Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Haoyu Tang

Hybrid attention convolutionacoustic model based on variationalautoencoder for speech recognition

Master's thesis in Electronic Systems Design
Supervisor: Torbjørn Karl Svendsen
June 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Hybrid attention convolution acoustic model based on variational autoencoder for speech recognition

Haoyu Tang

13-06-2019

Master's Thesis
Master of Science in Electronic system design
30 ECTS
Department of Electronic Systems
Norwegian University of Science and Technology,

Supervisor: Prof. Torbjørn Karl Svendsen
Co-Supervisor: Dr. Ali Shariq Imram; Dr. AdoIreza Sabzi Shahrebabaki

Preface

This Master Thesis was developed at the Department of Electronics system (Faculty of Information Technology, Mathematics and Electrical Engineering) at NTNU University (Trondheim, Norway). And the work is finished during my master period as my Master thesis from September 2017 to June 2018 including a 7.5 credits Specialization project as a forerunner, 7.5 credits Specialization course as a guide and 30 credits Master thesis.

The main work of this thesis is developing a new acoustic model in an automatic speech recognition system. And the main innovation in this project is that an new-introduced autoencoder in acoustics model transforms supervised-learning becoming semi-supervised learning. Also, attention mechanism is introduced as part of acoustics module for improvement.

For implement it in end-to-end CTC-attention architecture in further work, an training acceleration algorithm is developed and test as well.

The thesis could be read by speech relevant students, PhDs, and other industrial engineers. Since the acoustic module is realized by a neural network, this work might also inspire other deep learning relevant peoples.

13-06-2019

Acknowledgment

I wish to express my sincere gratitude to my supervisor at NTNU, Professor Torbjørn Karl Svendsen, for offering me the opportunity to work with him, and for his indispensable guide during the execution of the Master Thesis.

I want to thank my two co-supervisors Dr. Ali Shariq Imram and Dr. AdoIreza Sabzi Shahrebabaki for their great help during my master thesis especially for the help of about data's prepare.

During my 2018 summer internship in *Pattern Recognition Technique Center, Tencent*, my colleges *Jin Li* and *Jun Li* help me a lot. I would give my sincere thanks. Also, thanks for offering this internship from my supervisor in Tencent, *Long Ma* as well. After this internship, I decided to contribute my passion for speech field.

I would like to express my sincere great love to my parents. Their work ethic supports and confidence me during the whole of my life. Their support and confidence during all my life have made me as I am.

And thanks to the help from friends always next to me, Zhaoyi Liu (Peking University), Yingru Liu (The State University of New York at Stony Brook). With hundreds of online meetings, I can finish this master thesis. Thanks for their helpful advisement in academics again.

Haoyu Tang

Abstract

In a communication system, smart home and other speech-based application, Automatic Speech Recognition (ASR) plays a crucial role, and its inputs usually are features extracted from a raw speech signal. A well-designed feature extractor could improve the accuracy and reduced the computation complexity.

The feature extraction is a basic processing unit not only in ASR but also in Text to Speech (TTS), Speaker conversion and other conversion systems. Meanwhile, through the analysis of the feature extracted with a labeled phone in each frame, the result could be used for improving TTS as well.

The goal of the project is to build a new acoustic model used in ASR system based on neural network. Moreover, in the case, compared with main streaming acoustics modules, the most significant difference is that autoencoder introduces which makes the training of acoustic module becoming semi-supervised learning from supervised learning.

Feature extraction usually like Mel-frequency cpectrum coefficient (MFCC), Linear predictive coding (LPC), Warped Minimum Variance Distortionless response cepstral coefficients (WMVDRCC) are based on standard digital signal processing while neural network based feature extraction is uncommon.

Moreover, Deep Neural Network (DNN) has been proved as an efficient ASR method used the DNN module as an acoustic module in the ASR system. Meanwhile, the acoustic module could be built with Convolution Neural Network (CNN) or Long short-term memory recurrent neural network (LSTM).

In the master thesis, a hybrid attention mechanism is brought into acoustic. More specifically, time-frequency attention weighting in autoencoder, a front stage of the acoustic model. And channel attention weighting in a phone classifier, postage of acoustic model.

For accelerating this semi-supervised later development in VGG-BLSTM structure, a regularization method is developed for dynamic loss combine weighting. In this method, a regularization item is added to loss, which not also dynamic control weighting between two loss, but also push the weight to a preset value for smooth and steady loss decreasing.

Contents

List of Figures

List of Tables

1 Introduction

As speech becomes more and more nature and convenient interaction pattern in man-machine interaction and human communication. Automatic Speech Recognition (ASR) is much accounted and its inputs usually are features extracted from a raw speech signal. A well-designed feature extractor could improve the accuracy and reduced the computation complexity [9, Chapter 7].

How the feature extraction compose an essential processing unit in ASR, Text to Speech(TTS), Speaker conversion and other conversion systems is shown in the figure 1.

It shows that usually ASR and TTS don't process clean signal directly but through extracted

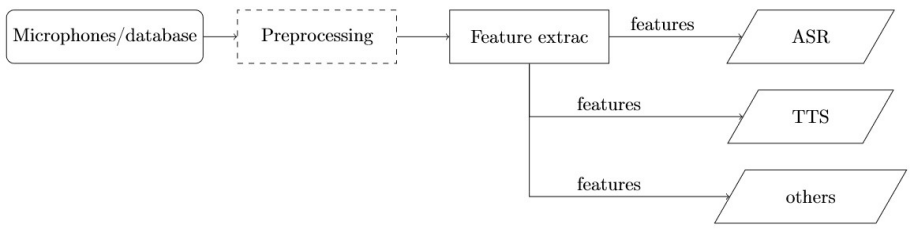


Figure 1: Top structure

features (sometimes preprocessing as well). Meanwhile, through the analysis of the features extracted with a linguistic unit in each frame, the result could be used for improving TTS as well.

The goal of the project is to build a new acoustic model used in ASR system based on neural network. And in the case, compared with mainstreaming acoustics module, the most significant difference is that introduced autoencoder makes the training becoming semi-supervised learning from supervised learning.

Feature extraction usually like Mel-frequency cpectrum coefficient (MFCC), Linear predictive coding (LPC), Warped Minimum Variance Distortionless response cepstral coefficients (WMVDRCC) [9, Chapter 7] are based on standard digital signal processing while neutral network-based feature extraction is uncommon.

And Deep Neural Network (DNN) has been proved as a efficient ASR improving [7]. [10] used the DNN module as acoustic module in ASR system. Meanwhile, the acoustic module could built with Convolution Neural Network(CNN) [11], [12], [13], [14] [15] or Long short-term memory recurrent

neural network(LSTM) [16]. Zhang [17] reachs the highest recognition rate on 96.04% LibriSpeech [18] with acoustics module, Deep improved Feedforward Sequential Memory Networks (DFSMN).

In other DNN speech application methods, [19] used DNN to realize speech enhancement in features' level. [20] introduced denoise autoencoder in (DAE) feature level for speech enhancement. [21] also used DAE for dereverberation as well. There also a some research about using autoencoder in speech motion emotion recognition area, like [22], [23].

[24] make a big breakthrough on Natural Language Processing (NLP) with an attention mechanism. In the speech field, as a natural extension, there is an attention mechanism in [25] as well. With popularization of end-to-end speech recognition [26], attention mechanism also plays a crucial import role in joint CTC/attention architecture [27] [5].

The mainly innovation points are listed above:

1. **semi-supervised autoencoder learning**: how to achieve semi-supervised learning structure with an autoencoder.

2. **hybird-attention**: As the whole acoustics module is divided as different part, different dimension attention mechanisms are performed in each section.

3. **dynamic loss combine in semi-supervised learning**: There are several different loss in semi-supervised autoencoder learning structure, combined dynamically.

4. **Dynamic loss combing weight**: viewing loss combine weight as a training parameter, not hyper-parameter and adding as a regularization item.

2 THEORETICAL CONCEPTS FOR AUTOMATIC SPEECH RECOGNITION AND DEEP LEARING

This Theoretical chapter will give a crucial part of the background of automatic speech recognition (ASR) and its some typical working architecture in order to understanding in how speech signal is model and recognized in the system.

The chapter is divided into four sections, first section 2.1 will introduce some understanding of how human generate and precept speech also how these generation and perceptions are modeled. section 2.2 is an introduction of some basic concept about computing modelling sequence and speech signal. Then section 2.3 introduce the recent successful discriminative model, deep neural network (DNN). Then section 2.4 is described some conventional ASR architectures including Gaussian Mixture Models - Hidden Markov Models (GMM-HMM), Deep neural network - Hidden Markov Models (HMM-DNN) and Joint Connectionist Temporal Classification-Attention (CTC-ATT).

2.1 Basic concept about speech and linguistics

2.1.1 Speech signal

In physics, a sound signal is mechanical pressure wave formed of vibration of the medium could be solid, liquid or air [28]. In mostly causes in the speech field, a speech signal is a pressure wave propagating in air. In original cases, as the base as communication and interaction, the main usage of speech is express by one person and reception by others. The flow graph of speech's generation, propagation, and understanding are drawn in figure 2.

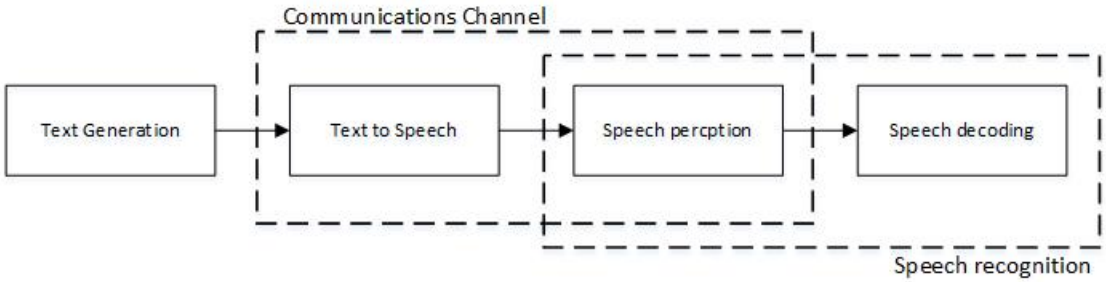


Figure 2: Flow graph of speech's generation, propagation and understanding

As described in the figure, speech is generated based on a specific text and propagated to someone

else. Then received speech signal will be proposed and decoded into text. So, primary analysis of a speech signal is based on its generation, propagation and perception.

Generation of speech signal

The main speech generation model used is the source-filter model. However, before introducing this model, an understanding of how humans generate speech is necessary. It also is reviewed as the basis of text-speech-generation. In figure 3, it is a clear description example of human's vocal apparatus and its corresponding alphabet symmetric relation in the Russian language.

As mention above the speech signal is air-pressure in the air. Compared with other acoustics

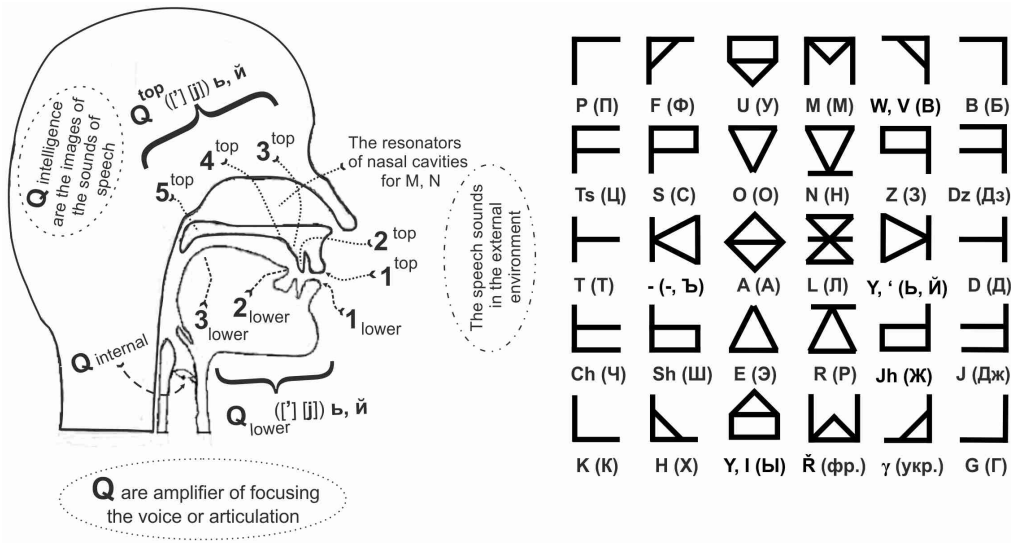


Figure 3: Vocal Apparatus with alphabet symmetric, freely reused from *wikimedia, File:Vocal apparatus and alphabet Symmetric by Makeyev.jpg*

signals, the unique is it emanated from the mouth and nostrils of a human speaker [6, Chapter 2]. In most language, the inventory of *phonemes*, talked later in subsection 2.1.3. These phonemes could be classified into two basic classes:

1. **Consonants**: the presence of constrained by the throat or obstruction in the mouth (tongue, teeth, lips) when air flow from lung as speaking.
2. **vowels**: there is not evident constrain during the speaking.

These constrain description could be verification in figure 3 as well especially with the position of each part.

For further classification into linguistics subunit, these properties derive from anatomy of a handful of essential articulators and the boundaries place of the human vocal tract. Besides, the whole

speech process is contributed by thousands of muscle's moving and shrink.

Classic source-filter model

To simplify the vocal apparatus model into a source-filter model, with input excitation signal $e[n]$ and output speech signal $s[n]$ showed in figure 4.

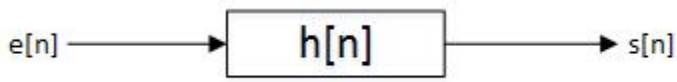


Figure 4: Source-filter model

In the figure, $h[n]$ represent an all-pole system function of vocal apparatus. Except for the silence status, the sound here could be grouped as two: *Unvoiced sounds* and *Voiced sounds*. The representative difference between them could be explained from whether vocal cords tense and vibrate. When vocal cords are tensed and vibrated periodically, a quasi-periodic speech waveform is emitted from a mouth (quasi-periodic waveform: stationary over short periods, but changes occasionally to a new mode of stationarity [29]). Moreover, this description is drawn in figure 5.

In the figure, it is clear that there are two types of sound, unvoiced sound and voiced sound.

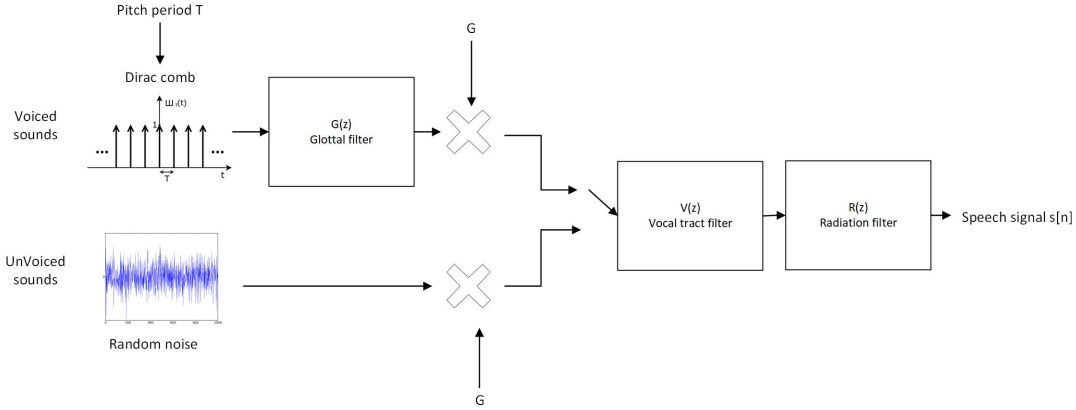


Figure 5: Unvoiced and voiced source-filter model for speech production

Both could be gain by a gain function G. The difference is that the voiced sound is passed through glottal filter $G(z)$ before the gain control. Moreover, the source of these two respectively is Dirac comb and random noise. Then gain controlled sound will be fed into vocal tract filter $V(z)$, radiation filter orderly $R(z)$.

For simplification, Glottal $G(z)$, vocal $V(z)$and radiation filter $R(z)$could be integrated as a all-pole filter as mentioned above, So this integrated function could be rewritten as [6]:

$$H(Z) \;=\; G(z)V(Z)R(Z) \;=\; \frac{G}{1-\sum_{i=1}^{p} a_i z^{-i}} \tag{2.1}$$

So as the system function $H(z)$, the relation between of excitation and speech signal is written as [6]:

$$s[n] \;=\; \sum_{i=1}^{p} a_i s(n-i) \;+\; Ge[n] \tag{2.2}$$

Perception of speech signal

In this paragraph, something about auditory perception system and its nonlinear system response will be present for a better understanding of human hearing. Further details will not be present since these are not directly relevant to this master thesis.

Mainly, the auditory system could be split to two-part based its propagation property:

- **Mechanical wave propagation part (Ear)**
- **Auditory nervous part (Brain)**

The human auditory system's anatomy is draw in figure 6.

When a speech signal reaches human auricle, this air vibration received by auricle and propagated through *auditory canal* which could be reviewed as acoustics filter increasing the effect of sound in $3-4\ Khz$. Then it will be transformed into mechanical acoustics vibration by *eardrum* keeping its frequency. During propagation this vibration to *cochlea* through *malleus*, the amplitude of mechanical vibration will be amplified via malleus.

With vibration finally transformed into a nervous signal in *cochlea*'s basilar membrane as these structures showed in figure 7. It is clear that signal with different frequency propagates with different basilar channels. That will result in a different response. Then the system function could be seen as a filter bank. Obviously, the final nerve signal will drive a representation into the brain as an electronical signal.

Psychological acoustics and Mel scale

Psychological acoustics is a kind of field to study the distinction between the perceptual attribute of sound and physical attribute of sound [6]. Also, even some perceptual merits are strongly related to other physical merits showed in the table 1.

Even there is still some mismatches between two merits, but these mismatches would not be further talk in this master thesis and will be ignored here.

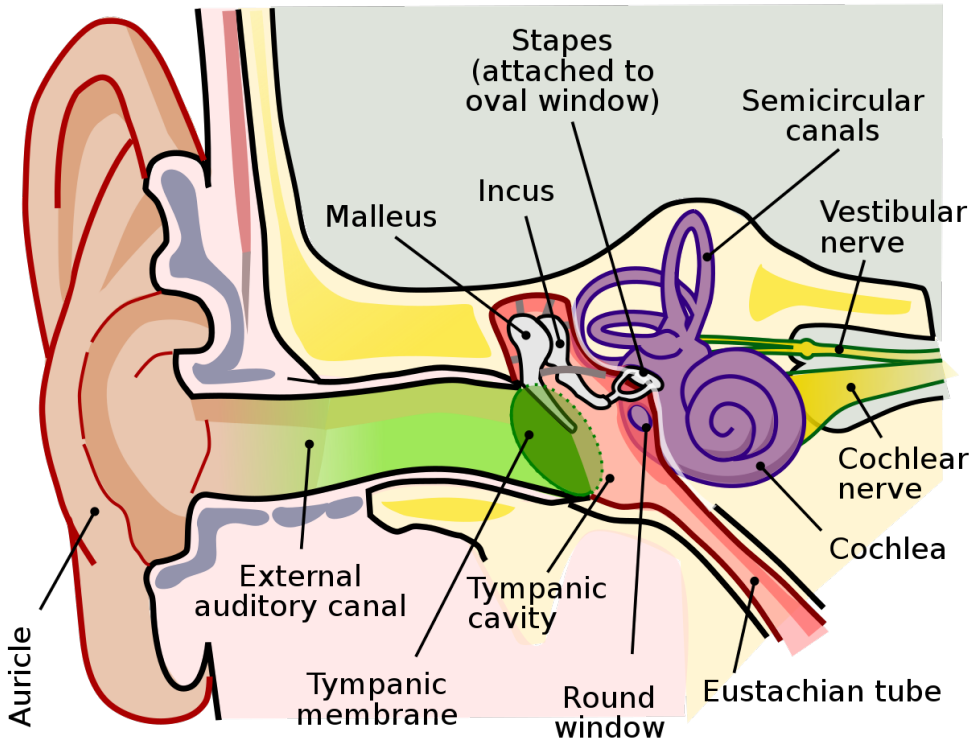


Figure 6: Anatomy of the Human auditory system, freely reused from *wikimedia, File:Anatomy of the Human Ear.svg*

Physical merit	**Perceptual merit**
Intensity	Loudness
Fundamental frequency	Pitch
Spectral shape	Timbre
Onset/Offset time	Timing
Phase difference in binaural hearing	Location

Table 1: Connection between physical and perception metric [6]

As described in the table 1, sound loudness mostly times could be think as same with sound wave intensity. However, ear sensitivity varies with incoming waves' frequency. So, this divergence could be called the phenomenon of non-uniform *equal loudness* perception of tones of varying frequencies. General speaking, tons with different pitch have different inherent *perceived loudness*. In ISO 226 [30], the graph of equal loudness contours defined, draw in figure 8.

It should be noted that the curve value of equal loudness in 1 KHz always equals its intensity.

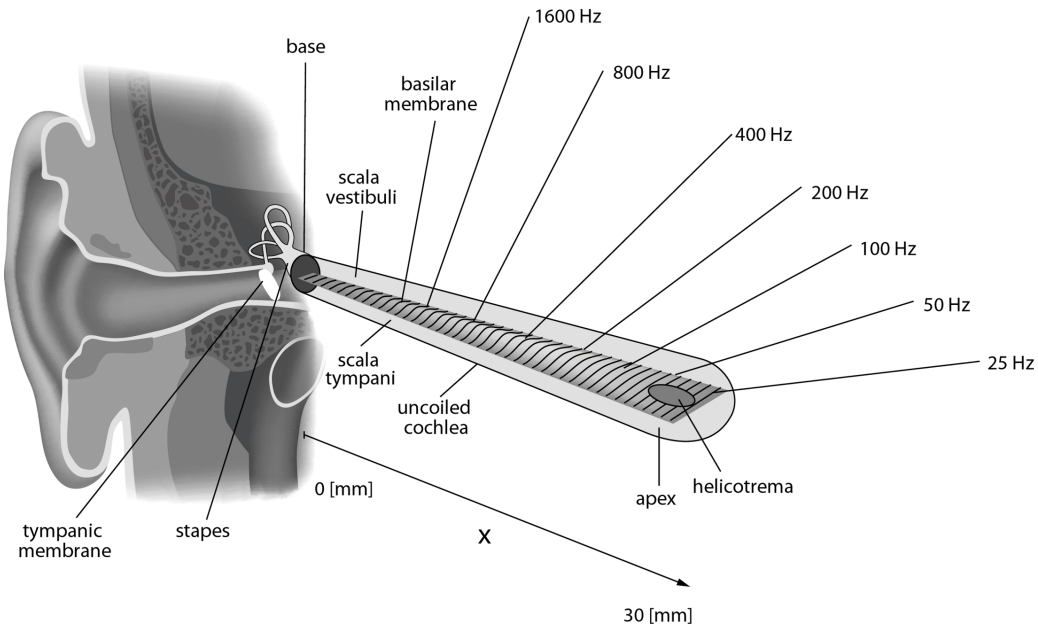


Figure 7: Uncoiled cochlea with the basilar membrane, freely reused from *wikimedia, File: Uncoiled cochlea with basilar membrane.png*

Hearing sensitivity have a low response at low frequency, and it climbs with increasing frequency until around 4 KHz reaching the maximum response. Also, it shows decreasing with frequency up to 10 KHz. So it is clear say that the relation between physical intensity with perception equal loudness is a nonlinear response.

For simulating this nonlinear response, Fletcher (1940) at the first time introduced a term *critical band*, which is inspired by exist cochlear response. After that, other experiments about investigation crucial band phenomena and its bandwidth were carried out. There are two usual scale nowadays: *Bark scale* and *Mel scale* [6].

Mel scale is a perceptual motivated scale, almost linear below 1 KHz and logarithm above, which could be represented in equation 2.3 and figure 9:

$$B(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.3}$$

in figure 9, it is clear that below 1 KHz the curve keeps linearity and above it grows slowly. Also, 1 KHz is the defining point in equal loudness curves.

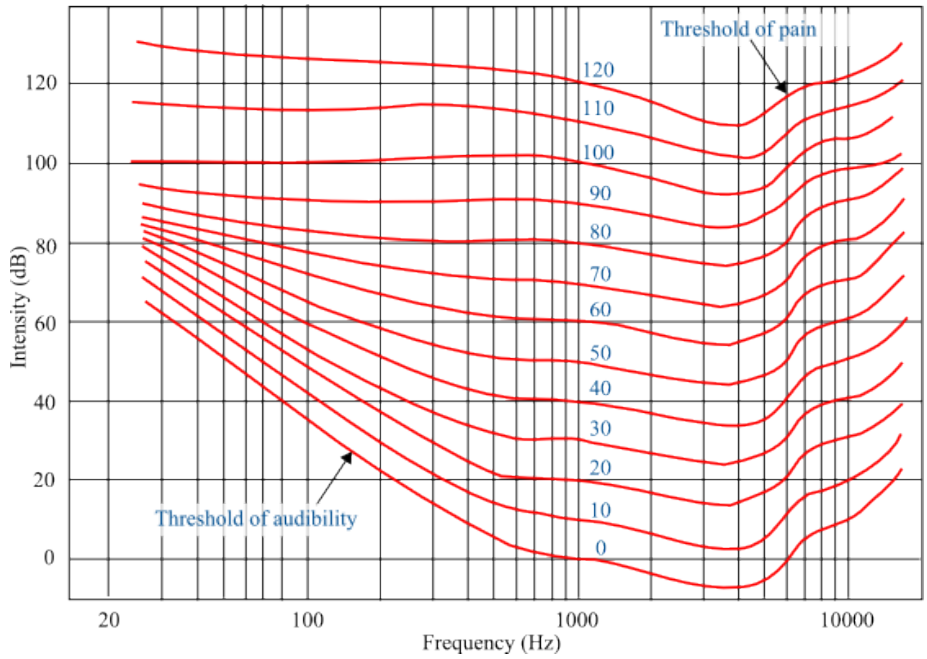


Figure 8: equal loudness contours, freely reused from *wikimedia, File:FletcherMunson ELC.png*

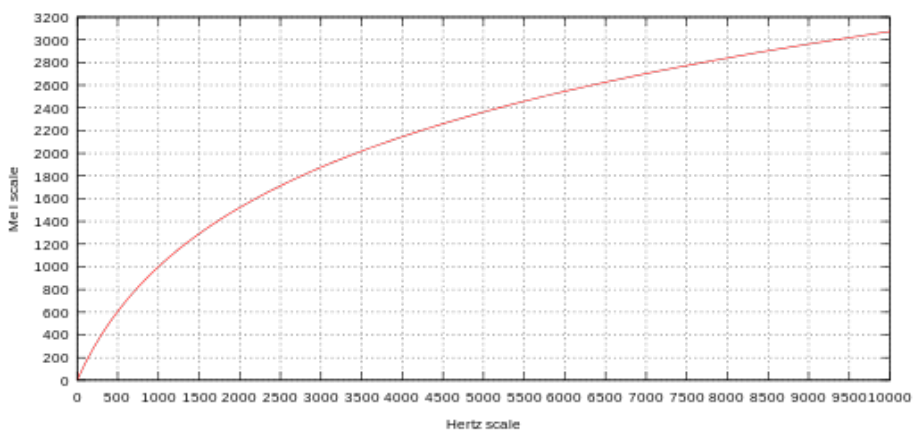


Figure 9: Mel scale, freely reused from *wikimedia, File:Mel-Hz plot.svg*

2.1.2 Short-time Fourier Analysis (STFT) and Mel filter bank

Compared with image, video signal and other higher dimensional signal, speech or audio signal is an only one-dimension signal in conventional view. Except for the temporal time scale, there is another inner characteristic. In a few decades, speech signals tend to be view as a two-dimensional signal [7, Chapter 3.6]. The one more dimension in this "new point" is the frequency, which is the

new representation of one-dimension data could not be described. The one more dimension means that the ASR system could capture more variety in frequency.

For a better using this property, speech signal, 1D time-series signal, should be transformed into a 2D time-frequency signal before further processing. One of a famous method of transformation is Short-time Fourier Analysis. An example of a 1D time-series signal and its transformation is drawn in figure 10. The upper image shows the raw signal while lower is its STFT signal, colors representing magnitude in 2D data.

Then, this STFT will be introduced step by step. First, in modern computer-based signal pro-

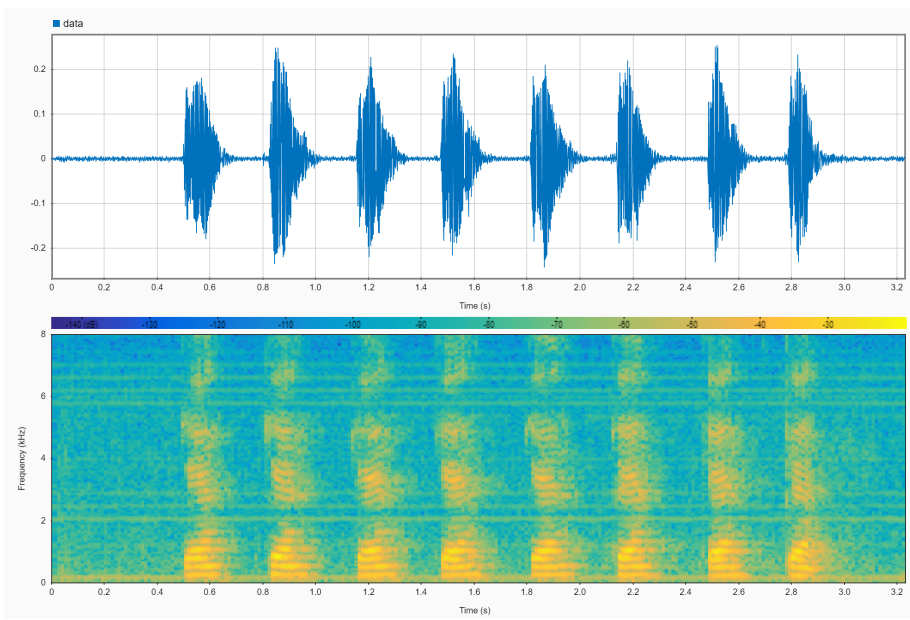


Figure 10: A typical speech signal and is STFT

cessing, STFT talked is discrete STFT. So, a Discrete Fourier Transform based STFT is showed in the equation below:

$$X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.4}$$

where, $x[n]$, $w[n]$ respectively are speech and window signal. Hann and Hamming windows are common used window function in STFT, both showed in the equation:

$$w[n] = a_0 - (1-a_0)\cdot\cos\left(\frac{2\pi n}{V}\right), \quad 0 \le n \le N \tag{2.5}$$

If $a_0 = 0.5$, that produce Hann window function. While if a_0 approximate 0.54, or more precisely 25/46, Hamming window function is produced [6]. These two window function and their Fouries

transform are shown in figure 11. More details, figure (a) is Hann window and figure (b) is Hamming window.

$X(m, \omega)$ is 2D short time Fourier analysis, where m and ω respectively represent time and fre-

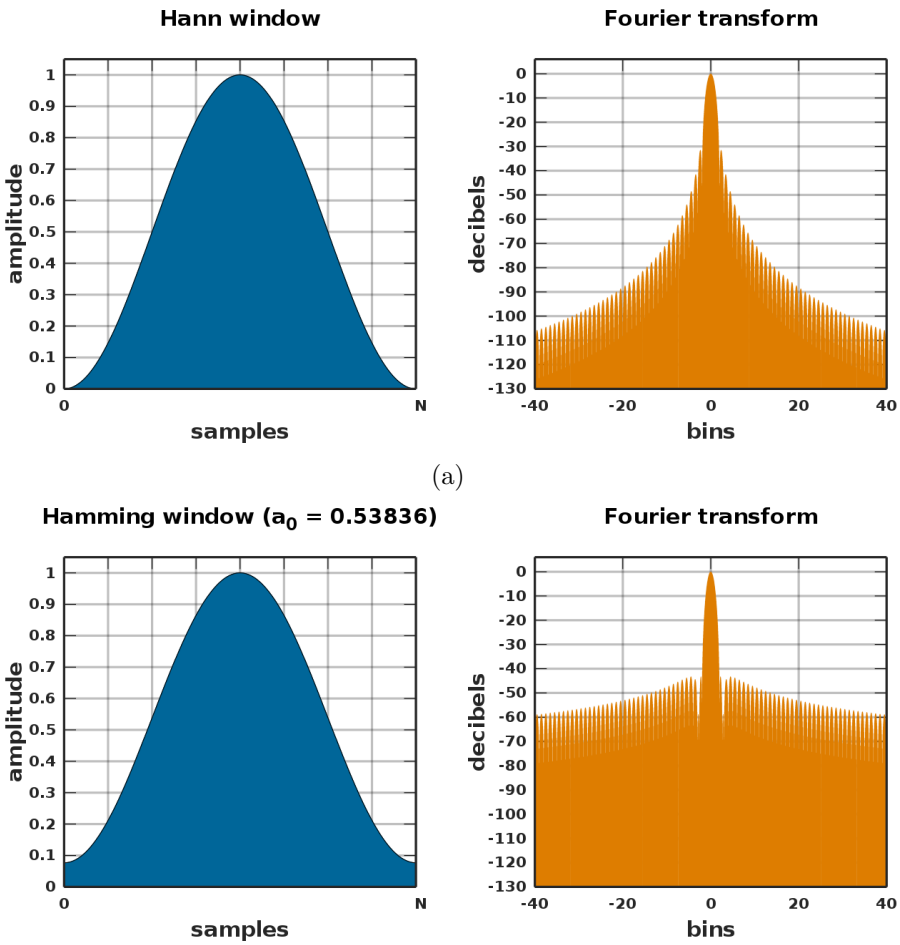


Figure 11: Hann(a) and Hamming(b) window function, freely reused from *wikimedia, File:Window function and frequency response - Hamming (alpha = 0.53836, n = 0...N).svg* and *File:Window function and its Fourier transform – Hann (n = 0...N).svg*

quency dimension. These 2D data will be processed for feature extraction.

In final steps of feature extraction, Mel filter bank (Fbank) is applied to STFT domain as a set of triangular filters, typically 40 filters. Each triangular filter in filter bank is having a maximum

response of 1 at the centre frequency and decrease linearly towards until 0. A mathematics equation and figure are showing in the equation 2.6 and figure 12.

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k-f(m-1)}{f(m)-f(m-1)} & f(m-1) \leq k < f(m) \\ 1 & k = f(m) \\ \frac{f(m+1)-k}{f(m+1)-f(m)} & f(m) < k \leq f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{2.6}$$

where, m represent the mth filter.

And different color in figure 12 represent different filter in the filter bank.

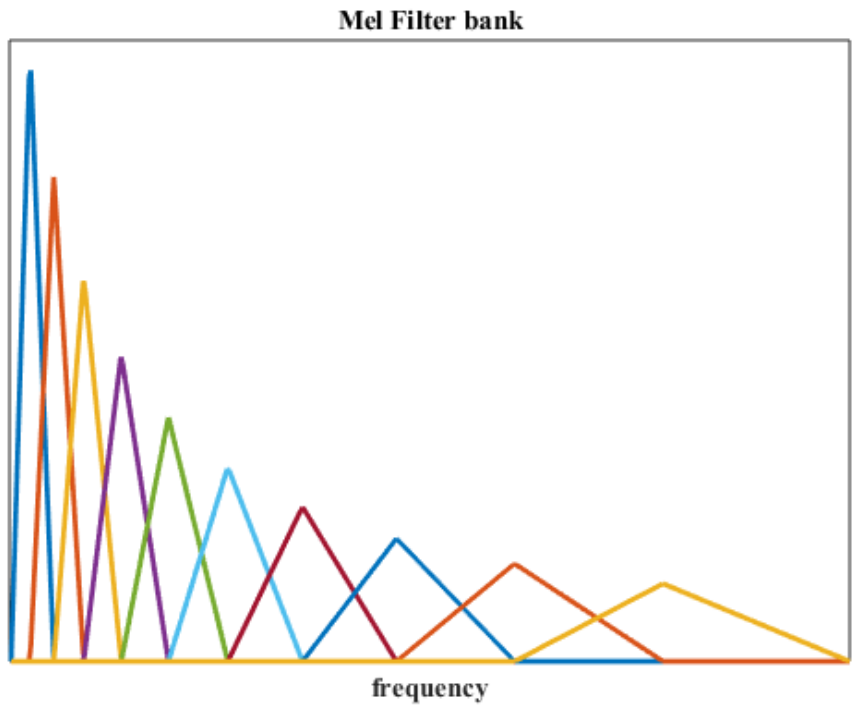


Figure 12: Mel filter banks, freely reused from *wikimedia, File:Mel RMel.png*

2.1.3 Speech phonics

There is an educational debate between "whole language" and "phonics" methods of teaching children reading and listening. In some invented languages, like Chinese and Sumerian, they were mainly logographic (one symbol represented one word). However, in most other languages, these language systems contain more subdivide unit like *syllabic* or *phonemic*, in which, symbols represent the sounds that make up the words [8].

In modern speech and language study, the idea implicit in a sound-based writing system, how the spoken words are composed of sub-units of speech, is call *phonology*. This decomposing ideas indeed also underlies ASR, TTS.

From a computational perspective, *phonetics* is the study of how linguistics sound is produced. Normally, words' pronunciation could be modeled as a sequence of symbols represent *phones*.

ARPAbet is a set of phonetic representation codes developed by Advanced Research Projects Agency (ARPA) developed in the 1970s for American English using ASCII symbol [8]. There is an example of ARPAbet symbols for transcription of English consonants in table 10. In the table, it is clear that word *dill* is composed by three phone symbol represented by ARPAbet symbol *d, ih, l*.

There is an other **International Phonetic Alphabet (IPA)** commonly used. That standard originally is developed by the International Phonetic Association with obtaining transcribe all speech of human language [8, Chapter 7].

2.2 Basic concept of computational model

2.2.1 Gaussian distribution

The most cases, the speech signal is considered as feature representation based random variables signal [6, Chapter 4]. Before introduced Gaussian Mixture Models(GMM) in speech signal, the *Gaussian distribution* will be described firstly.

Until now, Gaussian distribution (also called *Normalization distribution*) is the most important distribution for wide-spread variables in the real physical world. There are two parameters for each continuous random variable X: mean μ and variance σ^2 ($\sigma > 0$). This distribution could be described by equation 2.7:

$$f\left(x|\mu,\sigma^2\right) = N\left(\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{2.7}$$

And some Gaussian distributions with different variance are plotted by Probability Density Function (p.d.f.) in figure 13. Where, p.d.f. represents statistic expression for a continuous random variable. The area under the interval curve in the figure equals the probability of a continuous random variable occurring. Usually, a Gaussian distribution with mean $\mu = 0$ and $\sigma = 1$ denoted as $N(0,1)$ with a specific calling *Standard Gaussian Distribution*.

Also, some Gaussian distributions with different variance are showed at figure 13. Literally, it is apparent in the figure that the median and mean value of Gaussian distribution both equal μ. Moreover, the left side and right side in μ are symmetrical and its distribution probability becomes smaller with distance to μ growing. Another interesting appearance in this figure is with variance

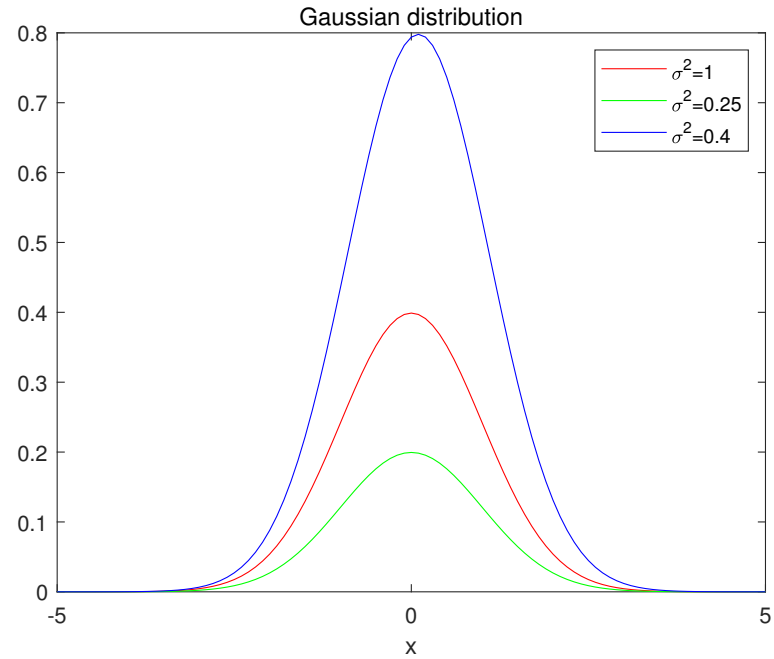


Figure 13: Gaussian distribution with different variance

σ increasing the whole distribution tend to concentrate around more mean μ.

2.2.2 Basic properties of Gaussian distribution

For later on extend single Gaussian distribution into GMM, there are some basic properties of Gaussian distribution should be noted here.

If there is a continuous random variable X with its linear conversion $Y = aX + b$ where a and b both are constant with $a > 0$. The mean and variance of Y are respectively $a\mu + b$ and $b^2\sigma$ [6, Chapter 3].

This linear conversion means that vice versa, all Gaussian distribution could be linearly transformed back into a standard Gaussian distribution as well. Its implement showed in the equation below:

$$X = \frac{Y - \mu}{\sigma} \sim N(0, 1) \tag{2.8}$$

Where, X is a continuous random variable with standard Gaussian distribution and Y is arbitrary Gaussian with mean μ and variance σ.

In next step, co-variance and correlation indexes will be talked. There are two continuous ran-

dom variable X, Y following respective distribution. The expectation of those two variables could noted as $E(X) = \mu_X$, $E(Y) = \mu_Y$ and each variance: $Var(X) = \sigma_X^2$, $Var(Y) = \sigma_Y^2$. If there is a index to describe mutual relation between two variable, denoted as $Cov(X, Y)$ with calling *co-variance* in equation 2.9:

$$\mathrm{Cov}(X, Y) = E\left[(X - \mu_X)(Y - \mu_Y)\right] = \mathrm{Cov}(Y, X) \tag{2.9}$$

In addition, for normalized this co-variance from two variance value, *correlation* ρ_{XY} is introduced. Compared with co-variance, the correlation is a normalized value. Its normalization is showed in the equation 2.10:

$$\rho_{XY} = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{2.10}$$

With these basic properties, the single Gaussian distribution could be expand into GMM in next part.

2.2.3 Gaussian Mixture Models

There is a n-dimensional continuous random variables vector $\mathbf{X} = (X_1, \ldots, X_n)$ whose p.d.f could be represented as the following equation 2.11:

$$f(\mathbf{X} = \mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = N(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right] \tag{2.11}$$

Where μ represent the n-dimensional mean vector of each random variables while σ is an n-dimensional co-variance matrix of each random variables in the vector. These mathematical relations are shown in the equation 2.12:

$$\begin{aligned} \boldsymbol{\mu} &= E(\mathbf{x}) \\ \boldsymbol{\Sigma} &= E\left[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^t\right] \end{aligned} \tag{2.12}$$

If each random variables X_i are independent to each other, the covariance matrix is degenerated to a diagonal matrix in which all off-diagonal matrix elements are zero. If there are only two independent variables in n-dimension variable vector, X, Y respectively, these joint p.d.f. are draw in figure 14.

In the figure, blue and red lines are respectively are p.d.f functions of X and Y. The centre green cycle surround all joint sample.

With multiple unimodal Gaussian distributions, a more complex distribution could be approximated by linear combine multiple Gaussian distributions with equation 2.13:

$$f(\mathbf{x}) = \sum_{k=1}^{K} c_k N_k\left(\mathbf{x}; \mu_k, \boldsymbol{\Sigma}_k\right) \tag{2.13}$$

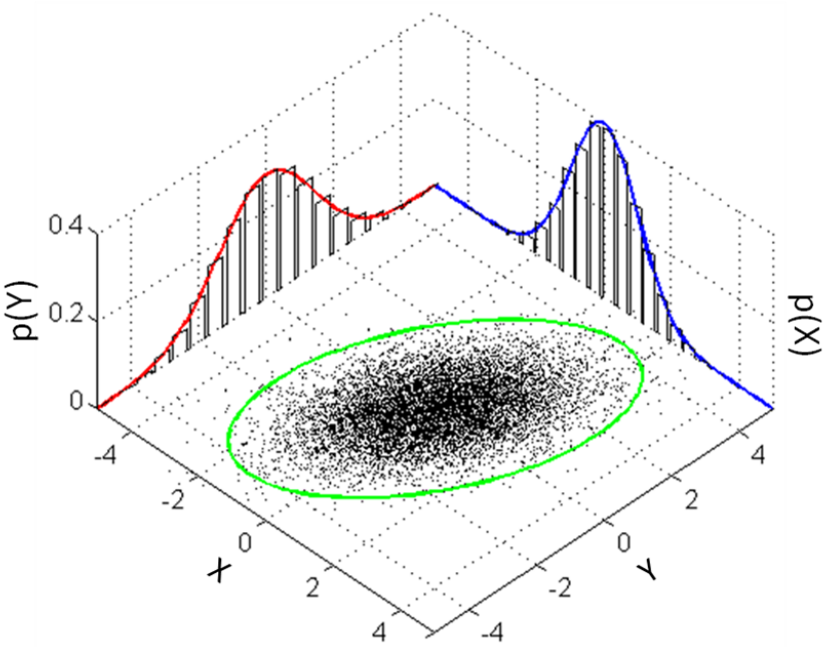


Figure 14: Multi-variance distribution, freely reused from *wikimedia, File:MultivariateNormal.png*

where c_k represent mixture weight for kth component and are subject with the following equation:

$$c_k \geq 0 \quad \text{and} \quad \sum_{k=1}^{K} c_k = 1 \tag{2.14}$$

This Gaussian mixture could approximate arbitrary complex distribution just with enough distribution components.

2.2.4 Hidden Markov Models

One of the most interesting and unique points in the speech signal is its variety in feature sequence length, even some sequences representative same linguistic sequence but still different in length. This unique property is the reason for the speech signal keeping the temporal dimension. The basic reason for this depends on how fast speakers speak and how many break during their speaking. Furthermore, the main cue of modeling and discriminate each linguistic unit is distributed in a temporal length sequence.

The Hidden Markov Models (HMM) is a powerful statistic model to build a discrete hidden states sequence from an observation sequence [6, Chapter 8]. This model offers not only a parametric states transform process and its parametric launch process from each state to observed feature.

There is an assumption of the launch process, from discrete states to sample observed data in HMM is a parametric random process, whose stochastic parameters could be estimated by a precise process for its discrete states. An example of an HMM model could be drawn in figure **??**.

It is clear that in the figure, $x1$ to $x3$ represent three different hidden states and $y1$ to $y3$ respectively

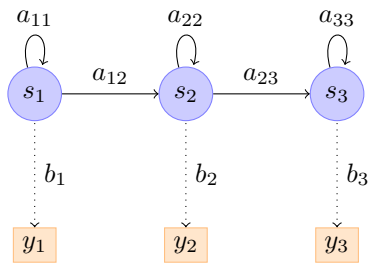


Figure 15: A triphone HMM model

are their launched observed data. Meanwhile, there are some transform and launch parameters as described above, a_{ij} and b_i. More precisely, a_{ij} are the transform parameters from state i to state j, and b_i describe how to launch observed data from each different state i.

2.2.5 Recognition Unit in linguistics

Normally, movement of articulators (tongues, lips velum) during speech are continuous and subject to physical constraints like momentum, which means articulators may start moving during one phone to get into place in time for the next phone. This constraints representative in phone sequence with phone-context-dependent relation [8].

To model phones in different context, most ASR replace original *context-independent (CI) phone* with *context-dependent (CD) phone*. For introduce context-dependent information into ASR, even phone is minimum unit within a word for linguistics speaking, phone still could be divided into sub units: *triphone*. Its means there are three states to represent a phone with *start-this-phone*, *median-this-phone* and *end-this-phone*. This triphone reresentation could be draw in figure 15.

In the figure, A triphone is represented by a HMM model. States $s_1 \sim s_3$ now are *start-this-phone*, *median-this-phone* and *end-this-phone*. And a_{ii} represent is stuck on the i state. Besides, y_i is the acoustic presentation of its state.

2.3 Deep neural network

The deep neural network is a method from Machine learning based on the layers used in artificial neural networks. It plays a crucial role in the modern speech recognition system. It could replace GMM in GMM-HMM system with DNN as DNN-HMM system in final recognition process [7]. And

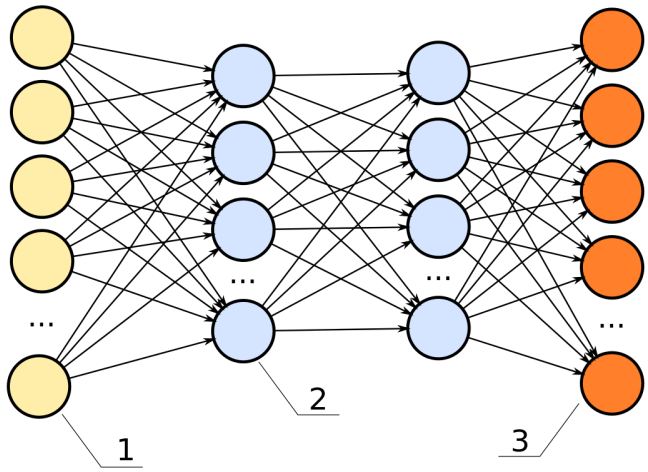


Figure 16: A example of MLP, freely reused from *wikipedia, File:Neural network bottleneck architecture.svg*

Recently with the population of end-to-end ASR architecture, the whole wave-to-word ASR process could be totally realized by DNN without HMM [31] [5] [27] [32]. GMM-HMM, DNN-HMM, Joint Connectionist Temporal Classification-Attention end-to-end Architecture will later are talked in the section 2.4.

2.3.1 Multilayer Perception

Modern first DNN-HMM ASR is implemented by Multilayer perceptron (MLP). Early DNN structure could be review as conventional MLP. And an example of this structure is showed in figure 16.

There are three colors nodes inside of MLP. Yellow, blue and orange respective means input, hidden and output nodes noted with 1, 2 and 3. In this MLP, there are one input layer, two hidden layers and one output layer. For simply notation, assuming in total there are L layers in an MLP, input and output layer are noted as 0 and $L+1$ layer respectively. And each node represents only one value conversion, which means one layer represents one vector conversion whose vector length N_ℓ equals how many nodes in this layer ℓ. Further, typically, the values in the MLP propagate layer by layer. A propagation equation from one layer to the next layer could be described by equation 2.15:

$$\mathbf{v}^\ell = f\left(\mathbf{z}^\ell\right) = f\left(\mathbf{W}^\ell \mathbf{v}^{\ell-1} + \mathbf{b}^\ell\right), \text{ for } 1 < \ell < L \tag{2.15}$$

where,
$\mathbf{z}^\ell = \mathbf{W}^\ell \mathbf{v}^{\ell-1} + \mathbf{b}^\ell \in \mathbb{R}^{N_\ell \times 1}$: the excitation vector. As mentioned above $v^0 = \mathbf{o}$, $v^L = \mathbf{y}$, could

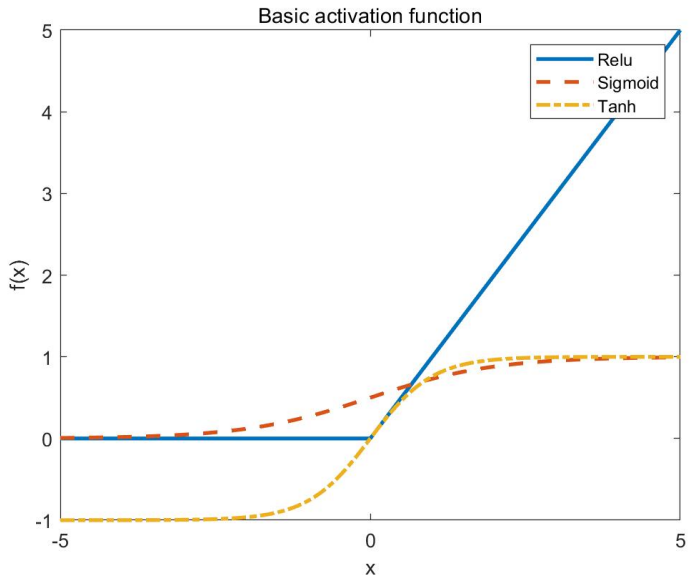


Figure 17: Some common activation function

respectively represent the input observable vector and the output observable vector.
$\mathbf{v}^\ell \in \mathbb{R}^{N_\ell \times 1}$: the activation vector.
$\mathbf{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$: the weight matrix.
$\mathbf{b}^\ell \in \mathbb{R}^{N_\ell \times 1}$: the bias vector.
$N_\ell \in \mathbb{R}$: the number of neurons at layer ℓ.

$f(\cdot)$ is the activation function, in most case nonlinear function. some activate function in common using could be found in the table 2 and drwan in figure 17.

It should be noted that ReLu is most common use activation function and could be reviewed

Name	Math equation
ReLu	$f(x) = max(x, 0)$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Tanh	$f(x) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_i}}$

Table 2: Activation function [7]

as a piece-wise linear function, linear in x greater than 0 and nonlinear in x smaller than 0. In the attention mechanism, *sigmoid* also is a widespread activation function, since its output range

starts from 0 to 1. Besides, the *tanh* could be reviewed as a rescaled version of sigmoid function scaled to from -1 to 1. One of the differences between these two activation function is both even and symmetric but different symmetric point. Different with above 3 activation function, the $Softmax$ function often be used in the last layer for a classification task, usually could be normalized into a probability distribution consisting of N_L probabilities where N_L is the number of neurons is the last layer L. In multi-class task, usually, the N_L equals the number of Class C. The value of the ith output neuron in laster layer v^L represents the posterior probability $P_{DNN}(i|o)$ that the observation vector o belongs that class i.

With propagating layer by layer from observation vector, the final output observation vector V^L could be computed with given parameters W, b. And this process could be called *forward propagation.*

2.3.2 Convolution neural network (CNN)

Convolution neural network original is used for processing grid-like topology data [33] [1]. For instance, 2D discrete time-frequency $X[m, f]$ is a typical topology data. However the 1D discrete time-series signal $x[n]$ could viewed as 2D topology data through STFT or Fbank method introducing time-frequency dimensions. A typical CNN is show in figure 18.

Literally, the core operation of CNN is *convolution* computing which is kind of local linear computing. The forward propagation of *2D convolution layer* has changed from 2.15 to following equation:

$$z_{ij}^{\ell} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab}^{l} v_{(i+a)(j+b)}^{\ell-1} + b_{ij}^{l} \tag{2.16}$$

where v, z are respective are the input and output features of CNN, while ω_{ab}^{l}, b_{ij}^{l} are weight and bias parameters of this convolution layer.

Besides, there is an option named *stride* which usually equal 1. More specifically, the stride is the number of pixels shifts over the input feature. When the stride is one then the filters move pixel by pixel at a time. For instance, when the stride is two then the filters jump 2 pixels at a time and so on.

Usually, there is another common layer in CNN named *pooling layer*, which could reduce the size of features. Moreover, spatial pooling could be seen as downsampling working as dimension reduction and most information retrieve. There are three types of common poolings:

- **Max pooling**: Fragment's maximum as retrieve
- **Average pooling**: Fragment's average as retrieve
- **Sum pooling**: Fragment's sum as retrieve

Since the speech 2D property has been talked in subsection 2.1.2. The speech, image and video

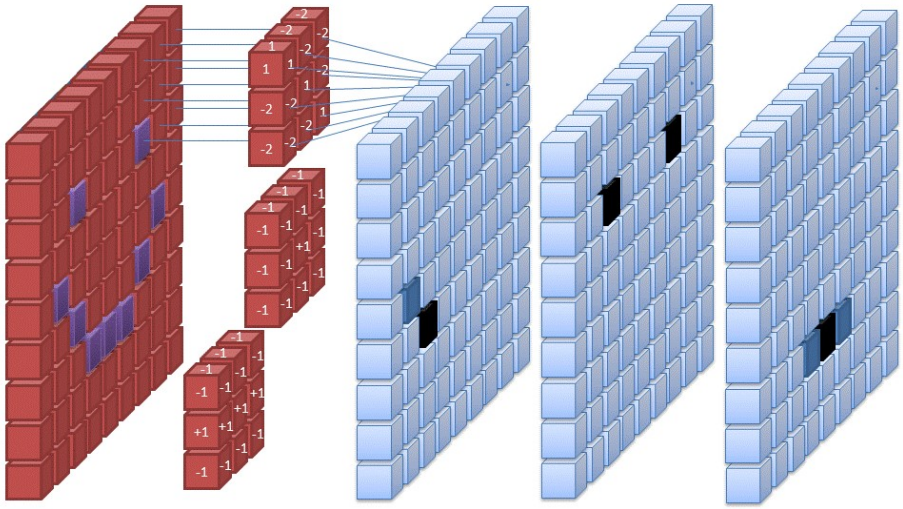


Figure 18: Typical CNN, freely reused from *wikipedia, File:3 filters in a Convolutional Neural Network.gif*

these topology data could be processed by CNN.

2.3.3 Recurrent neural network (RNN)

Recurrent neural network (RNN) is a set of neural networks for processing sequential data, obviously including speech signal. For sequential data, there is an import property, current output not only depend on current input not also past input. That means some memory blocks have to be inserted to memory past information. A basic RNN block has been shown in figure 19.

In the figure, a state vector is stored by memory block and converted from the input and past state vector. So this mathematics equation are shown in the below:

$$\begin{aligned} h^{\ell}(t) &= f_h(W_h^{\ell} v^{\ell}(t) + U_h^{\ell} h(t-1) + b^{\ell}(h)) \\ v^{\ell+1}(t) &= f_y(W_y^{\ell} h^{\ell}(t) + b_y^{\ell}) \end{aligned} \tag{2.17}$$

where, f_h and f_y are two different activation function. W_h^{ℓ}, W_y^{ℓ} and U^{ℓ} are weight parameters matrix.
Also b_y^{ℓ} and b_y^{ℓ} are bias parameter matrix. h_t^{ℓ} is the memory state vector as mentioned.

CNN and RNN have been tremendously successful in practical applications during decades, and there are plenty of books and paper have been talked, and that isn't an innovation in this master

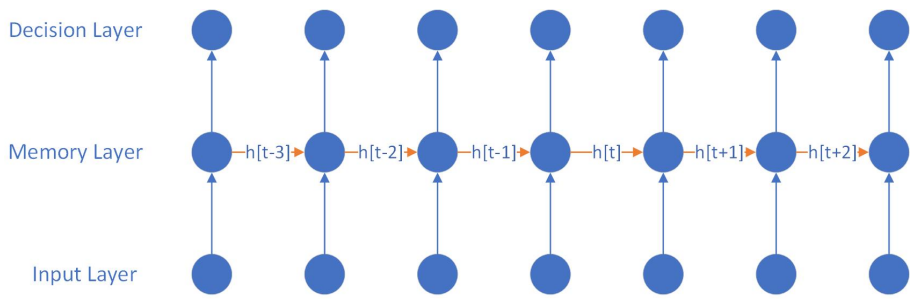


Figure 19: RNN sturcture

thesis. So, this structure would be introduced furthermore.

2.3.4 Loss function

As mentioned before, the observation output vector computation depends on the parameters on the network. Except for proper activations, parameters in the network is essential for forward propagation.

The training process is very complicated and there are many techniques during the training process. But these processes are not the innovation of this master thesis, only basic theorem will be talked.

First of all, the training process is implement by two basic concept: *loss function*, *back propagation*. The loss function is to quantification to show the difference between current output with ideal output. Back propagation will update parameters with a corresponding loss function.

Labelled data or called *training set* as well $\mathbb{S} = \{(\mathbf{o}^m, \mathbf{y}^m) \,|0 \leq m < M\}$ means a pair of ideal input and output data for each task. More precisely, the y^m is the mth ideal corresponding output of each input o^m.

In details, one of loss function mean square error (MSE) loss function could be expressed as:

$$J_{\mathrm{MSE}}(\mathbf{W}, \mathbf{b}; \mathbb{S}) = \frac{1}{M} \sum_{m=1}^{M} J_{\mathrm{MSE}}\left(\mathbf{W}, \mathbf{b}; \mathbf{o}^m, \mathbf{y}^m\right) \tag{2.18}$$

It is commonly use loss function in regression task, where

$$J_{\mathrm{MSE}}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = \frac{1}{2} \left\|\mathbf{v}^L - \mathbf{y}\right\|^2 = \frac{1}{2} \left(\mathbf{v} - \mathbf{y}\right)^{\mathrm{T}} \left(\mathbf{v} - \mathbf{y}\right) \tag{2.19}$$

where, $\mathbf{v}$ is the network's prediction from input $\mathbf{o}$.

For other classification task, there is a common used cross entropy loss function J_{CE}:

$$J_{\mathrm{CE}}(\mathbf{W}, \mathbf{b}; \mathbb{S}) = \frac{1}{M} \sum_{m=1}^{M} J_{\mathrm{CE}}\left(\mathbf{W}, \mathbf{b}; \mathbf{o}^m, \mathbf{y}^m\right) \tag{2.20}$$

where,

$$J_{\mathrm{CE}}(\mathbf{W}, \mathbf{b}; \mathbf{o}, \mathbf{y}) = -\sum_{i=1}^{C} y_i \log v_i \tag{2.21}$$

$y_i = P_{emp}(i|o)$ is the empirical (labeled data in the training set) probability that the input o belongs to class i , and $v_i^L = P_{DNN}(i|o)$ is the corresponding output probability estimated from the network. Minimize this cross-entropy, in statistics, equivalent to minimizing the Kullback-Leibler divergence (KLD) between empirical probability distribution and the probability distribution estimated from the network.

2.3.5 Back propagation

With a given loss, the back-propagation algorithm could update parameters layer-by-layer from the last L layer to the first 1 layer. The base of back propagation is gradient descent computation, which could be represented as the simplest form equation like with parameters $\{W,\ b\}$:

$$\begin{aligned} \mathbf{W}_{t+1}^{\ell} < \mathbf{W}_{\mathrm{t}}^{\ell} - \varepsilon \triangle \mathbf{W}_{t}^{\ell} \\ \mathbf{b}_{t+1}^{\ell} \leftarrow \mathbf{b}_{\mathrm{t}}^{\ell} - \varepsilon \triangle \mathbf{b}_{t}^{\ell} \end{aligned} \tag{2.22}$$

where, W_t^ℓ and b_t^ℓ respective represent the weight matrix and bias vector in ℓ layer at tth time update and ε is a parameter named *learning rate* to control the update rate of parameters. $\nabla_{\mathbf{X}} J$ is the gradient decent of J with regard to x.The gradient decent is computed with equation:

$$\Delta \mathbf{W}_t^{\ell} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{W}_t^{\ell}} J\left(\mathbf{W}, \mathbf{b}; \mathbf{0}^m, \mathbf{y}^m\right) \tag{2.23}$$

and,

$$\Delta \mathbf{b}_t^{\ell} = \frac{1}{M_b} \sum_{m=1}^{M_b} \nabla_{\mathbf{b}_t^{\ell}} J\left(\mathbf{W}, \mathbf{b}; \mathbf{o}^m, \mathbf{y}^m\right) \tag{2.24}$$

where M_b means the number sample in tth update within a batch. Equation 2.23 and 2.24 mean the each updating is based on a average of a batch's gradient descent.

2.3.6 Batch Normalization

The train of parameters needs amounts of data. Usually, the range and distrubution of these data are quite different. In conversational image and speech data processing, there is a classic preprocessing technique named *normalization* which usually could be written as:

$$\tilde{x}_i = \frac{x_i - \mu}{\sigma} \tag{2.25}$$

where μ and σ respectively are mean and deviation of whole or part of data set.

Later on, this normalization technique also is introduced into neural networks as well. *Batch normalization* is one of normalization implement used in DNN.

Literally, batch means normalization is applied within each batch, and the mean and deviation are computed within batch during training [34] like:

$$\begin{aligned} \mu_B &= \frac{1}{n}\sum_{i=1}^{n}(x_i) \\ \sigma_B &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_B)^2 \end{aligned} \tag{2.26}$$

With these parameter, the input feature could be normalized as:

$$x_i' = \frac{x_i-\mu_B}{\sqrt{\sigma_B^2+\varepsilon}} \tag{2.27}$$

Where ε is for a minimum value to avoid data overflow. Besides, μ_B and σ_B both are networks parameters, which means they can only be tunned in training not prediction.

The batch normalization indeed controls distribution difference of input feature to a standard distribution which will further help gradient back propagation. With a more efficient gradient back propagating, a higher learning rate could be applied in training [34].

2.3.7 Regularization item

The regularization item is introduced for *weight decay* [1]. Originally this technique is developed for control overfitting ("the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably" [35]).

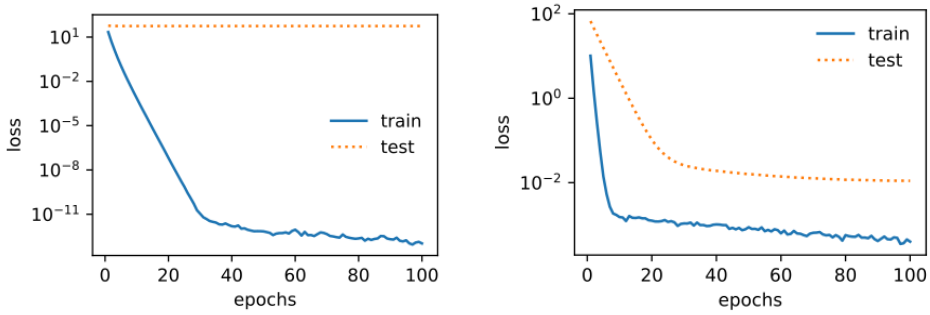
Considered a simple linear function $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}$ with weight vector $\mathbf{w}$, there is a method to keep it small, adding norm as a regularization item (or penalty item)to the problem of minimizing the loss. With the label pair data x^m, y^m, the MSE loss function 2.19 could be rewritten with regularization item:

$$J'_{\mathrm{MSE}}(\mathbf{W},\mathbf{b};\mathbf{o},\mathbf{y}) = \frac{1}{2}\left\|\mathbf{v}^L-\mathbf{y}\right\|^2 = \frac{1}{2}\left(\mathbf{v}^L-\mathbf{y}\right)^{\mathrm{T}}\left(\mathbf{v}^L-\mathbf{y}\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \tag{2.28}$$

The red part in the equation is an added regularization item. It is clear this extra regularization item will push weights to a small value compared with origin through a greater loss value.

There is a example figure 20 to show a training with and without regularization item [1].

In the figure, it is clear, this regularization technique does help ease overfitting. Compared with



(a) (b)

Figure 20: Comparsion of a example of without (a) and with (b) squared norm regularization [1], freely reuse with license Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License https://creativecommons.org/licenses/by-nc-sa/4.0/

the original, the testset's loss decrease not maintain.

And something should be noted that the regularization example in equation 2.28 added a squared norm regularization. Except for this squared regularization, other dimension regularizations like linear or high-dimensional linear regression are common methods as well.

2.3.8 Dropout

Dropout is a advanced regularization method for avoid over fitting [36].

Basically, during training the dropout works like part of neurons on a particular are deactivated. Meanwhile, predictions of left neurons will be amplified. This process shows like:

$$v^l = \begin{cases} 0 & \text{with probability } p \\ \frac{v^l}{1-p} & \text{otherwise} \end{cases} \tag{2.29}$$

Meanwhile, the dropout mechanism working with MLP could be shown in figure 21.

In the figure, x_i, h_i and o_i are respectively input weight and output of each neuron i. It is clear that some of the neurones are blocked by dropout. Since it is a random block, the blocked neurons are different at each time. This stochastic process could be reviewed as random training multi-models which are similar to the original model. These multi-models training could significantly improve generalisation ability.

Something should be pointed out is that since it is a regularization method, it only works during training not prediction.

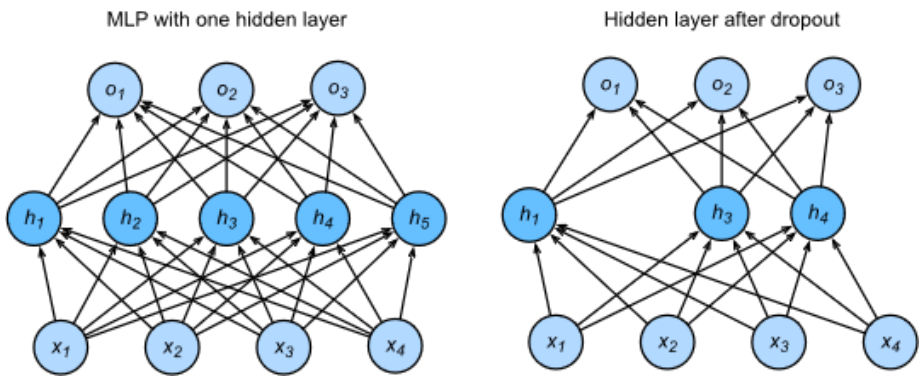


Figure 21: Dropout [1], freely reuse with license Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License https://creativecommons.org/licenses/by-nc-sa/4.0/

2.3.9 Autoencoder

Basic idea

An autoencoder is *unsupervised learning* neural network and trained to attempt to copy input to output showed in figure 22. Usually, it has a hidden layer h that describes a *code* z used to represent the input internally. So, the whole network could divided two *encode*, *decode*. These two respectively represent the process from input to code and the process from code to output noted as:

$$\begin{aligned} z &= f_\phi(x) \\ x' &= g_\theta(x) \end{aligned} \tag{2.30}$$

where, the f and g functions are encoded and decoded function. ϕ and θ are respective are parameters in encoder and decoder as well.

During the training, the labelled data is generated by copy input into the output. That is the reason the training is called unsupervised learning. There is no extra label needed indeed.

And if autoencoder could represent all $g(f(x)) = x$ ideally, it could be used for an unsupervised feature representative learning and compress data into a small dimension. When the dimension of code is smaller than input vector, the code layer h normally is call *bottleneck* layer.

Variational Autoencoder (VAE)

First of all, the two posterior $q_\phi(z|x)$ and $p_\theta(x|z)$ respective describe two posterior probability from input to code and from code to input. Assume there is a prior probability for code z noted as $p(z)$, the loss variational autoencoder could be rewritten by the following equation:

$$\mathcal{L}_{\mathrm{VAE}}(\theta, \phi) = \mathbb{E}_{\hat{p}(x)} \left[\mathbb{E}_{q_\phi(z|x)} \left[-\log p_\theta(x|z) \right] + \mathbb{E}_{\hat{p}(x)} \left[D_{\mathrm{KL}} \left(q_\phi(z|x) \| p(z) \right) \right] \right. \tag{2.31}$$

The first term in equation 2.31 is the reconstruction error also used for autoencoder loss, and the

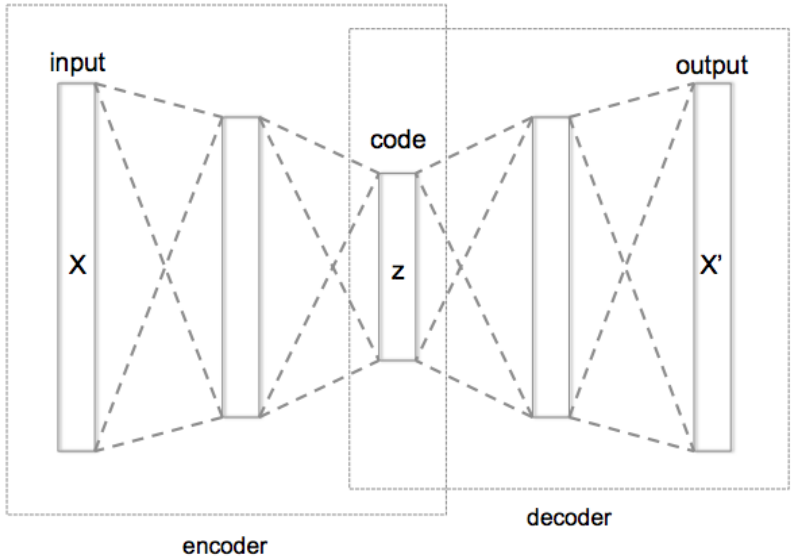


Figure 22: A typical autoencoder, freely reused from *wikimedia, File: Autoencoder structure.png*

second term is the KLD between priory probability with posterior probability. The KLD between two probabilities p_x and p_y definition could be written as:

$$D_f\left(p_x \| p_y\right) = \int f\left(\frac{p_x(x)}{p_y(x)}\right) p_y(x) dx \tag{2.32}$$

So, expect the normal autoencoder loss, there is more other KLD loss is added into VAE as a regularization term. This regularization will force the posterior probability approach to the set priory probability.

If the prior probability is Gaussian distribution (in most cases), the network structure could be implemented in figure 23. The encoder of VAE will computer a vector of mean μ and a vector of variance σ. These two vectors will be used for not only computing KLD later but also fed to decoder of VAE using *reparameterization trick* [37]:

$$z = \mu + \sigma\epsilon \tag{2.33}$$

where z is a code vector computed from reparameterization trick with μ and σ. Since in VAE normally priory probability is set with Gaussian distribution, ϵ usually is an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, 1)$.

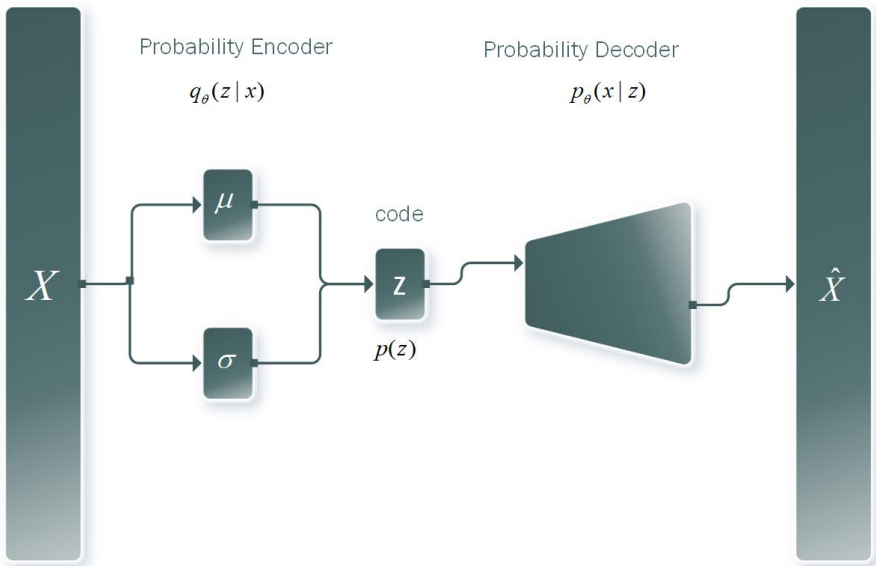


Figure 23: common variational autoencoder structure

2.3.10 Residual network

The study of short connection starts from [38]. Originally it works for the acceleration of gradient back propagation. Then, Raiko, Tapani and Valpola, Harri and LeCun, Yann give details of how it influence networks [39]. In this study, short connection indeed improves image classification accuracy. In 2015, Srivastava R K, Greff K, Schmidhuber J introduced a residual network and highway network extended from short connection, they borrow *gate control* idea from LSTM structure used in highway network [40]. But later in 2016, He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian extend resident network [41]. An example of a residual network is showed in figure 24.

In the figure, a short connection from layer $l-2$ to l is a typical short connection. During gradient back propagation, the gradient could propagate from l to $l-2$ directly. This type of short connection could help training very deep network significantly especially when short connection strides much more layers.

In mathematics, the strided layers' function (for example, from $(l-2) \sim (l-1) \sim (l)$) could be written as $f(x)$ and short connection is written as x. The whole residual network's function $g(x)$ could be rewritten as:

$$g(x) = f(x) + x \tag{2.34}$$

Compared with the original equation $\tilde{g}(x) = \tilde{f}(x)$, the new $f(x)$ tend to express original resident.

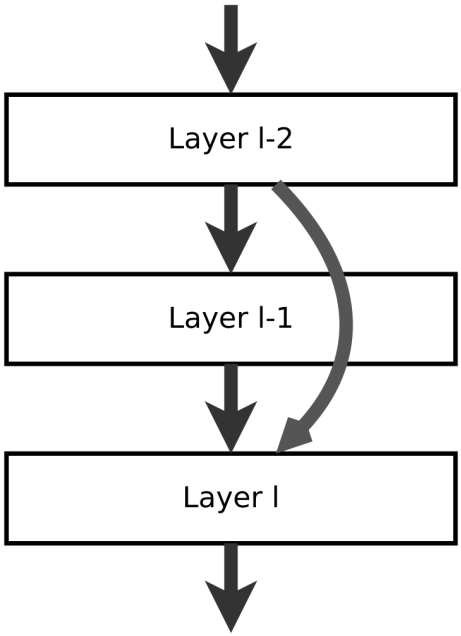


Figure 24: Residual network, freely reused from *wikimedia, File: ResNets.svg*

2.3.11 Attention mechanism

"Attention" is defined as the "how to direct of mind of an object", about choice. For human persons, people may make choices about how to direct their attention when they look at a garden to find a specific flower.

[42] initially introduced attention mechanism in neural machine translation application. Later on, it becomes more and more popular in common encoder-decoder RNN architecture, especially for common Natural Language Process (NLP) field. Later on, it has been introduced to ASR field as well [25] since ASR actually is a wave sequence to letter sequence task. Moreover, then [13] implement a CNN with attention mechanism for ASR task. More specifically, attention mechanism is applied at time and frequency dimension.

Basic attention

Attention mechanisms in neural networks serve to orient perception as well as memory access (could be a subset of whole memory as well). It filters and weights the perception stored in memory.

Before introducing this attention mechanism, the basic concept of RNN and its structure have been talked and show in figure 19. It is clear that RNNs cram everything they know about a sequence of data elements into the final hidden state of the network. However, the attention mechanism takes

input from several time steps into account to make one prediction. And this mechanism is shown in figure 25.

Further, if these several time steps are weighted by a layer named *attention layer* during prediction,

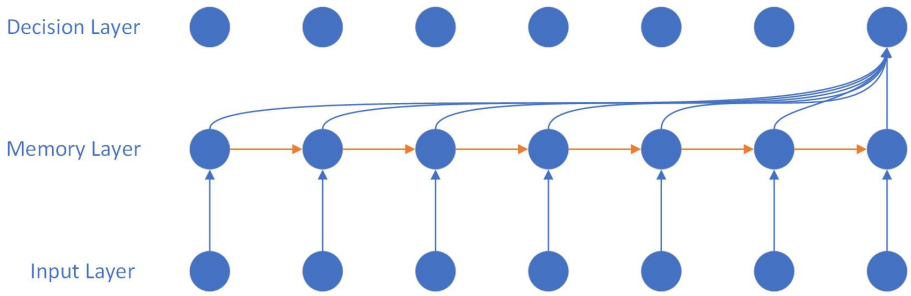


Figure 25: Attention structure

showing in figure 26, in neural networks, it is called *attention primarily serves as a memory-access mechanism.*

In the figure, green layer *attentionlayer* weights hidden vector.

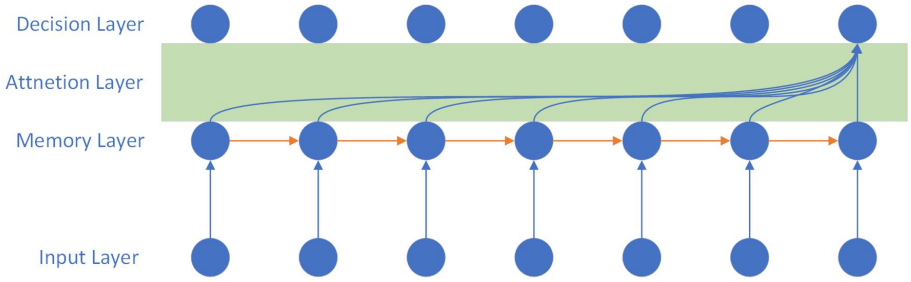


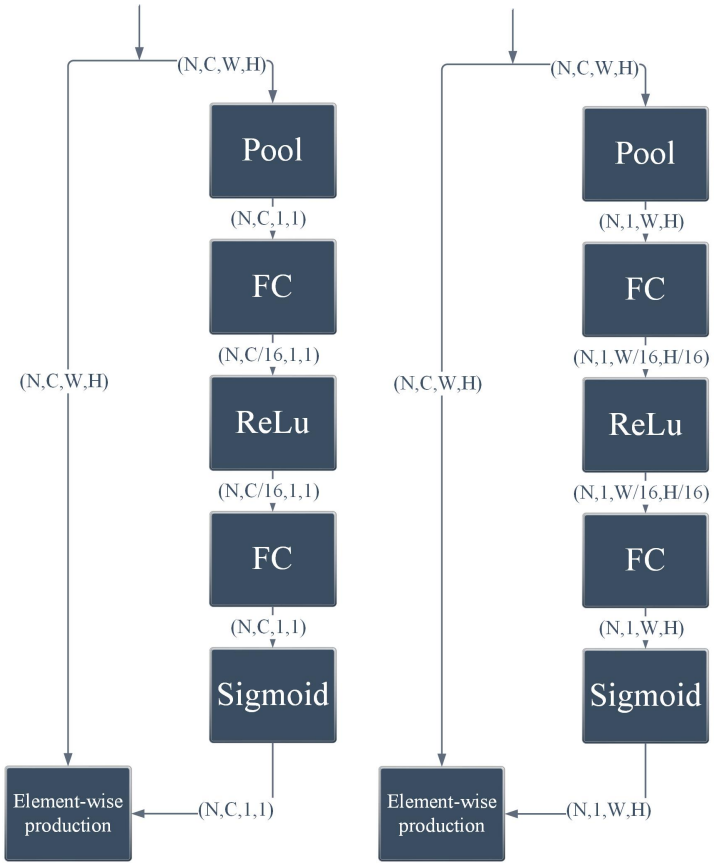
Figure 26: Widely use attention structure

Squeeze-and-Excitation Networks (SEnet)

Original used in image classification, squeeze-and-excitation and its variety are published in [43] shown in figure 27.

In speech application, as mentioned in subsection 2.3.2, speech could be reviewed as 2D data. If a channel dimension is added prior to the time-frequency dimension as channel-time-frequency, this data structure is equivalent to pictures' channel-width-hight data structure. These data structure could be seen in figure 27 as well.

Compared with other attention mechanisms [13], there are three unique points:



(a) Channel attention (b) Time-frequency attention

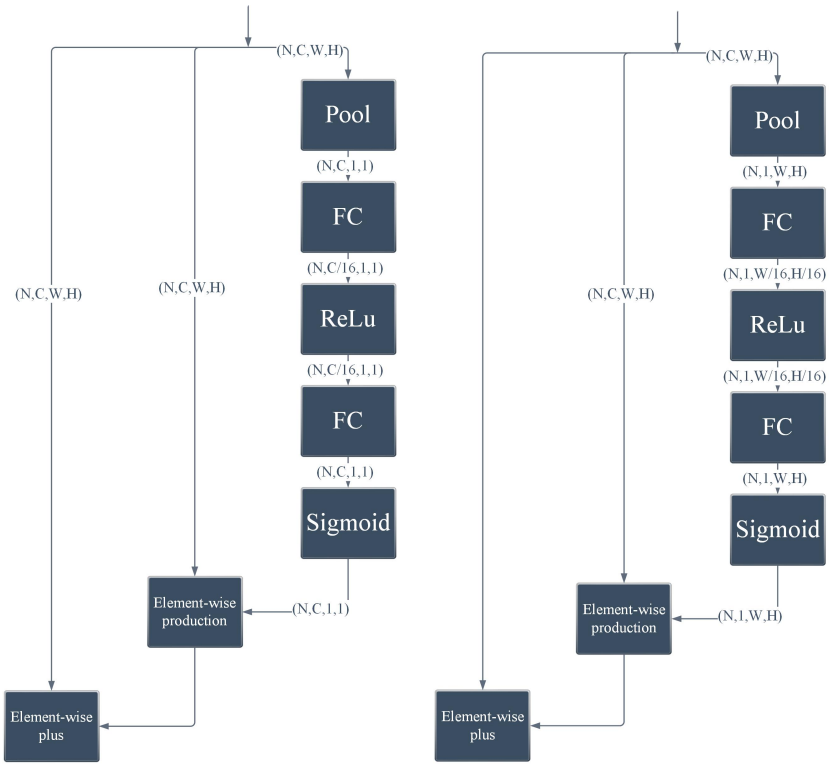
Figure 27: Two basic Squeeze-and-Excitation attention which could be used in this case
Pool:Average pooling layer, FC: Fully connect layer

1. **Self computing attention vector**: The weight vector is not set as trainable parameters but parameters transformed from a hidden vector. And then this weight vector weights its source hidden vector.
2. **Attention working dimension**: Usually, in image data or image-like speech data, attention mechanism is working on time and frequency dimension, but in SEnet, attention mechanism is working on channel dimension which means the network could more focus on the difference of each channel, not time or frequency.
3. **Attention vector excitation mechanism**: During this weight vector transformation, the dimension of the vector will be squeeze and excitation. in figure 27, the role of two Fully connect layer playing in SEnet is dimension alternation.

As [13] mentioned, the squeeze-and-excitation process could be reviewed as a *global information embedding* and *adaptive re-calibration*. The main idea of these squeeze process coming from tackle the inner dependencies of each channel but excitation just rebuild this attention vector to its original form.

If there is no squeeze "/16" process, the attention mechanism could be called *Self-attention*. Obviously, without the squeeze process, there is only one fully connected layer, fewer parameters.

Combining SEnet with a residual network, a residual SEnet commonly used in image classification as well. An application in the speech field is showed in figure 28



(a)
Res-channel attention

(b)
Res-time-frequency attention

Figure 28: Two residual Squeeze-and-Excitation attention which could be used in this case

2.3.12 Encoder-Decoder Architecture for variable length data

It is a common method to dealing with inequality of length in input and prediction sequence for instance in machine translation, video captioning, and ASR.

During this subsection, a new-introduced encoder-decoder architecture is used for dealing with variable length data. The conventional encoder-decoder architecture dealing with fixed length data could be review as an autoencoder has been introduced in subsection 2.3.9.

At the first time introducing the encoder-decoder architecture [44], Google implements a new automatic translation system. The automatic translation implements a mapping from one sequence to another correlation based on linguistics. So this architecture could be used in ASR well. And there is a simple encoder-decoder architecture showed in figure 29.

In the figure, inputs x_t is fed to encoder RNN which model all input as a *encoder vector*. This

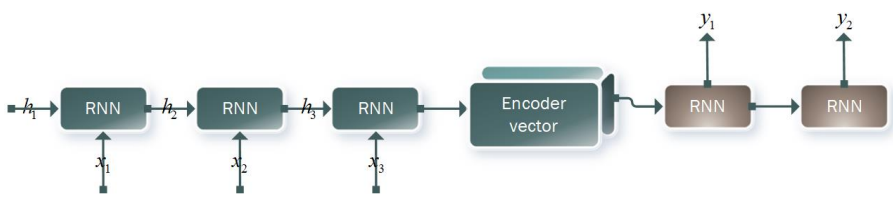


Figure 29: encoder-decoder architecture

encoder vector is resolved by decoder RNN. The decoding sequence's length could be adjusted and equals the expected output length.

2.3.13 Connectionist Temporal Classification (CTC)

Compared with encoder-decoder architecture mentioned in the subsection 2.3.12, Connectionist Temporal Classification (CTC) is another method to dealing with inequality length of feature and prediction and it does not need to implement a complex architecture.

Mainly, the CTC is implemented by add a "blank" (noted as "-" usually) probability to handle invariant length. Within the CTC don't annotate every prediction some of them could be annotated as a "blank". Benefited from "blank", two different lengths of feature and prediction would be aligned as well.

This "blank" implement gonna will be described by the following steps:

1. **Loss calculation**
2. **Decoding**

Loss calculation

Before talking CTC loss, the text encoder has to be introducing firstly. A label without blank could be inserted some "blank" and repetitive character as an encoded prediction which is directly corresponding with prediction. Some examples are shown the following:

- "to" → "—ttttttooo", or "-t-o-", or "to"

- “too” → “—tttto-o”, or “-t-o-o-”, or “to-o”, but not “too”

These examples prove that the predictions of DNN “—ttttttooo”, or “-t-o-”, or “to” all correspond with a label "to". And these predictions to label process could be called *decoding* introducing later. The inverse encoding process, decoding, could be noted as:

$$\beta(c) = \mathbf{y} \tag{2.35}$$

where $\mathbf{y}$ is prediction, same with before and c is encoded prediction, like $\beta(”---tttttttooo”) = ”to”$

Moreover, it is clear that a prediction could be decoding from several encoded predictions whose probabilities which are directly from network ("-" are one of the outputs of a network as well). A decoded prediction probability adding is written as:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{c \in \beta^{-1}(y)} p(c|\mathbf{o}) \tag{2.36}$$

Then the loss value should be calculated for given pairs of input features and labels to train the network. The network outputs a matrix containing probabilities for each character at each time-step, and the character-scores sum to 1 for each time-step. The probability of prediction calculation step by step is showed in figure 30.

In the figure, one column represents all prediction at the same time step, but the "blank" la-

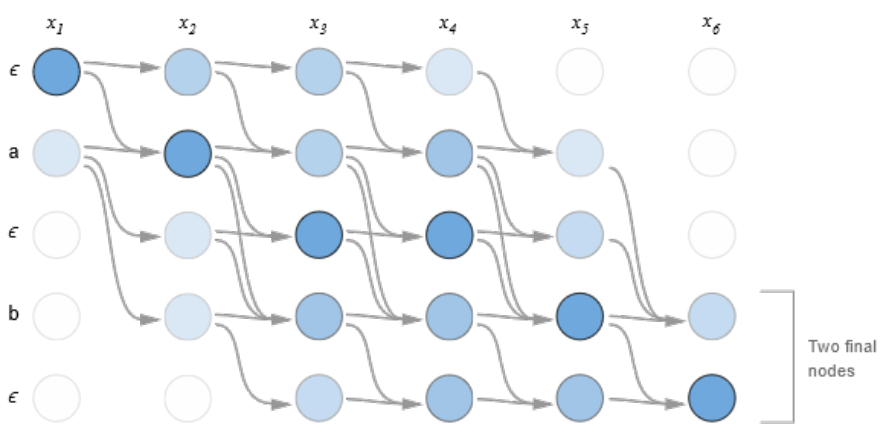


Figure 30: ctc cost computing [2], freely reuse with license Creative Commons Attribution license (CC BY 4.0) https://creativecommons.org/licenses/by/4.0/

bel (here is ϵ) are inserted between every character prediction for the following description. This inserted label sequence is represented as:

$$Z = [-, y_1, -, y_2, \ldots, -, y_U, -] \tag{2.37}$$

where, y_t is character of label sequence at each time step and U is the length of y_t.

Encoded sequence probabilities are computed one step by step. This forward computing is issued with dynamic programming. A probability of encoded sequence in a time step is noted as $\alpha_{s,t}$, the next time step probability could be rewritten as:

$$\alpha_{s,t} = \begin{cases} (\alpha_{s-1,t-1} + \alpha_{s,t-1})\, p_t\,(z_s|X)\,, & \textit{last character in } \alpha_{s,t} \textit{ is "blank"} \\ (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1})\, p_t\,(z_s|X)\,, & \textit{otherwise} \end{cases} \tag{2.38}$$

where, $p_t(z_s|X)$ is a probability of the current character at input step t. Obviously, this decoded forward computing is issued with dynamic programming. This forward probabilities computing process is showed in figure 30.

With feature sequence's personalities computed by the equation 2.38, the CTC loss could be computed from paths probabilities decoded from outputs matrix could be described as:

$$loss = -\log p(\mathbf{y}|\mathbf{x}) \tag{2.39}$$

Decoding

Even a decoding process has been described above, but the described process usually used only in the loss computing. However, based on this basic process, recognition process usually is issued with two algorithms: *Beam search* and *Token passing*.

For the beam search algorithm, the genesis beam is the empty labeling and at later each time-step, the beams in the set beams are extended by all possible characters. The extensions are weighted by a language model normally which scores depending on the new character and the last character in the beam y next to each other.

Typically, there is a dictionary W containing words given by train set, and output of it is constrained to a sequence of these words. For each model, there is a corresponding word phone-level model which is a state machine undirectional connecting an encoded phone sequence like equation 2.37. A sequence is modeled by putting multiple words-models in parallel, connecting all end-states with all begin-states. The decoding flow is implemented by tokens like *viterbi decoding*, which are passed from state to state. Each token holds the score and the history of already visited words. As time step goes by, a final characters sequence could be decoded.

A comparison of these two decoded algorithms is offered by [45]. Usually, Output of token pass is constraint by a dictionary. With a bigger dictionary, the decoding running time is increased quadratically. A smaller diction could not handle out of vocabulary (OOV) which will significantly increase error. But beam search decoding could handle arbitrary output word.

2.3.14 Loss combine

In most multitask-training, since there are multiple pairs of labels and prediction, there are multiple losses corresponding. And there are several methods about how to integrate these losses into one total loss.

Static loss combine

Usually, if there are several losses in a multitask learning network $loss_i$. The classic and simple static losses combine could be written as below equation:

$$loss \ = \ \sum_i w_i loss_i \tag{2.40}$$

where, w_i are tunable hyper-parameters representing corresponding weight of each loss. So in the equation, each w_i is a tunable parameter, which could combine loss value as a preset ratio.

Dynamic loss combine

Original used in multitask image field [46], it could adjust the loss weight dynamic based on its task's uncertainty.

This advanced method will be introduced step by step. Firstly, assume all task implement by the network could be classified as two subgroups: *regression* and *classify*. For regression task, all regression tasks are represented by a set of functions $f_i(x)$, while all classify task are represented by a set of functions $g_i(x)$.

Second, assume likelihood as a Gaussian with mean given by the model regression output with labelled data as:

$$p\left(\mathbf{y}|\mathbf{f_i}(\mathbf{x})\right) = \mathcal{N}\left(\mathbf{f_i}(\mathbf{x}), \sigma_i^2\right) \tag{2.41}$$

Where σ_i is an observation noise scala of each regression task.

And then, the log likelihood of the network is led to the equation as:

$$\log p\left(\mathbf{y_i}|\mathbf{f_i}(\mathbf{x})\right) \propto -\frac{1}{2\sigma^2}\left\|\mathbf{y_i} - \mathbf{f_i}(\mathbf{x})\right\|^2 - \log\sigma \tag{2.42}$$

In that equation, $\|\mathbf{y_i} - \mathbf{f_i}(\mathbf{x})\|^2$ noted as $\mathcal{L}$ exactly is the *minimum square error* (MSE) loss.

If two-tasks of a model are both regression task, the two-regression likelihood function could be written as:

$$\begin{aligned} p\left(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}(\mathbf{x})\right) &= p\left(\mathbf{y}_1|\mathbf{f_1}(\mathbf{x})\right) \cdot p\left(\mathbf{y}_2|\mathbf{f_2}(\mathbf{x})\right) \\ &= \mathcal{N}\left(\mathbf{y}_1; \mathbf{f_1}(\mathbf{x}), \sigma_1^2\right) \cdot \mathcal{N}\left(\mathbf{y}_2; \mathbf{f_2}(\mathbf{x}), \sigma_2^2\right) \end{aligned} \tag{2.43}$$

Then, multitask negative log lilihood leads to a joint loss function $\mathcal{L}\left(\mathbf{W}, \sigma_1, \sigma_2\right)$ and is written as:

$$\begin{aligned} &-\log p\left(\mathbf{y}_1, \mathbf{y}_2|\mathbf{f}(\mathbf{x})\right) \\ &\propto \tfrac{1}{2\sigma_1^2}\left\|\mathbf{y}_1 - \mathbf{f_1}(\mathbf{x})\right\|^2 + \tfrac{1}{2\sigma_2^2}\left\|\mathbf{y}_2 - \mathbf{f_2}(\mathbf{x})\right\|^2 + \log\sigma_1\sigma_2 \\ &= (\tfrac{1}{2\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \tfrac{1}{2\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log\sigma_1\sigma_2 \end{aligned} \tag{2.44}$$

where, $\|\mathbf{y_i} - \mathbf{f_i}(\mathbf{x})\|^2 = \mathcal{L}_i(\mathbf{W})$ exactly is the MSE loss of each task.

Normally, the classification predictions are squashed by softmax function.

$$p\left(\mathbf{y}|\mathbf{g_i}(\mathbf{x})\right) = \text{Softmax}\left(\mathbf{g_i}(\mathbf{x})\right) \tag{2.45}$$

For each task $p\left(\mathbf{y}_1|\mathbf{g}(\mathbf{x})\right)$, if a positive scale times each classification probability, its likelihood equation is written as:

$$p\left(\mathbf{y}|\mathbf{g}(\mathbf{x}), \sigma\right) = \text{Softmax}\left(\frac{1}{\sigma^2}\mathbf{g}(\mathbf{x})\right) \tag{2.46}$$

This euqation could be proved by Boltzmann distribution. Similar with above, the log likelihood is leaded as joint loss $\mathcal{L}\left(\mathbf{W}, \sigma_1, \sigma_2\right)$, eqaution is written as:

$$\begin{aligned}\log p\left(\mathbf{y} = c|\mathbf{g}(\mathbf{x}), \sigma\right) &= \frac{1}{\sigma^2}g_c(\mathbf{x}) - \log\sum_{c'}\exp\left(\frac{1}{\sigma^2}g_{c'}(\mathbf{x})\right) \\ &= \log p\left(\mathbf{y}_2 = c|\mathbf{g_1}(\mathbf{x}), \sigma_1\right) + \log p\left(\mathbf{y}_2 = c|\mathbf{g_2}(\mathbf{x}), \sigma_2\right)\end{aligned} \tag{2.47}$$

where, $g_c(\mathbf{x})$ respectively represent the cth class's prediction.

Further, a two task network's negative log likelihood is rewritten as following:

$$\begin{aligned}&-\log p\left(\mathbf{y}_1, \mathbf{y}_2 = c|\mathbf{g}(\mathbf{x})\right) \\ &= \text{Softmax}\left(\mathbf{y}_1 = c; \mathbf{g}(\mathbf{x}), \sigma_1\right) + \text{Softmax}\left(\mathbf{y}_2 = c; \mathbf{g}(\mathbf{x}), \sigma_2\right) \\ &= \log p\left(\mathbf{y}_1 = c|\mathbf{g}(\mathbf{x}), \sigma_1\right) + \log p\left(\mathbf{y}_2 = c|\mathbf{g}(\mathbf{x}), \sigma_2\right) \\ &= \tfrac{1}{\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \tfrac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log\frac{\sum_{c'}\exp\left(\frac{1}{\sigma_1^2}g_{c'}(\mathbf{x})\right)}{\left(\sum_{c'}\exp(g_{c'}(\mathbf{x}))\right)^{\frac{1}{\sigma_1^2}}} + \log\frac{\sum_{c'}\exp\left(\frac{1}{\sigma_2^2}g_{c'}(\mathbf{x})\right)}{\left(\sum_{c'}\exp(g_{c'}(\mathbf{x}))\right)^{\frac{1}{\sigma_2^2}}} \\ &\approx \tfrac{1}{\sigma_1^2}\mathcal{L}_1(\mathbf{W}) + \tfrac{1}{\sigma_2^2}\mathcal{L}_2(\mathbf{W}) + \log\sigma_1 + \log\sigma_2 \\ &= \textstyle\sum_{i=1}^{2}(\tfrac{1}{\sigma_i}\mathcal{L}_i(\mathbf{W}) + log\sigma_i)\end{aligned} \tag{2.48}$$

It should be noted that comparing equation 2.44, 2.48 with 2.40, except each weight loss, there are corresponding extra uncertainty added as a regularization part for each loss.

The only difference between this dynamic loss combine of regression and classification is the additivity index of each loss.

2.4 Common ASR Architectures

In general ASR architecture, the observed vector at each time could be represented as:

$$\mathbf{O} = o_1, o_2, o_3 \ldots, o_t \tag{2.49}$$

Similar, traeat an sentence will be predicted as:

$$\mathbf{W} = w_1, w_2, w_3 \ldots, w_t \tag{2.50}$$

The general ASR task could be explained as Bayes classification task:

$$\hat{W} = \underset{W \in \mathscr{L}}{\operatorname{argmax}} P(W|O)P(O) \tag{2.51}$$

where $\mathscr{L}$ is a lexicon, which means all words recognized belongs to a finite set. The acoustic model will compute a posterior probability $P(W|O)$ based on input vector O. And language model, usually representing linguistic probability, will model a language probability $P(O)$. For most similar model, it can be model as:

$$p(s) = \frac{T_{\mathrm{s}}}{T} \tag{2.52}$$

where $p(s)$ is the prior probability of each state estimated from the training set. T_s is the number of frames labeled as state s, and T is the total number of frames.

These two probabilities are noted as *Acoustics* and *Languate score* respective. A *decoding* algorithm model an optimal word sequence based on two these scores.

2.4.1 Gaussian Mixture Models - Hidden Markov Models Architecture

The GMM-HMM is excellent architectures of selected speech modelling and recognition applications has been review in [7, section 3.6]. In this architectures, the HMM in speech recognition system plays a role as a generative sequence model of acoustic features of speech while the popularity of the GMM from its ability as a generative single unit model of acoustic features.

With recognizing sequence frame by frame, the HMM hidden state will transition from state to state. Each state is corresponding a unique GMM (sometimes, several state will share parameters [7] [8]). Within a frame, current state GMM will compute a generative probability corresponding current input feature vector.

As showed in figure 31, HMM hiden state corresponding GMM models input vector as a posterior probability $P(W|O)$.

2.4.2 Deep neural network - Hidden Markov Models Architecture

The DNN-HMM takes advantage of DNN's strong nonlinear representation learning power and HMM's sequential modeling ability. The main improvements compared with classic GMM-HMM architecture is DNN usually have a strong nonlinear representation learning, but DNN model could not be modeled speech signal directly.

As described in figure 32, the difference with GMM generation model, DNN usually is viewed as discrimination model. So ASR architecture is different from GMM-HMM. The first item in

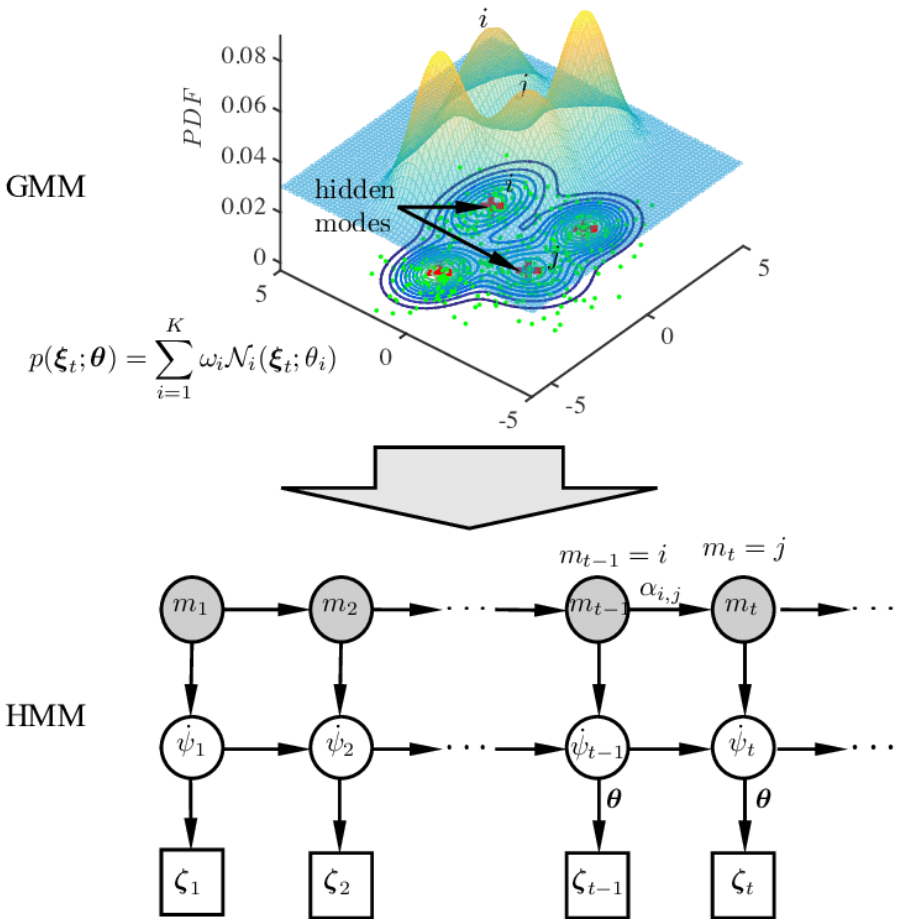


Figure 31: GMM-HMM architecture [3], copyright in section A.1
The circle with shadow represents the hidden mode, circle without shadow represents the unobservable state, and square represents the observable state

equation 2.51 in DNN could be rewritten as:

$$\begin{aligned} p(o|w) &= \sum_{q} p(o|q,w)p(q|w) \\ &\approx \max \pi\left(q_0\right) \prod_{t=1}^{T} a_{q_{t-1}q_t} \prod_{t=0}^{T} p\left(q_t|o_t\right)/p\left(q_t\right) \end{aligned} \tag{2.53}$$

where q is states of HMM. $p(q_t|x_t)$ probability is computed from the DNN, describing a discriminative probability of a state based on feature vector. Also, $\pi(q_0)$ and $a_{q_{t-1}q_t}$ are the initial state probability and state transition probability, respectively, determined by the HMM.

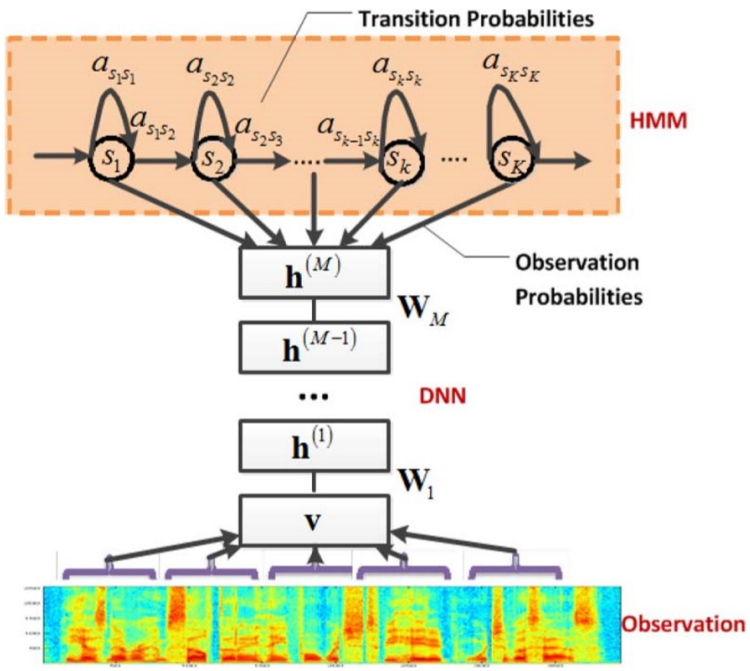


Figure 32: DNN-HMM architecture [4], copyright in section A.1

2.4.3 Joint Connectionist Temporal Classification-Attention (CTC-ATT) end-to-end Architecture

From architecture GMM-HMM to DNN-HMM, the main innovation is improving the model's representation learning within a single frame, but sequences are always modelled by HMM. The recently improving focus on sequential modelling power. CTC and encoder-decoder architecture both are recent academic achievements for dealine with unequality of different length in label and features.

As mention above, subsection 2.3.12 and 2.3.13 have introduced encoder-decoder and CTC mechanism. And [5] and [27] developed a CTC/attention (CTC-ATT) architecture for end-to-end speech recognition. This architecture implements an encoder-attention-decoder architecture, adding an attention mechanism as the "attention" part. Also, CTC is working as another loss function of the encoder in encoder-decoder architecture to train it. This architecture is showed in figure 33.

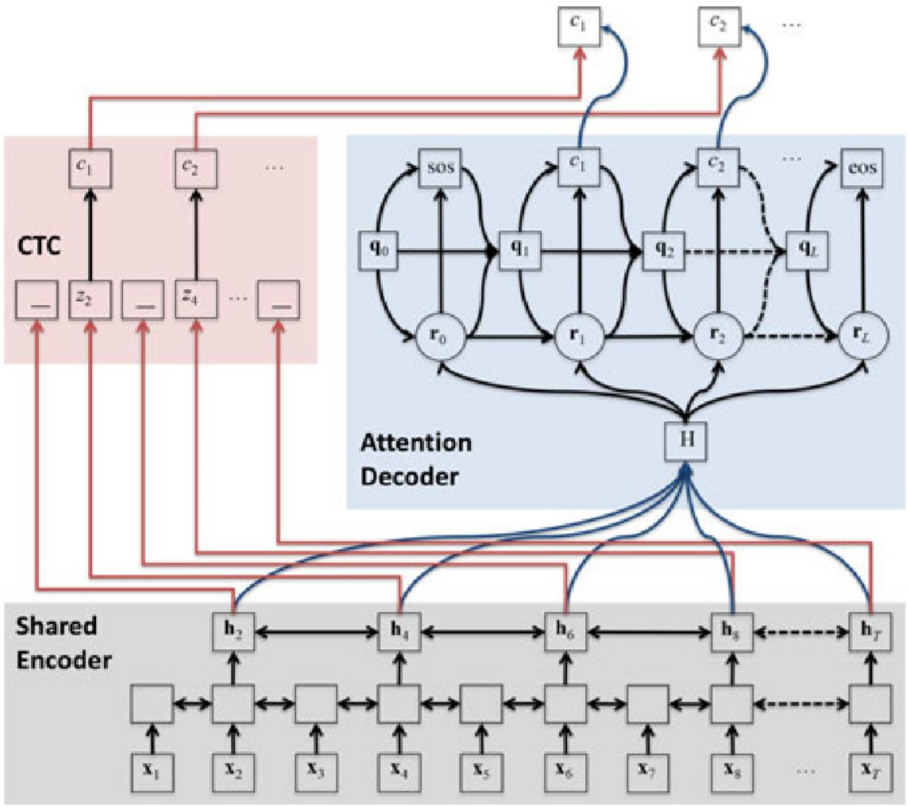


Figure 33: CTC-ATT architecture [5], copyright in sectio A.1

3 Implement of Hybrid attention convolution acoustic model based on variational autoencoder

In the implement, a time-frequency attention autoencoder extracts feature from FBank and send secondary extracted feature into a channel-attention convolutional acoustic model. During this architecture, autoencoder also introduces an unsupervised-learning into this original supervised-learning ASR system. So, this semi-supervised learning system could be reviewed as a two-stage system. The top view of the whole system (HACAM-VAE) is shown in figure 34.

In the figure, it is clear that there are two losses $loss_{super}$ and $loss_{upser}$, respectively supervised-

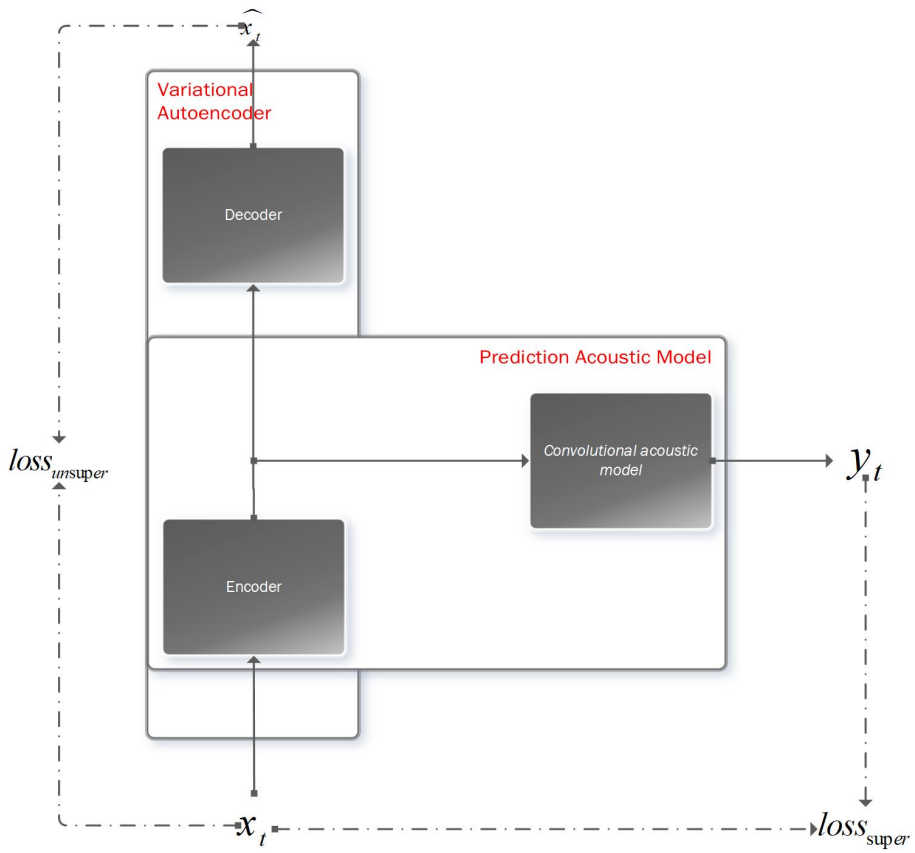


Figure 34: Top view of HACAM-VAE architecture

learning loss and unsupervised-learning loss. These two could dynamic combine mention in the subsection 2.3.14.

In this chapter, section 3.1 describe a time-frequency attention variational autoencoder (for simply notation, later this variational autoencoder will be stated as autoencoder) which is preceding stage. Then the next section 3.2 shows the composing of postage, a convolution acoustic model.

3.1 Time-Frequency attention autoencoder

The example implement of this forestage is shown in figure 35. The input feature is working on the Fbank level.

In the figure, there are four convolution layers in encoder and decoder respectively, two of them are

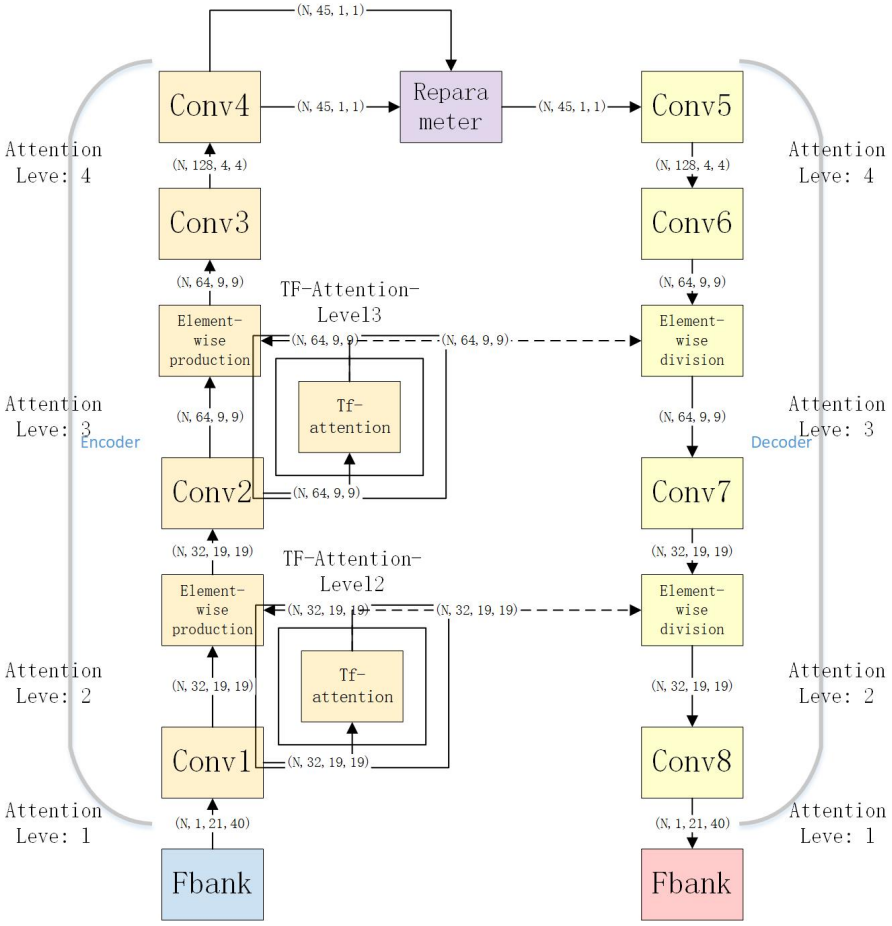


Figure 35: Variational Autoencoder in this case

Conv number	channel_in	channel_out	kernal_size	stride	Activation function
Conv1	1	16	(3,4)	(1,2)	ReLu
Conv2	16	32	(3,3)	(2,2)	ReLu
Conv3	32	64	(3,3)	(2,2)	ReLu
Conv4	64	90	(4,4)	(1,1)	ReLu
Conv5	45	64	(4,4)	(1,1)	ReLu
Conv6	64	32	(3,3)	(2,2)	ReLu
Conv7	32	16	(3,3)	(2,2)	ReLu
Conv8	16	1	(3,4)	(1,2)	Sigmoid

Table 3: Details of each convolution layers

weighted by attention vector. One of interesting thing is the attention vector at same level in encoder and decoder is same (multiplication weight in encoder and division weight in decoder), which could be seen at how *Conv3*, *Conv7* and *Conv2*, *Conv8* are weighted. Meanwhile, since normally, there is a reconstruction error from input feature x_t to rebuild feature $\hat{x_t}$, the weight self-attention vector is computed from encoder's hidden state but decoder's hidden state.

Another interesting point in the network, there are two types of connection from attention weight propagating to decoder division: *regular* and *detach*. For detach connection, there is only forward propagating without back propagating. This idea comes from there always exists a reconstruction error. Encoder's hidden vector have a stronger relevant decoder's hidden vector. Detach connection keeps only encoder's back propagation training weight vector. For regular connection, during training, loss backward propagation still could propagate from decoder, which will stronger help train not only attention parameters but also convolution parameters in the encoder. Actually, its behaviors act like *ResNet* helping train deep layer avoid gradient descent. Besides, these *Conv3*, *Conv7* and *Conv2*,*Conv8* could be noted as attention at level 2 and 3. There are four levels for adding attention mechanism seen at figure 35.

Present original in [47] in image recognition field, and then [11] [12] and [13] replace all 7×7 big kernels with 3×3 small kernels in convolution layer in ASR. The reason for this replacement is explained as multiple small kernels are equivalent with a single large core, but the deeper layer has a strong nonlinear represent learning ability. In this work, most sizes of kernels are close to 3×3 small kernels. Moreover, all of the details of each convolutional layer are shown in table 3.

It should be noted that $Conv4 \sim Conv8$ exist at decoder, so these convolution layers actually are transposed convolution. And the last activation function in the decoder, *"Sigmoid"*, actually is inserted for matching the range of input and output.

The attention mechanism is work on time-frequency these two dimensions since the number of these two dimensions are greater than the size of the channel which could improve more. And the

attention mechanism is implement with the SEnet mentioned in the figure 27 (b).

The input feature is 40 dimension Fbank, and frame windows are extended to 21 frames. Assume feature at one frame t could be represented as x_t, the input feature at this time are represented as $\mathbf{x_t} = [x_{t-10},\ x_{t-9},\ \ldots,\ x_t,\ \ldots,\ x_{t+9},\ x_{t+10}]$. These input feature could be organized as (N, C, W, H) format, where $N,\ C,\ W,\ H$ is the batch size, channel size, frame size, feature size. The reason for using $W,\ H$ not $T,\ F$ represent frame size, the feature size is keeping the same symbol as image field application. When features are fed into the network, obviously $C\ =\ 1$, and N is a set hyper-parameter. Besides, input and rebuild feature are normalized to 0 1 before training and testing.

3.2 Channel attention classifier

Then, a phone classifier will be built as postage. Since a time-frequency attention has been introduced in the forestage. Within this stage, the network focus on the attention on channel dimension. This channel attention is implemented with a resident network together. Specifically, this structure is showed in the figure 36.

In each *junpnet*, there is a resident network, from *Conv* to *Element-wise plus* forming a short connection. The *Batch Normalization-ReLu* structure could import local connection sparsity for further brain simulation and performance improvement since batch Normalization will transform features into normalization distribution and then half of the features could be activated in ReLu. The first *Conv1* is set in the beginning of JumpBlock for dimension transformation since the residual network could not transform dimension. At the end of Jumpblok, there is a channel-attention mentioned in figure 27 (a) for composing a hybrid attention mechanism with time-frequency attention in autoencoder.

In total phone classifier, there are three jump block and two fully connected layers showed in the figure 37.

There is a table 4 to show the details of each block and layers.

It should be noted that there is a reshape layer between JumpBlock4 and FC1 to transform data from (N,1024,4,4) to (N,1024*4*4). In JumpBlock1, JumpBlock2 and JumpBlock4, Conv1 only double the number of channels, so 1×1 kernels and 1 stride are used. And in another JumpBlock not only double the number of channels but also decrease the size of time-frequency dimension through 3×3 kernels and 2 stride. The finnal channel number in last JumpBlock is a common-used 1024 keeping same channel number with other popular CNN AMs. The first fully connection FC1 is built with dropout, batch normalization and Relu activation function to increase sparsity and generalization ability, while there are a dropout and a softmax classification activation function in FC2.

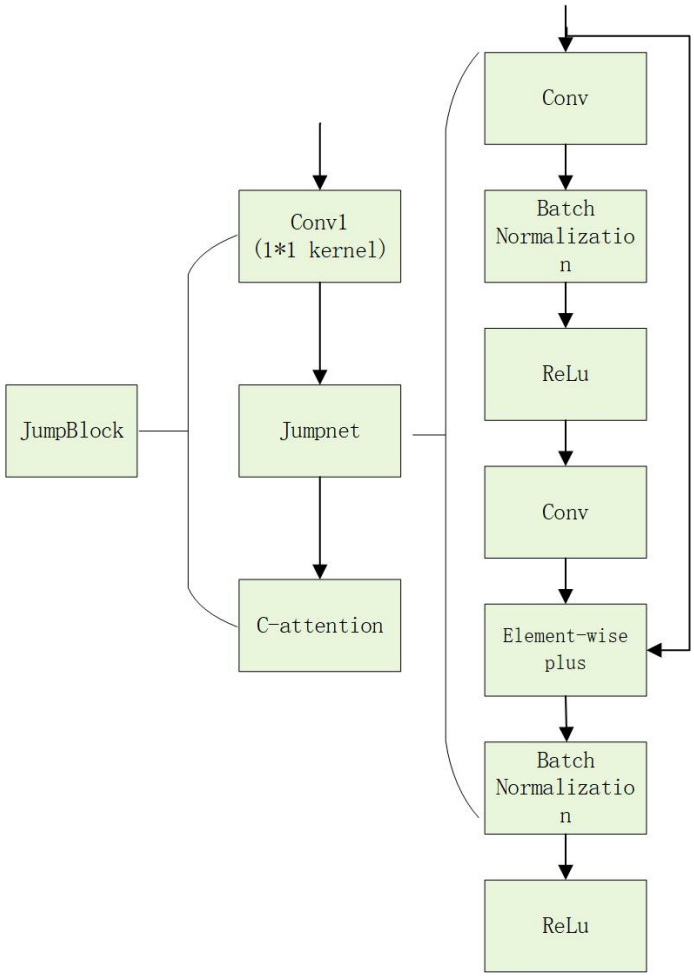


Figure 36: Jumpblok

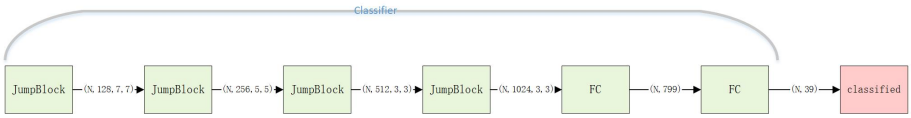


Figure 37: Classifier

This phone classification idea comes from [13], a pure attention CNN AM. In this master, these, JumpBlock structure and attention mechanism are adjusted for autoencoder forestage.

The total structure of autoencoder and phone classifier are showed in figure 38. It should be noted that to avoid training disturb, the connection between classifier with autoencoder is a detached

Layer name	Input format	Output format
JumpBlock1	(N,64,9,9)	(N,128,9,9)
JumpBlock2	(N,128,9,9)	(N,256,9,9)
JumpBlock3	(N,256,9,9)	(N,512,4,4)
JumpBlock4	(N,512,4,4)	(N,1024,4,4)
FC1	(N,1024*4*4)	(N,799)
FC2	(N,799)	(N,39)

Table 4: Details in classifier

connection as blue text noted in the figure.

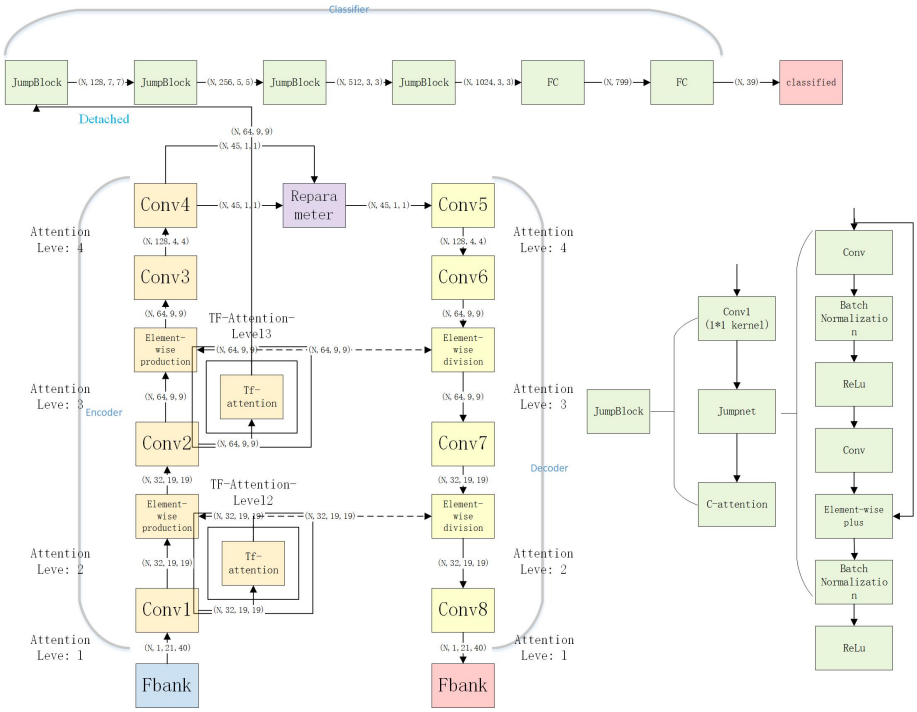


Figure 38: Total model structure

3.3 Dynamic loss combine in CTC-ATT architecture

In subsection 2.3.14, a dynamic loss combine for multitasking has been introduced. However, in CTC-ATT architecture, actually, there are two losses (respectively CTC and ATT loss) for a single task. So, it means that the above dynamic loss combine algorithm actually does not work for this situation. A new algorithm is developed and introduced in this section.

In multitask loss combine algorithm, losses for different tasks are assumed mutual independent to others. However, single task multiloss architecture like CTC-ATT, each loss are dependent on others. Based on this dependent assume, the loss combine should be rewritten as:

$$loss = w \underset{ctc}{loss} + f(w) \underset{att}{loss} + g(w) \tag{3.1}$$

where w is CTC loss weight. $f(w)$ is the corresponding ATT loss weight base on w. Besides, for multiloss control, there is a regularization term $g(w)$.

As the assumption, $f(w)$ should be decreasing function, usually could be:

$$f(w) = 1 - w \tag{3.2}$$

For regularization term $g(w)$, it should be noted be that it act as penalty when weight deviates a preset value. If $g(w)$ is setted as $g(w) = a(w - 0.5)^2$ (The preset value here is 0.5, a is numerical matching factor), partial derivatives could rewritten as:

$$\frac{\partial loss}{\partial w} = \underset{ctc}{loss} - \underset{att}{loss} + 2a(w - 0.5) \tag{3.3}$$

When $\frac{\partial loss}{\partial w} > 0$, it means:

$$\underset{ctc}{loss} + 2a(w - 0.5) > \underset{att}{loss}$$

Furthen, if there is no penalty, this could become like:

$$\underset{ctc}{loss} > \underset{att}{loss}$$

At this condition, the w should be decreased to keep two loss synchronous decline. And $\frac{\partial loss}{\partial w} > 0$ do increase w.

This weight control still working on when $\frac{\partial loss}{\partial w} < 0$:

$$\underset{ctc}{loss} + 2a(w - 0.5) < \underset{att}{loss}$$

without penalty:

$$\underset{ctc}{loss} < \underset{att}{loss}$$

Within this condition, w will be increased.

Combining regularization term with loss synchronous decline, weight w could dynamic float between a certain range. Obviously, w should be at a range $(0, 1)$, but for two-loss contribute to a total loss.

3.4 Corpus and software

For a more flexible model building, a dynamic graph model toolkit needs. With this suppose, Pytorch [48] toolkit is the best choice in the case. Besides, Pytorch offers flexible training for multiple optimizers control for different part (encoder, decoder, phone classifier) in HACAM-VAE. Espnet[31] is a end-to-end ASR toolkit which implement Kaldi's [49] feature extraction and Pytorch or Chainer [50] [51] as backend DNN toolkit.

The project starts with TIMIT speech Corpus [52], since it is a rare phone-level label dataset. This corpus is designed to provide speech data for acoustic phonetic and ASR studies. Since the amount of TIMIT is not enough, a much bigger Corpus is needed. Then librispeech [18], a public domain speech corpus is added as a supplement.

4 Result

In this chapter, some metrics about how to evaluate models are introduced in section 4.1. Then, some setting during training the model are list in section 4.2. A extra preprocessing result are shown in the section 4.3.Some result about autoencoder, phone classifier and joint training are respectively are placed in the section 4.4, 4.5 and 4.6. And for dynamicloss CTCATT loss combine algorithm's results are showed in the section 4.7.

4.1 Metrics

4.1.1 Reconstruction error

Usually, the Reconstruction error is specific metrics for evaluation autoencoder. It could be represented as:

$$\epsilon = \frac{\hat{\mathbf{x_t}} - \mathbf{x_t}}{\mathbf{x_t}} \tag{4.1}$$

This metrics is a quantity to describe the distance two distribution. It is clear that a decent recovery is according to a small reconstruction error.

4.1.2 Frame error rate (FER)

For evaluation phone classifier, since it is a typical classification task, this basic metrics accuracy could be represented as:

$$\epsilon_t = \begin{cases} 1 & \forall j \ y_t(c_t) > y_t(c_j) \\ 0 & otherwise \end{cases} \tag{4.2}$$

This equation shows that at time t, the accurate linguistics have the biggest output probability ($\forall j \ y_t(c_t) > y_t(c_j)$).

For a whole sequence, accuracies are extended as a sequence accuracy like $\epsilon = \frac{\sum_t(\epsilon_t)}{T}$ which are some common metrics for an acoustic model.

4.1.3 Word error rate (WER)

It is general that the recognized word sequence is different with reference word sequence. And WER is come from the Levenshtein distance, instedad working at phone level but at the word level. And there is quantity named word error rate (WER) described as:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{4.3}$$

where, S, D, I and C respectively are the number of substitutions, deletions, insertions, correct. Obviously, N is the number of words in the reference ($N = S + D + C$).

With Normalization to $0 \sim 1$	**Without Normalization**
convergence	fail

Table 5: Different result of with and without normalization

4.2 Traning setting

For a higher accuracy, the number of phones in TIMIT is reduced to 39 [53].

In the case, there are four Adam optimizers respectively trains encoder, decoder, phone classifier, and twos trainable loss combine weights. All these optimizer are setted with 1.0 learning rate, $(0.9, 0.99)$ beat and 0 weight decay.

Moreover, loss functions for autoencoder and phone classifier are minimus square error and cross entropy. These trainable two loss weights are initialized with 0.5.

All training processes are computed by two NVIDIA 1080Ti GPU with CUDA 10.1 driver and a 7.3 cuDNN toolkit. Sixteen cores Intel i7-5960X CPU is used for building model and data reading.

4.3 Fbank feature preprocessing

Something has to be noted is extracted Fbank feature is preprocessed with a normalization to (0,1) range. Also, a test result about this preprocessing is showed in table 5.

4.4 Autoencoer

A comparison of input feature and reconstruction feature from a autoencoder with time-frequency attention are listed in the figure 39.

Another a comparison of input feature and reconstruction feature from a autoencoder without time-frequency attention are listed in the figure 40.

There are table 6 to show reconstriction error comparision of attention working at different level as figure 35. The Baseline in the table 6 means there is no attention mechanism in autoencoder.

4.5 Phone classifier

For compare the result of different attention could be used in the phone classifier, two different phone classifier are built and tested. These results are showed in table 7.

4.6 Joint training

For testing the joint training algorithm, the model is training with two differesent method. The joint training means forestage and postage are training together with joint training algorithm while

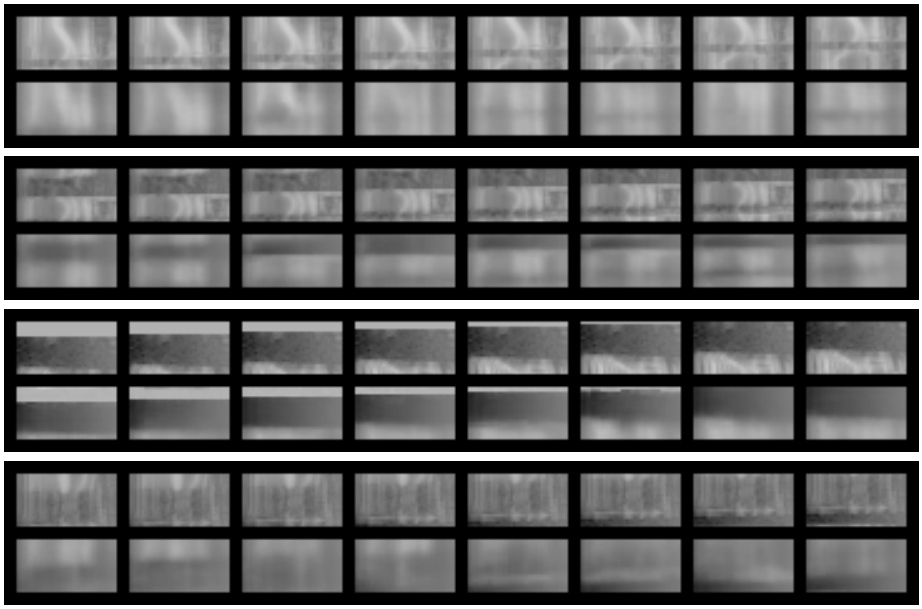


Figure 39: Four comparsion of input features and reconstruction features from a autoencoder with time-frequency attention, Upper is input features, lower is reconstruction features

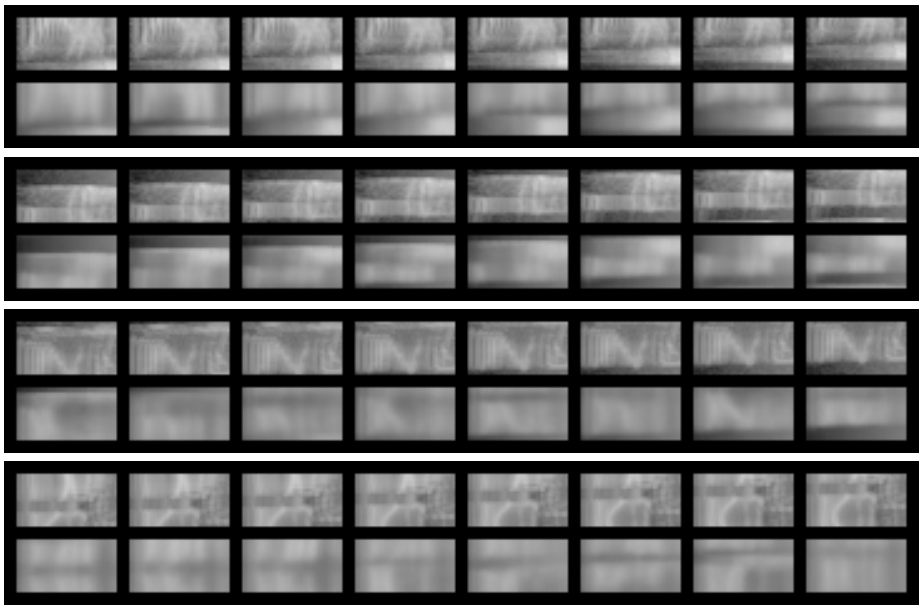


Figure 40: Some comparsion of input features and reconstruction features from a autoencoder without attention, Upper is input features, lower is reconstruction features

Attention1	Attention2	Detach	Reconstruction error
Hybrid SE SA attention			
Leve1: SE	Level2: SE	true	20.1%
		false	6.7%
Leve1: SE	Level3: SE	true	20.1%
		false	7.5%
Leve1: SE	Level4: SA	true	20.1%
		false	7.8%
Leve2: SE	Level3: SE	true	20.1%
		false	8.5%
Leve2: SE	Level4: SA	true	20.1%
		false	9.3%
Leve3: SE	Level4: SA	true	20.1%
		false	8.6%
Pure SA attention			
Leve1: SE	Level2: SE	true	20.0%
		false	3.7%
Leve1: SE	Level3: SE	true	20.0%
		false	4.7%
Leve1: SE	Level4: SE	true	20.0%
		false	7.2%
Leve2: SE	Level3: SE	true	20.2%
		false	5.4%
Leve2: SE	Level4: SE	true	20.2%
		false	7.6%
Leve3: SE	Level4: SE	true	20.2%
		false	9.1%
Baseline			20.1%

Table 6: A reconstruction error comparision about attention working on different level and baseline SA:Self attention, SE:SEnet

FER	
Time-frequency attention	Channel attention
26%	22%

Table 7: Time-frequency and channel attention in classifier

separate training means forestage and postage are trained in order. The test FERs are shown in table 8.

FER	
joint training	seperate training
22%	25%

Table 8: Result of joint training and separate training

4.7 Dynamic loss combine in CTC-ATT architechture

This test is different from above sections, so new architecture and dataset are adopted. Within the test, CTC-ATT architecture is implemented with VGGBLSTM in [54] encoder and RNN language model. This structure could be seen in figure 41.

It is clear that the character sequences are decoded from the RNN language model and CTC-

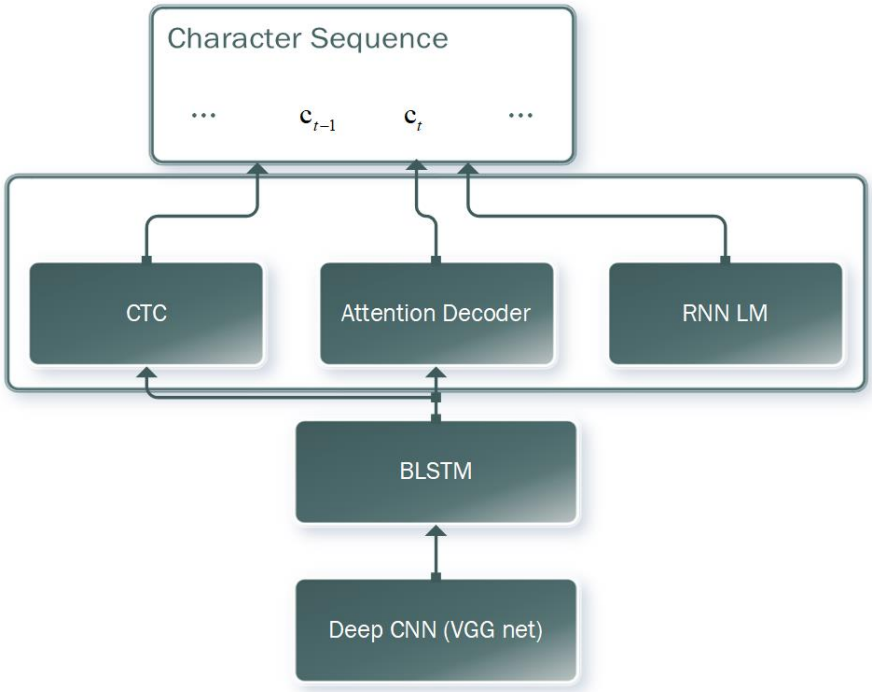


Figure 41: VGGBLSTM structure

ATT joint acoustic model.

Also, this model is trained with adadelt optimizer and numerical matching factor is selected to a detached root mean squared $\sqrt{loss_{att}^2 + loss_{att}^2}$ for its decreasing more smooth and steady. A table 9 showed the result of WERs for different training method.

	WER			Training time		
	Epoch8	Epoch9	Epoch10	Epoch8	Epoch9	Epoch10
Static loss combine	4.9%	4.4%	4.4%	39.38h	45.03h	50.86h
Dynamic CTC-ATT loss combine	4.4%	4.4%	4.4%	39.25h	44.83h	50.38h

Table 9: Dynamic CTC-ATT loss combine

5 Discussion

5.1 Autoencoder

In table 5, it is clear that without this preprocessing, the autoencoder couldn't convergence, and postage phone off course cannot reach a higher accuracy as well.

As seen in the figure 39 and 40 in section 4.4, it is clear that compared with original input feature, the construction feature looks like a "smoothed" feature. That is because encoded space is a continuous distribution (i.e. N(0,1)), hence there bound to be some smooth transition on the edge of the clusters.

Also in table 6, it should be noted that all *detach false* connection attention autoencoders indeed decrease reconstruction error for kinds of attention method. Moreover, attention at a low level has a better result compared with at a high level. The reason for this could be explained with low-level attention could have a more direct influence on the decoder's output, which could significantly reduce the reconstruction error.

And with two types of attention implement, the self-attention have a lower reconstruction error. Maybe SEnet's performance worse than self-attention could be explained by the reason SEnet has more parameters. It usually needs a considerable training dataset. Also, SEnet original working at a dimension with a considerable size. Its squeeze mechanism could not come into play at all. This might be another reason that SEnet could not work better that self-attention.

Another interesting point should be noted is that all detached connection does not improve the performance. It proves the attention in this model also play a short connection role. More specifically, this role could help deep layer in encoder training.

Self-attention working on level 1 and 2 autoencoder reaches the lowest reconstruction error at 2.7%. Meanwhile, the baseline model (an autoencoder without autoencoder) have a reconstruction error of around 20.1%.

5.2 Phone Classifier

In the Classifier, composed hybrid attention has been improved the performance compared with a single type of attention.

From the point of attention, it could be reviewed as the network should focus not only on time-

frequency dimension as a conventional acoustic network but only in channel dimension.

5.3 Joint training

In the comparison of joint training with separate, joint not only indeed improve the FER performance but also reduce the training process.

Since there are two types of label, autoencoder's and phone classifier's label during this semi-supervised learning, these two learning indeed could be weighted integrated through by each uncertainty.

5.4 Dynamic loss combine in CTC-ATT architechture

It is clear that transforming tunable to trainable parameters don't remarkable increase the training time for each epoch. However, it indeed decreases the number of reach to the highest WER from 9 Epoch to 8 Epoch.

6 Conclusion

6.1 Model

During this master thesis, a hybrid attention convolution acoustic model autoencoder based is built. For autoencoder, it indeed reduces reconstruction error with time-frequency attention compared with the baseline structure. With another attention, channel attention inserted into post stage of autoencoder, the hybrid attention does improve its performance as well. Also, this semi-supervised learning could be combined loss by an uncertainty assumption.

6.2 Future Work

Since semi-supervised HACAM-VAE and dynamic loss combine algorithm in CTC-ATT architecture have been separately developed to a mature level, the next step of work should be inserted this HCAM-VAE and VGGBLSTM in CTC-ATT architecture to improve performance. More specifically, the time-frequency autoencoder' encoder could replace original VGG, and channel attention mechanism could be inserted into BLSTM network. Meanwhile, there are three loss in total, autoencoder, CTC, ATT loss. Last two loss could be integrated with the dynamic loss combine algorithm in CTC-ATT architecture, and then combined supervised loss will be integrated with autoencoder loss with dynamic loss combine algorithm based on uncertainty.

Bibliography

[1] Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. 2019. *Dive into Deep Learning*. http://www.d2l.ai.

[2] Hannun, A. 2017. Sequence modeling with ctc. *Distill*. https://distill.pub/2017/ctc. doi:10.23915/distill.00008.

[3] Wang, W., Zhao, D., Han, W., & Xi, J. 2018. A learning-based approach for lane departure warning systems with a personalized driver model. *IEEE Transactions on Vehicular Technology*, 67(10), 9145–9157.

[4] Dahl, G. E., Yu, D., Deng, L., & Acero, A. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1), 30–42.

[5] Kim, S., Hori, T., & Watanabe, S. 2017. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 4835–4839. IEEE.

[6] Huang, X., Acero, A., Hon, H.-W., & Reddy, R. 2001. *Spoken language processing: A guide to theory, algorithm, and system development*, volume 1. Prentice hall PTR Upper Saddle River.

[7] Yu, D. & Deng, L. 2016. *AUTOMATIC SPEECH RECOGNITION*. Springer.

[8] Jurafsky, D. & Martin, J. H. 2014. *Speech and language processing*, volume 3. Pearson London.

[9] Virtanen, T., Singh, R., & Raj, B. 2012. *Techniques for noise robustness in automatic speech recognition*. John Wiley & Sons.

[10] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6), 82–97.

[11] Sercu, T., Puhrsch, C., Kingsbury, B., & LeCun, Y. 2016. Very deep multilingual convolutional neural networks for lvcsr. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4955–4959. IEEE.

[12] Sercu, T. & Goel, V. 2016. Advances in very deep convolutional neural networks for lvcsr. *arXiv preprint arXiv:1604.01792*.

[13] Yu, D., Xiong, W., Droppo, J., Stolcke, A., Ye, G., Li, J., & Zweig, G. 2016. Deep convolutional neural networks with layer-wise context expansion and attention. In *Interspeech*, 17–21.

[14] Palaz, D., Collobert, R., et al. Analysis of cnn-based speech recognition system using raw speech as input. Technical report, Idiap, 2015.

[15] Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., & Penn, G. 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, 4277–4280. IEEE.

[16] Tian, X., Zhang, J., Ma, Z., He, Y., Wei, J., Wu, P., Situ, W., Li, S., & Zhang, Y. 2017. Deep lstm for large vocabulary continuous speech recognition. *arXiv preprint arXiv:1703.07090*.

[17] Zhang, S., Lei, M., Yan, Z., & Dai, L. 2018. Deep-fsmn for large vocabulary continuous speech recognition. *arXiv preprint arXiv:1803.05030*.

[18] Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. 2015. Librispeech: an asr corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 5206–5210. IEEE.

[19] Han, K., He, Y., Bagchi, D., Fosler-Lussier, E., & Wang, D. 2015. Deep neural network based spectral feature mapping for robust speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

[20] Lu, X., Tsao, Y., Matsuda, S., & Hori, C. 2013. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, 436–440.

[21] Feng, X., Zhang, Y., & Glass, J. 2014. Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 1759–1763. IEEE.

[22] Deng, J., Zhang, Z., Marchi, E., & Schuller, B. 2013. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, 511–516. IEEE.

[23] Deng, J., Zhang, Z., Eyben, F., & Schuller, B. 2014. Autoencoder-based unsupervised domain adaptation for speech emotion recognition. *IEEE Signal Processing Letters*, 21(9), 1068–1072.

[24] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R., eds, 5998–6008. Curran Associates, Inc. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[25] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, 577–585.

[26] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., & Bengio, Y. 2016. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4945–4949. IEEE.

[27] Watanabe, S., Hori, T., Kim, S., Hershey, J. R., & Hayashi, T. 2017. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8), 1240–1253.

[28] Garrett, S. L. 2017. Understanding acoustics. *Cham, CH: Springer*.

[29] Anderson, J. & Law, C.-W. 1977. Real-number convolutional codes for speech-like quasi-stationary sources (corresp.). *IEEE Transactions on Information Theory*, 23(6), 778–782.

[30] for Standardization, I. O. Iso 226:2003 acoustics – normal equal-loudness-level contours. *International Organization for Standardization, Geneva, Switzerland*.

[31] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Enrique Yalta Soplin, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. 2018. Espnet: End-to-end speech processing toolkit. In *Interspeech*, 2207–2211. URL: http://dx.doi.org/10.21437/Interspeech.2018-1456, doi:10.21437/Interspeech.2018-1456.

[32] Chan, W., Jaitly, N., Le, Q., & Vinyals, O. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4960–4964. IEEE.

[33] Goodfellow, I., Bengio, Y., & Courville, A. 2016. *Deep Learning*. MIT Press, http://www.deeplearningbook.org.

[34] Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. 2018. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, 2483–2493.

[35] Leinweber, D. J. 2007. Stupid data miner tricks: overfitting the s&p 500. *Journal of Investing*, 16(1), 15.

[36] Everitt, B. S. 2006. *The Cambridge dictionary of statistics*. Cambridge University Press.

[37] Doersch, C. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[38] Schraudolph, N. 1998. Accelerated gradient descent by factor-centering decomposition. *Technical report/IDSIA*, 98.

[39] Raiko, T., Valpola, H., & LeCun, Y. 2012. Deep learning made easier by linear transformations in perceptrons. In *Artificial intelligence and statistics*, 924–932.

[40] Srivastava, R. K., Greff, K., & Schmidhuber, J. 2015. Training very deep networks. In *Advances in neural information processing systems*, 2377–2385.

[41] He, K., Zhang, X., Ren, S., & Sun, J. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

[42] Bahdanau, D., Cho, K., & Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[43] Hu, J., Shen, L., & Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

[44] Sutskever, I., Vinyals, O., & Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

[45] Scheidl, H. Comparison of connectionist temporal classi
cation decoding algorithms. URL: https://github.com/githubharald/CTCDecoder/blob/master/doc/comparison.pdf.

[46] Cipolla, R., Gal, Y., & Kendall, A. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7482–7491. IEEE.

[47] Simonyan, K. & Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[48] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. 2017. Automatic differentiation in pytorch.

[49] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., & Vesely, K. December 2011. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[50] Tokui, S., Oono, K., Hido, S., & Clayton, J. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*. URL: http://learningsys.org/papers/LearningSys_2015_paper_33.pdf.

[51] Akiba, T., Fukuda, K., & Suzuki, S. 2017. Chainermn: Scalable distributed deep learning framework. In *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*. URL: http://learningsys.org/nips17/assets/papers/paper_25.

[52] Garofolo, J. S. 1993. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*.

[53] Lee, K.-F. & Hon, H.-W. 1989. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11), 1641–1648.

[54] Hori, T., Watanabe, S., Zhang, Y., & Chan, W. 2017. Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm. *arXiv preprint arXiv:1706.02737*.

A Theoretical data

A.1 IEEE Thesis / Dissertation Reuse copyright

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE. 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table. 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication] 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line. 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

A.2 Recognition Unit

ARPAbet sysbol	**Word**	**ARPA Transcription**
p	parsley	p aa r s l iy
t	tea	t iy
k	cook	kuh k
b	bay	b ey
d	dill	d ih l
g	garlic	g aa r l ix k
m	mint	m ih n t
n	nutmeg	n ah t m eh g
ng	baking	b ey k ix ng
f	flour	k l ow v
v	clove	k l ow v
th	thick	th ih k
dh	those	dh ow z
s	soup	s uw p
z	eggs	eh g z
sh	squash	s k w aa sh
zh	ambrosio	ae m b r ow zh ax
ch	cherry	ch eh r iy
jh	jar	jh aa r
l	licorice	l ih k axr ix sh
w	kiwi	k iy w iy
r	rice	r ay s
y	yellow	y eh l ow
h	honey	h ah n iy
Less commonly used phones		
q	uh-ph	q ah q ow
dx	butter	b ah dx axr
nx	winner	w oj mx axr
el	table	t ey b el

Table 10: ARPAbet symbols for transcription of English consonants, [8, Chapter 7]

NTNU
Norwegian University of
Science and Technology