A Research about the speech recognition using CNN(Convolutional Neural Network) and LSTM(Long Short Term Memory)

Guiming Xu
Georgetown University
gx26@georgetown.edu

Yuxuan Yao
Georgetown University
yy560@georgetown.edu

Yuan Liu
Georgetown University
yl1130@georgetown.edu

Kuiyu Zhu
Georgetown University
kz175@georgetown.edu

Abstract—Nowadays, Deep learning models are receiving more and more attention, and being used more and more in many fields. In this paper, we focus on using deep learning models to classify and make predictions on a large number of speech files. We will also compare the accuracy rates of CNN, LSTM, and other models.

I. INTRODUCTION

Speech recognition is a voice classification problem. In this project, we build different convolutional neural network models on audio files from the Tensorflow Speech Recognition Challenge dataset in Kaggle, and get predictions on the data set.

In the field of speech recognition, deep learning is a hot research area these days. We construct two kinds of CNN models to predict the label corresponding to speech files, and finally compare these models. The way we used to evaluate is accuracy rate, and the reason behind is that audio files in the dataset are represented for different labels like “Yes”, “No”, and so on, so accuracy rate is a good metric to compare models.

From the history of speech recognition, the earliest advances in speech recognition focused on the creation of vowels as the basis for a system that was intended to build a talking machine.[2] The real breakthrough in the field of speech recognition was the Google Voice Search App, launched in 2008. [3]

Meanwhile, Gartner study group predicts that by 2020, 30% of browsing sessions will include voice search.

II. RELATED WORKS

With many possible deep learning models to predict audio files, we research articles related to our topic on the internet, and these researches help us get better acknowledgement to understand.

A. CNN (Convolutional Neural Network)

It is called Convolutional Neural Network because it is a mathematical or computational model that

mimics the structure and function of a biological neural network (an animal's central nervous system, especially the brain). At present, CNN is very widely used in the field of speech recognition, and the reasons behind it are mainly attributed to the following.

1. To improve the speech recognition rate, it is necessary to overcome the diversity of speech signals, including the diversity of speakers, the diversity of environments, etc. Because a convolutional neural network provides a flattening invariant convolution in time and space, it is necessary to overcome the diversity of speech signals. Since a convolutional neural network provides translation-invariant convolution in time and space, the invariance of the convolution can be used to overcome the diversity of the speech itself.
2. From a practical point of view, CNNs are also easy to implement for massively parallel computing.

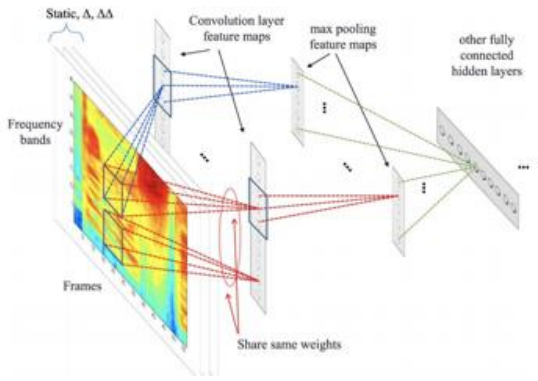

Figure 1: An illustration of the regular CNN[4]

B. LSTM (Long short-term memory)

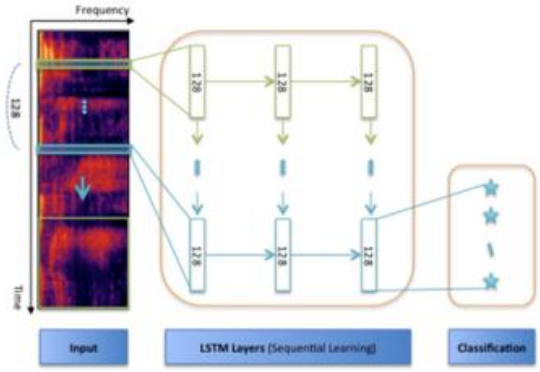

Figure 2: The sample LSTM structure for speech emotion recognition[5]

LSTM is a special type of RNN that is designed to solve the gradient loss and gradient explosion problems during training of long sequences. Simply put, LSTM performs better over longer sequences than regular RNNs.

C. Model Estimation

In paper [6], the authors proposed the naive bayes model, Markov random field, and the supervised learning and unsupervised learning models to test speech recognition, and the results show that the machine learning models perform well on speech recognition except for the excessive computation volume and computation time. Therefore, this research paper raises us curiosity to know whether deep learning models will present better in speech recognition.

III. Dataset

The dataset we used is from [7]. In this dataset, for the train set, there are 64727 audio files with 31

different labels like ‘yes’, ‘no’, and for the test set, there are 150000 audio files without labels.

Core Words: label with more than 5 audio files

No.	Label Name	No.	Label Name
1	Yes	2	No
3	Up	4	Down
5	Left	6	Right
7	On	8	Off
9	Stop	10	Go
11	Zero	12	One
13	Two	14	Three
15	Four	16	Five
17	Six	18	Seven
19	Eight	20	Nine

Auxiliary Word: Label with only one audio files

No.	Label Name	No.	Label Name
1	Bed	2	Bird
3	Cat	4	Dog
5	Happy	6	House
7	Sheila	8	Tree
9	Marvin	10	Wow

IV. Data Preprocessing

The main required packages we used for audio files processing are SoundFile and Librosa. SoundFile is an audio library based on libsndfile, CFFI and NumPy. Librosa is a package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. Generally, they help us to convert audio files into arrays.

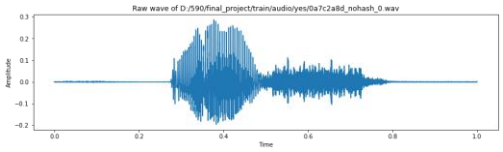

Figure 3: The wave plot of “yes” audio file

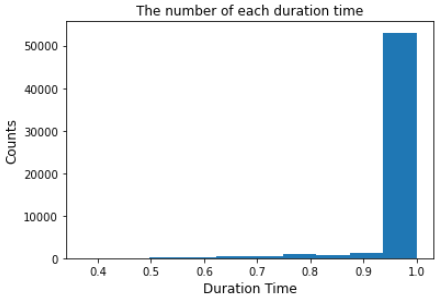

Figure 4: The number of each duration time

In dataset, most of the audio files in the dataset are 1 second long, as shown in Figure 3, which has a time duration of 1 second. However, there are some data that are not 1 second long. As shown in Figure 4, some data under the label is longer than 1 second and some is shorter than 1 second, so we first remove such outliers. Adding this will normalize the rest of the audio files to 1 second, which will be useful for modeling purposes.

In addition, Considering the computational effort of subsequent modeling, we downsampled our audio. The sample rate of our audio file is 16000. With Librosa, we downsample the file directly to 8000.

V. Models

CNN Model

```
K.clear_session()
inputs = Input(shape=(8000,1))

# The input are 1D arrays because we audio files into arrays, and we use Conv1D and MaxPoo
#First Conv1D layer
x = Conv1D(8,13, padding='valid', activation='relu', strides=1)(inputs)
x = MaxPooling1D(3)(x)
x = Dropout(0.3)(x)

#Second Conv1D layer
x = Conv1D(16, 11, padding='valid', activation='relu', strides=1)(x)
x = MaxPooling1D(3)(x)
x = Dropout(0.3)(x)

#Third Conv1D layer
x = Conv1D(32, 9, padding='valid', activation='relu', strides=1)(x)
x = MaxPooling1D(3)(x)
x = Dropout(0.3)(x)

#Fourth Conv1D layer
x = Conv1D(64, 7, padding='valid', activation='relu', strides=1)(x)
x = MaxPooling1D(3)(x)
x = Dropout(0.3)(x)

# Convert 1D arrays to a single long continuous linear vector
#Flatten layer
x = Flatten()(x)

# Connect to entire neural network
#Dense Layer 1
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)

#Dense Layer 2
x = Dense(128, activation='relu')(x)
x = Dropout(0.3)(x)

outputs = Dense(len(labels), activation='softmax')(x)

CNN_model = Model(inputs, outputs)
CNN_model.summary()
```

Figure 5. CNN model building

Since we convert the audio files into 1D arrays, we use Conv1D and MaxPooling1D in Convolutional layers. As we can see in Figure 5, our CNN model building codes, we have total 4 convolutional layers and each convolutional layer has a dropout layer, which is regularization randomly set some of the dimensions of input vector to be zero. What's more, the dropout layer has the effect of averaging, reducing the complex co-adaptive relationships between neurons, reducing time and avoiding overfitting problems. In Conv1D layers, the activation functions are relu. Then we use flattening layer to convert 1D arrays into a single long continuous linear vector. In the last, we have 2 dense layers, which connect to the entire neural network.

After modeling, then we train the CNN model. The loss function is categorical crossentropy. In the training model, we have reached 100 epochs, and each training epoch takes about 98 seconds. The whole model training took a total of 2 hours and 50 minutes. As a result, this is what we get after training the model:

```
322/322 [==============================] - 5s 15ms/step
- loss: 0.5705 - accuracy: 0.8266
```

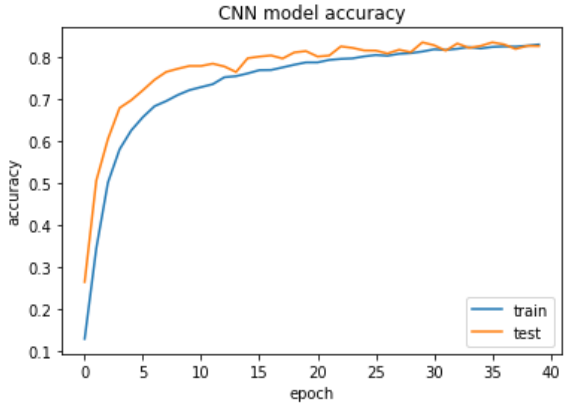

Figure 6. The accuracy plot of CNN model

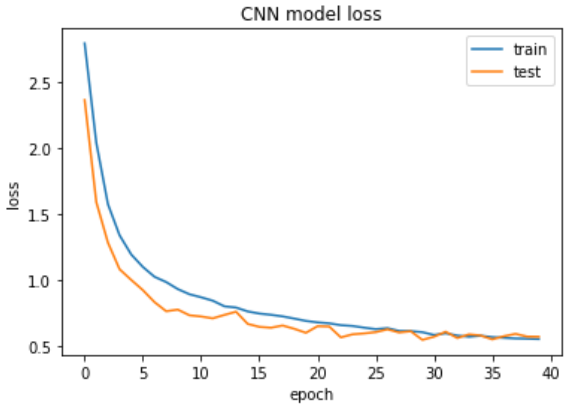

Figure 7. The loss plot of CNN model

As we can see in Figure 6 and 7, the train loss and test loss are both decreasing and the accuracy is increasing, which means the model is still learning and it is the best model. Generally, out model is trained well.

With the models we trained, we try some simple basic speech recognition. As mentioned before, speech recognition is based on speech classification.

We randomly selected 10 audio files from the test set and performed the recognition. The results are as follows:

```
No. 1 :
Original Audio: no
Recgnition Text: no
Recognition Success
--------------
No. 2 :
Original Audio: no
Recgnition Text: no
Recognition Success
--------------
No. 3 :
Original Audio: two
Recgnition Text: two
Recognition Success
--------------
No. 4 :
Original Audio: go
Recgnition Text: go
Recognition Success
--------------
No. 5 :
Original Audio: bird
Recgnition Text: three
Recognition Failed
--------------
--------------
No. 6 :
Original Audio: six
Recgnition Text: six
Recognition Success
--------------
No. 7 :
Original Audio: left
Recgnition Text: left
Recognition Success
--------------
No. 8 :
Original Audio: nine
Recgnition Text: nine
Recognition Success
--------------
No. 9 :
Original Audio: no
Recgnition Text: go
Recognition Failed
--------------
No. 10 :
Original Audio: eight
Recgnition Text: eight
Recognition Success
```

Figure 8. CNN model speech recognition results

As Figure 8 showed, most of the recognitions are correct but No.5 and No.9 are failed. In general, the accuracy of this recognition test is 80%, which is close to 82%, which is the model accuracy.

1D CNN Model 2

As it was mentioned above, we process and convert the audio data into 1D arrays. There may be many possible architectures to build 1d cnn models. The main idea of the following model is using 1d convolutional neural network with 10 pooling layers and kernel size 9. In this way, the model is more complex than the previous one, and its performance will be better as expected.

```
from keras.layers import BatchNormalization,Activation, GlobalMaxPool1D
x_input = Input(shape = (8000,1))
x = BatchNormalization(name = 'batchnormal_1d_in')(x_input)
for i in range(9):
    name = 'step'+str(i)
    x = Conv1D(8*(2 ** i), (3),padding = 'same', name = 'conv'+name+'_1')(x)
    x = BatchNormalization(name = 'batch'+name+'_1')(x)
    x = Activation('relu')(x)
    x = Conv1D(8*(2 ** i), (3),padding = 'same', name = 'conv'+name+'_2')(x)
    x = BatchNormalization(name = 'batch'+name+'_2')(x)
    x = Activation('relu')(x)
    x = MaxPooling1D((2), padding='same')(x)
x = Conv1D(1024, (1),name='last1024')(x)
x = GlobalMaxPool1D()(x)
x = Dense(1024, activation = 'relu', name= 'dense1024_onlygmax')(x)
x = BatchNormalization(1)(x)
x = Dense(len(labels), activation = 'softmax',name='cls_1d')(x)
model1 = Model(x_input, x)
model1.summary()
```

Figure 9. 1D CNN Model 2

On the other hand, this model will be more computationally expensive. We are sorry to say that, due to the limitation of the hardware and machine we are using, the model did not successfully fit the data. In another word, we had spent a lot of time training the model and had to stop it before finishing. We decide to use 1 as the number of epoch(s), and the result is shown below.

```
1286/1286 [==============================] -
3386s 3s/step - loss: 1.5068 - accuracy: 0.54
47 - val_loss: 1.5781 - val_accuracy: 0.5820
```

Figure 10. 1D CNN Model 2 Result

LSTM Model

The last model we select is LSTM. We first use Conv1D as the first layer for input and then use LSTM. At the beginning of the LSTM model, we set too large parameters, which causes a ridiculous running time. Even though we finally adjust the parameters down to 32, the model still needs about 285 hours to run.

```
model = Sequential()
inputs = Input(shape=(8000,1))

x = Conv1D(8,13, padding='same', activation='relu', strides=1)(inputs)
x = MaxPooling1D(strides =1)(x)
x = BatchNormalization()(x)

x = Conv1D(16,11, padding='same', activation='relu', strides=1)(x)
x = BatchNormalization()(x)

x = Conv1D(32,9, padding='same', activation='relu', strides=1)(x)
x = MaxPooling1D(strides =1)(x)
x = BatchNormalization()(x)

# x = Conv1D(64,7, padding='same', activation='relu', strides=1)(x)
# x = BatchNormalization()(x)

#x = Reshape(12, 10)(x)

x = LSTM(32, return_sequences = True)(x)
x = LSTM(32, return_sequences = True)(x)

x = Dense(128)(x)
x = TimeDistributed(Dense(64))(x)
x = Activation('relu')(x)
x = Flatten()(x)
x = Dense(len(labels))(x)
outputs = Activation('softmax')(x)

adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model  = Model(inputs, outputs)
model.summary()
```

Figure 11. LSTM Model

We have the same situation as we mentioned in the

ID CNN Model part. Therefore, we decide to use 1 as the number of epoch, and the result is shown below:

```
322/322 [==============================] - 298s
925ms/step - loss: 1.9989 - accuracy: 0.4261
The test accuracy: 0.42608442902565
```

VI. Results

Model	Accuracy
CNN	**0.8266**
1D CNN Model2	**0.5820+**
LSTM	**0.4261**

VII. Conclusions

The paper focuses on a deep learning algorithm (CNN) to classify and predict the voice. At present, CNN is very widely used in the field of speech recognition since it overcomes the diversity of speech signals. The first CNN model was used Conv1D and MaxPooling1D in Convolutional.The reactive function is relu. Then we use a flattening layer to convert 1D arrays into a single long continuous linear vector. And the accuracy is about 0.8266. For the second one, a more complex CNN model was used. The main idea of this one is using a 1d convolutional neural network with 10 pooling layers and kernel size 9. And for the LSTM model, due to hardware limitations and time constraints, the results we obtained were not very satisfactory. The accuracy of the test is only 39.12%.

For future improvement, the solution is how to figure out the limitation of hardware and machine will be obtained and more epochs should be presented to get better performance of the model.

Bibliography

[1]: Abdel-Hamid, Ossama. "Convolutional Neural Networks for Speech Recognition." 2014, p. 11.

[2]: Clark, Boyd. "Speech Recognition Technology: The Past, Present, and Future." ***TheStartUp,*** **10 1 2018, https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf.**

[3]: Li, Deng. "Machine Learning Paradigms for Speech Recognition: An Overview." ***cvsp,*** **2013, pp. 12-20.**

[4]: Lim, Wootaek. "Speech Emotion Recognition using Convolutional and Recurrent Neural Networks." ***apsipa,*** **2015, http://www.apsipa.org/proceedings_2016/HTML/paper2016/137.pdf. Accessed 09 12 2020.**

[5]: Praveen, Mishra. "Applications of AI-Based Speech Recognition in Business Today." ***readwrite,*** **19 08 2020, https://readwrite.com/2020/08/19/applications-of-ai-based-speech-recognition-in-business-today/#:~:text=A%20study%20by%20Gartner%20predicts,convert%20them%20into%20readable%20text.**

[6]: Ronnan, Collobert. "Analysis of CNN-based speech recognition system using raw speech as input." ***semanticscholar,*** **2015, https://www.semanticscholar.org/paper/Analysis-of-CNN-based-speech-recognition-system-raw-Palaz-Magimai-Doss/0f94b1b699f1320a5fadbb34d29e8e255da8942f. Accessed 09 12 2020.**

[7]: https://www.kaggle.com/c/tensorflow-speech-recognition-challenge