recorded speech(sample rate: 21.74 kHz).
3. Simultaneous 2D articulators' trajectories ($([n, 16])$), where row is time, sampled every 6.866 milliseconds and column is x and y-coordinate history for 8 pellets on the tongue, lips, and jaw as shown in Fig.4.11.

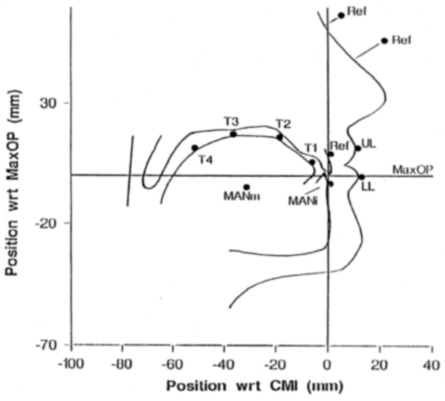


FIGURE 4.11: *Placement of the 8 pellets on T1,T2,T3,T4,MANm, MANi,UL,LL points.*

Pronouncing Dictionary

In my experiments, the pronouncing Dictionary is the Carnegie Mellon University (CMU) Pronouncing Dictionary without stress [87]. The CMU dictionary is an open-source machine-readable pronunciation dictionary for North American English that contains over 134,000 words and their pronunciations. The current phoneme set has 39 phonemes, not counting varia due to lexical stress. Along with silence, there are 40 phonemes in my systems, which are listed in A.

TABLE 4.2: Details of the train, validation, and test sets.

	Train	Test	Validation	Total (Unique)
Speakers	32	6	4	42
Female	17	3	2	22
Male	15	3	2	20
Sentences	2652	491	295	3438 (81)
Words	24632	4105	2506	31243 (213)
Phonemes	86067	13407	8220	107694 (39)
Hours	2:12:46	0:22:8	0:14:6	2:49:0

4.5.2 Data Processing

Acoustic Data Processing

We downsampled the acoustic signal from 21.74 kHz to 16 kHz, and my acoustic features are 13-dimensional mel-frequency cepstral coefficients (MFCCs) computed every 10ms over a 25ms window, along with their first and second derivatives, resulting in 39 dimensional frames.

Articulatory Data Processing

We also down-sampled articulatory data from the original rate of $145.7Hz$ to $100Hz$ to match the frame rate ($10ms$) of acoustic features and use the x,y coordinates of the 8 articulators along with their first and second derivatives as articulatory feature vectors of 48 dimensions. Including the first and second derivatives of the articulatory data is helpful since the movement itself can't tell apart speech pause from other phones.

Data Cleaning

Due to limitations in the recording technologies, articulatory measurements contain missing data when individual pellets are mistracked. Though there are methods to reconstruct missing data [88], we decided to use only complete data samples. Phoneme alignment was done using the Penn Phonetics Lab Forced Aligner [89] and the missing entries, as well as speech data which are not consistent with their orthographic transcripts, were removed. Utterances are split into files, each containing only one sentence with silence parts at the beginning and end reduced to 150 ms. After excluding the speakers who had only few utterances left, our dataset was reduced to about 3 hours, while the whole database has 19 hours in total. All experimental results are obtained from a 7-fold cross validation with 6 speakers for testing, 4 speakers for validation and 32 speakers for training in each fold. Unlike many other studies, this makes our models as *speaker-independent* as possible. Table 4.2 summarizes the details about our data sets.

4.5.3 GMM-HMM baseline

The global information about systems used in the experiments is listed below:

1. HMM model: standard 3-state left-to-right monophone model

2. Phoneme language mode: a simple bi-gram trained on data transcriptions including the paragraph task

3. HMM states: 120 states

Speaker Dependent
To see how recognizers using the data in XRMB preforms, four conventional GMM-HMM recognizers were built using HTK Toolkits:

1. only use acoustic features (39D)

2. use MFCCs and features of T1,T2,T3,T4 articulators (39+24D)

3. use MFCCs and features of T1, T2, T3, T4, UL, LL articulators (39+36D)

4. use MFCCs and features of all articulators (39+48D)

The phoneme error rate (PER) of speaker JW45 (102 utterances) when using different number of Gaussian components are collected in Fig.4.12. The results verified as well that with articulatory information, the ASR can achieve much better preference. However, the recognizer with all articulators is not as good as the one with T1, T2, T3, T4, UL, LL articulators. This is probability due to the lack of data and also indicate the pattern of articulators MANm, MANi is not as clear as others.

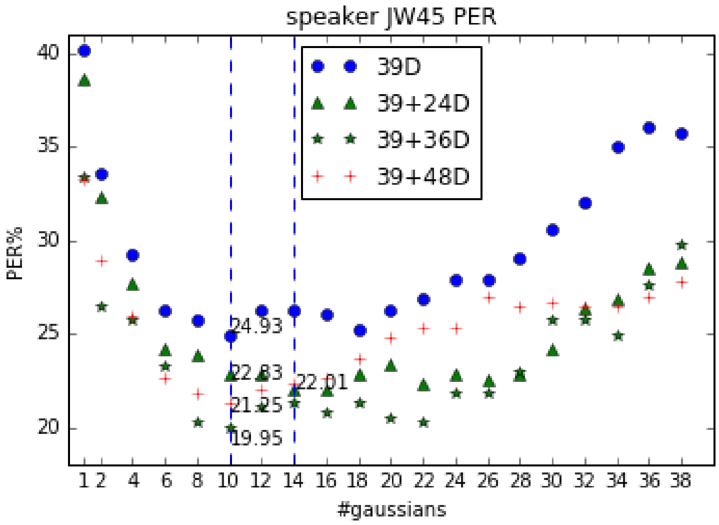


FIGURE 4.12: *Results of speaker JW45*

Speaker independent
The recognizers above also were also trained in speaker independent (Table.4.2) task and the results are collected in Fig.4.13. This time, with more data, the articulatory information gives big improvement. The systems with more than 38 Gaussian components don't give clear improvement, therefore the system with 38 Gaussian components was chosen to be the baseline.

After optimized the language model's weight/penalty, the final results decoded using Julius are summarized in Table.4.3. Frame level DNN training targets were generated from this GMM-HMM system.

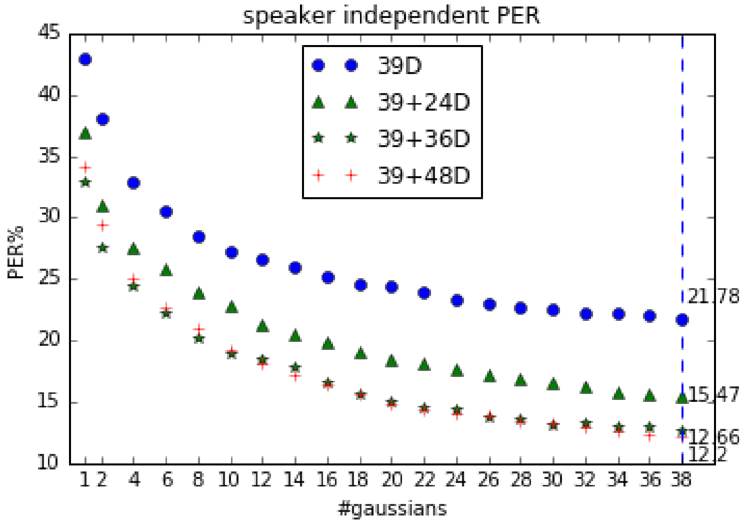


FIGURE 4.13: *Results in speaker independent task*

TABLE 4.3: *Phone error rates for conventional GMM-HMM system.*

LM weight/penalty	**MFCC**	**MFCC+ART**
0/0	29.95	12.01
7.0/2.0	18.85	9.67
7.0/1.0	18.73	9.72

4.5.4 Common DNN settings

In our experiments, we used RNN for both the acoustic and inversion models. DNNs have a lot of hyper parameters, such as number of layers, number of nodes, activation function type, etc. In a series of preliminary experiments, we tried various RNN structures and parameters in order to achieve the best possible baseline performance. Finally, we chose two biGRU layers stacked in between feedforward dense layers which showed the best performance. Similar findings are reported in [90] [91]. Although in principle a DNN with more recurrent layers should be able to provide similar performance, yet it takes much longer to propagate the information through the recurrent layers than feedforward layers and deeper RNNs easily become over-fitted after several epochs of training. Thus, most DNNs in our experiments have following hidden layers: 2 feedforward (F) layers with ReLU activation followed by 2 biGRU (B) layers on top of which there are another 2 ReLU feedforward layers. This structure is denoted as FFBBFF and is shown in Fig.4.14 for both the acoustic and inversion DNNs. The other common settings are summarized in Table. 4.4.

For regularization, a dropout layer with 30% dropout rate is inserted after every feedforward layer and biGRU layer and L2 regularizations of feedforward layers with a rate of 1e-3 are also added to the final loss, which is $10^{-3}\sum(\|\theta\|^2)/2$ and θ is the weights of a layer.

For the training, a gradient clipping norm of 10.0 is set to prevent the exploding gradient and the weights of gates in GRU layers are initialized using orthogonal matrix initialization [92], which we found important for

TABLE 4.4: Common DNN parameters

Regularization	dropout (30%), L2 (1e-3)
Regression loss function	MSE
Regression Output activation	Linear
Classification loss function	Categorical Cross-entropy
Classification Output activation	SoftMax
Hidden feedforward activation	ReLU
Hidden feedforward nodes	2048 per layer
GRU (per direction) nodes	1024 per layer
Update	Adam

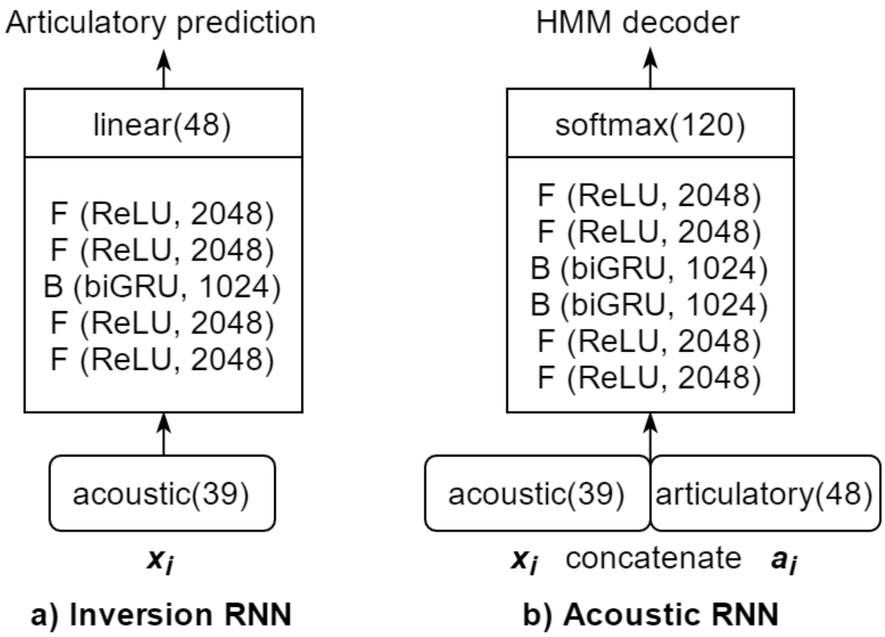


FIGURE 4.14: Inversion and acoustic RNN structures. The number of nodes and the activation function of each layer is given. Note that the biGRU layer consists of two GRUs layers, so the number of nodes is for each of them.

the training. Finally, all DNNs were first trained with 9e-5 learning rate and fine-tuned with 5e-6 learning rate once the validation data losses did not go down for 3 epochs.

4.5.5 DNN-HMM baseline

Settings

As a baseline, we adopt a system where the inversion model and acoustic model are trained separately. The inversion model architecture is FFBFF illustrated in Fig.4.14-a. The inputs and outputs are the MFCC (39dim) and articulatory feature (48dim) vectors respectively.

Results

Our acoustic model architecture is FFBBFF and is shown in Fig.4.14-b. The train data x_i^* are concatenated acoustic and generated articulatory vectors (87dim) and the "hard" targets y_i are one-hot vectors (120dim) where the component corresponding to the target state is 1 and all other components are set to 0.

Although impractical, it is possible to train and evaluate the system performance using the true articulatory data. This would give us the maximum achievable performance, or in terms of phoneme error rate, the PER lower bound. On the other hand, performance of the system trained on acoustic data only would serve as the PER upper bound. Any PER in between those bounds would show improvement, but the goal is to get as close as possible to the lower PER bound.

Figure 4.15 shows those bounds for the GMM-HMM and RNN-HMM acoustic models. Previously, we have built an DNN-HMM model with feedforward layers only, and its results are also shown as FNN-HMM.

Obviously, models using the articulatory features are always better than those without them. As the model becomes more and more powerful, i.e. GMM→FNN→RNN (whose numbers of parameters are about 0.8, 20, 42.4 millions respectively), the gap between the upper and lower bounds reduces significantly.

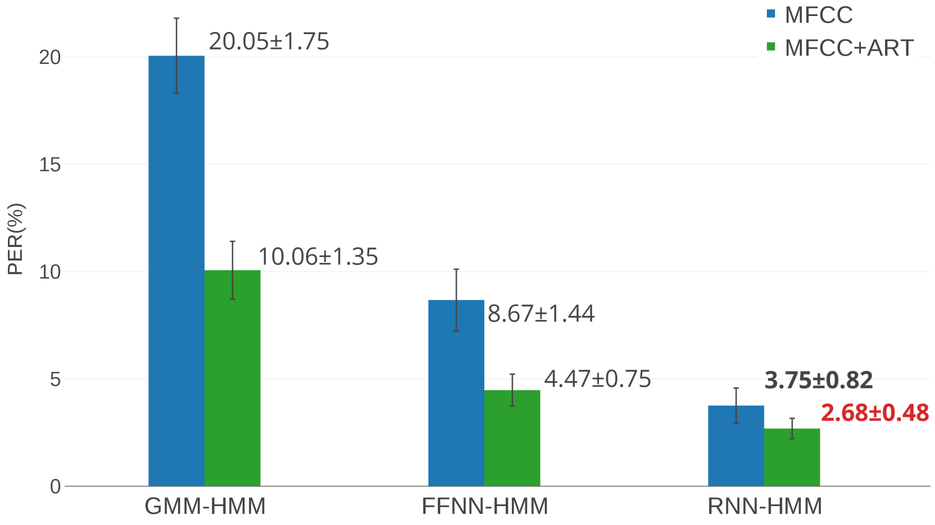


FIGURE 4.15: The PER results of different acoustic models with and without true articulatory (ART) features. The numbers correspond to the upper and lower PER bounds for each model.

TABLE 4.5: Speaker independent inversion results (the 1st and 2nd derivatives are excluded).

Inversion & acoustic Model	RMSE	r	PER(%)
FNN	0.632±0.021	0.770±0.029	7.35±1.04
RNN	0.618±0.023	0.923±0.006	**3.15±0.59**

4.5.6 Inversion Baseline

First, we investigated how our models perform the acoustic-to-articulatory mapping. Table 4.5 shows the inversion results for a 5 hidden layers feedforward network (FNN) with input window size of 17 frames and the RNN

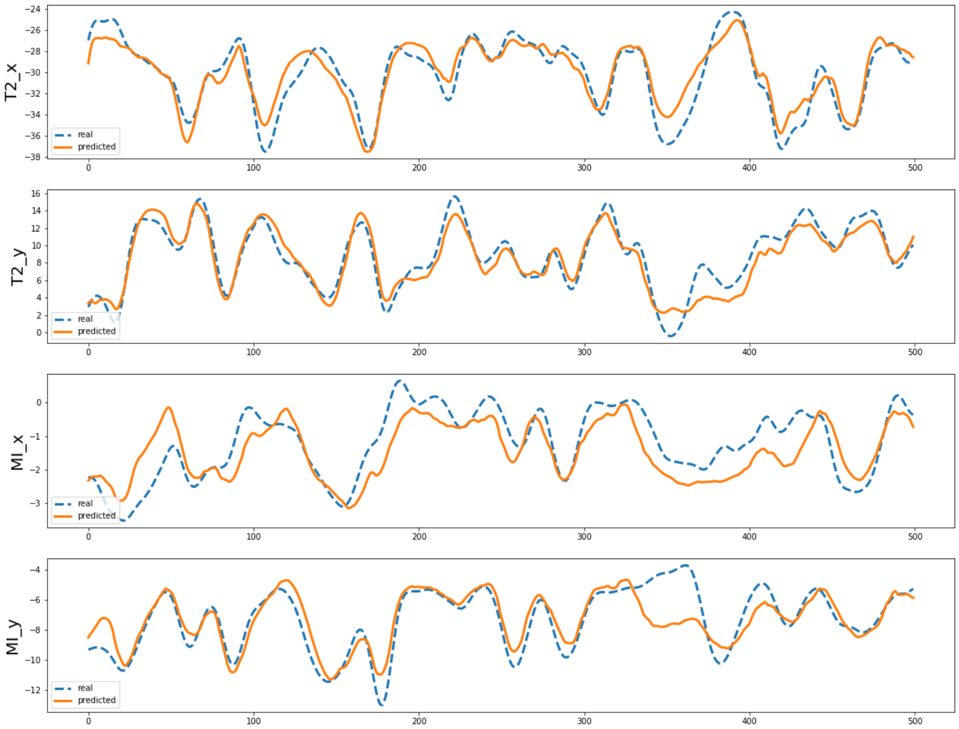


FIGURE 4.16: The predictions of T2_x, T2_y, MI_x, MI_y articulatory movements obtained from the RNN inversion model.

(from Fig.4.14a) in terms of Root Mean Squared Error (RMSE) and the Pearson correlation coefficient r computed using the true articulatory data. In Fig.4.16 plots of predicted trajectories for several articulatory features using the RNN inversion model are given. The predictions from the RNN inversion model are smooth enough even without any post-processing as mentioned in [93]. This indicates that RNN accounts for the previous and future information quite well. When inversion model predictions are used as articulatory features in the corresponding DNN-HMM acoustic models, clear error reduction is observed as shown in PER(%) column.

4.5.7 Joint Inversion

Settings

The architectures of inversion and acoustic RNN in the joint training experiment are the same as in baseline system. The difference is that the two models are trained jointly as illustrated in Fig.4.7.

In the first series of experiments, we initialized weights of the RNNs randomly. However, since the number of model parameters has doubled, finding a good initialization strategy is essential for the success of the training. Here, we use a pretraining strategy to help the network to start from a good position. In this case, the weights of the network are initialized with the weights from well trained inversion model and acoustic model of the baseline system. This initialization reduced 2 to 3 times the number of iterations necessary to train the models.

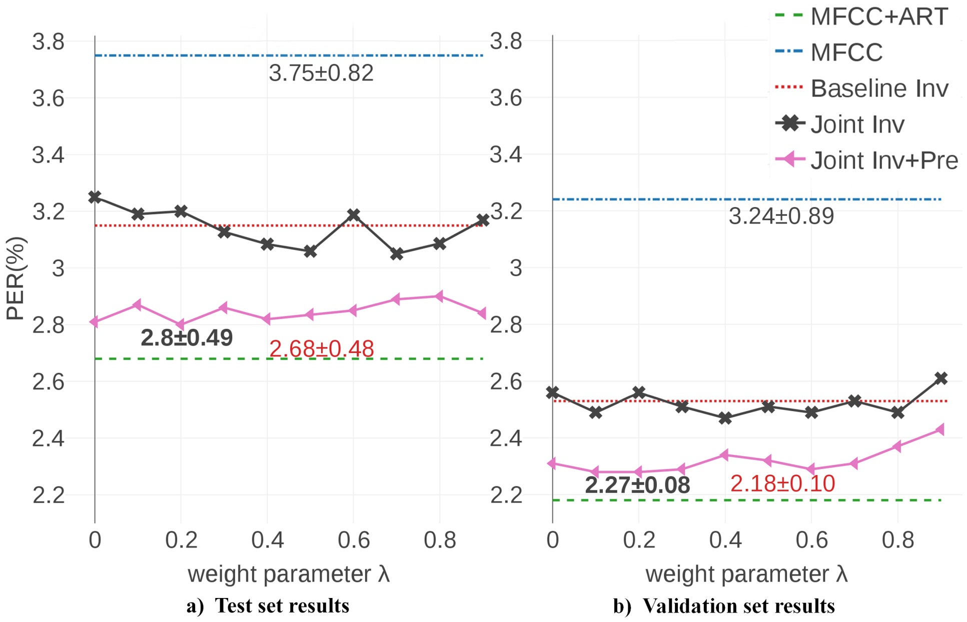


FIGURE 4.17: Results of the joint inversion and acoustic model training. $\lambda = 0$ corresponds to the case when the acoustic model uses MFCC features only but is trained with targets obtained from the MFCC+ART GMM-HMM.

During training, the joint loss function parameter λ was varied from 0 to 0.9 in steps of 0.1. As we explained above, $\lambda = 1.0$ is meaningless with respect to the training goal.

Results

The joint inversion + acoustic model results are summarized in Fig.4.17, where Fig.4.17-a shows the results for the test set and Fig.4.17-b gives the results for the validation set, the blue dashed line and green dashed line represent the upper and lower bounds respectively and the red dashed line is the inversion baseline result. The “Joint Inv” curve shows the results of jointly trained model with different λ. The “Joint Inv+Pre” denotes the results with pretraining, i.e when networks are initialized with the weights from the separately trained inversion and acoustic DNNs as explained in Section V.C-2.

When the RNN networks are randomly initialized, the joint training gives slight improvement for several values of λ. However, the effect of the pretraining is obvious.

From the figure we can see that the test set and validation set both achieve best results with $\lambda = 0.2$. Because the validation data was used to tune the DNN parameters and to train the GMM-HMM systems, the results are better than the ones on test set. The best joint training result with $\lambda = 0.2$ is **2.80±0.49%**, which is very close to the lower bound of 2.68%.

4.5.8 Generalized Distillation

Settings

In a similar way to our previous work [84], we applied the Generalized Distillation framework, but this time for RNN training. The teacher model in this case is the same as the acoustic RNN used in the articulatory inversion baseline system. However, here it is used only to obtain the soft targets for the student RNN model training which has the same architecture, except for the input layer. It takes only acoustic feature vectors (39 dim).

The two hyper-parameters of the distillation training, the temperature T and the imitation parameter λ were changed as follows. T was set to 1, 2, and 5, and λ was varied from 0 to 1 with steps of 0.2. Note that $\lambda = 0$ reduces the distillation training to conventional training, with the only difference that the hard targets are obtained from the GMM-HMM model trained with both the acoustic and articulatory features. On the other hand, $\lambda = 1$ means that the training is done using only the soft targets.

Results

The RNN distillation training results in terms of PER are summarized in Fig.4.18, Fig.4.18-a and Fig.4.18-b show the results on test and validation sets respectively. The blue dashed line represents the result of the student when trained alone which corresponds to the upper PER bound. The teacher's result is the lower bound distilled student can achieve. As can be seen, for $T = 2$ and $\lambda = 0.8$, both sets achieve the best results. The distillation result of **2.93±0.52%** PER is 21.9% better than the result of the student alone. When $\lambda = 0$, the distillation training is reduced to standard training with MFCC features using hard targets provided by the GMM-HMM trained on MFCC+ART vectors, which is already much better than training without any articulatory information. On the other hand, $\lambda = 1$ means the model is trained using the soft targets only and is even better than training with hard targets from the teacher. This suggests that soft targets provide a more informative objective than the hard targets alone.

Finally, we compare the best performances from all the different methods in Fig.4.19. Here, the most left and most right bars show the upper and lower performance bounds respectively. The best articulatory information fusion result is **2.80±0.49%** obtained from the joint inversion and acoustic model pretraining method. Distillation training result is little bit worse, but network size in this case is two times smaller and consequently faster to train and operate.

4.5.9 Discussion

The XRMB data were collected by asking all the speakers perform the same tasks. This makes the lexical content of the speech data the same for all the speakers. While the focus of this study is the acoustic and articulatory information fusion, we cannot ignore the fact that lexically the training and test data are the same. With respect to the phoneme language model, this would mean that it is a closed set LM and may have a boosting effect on all the results. In addition, the RNNs input consists of full utterances, so they may learn not only the acoustic dependencies, but the linguistic ones as well, which in turn can lead to biased performance. To check this hypothesis we performed a series of additional experiments.

The closed set language model effect

To explore the effect of the LM on our results, we did tests without LM as well as with a LM trained on the TIMIT database transcriptions which

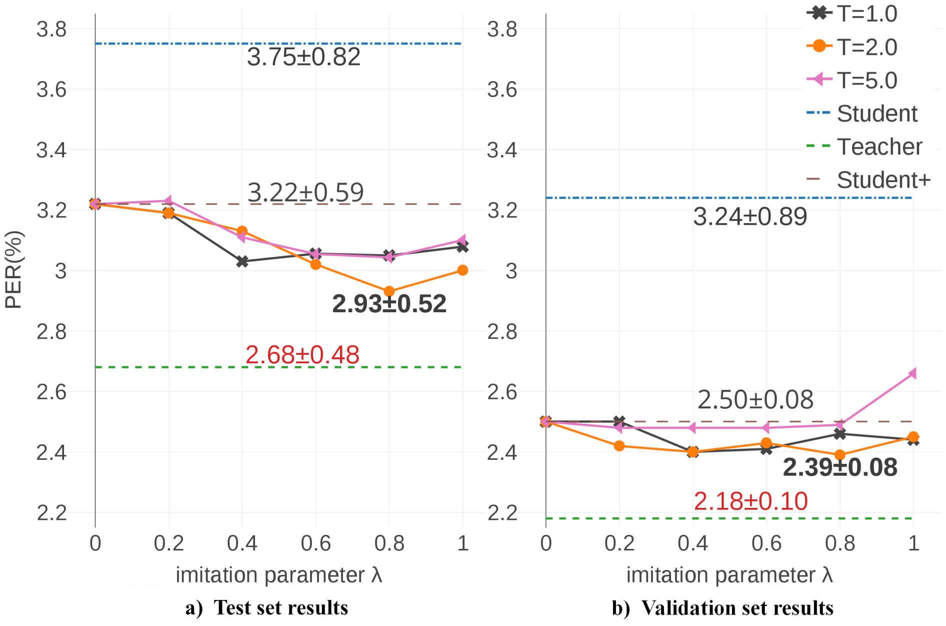


FIGURE 4.18: Results of distillation training. The lower and upper bound for the PER are shown as teacher and student only results. $\lambda = 0$ corresponds to the case when the student is trained using hard targets only. "Student+" corresponds to the case when the acoustic model uses MFCC features only but is trained with targets obtained from the MFCC+ART GMM-HMM.

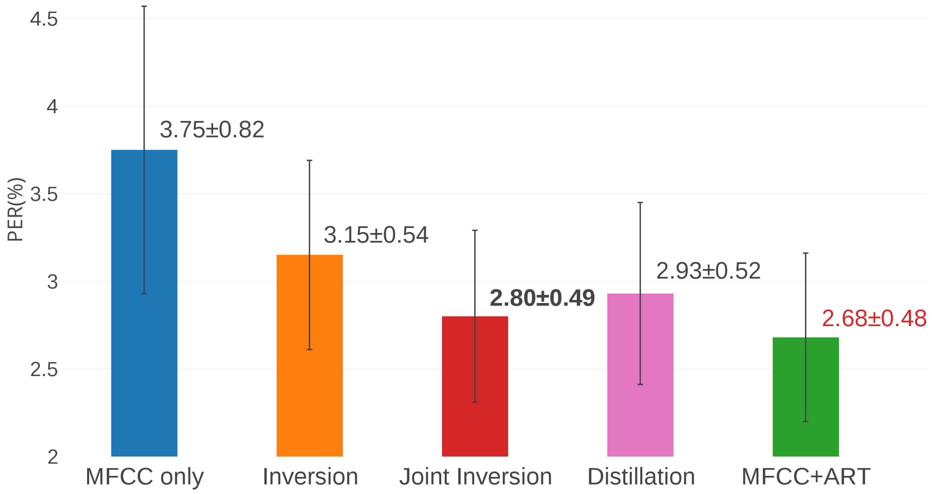


FIGURE 4.19: Performance of different methods and two acoustic baseline models. The "MFCC only" is the PER upper bound result. The "MFCC+ART" is the lower bound.

can be considered as a "general purpose" LM for this task. The results of the upper and lower PER bounds for the GMM, FNN and RNN acoustic models are summarized in Table 4.6. In these experiments, we excluded 5 utterances (per speaker) with the same lexical content across the speakers, so the results are slightly different from those from Fig 4.15.

Table 4.6 shows that the closed set XRMB LM has big effect on the GMM

TABLE 4.6: 7-fold CV results on different data sets with different language models. Results are shown for MFCC / MFCC+ART features.

Language Model	Train	Test
GMM acoustic model		
NO LM	19.3 / 7.44	30.9 / 11.1
TIMIT LM	17.1 / 7.67	25.1 / 10.3
XRMB LM	13.3 / 6.35	19.5 / 8.92
FNN acoustic model		
NO LM	2.90 / 2.16	9.82 / 5.12
TIMIT LM	2.75 / 2.04	9.54 / 4.82
XRMB LM	2.67 / 1.88	9.06 / 4.50
RNN acoustic model		
NO LM	1.46 / 1.17	4.23 / 2.99
TIMIT LM	1.38 / 1.11	4.02 / 2.85
XRMB LM	1.33 / 0.97	3.87 / 2.66

TABLE 4.7: Details of the train, validation, and test sets when utterances with the same lexical content are removed. The number in ( ) is the percentage of the amount from Table 4.2.

	Train	**Test**	**Validation**	**Total**
Sentences	1394 (53)	66 (13)	155 (53)	1515 (44)
Words	14532 (59)	691 (17)	1615 (64)	16838 (54)
Phonemes	41393 (48)	1945 (15)	4599 (56)	47937 (45)
Hours	1:05:12 (49)	0:02:59 (13)	0:07:15 (51)	1:15:26 (45)

model performance, but less effect on the DNN acoustic models. Furthermore, even without LM their performance is quite good. This suggests that DNN may have learned some lexical information as well.

The lexical content effect on DNN training

Although the utterances in the test set are from different speakers, they contain words and word sequences seen in the training set. For the neural network based acoustic models this could be significant since the input context in NNs is much larger (the whole utterance in RNNs) and the long span dependencies learned during training to some extend would match those in the test data.

To check this assumption, we repeated all joint inversion and distillation training experiments using an updated setting. This time we split the data into seven folds in terms of both speaker and sentence ids, so we got a two dimensional split with 49 sets as shown in Fig.4.20 and we used the 7 sets on the diagonal that are unique in both speakers and sentences for testing. All sets from the rows and columns other than those of the test set are used for training. This reduced the amount of data by more than half. Table 4.7 summarizes the details about the new dataset. We used the same DNN hyperparameters and XRMB language model.

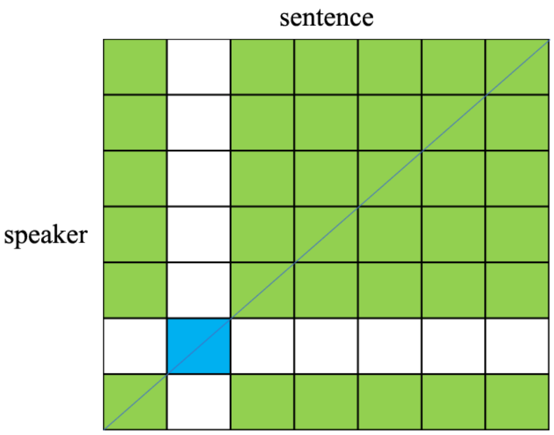
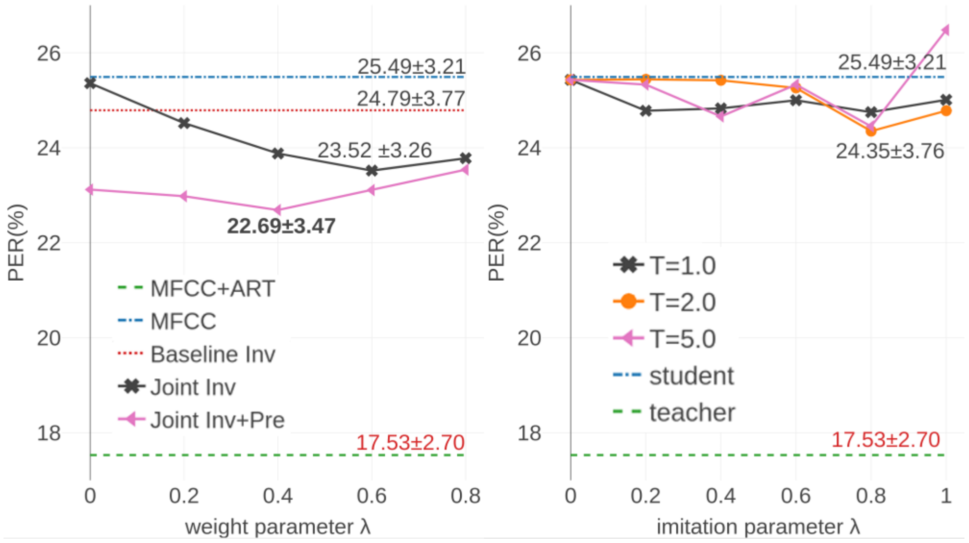


FIGURE 4.20: Train (green) and test (blue) datasets split for the second fold. Similarly, for other folds diagonal boxes data are used for testing.



a) Inversion results **b) Distillation results**

FIGURE 4.21: 7-fold CV results when utterances with the same lexical content are removed. a) PER of the inversion methods, b) PER of the distillation training.

The RNN-HMM inversion and distillation results are summarized in Fig.4.21 and results for all usable systems are summarized in Fig.4.22. As can be seen, both proposed methods work in this case as well. The absolute values of the PERs, however, are about ten times higher. Since the presence of the same lexical material in both train and test data and both the acoustic and inversion models are better suited for such test data, the results show less improvement using the proposed methods when utterances with the same lexical content are removed. Nevertheless, the same performance pattern can be observed in this case: the pretrained joint inversion is the best; the joint inversion is better than the distillation training, which in turn is better than the standard inversion. The optimized hyperparameters

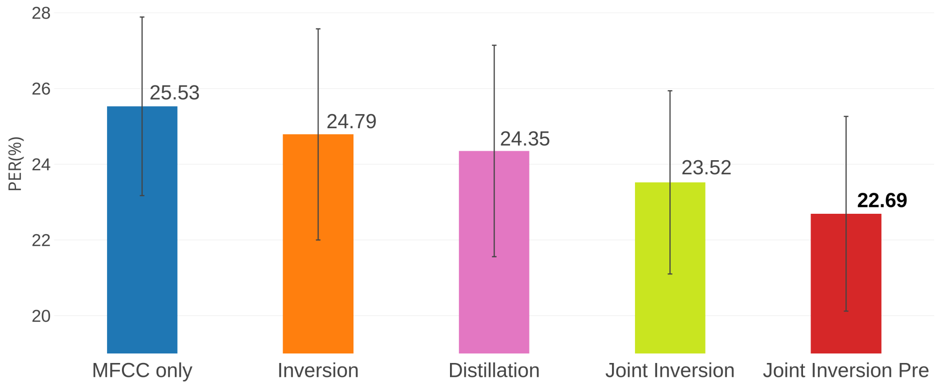


FIGURE 4.22: The results with 95% confidence intervals for all systems that can be used in practice after removing lexical content effect.

for the distillation training still are $T = 2.0$ and $\lambda = 0.8$, while the optimized hyperparameter for the joint inversion is $\lambda = 0.4$. Although $\lambda = 0$ means no articulatory information is integrated in the joint inversion, it is introduced by the pretraining, which is the reason that the pretrained joint inversion in this case is still better than MFCC only.

It is difficult to directly compare our results with results from other studies because the experimental conditions vary significantly. The closest experimental settings are the ones reported in [94] [95] where DCCA method showed significant improvements.

4.6 Summary

In this work, we proposed two methods to integrate articulatory information into ASR systems. One method utilizes the Generalized Distillation framework to build a biGRU-RNN based acoustic model which is trained with the guidance of the soft targets from a teacher biGRU-RNN learned from "rich" data which include articulatory features. The other method combines the inversion model and acoustic model into a single neural network which is trained jointly. When properly initialized, it achieves significant improvements.

The main findings of this study are:

1. Using deep RNNs as acoustic and inversion models provides big performance boost compared to the deep FNNs.
2. An RNN acoustic model trained using generalized distillation framework leads to up to 21.9% PER reduction having the same number of parameters as standard MFCC AM.
3. The PER is reduced by 25.3% using the joint inversion training strategy at the expense of increasing the size of the neural network.
4. The long term dependency learning capabilities of the RNNs are powerful enough to learn not only the temporal acoustic but also lexical information. As our experiments showed, this however may lead to

biased results when the data set is rather small and the train and test data are lexically similar.

In the future work, we are going to investigate how much reduction in the network size is possible by the joint inversion training. We also plan to experiment with bigger databases which don't provide articulatory measurements and try to integrate the available articulatory data based on our joint training approach.

Chapter 5

Multimodal Personality Recognition

Social networks such as Facebook, Twitter, and Weibo have become essential components of everyday life and hold rich sources that reflect individual's personality. Our personality affects our life choices, well-being, and many other behaviors. During the social interaction, people have to interact with unknown individuals. In order to achieve effective cooperation, it is important to predict the preferences and behaviors of the people we deal with. Such predictions can be found everywhere in the daily life and are often based on the personality of that person. For example, interviewers also consider whether the interviewee's personality is suitable for their company. A girl may consider marriage based on her boyfriend's personality. It is very useful for the systems to take into account of all the modalities and to fill missing ones given only those that are observed. In this chapter, we will describe Automatic personality recognition (APR) and Apparent Automatic Personality Recognition (AAPR) systems based on multimodal learning. The modalities we are going to fuse for this task are illustrated in Fig.5.1.

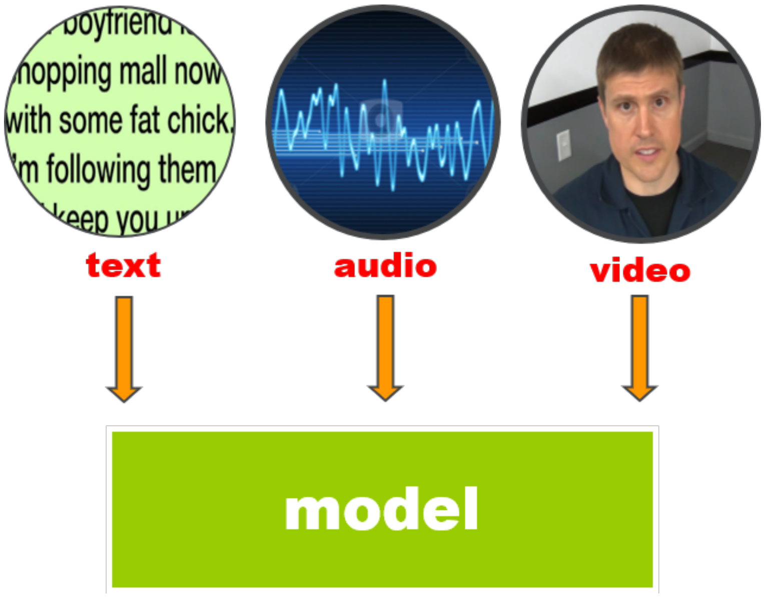


FIGURE 5.1: *The diagram of a system using text + audio + video modalities (or any combinations of three). Unlike the articulatory + acoustic fusion, this task can use any modalities in both training and testing time.*

The rest of this chapter is organized as follows. Section 5.1 describes the motivation and applications of APR and AAPR as well as how the personality is represented and evaluated in the systems. Section 5.2 talks about the related studies of two problems. Section 5.3 introduces the multimodal learning based on deep neural networks. Section 5.4 reports the experiments and results of this research. Finally, section 5.5 will be a summary of this chapter.

5.1 Background

Automatic personality recognition (APR) from his/her social network activities allows to make predictions about preferences across contexts and environments [96] and has many important practical applications, such as products, jobs, or services recommendation [9] [10], word polarity disambiguation, mental health diagnosis, etc. For example, Fei-Fei Li 's team has analyzed millions of publicly available images on Google Street View and use this knowledge to determine the political leanings of a given neighborhood just by looking at the cars on the streets.

However, the complexity of the personality formation makes it hard for automatic recognition [97]. One way to handle this issue is to predict the first impression (Apparent Automatic Personality Recognition (AAPR)) instead. The first impression also plays a very important role during social interaction like during an interview and is based on a lot of information such as physical appearance, voices, body language, facial expression, and the surrounding environment, which comes from different modalities.

APR and AAPR usually use the same model for personality representation, which is called Big Five Model.

Big Five Model

Big Five Model is formally described by five dimensions known as the Big-Five personality traits [24], whose examples can be seen in Fig.5.2:

- **EXT**raversion vs. Introversion (sociable, assertive, playful vs. aloof, reserved, shy).
- **NEU**roticism vs. Emotional stability (calm, unemotional vs. insecure, anxious).
- **AGR**eeableness vs. Disagreeable (friendly, cooperative vs. antagonistic, faultfinding).
- **CON**scientiousness vs. Unconscientious (self-disciplined, organised vs. inefficient, care-less).
- **OPE**ness to experience (intellectual, insightful vs. shallow, unimaginative).

Automatic recognition of personality typically involves binary classifications or regressions of which trait types an user belongs to given the content generated by him/her. The goal of APR is to predict the true personality traits of a person and the true labels are usually obtained by self-assessment questionnaire [98]. The goal of AAPR is to predict how the other people's impression on a person and the labels are usually tagged by averaging many reviewers' impressions.

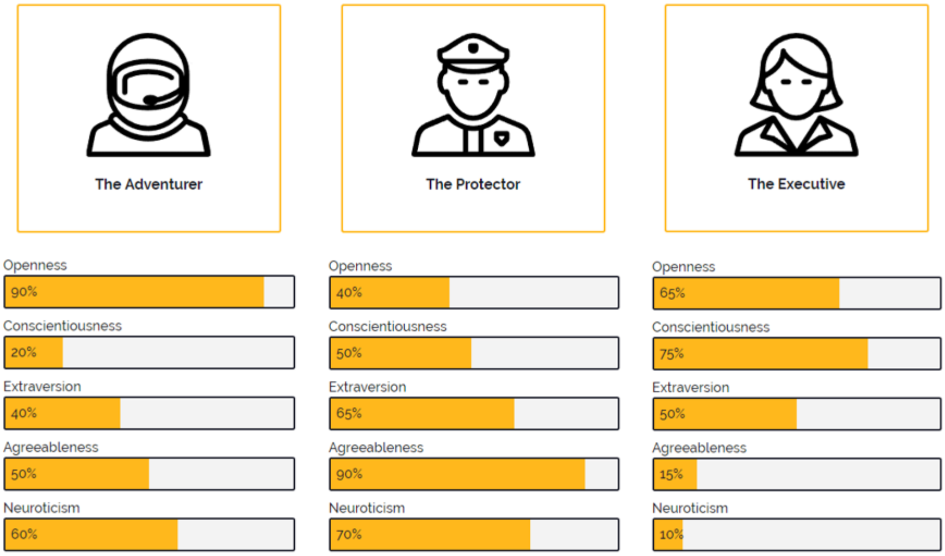


FIGURE 5.2: How the personality traits vary between different people.

5.2 Related Studies

Automatic personality recognition

A variety of approaches have been proposed for this task utilizing different classifiers and feature spaces. Until recently, most of the models were based on shallow learning approaches such as Support Vector Machine (SVM) [99] [100], Naive Bayes classifier (NB) [101], K-Nearest Neighbors (kNN) [102], and Logistic Regression (LR) [103]. In the early studies, text features were typically extracted by tools like Linguistic inquiry and word count (LIWC) [104] and good results were usually achieved by selecting features from a very large feature space like [105], which achieved a very high classification performance on the myPersonality task using ranking algorithms for feature selection and SVMs and Boosting as learning algorithms.

However, the performance of these approaches depends heavily on the data representation which often is based on hard-coded prior knowledge. Recently, deep learning approaches have obtained very high performance across many different natural language processing (NLP) tasks. Unlike traditional methods, deep learning approaches can learn suitable representation automatically. Deep learning based method was also proposed in [106] recently, where convolutional neural networks are applied to extract n-gram information from stream-of-consciousness essays [107].

Apparent Automatic Personality Recognition

As mentioned above, automatic personality recognition is difficult, since the subject himself may not know his own personality traits. Therefore, researchers tend to focus on the less difficult apparent automatic personality recognition. For example, in 2016, ChaLearn Looking at People First Impression Challenge [108] released a dataset of HD Youtube videos with annotations of Big Five impression personality traits. All three winners [109] [110] [111] of this challenge used deep learning based systems, where audio and video modalities were used. In 2017, the organizers also released a new first impression dataset [112] with an additional interview variable label to help the analysis of job candidate screening.

5.3 APR from Text

Deep neural networks can be used for a wide range of problems, this section will introduce the ways that we used for personality recognition problem.

Text carries potential information about the author, hence, when the observations involve text information, word embedding will be adopted.

5.3.1 Word embedding

Currently, word embedding has become a standard component for the DNN based natural language processing. It converts the one-hot representation of the word to a distributed representation [57], which has many benefits and allows to map words with similar meaning to similar values: the learning of one word can indirectly help the learning of the other words with similar meaning. This is especially helpful for tasks with small training data.

In order to utilize the statistical knowledge of the text, we pre-train the word embedding matrix with the text data using the skip-gram method [113]. We didn't use Google pre-trained word2vec because the statuses contain internet-slang, emoticons (e.g., :-D), acronyms (e.g., BRB-be right back) and various shorthand notations, which carry rich information about personality, but are not included in the Google model.

5.3.2 Network Architecture

In our neural networks, the first several layers are intended to automatically extract features from the raw text. Then, extracted features can be concatenated with other features and fed to the output layer for final classification. These are the main components of our networks:

- **Inputs:** our network takes input pairs of the form $\{(x_i, a_i)\}_{i=1}^{n}$, where n is the number of statuses. The text input $x_i \in R^{W \times V}$ contains W one-hot vectors of words with V vocabulary size. The $a_i \in R^{A}$ is nonverbal information data of A dimensions.

- **Target:** learning targets $\{t_i\}_{i=1}^{n}$ are binary classification labels for one trait (there are five traits in total) and $t_i \in R^2$ is represented as 2 dimensional one-hot vector.

- **Embedding layer:** the text input x_i is be transformed by the embedding matrix W_E to a distributed representation $e_i = W_E \cdot x_i$, where $W_E \in R^{V \times E}$ and E is the dimension of word embedding.

- **Convolution layer:** multiple convolutional filters extract n-gram information from e_i as shown in Fig.5.3. The green box represents a bigram CNN filter which considers two words each time and slides over all words to create a feature map.

  A convolutional filter of size $n \times E$ is applied on status $e_i \in R^{W \times E}$ to extract the n-gram features, where n =1, 2, 3 for the unigram, bigram, and trigram. In every convolutional layer, K filters are applied

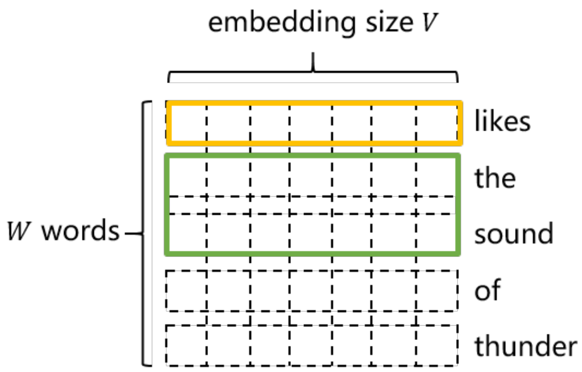


FIGURE 5.3: Illustration of unigram and bigram convolutional filters.

to each status e_i producing a matrix $F_n^{conv} \in R^{K \times n \times E}$ to which bias $B_n^{conv} \in R^K$ is added, resulting in $FM_n^{conv} \in R^{K \times (W-n+1) \times 1}$. A Rectified Linear Unit (ReLU) function is then applied on FM_n^{conv} to introduce non-linearity.

- **Avg / max pooling layer:** it performs a max or average pooling operation on the convolutional layer output FM^{conv} to obtain a feature vector $PFM_n \in R^K$. All n-grams CNN feature maps will be concatenated into $PFM_{all} \in R^{(K \times n)}$. It seems that max pooling extracts the most important information from the status but doesn't take into account how many times such information appears in a status while average pooling does the opposite. The diagram of combination of embedding layer and convolutional layer is illustrated in Fig.5.4.

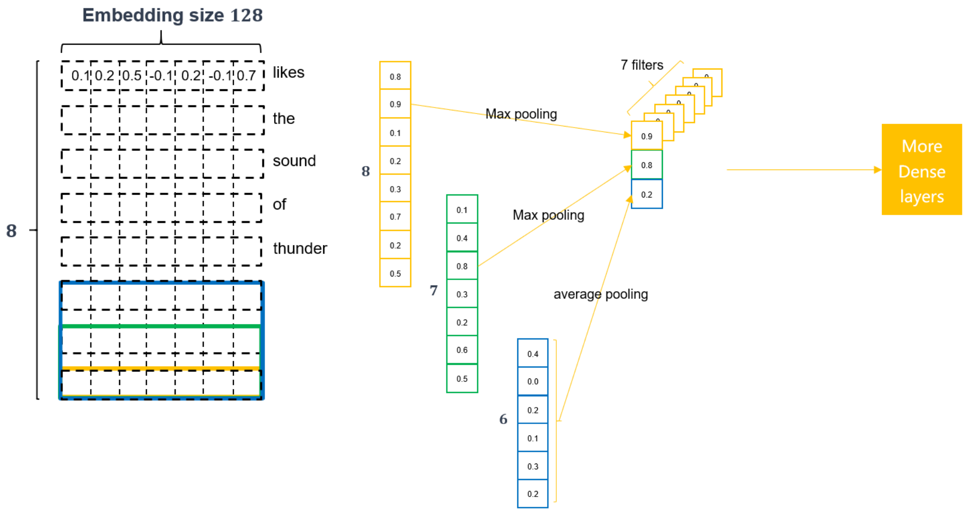


FIGURE 5.4: Word embedding + convolutional layer for APR.

- **Non-lexical features:** recognition of personality which depends only on text features could be misleading since words meanings vary in different contexts. Therefore, it is beneficial to use other non-lexical features, if such are available. This can be done by concatenating the text features $PFM_{all} \in R^{(K \times n)}$ extracted using convolutional nets with the non-lexical features $a_i \in R^A$, resulting in $concat(PFM_{all}, a_i) \in R^{(K\times n+A)}$.

- **Fully connected layer:** to transform the combined features $concat(PFM_{all}, a_i)$ to higher-level representation which may be shared across different samples, we added several fully-connected layers. The activation function of these layers is also ReLU. The output of two fully-connected layers is computed as:

$$x^{fc_1} = \sigma(W_{fc_1} \cdot concat(PFM_{all}, a_i) + b_{fc_1}) \tag{5.1}$$

$$x^{fc_2} = \sigma(W_{fc_2} \cdot x^{fc_1} + b_{fc_2}) \tag{5.2}$$

where $W^{fc_1} \in R^{(K\times n+A)\times F}$, $W^{fc_2} \in R^{F\times F}$, and σ is $max(x, 0)$.

- **Output layer:** a softmax output layer is added to maximize the probabilities of the trait being yes and no. It's output is obtained from:

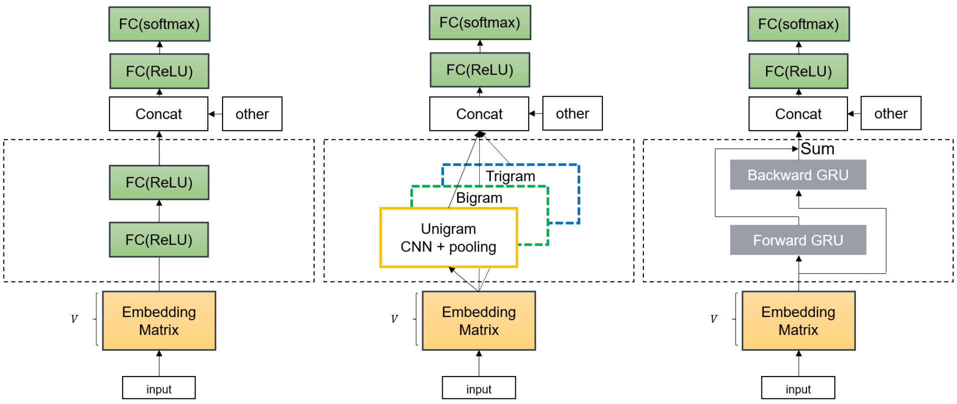
$$y_i = \sigma(W_o \cdot x^{fc_2} + b_o) \tag{5.3}$$

where $W_o \in R^{F\times 2}$, and σ is the softmax function.

- **Loss function:** we use cross entropy (5.4) as the loss function to minimize:

$$H(t_i, y_i) = -\sum_{j=1}^{2} t_{ij} \log y_{ij} \tag{5.4}$$

We can also replace the convolutional layers with other type of layers, illustrated as Fig.5.5.



A. **Fully connected neural network** *B.* **Convolutional neural network** *C.* **Bidirectional recurrent neural network**

FIGURE 5.5: Network architecture for APR from text

5.4 APR from speaking style

The personality traits can be inferred based on many types of observations, such as text [114, 115, 116], audio [117, 118], video [119, 109], or any combination of them, each of which has its own applications, depending on the availability of observations in different situations. For example, the audio based AAPR is very useful for the producers who make education or explainer videos since the audiences' first impression on their voices can largely affect the trustiness and attractiveness of the videos.

The conventional methods of AAPR from audio typically use a large pool of potentially prosody features (e.g. Mel Frequency Cepstral Coefficients, pitch, energy, and their 1st/2nd order temporal derivatives) and "Interspeech 2012 Speaker Trait Challenge" [120] is the first, rigorous comparison of different approaches over the same data and using the same experimental protocol for audio based AAPR, where the performances of most approaches depend heavily on careful feature selection [121, 122, 123, 124]. Many of those features are included in the open-source openSMILE tool [125] and can serve as baseline for audio based AAPR. For example, the winner in the ChaLearn 2017 Job Candidate Screening Competition also used the openSMILE feature configuration that served as challenge baseline in the INTERSPEECH 2013 Computational Paralinguistics Challenge, which is 6373-dimensional feature set and was found to be the most effective acoustic feature set among others for personality trait recognition [126]. In order to learn useful features automatically, deep learning based methods have also been proposed for audio based AAPR. The audio model baseline provided by the organizer is a variant of the original ResNet18 model [112], which was trained on random 3s crops of the audio data and tested on the entire audio data. However, since the general network architecture is not specifically designed for AAPR, it doesn't appear to clearly outperform the conventional methods.

5.4.1 Neural Style Transfer

The neural style transfer became popular after the paper [127], where the style representation of an image is described as the correlation between different filter responses given by the Gram matrix. The basic idea was developed to classify image style in work [128], where the VGG-19 network [129] trained on the ImageNet dataset was used to obtain filter responses at different layers whose Gram matrix is calculated and transformed into a style vector, which is then classified by an SVM (support vector machine) classifier.

But the characteristics of audio signals are different from those of the images, e.g. speech is a sequential signal while the image is a 3D-tensor, and the duration varies for different utterances. Moreover, the Gram matrix representing styles is usually calculated from pre-trained networks and might not hold the best features for the desired task. In this work, we propose a system that automatically captures speaking styles for apparent personality recognition.

5.4.2 Automatic Speaking Style Extraction

The proposed system evaluates a speech signal and returns 6 scores for the 5 personality traits and an interview variable (whether a candidate will be invited for a job interview).

In our neural network, the Gram matrix is not calculated from any pretrained networks. Everything is jointly learned from scratch. The overall architecture is illustrated in Fig.5.6.

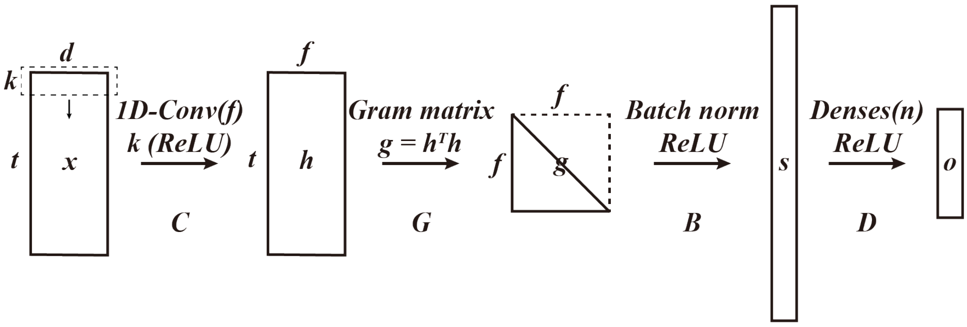


FIGURE 5.6: Neural Network architecture used in our system.

- **Input:** the input $x \in R^{t \times d}$ to our network contains d-dimensional speech features obtained at t timesteps.

- **Target:** the learning target $t \in R^6$ is a 6-dimensional vector (representing five traits and the interview variable), whose range is [0,1].

- **Convolutional layer:** the input x is first fed to a convolutional layer with f number of filters, $k \times d$ kernel size, 1 stride, and "same" zero padding, resulting in a feature map $h \in R^{t \times f}$. This is intended to automatically filter out the silence and extract useful features for computing the speaking styles. A Rectified Linear Unit (ReLU) activation function is then applied to introduce non-linearity.

- **Gram layer:** Gram matrix g is then calculated from the feature map h, where $g = h^T h$. The lower (or upper) triangular matrix and diagonal are flattened into a vector $g* \in R^{(f+1)*f/2}$ for the next layer. A Gram layer actually represents the speaking styles as the correlations between different channels of the feature maps from the previous convolutional layer.

- **Batch norm layer:** since the norms of values in $g*$ are very big, a batch normalization layer with a ReLU activation function is added to solve this issue, resulting in a vector s that represents the speaking styles.

- **Fully connected layers:** the style vector s is then fed to one or more fully connected layers (dense layers) with ReLU activation function that further transforms s to higher level features.

- **Output layer:** finally, an output layer without activation function follows the dense layer(s) to produce an output o with 6 dimensions.

- **Loss function:** We tackle this task as a regression problem, so the mean squared error (MSE) is used as loss function.

5.4.3 Low Level Feature Extraction

16kHz audio signals are extracted from the video clips and 13 dimensional Mel frequency cepstral coefficients (MFCCs) are computed every 10ms over a 25ms window, along with their first and second derivatives and used for our acoustic feature vector $x \in R^{1528 \times 39}$, where 1528 is the number of timesteps.

5.4.4 Overall Settings

In all the networks to be trained, every hidden dense layer has 512 nodes and is followed by a dropout layer with a drop rate of 40%. The kernel size of every convolutional layer is 3. Each network was trained by 300 epochs using Adam [130] update method with a learning rate of 1e-4 and a batch size of 16. We chose 300 epochs because the networks after 300 epochs perform fairly well on the validation set. The L2 regularization with a rate of 1e-4 is also added to the final loss, which is $10^{-4}\sum(\|\theta\|^2)/2$ and θ is the weights vector of a layer.

5.5 AAPR from Text, Audio, Video

This task takes a short video clip of a single person with three modalities (text, audio, video) and returns 6 scores for the 5 personality traits and the interview variable.

The typical fusion method is to process different modalities separately, then combine them together. But the processed features may not be complementary to each other. So we can train all modalities in a single network, but the convergence speeds of different modalities are also different, it doesn't work well either. To address this problem, we propose an attention network with multiple training stages.

The purpose of our method is to divide the multimodal learning task into two sub-tasks: 1. An attention network which learns how to change the scales of different modalities depending on their correlations. 2. A network that takes the outputs of the attention network to predict the targets.

5.5.1 Preprocessing

From each video clip, we extract low-level features for each text, audio, and video modality and form matrices with a shape of (timesteps, channels), resulting in a dataset $\{(T_i, A_i, V_i, l_i)\}_{i=1}^{N}$, where $T_i \in R^{s_t \times c_t}$,$A_i \in R^{s_a \times c_a}$, $V_i \in R^{s_v \times c_v}$ stand for text, audio, and video respectively, s_t, s_a, s_v are the number of timesteps and can vary among different clips, $l_i \in R^6$ is the labels vector and N is the number of sample clips.

5.5.2 Single Modality

Three 1D-convolutional networks are trained individually for each modality as illustrated in Fig.5.7. They all have the same architecture: one or more convolutional layer(s) followed by a global average pooling layer (kernel size is the maximum length). Each network is trained by minimizing the mean squared error (MSE) between the predictions and the corresponding labels. Once the network is trained, we fix its parameters and take the output of the global average pooling layer as our final representation for each modality. The reason we use neural networks for feature extraction is that we don't know which features are good for predicting personality traits and we let neural networks to automatically extract them. It is possible to use different architectures to extract better features, but here we focus on the joint multimodal learning using correlational networks.

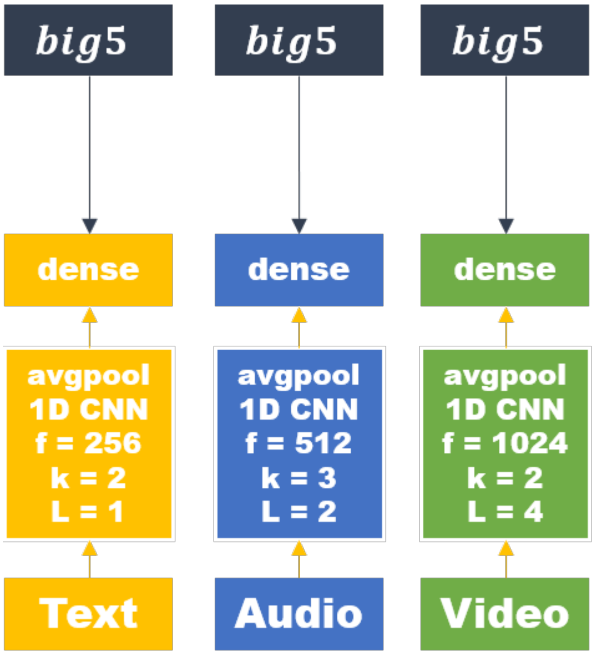


FIGURE 5.7: Single modality architectures used in our system.

After the feature extractions, 2D-array data is reduced into 1D-array and our dataset becomes $\{(t_i, a_i, v_i, l_i)\}_{i=1}^{N}$, where $t \in R^{d_t}$, $a \in R^{d_a}$, $v \in R^{d_v}$ with dimensions determined by the number of filters in the last convolutional layer of the corresponding CNN.

In order to utilize the statistical knowledge of the text, we pre-train the word embedding matrix with the text data using the skip-gram method [113]. We didn't use Google pre-trained word2vec because the statuses contain internet-slang, emoticons (e.g., :-D), acronyms (e.g., BRB-be right back) and various shorthand notations, which carry rich information about personality, but are not included in the Google model.

5.5.3 Multimodal Network Architecture

After computed t_i, a_i, v_i from their own networks mentioned above, we fuse t_i, a_i, v_i using the proposed architecture shown in Fig.5.8.

- **Higher feature extraction** To give the network some spaces to change different modalities' representations for the fusion, the architecture first maps all modalities into high-level spaces t_h, a_h, v_h respectively using formula Eq.5.5, where d() denotes a dense layer, θ is the parameters of the dense layer.

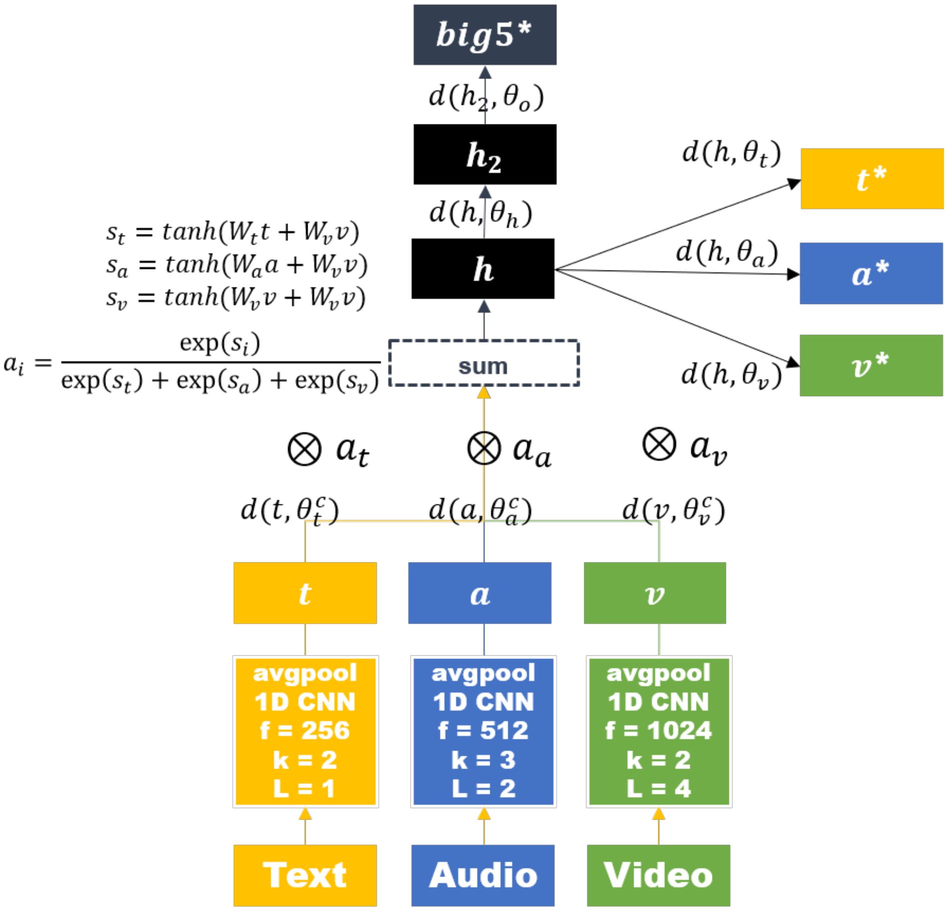

$big5^*$
$d(h_2, \theta_o)$
h_2
$d(h, \theta_h)$
h
$d(h, \theta_t)$
t^*
$d(h, \theta_a)$
a^*
$d(h, \theta_v)$
v^*
$s_t = tanh(W_t t + W_v v)$
$s_a = tanh(W_a a + W_v v)$
$s_v = tanh(W_v v + W_v v)$
$a_i = \frac{\exp(s_i)}{\exp(s_t) + \exp(s_a) + \exp(s_v)}$
sum
$\otimes\ a_t$
$\otimes\ a_a$
$\otimes\ a_v$
$d(t, \theta_t^c)$
$d(a, \theta_a^c)$
$d(v, \theta_v^c)$
t
a
v
avgpool 1D CNN f = 256 k = 2 L = 1
avgpool 1D CNN f = 512 k = 3 L = 2
avgpool 1D CNN f = 1024 k = 2 L = 4
Text
Audio
Video

FIGURE 5.8: Multistage Training Strategy with Attention .

$$\begin{aligned} t_h &= d(t, \theta_t^c) \\ a_h &= d(a, \theta_a^c) \\ v_h &= d(v, \theta_v^c) \end{aligned} \tag{5.5}$$

- **Attention weights** Then the network computes the attention weights a_t, a_a, a_v that change the corresponding modalities into suitable scales using formula Eq.5.6. The attention weight of each modality is based on the relationship between this modality and video modality because video modality holds the most information and we want to make other modalities complementary to video modality.

$$\begin{aligned} s_t &= tanh(W_t \cdot t + W_v \cdot v) \\ s_a &= tanh(W_a \cdot a + W_v \cdot v) \\ s_v &= tanh(W_v \cdot v + W_v \cdot v) \\ a_t &= \frac{e^{s_t}}{e^{s_t} + e^{s_a} + e^{s_v}} \\ a_a &= \frac{e^{s_a}}{e^{s_t} + e^{s_a} + e^{s_v}} \\ a_v &= \frac{e^{s_v}}{e^{s_t} + e^{s_a} + e^{s_v}} \end{aligned} \tag{5.6}$$

- **Common representation space** the modalities will be scaled by their corresponding attention weights and summed into a common representation space that holds all modalities information Eq.5.7.

$$\begin{aligned} t' &= t_h \otimes a_t \\ a' &= a_h \otimes a_a \\ v' &= v_h \otimes a_v \\ h &= t' + a' + v' \end{aligned} \tag{5.7}$$

- **Big5 predictions** our network predict the big5 scores $big5^*$ for the given inputs using formula Eq.5.8.

$$\begin{aligned} h2 &= d(h, \theta_h) \\ big5^* &= d(h2, \theta_o) \end{aligned} \tag{5.8}$$

- **Auto-encoder** our network also has 3 outputs for predicting each modality t^*, a^*, v^* using formula Eq.5.9.

$$\begin{aligned} t^* &= d(h, \theta_t) \\ a^* &= d(h, \theta_a) \\ v^* &= d(h, \theta_v) \end{aligned} \tag{5.9}$$

5.5.4 Multistage Training Strategy

Since we divide the task into two sub-tasks, we need different lost functions for each of them. The whole network is trained end to end, but with some parameters fixed in certain stage.

3 groups of loss functions

We used many loss functions, to make it clear, we summarize them into 3 groups.

- **Auto-encoder losses** as described in Eq.5.10, where model() denotes our network, concat() means concatenation and 0 stands for the tensor with the same shape of that modality.

$$\begin{aligned} &mse(model(t, 0, 0), concat(t, a, v)) \\ &mse(model(0, a, 0), concat(t, a, v)) \\ &mse(model(0, 0, v), concat(t, a, v)) \end{aligned} \tag{5.10}$$

- **Lack modality losses** as described in Eq.5.11.

$$\begin{aligned} &mse(model(t, 0, 0), big5) \\ &mse(model(0, a, 0), big5) \\ &mse(model(0, 0, v), big5) \end{aligned} \tag{5.11}$$

- **Big5 loss** as described in Eq.5.12.

$$mse(model(t, a, v), big5) \tag{5.12}$$

4 groups of parameters

We also summarize the parameters into 4 groups:

- **content parameters:** $\theta_t^c, \theta_a^c, \theta_v^c$**.**
- **Attention parameters:** W_t, W_a, W_v**.**
- **Auto-encoder parameters:** $\theta_t, \theta_a, \theta_v$**.**
- **big5 parameters:** θ_h, θ_o**.**

3 stages

Then we train the network for 3 times with different loss function and parameters.

- **Higher feature extraction stage**

  minimize: (auto-encoder losses)*0.8 + (lack modality losses + big5 losses)*0.2

  trainable: contents parameters, weights parameters, auto-encoder parameters

  non-trainable: big5 parameters

- **Attention weights learning**

  minimize: lack modality losses*0.8 + big5 losses*0.2

  trainable: weights parameters, auto-encoder parameters

  non-trainable: contents parameters, big5 parameters

- **Big5 prediction training**

  minimize: big5 losses

  trainable: big5 parameters

  non-trainable: contents parameters, weights parameters, auto-encoder parameters

5.6 Experiments

5.6.1 Database Description

myPersonality

Workshop on Computational Personality Recognition (Shared Task) The dataset used in our experiments is a subset (250 users with 9917 status updates) of the database released by organizers of the "Workshop on Computational Personality Recognition (Shared Task)" [23]. It contains Facebook statuses in raw text, author information (network size, betweenness, nbetweenness, density, brokerage, nbrokerage, and transitivity) and gold standard Big-5 personality labels (obtained using self-assessments questionnaire). The personality labels have both scores and classes. Classes have been derived from scores with a median split. In this work, we focus on personality classification.

It is suggested in the shared task guidelines to split the data as train (66%) and test (33%). Because each author has multiple statues in this dataset, the train set can see all examples from 250 authors if randomly splitting 9917 statuses into train/test sets. So we divided 250 authors into 3 parts, each of which contains about 3300 statuses. We used 3-fold cross validation (CV) for our performance evaluation.

During tokenization, we treat each internet-slang word as an unique word and map all web addresses to the same word "*URL". Such processing was also applied to digit numbers, time and currency. This way, we got about 15000 unique words and maximum word length of a single status was 78. We set the vocabulary size to 4400, which is the number of words appearing more than 2 times in the data. Uncommon words then are replaced by "UNK". We concatenated a zero-likes vector $z \in R^{1 \times V}$ to embedding matrix W_E which is not trainable as shown in Fig.5.5, and pad all statuses to the length of 78 with 4401 index.

We trained the embedding matrix W_E learning in advance from the raw text of this dataset using skip-gram. The skip window was set to 1 and embedding size to 128 dimensions. The embedded matrix is kept fixed during the network training.

First impressions V2

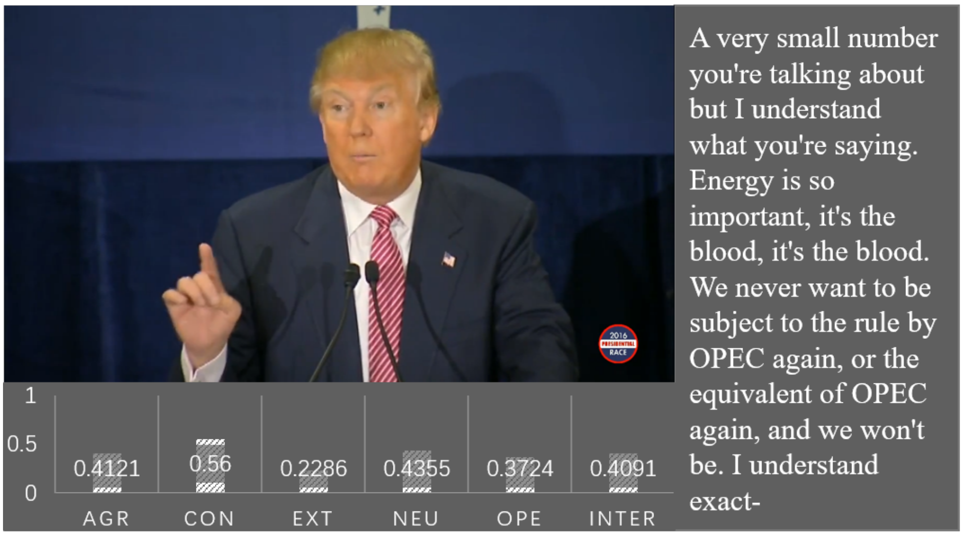
The first impressions data set (2017) [112] comprises of 10,000 clips (with an average duration of 15s) extracted from more than 3,000 different YouTube high-definition (HD) videos of people facing a camera and speaking in English. People in videos have different gender, age, nationality, and ethnicity. Each clip is labeled for the Big Five personality traits scores along with an interview variable score that recommends whether a job candidate should be invited for an interview or not. The range of scores is [0,1]. We use 5992 videos for training, tune networks using 2000 videos and test on 2000 videos. This dataset division is the same as used by the CVPR 2017 [112] workshop participants. Fig.5.9 shows an example of one image and transcription from a video clip.

For each of the five traits and the interview variable, the performance was evaluated by the Mean Absolute Error subtracted from 1, which is formulated as follows:

$$1 - \frac{\sum_{i=1}^{N} |target_i - predicted_i|}{N} \tag{5.13}$$

The score varies between 0 (worst case) and 1 (best case).

In all the networks to be trained, each hidden layer is followed by a dropout layer with drop rate of 50%. Each network was trained using Adam [130] update method with a learning rate of 1e-4. We choose the epoch that performs best on the validation set. The feature extraction network for text has



A very small number you're talking about but I understand what you're saying. Energy is so important, it's the blood, it's the blood. We never want to be subject to the rule by OPEC again, or the equivalent of OPEC again, and we won't be. I understand exact-

FIGURE 5.9: An example of one image and transcription from a video clip in test set along with the predictions (also range [0,1]) using our system. The scores are predictions indicating personality traits and interview variable, not the evaluation metric (1 - mean absolute error).

1 convolutional layer with kernel size 1 and 256 filters. The feature extraction network for audio has 2 convolutional layers with kernel size 3 and 512 filters (all layers use the same setting). While the feature extraction network for video has 4 convolutional layer with kernel size 2 and 1024 filters.

5.6.2 Experimental Results

myPersonality

In our experiments, each network was trained by 100 epochs using Adam [130] update method with a learning rate of 1e-4. Then, using their predictions on the test data we calculated the classification accuracy and F1 score metrics results for test set.

Each of our architectures can be seen as three consecutive parts: embedding part (embedding layer), feature extraction part and classification part (one fully connected layer). We experimented feature extraction part with Convolutional architecture (Fig.5.5-A), bidirectional recurrent architecture (Fig.5.5-B) and fully-connected architecture where the module in the dotted box is replaced one fully-connected layer. Convolutional architectures are also tested with max pooling and average pooling layers. The output of feature extraction part is also concatenated with 7 dimensional author information. The Overall DNN settings are shown in Table 5.1. All results are given as mean and standard deviation of 3-fold cross validation experiments.

- **Fully-connected architecture**

  Table 5.2 shows the accuracy and F1 score results when the module in the dotted box (Fig.5.5) is replaced one fully-connected layer.

  When using a fully-connected layer for feature extraction, it overfits heavily. The help of dropout is tiny. We also tested models that use

TABLE 5.1: *Overall DNN settings*

Length of status	78
Embedding size	128
loss function	Categorical Cross-entropy
Output activation	SoftMax
hidden activation	ReLU
Nodes of FC layer	32
# CNN filters per n-gram	32
GRU (per direction) nodes	32

only text or author information as input. The results using author information were tested on author level (250 examples).

The results shows that it gives better results to combine both text and author information as input. However, models with only text input didn't perform well. So the following results are all models with both text and author information.

We tried to initialize embedding matrix randomly and learn it during the training of classification task. We also used the embedding matrix pre-trained on Wikipedia using FastText [131]. But both were worse than the embedding pre-trained using test from this dataset. Perhaps, the expression people use for chatting is quite different from the one for writing.

- **Convolutional architecture**

  Table 5.3 shows the accuracy and F1 score results when using a unigram convolutional layer with average pooling. We tuned the L2 regularization and drop out rates to get better results for each trait.

  We found that no improvement was achieved when the combination of unigram, bigram, trigram CNN filters was used, but it did enhance the learning capacity. We also found that max pooling can learn faster than average pooling, but overfits heavily. Convolutional architecture with average pooling achieved the best results 60.0±6.5%, which is better than competition results of F1 58.6% from two teams [101] [102] on *myPersonality* shared task in 2013.

- **Recurrent architecture**

  With this architecture, we replaced the module in the dotted box by a bi-directional GRU layer[131] where forward GRU and backward GRU are summed up in the output of this layer. The RNN result is listed in Table 5.4.

  The RNN results showed that recurrent architecture is able to extract some sequential meanings from the text for personality recognition, but still not as good as convolutional layer with average pooling. It seems that the selection of words reflects more personality than the meaning itself.

TABLE 5.2: Classification test results with different features using fully-connected architecture.

Trait	ACC%	F1%
Author Information Only		
EXT	68.4±3.0	63.3±2.7
NEU	62.4±1.7	55.5±5.1
ARG	57.6±3.6	55.8±3.6
CON	52.0±4.3	49.8±4.2
OPN	70.4±4.2	42.5±2.5
Overall	61.7±7.4	52.5±7.7
Text Only		
EXT	51.8±1.6	49.9±1.6
NEU	53.0±1.7	49.3±1.3
ARG	49.7±1.4	49.4±1.3
CON	49.8±1.6	49.3±1.7
OPN	64.5±5.7	49.1±0.5
Overall	53.8±6.0	49.3±1.1
Author+Text		
EXT	59.5±0.2	58.0±1.3
NEU	60.7±4.8	55.6±3.5
ARG	53.8±2.8	53.0±2.0
CON	51.4±0.9	51.2±1.4
OPN	72.4±8.5	53.1±3.4
Overall	59.6±8.2	53.8±3.1

Automatic speaking style extraction

Here, we present the APR results based on speaking style. In order to verify whether the performance improvement is provided by the speaking styles captured by the Gram matrix, we also trained networks without it. We tried recurrent networks with GRU (Gated Recurrent Unit) cell [50] and found they are not as good as convolutional networks for this task. The networks with max pooling layer or more than one convolutional layers didn't show improvement either. We found the best network architecture without Gram layer is the network with one 1D-convolutional layer, one average pooling layer over all timesteps, one dense layer, and the output layer.

The experimental results of testset in terms of $1-MAE$ are summarized in Table 5.5. The column "System" denotes different DNN configurations. Thus, C(32) stands for a convolutional layer with 32 filters, P - an average pooling layer over all timesteps, B - a batch normalization layer with ReLU activation and D - a dense layer (2D means 2 consecutive dense layers).

Because it is hard to keep the numbers of parameters in the baseline and proposed architectures the same, we tried many hyper-parameter combinations and found that C(32)+P+2D was the best one among architectures without speaking styles. From the results, we can see that batch normalization layer didn't show any improvement in these cases and could not outperform the ResNet18. However, when the Gram layer along with a

TABLE 5.3: Classification results using convolutional architecture with average or max pooling

Big-5	Average pool		Max pool	
Trait	ACC%	F1%	ACC%	F1%
EXT	65.8±4.9	65.2±5.0	59.7±2.4	58.6±1.8
NEU	67.9±2.1	62.7±4.7	59.0±3.1	54.9±3.8
ARG	59.8±4.9	57.0±3.4	51.9±0.3	51.4±0.6
CON	53.7±1.1	53.3±1.1	52.8±1.0	52.6±1.2
OPN	74.0±12.2	61.3±11.1	64.9±5.4	54.0±4.6
Overall	64.2±8.7	60.0±6.5	55.4±5.4	54.1±3.4

TABLE 5.4: Classification results for recurrent of architectures

Trait	ACC%	F1%
EXT	60.9±0.3	59.8±0.9
NEU	61.1±4.5	55.5±3.1
ARG	54.3±3.7	53.7±3.0
CON	50.9±0.2	50.6±0.3
OPN	70.6±7.9	56.4±9.1
Overall	59.6±7.6	55.2±4.8

batch normalization layer is used, all configurations shows significant performance increase with the C(128)+G+B+2D achieving the best audio based AAPR results.

Table 5.6 shows the Big-Five traits and the interview score classification results. The ground truth labels and the system predictions were binarized based on the training set mean scores. If a given score is above the corresponding mean, the label or the prediction is considered positive, otherwise - negative. The accuracy results also show that our proposed architecture brings significant improvements for both the personality traits and interview variable.

We also noticed that the Gram layer cannot be jointly trained without a batch normalization layer (e.g. C(32)+G+D didn't converge). The reason might be that the values of the Gram matrix are changing dramatically for each batch when the Gram matrix is not calculated from the pre-trained (fixed) convolutional layer, but from a convolutional layer that is also being trained.

Text, audio, video Fusion

Finally, the APR results of all modalities will be presented. First, all three feature extraction networks are trained separately using only one modality as illustrated in Fig.5.7.

- **Single modality** the results are summarized in Fig.5.10. We can see that the video modality holds the most information.

TABLE 5.5: $1 - MAE$ results. OPE: openness to experience. CON: conscientiousness. EXT: extroversion. AGR: agreeableness. NEU: (non-)neuroticism. Inter: interview invite variable. Ave: the average score of 5 traits (interview variable is not included).

System	Ave	OPE	CON	EXT	AGR	NEU	Inter
Published Results							
ResNet18 [112]	0.9004	0.9024	0.8966	0.8994	0.9034	0.9000	0.9032
OS_IS13 [132]	0.8996	0.9022	0.8919	0.8980	0.9065	0.8991	0.8999
Models without Speaking Style							
C(256)+B+P+D	0.8996	0.9017	0.8981	0.8980	0.9034	0.8968	0.9013
C(32)+B+P+2D	0.8999	0.9021	0.8970	0.8984	0.9038	0.8981	0.9017
C(32)+P+2D	0.9004	0.9023	0.8964	0.9005	0.9047	0.8983	0.9020
C(128)+P+2D	0.8993	0.9027	0.8948	0.8983	0.9040	0.8967	0.9013
C(256)+P+2D	0.9001	0.9022	0.8967	0.8994	0.9043	0.8979	0.9022
Models with Speaking Style							
C(32)+G+B+D	0.9013	0.9025	0.9008	0.9004	0.9035	0.8993	0.9044
C(128)+G+B+D	0.9050	0.9055	0.9054	0.9040	0.9063	0.9038	0.9083
C(256)+G+B+D	0.9053	0.9058	0.9055	0.9049	0.9068	0.9037	0.9078
C(128)+G+B+2D	**0.9061**	0.9062	0.9072	0.9049	0.9073	0.9049	**0.9101**

TABLE 5.6: Big five traits and interview variable F1 score results for different systems.

System	Ave	OPE	CON	EXT	AGR	NEU	Inter
Published Results							
OS_IS13 [133]	67.93	-	-	-	-	-	69.25
Models without Speaking Style							
C(32)+P+2D	68.35	70.15	69.90	68.50	64.79	68.40	69.30
Models with Speaking Style							
C(128)+G+B+2D	**70.92**	70.45	74.16	70.50	66.44	73.05	**72.20**

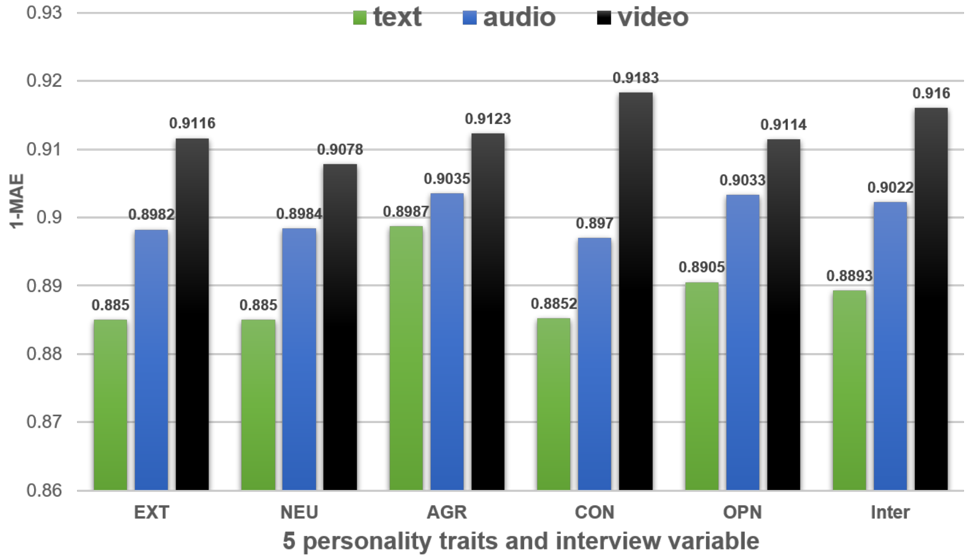


FIGURE 5.10: Single modality architecture used in our system.

- **Standard concatenation** Next, all modalities are combined using the standard concatenation method as illustrated in Fig.5.11.
- **Multistages training** For comparison, we also trained the network with our proposed architecture shown in Fig.5.8 by only minimizing big5 loss function once (without multiple training stages). The state of the art (SOTA), Standard concatenation (concat), without multiple training stages (only structure) and our proposed multistages training (3 stages) results are summarized in in Fig.5.12.

  Yellow one is the state-of-the-art system. Green one is simple concatenation of these 3 modalities. Blue one is the system that keeps the architecture but only trained in standard way. The black one is the system that trained using our proposed method. We can see only that the structure itself doesn't give much improvement. But train it in the proposed method, it even become comparable with state-of-the-art system.

5.7 Summary

5.7.1 myPersonality

In this work, we applied deep learning approaches including convolutional neural networks and recurrent neural networks on the shared task from "Workshop on Computational Personality Recognition (Shared Task)" [23].

The results showed that FC models with only text are worse than the ones with author information. DNN with both text and author information achieves the best results. CNN, RNN and FC are able to automatically extract useful features for personality recognition and the best result of 60.0±6.5% F1 score was obtained using CNN with average pooling. We

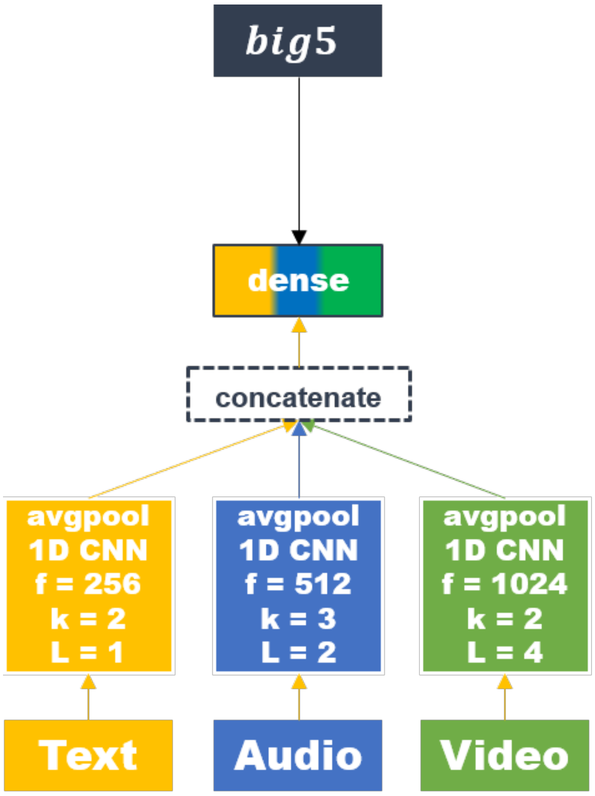


FIGURE 5.11: Standard concatenation architecture used in our system.

found that Bi-gram, tri-gram and recurrent architecture didn't get better results. It may indicate that words chosen by the author tell more about author's personality than the meaning author expresses. We also noticed that applying regularization barely restrains the overfitting, the network learns the patterns that only exist in the training set. This may be improved by collecting more data or by transforming the text into a representation which is stable for personality by external knowledge.

In the future work, we plan to apply unsupervised learning on the text data and use external knowledge about personality to cluster the text.

5.7.2 First impressions V2

In this work, we applied convolutional neural networks for high-level feature extraction and multistages training for multimodal learning fusito build an automatic apparent personality recognition system that is capable to make good predictions even some modalities are missing.

The experimental results showed that our multistage training strategy with attention can significantly improve improved the AAPR result from 0.9158 to 0.9175 (the state of the art result is 0.9173).

We are focusing on the multimodal learning in this work, but many parts of our current system can be improved and combined with other techniques, such as increasing the sampling rate of images, improving the architectures of feature extraction networks. In the future work, we plan to apply techniques from action detection to improve the deep learning based

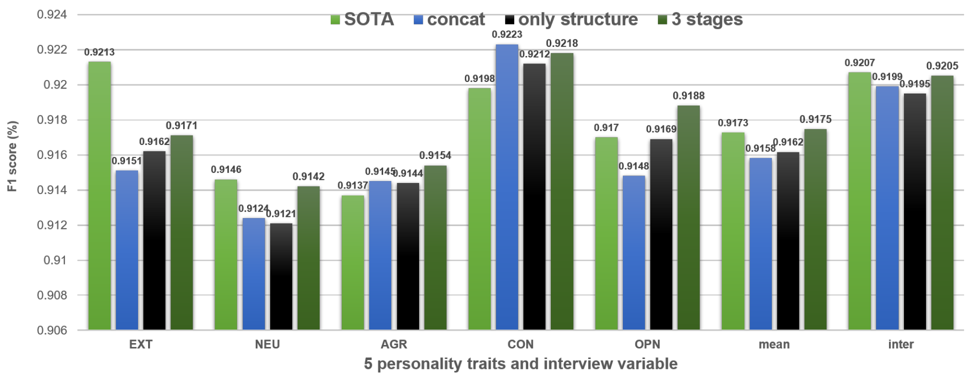


FIGURE 5.12: multistages training architecture used in our system.

high-level feature extraction part and try to learn the common representations between (T, A, V) directly instead of (t, a, v).

Chapter 6

Contributions and Conclusions

6.1 Contributions

Here we listed our contributions:

1. Proposed an ASR system that use acoustic and articulatory information as a regularization to guide training of a model using only acoustic features.

2. Proposed an ASR system that jointly train a hybrid model combining inversion model and the acoustic model.

3. Proposed a system to automatically learn the text representation for APR, which is combined with author information.

4. Proposed an architecture that automatically captures speaking styles for AAPR.

5. Proposed a training strategy that deals with the different convergence speeds of multiple modalities.

6.2 Conclusions

From our ASR work we can observed that:

1. The conventional GMM-HMM system using only acoustic feature gets the PER of 20.05±3.05%. When real articulatory measurements are also used, the system got the PER of 10.06±1.82%. This is a relative PER reduction of **49.8%**, which also confirmed that articulatory information is very helpful for ASR system.

2. When replacing the GMM with feedforward neural network for acoustic modelling, MFC-feature system got a PER of 8.23±1.49%, MFC+ART feature system got a PER of 4.74±0.55%. Both systems got a big performance boost compared to the conventional GMM-HMM systems.

3. The ASR system based feedforward neural network without any help of articulatory information have the PER of 8.23±1.49. Surprisingly, when the hard targets during training of student DNN model are provided from the GMM-HMM baseline containing both acoustic and articulatory feature, about **8.5%** PER reduction can be achieved. This is a very simple but effective way to utilize articulatory information.

4. Furthermore, about **14.6%** PER reduction can be observed when distillation training are also applied. Distillation training using soft-targets provided by a "rich" informative teacher model provides the relationships between classes in output space and can be treated as a very good regularization method or pre-training method.

5. The articulatory inversion method based on feedforward neural network is the best system that achieves **20%** PER reduction. Dislike distillation method, inversion method provides more more informative input representation that helps system to distinguish different classes. However inversion method requires doubled computational cost and time than distillation method, since it has another DNN model that learns the mapping between acoustic and articulatory feature spaces. Although it is the most accurate ASR system, but it is also the most expensive ASR system.

6. The same conclusions can also be observed when replacing the feedforward neural network with the Recurrent neural network. Recurrent neural network acoustic model using only acoustic feature can achieve the performance that MFC+ART-feature feedforward neural network gets and don't need to pre-defined the window-size, since it can learn which input is important or irrelevant.

From both works we can observed that:

1. When automatically ex-tracting features for different modalities, it is really helpful for the extrac-tors to know the final objective, which can be done by introducing the fi-nal loss function to the extractors.

2. The typical way to merge differentmodalities using deep learning is to put them into a big single network.However, because this big network has many parts whose speeds of con-vergences are different, for example, the parts related to the audio modalitymay have already converged, but the parts related to the video modality arestill training, it is hard for this single network to find optimal parametersfor all modalities. One way to tackle this problem is to set the parametersof particular modalities untrainable and force the other parts to map thedesired modalities to a good latent space first. In other words, constrain thefreedom of deep neural network's automatic feature extraction.

6.3 Future Work

In this work, all systems preform as good as expected, but many parts can also be improved as below:

1. The articulatory feature used in this work is only the movements of articulators, which is very simple. However potential articulatory information can be lost in such representation. The next attempt is to use images of vocal tract through time. The dimension can be reduced using convolutional autoencoder.

2. There also are other issues to be investigated within this framework including the effect of the teacher performance on training set, more sophisticated ways of "teaching", not just linear combination of loss functions.

3. Recurrent neural network preforms extremely good, but its computational cost prevents it from widely using in practice. My next experiment could be transferring information from RNN to DNN model.

4. articulatory inversion based RNN is the best ASR system among the proposed ones, but it has to learn the mapping between acoustic and articulatory feature spaces, then learns the posterior probability $P(o|w)$, which seems to be redundant. Because neural network is a good technique for regression, I could learn the mapping between acoustic feature and the representation transformed from MFC+ART feature directly.

Appendix A

Phoneme list

TABLE A.1: *Phoneme list*

Phoneme	Example	Translation	Phoneme	Example	Translation
AA	odd	AA D	L	lee	L IY
AE	at	AE T	M	me	M IY
AH	hut	HH AH T	N	knee	N IY
AO	ought	AO T	NG	ping	P IH NG
AW	cow	K AW	OW	oat	OW T
AY	hide	HH AY D	OY	toy	T OY
B	be	B IY	P	pee	P IY
CH	cheese	CH IY Z	R	read	R IY D
D	dee	D IY	S	sea	S IY
DH	thee	DH IY	SH	she	SH IY
EH	Ed	EH D	T	tea	T IY
ER	hurt	HH ER T	TH	theta	TH EY T AH
EY	ate	EY T	UH	hood	HH UH D
F	fee	F IY	UW	two	T UW
G	green	G R IY N	V	vee	V IY
HH	he	HH IY	W	we	W IY
IH	it	IH T	Y	yield	Y IY L D
IY	eat	IY T	Z	zee	Z IY
JH	gee	JH IY	ZH	seizure	S IY ZH ER
K	key	K IY			

Appendix B

Articulatory Inversion Plots

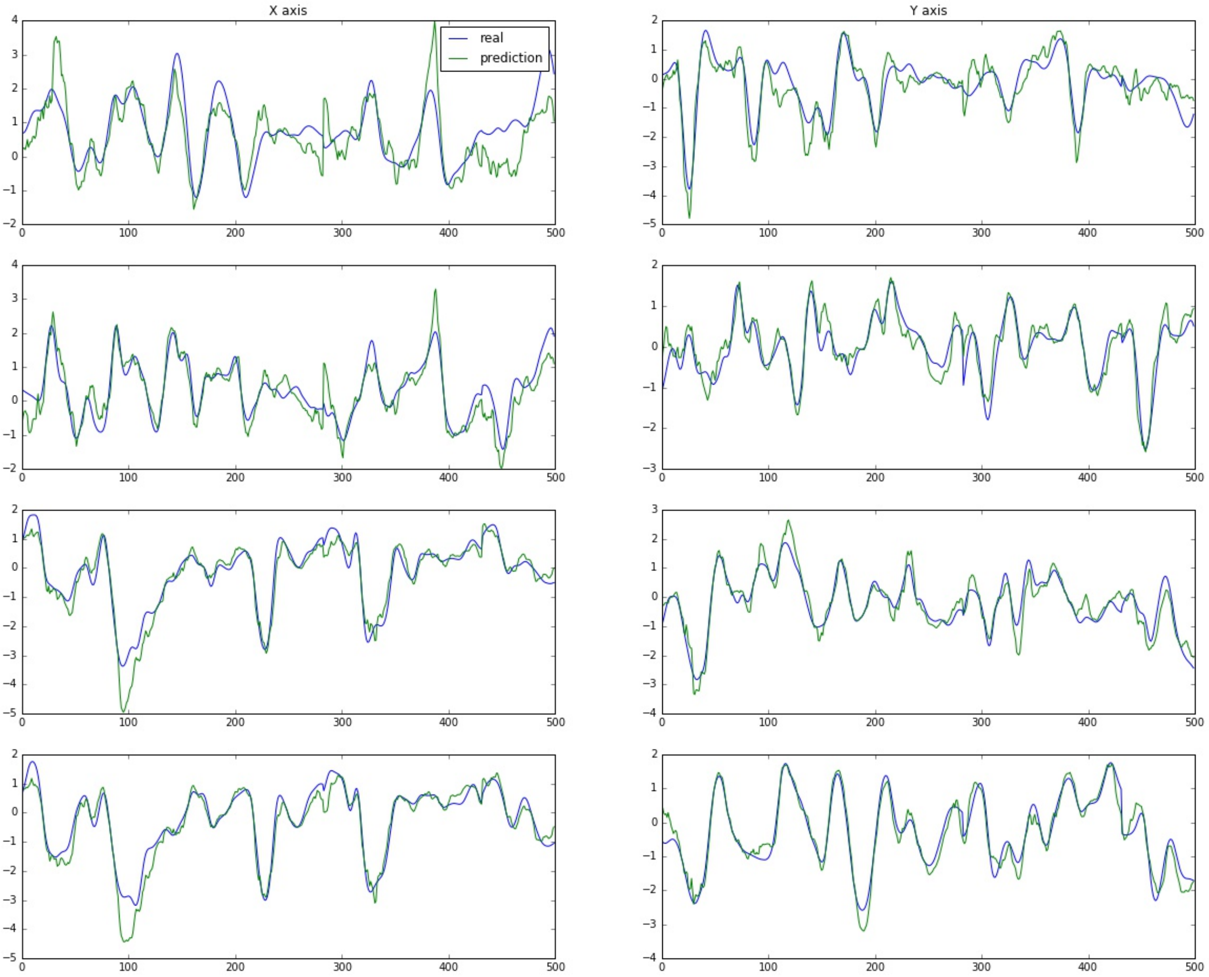


FIGURE B.1: *feedforward net predictions of articulators UL (upper lip), LL (lower lip), T1 (ventral tongue), T2 (mid-tongue). The x axis of all plots is timestep (10ms). The y axis of all plots is the normalized position*

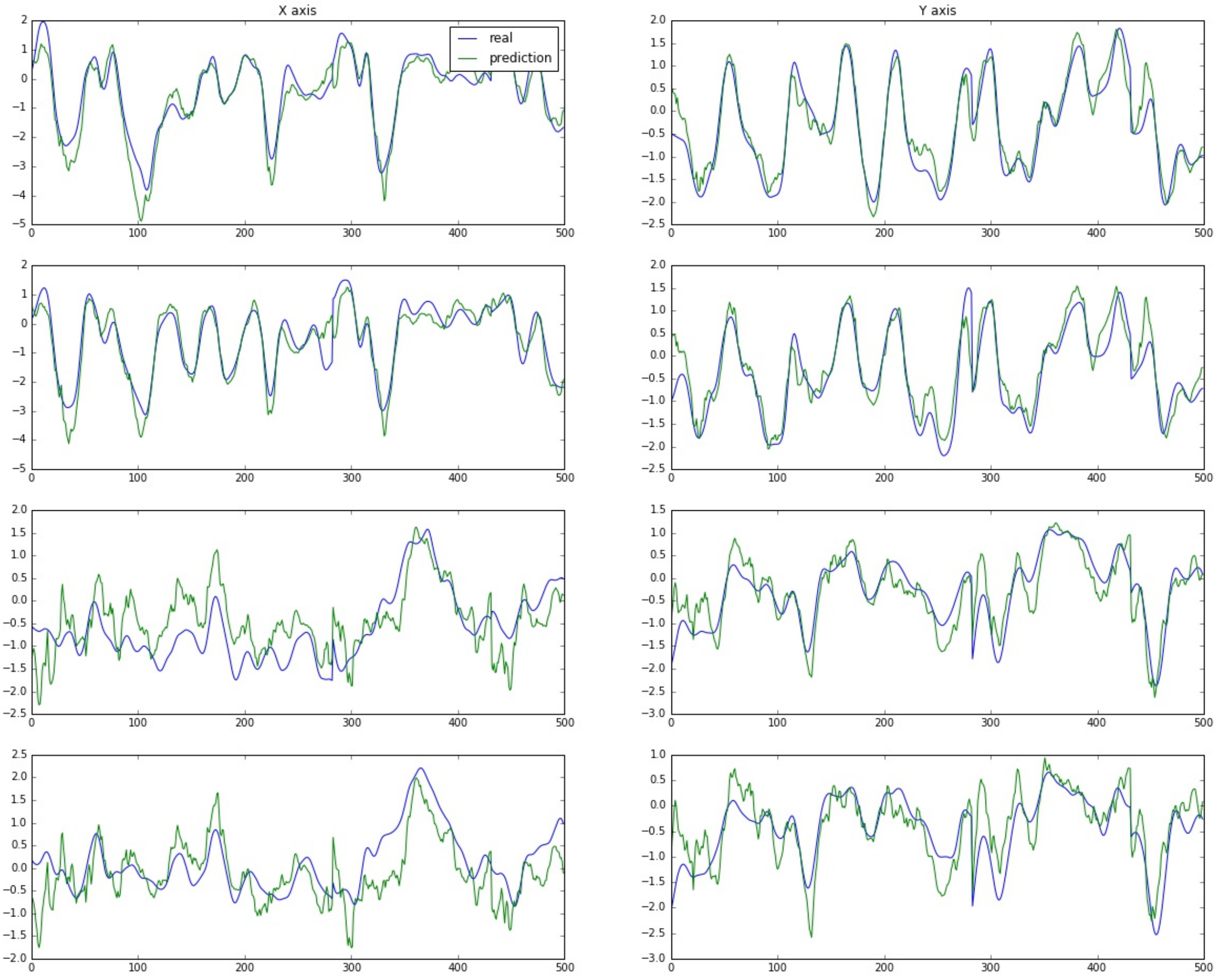


FIGURE B.2: *feedforward net predictions of articulators T3 (mid-tongue), T4 (dorsal tongue), MANm(mandibular), MANi*

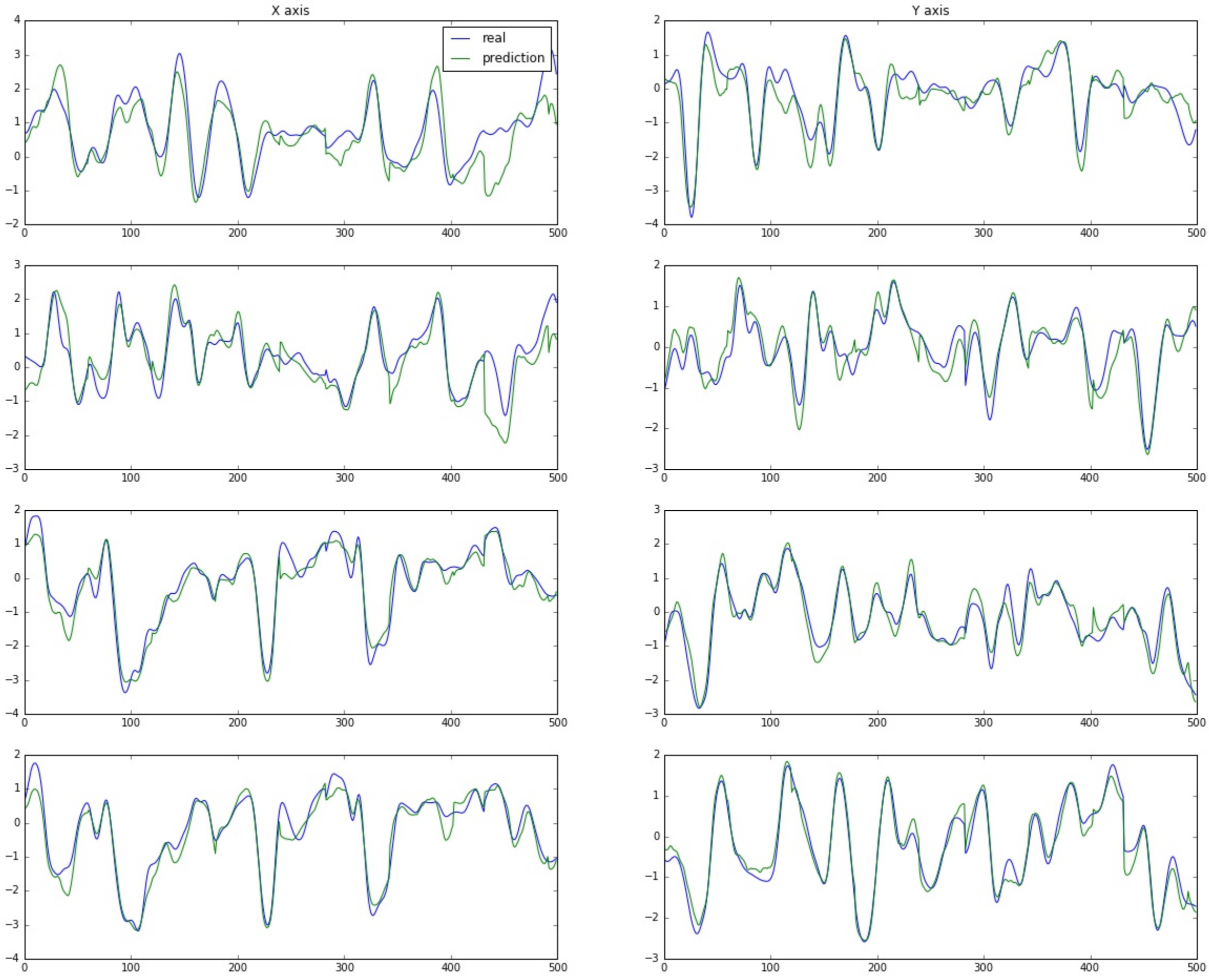


FIGURE B.3: *rnn net predictions of articulators UL, LL, T1, T2*

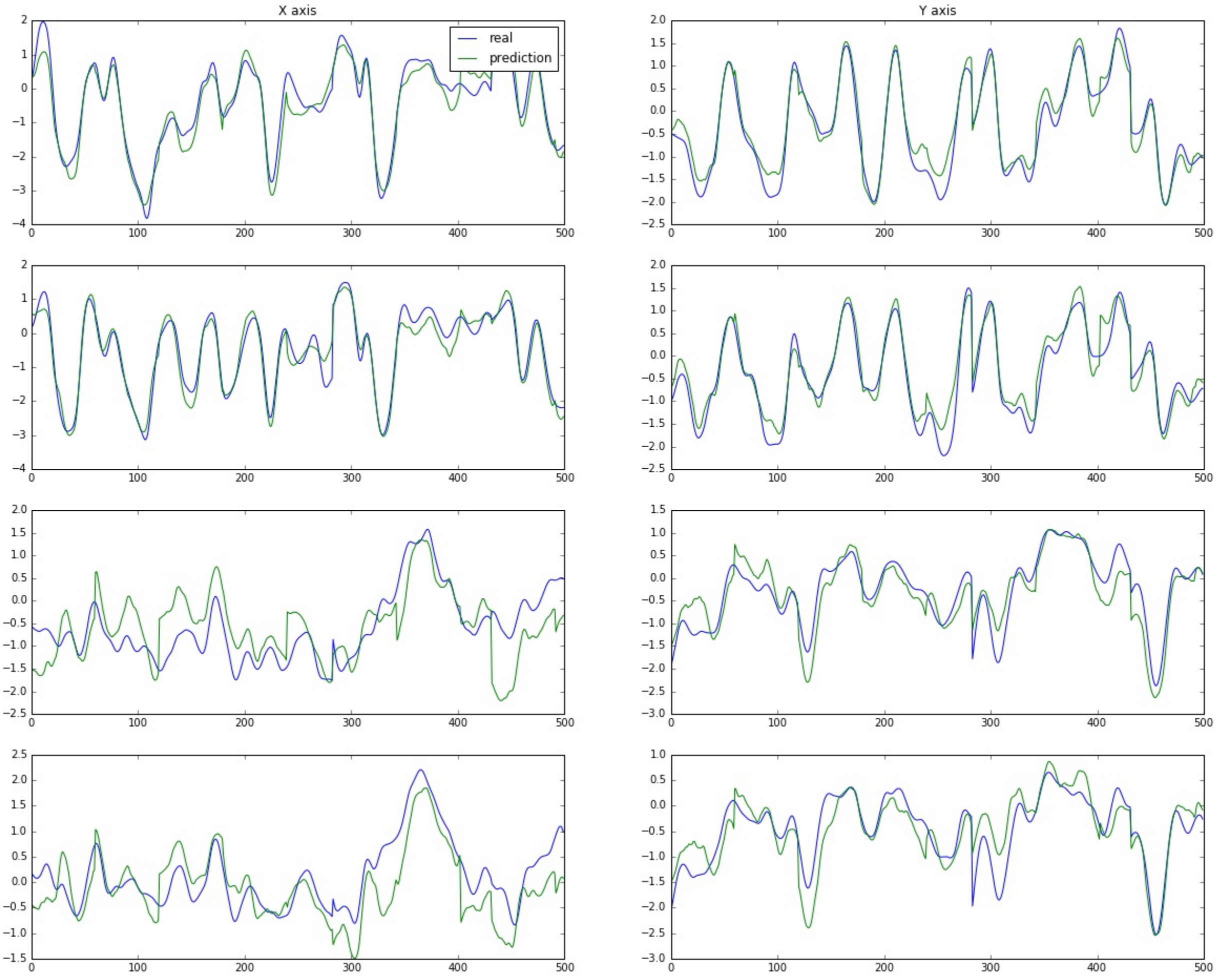


FIGURE B.4: *rnn net predictions of articulators T3, T4, MANm, MANi*

Appendix C

Distillation Plots

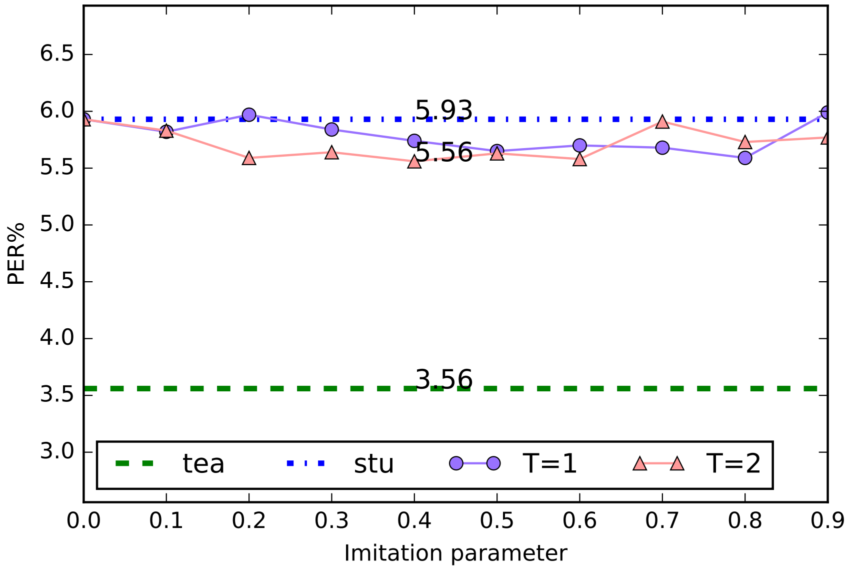


FIGURE C.1: *distillation results for fold 0*

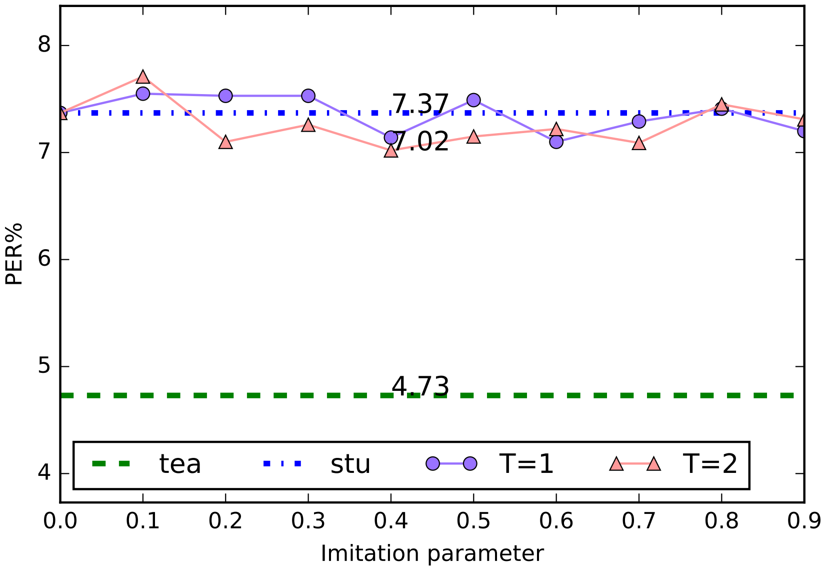


FIGURE C.2: *distillation results for fold 1*

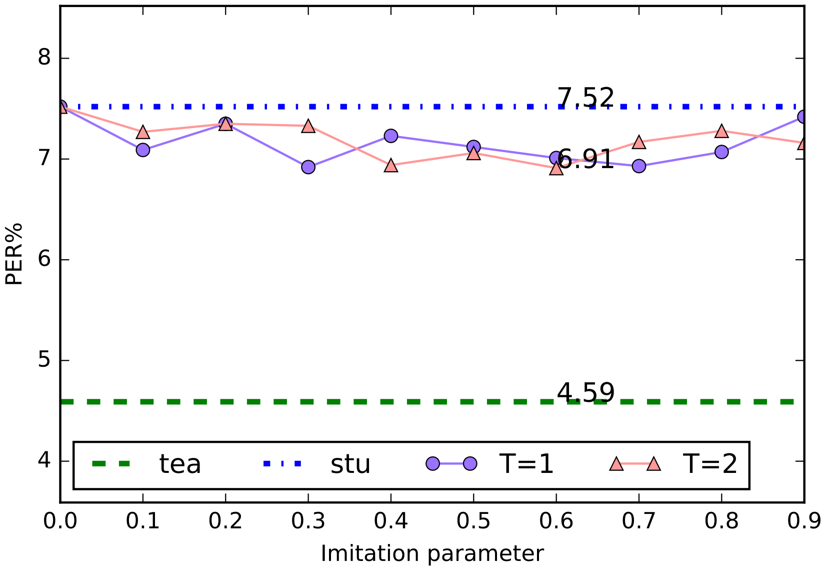


FIGURE C.3: *distillation results for fold 2*

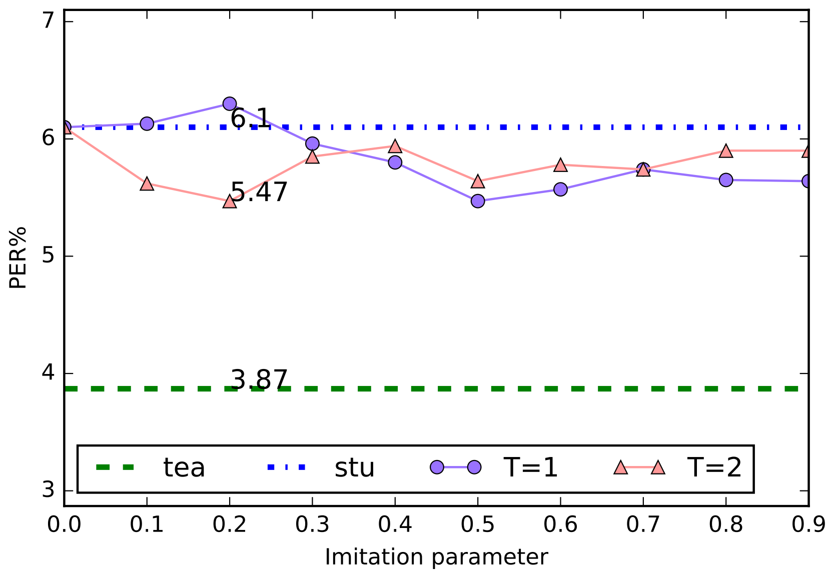


FIGURE C.4: *distillation results for fold 3*

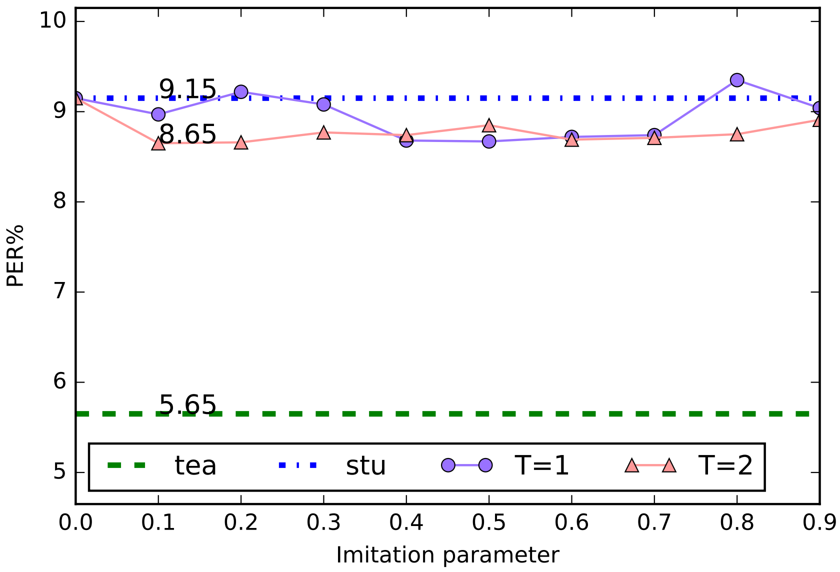


FIGURE C.5: *distillation results for fold 4*

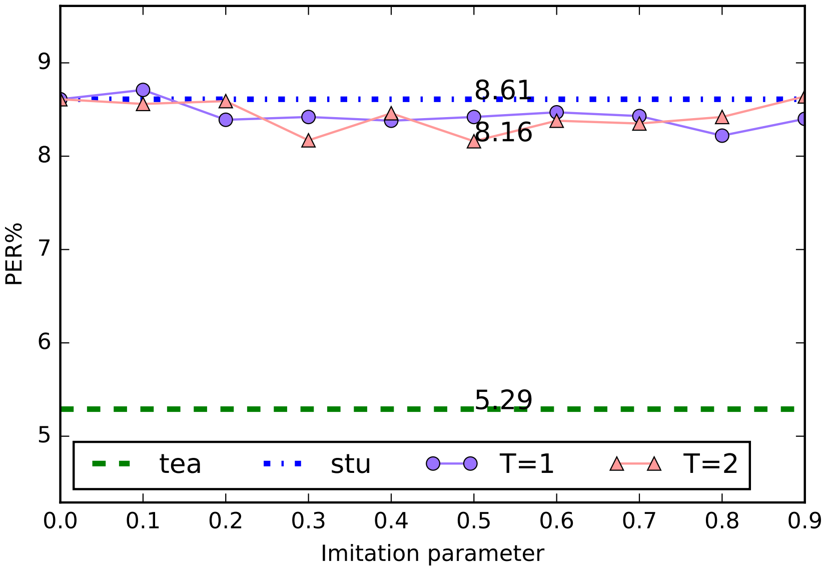


FIGURE C.6: *distillation results for fold 5*

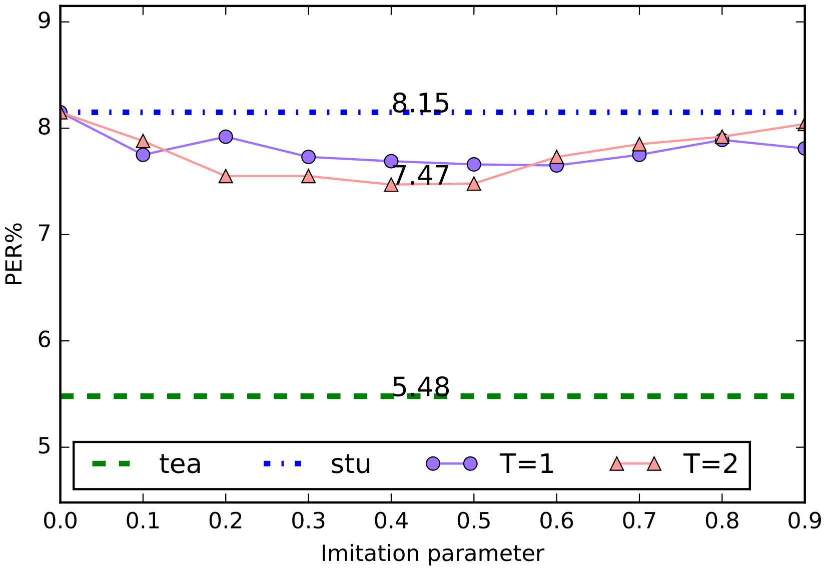


FIGURE C.7: *distillation results for fold 6*

Bibliography

[1] Ben P Yuhas, Moise H Goldstein, and Terrence J Sejnowski. "Integration of acoustic and visual speech signals using neural networks". In: *IEEE Communications Magazine* 27.11 (1989), pp. 65–71.

[2] Harry McGurk and John MacDonald. "Hearing lips and seeing voices". In: *Nature* 264.5588 (1976), p. 746.

[3] Ryan Kiros et al. "Stacked multiscale feature learning for domain independent medical image segmentation". In: *International workshop on machine learning in medical imaging*. Springer. 2014, pp. 25–32.

[4] Pengcheng Wu et al. "Online multimodal deep similarity learning with application to image retrieval". In: *Proceedings of the 21st ACM international conference on Multimedia*. ACM. 2013, pp. 153–162.

[5] Siqi Liu et al. "Multimodal neuroimaging feature learning for multiclass diagnosis of Alzheimer's disease". In: *IEEE Transactions on Biomedical Engineering* 62.4 (2015), pp. 1132–1140.

[6] Polina Mamoshina et al. "Applications of deep learning in biomedicine". In: *Molecular pharmaceutics* 13.5 (2016), pp. 1445–1454.

[7] Dhanesh Ramachandram and Graham W Taylor. "Deep multimodal learning: A survey on recent advances and trends". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 96–108.

[8] Alessandro Vinciarelli and Gelareh Mohammadi. "A survey of personality computing". In: *IEEE Transactions on Affective Computing* 5.3 (2014), pp. 273–291.

[9] Lukasz Piwek et al. "The rise of consumer health wearables: promises and barriers". In: *PLoS Medicine* 13.2 (2016), e1001953.

[10] Martyn Denscombe. *The good research guide: for small-scale social research projects*. McGraw-Hill Education (UK), 2014.

[11] Chung-Hsien Yu et al. "Crime forecasting using data mining techniques". In: *2011 IEEE 11th international conference on data mining workshops*. IEEE. 2011, pp. 779–786.

[12] Pradeep K Atrey et al. "Multimodal fusion for multimedia analysis: a survey". In: *Multimedia systems* 16.6 (2010), pp. 345–379.

[13] Bahador Khaleghi et al. "Multisensor data fusion: A review of the state-of-the-art". In: *Information fusion* 14.1 (2013), pp. 28–44.

[14] Nicu Sebe et al. *Machine learning in computer vision*. Vol. 29. Springer Science & Business Media, 2005.

[15] Konstantin Markov, Jianwu Dang, and Satoshi Nakamura. "Integration of articulatory and spectrum features based on the hybrid HMM/BN modeling framework". In: *Speech Communication* 48.2 (2006), pp. 161–175.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[17] Natalia Neverova et al. “Moddrop: adaptive multi-modal gesture recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.8 (2016), pp. 1692–1706.

[18] Kihyuk Sohn, Wenling Shang, and Honglak Lee. “Improved multimodal deep learning with variation of information”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2141–2149.

[19] Ashesh Jain et al. “Recurrent neural networks for driver activity anticipation via sensory-fusion architecture”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3118–3125.

[20] David Lopez-Paz et al. “Unifying distillation and privileged information”. In: *ICLR*. 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[22] Jianguo Yu, Konstantin Petrov Markov, and Tomoko Matsui. “Articulatory and Spectrum Information Fusion Based on Deep Recurrent Neural Networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2019).

[23] Fabio Celli et al. “Workshop on computational personality recognition (shared task)”. In: *Proceedings of the Workshop on Computational Personality Recognition*. 2013.

[24] Lewis R Goldberg. “The structure of phenotypic personality traits.” In: *American psychologist* 48.1 (1993), p. 26.

[25] Steven Davis and Paul Mermelstein. “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences”. In: *IEEE transactions on acoustics, speech, and signal processing* 28.4 (1980), pp. 357–366.

[26] Dan Jurafsky et al. “What kind of pronunciation variation is hard for triphones to model?” In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*. Vol. 1. IEEE. 2001, pp. 577–580.

[27] Vikramjit Mitra et al. “Gesture-based dynamic Bayesian network for noise robust speech recognition”. In: *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5172–5175.

[28] Vikramjit Mitra et al. “Recognizing articulatory gestures from speech for robust speech recognition”. In: *The Journal of the Acoustical Society of America* 131.3 (2012), pp. 2270–2287.

[29] William J Hardcastle and Nigel Hewlett. *Coarticulation: Theory, data and techniques*. Cambridge University Press, 2006.

[30] Hosung Nam et al. “A procedure for estimating gestural scores from speech acoustics”. In: *The Journal of the Acoustical Society of America* 132.6 (2012), pp. 3980–3989.

[31] Karen Simonyan and Andrew Zisserman. “Two-stream convolutional networks for action recognition in videos”. In: *Advances in neural information processing systems*. 2014, pp. 568–576.

[32] Joao Carreira and Andrew Zisserman. “Quo vadis, action recognition? a new model and the kinetics dataset”. In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[33] Will Y Zou et al. “Bilingual word embeddings for phrase-based machine translation”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1393–1398.

[34] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117.

[35] Christopher Olah. “NN-Manifolds-Topology”. In: (). URL: `http://goo.gl/yA7kUW`.

[36] Henry W Lin, Max Tegmark, and David Rolnick. “Why does deep and cheap learning work so well?” In: *Journal of Statistical Physics* 168.6 (2017), pp. 1223–1247.

[37] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[38] Guido F Montufar et al. “On the number of linear regions of deep neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 2924–2932.

[39] Hrushikesh Narhar Mhaskar and Charles A Micchelli. “How to choose an activation function”. In: *Advances in Neural Information Processing Systems* (1994), pp. 319–319.

[40] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. “Advances in optimizing recurrent networks”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 8624–8628.

[41] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[42] James Martens. “Deep learning via Hessian-free optimization”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 735–742.

[43] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: (). URL: `http://goo.gl/MYZmxA`.

[44] Bert Moons, Daniel Bankman, and Marian Verhelst. “Embedded Deep Neural Networks”. In: *Embedded Deep Learning*. Springer, 2019, pp. 1–31.

[45] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[46] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks.” In: *ICML (3)* 28 (2013), pp. 1310–1318.

[47] Christopher Olah. “Understanding LSTMs”. In: (). URL: http://goo.gl/hwG4OC.

[48] Andrej Karpathy. “The Unreasonable Effectiveness of Recurrent Neural Networks”. In: (). URL: https://goo.gl/hwPVkF.

[49] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[50] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[51] Wojciech Zaremba. “An empirical exploration of recurrent network architectures”. In: (2015).

[52] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *arXiv preprint arXiv:1511.07122* (2015).

[53] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*. 2016, pp. 3844–3852.

[54] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[55] Thomas N. Kipf. “GRAPH CONVOLUTIONAL NETWORKS”. In: (). URL: https://tkipf.github.io/graph-convolutional-networks.

[56] Mingyi He, Bo Li, and Huahui Chen. “Multi-scale 3D deep convolutional neural network for hyperspectral image classification”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 3904–3908.

[57] Joseph Turian, Lev Ratinov, and Yoshua Bengio. “Word representations: a simple and general method for semi-supervised learning”. In: *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics. 2010, pp. 384–394.

[58] Thang Luong, Hieu Pham, and Christopher D Manning. “Bilingual word representations with monolingual quality in mind”. In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 2015, pp. 151–159.

[59] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[60] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[61] Armand Joulin et al. “FastText.zip: Compressing text classification models”. In: *arXiv preprint arXiv:1612.03651* (2016).

[62] Simon King et al. “Speech production knowledge in automatic speech recognition”. In: *The Journal of the Acoustical Society of America* 121.2 (2007), pp. 723–742.

[63] Slava Katz. “Estimation of probabilities from sparse data for the language model component of a speech recognizer”. In: *IEEE transactions on acoustics, speech, and signal processing* 35.3 (1987), pp. 400–401.

[64] Sean R Eddy. “Hidden markov models”. In: *Current opinion in structural biology* 6.3 (1996), pp. 361–365.

[65] Lloyd R Welch. “Hidden Markov models and the Baum-Welch algorithm”. In: *IEEE Information Theory Society Newsletter* 53.4 (2003), pp. 10–13.

[66] Jeff A Bilmes et al. “A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models”. In: (1998).

[67] George E Dahl et al. “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (2012), pp. 30–42.

[68] Geoffrey Hinton et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.

[69] G David Forney. “The viterbi algorithm”. In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[70] Karen Livescu et al. “Articulatory feature-based methods for acoustic and audio-visual speech recognition: Summary from the 2006 JHU summer workshop”. In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–621.

[71] Katrin Kirchhoff, Gernot A Fink, and Gerhard Sagerer. “Combining acoustic and articulatory feature information for robust speech recognition”. In: *Speech Communication* 37.3 (2002), pp. 303–319.

[72] Joe Frankel et al. “An automatic speech recognition system using neural networks and linear dynamic models to recover and model articulatory traces”. In: (2000).

[73] Korin Richmond. “Estimating articulatory parameters from the acoustic speech signal”. PhD thesis. University of Edinburgh, 2002.

[74] Le Zhang and Steve Renals. “Acoustic-articulatory modeling with the trajectory HMM”. In: *IEEE Signal Processing Letters* 15 (2008), pp. 245–248.

[75] Tomoki Toda, Alan W Black, and Keiichi Tokuda. “Acoustic-to-articulatory inversion mapping with Gaussian mixture model.” In: *INTERSPEECH*. 2004.

[76] Korin Richmond. “Trajectory mixture density networks with multiple mixtures for acoustic-articulatory inversion”. In: *International Conference on Nonlinear Speech Processing*. Springer. 2007, pp. 263–272.

[77] Rajkumar Arora and Karen Livescu. “Multi-view CCA-based acoustic features for phonetic recognition across speakers and domains”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 7135–7139.

[78] Weiran Wang et al. “Unsupervised learning of acoustic features via deep canonical correlation analysis”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4590–4594.

[79] Benigno Uria, Steve Renals, and Korin Richmond. “A deep neural network for acoustic-articulatory speech inversion”. In: *NIPS 2011 Workshop on Deep Learning and Unsupervised Feature Learning*. 2011.

[80] Benigno Uria et al. “Deep Architectures for Articulatory Inversion.” In: *INTERSPEECH*. 2012, pp. 867–870.

[81] Vikramjit Mitra et al. “Articulatory features from deep neural networks and their role in speech recognition”. In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 3017–3021.

[82] Leonardo Badino et al. “Integrating articulatory data in deep neural network-based acoustic modeling”. In: *Computer Speech & Language* 36 (2016), pp. 173–195.

[83] Vladimir Vapnik and Rauf Izmailov. “Learning Using Privileged Information: Similarity Control and Knowledge Transfer”. In: *Journal of Machine Learning Research* 16 (2015), pp. 2023–2049.

[84] Jianguo Yu, Konstantin Markov, and Tomoko Matsui. “Articulatory and spectrum features integration using generalized distillation framework”. In: *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE. 2016, pp. 1–6.

[85] Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. “Recurrent neural network training with dark knowledge transfer”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 5900–5904.

[86] J Westbury. “X-RAY MICROBEAM SPEECH PRODUCTION DATABASE USER’S HANDBOOK. 1994”. In: *Waisman Center, University of Wisconsin: Madison, USA* (), pp. 1–100.

[87] Carnegie Mellon University. “The CMU dictionary”. In: (). URL: http://goo.gl/xW7VAM.

[88] Weiran Wang, Raman Arora, and Karen Livescu. “Reconstruction of articulatory measurements with smoothed low-rank matrix completion”. In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE. 2014, pp. 54–59.

[89] Jiahong Yuan and Mark Liberman. “Speaker identification on the SCOTUS corpus”. In: *Journal of the Acoustical Society of America* 123.5 (2008), p. 3878.

[90] Pengcheng Zhu, Lei Xie, and Yunlin Chen. “Articulatory Movement Prediction Using Deep Bidirectional Long Short-Term Memory Based Recurrent Neural Networks and Word/Phone Embeddings”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[91] William Chan and Ian Lane. “Deep Recurrent Neural Networks for Acoustic Modelling”. In: *CoRR* abs/1504.01482 (2015).

[92] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *CoRR* abs/1312.6120 (2013).

[93] Peng Liu et al. "A deep recurrent approach for acoustic-to-articulatory inversion". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 4450–4454.

[94] Weiran Wang et al. "On deep multi-view representation learning". In: *International Conference on Machine Learning*. 2015, pp. 1083–1092.

[95] Qingming Tang, Weiran Wang, and Karen Livescu. "Acoustic Feature Learning Using Cross-Domain Articulatory Measurements". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 4849–4853.

[96] Alessandro Vinciarelli and Gelareh Mohammadi. "A survey of personality computing". In: *IEEE Transactions on Affective Computing* 5.3 (2014), pp. 273–291.

[97] Heysem Kaya and Albert Ali Salah. "Continuous mapping of personality traits: A novel challenge and failure conditions". In: *Proceedings of the 2014 Workshop on Mapping Personality Traits Challenge and Workshop*. ACM. 2014, pp. 17–24.

[98] Paul T Costa Jr and Robert R McCrae. "Domains and facets: Hierarchical personality assessment using the Revised NEO Personality Inventory". In: *Journal of personality assessment* 64.1 (1995), pp. 21–50.

[99] Tim Polzehl, Sebastian Moller, and Florian Metze. "Automatically assessing personality from speech". In: *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*. IEEE. 2010, pp. 134–140.

[100] Ben Verhoeven, Walter Daelemans, and Tom De Smedt. "Ensemble methods for personality recognition". In: *Proceedings of the workshop on computational personality recognition*. 2013, pp. 35–38.

[101] Firoj Alam, Evgeny A Stepanov, and Giuseppe Riccardi. "Personality traits recognition on social network-facebook". In: *WCPR (ICWSM-13), Cambridge, MA, USA* (2013).

[102] Golnoosh Farnadi et al. "Recognising personality traits using Facebook status updates". In: *Proceedings of the workshop on computational personality recognition (WCPR13) at the 7th international AAAI conference on weblogs and social media (ICWSM13)*. AAAI. 2013.

[103] Marc T Tomlinson, David Hinote, and David B Bracewell. "Predicting conscientiousness through semantic analysis of facebook posts". In: *Proceedings of WCPR* (2013).

[104] James W Pennebaker, Martha E Francis, and Roger J Booth. "Linguistic inquiry and word count: LIWC 2001". In: *Mahway: Lawrence Erlbaum Associates* 71.2001 (2001), p. 2001.

[105] Dejan Markovikj et al. "Mining facebook data for predictive personality modeling". In: *Proceedings of the 7th international AAAI conference on Weblogs and Social Media (ICWSM 2013), Boston, MA, USA*. 2013, pp. 23–26.

[106] Navonil Majumder et al. “Deep Learning-Based Document Modeling for Personality Detection from Text”. In: *IEEE Intelligent Systems* 32.2 (2017), pp. 74–79.

[107] James W Pennebaker and Laura A King. “Linguistic styles: language use as an individual difference.” In: *Journal of personality and social psychology* 77.6 (1999), p. 1296.

[108] Víctor Ponce-López et al. “Chalearn lap 2016: First round challenge on first impressions-dataset and results”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 400–418.

[109] Chen-Lin Zhang et al. “Deep bimodal regression for apparent personality analysis”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 311–324.

[110] Arulkumar Subramaniam et al. “Bi-modal first impressions recognition using temporally ordered deep audio and stochastic visual features”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 337–348.

[111] Yağmur Güçlütürk et al. “Deep impression: Audiovisual deep residual networks for multimodal apparent personality trait recognition”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 349–358.

[112] Hugo Jair Escalante et al. “Explaining First Impressions: Modeling, Recognizing, and Explaining Apparent Personality from Videos”. In: *arXiv preprint arXiv:1802.00745* (2018).

[113] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[114] François Mairesse et al. “Using linguistic cues for the automatic recognition of personality in conversation and text”. In: *Journal of artificial intelligence research* 30 (2007), pp. 457–500.

[115] Navonil Majumder et al. “Deep learning-based document modeling for personality detection from text”. In: *IEEE Intelligent Systems* 32.2 (2017), pp. 74–79.

[116] Jianguo Yu and Konstantin Markov. “Deep learning based personality recognition from Facebook status updates”. In: *Awareness Science and Technology (iCAST), 2017 IEEE 8th International Conference on*. IEEE. 2017, pp. 383–387.

[117] Tim Polzehl, Sebastian Moller, and Florian Metze. “Automatically assessing personality from speech”. In: *2010 IEEE Fourth International Conference on Semantic Computing*. IEEE. 2010, pp. 134–140.

[118] Gelareh Mohammadi and Alessandro Vinciarelli. “Automatic personality perception: Prediction of trait attribution based on prosodic features”. In: *IEEE Transactions on Affective Computing* 3.3 (2012), pp. 273–284.

[119] Fabio Pianesi et al. “Multimodal recognition of personality traits in social interactions”. In: *Proceedings of the 10th international conference on Multimodal interfaces*. ACM. 2008, pp. 53–60.

[120] Björn Schuller et al. "The interspeech 2012 speaker trait challenge". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[121] Clément Chastagnol and Laurence Devillers. "Personality traits detection using a parallelized modified SFFS algorithm". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[122] Alexei Ivanov and Xin Chen. "Modulation spectrum analysis for speaker personality trait recognition". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[123] Claude Montacié and Marie-José Caraty. "Pitch and Intonation Contribution to Speakers' Traits Classification". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[124] Michelle Hewlett Sanchez et al. "Multi-system fusion of extended context prosodic and cepstral features for paralinguistic speaker trait classification". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[125] Florian Eyben, Martin Wöllmer, and Björn Schuller. "Opensmile: the munich versatile and fast open-source audio feature extractor". In: *Proceedings of the 18th ACM international conference on Multimedia*. ACM. 2010, pp. 1459–1462.

[126] Furkan Gürpinar, Heysem Kaya, and Albert Ali Salah. "Multimodal fusion of audio, scene, and face features for first impression estimation". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 43–48.

[127] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A neural algorithm of artistic style". In: *arXiv preprint arXiv:1508.06576* (2015).

[128] Wei-Ta Chu and Yi-Ling Wu. "Image style classification based on learnt deep correlation features". In: *IEEE Transactions on Multimedia* 20.9 (2018), pp. 2491–2502.

[129] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[130] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[131] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015, pp. 2342–2350.

[132] Heysem Kaya, Furkan Gürpınar, and Albert Ali Salah. "Multi-modal Score Fusion and Decision Trees for Explainable Automatic Job Candidate Screening from Video CVs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 1–9.

[133] Heysem Kaya and Albert Ali Salah. "Multimodal Personality Trait Analysis for Explainable Modeling of Job Interview Decisions". In: *Explainable and Interpretable Models in Computer Vision and Machine Learning*. Springer, 2018, pp. 255–275.