NANYANG TECHNOLOGICAL UNIVERSITY

WAKEUP WORD DETECTION USING END TO END DEEP LEARNING FRAMEWORKS

LIU BOZHONG

School of Computer Science and Engineering

2021

NANYANG TECHNOLOGICAL UNIVERSITY

MSAI Master Project MSAI/20/001

WAKEUP WORD DETECTION USING END TO END DEEP LEARNING FRAMEWORKS

Submitted by
LIU BOZHONG
under the supervision of
Asso. Prof. Chng Eng Siong

School of Computer Science and Engineering
2020

Abstract

Wake-up word detection is also called, keyword spotting (KWS). It is an essential component to trigger the user voice on smart devices. In other to obtain good user experience, it needs real-time interaction and high accuracy. Neural networks have recently become a popular approach for keyword spotting, as they are superior to conventional voice processing algorithms. In this thesis, some typical popular networks are trained such as VGG, Deep Residual Network (ResNet), WideResNet (WRN), Densely Connected Convolutional Network (DenseNet), Dual Path Network (DPN) and Multi-head attention Based network in the objective of comparison in accuracy and storage. In addition, a novel loss function is proposed and implemented on the multi-head attention neural network. When comparing it against other neural network architectures, it achieves an accuracy of 98.2%, which outperforms the Start of Art Result in [1].

Table of Contents

Chapter 1 Introduction

1.1 Background and Significance

Wake-up word detection, or keyword spotting (KWS), is a branch of speech recognition task. It detects a limited number of pre-defined activation words or keywords from a series of speech streams and triggers the corresponding downstream system. For example, Apple's conversational assistant named Siri listens to 'Hey Siri' to initiate voice input for speech recognition. This kind of technology is the basis of enabling the voice interaction ability of embedded devices, and can be applied to various fields, such as mobile phones, smart speakers, robots, smart home, vehicle devices, wearable devices and so on. When the device is in the sleep state, the sound would be continuously picked up to wake-up words. Once the wake-up words are detected, the device switches from the sleep state to the working state, waiting for subsequent interaction.

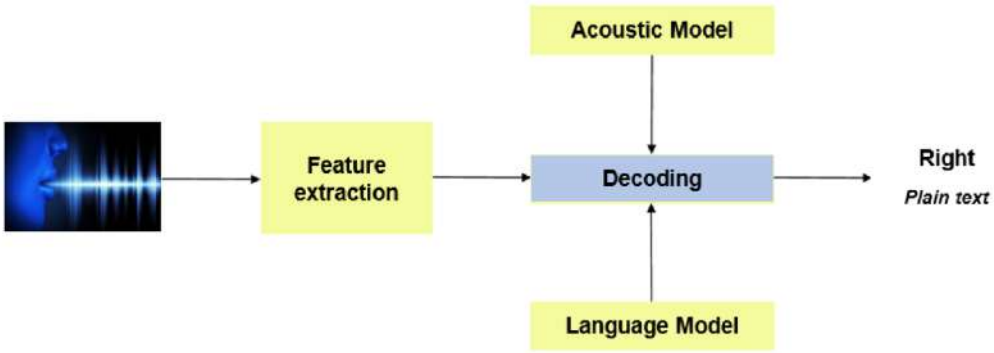

Figure 1. Generic model of a keyword detection system.

Two essential aspects of a speech recognition system are acoustic function extraction and acoustic models. The acoustic features help derive accurate information from the acoustic waveform data, which can be used to identify word and phonetic content. Likewise, an excellent design and training system of the acoustic model ensures that the vectors are well matched to the Phoneme groups, to ensure proper classification of the function. Figure 1 shows the generic model of a KWS device. The acoustic characteristics are taken from the raw speech signal, and the output of the acoustic model is likely to correlate to the specified speech output.

1.2 Open Challenges

1.2.1 Low Latency

The KWS system needs to process real-time speech waveform and generates trigger signals as soon as the user utters the specified keywords. Typically, a frame of speech features is extracted and needed to be processed every 10 ms to 20 ms for KWS. This requires low latency in the hardware implementation.

1.2.2 Low Power Consumption

The KWS system is always-on and serves as a trigger to some downstream systems. Since the battery life is important to mobile or IoT applications, power consumption of the KWS system should be minimized.

1.2.3 High Accuracy

When the KWS system detects specified keywords, it powers up the downstream system. The downstream system such as speech recognizer, is usually much larger, consumes more power than the KWS system and leads to large overall system power. In addition, a false alarm generated by the KWS system will waste a large amount of energy on powering up the downstream system. Furthermore, large false rejection rate leads to frequent miss of keywords and thus has a negative impact on user experience. Given those two aspects, achieving high accuracy in the KWS system is critical.

1.3 Motivation

This research carries out a comprehensive study, experimentation, and comparative analysis of different popular deep learning models to find the most accurate and efficient architecture for wake-up word detection system or also called small footprint keyword spotting system (KWS). In this study, various models are studied using such as VGG, Deep Residual Network (ResNet), WideResNet (WRN), Densely Connected Convolutional Network (DenseNet), Dual Path Network (DPN) and Multi-head attention Based network.

In addition, this thesis also aims to propose a novel loss function to further improve the performance of the KWS system.

1.4 Contribution

(a) To transfer popular neural networks in image classification and natural language processing from the literature [2-7] to KWS field on Google speech commands dataset [8] and compare them in terms of accuracy and model parameters.

(b) To propose a novel loss function inspired by triplet loss used in face recognition [9] and implement it on the end-to-end multi-head attention model. This model outperforms the state-of-the-art performance in accuracy and relatively small model size.

(c) Discussion about how different model topology improves accuracy and makes the keyword spotting models explainable.

(d) To implement a demo to locally detect the keywords in real-time.

1.5 Thesis Outline

The organization of the rest of this dissertation is as follows. Chapter 2 is the literature review of the previous work in traditional methods and recent progress in using neural networks concerning this work. Chapter 3 explains the method that will apply in achieving the aim including the feature extraction, foundation components in deep neural networks and evaluation metrics in the KWS field. Chapter 4 gives a general overview of a standard end to end KWS system and illustrates the different neural network structures and loss functions used in the system. Chapter 5 demonstrates training details and evaluate the models' performance and significance of the proposed loss function. Demo of the project and possible improvements to the system would be given in this chapter as well. Finally, Chapter 6 concludes the work and provides an outline of future plans.

Chapter 2 Literature Review

2.1 HMM Based KWS

Hidden Markov Models (HMMs) and Viterbi decoding[10-12] are traditional speech recognition technology for KWS. These methods form a group of GMM-HMM models consisting for each keyword of the state phonemes set. Hidden Markov models (HMMs) are used to model both keywords and fillers and the detection of the keywords and fillers occurs by searches through a decoding graph. The Viterbi Algorithm after the training is used for decoding the input keyword status sequence and matching it with the learned GMM-HMM keyword model. This system, with an emphasis on the filler word modelling [13-16] and sophisticated scoring procedures [17-20], is followed by more or less later study in this group. In this sense even discriminative teaching techniques to specifically improve the efficiency of keyword spotting [21, 22] have been examined. This approach, however, achieves reduced precision as a non-linear model cannot be constructed using the GMM-HMM-based KWS methods. In addition, they are difficult to train and inferentially costly.

2.2 Neural Network Based KWS

Neural network technologies have however dominated the field in recent years and increased the accuracy of these systems. Standard feedforward deep neural networks (DNNs) [23–25], with relatively little detection latency, are the common architecture. The biggest downside of DNNs is that the local time and spectral similarity in the input speech functions is ignored. The convolutionary neural network (CNN)[26] has recently grown in popularity for KWS in limited memory footprint applications[27] to take advantage of these associations that have been heavily influenced by progress in techniques for computer vision (e.g., image detection and facial recognition).

McMahan et al. [27] used CNN (SB-CNN [28]), ResNet [3] and DenseNet [5] for the recognition of language commands for transferral learning. For their experimentation, they used UrbanSound8k and a speaking data package. With a dilated convolution, they added a multiscale input representation. In addition, they took advantage of various transition training methods and noticed that pre-training on another dataset, namely UrbanSound8k, yielded better outcomes than a direct voice command data set training. They also concluded that only 40% of the training data was

to obtain the same outcome as training on 100% of the data if transfer learning was used with multiscale inputs.

For identification of spoken commands by using raw wave shape as feedback, Jansson [29] used CNN with one-dimensional (1D) convolutions. Three data enhancement methods have been used to improve training data: (i) changing time scope, (ii) scaling amplitudes, and (iii) added noise. Pseudo-labelling was often used to identify unlabelled data and obtain improved outcomes using the learned model.

The classification accuracy was further increased by Andrade et al. [30] by proposing a centred coevolutionary model. Raw wave fires were used to feed the network to derive long-term and short-term dependencies for input and calculated spectrograms. These features were then transmitted to the attention layer that identified the area with useful information, and this information was then used for classification as an input in a dense layer.

The downside of CNNs is that they neglect long term temporal dependence when modelling time varying signals (i.e. language). In [31], a convolutional recurrence network dependent KWS is investigated to combine the strengths of the CNNs with the RNNs and show that the model is resilient to the noise level. Although all previous KWS neural networks have been trained in cross entropy loss function, a max pooling-based loss function for the training of a long-term short-term memorial (LSTM) KWS model is proposed[8], which ensures greater precision than cross entropy loss-trained DNNs and LSTMs. Zhang et al. [32] discussed various architecture of the neural networks, DNN [23], CNN, CRNN [31] and LST M,[33], and even DS-CNN [34], who performed well for everyone in terms of precision, with memory and computing energy limitations, for the same work. Segal et al. [35] have used a data collection for pre-training their convolutional neural network and have used the YOLO architecture (you search only once) [36], taking audio as elements. Their objective was to locate and subsequently classify utterances at borders.

Since successfully implemented the same for other speech domain activities, ResNet was used for the identification of keywords by Tang et al. [37]. [38]. Tang et al. They used res15 as the base model to train their variants of ResNet on the speech command data collection (12 commands). It increased the residual block, examined the effect on the word error rate (WER) of deeper models, and set up new state-of-the-art language command dataset referencing models.

Although several KWS neural network models have been introduced in literature, they are trained and tested in various proprietary datasets (e.g. TalkType data set [39], Alexa dataset [40], etc.). In order to solve this problem, it is necessary to compare these different models on the same dataset.

Chapter 3 Methodology

3.1 Feature Extraction

Speech signals can be digitally interpreted as a number sequence with the same number of items per second as the specimen rate. This representation, however, does not contain audio detection detail. To address this challenge, the extraction of features is often called voice parameterisation, using spectral features of an input audio signal to enable speech decoding. Fast Fourier transform (FFT) on a narrow audio window allows you to translate the raw audio to the frequency dominate. Whilst at any frequency band in the audio window the FFT contains energy information, it doesn't stress the band that is critical for human hearing, which is below roughly 1000 Hz. Davis etal.[41] proposed a cepstral coefficient Mel frequency to imitate the behaviour of the human ear in order to solve this challenge. The main steps involved in computing MFCC features are shown in Figure 2.

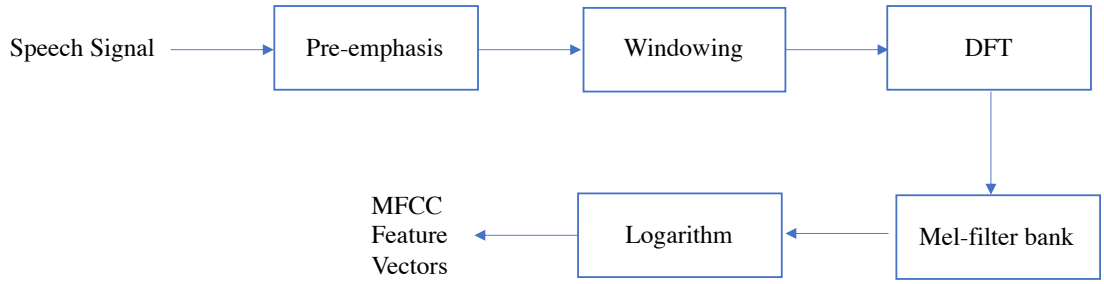

Figure 2. Steps involved in MFCC calculation.

3.1.1 Pre-emphasis

Pre-emphasis is the first step in this process. The range of voiced segments such as vowels rely more on lower frequencies, making higher frequencies meaningless. The first step is to increase signal intensity at high frequencies.

3.1.2 Windowing

Since the characteristic of speech varies throughout an utterance, spectral characteristics achieved throughout the entire utterance are not useful. Functions are usually collected through a small window instead. A streaming audio signal is typically blocked into frames of 25 milliseconds shifted by 10 milliseconds.

After breaking the speech signal into chunks, each frame is multiplied by a window using a windowing function. The Hamming window is generally used since it

approaches 0 at its edges, shrinks the value of the input signal toward 0 at the boundaries and therefore avoid the signal to be abruptly cuts off at the edge. Its equation is shown in Equation (1):

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \; 0 \leq n \leq N-1 \quad (1)$$

where N is the number of samples in a frame and multiplying every frame by the Hamming window reduces discontinuities at the beginning and end of each frame. This step was also required to do the frequency analysis of each frame.

3.1.3 Discrete Fourier Transform (DFT)

Spectral study of speak signals found that various signal timbres vary over frequency in their energy distributions. In order to obtain a magnitude response, DFT was added in each frame of N-samples. This method converts signals to frequency domain from time to time.

3.1.4 Mel-Scale Log Filter Bank Features

Through DFT, energy distribution in frequency domain can be obtained. However, both wavelengths in the human ear are not just as sensitive. Human ear is less alert and less sensitive to frequencies smaller than 1000 Hz. The Mel frequency mimics this non-linear perception of sound by the human ear. Similarly, at lower frequencies it is more discriminatory and at higher frequencies less discriminatory. Therefore, frequencies can be wrapped in Mel scale to improve speech recognition performance. The conversion between Mel frequency (mel) and frequency (f) in hertz can be made by using Equation (2). This wrapping is created at the time of filter bank computation which collects energy from each frequency band. Then logarithmic is applied for frequency values. For speech recognition tasks, 40 Mel-filters is a common number of filters to use [42]

$$mel(f) = 1127\ln\left(1 + \frac{f}{700}\right) \quad (2)$$

3.1.5 Comparison Between Mel-Spectrum and MFCC

In the feature extraction stage of KWS system, log-mel filterbanks instead of the MFCCs is applied. The log-mel filter banks are the log of the triangular mel filters applied to the power spectrogram whereas MFCC is then applying a discrete cosine transform (DCT) to those filters to obtain the final coefficients. However, DCT is a linear transformation on the non-linear speech signal, which may attenuate the

characteristics of the acoustic signal irrelevant to the spoken content, such as the intonation or accent. MFCC decorrelates features whereas using the log-mel filter banks can exploit the strong correlations between adjacent time-frequency components of speech signals.

3.2 Convolutional Neural Networks

Deep neural networks take advantage of the property that certain natural signals are compositional hierarchies in which lower-level functions are achieved. Local variations of borders shape motives, patterns and pieces are assembled into objects in pictures. In speech and text, similar hierarchies occur in phone, phony, syllable, phrase and vocabulary [43].

AlexNet [44] popularized the revolutionary neural networks (CNN) initially launched by LeCun et al. [26] after considerable advances over other subventions at ILSVRC in 2012. This similarity is used by CNNs by treating the time and spectral domain functions of the input as an image and by doing 2-D convolution activities over the input domain.

The neural networks are composed of four major operations: convergences, non-linearities, pooling and sorting. Optionally batch normalization and dropout should be used in the templates. These operations are mostly clustered together to allow for a non-linearity such as the ReLU to obey convolutions. This operation is then replicated a couple times until a combined operation is used to minimize the size of the features. If a network is sufficiently deep and the initial data is subsequently collected in a manageable amount, the input is transferred into the classification section of the network. Typically used after convolution but before non-linearity is batch normalization. In some instances, a linear low-rank layer is inserted between the convolution layers and dense layers to reduce parameters and speed up training, which is simply fully connected without a linear activation [45, 46].

3.2.1 Convolutional Layers

The convolutional layers of a CNN are made up of a number of learning filters, often kernels. Each filter is implemented automatically striding the whole input and generating a map for every filter's output. The idea that the same filter is used over the whole input room makes it possible to display the same motif at various points in the input. Convolutional layers usually use comparatively small filter sizes to detect local

characteristics, which are important for data like images or music. Filters in stacked convolution layers run on the previous layer function diagram, which combines characteristics to detect higher order [43]. On two-dimensional entrances, the highest and width of images, for example, the most common convolution layer operates. Since the work uses input 1-dimensional raw audio waveform files, one-dimensional filters are used by convolutional layers.

3.2.2 Pooling layers

While convolution layers detect local characteristics from the data, a pooling layer fuses semantic related characteristic only by maintaining its maximal value (max-pooling) or averaging the values within a patch (43). [43]. This decreases the susceptibility to changes and characteristic distortions by decreasing the characteristic map scale. Pooling layers after one or more overlay layers are used to continually decrease the input size as the network deepens. Another reduction is the sum of calculations needed in the network in the size of the input, followed by the pooling procedure [26]. While most modern neural networks use layers of max-pooling, some contend against the use of collection layers because they literally discard useful information. Springenberg et al. [47] prove that a greater step in replaced max pools will lead to competitive outcomes relative to max pooling models.

3.2.3 Batch Normalization

When the weight activation distributions change, this is known as the internal covariate transition, since a neural network is developed, and the network parameters are modified. Interior covariate changes make preparation even harder by involving lower levels of learning and vigilant initialization of the parameters. Ioffe and Szegedy [48] also adopted batch normalization to solve this issue, making standardization itself a part of the model. Standardization by batch normalizes the contribution from a preceding layer by using average averages and batch variance. When the weight activation distributions change, this is known as the internal covariate transition, since a neural network is developed, and the network parameters are modified. Interior covariate changes make preparation even harder by involving lower levels of learning and vigilant initialization of the parameters. Ioffe and Szegedy [48] also adopted batch normalization to solve this issue, making standardization itself a part of the model. Standardization by batch normalizes the contribution from a preceding layer by using average averages and batch variance.

3.2.4 Activation Functions

Activation functions are non-linearities between convolutionary strata that make the neural network more dynamic than linear data to model. The rectified linear unit, a function which takes input x and max returns, is a common activation feature (0, x). Older triggering functions Sigmoid and Tanh have been replaced by the ReLU, and derivatives like PReLU. This is because the teaching productivity is much better than the older functions [49].

3.2.5 SoftMax Classification

In most situations there are totally linked layers and SoftMax functions as a grouping aspect of convolutionary neural networks. After the final convolution layer, entirely connected layers are used to balance the neural network output size with the necessary output size. The performance is then passed through SoftMax to construct a probability representation for each class in the controlled learning environment. The contribution from the final convolutions layer is usually flattened to use the completely connected layers, or the characteristic maps are sampled to a dimension of 1.

3.2.6 Dropout

Dropout is a strategy of regularization in which a percentage of links are removed in a layer before the next layer. Randomly lowering a number of units for each workout avoids co-adaptation by untrusted involvement of other units in the network. This forces the network not to learn those relations, but to learn a wider representation. An inconvenience of dropout is increased training time as broken ties increase noise in the modified parameter. During inference, no relations are discarded [50].

3.3 Recurrent Neural Networks

The processing of sequential data is suggested for recurring neural networks (RNNs). They are all-round in any speech application as speech sequential info. RNNs use the "gating" method not only to manipulate the temporal relation between the input signal, but also to catch long-term dependencies. In contrast to CNNs in which input functionality is viewed as a 2-D image, T-time steps work on RNNs, in which the corresponding spectral functional vector is linked to the earlier output, h (t-1, input to RNN, at any step. Figure 3 shows the model architecture of a standard RNN model in

which the LSTM cell[51] or a Gated Recurrent Unit (GRU) cell will constitute an RNN cell[52].

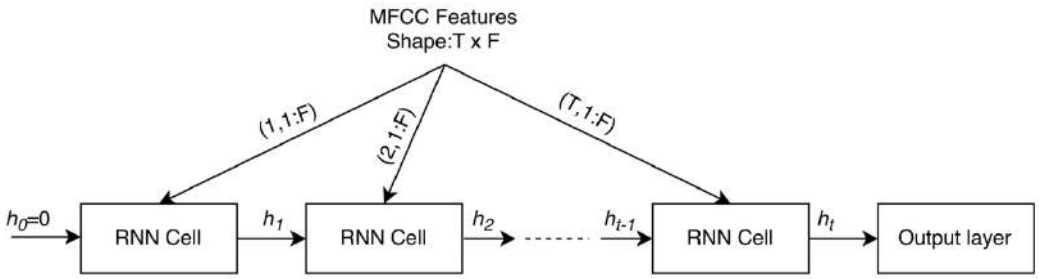

Figure 3. An example of RNN in KWS system.

The RNN models appear to have fewer parameters than the CNNs because all T time stages are repeated. As with normalization in batch for CNNs, the application of layer normalization can be helpful for training RNNs [53] in which each phase of the hidden state is normalized.

3.4 Convolutional Recurrent Neural Network (CRNN)

CRNN [31] combines properties of both CNN and RNN. It uses convolutions layers and global temporal dependence in speech functions to manipulate the local time/space connection using recurrent layers. A CRNN model begins with a series of 2D convolutions for speech functionality followed by a completely connected GRU layer.

The recurrent layer is bi-directional [54] and has many phases, expanding the capacity for network learning. The base cell of recurrent layers is Gated Recurrent Units (GRU)[52], since the base cell uses less parameters and improves convergence than the LSTMs.

3.5 Attention Mechanism

Vaswani et al. [7] suggest the Transformer Model as a way to execute sequence-to-sequence operations, without relying on recurring links. The Multi-Headed Attentiveness (MHA) definition in Figure 4 is used to highlight critical sections of a series. Every MHA consists of several scalable dot-product attention modules that calculate different attention matrices such that the MHA can concentrate at once on several sections of the series. Since the series is a matrix, an MHA module can calculate scores on a simultaneous basis at all periods. Multi-headed attention blocks is shown in

Figure 4(a), which calculates several scaled dot attentions for each series of transformer models, as shown in Figure 4(b).

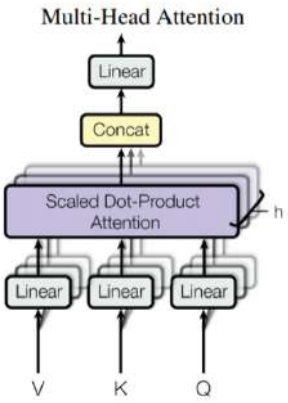

Figure 4 (a). Multi-Headed attention blocks.

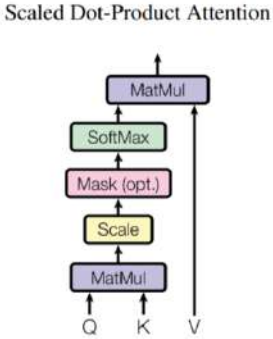

Figure 4 (b). Multiple scaled dot-product attentions block.

The development of the Attention mechanism [7, 55] improved the accuracy on multiple tasks including KWS [30]. Similar to human listening attention, the attention mechanism can identify the location of the query in the model to select the speech parts which are more likely to contain the keyword while ignoring the unrelated parts. It is computed between acoustic hidden features and keyword embedding. Assume the acoustic hidden features as $h_1, h_2 \dots h_T$ and keyword embedding as w_N the attention is computed as follows.

The scalar score is firstly learned in Equation (3).

$$e_t = w_N^T h_t \tag{3}$$

Then, the normalized learned weight α_t using these scalar scores is computed.

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{T} \exp(e_j)} \tag{4}$$

From the attention weights, context vector from acoustic representations is finally computed.

$$C = \sum_{t=1}^{T} \alpha_t h_t \tag{5}$$

3.6 Evaluation Metrics

The common confusion matrix in KWS is shown in Table 1.

Table 1. Confusion matrix in KWS.

		True condition	
		Keyword present	**Keyword absent**
Predicted condition	**Keyword detected**	True positive	False positive
	Keyword not detected	False negative	True negative

The recognition accuracy is the prioritized matric, which is measured as the percentage of the number of true positives to the total number of positives, as shown in Equation (6).

$$accuracy = \frac{\sum ture\ positive}{\sum (ture\ positive + false\ negative)} \tag{6}$$

Receiver operating characteristic (ROC) curves can be used as the second metric, where x and y-axis represent false alarm rate (FAR) and false reject rate (FRR) respectively. The less area under ROC, the better performance the model achieves.

$$FAR = \frac{\sum false\ positive}{\sum (ture\ negetive + false\ positive)} \tag{7}$$

$$FRR = 1 - accuracy \tag{8}$$

Chapter 4 System Architecture

4.1 Standard End to End KWS System

The standard end to end model architecture of KWS consists of a feature extractor and a neural network based classifier, as shown in Figure 5. An input audio signal is fed into a speech feature extractor. Function extraction through Log-Mel bank energy filters (LFBE) or mel frequency cepstral coefficients (MFCCs) involves the translation of the spoken time domain signal into a collection of spectral frequency-domains coefficients that allow input signal dimension compression. The extracted features are classified by a neural network which produces the probabilities of the output classes. The model could be trained by using a loss function with an optimizer, such as the cross-entropy loss function with Adam optimizer. Overall, the definition of end-to-end model can directly outputs keyword detection with no complicated searching involved and no alignments needed beforehand to train the model.

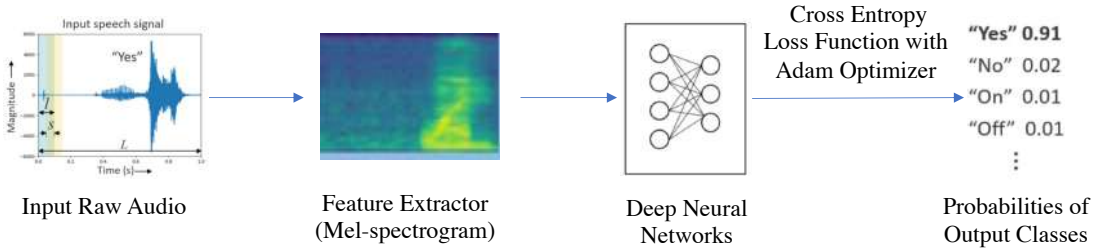

Figure 5. Standard end to end KWS system architecture.

4.2 Neural Network Architectures for KWS

4.2.1 VGG

VGG is the very deep convolutional networks proposed for large-scale image recognition by the visual geometry group of Oxford on ILSVRC 2014 [2]. The main work is to prove that increasing the depth of the network can affect the final performance of the network to a certain extent.

VGG network is a classification network composed of several convolutional neural networks. According to different network depth, it has four structures, VGG11, VGG13, VGG16, and VGG19 show in Table 2. VGG contains five groups of convolution operations as a five-stage convolution feature extraction. The number, 11, 13, 16 and 19 after VGG represent the number of convolution layers in the group of

convolution and full connection (FC) layers from 11 weighted layers (8 convolution layers and 3 FC layers) in column A to 19 weighted layers (16 convolution layers and 3 FC layers) in column E. After convolution, each group is pooled with a maximum of 2 × 2, and finally there are three fully connected layers.

Table 2. The network structure of VGG.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

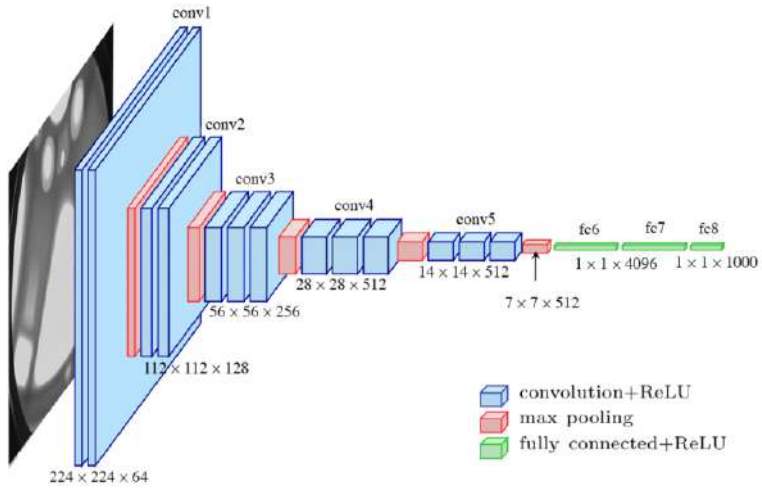

Figure 6. A visualized network structure of VGG 19 which has 16 convolutional layers followed by 3 fully connected layers..

4.2.2 Deep Residual Network (ResNet)

ResNet was proposed by Kaiming HE in 2015 for image classification [3]. In his experiments, it has been found that as the depth of the network increases, the accuracy of the network saturates or even decreases. The author suggests that this may be due to the gradient vanishing and exploding problem in deeper networks, which is difficult to achieve the convergence. In this paper, the phenomenon of deepening the network depth but decreasing the network performance is called degradation problem. The author thought of building identity mapping to solve this problem to achieve the goal of increasing the number of network layers while the training error does not increase. For a stacked layer structure, when the input was x and the output was supposed to be H(x). By making H(x)=F(x)+x, the network only needs to learn to output a residual F (x) = H (x)-x. This residual learning is much easier than learning original feature H (x) directly. When the residual is 0, the stack layer only does the identity mapping, at least the network performance will not decline. In fact, the residual will not be 0, which will make the stack layer learn new features based on the input features to achieve better performance. The structure of residual learning is also called shortcuts or skip connection as shown in Figure 7.

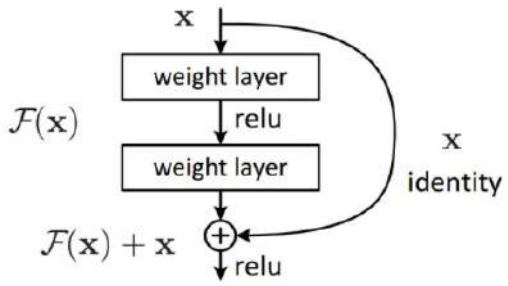

Figure 7. The residual learning unit.

ResNet has five main forms: Res18, Res34, Res50, Res101, Res152. Due to the constrain of small storage of the KWS system, only Res18, Res34 and Res50 would be applied. Each network consists of three main parts: input part, output part and intermediate convolution part. The middle convolution part includes four stages from stage 1 to stage 4 as shown in Figure 8. Although there are many varieties of ResNet, they all follow the above structural characteristics. The main difference between networks lies in the difference of block parameters and number of intermediate convolution parts. The middle convolution part is mainly the blue box part in Table 3. Information extraction is realized by stacking 3 × 3 convolutions. [2, 2, 2, 2] and [3, 4, 6, 3] in the red box represent the number of repeat stacks of blocks.

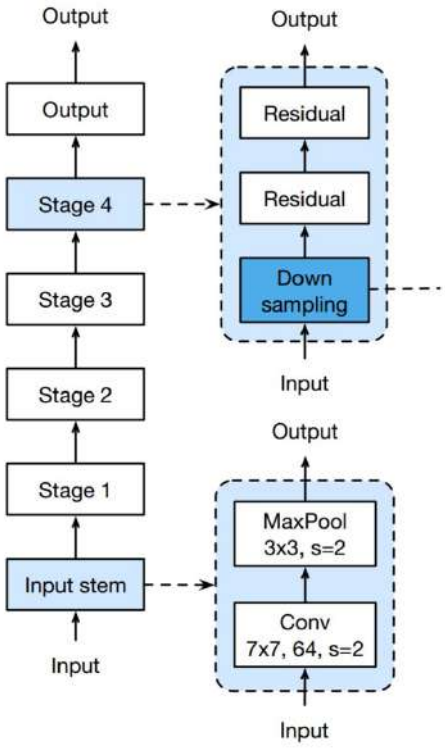

Figure 8. Overview of ResNet structure.

Table 3. The network structure of Res18, Res34, Res50, Res101, Res152.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

4.2.3 WideResNet（WRN）

Although the network continues to develop to a deeper level, sometimes in order to increase a small amount of accuracy, it requires to double the number of network layers, which reduces the reuse of features and reduces the training speed. The deeper the layers, the lower the reuse rate of its features. On this basis, Sergey zagoruyko and

Nikos komodakis proposed wideResNet [4], which mainly widens the output features when training features, and then adds dropout between convolutional layers to replace the batch normalization in ResNet.

The network structure is shown in Table 4. The network width is determined by the factor k. When k is 1, the number of convolution kernels is equal to ResNet. The larger k is, the wider the network is.

Table 4. The network structure of wide residual network

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	[8 × 8]

4.2.4 Densely Connected Convolutional Networks (DenseNet)

In 2016, Gao Huang of Cornell University and Zhu Liu of Tsinghua University proposed a new architecture called DenseNet in their paper densely connected revolutionary networks [5]. Different from ResNet short connecting the output and input to form a residual structure, DenseNet parallels the output and input to achieve the fact that each layer can directly obtain the output of all previous layers.

The traditional forward propagation network structure can be seen as the transmission of state in different layers. Each layer reads the state information and writes it to the next layer. In this process, the state will change, but at the same time, the information that needs to be saved will continue to be transmitted. There are L connections in the layer L, which is a one-to-one mode. In ResNet, a short-circuit connection is built between each layer and a previous 2-3 layers through element level addition, as an example shown in Figure 9. The information is explicitly saved by these shortcuts.

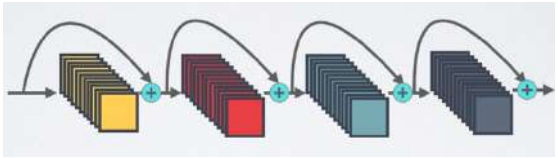

Figure 9. Elementwise addition in ResNet

However, DenseNet is a dense convolutional neural network. The structure of DenseNet is shown in Figure 10. Each layer with the same feature map is concatenated with all previous layers as its additional input in the channel-wise dimension. In other words, one layer is connected with all other layers, and therefore there are $\frac{L(L+1)}{2}$connections in layer L. The purpose of this is to ensure that the information flow between the layers is maximized, to realize feature reuse and improve efficiency.

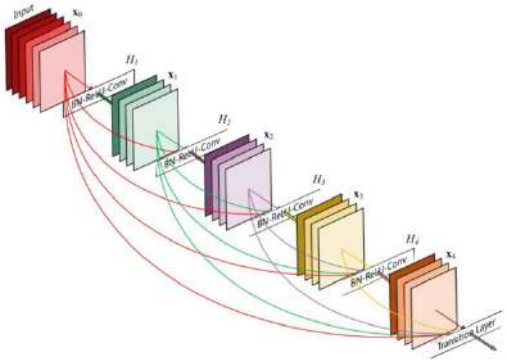

Figure 10. An example of 5-layer channel-wise concatenation dense block.

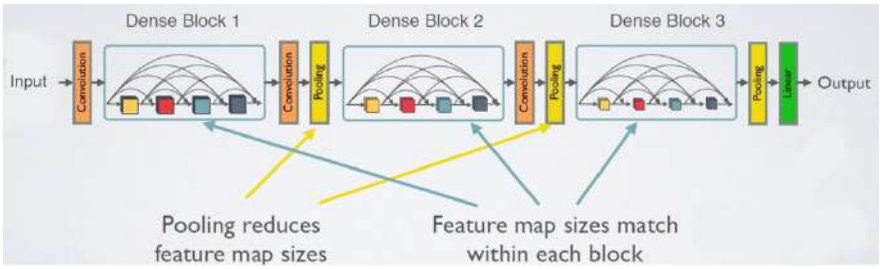

Figure 11. DenseNet network structure.

The network structure of DenseNet is mainly composed of dense block and transition, as shown in Figure 11. DenseNet consists of three dense blocks, and the feature graph sizes of each module are 32×32, 16×16, and 8×8. Each dense block has the same number of layers. All 3x3 convolutions adopt padding equals to 1 to keep the size of feature graph unchanged. In dense block, the same size of feature graphs of each layer can be concatenated on the channel dimension. k is a hyperparameter named growth rate representing the number of feature maps output by each layer in each dense block. Generally, better performance can be achieved by using smaller k such as 12. Assuming that the channel number of the input layer's feature graph is k_0, then the channel number of the l layer's input is $k_0 + k(l-1)$. Therefore, with the increase of the number of layers, even if k is set smaller, denseblock's input will be relatively large due to feature reuse. Only k features of each layer are unique. In addition, a 1×1

convolution is added in front of each dense block, which is the so-called bottleneck layer. The purpose is to reduce the number of input feature maps, the dimension and computation. Furthermore, the transition layer connects two adjacent dense blocks and reduces the size of the feature graph by using a 1×1 convolution and a 2×2 average pooling.

This dense connection mode, at first glance, seem to be cumbersome and may introduce many parameters. In fact, on the contrary, DenseNet requires fewer parameters than traditional convolution network because dense connection brings feature reuse, and it does not need to relearn redundant feature graphs. In addition, the directly concatenated feature graphs from different layers brings rich feature information. It improves the flow of information and gradient in the whole network. The dense connected structure enables each layer to obtain the gradient directly from the loss function and the original input signal, which is beneficial for training deeper networks.

There are three common DenseNet structures with batch normalization, {L=100, k=12}, {L=250, k=24}, {L=190,k=40. L refers to the total number of network layers (network depth) and k refers to growth rate. Compared with ResNet, the network depth of DenseNet cannot be too deep since deeper DenseNet consumes much GPU memory. Therefore, due to the storage and computation constrain of the application of KWS on small foot-print device, only {L=100, k=12} is applied.

4.2.5 Dual Path Networks (DPN)

ResNet directly elementwise adds the input to the output of convolution, while DenseNet concatenates the output of each layer to the input of each subsequent layer. In DPN, high order RNN structure (HORNN) links DenseNet and ResNet together and proves that DenseNet can extract new features from the previous level, and ResNet is essentially the reuse of the extracted features from the previous level. DPN is the combination of the advantages of these two structures.

Figure 12 (a), (b) and (c) demonstrate that Residual networks is actually a special case of densely connected networks. Figure 12(a) is the standard ResNet. Figure 12(b) is the DenceNet and concatenation was represented in the addition form. Figure 12(c) reformulates the form of (b) and the addition is written as an independent main branch. The significant difference between (b) and (c) is that the first 1×1 convolution in the

block of Figure 12(b) is shared at different times. In other words, the green and orange arrows in (b) can be different each time a new feature is generated. However, in Figure 12(c), the green and orange arrows would only appear once in the residual block. They cannot not be changed after being added to the main branch. This is the meaning of with shared connections.

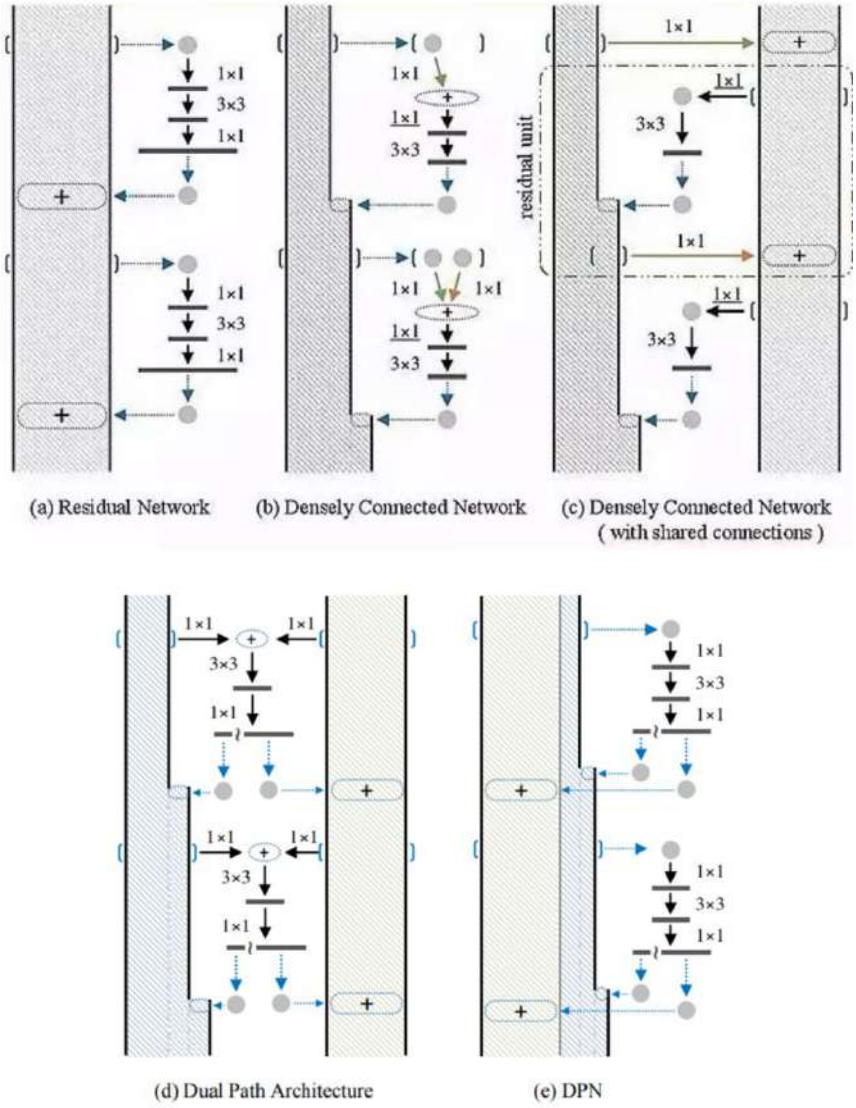

Figure 12. The comparison between ResNet, DenseNet, and DPN structure.

Figure 12(d) illustrates that the network structure of DPN with DenseNet and ResNet combined. The output of the last 1×1 convolution in the structure is divided into two parts. Figure 12(e) is the real network structure of DPN. The difference from (d) is that DenseNet and ResNet branches share the first 1x1 convolution. To be specific, the DPN92 is applied in KWS problem and its detail network structure is shown in Table 5.

Table 5. The network structure of DPN.

stage	output	DPN-92
conv1	112×112	7×7, 64 stride 2
conv2	56×56	3×2 max pool, stride 2
		$\begin{bmatrix} 1\times1, 96 \\ 3\times3, 96 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 192 \\ 3\times3, 192 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 384 \\ 3\times3, 384 \\ 1\times1, 1024 \end{bmatrix} \times 20$
conv5	7×7	$\begin{bmatrix} 1\times1, 768 \\ 3\times3, 768 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool fully connected layer softmax

4.2.6 Multi-Head Attention Based Network

Most of the current state-of-the-art keyword spotting architectures are present in the work of Rybakov et al. [1], with the best model to date being the Bidirectional GRU-based Multiheaded attention RNN. It takes a spectrum in Mel scale and complements it with a series of 2D convolutions. The Audio Data is then used with two two-way long-term relationship to record two-way GRU layers. The function in the middle of the bidirectional GRU output series is projected with a dense layer and is used as a multi-head (4-head) query vector. At present, the GRU layer output is sampled, projected with a thick layer to distinguish which portion of the audio is more significant and is used as question vectors. The weighted average bidirectional GRU performance (by attention) is ultimately processed by a series of completely connected classification layers.

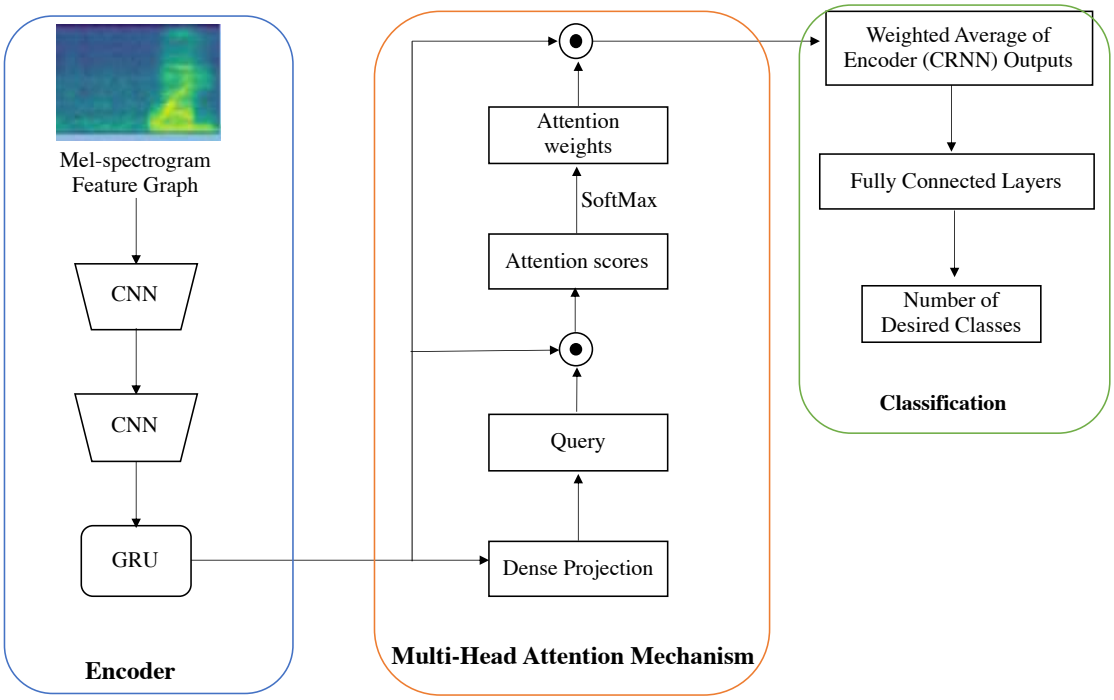

Figure 13. Multi-head attention network structure in KWS

4.3 Loss Function

4.3.1 Cross Entropy Loss

With regard to multi-class classification problem, the network is generally trained using standard multi-class cross entropy loss with L2 regularization. Its objective function is shown in Equation (9).

$$J(\theta) = -\sum_{i=1}^{N}[y_i log\hat{y}_i + (1-y_i)\log(1-\hat{y}_i)] + \lambda\|\theta\|_{L2} \quad (9)$$

where θ are the parameters of the network,

$\hat{y}$ is the output of the network,

$\|\theta\|_{L2}$ is the L2 regularization and λ is the weight of the regularizer.

4.3.2 Triplet Loss

Triplet loss is a loss function of deep learning, which is mainly used to train samples with small differences, such as faces. Also, triplet loss is often used in embedding tasks, such as text and image embedding. The loss function of triplet loss is shown in Equation (10).

$$L = \max(d(a,p) - d(a,n) + margin, 0) \tag{10}$$

d: the function to calculate distance

a: anchor

p: positive

n: negative

margin: a constant ≥ 0

The input is a triple, including anchor, positive and negative. By optimizing the distance between anchor and positive is less than that between anchor and negative, the similarity between samples is calculated. The final optimization goal is to shorten the distance between a and p, and to extend the distance between a and n.

The samples can be divided into three categories, easy triplets, hard triplets and semi-hard triplets.

(a) Easy triplets

L=0

$d(a,p) + margin < d(a,n)$

In this case, there is no need to optimize. The natural distance between a and P is very close, and the distance between a and N is very long,

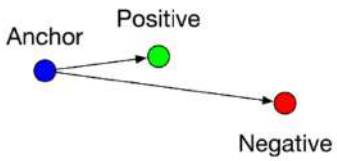

Figure 14. An example of easy triplets.

(b) Hard triplets

$L > margin$

$d(a,n) > d(a,p)$

The distance between a and n is close, and the distance between a and p is far. In this case, the loss is the largest and needs to be optimized, as shown in the figure below.

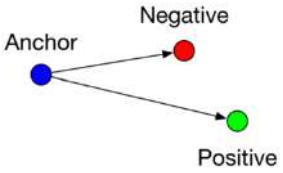

Figure 15. An example of hard triplets.

(c) semi-hard triplets

$L < margin$

$d(a,p) < d(a,n) < d(a,p) + margin$

The distance between a and p is closer than that between a and n. Although there is a loss, it is smaller than that of hard triplets, which also needs to be optimized, as shown in Figure 16.

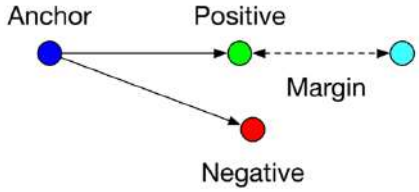

Figure 16. An example of semi-hard triplets.

In theory, hard triplets training model is the best since the model can have good learning ability. However, due to the existence of margin, this kind of sample model may not fit well, and it is difficult to train. Generally, semi hard training model is the best choice in practical use since the model is easier to learn the differences between the samples, and therefore easier to converge.

Triplet loss was proposed in FaceNet by Google [9]. In this paper, each image is transferred to the same convolutional neural networks with the same weights to calculate the embedding of each image. Similarly, in KWS architecture triplet-loss based embeddings could be generated after convolutional neural networks.

4.3.3 Proposed Loss Function

In Chapter 4.1, the standard end to end KWS System would be introduced. From Figure 5, it can be seen that a SoftMax layer that outputs input keyword recognition results is the last layer of the original KWS model. With the cross-entropy loss between the prediction results and soil reality, the loss function for KWS training is created.

Inspired by triplet loss, the loss feature is designed to minimize the sample distance from one keyword and increase the sample distance from various keywords. In the updated KWS model, the SoftMax layer is omitted, and the output of the second layer is a fixed-dimensional vector that maps the function of the input keyword into a fixed-dimensional space instead of direct prediction effects with a soft max layer. This fixed-

dimension vector can be compared with the template keyword feature representation obtained by Equation (11).

$$c = \frac{\sum_{i=1}^{K} e_{ti}}{K} \tag{11}$$

where c is the centroid of the keyword representing as the template keyword feature representation,

K is the number of samples of the keyword,

e_{ti} is the output vector of each sample of the keyword.

The similarity matrix between sample's output vector and the centroids can be computed by the cosine similarity value. Assume $S_{c:keyword}$ as the similarity matrix between the sample output vector and the centre vector of the keyword and $S_{c:non-keywords}$ as the similarity matrix between sample's output vector and the centroids of the non-keywords. As for $S_{c:keyword}$, the bigger the value of the cosine, the closer the sample is to the keyword. The loss function can be calculated in Equation (12).

$$L = log \sum_{i=1}^{N} \exp(S_{c:keyword}\ ) - S_{c:non-keywords} \tag{12}$$

Chapter 5 Experiments

5.1 Speech Command Dataset

Version 1 (V1) of the Command Speech Dataset [8] includes 30 commands made up of 64,727 utterances spoken by 1881 speakers, respectively. A single utterance, captured in wave format at 16 kHz, was used for each audio length for one second. In addition, the dataset provides six types of noises such as running tap, doing dishes, white noise, pink noise, etc. A few noises are recorded in real time while others were computer-generated. Table 6 presents the descriptions of orders and noises. These data sets were collected and primarily supported by European citizens inside a managed climate.

Table 6. Google Speech Command Dataset Version 1 (V1).

Command	Noise
core words down, eight, five, four, go, left, nine, no, off, on, one, right, seven, six, stop, three, two, up, yes, zero	doing_the_dishes (1 min 35 s) dude_miaowing (1 min 1 s) exercise_bike (1 min 1 s)
auxiliary words bed, bird, cat, dog, happy, house, marvin, sheila, tree, wow	pink_noise (1 min 0 s) running_tap (1 min 0 s) white_noise (1 min 1 s)

The neural network models are trained to classify the incoming audio into one of the 12 labels with 10 keywords - "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", along with "silence" (i.e. no word spoken) and "unknown" word, which is the remaining 20 keywords from the dataset. The dataset will be divided into 80:10:10 training, validation and test, thus ensuring that the audio clips are kept in the same collection by the same user.

5.2 Data Augmentation

5.2.1 Time shifting

Time shifts are added to each study with a 50 per cent possibility. The increased collections are up to 20 per cent of the initial duration moved forward or backward in time. Perhaps, this may lead to a partial decrease in the utterance of the sample, but the chances are insignificant. This strategy to increase the model is designed to help the

interpretation of words more time-invariant, since they can appear anywhere in model learning.

5.2.2 Amplitude Scaling

The amplitude of samples is 50% probability randomly scaled with the range from -50db to 50db.

5.2.3 Mixing with Noise

Mixing the samples with noise is introduced at a probability of 50 percent. There is noise produced in the same way as silence is generated when up to two samples are scaled and added together from the background audio of the dataset. The initial input, which was scaled from 75 and 125% of the original amplitude, was then applied to this noise. The addition of noise can help the model differentiate better from data pertinent information.

5.3 Training Details

The input to the model is the raw WAV data with original sampling rate of ∼16 kHz. In the stage of feature extraction, Mel-scale spectrogram is computed using 80-band Mel scale, 1024 discrete Fourier transform points and hop size of 128 points (∼8 s).

With a batch size of 512, the models are trained in PyTorch for 50 iterations with the Adam optimizer [153] and initial learning rate of 10^{-4}. L2 regularization is set to 10^{-4}. The ReduceLROnPlateau in PyTorch is applied as the learning rate scheduler to adjust the learning rate when the loss of the verification set no longer decreases. The trained models are evaluated based on the classification accuracy on the test set.

5.4 Experimental Results

5.4.1 Evaluation of Different KWS Models

The popular neural network architectures, VGG, ResNet, WideResNet, denseNet, DPN and multi-head attention, are implemented. The result is shown in Table 7.

In VGG, deeper network achieved better performance whereas in ResNet, simply increasing the depth of the neural network obtained worse performance, brought

difficulties to the training and caused non-convergence problems. In other words, despite the constraints of numerical computing and memory, networks with multiple filters and less layers achieved greater accuracy than networks with multiple layers and fewer filters. The first layer detects the characteristic patterns of input voice features and each following convolutional layer detects patterns in the previous layer patterns, introducing an abstraction degree. Therefore, an explanation of the effects of the grid quest may be that if the network has enough abstraction levels (layers), the number of different patterns required for the spoken contents (filter numbers) at each abstraction level will be very small. Even if the network with many layers, the network does not learn much useful information. On the other hand, the constrain of large filter size is to cause much trainable parameters. Despite VGG achieved higher accuracy than ResNet, it also consumes more computing resources and uses more parameters.

The best KWS models is multi-head attention-based network. It achieves a balance between memory and operations while still achieving good accuracy.

Table 7. Accuracy results on 12 commands from Google Speech Command Dataset V1.

Model	Accuracy (%)	Trainable Parameters (M)
Multi_Head Attention	97.60	0.84
WideResNet_28_10	97.49	36.5
VGG19_bn	96.98	39.0
VGG16_bn	96.79	33.7
VGG13_bn	96.54	28.3
DPN_92	96.52	34.2
VGG11_bn	96.38	28.2
ResNet18	94.60	11.2
DenseNet_bc_100_12	94.56	0.77
ResNet34	92.95	21.3
VGG11	91.76	28.2
ResNet50	89.68	23.5

*WideResNet_28_10 means the WideResnet network depth is 28 and widen factor is 10

*VGG_bn means VGG network with batch normalization

*DenseNet_bc_100_12 means the Densenet network with bottleneck layer and translation layer, network depth 100 and growth rate 12.

*DPN_92 means the depth of dual path network is 92.

5.4.2 Evaluation on the Proposed Loss Function

These models have been updated to identify the user-defined keywords by extracting the layer SoftMax and compare the performance vector of the keyword to each keyword's prototype.

The proposed loss function is applied on multi-head attention model as it shows relatively good performance. Table 8 shows the comparison of recognition accuracy between using cross-entropy and the proposed loss function. From Table 8, it can be seen that with the new loss function, the recognition of KWS system increases significantly by around 0.6% for the models.

Table 8. The accuracy of multi-head attention model with cross entropy loss and proposed loss.

Model	Cross Entropy	Proposed Loss	State of Art Result
Mult-Head Attention	97.60%	98.20%	98%

5.5 Demo

The demo is implemented to classify the 10 keywords "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go" on Python. It can detect whether there are keywords in the speech in real time via the local device microphone. If there is a keyword, it will be highlighted in the pie chart, as shown in Figure 17. Otherwise, it would be treated as unknown.

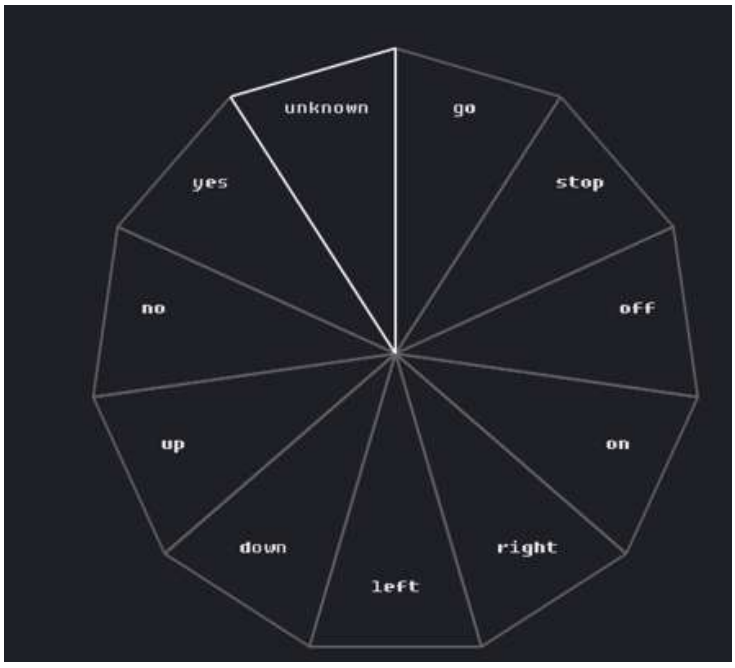

Figure 17. The demo of the KWS system.

5.6 Possible Improvements

5.6.1 Evaluating Models on FRR and FAR Matric

Although our model is close to 100% in accuracy, we can try to improve my model in another indicator. It is important to keep the count of false alarms as low as possible

for devices that are still in use. In this case, FRR and FAR can be used to replace accuracy as the prioritized evaluation metric. The ratio of negative to positive samples in training, i.e. utilizing more 'unknown' and 'silence' samples, could become an optional tool for reducing FPR and increasing the true negative rate. In other machine learning detection tasks this has been shown to be an efficient approach. The FPR may also be reduced by creating a failure feature for the optimizer during training that penalizes errors that trigger false alarms rather than missing errors.

5.6.2 Reducing Model Size by Teacher-Student (T/S) Learning

In [58], a typical DNN model was proposed to compact the teacher-pupil (T/S) learning. Learning is the soft mark created by the instructor paradigm as a goal of student learning for a student classification instruction. The idea of "T/S learning" has been generalized to "the concept of information distillation"[59] by combining cross-entropy classification with the traditional cross-entropy classification workout using soft labeling and the single hot vector as the objective. As the regularization concept for the normal cross-entropy classification instruction, the soft objective for information distillation.

The constructed structures are generally an ensemble of gigantic models with many deep models. Unable to deploy such a device in real-time. In this case, learning or the distillation of information offers a good way to achieve a compact model with strong modelling capacity. T/S Compression has been a most recent achievement in producing a CTC model [60] with almost the same KWS, but with just a 1/27 footprint of a massive KWS CTC model.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Various Neural Network Architectures in including VGG, Deep Residual Network (ResNet), WideResNet (WRN), Densely Connected Convolutional Network (DenseNet), Dual Path Network (DPN) and Multi-head attention Based network were trained and compared. In addition, inspired by Triplet loss, a novel loss function is proposed and implemented on Multi-head Attention Based Network. The experimental results show that the Multi-head Attention Based Network achieved the best accuracy, 97.6% and with the proposed loss function, the accuracy is further improved to 98.2%, which matches or exceeds the state-of-the-art result.

6.2 Future Work

The testing limit has been moved from a near-conversation microphone to a distant microphone and is motivated by greater user interaction with smartphones without the use or use of a close-conversations microphone. For instance, Amazon's Echo and Google Home have been deployed worldwide in many families. Many problems are now surfaced by close-speaking microphones as microphones are used further apart. This is because the voice signal power is very poor when reaching the microphones in the far-flung situation. In comparison, interfering sounds from other speakers such as background noise, reverb and voice become so distinct that they can no longer be overlooked.

Although many of the keyword spotting techniques for microphones can be used directly on far-flung microphones, these techniques show less efficiency in a remote recognition scenario. Finally, we need to refine the entire pipeline, from audio capture (for instance, microphone array signal processing), to acoustic modelling and decoding, to solve the distant keyword spotting issue. There are many potential researches on this area in the future.

Bibliography

[1] Rybakov, O., Kononenko, N., Subrahmanya, N., Visontai, M., & Laurenzo, S. (2020). Streaming keyword spotting on mobile devices. arXiv preprint arXiv:2005.06720.

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

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[4] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[5] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[6] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., & Feng, J. (2017). Dual path networks. arXiv preprint arXiv:1707.01629.

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762.

[8] Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209.

[9] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[10] R. C. Rose, D. B. Paul, in International Conference on Acoustics, Speech, and Signal Processing. A hidden Markov model based keyword recognition system, (1990), pp. 129–1321. https://doi.org/10.1109/ICASSP.1990. 115555

[11] J. R. Rohlicek, W. Russell, S. Roukos, H. Gish, in International Conference on Acoustics, Speech, and Signal Processing,. Continuous hidden Markov modeling for speaker-independent word spotting, (1989), pp. 627–6301. https://doi.org/10.1109/ICASSP.1989.266505

[12] J. G. Wilpon, L. G. Miller, P. Modi, in [Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing. Improvements and applications for key word recognition using hidden Markov modeling techniques, (1991), pp. 309–312. https://doi.org/10. 1109/ICASSP.1991.150338. http://ieeexplore.ieee.org/document/150338/

[13] J. Junkawitsch, L. Neubauer, H. Hoge, and G. Ruske, “A new keyword spotting algorithm with pre-calculated optimal thresholds,” in Proceedings of the International Conference on Spoken Language Processing (ICSLP), vol. 4. IEEE, 1996, pp. 2067–2070.

[14] A. S. Manos and V. W. Zue, “A segment-based wordspotter using phonetic filler models,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2. IEEE, 1997, pp. 899–902.

[15] M.-C. Silaghi and H. Bourlard, “Iterative posterior-based keyword spotting without filler models,” in Proceedings of the Automatic Speech Recognition & Understanding (ASRU) Workshop. Citeseer, 1999, pp. 213–216.

[16] A. Tavanaei, H. Sameti, and S. H. Mohammadi, “False alarm reduction by improved filler model and post-processing in speech keyword spotting,” in Proceedings of Machine Learning for Signal Processing (MLSP) Workshop. IEEE, 2011, pp. 1–5.

[17] R. A. Sukkar and J. G. Wilpon, “A two pass classifier for utterance rejection in keyword spotting,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2. IEEE, 1993, pp. 451–454.

[18] M. Weintraub, “LVCSR log-likelihood ratio scoring for keyword spotting,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1. IEEE, 1995, pp. 297–300.

[19] Y. Benayed, D. Fohr, J.-P. Haton, and G. Chollet, “Confidence measures for keyword spotting using support vector machines,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1. IEEE, 2003, pp. 588–591.

[20] H. Ketabdar, J. Vepa, S. Bengio, and H. Bourlard, “Posterior based keyword spotting with a priori thresholds,” in Proceedings of INTERSPEECH. ISCA, 2006.

[21] R. A. Sukkar, A. R. Setlur, M. G. Rahim, and C.-H. Lee, “Utterance verification of keyword strings using word-based minimum verification error (WB-MVE) training,” in Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1. IEEE, 1996, pp. 518–521.

[22] M. G. Rahim, C.-H. Lee, and B.-H. Juang, “Discriminative utterance verification for connected digits recognition,” IEEE Transactions on Speech and Audio Processing, vol. 5, no. 3, pp. 266–277, 1997.

[23] G. Chen, C. Parada, G. Heigold. Small-footprint keyword spotting using deep neural networks, (2014). https://doi.org/10.1109/icassp.2014. 6854370

[24] K. Shen, M. Cai, W.-Q. Zhang, Y. Tian, J. Liu, Investigation of DNN-based keyword spotting in low resource environments. Int. J. Future Comput. Commun. 5(2), 125–129 (2016). https://doi.org/10.18178/ijfcc.2016.5.2. 458

[25] G. Tucker, M. Wu, M. Sun, S. Panchapagesan, G. Fu, S. Vitaladevuni. Model compression applied to small-footprint keyword spotting, (2016), pp. 1878–1882. https://doi.org/10.21437/Interspeech.2016-1393

[26] Y. LeCun, Y. Bengio, in Chap. Convolutional Networks for Images, Speech, and Time Series. The Handbook of Brain Theory and Neural Networks (Press, MIT, Cambridge, MA, USA, 1998), pp. 255–258. http://dl.acm.org/ citation.cfm?id=303568.303704

[27] McMahan, B.; Rao, D. Listening to the world improves speech command recognition. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.

[28] Salamon, J.; Bello, J.P. Deep convolutional neural networks and data augmentation for environmental sound classification. IEEE Signal Process. Lett. 2017, 24, 279–283.

[29] Jansson, P. Single-Word Speech Recognition with Convolutional Neural Networks on Raw Waveforms; Arcada University: Helsinki, Finland, 2018.

[30] de Andrade, D.C.; Leo, S.; Viana, M.L.D.S.; Bernkopf, C. A neural attention model for speech command recognition. arXiv 2018, arXiv:1808.08929.

[31] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390, 2017.

[32] Zhang, Y.; Suda, N.; Lai, L.; Chandra, V. Hello edge: Keyword spotting on microcontrollers. arXiv 2017, arXiv:1711.07128.

[33] Sun, M.; Raju, A.; Tucker, G.; Panchapagesan, S.; Fu, G.; Mandal, A.; Matsoukas, S.; Strom, N.; Vitaladevuni, S. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. In Proceedings of the IEEE Spoken Language Technology Workshop (SLT), San Juan, Puertico Rico, 13–16 December 2016; pp. 474–480.

[34] Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.

[35] Segal, Y.; Fuchs, T.S.; Keshet, J. SpeechYOLO: Detection and Localization of Speech Objects. arXiv 2019, arXiv:1904.07704.

[36] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection.In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.

[37] Tang, R.; Lin, J. Deep residual learning for small-footprint keyword spotting. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 5484–5488.

[38] Xiong, W.; Wu, L.; Alleva, F.; Droppo, J.; Huang, X.; Stolcke, A. The Microsoft 2017 conversational speech recognition system. In Proceedings of the 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 5934–5938.

[39] Sercan O Arik, Markus Kliegl, Rewon Child, Joel Hestness, Andrew Gibiansky, Chris Fougner, Ryan Prenger, and Adam Coates. Convolutional recurrent neural networks for small-footprint keyword spotting. arXiv preprint arXiv:1703.05390, 2017.

[40] Ming Sun, Anirudh Raju, George Tucker, Sankaran Panchapagesan, Gengshen Fu, Arindam Mandal, Spyros Matsoukas, Nikko Strom, and Shiv Vitaladevuni. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. In Spoken Language Technology Workshop (SLT), 2016 IEEE, pages 474–480. IEEE, 2016.

[41] Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoust. Speech Signal Process. 1980, 28, 357–366.

[42] Tara N. Sainath, Brian Kingsbury, Abdel-rahman Mohamed, and Bhuvana Ramabhadran. Learning filter banks within a deep neural network framework. Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013, pages 297–302, 2013.

[43] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[44] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[45] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Advances in neural information processing systems, pages 2654–2662, 2014.

[46] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6655–6659. IEEE, 2013.

[47] Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[48] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

[49] Montúfar, G., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. arXiv preprint arXiv:1402.1869.

[50] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[51] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[52] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[53] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[54] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11):2673–2681, 1997.

[55] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. [Online]. Available: http://arxiv.org/abs/1409.0473

[56] Z. Cheng, K. Huang, Y. Wang, H. Liu, J. Guan, S. Zhou, Selecting high-quality negative samples for effectively predicting protein-RNA interactions. BMC Syst. Biol. 11(2), 9 (2017). https://doi.org/10.1186/s12918-017-0390-8

[57] R. Kurczab, S. Smusz, A. J. Bojarski, The influence of negative training set size on machine learning-based virtual screening,. J Cheminformatics. 6, 32 (2014). https://doi.org/10.1186/1758-2946-6-32

[58] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria." in Proc. INTERSPEECH, 2014, pp. 1910–1914.

[59] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[60] J. Li, R. Zhao, Z. Chen et al., "Developing far-field speaker system via teacher-student learning," in Proc. ICASSP, 2018.