Build a Speech-to-text Converter

Artificial Intelligence Techniques PG (6685)

Final Report

Jiajia Li

U3215584

Abstract

Automatic speech to text translators has been broadly used in many environments such as mobile applications, intelligent virtual assistants, and self-driving cars. To study the core theory of this technology, we build and test a CNN model on the Speech Commands Dataset. The dataset contains more than 100k audio files of 35 words stored in one-second WAVE format files. In this report, we will introduce and explain some basic knowledge of audio processing in machine learning applications as well as discuss the differences between the Speech Commands Dataset and conventional datasets used for automatic speech recognition of full sentences. We will train and test a deep learning model on the dataset chosen, and we will build a small self-made audio dataset for testing the model on brand new audios.

1 Introduction

1.1 History of Speech Recognition

Language is the primary way for communications among human beings. To create intelligent human computer interactive (HCI) systems, it is essential for machines to recognise human speech. Early back to the 1960s, computer scientists had started to research on this problem. However, speech recognition is a difficult and challenging task, and the first system that can really understand speech was created until the 1980s [1].

The first attempt to build a speech recognition system was happened in the Bell Laboratories in 1952, the Audrey system was designed to recognise digits spoken aloud by a single voice. Later in ten years, IBM introduced a system that can understand and respond to 16 words in English. In the 1970s, speech recognition had developed rapidly because of The Speech Understanding Research (SUR) program hosted by the US Department of Defence and DARPA. Carnegie Mellon's Harpy system was capable of understanding over 1,000 words, and in the meantime, the Bell Laboratories introduced a system that could interpret multiple voices [2].

A breakthrough came in the 1980s, the Hidden Markov Model (HMM) was invented, and it can model a wide range of time series data. The application of HMM on speech recognition received huge success on some tasks like speech classification. Going to the 1990s, speech recognition was developed hugely because of the invention of personal computer and faster processors, speech recognition systems were widely used by people among many fields in the first time. In 2001, Google launched its first voice search system, and was built as an application. Hence, the speech recognition system can be used by millions of people. In 2010s, many intelligent voice assistants were developed, and were used by people during their daily life.

1.2 Neural Networks for Speech Recognition

In the late 1980s and early 1990s, there was a popular trend to implement neural networks on speech recognition tasks. However, it could not outperform the combination of HMMs with acoustic models based on Gaussian mixtures, then the research becomes silent in this field.

In the beginning of 2010s, it sees a revival of using neural networks for speech recognition. There are three main factors that contribute to this phenomenon: 1) Deep Neural Networks (DNNs): the neural network can be built with many hidden layers to make the architecture of the network deep enough to solve complicated problems. 2) much faster hardware was invented and the DNNs can be trained quickly and effectively. 3) using a large number of output units hugely improves the performance of neural networks [4]. The success of recent applications of neural networks on speech recognition is

due to the availability of more computing power, larger training dataset, and better software engineering.

1.3 Convolutional Neural Networks for Speech Recognition

Convolutional Neural Networks have been widely used in computer vision tasks, and it has been successful. It had been used in solving speech recognition problems but then got abandoned because of dissatisfied performance, that maybe because they structured the audio information across time series rather than frequencies. More recent works have demonstrated that convolution across frequency was very effective, and multiple convolutional layers with a large number of convolution kernels in each layer can improve the performance of the neural network.

2 Literature Review

2.1 Long Short-term Memory RNN

Many studies have shown that Recurrent Neural Network is powerful for sequential data modelling. The Long Short-term Memory RNN architecture hold the state-of-the-art results of cursive handwritten digit recognition. However, the performance of RNN on speech recognition tasks were so far disappointing at that time with better result got by deep-forward networks. In 2013, Alex Graves et al. from University of Toronto wanted to investigate the application of the Long Short-term Memory RNNs on speech recognition since it has very good performance for digits recognition [5]. Their results achieved a 17.7% test error rate on the TIMIT phoneme recognition task.

2.2 CNN-HMM

In 2014, a group of researchers from York University, University of Toronto, and Microsoft conducted research on combine the CNN with HMM together to perform speech recognition tasks [6]. Their idea came from a previous study on combining deep neural networks (DNN) with HMM for automatic speech recognition, and it has significantly improved the performance over the conventional Gaussian Mixture Model (GMM)-HMM. They proposed a limited-weight-sharing-scheme that can better model the speech patterns. CNNs show some resistance to small variations of features, thus, it is good to use on speech recognition as speech features will often influenced by environment noises, individual speakers, and regional dialects. Their CNN-HMM model reduced 6%-10% of error rate for the TIMIT phone recognition and the voice search large vocabulary speech recognition tasks.

3 Methodology

3.1 Google Colab

Google Colab is a website that allows you write and run python codes in a web browser, it is similar to Jupyter Notebooks, you can write codes in code cells as well as write rich format texts in markdown cells. Google Colab require no configurations and you can use it by just create a new Colab notebook, and it provides GPUs for people free to use. For each Colab notebook, you will also have around 70GB storage space for you to download files from internet into working environments, and also for uploading local files to the working environment.

For this project, I need to use a comparatively big dataset also the GPU is required so I choose to use Google Colab. It is also faster when downloading the dataset from the internet because you can use the bandwidth that provided by google.

3.2 CNN

Although various kinds of deep neural networks can perform speech recognition tasks, the CNN is the most widely used deep learning technique and its develop environment is very mature. Many tools have been built for computer science engineers and researchers to use the technique more conveniently. For example, the deep learning frameworks TensorFlow, PyTorch, and Caffe. It is good to use a mature system to develop the recognition model.

3.3 TensorFlow

TensorFlow is an open-source library for people to build machine learning models faster and easier. It uses Python as the API for people to write codes while executing them in high-performance C++. There are also other open source frameworks like PyTorch and Caffe, but for this project TensorFlow is the CNN framework we use.

3.4 Spectrogram

Before we move on to the actual implementation step, we need to have more knowledge of the audio data. The audio data that will be input into the CNN are spectrograms. The x axis of a spectrogram are time series, and the y axis of a spectrogram are strengths of frequencies. What it different from images is that when you move the pixels up and down, it may influence the actual meaning of the spectrogram.

4 Materials

4.1 Speech Commands Dataset

This dataset was published in 2018 by Warden [7]. This dataset was built to help develop and train keyword spotting systems. The Speech Commands Dataset consists of 105,829 utterances of 35 words, for each utterance it stored in a one-second WAVE format audio file. The total size of the dataset is 3.8 GB.

4.2 Mini Speech Commands Dataset

This dataset is a portion of the Speech Commands Dataset. It consists of 8,000 utterances of 8 words, and 1,000 examples for each word. The words contained in this dataset are “no”, “yes”, “left”, “right”, “up”, “stop”, “go”, and “down”. We can use this dataset as the sample dataset to train and test the CNN model with a shorter period. After build and test the model, we then can start train on the whole dataset.

5 Results

5.1 Network Architecture

```
Input shape: (124, 129, 1)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resizing (Resizing)          (None, 32, 32, 1)         0
_________________________________________________________________
normalization (Normalization (None, 32, 32, 1)         3
_________________________________________________________________
conv2d (Conv2D)              (None, 30, 30, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 12544)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1605760
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 35)                4515
=================================================================
Total params: 1,629,094
Trainable params: 1,629,091
Non-trainable params: 3
_________________________________________________________________
```

Figure 1. Network Architecture

From figure 1 we can see this network has total ten layers, with two convolutional layers. The input shape of the neural network is a three-dimensional matrix which is 124x129x1. The output layer of the network has 35 neurons which means the output has 35 different classes. The total parameters of this network are 1,629,094.

5.2 Training Process

```
Epoch 1/10
100/100 [==============================] - 39s 381ms/step - loss: 1.7093 - accuracy: 0.3839 - val_loss: 1.2853 - val_accuracy: 0.5987
Epoch 2/10
100/100 [==============================] - 21s 210ms/step - loss: 1.1378 - accuracy: 0.6019 - val_loss: 0.9575 - val_accuracy: 0.6988
Epoch 3/10
100/100 [==============================] - 21s 211ms/step - loss: 0.8740 - accuracy: 0.6848 - val_loss: 0.8152 - val_accuracy: 0.7400
Epoch 4/10
100/100 [==============================] - 21s 208ms/step - loss: 0.7197 - accuracy: 0.7447 - val_loss: 0.7683 - val_accuracy: 0.7525
Epoch 5/10
100/100 [==============================] - 21s 209ms/step - loss: 0.6268 - accuracy: 0.7808 - val_loss: 0.6691 - val_accuracy: 0.7887
Epoch 6/10
100/100 [==============================] - 21s 211ms/step - loss: 0.5448 - accuracy: 0.8072 - val_loss: 0.6751 - val_accuracy: 0.7875
Epoch 7/10
100/100 [==============================] - 21s 210ms/step - loss: 0.5012 - accuracy: 0.8270 - val_loss: 0.6441 - val_accuracy: 0.7875
Epoch 8/10
100/100 [==============================] - 21s 208ms/step - loss: 0.4489 - accuracy: 0.8394 - val_loss: 0.6123 - val_accuracy: 0.8225
Epoch 9/10
100/100 [==============================] - 21s 209ms/step - loss: 0.4087 - accuracy: 0.8516 - val_loss: 0.6057 - val_accuracy: 0.8238
Epoch 10/10
100/100 [==============================] - 21s 210ms/step - loss: 0.3754 - accuracy: 0.8652 - val_loss: 0.6049 - val_accuracy: 0.8425
```

Figure 2. Training Detail of Mini Speech-Commands Dataset

```
Epoch 1/10
1323/1323 [==============================] - 545s 411ms/step - loss: 2.0051 - accuracy: 0.4312 - val_loss: 1.0785 - val_accuracy: 0.7011
Epoch 2/10
1323/1323 [==============================] - 301s 227ms/step - loss: 1.2481 - accuracy: 0.6310 - val_loss: 0.8287 - val_accuracy: 0.7643
Epoch 3/10
1323/1323 [==============================] - 303s 229ms/step - loss: 1.0482 - accuracy: 0.6873 - val_loss: 0.7359 - val_accuracy: 0.7901
Epoch 4/10
1323/1323 [==============================] - 302s 229ms/step - loss: 0.9290 - accuracy: 0.7197 - val_loss: 0.6575 - val_accuracy: 0.8122
Epoch 5/10
1323/1323 [==============================] - 300s 227ms/step - loss: 0.8554 - accuracy: 0.7427 - val_loss: 0.6428 - val_accuracy: 0.8141
Epoch 6/10
1323/1323 [==============================] - 298s 225ms/step - loss: 0.7994 - accuracy: 0.7565 - val_loss: 0.6065 - val_accuracy: 0.8245
Epoch 7/10
1323/1323 [==============================] - 305s 230ms/step - loss: 0.7473 - accuracy: 0.7706 - val_loss: 0.5766 - val_accuracy: 0.8372
Epoch 8/10
1323/1323 [==============================] - 313s 237ms/step - loss: 0.7051 - accuracy: 0.7835 - val_loss: 0.5698 - val_accuracy: 0.8374
Epoch 9/10
1323/1323 [==============================] - 302s 228ms/step - loss: 0.6746 - accuracy: 0.7913 - val_loss: 0.5628 - val_accuracy: 0.8392
Epoch 10/10
1323/1323 [==============================] - 304s 230ms/step - loss: 0.6434 - accuracy: 0.8005 - val_loss: 0.5572 - val_accuracy: 0.8446
```

Figure 3. Training Detail of Speech-Commands Dataset

From the above two pictures, we can see for mini speech-commands dataset, it took 39 seconds for the first epoch and 21 seconds for the remaining epochs, totally it took around 4 minutes to run. For

training the speech-commands dataset, it took 545 seconds to run for the first epoch and about 300 seconds for the remaining epochs, totally it took around 55 minutes to run, it is much longer than training the mini speech-commands dataset.

5.3 Test Accuracy

The test dataset is separated from the whole dataset, and it is not participating in the training process, the training process contains using the training dataset to train the neural network and the validation dataset to validate the training result for each epoch. The test dataset is a brand-new dataset that has never been seen by the neural network before, so the testing process can best estimate the performance of the trained network.

```
y_pred = np.argmax(model.predict(test_audio), axis=1)
y_true = test_labels

test_acc = sum(y_pred == y_true) / len(y_true)
print(f'Test set accuracy: {test_acc:.0%}')
```

```
Test set accuracy: 83%
```

Figure 4. Test Accuracy for Mini Speech-Commands Dataset

The test accuracy for mini speech-commands dataset is 83% which is good but not good enough.

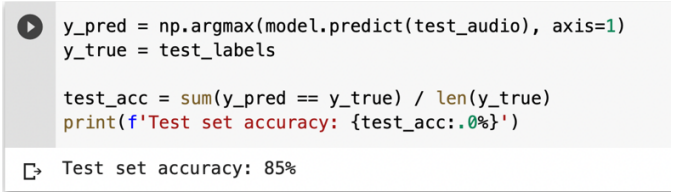

Figure 5. Test Accuracy for Speech-Commands Dataset

The test accuracy for speech-commands dataset is 85% which is a little better than the mini speech-commands dataset even though they have the same neural network architecture and training epochs. The reason why it is higher is because the speech-commands dataset has more samples for individual words, it of course will have a higher accuracy.

5.4 Confusion Matrix

Using confusion matrix to see the prediction results for the neural network is a clearer way. The x axis and y axis of the confusion matrix are the same, they are the output classes of the neural network. You can see for each class, how many samples got predicted correct or incorrect from the trained network, and the colours can show the volume of the samples.

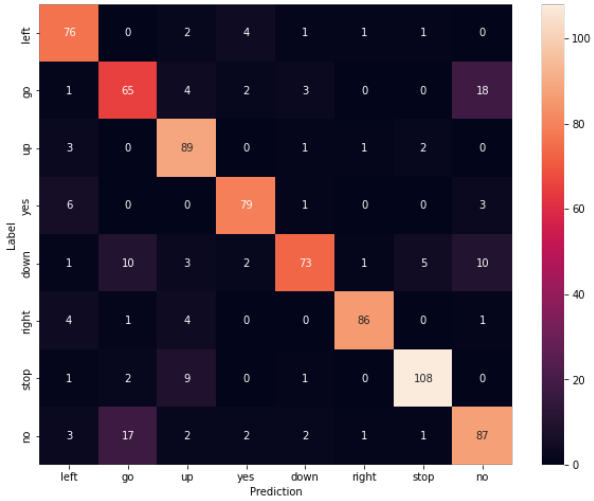

Figure 6. Confusion Matrix of the Prediction Results of Mini Speech-Commands Dataset

From figure 6 we can see the model has good prediction results, but it is confused by the word “go” and “no”. It predicts 17 “no” to the wrong label “go” and 18 “go” to the wrong label “no”.

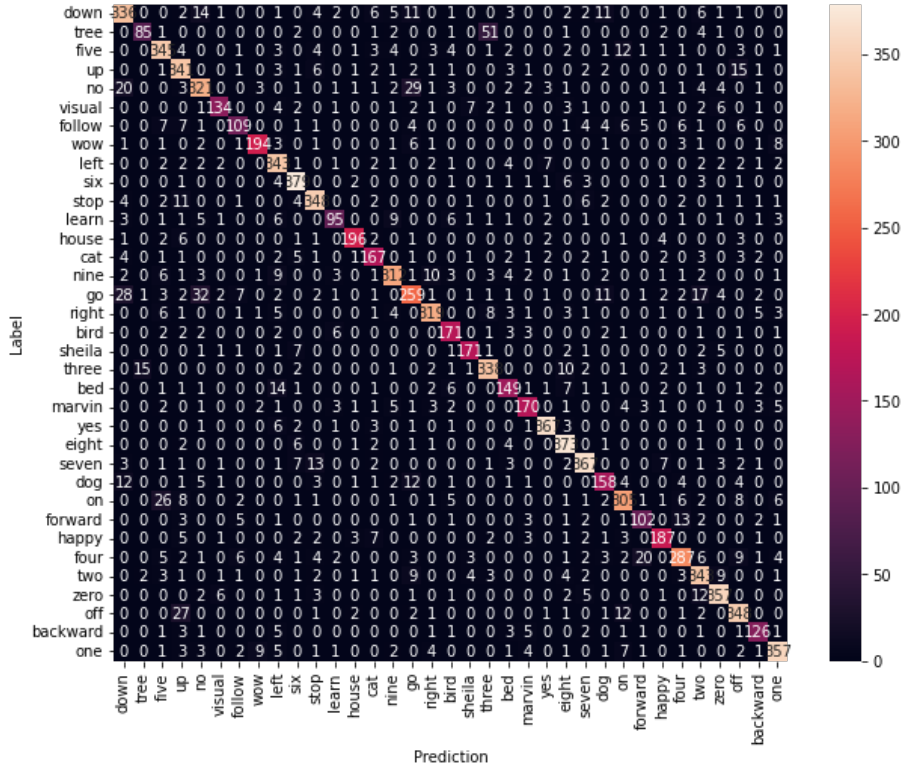

Figure 7. Confusion Matrix of the Prediction Results of Speech-Commands Dataset

This confusion matrix is bigger and contains more information. The matrix contains 35 x axis labels and 35 y axis labels. And because it contains many information, we cannot see clearly of the values in each cell, but we can see it from the colour. The whiter the figure is larger, and the blacker means the figure is smaller. Except for the word “go” and “no”, the network also got confused a little by another two words which are “tree” and “three”. The network predicts 51 “tree” to a wrong label “three”.

But except this, the network performs very good on the testing dataset.

6 Discussion

Keyword spotting is a very useful function in many scenarios, such as virtual voice assistants in many working environments like automatic driving systems. To train this dataset it can help deploy such keyword spotting functions. From our exploration, we trained a CNN that can perform this task very well, fast and with high accuracy. Although in my view, I believe a testing accuracy of 90 percent or higher would be better for daily employment. With future development, we expected to modify the architecture of our network, try different convolutional and pooling layers, activation methods, and even with more training epochs even though it costs much more time.

7 References

[1] Gaikwad, S. K., Gawali, B. W., & Yannawar, P. (2010). A review on speech recognition technique. *International Journal of Computer Applications*, 10(3), 16-24.

[2] A short history of speech recognition. (2021). *Sonix.ai/history-of-speech-recognition.*

[3] Al Smadi, T., Al Issa, H. A., Trad, E., & Al Smadi, K. A. (2015). Artificial intelligence for speech recognition based on neural networks. *Journal of Signal and Information Processing*, 6(02), 66.

[4] Deng, L., Hinton, G., & Kingsbury, B. (2013, May). New types of deep neural network learning for speech recognition and related applications: An overview*. In 2013 IEEE international conference on acoustics, speech and signal processing* (pp. 8599-8603). IEEE.

[5] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. *In 2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645-6649). Ieee.

[6] Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10), 1533-1545.

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