JACOBS
UNIVERSITY

Speech Command Recognition with Echo State Networks

by

Min Wu

Bachelor Thesis in Computer Science

Prof. Herbert Jaeger
(Bachelor Thesis Supervisor)

Date of Submission: May 17, 2019

Jacobs University — Computer Science and Electrical Engineering

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature	Place, Date
Min Wu	Jacobs University Bremen, May 17, 2019

[illegible]

Abstract

Due to the rapid development of mobile technology, interacting with devices using voice technology became very accessible. Applications such as Apple Siri, Google Assistant, or Amazon Alexa all use the speech command technology to give users a hands-free experience. Such technology includes combined effort from computer scientists, linguists, and signal processing experts. Keyword spotting (KWS), referring to the initial stage of responding to the user's voice command, is done locally on devices such as phone or tablets. Thus, minimal computation and low-latency are required.

Echo State Networks (ESNs), a special form of Recurrent Neural Networks (RNNs), provide a powerful technique for modeling non-linear dynamic systems. The fast training and computationally inexpensive characteristics of ESN make it very suitable for modeling stochastic processes. ESNs have already been successfully applied in domains such as time-series prediction, pattern generation, dynamic pattern recognition, and communication.

This guided research project aims to build an accurate, efficient, and robust speech command recognition system that is capable of detecting predefined keywords using an ESN based classifier.

Keywords - *Speech Command Recognition*, *Keyword Spotting*, *Echo State Networks*, *Automatic Speech Recognition*

Contents

1 Introduction

Speech recognition, also known as automatic speech recognition (ASR), is a field of study which aims to transform speech utterances into their corresponding text form, in word or letter sequence. ASR has always been a great interest for computer scientists, linguists, and signal processing experts for the development of human language technology (HLT). For example, areas of research include but are not limited to: speech understanding - comprehending the semantics of speech, spoken translation - translating a spoken language into corresponding foreign text, and keyword search - finding a match for specific words in the speech.

ASR has a long history with several waves of major developments over the past decades, especially in the following five areas: infrastructure, knowledge representation, models and algorithms, search, and metadata [1]. Due to the long history and the dense research in the field, it is difficult to give a summary of ASR in this thesis. Nevertheless, the most important shifts in the development of ASR are listed. The first significant transformation happened in the early 1970s. Instead of using naive pattern recognition by evaluating the spectral distance between sound labels, ASR began to utilize statistical approaches, especially hidden Markov models (HMMs) [1] [2]. Rabiner wrote a comprehensive tutorial to explain the richness in the mathematical structure of HMM, which became widely used to understand how such models could be applied [3]. As a result, the establishment of such a statistical approach enabled researchers to combine different sources of knowledge, such as phonetics, words, syntax, and semantics, and treat them in a unified probabilistic manner [2]. Gaussian mixture model-based HMM (GMM-HMM) systems stayed as the state-of-the-art for a lengthy period around 30 years [2]. The second significant shift happened more recently. Starting from 2010, after discovering remarkable improvements that were gained by replacing GMMs with deep neural networks (DNNs), feed-forward DNN with multiple hidden layers of units became the new state-of-the-art of ASR systems [4] [5].

One of the many ASR applications exists in our day to day life. Interacting with mobile or embedded systems using voice technology has become increasingly popular during the past few years. The technology behind it considerably altered the way humans interact with machines. Related applications like Apple Siri, Google Assistant, or Amazon Alexa all use the speech command technology to give users a hands-free experience. Additionally, this technology could serve people with disabilities and be used while driving or during emergency situations. In essence, those systems ”wake up” after hearing certain keywords/phrases. The step of continuously listening from audio input and detecting trigger phrases to act upon is called keyword spotting (KWS). Since speech command recognition systems normally run on mobile or embedded devices, the initial detection for the start of interaction is always done locally, while further analysis and response computing is done through cloud-based services. Therefore, optimized sound recognition models that are fast enough to work in real time with low memory footprint, are much needed [6].

Recurrent neural network (RNN) is a class of biologically inspired neural networks. Unlike feedforward neural networks, where information flows only in one direction from one layer to another, in RNNs, recurrent connections between its neurons form a directed graph. As a result, RNNs can use their internal states as *memory* to process sequences of input and thus allow modeling dynamic temporal behaviors of the time series inputs.

Echo state networks (ESNs), belonging to the broader family of *reservoir computing* (RC), provide a powerful method for modeling non-linear dynamic systems. ESNs have a special approach of understanding and using RNNs, essentially separating the network into two parts: the recurrent part of the network (*reservoir*) and the readout layer that can be trained. As a result, the training of an ESN is simply a linear regression task. The ”conceptually simple” and ”computationally inexpensive” nature of ESN [7] makes it very suitable for modeling stochastic processes [8].

In the work described in this thesis, an ESN-based classifier is implemented with the aim of building a fast and accurate speech command recognition system that is capable of detecting pre-defined keywords. The report is structured as follows. Section 2 explains the research question and objectives behind the task. It describes the speech command recognition problem and the related work done before. Furthermore, it takes an in-depth look into an ESN, the model we are using to solve the speech command recognition task. Section 3 explains the investigation and the experiment setup used to tackle the task. Section 4 discusses the experiment results based on the defined evaluation criteria. Section 5 concludes the thesis with a restatement of the main aspects and an answer to the research question mentioned in the writing. The thesis report ends with suggestions for possible further investigations.

2 Statement and Motivation of Research

2.1 Isolated Word Recognition and Related Work

In this section, the isolated word recognition problem will be introduced. Relevant work that has been done before is stated, followed by defining the learning task.

In the classical approach of an ASR system, the components of human speech were intended to be understood. Phonemes, the smallest contrastive units of spoken words, can be used to differentiate words from a set of similar sounds [9]. For example, the word ”five” with three phonemes: ”f”, ”ay”, and ”v”, is different than ”dive” with phonemes ”d”, ”ay”, and ”v”.

In order to identify phonemes and reduce variability in speech (such as different speakers, emotions, background noises, etc), features are extracted from the raw speech signals. This process of extracting features is referred as the front-end of an ASR system. Acoustic observations are discretized into speech frames or windows of length between 5 to 100 ms, typically 25 ms. The characteristics of the speech signals are considered to be stationary within each such short speech frame. The following algorithmic computations are usually done for spectral analysis: Fast Fourier Transformation (FFT), logarithm calculations (log), Discrete Cosine Transformation (DCT), and Linear Discriminate Analysis (LDA). Cepstral based features such as Mel-frequency Cepstral Coefficients (MFCCs), Relative Spectral Transform - Perceptual Linear Prediction (RASTA-PLP), and through computing Linear Predictive Coding (LPC), are some of the most widely used features for ASR. Essentially, cepstral based features enable a frequency warping over the computed spectrum, similar to cochlea in human ears that does a frequency analysis [10] [11].

A traditional ASR pipeline includes three parts: firstly, an acoustic model with a hidden

Markov model (HMM) for each unit, which maps the acoustic features to the corresponding phonemes, a dictionary or pronunciation model provided by experts which maps possible words to their corresponding phonemes in sequence, and finally, a language model which computes word sequence probability [12].

Computational constraints and the amount of data available restricted the speed for ASR to make great progress. Chen et al. stated that a commonly used technique for KWS is the Keyword/Filler Hidden Markov Model, before neural networks became the state-of-the-art. An HMM is trained for each of the keyword, and a filler model HMM is also trained from the non-keyword segment of the speech signal. Although the Keyword/Filler HMM has a long history of more than two decades, it still remains very competitive. However, this method is very computationally expensive due to the requirement of Viterbi decoding [13].

Discriminative models based on large-margin formulation were also explored. Margin, the minimum distance between a data point and a decision boundary, is utilized for learning a classifier. Maximizing margins have shown to produce remarkable results for both classification and regression tasks [14]. Although improvements have shown when comparing this method to the HMM approach, the models tend to have relatively long latency, e.g. processing over the whole speech to find the keyword is required [13].

Recently, deep learning based approaches demonstrated performance improvements over the conventional machine learning methods for many different applications [5]. The neural networks built with memory capabilities have made speech recognition improve its accuracy by far [4].

In the work from Chen et al. from 2014, a deep neural network (DNN), trained directly to predict the keyword, outperformed the standard HMM based system under both clean and noisy conditions [13]. This is largely due to DNN's ability to model complex correlations in speech features. One year later, the same group of researchers extended the work and found a convolutional neural network (CNN) system that could outperform the DNN based approach under both clean and noisy conditions [15] [16].

CNNs perform better than DNNs for the KWS task mainly due to two reasons. Firstly, modeling local correlations in spectral representations between time and frequency can be done using CNNs, while DNNs ignore the input topology by possibly taking any order of the input without affecting the performance outcome. Secondly, CNNs can capture translational invariance (frequency shifts due to different speaking style) with far fewer parameters compared to DNNs for the same task. Consequently, CNNs have better performance with reduced model size comparing to DNN. CNNs now are the new state-of-the-art technique for the KWS task [15].

Nonetheless, reservoir based techniques were also used for solving the isolated word recognition tasks. For example, the Liquid State Machines (LSMs) outperformed HMM based recognizers with a strong noise-robustness [17]. Studies have also shown that a simple trained, computationally cheap Echo State Network with strong noise robustness is a great alternative for HMM-based isolated word recognition tasks [18]. Note that comparing to the current state-of-the-art deep learning techniques, the results using reservoir based techniques mentioned-above are outdated. Nevertheless, we re-consider a reservoir based technique and utilize the nature of its computational cheapness while comparing to deep learning.

In the work of this report, an ESN-based classifier is used to solve the speech command

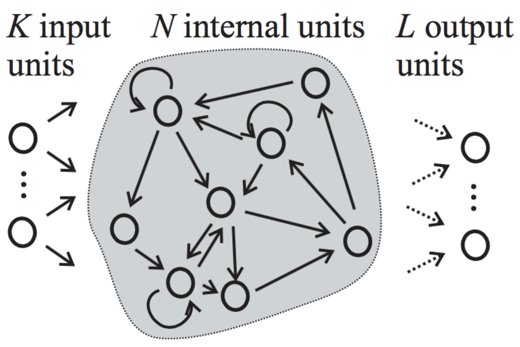

Figure 1: The basic structure of an Echo State Network. Image obtained from [8]

recognition task. Before diving into the research motivation, the general structure of an ESN is introduced below.

2.2 Echo State Networks: The General Structure

Inspired by biological brains, neural networks are computational models consisting of connected neurons. There are two major types of neural networks: recurrent neural networks (RNNs) and feedforward neural networks. RNNs form cyclic paths of synaptic connections, meaning the neurons can send feedback signals between each other, and thus form a directed cycle [19].

Echo State Networks (ESNs), based on the concept of RNNs, form a special approach of RNN. Different from the classical RNNs, the parameters of an ESN are essentially divided into two separate parts: (1) a RNN part (called *reservoir*) that is randomly initialized and fixed based on a few global parameters, and (2) a readout layer in which only the outputs weights can be trained. ESN has a very simple and fast training procedure and low computational cost, providing a powerful method for modeling non-linear dynamic systems. ESN has been successfully applied to various engineering problems, such as channel optimization, speech recognition, and prediction for chaotic Mackey Glass time series [20].

In this section, the general structure of an ESN will be given, and the mathematical equations for updating the reservoir and training of the readout layer will also be listed. The explanations will help to derive a general ESN model for time-series input sequences.

As seen in Figure 1, the core component of an ESN exists in the center, also called the reservoir. The reservoir is a randomly generated and fixed upon a few global parameters. It drives a non-linear high-dimensional expansion $x(n)$ from the input signal $u(n)$, acting upon each neuron within the reservoir. Additionally, it acts as a short term memory of the input $u(n)$, which makes an ESN suitable for solving temporal tasks [7] [8].

Combining both aspects mentioned above, the reservoir should have a sufficiently large signal space in $x(n)$, such that the desired output $\mathbf{y}^{target}(n)$ could be obtained by essentially calculating a linear combination from it [7].

The update functions for a reservoir are shown as follows:

$$\tilde{\mathbf{x}}(n) = tanh(\mathbf{W}^{in}[1;\mathbf{u}(n)] + \mathbf{W}\mathbf{x}(n-1)), \tag{1}$$

$$\mathbf{x}(n) = (1-\alpha)\mathbf{x}(n-1) + \alpha\tilde{\mathbf{x}}(n), \tag{2}$$

where $n = 1, ..., T$ represents discrete time, $x(n)$ is the vector of reservoir neuron activations and $\tilde{\mathbf{x}}(n)$ is its updates, $\mathbf{W}$ is the recurrent weight matrix and $\mathbf{W}^{in}$ is the input weights. α is the leaking rate, representing the speed of the dynamical reservoir updates [7].

The next step followed is to compute the output weights and obtain the output. A linear readout is given by the equation:

$$\mathbf{y}(n) = \mathbf{W}^{out}x(n), \tag{3}$$

where $\mathbf{y}(n)$ is the reservoir output and $\mathbf{W}^{out}$ is the output weights.

The goal of learning an ESN model is to make $\mathbf{y}$ as close to $\mathbf{y}^{target}(n)$ as possible, minimizing the error $\mathbf{E}(\mathbf{y}(n), \mathbf{y}^{target}(n))$, which is commonly a type of Mean Square Error (MSE) [7].

Ridge regression, which is more frequently used in practice, allows for regularization and avoids numerical instability by introducing a regularizer. The output weight $\mathbf{W}^{out}$ is then calculated by:

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \beta\mathbf{I})^{-1}, \tag{4}$$

where $\mathbf{Y}^{target} \in \mathbb{R}^{N_y \times T}$ are all $\mathbf{y}(n)$ with N_y being the number of outputs, $\mathbf{X} \in \mathbb{R}^{(1+N_u+N_x)\times T}$ represents the all [1;$\mathbf{u}(n)$; $\mathbf{x}(n)$] produced by presenting the reservoir with $\mathbf{u}(n)$ with N_u being the number of inputs and N_x being the reservoir size, β is the regularizer coefficient, and $\mathbf{I}$ is the identify matrix [7].

For solving a classification task like the speech command recognition problem, there are as many classes as the dimensions of the output. The $\mathbf{y}^{target}(n)$ is 1 if the corresponding class is classified as true, else 0. The class that input $u(n)$ belongs to is calculated by:

$$class(\mathbf{u}(n)) = \underset{k}{\operatorname{argmax}}(\frac{1}{|\tau|}\sum_{n\in\tau} y_k(n)) = \underset{k}{\operatorname{argmax}}((\sum \mathbf{y})_k) \tag{5}$$

where $y_k(n)$ is the *k*th dimension of the output $\mathbf{y}(n)$, τ is an integration interval (can be, for example, the length of the entire sequence $u(n)$), $\sum \mathbf{y}$ is the time-averaged $y(n)$ over τ [7].

2.3 Research Objectives

In this guided research experiment, the main objective is to develop an ESN classifier for the speech command recognition task, using the TensorFlow Speech Recognition Challenge dataset organized by Google on the Kaggle platform [21].

As stated above, a KWS system listens continuously from the input microphones. Rather than sending the data to a server continuously, it runs locally and responds to trigger phrases. From the KWS task perspective, a system that can react quickly to the initial acknowledgement is needed, even if the full response from the server is delayed. The application for such system would live on either mobile or embedded systems, which have limited battery lives and thermal constrains. Thus, a successful KWS system should not only be fast, but also efficient with low memory footprint [22].

ESNs have been successfully applied for speech recognition problems, for instance, the "Japanese Vowel" speaker identification task [23]. An ESN has several characteristics that make it a suitable time series (e.g. speech utterances) classifier: short-term memory capability to model time series data, conceptual simplicity for understanding, and control of the system dynamics by tuning global parameters (e.g. spectral radius, leaking rate, reservoir size, and input scaling). Combining such aspects, solving the KWS or the speech command recognition task using an ESN-based classifier seems reasonable.

Knowing the current state-of-the-art for solving such KWS task using deep learning, the use of ESN does not have full advantage. Thus, this guided research project does not aim to achieve the state-of-the-art results for the KWS task. The main objective of the research is to develop a fast-trained ESN based classifier, and see how close the outcomes can get to the current state-of-the-art results.

More specifically, the detailed objectives of the current research are as follows:

- to develop an ESN-based classifier for the speech command task using MFCC features.
- to make the model as accurate and robust as possible by tuning global parameters (spectral radius, leaking rate, reservoir size, and input scaling).
- to compare model performance by designing different formats of target outputs.
- to analyze model performance and identify the best model parameter configurations by measuring accuracy, precision, and recall scores for each model.
- to propose further improvements and investigation that can be done for better result outcomes.

3 Conducted Investigation

In this section, each component of the experiment, necessary for reaching the above-mentioned objectives, will be explained in detail, including the Speech Command dataset, data preprocessing process, feature extraction with MFCC, reservoir setup, and readout classification.

3.1 Dataset Description

The dataset that is used for this guided project is taken from the TensorFlow Speech Recognition Challenge hosted on Kaggle. The Speech Commands Dataset is developed by Google's TensorFlow and AIY teams. The first version, which is the one we are using, was released on August 3, 2017. It contains 64,727 one-second utterances from 1,881 speakers saying 30 different words. It is organized into 30 directories with the corresponding word labels. The 30 different words are categorized into 3 types: digits (0-9), commands that we want to identify, and auxiliary words. The goal of the learning task is to classify the audio files into either the ten keywords, unknown, or silence, meaning there are less labels that need to be predicted (n = 12) than the labels in the given training set (n = 30). The labels that should be classified are: *yes*, *no*, *up*, *down*, *left*, *right*, *on*, *off*, *stop*, and *go*. The ten auxiliary words: *bed*, *bird*, *cat*, *dog*, *happy*, *house*, *Marvin*, *Sheila*, *tree*, and *wow*, along with the ten digits should be classified as *unknown*. Everything else should be considered as *silence*.

In this dataset, no personally identifiable information was recorded from any contributors. Instead, each person has an unique and stable id which is included at the beginning of each file name. For example, the file path ”happy/3cfc6b3a_nohash_2.wav” indicates the word ”happy” was pronounced by the speaker with id ”3cfc6b3a”, and this is the third recorded utterance of that word by the same speaker [21] [22].

Each audio file in the data set is a one second long audio clip that is converted into a 16-bit PCM-encoded WAVE file at a 16,000 sample rate. After a bit of exploration, this dataset is not completely cleaned up to use. For example, a few files in the training dataset have duration way below one second. Some files are corrupt and only contain noise. Some audio files contain extremely low volume. For better performance outcome, one could manually double check the dataset by listening to each speech signal and filter out the clear outliers. In addition, there is no such file labeled as *silence*. Instead, the files *background_noises* are provided [22]. Thus, data preprocessing is needed to solve the above-mentioned issues.

3.2 Data Preprocessing

The *background_noises* files containing sounds captured from machinery and household activities, each lasting approximately 60 seconds long, are split up into one second audio fragments to match with other audio signals.

For modeling, the full dataset is divided into three parts: 80% training set, 10% validation set, and 10% test set, separated according to the lists of file names in *validation_list.txt* and *testing_list.txt*. In total, there are 51,407 speech sample for training, 6,838 for validation, and 6,878 for testing. The distribution of the different target classes after re-organizing the background noise files is shown in Figure 2 [21] [22].

In order to have faster modeling computation and optimization, 20% of the full dataset divided into training, validation, and testing data are used at first to find optimal parameters. Global parameters are firstly optimized using this subset of data for each model. Different ESN models are then compared using a 5-fold cross validation on the training and validation dataset. The performance of each set of global parameters is averaged over the 5 folds. Thereafter, the full dataset is used for the model with the discovered optimal global parameters. The model's performance is then calculated and recorded with a 2-fold cross validation.

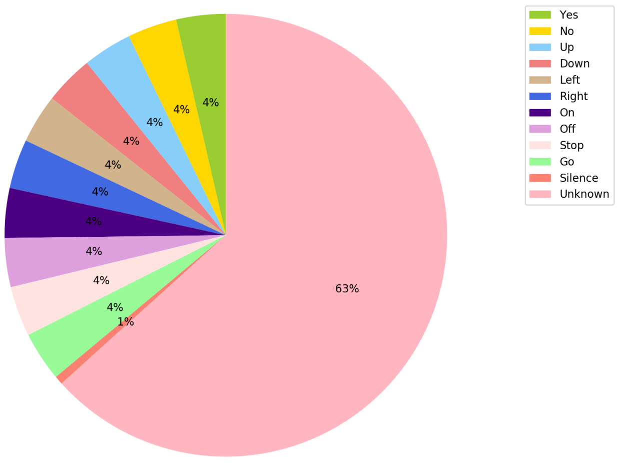

Figure 2: The (self-generated) distribution of speech command of the target classes in the full training dataset after splitting *background_noises* into one-second long files.

3.3 Feature Extraction with MFCC

The first step in any ASR system is to extract features. Only the key components of the audio signal that are good for detecting the linguistic nature of the content should be left in a compact representation, leaving noises, emotions, and other inefficient components behind.

For our current research, Mel-Frequency Cepstral Coefficients (MFCCs) are used. Calculating MFCCs is one of the most popular and dominant method to extract spectral features based on frequency using Mel scale [10]. MFCC is a representation of the real cepstral of a short-time windowed audio signal derived from the Fast Fourier Transformation. The difference from the real cepstral is that a nonlinear frequency scale (Mel scale) is used, which is very similar to the behavior of a human auditory system [10] [11].

The MFCCs are calculated as follows (also shown in Figure 3) [10] [11]:

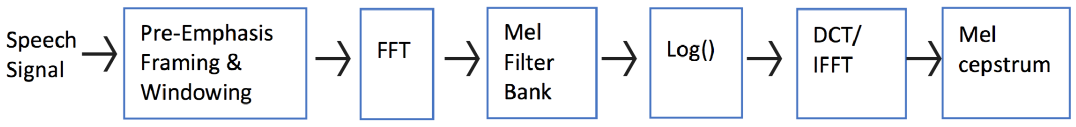

Figure 3: A self-generated diagram of the MFCC derivation process.

1. the raw audio data is windowed using a hamming window.
2. the Fast Fourier Transformation (FFT) is computed for each frame to get power spectrum.
3. Mel-scale filter bank, motivated by human ears, is applied, and the log_{10} of the resulting values is computed.

4. a Discrete Cosine Transformation (DCT) is applied to reduce the correlation among the individual features. The resulting outcome is the so-called cepstrum.

Additionally, MFCCs are very robust and reliable to variations according to speakers and recording conditions, making it a reliable feature for solving the speech command recognition task [11].

In the work of the thesis, a public MFCC extractor function is used from a python library called *python_speech_features*. Figure 4 gives a visualization of how the MFCCs of some clips within the speech command dataset look like. The speech files are sampled with a sample rate of 16 kHz. Each file is divided into 0.01 second frame, resulting in approximately 99 frames per file. Thus, each speech data roughly has a 99 $\times$ 13 dimension feature vector.

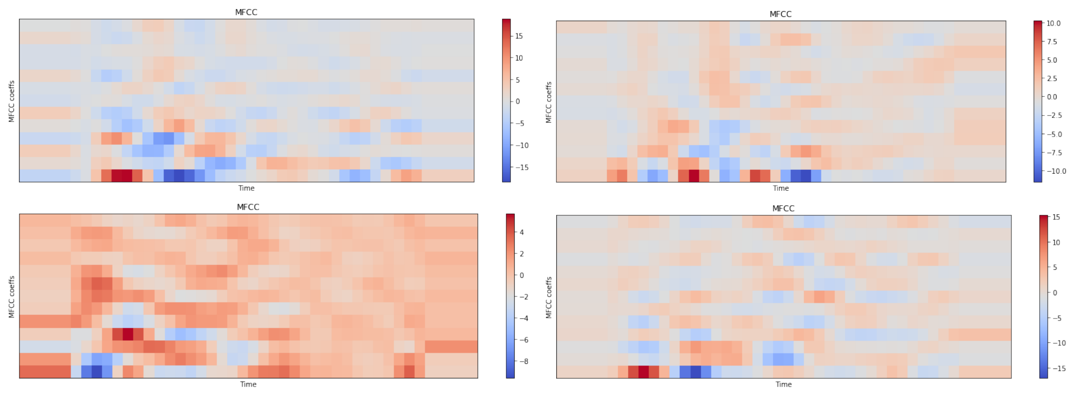

Figure 4: MFCC representations for audio clips selected from the speech command recognition dataset. Top left represents the word ”yes” in file ”yes/0a7c2a8d_nohash_0.wav”, Top right represents the word ”no” in file ”no/0a9f9af7_nohash_0.wav”, bottom left represents the word ”three” in file ”three/1a673010_nohash_0.wav”, and bottom right represents the word ”tree” in file ”tree/0fa1e7a9_nohash_0.wav”. Python code adopted to graph such figures are taken from [24].

3.4 Network Design

This section will be dedicated to explain the technical details for the ESN model setup and optimization, including the initial setup of the model, the method for collecting the neutral states' activations, global parameters tuning, readout layer training, and classification for output labels.

3.4.1 Reservoir Initialization

As mentioned in Equations (1) and (2), the weights $\mathbf{W}$ and $\mathbf{W}^{in}$ are initialized randomly. In this setup, they are centered around 0 with a uniform distribution between [-0.5, 0.5] as suggested from [7].

The $x(n)$ data from the beginning of the training procedure is usually discarded and not used for learning $\mathbf{W}^{out}$, due to being contaminated by the initial transients. The initial transients are the result of setting an arbitrary $x(n)$, commonly $x(n) = 0$. The unnatural starting activation states are normally not visited again once more data are inputted into the network. Therefore, hundreds of the $x(n)$ could be discarded. For short separated sequence classification tasks, like our speech command classification task, discarding the initial transients is necessary for each sequence. However, discarding information from each of those short sequences becomes very dangerous since we risk not keeping any of the time points. Thus, a default state is calculated and used to reset the state activation values every time before a new sequence of data is passed into the reservoir, making the classification task of each sequence independent [7].

In the model setup, a set of time-averaged activations $\frac{1}{T}\sum \mathbf{x}$ is calculated and used as the default state activation values for both training and testing, instead of an arbitrary $x(n)$. Thus, every frame from the obtained MFCC feature vector of each speech file in the training and validation datasets, is passed into the initialized network. For every frame or time point, the neuron state activation x is recorded. After all activations are collected, $\frac{1}{T}\sum \mathbf{x}$, or the ”neutral state”, is calculated by averaging over the length of total time points, T. For the training and testing step, the reservoir is set to the neutral states before each different audio input.

The global parameters: spectral radius, leaking rate, input scaling, and the regularizer coefficient are first tuned on the 20% subset of the entire dataset. Manual parameter selection is used to find an optimal set of global parameters. When tuning, one parameter is changed at a time for observing the effect each change of the parameter has. The parameters used are also documented during the process to avoid going in loops. Once an optimal set of global parameter is calculated, they are then applied to the entire dataset. Increasing the size of the reservoir is found to be beneficial when the size of training data increases. However, due to computation time constraints, the reservoir size is kept to 400 at most during this research experiment for the first model and 600 for the second model, which both be explained later. Consequently, better performance of ESN models could be improved by simply enlarging the reservoir size to a bigger number.

In addition, during the tuning process, a combination of a high valued spectral radius (between 0.9 and 1.0) and a low-valued leaking rate (below 0.1) led to the best model performance, while tuning other global parameters (input scaling and the regularizer coefficient) did not lead to noticeable performance difference. A small leaking rate implies longer short-term memory, thus more frames are remembered from the same audio sequence. A more intensive search can be utilized to locate a combination of better global parameters.

3.4.2 Ridge Regression

In order to calculate the output, $\mathbf{W}^{out}$ needs to be determined. One of the mostly frequently used and generally recommended method to learn the linear output weights from an ESN model is ridge regression, also known as regression with Tikhonov regularization (shown in Equation (4)).

The $\mathbf{X}$ in the ridge regression equation represents the collected activations produced by presenting the reservoir with $\mathbf{u}(n)$. Note that the states are only collected after an initial transient to avoid contamination as described in Section 3.4.1. In the setup, the initial transient is set up 20 for each audio sequence of length (almost) 100.

Regularization is used to avoid the danger of overfitting or feedback instability. Tuning the regularizer coefficient β can be done without rerunning the entire ESN model, since all the other variables in Equation (4) are already calculated and thus do not need to change [7].

3.4.3 Output Design & Classification

As mentioned in Section 2.2, for a classification task, the output dimension should be the same as the number of classes we want to classify, in our case, 12.

In the experiment, two designs of the target output have been implemented and tested.

Both designs follow the same principle: the dimension of each target output in $\mathbf{y}^{target}(n)$ is 1 if the corresponding class is classified as true, else 0.

The first design is for every frame in each audio data, the target output of the same dimension is produced as mentioned above. For example, the target output of an audio data belonging to the first class with 99 frames is a 12 $\times$ 99 matrix, where the first row of the matrix is 1, and all the rest entires are 0. The second design, on the other hand, only marks the last frame with the correct class label. In addition, only the last reservoir state will be recorded into $\mathbf{X}$. Following our previous example, for such an audio data input, the dimension of the target output will be 12 $\times$ 1 instead, with the first entry setting to 1 and the rest to 0.

In the second approach, the dimensions of $\mathbf{X}$ and $\mathbf{Y}^{target}$ decrease from the number of all the frames in the training data audio to the length of training data. Consequently, the computational time needed for the ridge regression step is much shortened.

Once all the outputs of frames in each data point are calculated, the class that each input $u(n)$ belongs to is calculated by:

$$class(\mathbf{u}(n)) = \underset{k}{\operatorname{argmax}}(\frac{1}{|\tau|}\sum_{n\in\tau} y_k(n)) = \underset{k}{\operatorname{argmax}}((\sum \mathbf{y})_k) \tag{6}$$

where $y_k(n)$ is the *k*th dimension of the output $\mathbf{y}(n)$, τ is the integration interval (in this case, the length of each input data), $\sum \mathbf{y}$ is the time-averaged $y(n)$ over τ [7].

Essentially, we are trying to find the class which produces the highest probability of the averaged output for each audio data.

4 Experiment Results & Encountered Difficulties

In this section, the criteria that will evaluate the model correctness and effectiveness will be presented, followed by the actual experiment outcomes. Some encountered difficulties during the experiment will be explained at the end.

4.1 Performance Metrics

This section discusses the criteria which are used to evaluate the performances of the ESN classifier used in our experiment. The already known state-of-the-art outcomes are also provided in this section.

For convenience, a few evaluation methods will be calculated for comparing between different literature results as follows [25]:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ is calculated for each class. The closer the value is to 1, the better the result of the model.
- $Precision = \frac{TP}{TP+FP}$ measures the proportion of the data points that our model indicates as relevant actually are relevant in each class.
- $Recall = \frac{TP}{TP+FN}$ measures the ability to find all relevant data points in each class.
- $Overall\ Accuracy = \frac{1}{k}\sum_{i=1}^{k} \frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}$ calculates the averaged accuracy of each class i.
- $Overall\ Precision = \frac{1}{k}\sum_{i=1}^{k} \frac{TP_i}{TP_i+FP_i}$ calculates the averaged precision across all classes i.
- $Overall\ Recall = \frac{1}{k}\sum_{i=1}^{k} \frac{TP_i}{TP_i+FN_i}$ calculates the averaged recall across all classes i.
- $F1\ score = 2 * \frac{Precision*Recall}{Precision+Recall}$ measures the performance of a model with the combination of precision and recall. A good F1 score indicates low false positives and low false negatives. A F1 score of 1 suggests a perfect model, while the model is not at all useful when the score is 0.

True positives (TPs) are data points classified as positive by the model that actually are positive or correct; true negatives (TNs) are data points classified as negative by the model that actually are negative; false negatives (FNs) are data points the model identifies as negative that actually are positive; false positives (FPs) are data points the model identifies as positive that actually are negative.

According to Kaggle, the competition results are evaluated by Multi-class Accuracy, which is simply the average number of observations with the correct label across all classes, or the $Overall\ Accuracy$ we defined above [21].

Training the default CNN on this dataset results in a Top-One score of 85.4% . If the classifier's top guess is correct, then it is defined that this observation is in Top-One [22] [15]. Essentially, a Top-One score for the ESN classifier will be the number of correct classification over the number of all testing cases, or simply the $Accuracy$ as defined above for all classes.

Moreover, by plotting a receiver operating curve (ROC), which calculates the false rejection (FR) rate per false alarm (FA) rate, the performance of the classification model can also be seen visually, the lower the FR per FA rate the better [15].

To summarize, the above-mentioned methods are used for comparing results internally between iterations of models, and externally between different literature reviews or Kaggle competition results.

4.2 Evaluation Results

As mentioned, to optimize computation time, 20% of the full dataset is firstly used to construct the model and tune for global parameters. Later, the full dataset is used for calculating the model performance.

4.2.1 Model 1 with Target Output Design 1

In this model, the first target output design is used. For each input data frame, the target output is set to 1 when the corresponding class is true, else 0.

- 5-fold cross validation on 20% of the entire dataset

	Accuracy	Precision	Recall
yes	0.67	0.76	0.62
no	0.27	0.42	0.51
up	0.41	0.49	0.53
down	0.28	0.35	0.53
left	0.43	0.48	0.56
right	0.59	0.58	0.57
on	0.38	0.48	0.54
off	0.65	0.63	0.53
stop	0.48	0.52	0.61
go	0.38	0.49	0.54
silence	0.74	0.98	0.61
unknown	0.80	0.76	0.54
Overall	0.55	0.58	0.56

Table 1: Performance matrices for each class averaged over 5 folds.

The leaking rate α for this model is 0.04, spectral radius $\rho(\mathbf{X})$ is 1, regularization parameter β is 0.1, and the size of the reservoir N_x is 250.

The same parameters listed above are used for the 2-fold cross validation on the full dataset.

- 2-fold cross validation on the entire dataset

	Accuracy	Precision	Recall
yes	0.85	0.74	0.85
no	0.42	0.58	0.42
up	0.52	0.51	0.52
down	0.34	0.53	0.34
left	0.52	0.59	0.52
right	0.61	0.60	0.61
on	0.48	0.65	0.48
off	0.75	0.61	0.75
stop	0.61	0.65	0.61
go	0.47	0.60	0.47
silence	0.98	0.67	0.97
unknown	0.91	0.62	0.93
Overall	0.62	0.61	0.62

Table 2: Performance matrices for each class averaged over 2 folds.

4.2.2 Model 2 with Target Output Design 2

In this model, the second target output design is used. The one-frame-only approach is used for each audio data. The entries in the one-frame target output are 1 where the corresponding class is true, all the others are 0.

- 5-fold cross validation on 20% of the entire dataset

	Accuracy	Precision	Recall
yes	0.68	0.66	0.44
no	0.25	0.27	0.51
up	0.49	0.48	0.54
down	0.37	0.34	0.46
left	0.47	0.50	0.52
right	0.67	0.62	0.56
on	0.46	0.46	0.49
off	0.61	0.65	0.53
stop	0.25	0.27	0.36
go	0.54	0.56	0.49
silence	0.98	0.97	0.98
unknown	0.79	0.78	0.60
Overall	0.54	0.55	0.54

Table 3: Performance matrices for each class averaged over 5 folds.

The leaking rate α for this model is 0.05, spectral radius $\rho(\mathbf{X})$ is 0.9, regularization parameter β is 0.3, and the size of the reservoir N_x is 500.

- 2-fold cross validation on the entire dataset

	Accuracy	Precision	Recall
yes	0.67	0.64	0.52
no	0.28	0.31	0.44
up	0.51	0.47	0.54
down	0.34	0.40	0.48
left	0.53	0.41	0.55
right	0.72	0.73	0.60
on	0.60	0.58	0.55
off	0.68	0.65	0.67
stop	0.38	0.38	0.40
go	0.54	0.59	0.57
silence	0.90	0.89	0.97
unknown	0.89	0.73	0.65
Overall	0.59	0.57	0.57

Table 4: Performance matrices for each class averaged over 2 folds.

4.3 Encountered Difficulties

While comparing to the current state-of-the-art result of the same task, a noticeable gap exists between the baseline model result and our experiment performance. In addition, the fact that the performance of the current state-of-the-art baseline model is only 85.4% shows that solving the speech command recognition task is not trivial. A short discussion for the difficulties experienced is shown below.

The construction for the ESN requires tuning global parameters for obtaining better performance. Not only it requires numerous trial and error experiments, but also a deep understanding of the nature of the research problem, since the structure of ESN is rather task specific [7]. The difficulty experienced in this experiment is not finding a method to locate an optimal set of global parameters for each model, since it is already a standard practice; instead, the difficulty lies in finding the optimal design of the entire ESN when computing such dataset (more than 6.5 million time points), especially under time constrained conditions. The alternative designs to what we have done now could be: altering different target outputs, selecting and using a subset of $\mathbf{X}$, incorporating multiple features, and many more. Thus, there are still innumerable models with different sets of parameters that we can build to improve model performance.

5 Conclusions

This section will conclude the thesis report by restating the main aspects of the writing.

Automatic speech recognition (ASR) is a dense field with a long history of research. Two influential improvements of ASR happened with the help of Hidden Markov Models and the current deep learning models. Among applications of ASR, keyword spotting is widely implemented in our daily life.

The main objective of this research project is to create an accurate, efficient, and robust speech command recognition system that is capable of detecting pre-defined keywords using an ESN-based classifier and MFCC features. The characteristics of an Echo State Network make it an excellent tool for modeling time series sequences such as speech sequence.

In the work of this thesis project, different designs of ESN models are explored and evaluated. The original objective of creating an accurate and efficient classifier is not quite achieved when comparing to the current state-of-the-art baseline model. However, there exist numerous ways of improving current research. Further investigation could be done when more time and computation power are obtained.

6 Further Investigation

This section ends the thesis report by proposing potential steps to optimize current research outcomes:

- cleaning the audio clips that are not in the close range of one-second long (either too long or too short). Audio files with heavy background noises should also be eliminated.
- silence removal for the parts of audio clips that do not contain any useful information.
- integrating segmented background noises randomly into the original speech command audio files with an adjusted lower volume, in order to have a more noise-robust model.
- using a collection of equally spaced snapshots of the state activation values for computing regression weights, instead of using the entire design matrix, as described in [23].
- additional features could also be explored.

References

[1] Janet M. Baker, Li Deng, James Glass, Sanjeev Khudanpur, Chin-Hui Lee, Nelson Morgan, and Douglas O'Shaughnessy. Developments and directions in speech recognition and understanding, Part 1 [DSP Education]. *IEEE Signal Processing Magazine*, 26(3):75–80, May 2009.

[2] Xuedong Huang, James Baker, and Raj Reddy. A Historical Perspective of Speech Recognition. *Communications of the ACM*, 57(1):94–103, January 2014.

[3] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[4] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep Learning. *Nature*, 521:436–444, May 2015.

[6] Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv:1804.03209*, April 2018.

[7] Mantas Lukoševičius. A Practical Guide to Applying Echo State Networks. *Neural Networks: Tricks of the Trade*, 7700:659–686, 2012.

[8] Herbert Jaeger. Short term memory in echo state networks. Technical Report GMD-Report 152, GMD-Forschungszentrum Informationstechnik, March 2002.

[9] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, November 1989.

[10] Urmila Shrawankar and Vilas M. Thakare. Techniques for Feature Extraction In Speech Recognition System : A Comparative Study. *International Journal of Computer Applications in Engineering, Technology and Sciences (IJCAETS)*, pages 412–418, 2010.

[11] Namrata Dave. Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition. *International Journal for Advance Research in Engineering and Technology*, 1:1–5, July 2013.

[12] Automatic Speech Recognition: State-of-the-Art in Transition: A Neural Paradigm Change? `http://online.kitp.ucsb.edu/online/hearing17/schlueter/pdf/Schlueter_Hearing17_KITP.pdf`, June 2017. Accessed: 2019-04-13.

[13] Guoguo Chen, Carolina Parada, and Georg Heigold. Small-footprint keyword spotting using deep neural networks. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 4087–4091, July 2014.

[14] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *Advances in Neural Information Processing Systems 31*, pages 842–852, 2018.

[15] Tara N Sainath, Carolina Parada. Convolutional Neural Networks for Small-Footprint Keyword Spotting. *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[16] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22:1533–1545, October 2014.

[17] David Verstraeten, Benjamin Schrauwen and Dirk Stroobandt. Reservoir-based techniques for speech recognition. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1050–1053, July 2006.

[18] Mark D. Skowronski and John G. Harris. Noise-Robust Automatic Speech Recognition Using a Predictive Echo State Network. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1724–1730, July 2007.

[19] Herbert Jaeger. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the ”echo state network” approach. Technical Report GMD-Report 159, German National Research Center for Information Technology, October 2002.

[20] Herbert Jaeger. The ”echo state” approach to analysing and training recurrent neural networks-with an erratum rate. Technical Report GMD-Report 148, German National Research Center for Information Technology, 2001.

[21] Tensorflow speech recognition challenge. https://www.kaggle.com/c/tensorflow-speech-recognition-challenge. Accessed: 2019-03-02.

[22] Pete Warden. Speech Commands: A public dataset for single-word speech recognition. http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz, 2017.

[23] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and Applications of Echo State Networks with Leaky Integrator Neurons. *Neural Networks*, 20(3):335–352, April 2007.

[24] Speech representation and data exploration kernel description. https://www.kaggle.com/davids1992/speech-representation-and-data-exploration. Accessed: 2019-04-13.

[25] Marina Sokolova and Guy Lapalmeb. A systematic analysis of performance measures for classification tasks. 45(4):427–437, July 2009.