Gender Voice Recognition Using CNN

Jiamin Shang, Zhixin Wang

1.Abstract

This study attempts to identify the gender of one certain voice. Our project. Specifically this project based on the voice dataset from kaggle which contains more than 3000 records. The ultimate goal is when collected voice go through our trained model, the result will show “male” or “female”.
Convolutional neural network is used on this prediction and it has an adequate feedback. Features that have different density distribution on different genders would be considered as model input. After the preprocessed data is used to generate training model, our test data indicated that our model has a good performance, the accuracy is over 95%.

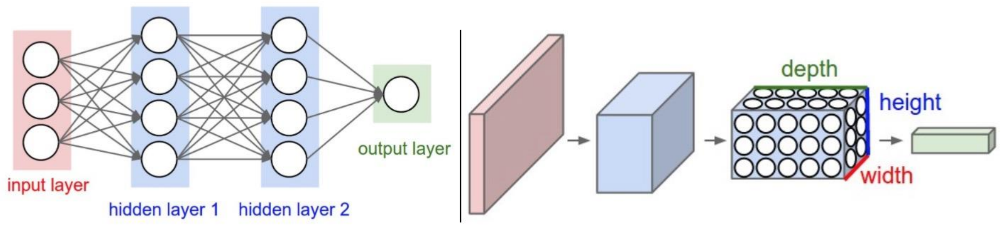
2.Introduction

2.1 Motivation & Background

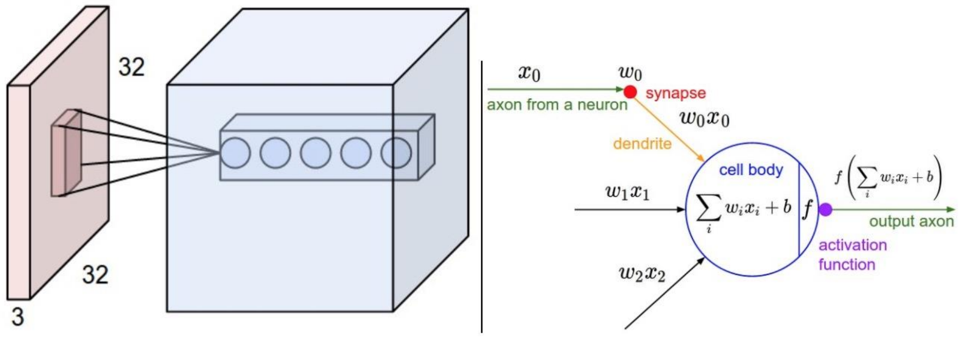
Speech recognition is everywhere, the most representative one is smartphone. Siri can understand what you said, it can tell whether you are the owner of this cell phone, it can set a alarm clock or call your friends when you are busy preparing food in kitchen. So we want to know the performance of speech recognition, how accurate it can be, how sensitive it could be to a voice. In addition, our project can let us understand how machine learning techniques help build the architecture of speech recognition.

2.2 Algorithm

Convolutional neural network is very similar to the ordinary Deep Neural Network, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, and produces non-linearity output. And they still have a loss function on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.



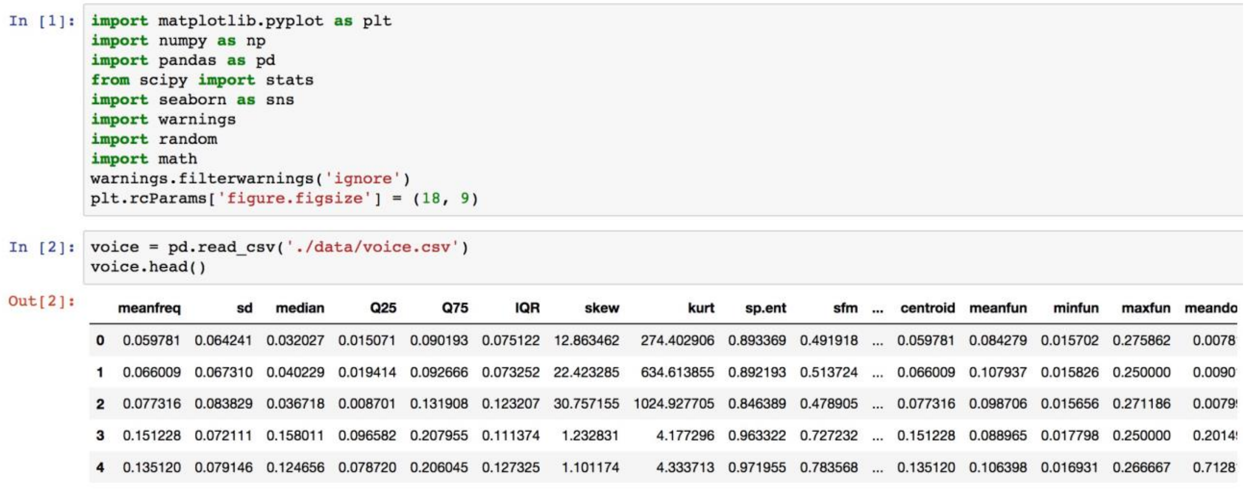
The whole network is different at ConvNet and MaxPoolNet, which is designed for input that have too many dimensions,30*30 image would have 30*30*3 dimensions input. ConvNet would use a small local receptive field to scan the input which is be able to reduce the dimensions to save time and resource to train our model. And MaxPoolNet is like to calculate the value from the feature map that we got from last layer, the method maybe the maximum value of ConvNet output or average value of it.



3. Code with document

3.1 Feature Engineering

The first step is to load data, the dataset has more than 3000 records and includes 21 potential attributes to describe a voice. But we have to figure out is every attribute is meaningful for our research goal. Density distribution helps us to make it happen.

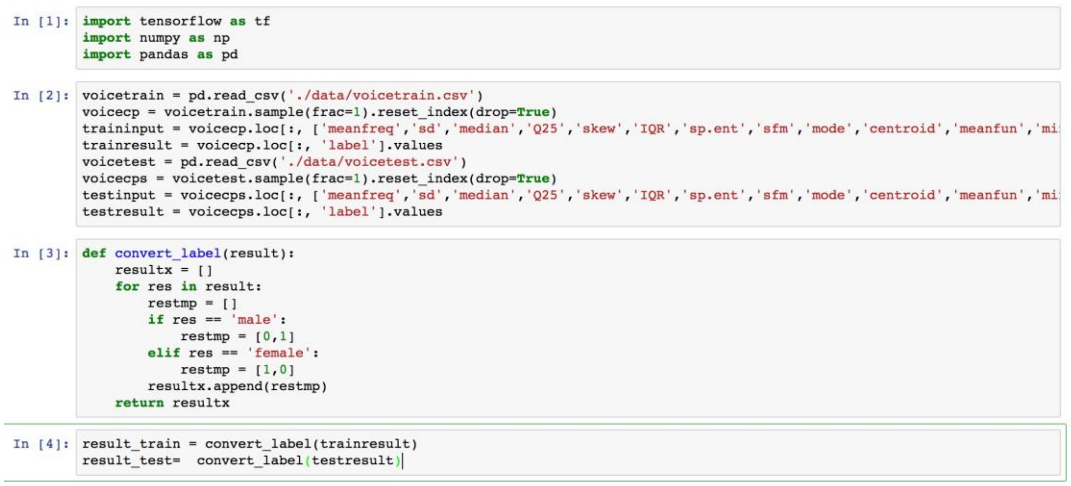


```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        from scipy import stats
        import seaborn as sns
        import warnings
        import random
        import math
        warnings.filterwarnings('ignore')
        plt.rcParams['figure.figsize'] = (18, 9)
```

```
In [2]: voice = pd.read_csv('./data/voice.csv')
        voice.head()
```

```
Out[2]:
   meanfreq        sd    median       Q25       Q75       IQR       skew         kurt    sp.ent       sfm  ...  centroid   meanfun    minfun    maxfun  meando
0  0.059781  0.064241  0.032027  0.015071  0.090193  0.075122  12.863462   274.402906  0.893369  0.491918  ...  0.059781  0.084279  0.015702  0.275862  0.0078
1  0.066009  0.067310  0.040229  0.019414  0.092666  0.073252  22.423285   634.613855  0.892193  0.513724  ...  0.066009  0.107937  0.015826  0.250000  0.0090
2  0.077316  0.083829  0.036718  0.008701  0.131908  0.123207  30.757155  1024.927705  0.846389  0.478905  ...  0.077316  0.098706  0.015656  0.271186  0.0079
3  0.151228  0.072111  0.158011  0.096582  0.207955  0.111374   1.232831     4.177296  0.963322  0.727232  ...  0.151228  0.088965  0.017798  0.250000  0.2014
4  0.135120  0.079146  0.124656  0.078720  0.206045  0.127325   1.101174     4.333713  0.971955  0.783568  ...  0.135120  0.106398  0.016931  0.266667  0.7128
```

After the analysis of density distribution, 16 of them were chosen as the input parameters. We reshaped the dataset as our input dataset, and mapped the gender label to make each record distinguishable using [1,0] and [0,1]. Accordingly, split the dataset into training set and test set.



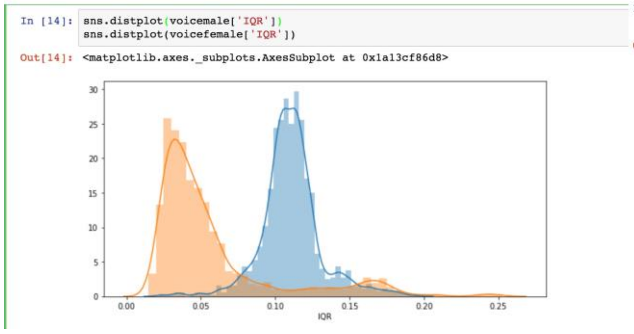
```
In [1]: import tensorflow as tf
        import numpy as np
        import pandas as pd
```

```
In [2]: voicetrain = pd.read_csv('./data/voicetrain.csv')
        voicecp = voicetrain.sample(frac=1).reset_index(drop=True)
        traininput = voicecp.loc[:, ['meanfreq','sd','median','Q25','skew','IQR','sp.ent','sfm','mode','centroid','meanfun','mi
        trainresult = voicecp.loc[:, 'label'].values
        voicetest = pd.read_csv('./data/voicetest.csv')
        voicecps = voicetest.sample(frac=1).reset_index(drop=True)
        testinput = voicecps.loc[:, ['meanfreq','sd','median','Q25','skew','IQR','sp.ent','sfm','mode','centroid','meanfun','mi
        testresult = voicecps.loc[:, 'label'].values
```

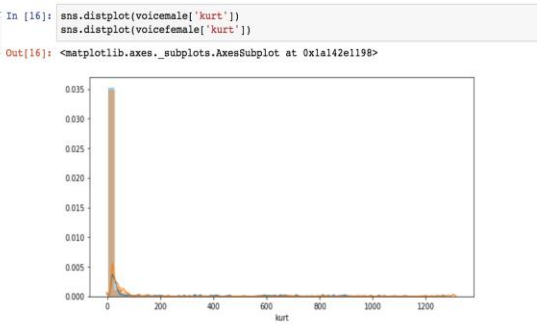
```
In [3]: def convert_label(result):
            resultx = []
            for res in result:
                restmp = []
                if res == 'male':
                    restmp = [0,1]
                elif res == 'female':
                    restmp = [1,0]
                resultx.append(restmp)
            return resultx
```

```
In [4]: result_train = convert_label(trainresult)
        result_test=  convert_label(testresult)
```

For example, from the below two plots, we could tell that IQR has distinct difference while kurt does not.



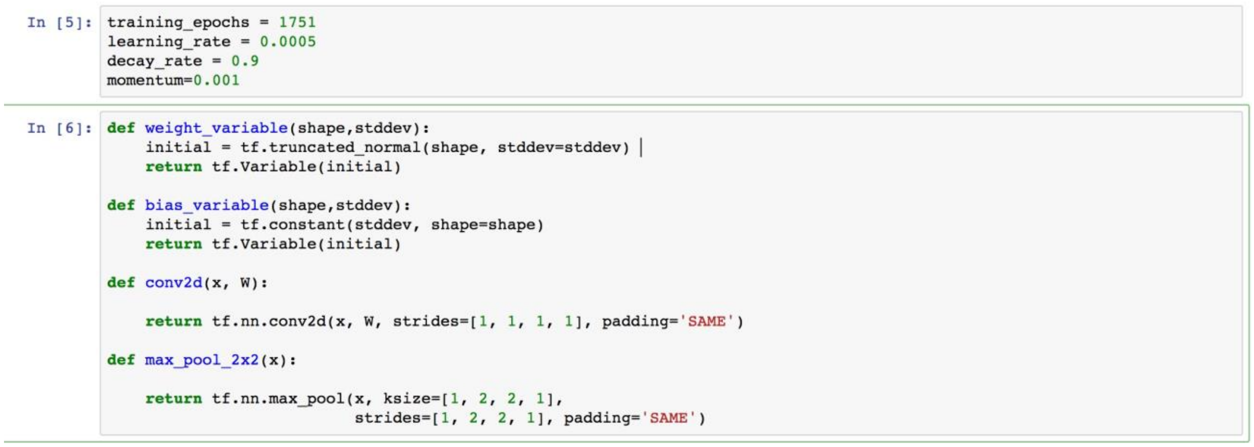
```
In [14]: sns.distplot(voicemale['IQR'])
         sns.distplot(voicefemale['IQR'])
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1a13cf86d8>
```



```
In [16]: sns.distplot(voicemale['kurt'])
         sns.distplot(voicefemale['kurt'])
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1a142e1198>
```

3.2 Build Model and Fucntion

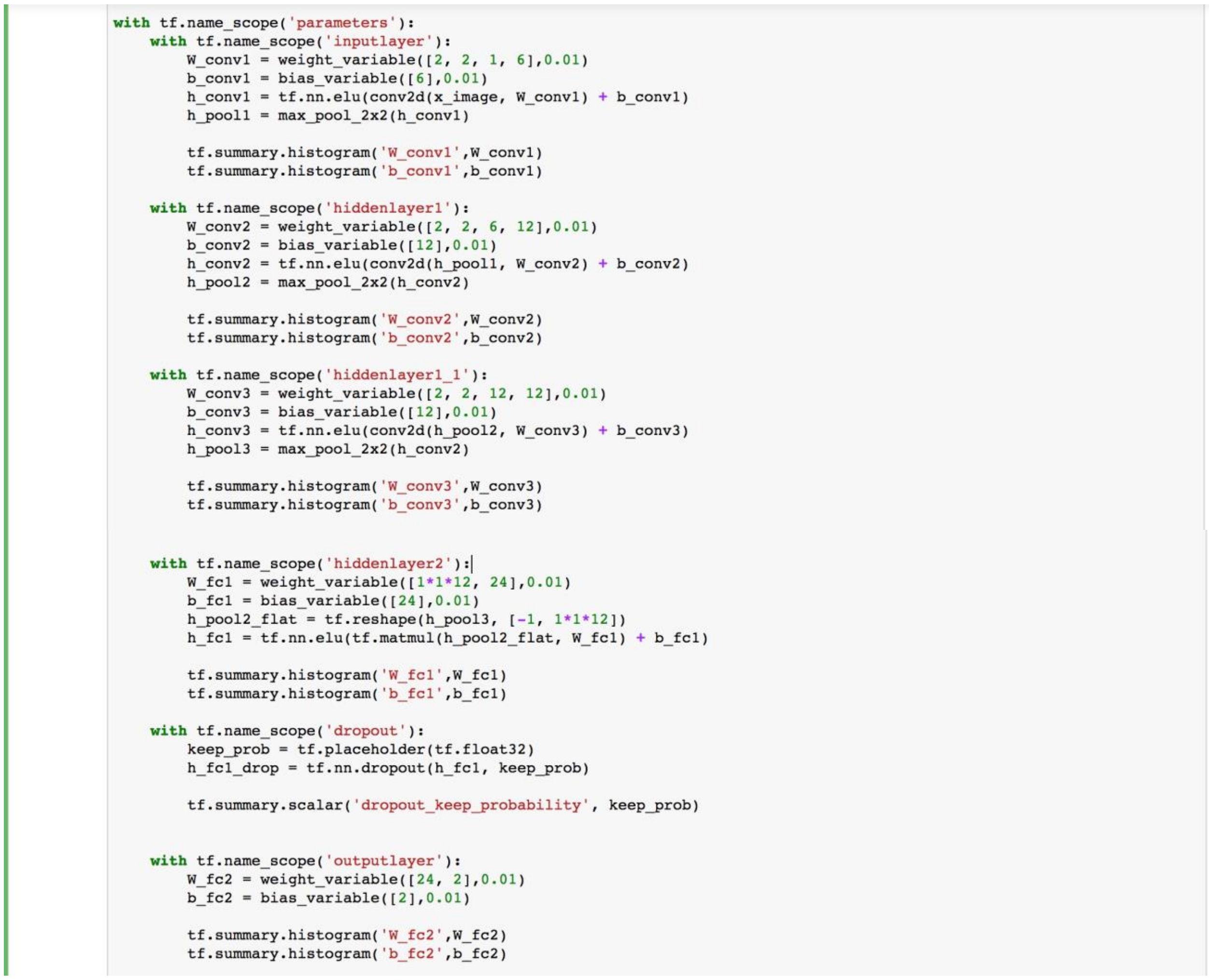
Then we defined the parameters that used for the training of neural network, and the ConvNet and MaxpoolNet function, we will use a 2*2 receptive field to scan the whole 16 dimensions input and the stride is 1, then we take the maximum value of receptive field before we send it to the activation function.



```
In [5]: training_epochs = 1751
        learning_rate = 0.0005
        decay_rate = 0.9
        momentum=0.001
```

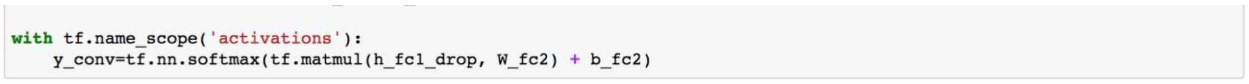
```
In [6]: def weight_variable(shape,stddev):
            initial = tf.truncated_normal(shape, stddev=stddev)
            return tf.Variable(initial)

        def bias_variable(shape,stddev):
            initial = tf.constant(stddev, shape=shape)
            return tf.Variable(initial)

        def conv2d(x, W):

            return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

        def max_pool_2x2(x):

            return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                                  strides=[1, 2, 2, 1], padding='SAME')
```

The third part of our work is to build the architecture of our neural network. We implemented three hidden layers in the network, for example, in the first hidden layer, we used 2*2 patch to calculate 6 features, 1 means the input channel and 6 means the output channel, 0.01 is the parameter for the initialization of weights. In addition, we set the dropout variable to avoid the overfitting situation.

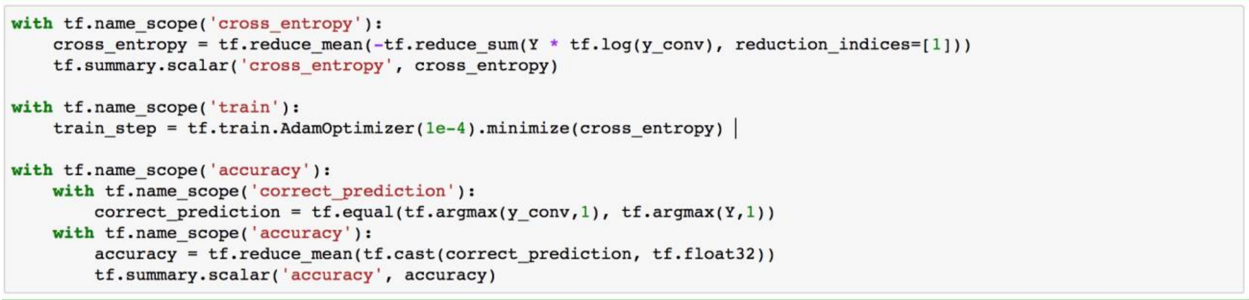


```
with tf.name_scope('parameters'):
    with tf.name_scope('inputlayer'):
        W_conv1 = weight_variable([2, 2, 1, 6],0.01)
        b_conv1 = bias_variable([6],0.01)
        h_conv1 = tf.nn.elu(conv2d(x_image, W_conv1) + b_conv1)
        h_pool1 = max_pool_2x2(h_conv1)

        tf.summary.histogram('W_conv1',W_conv1)
        tf.summary.histogram('b_conv1',b_conv1)

    with tf.name_scope('hiddenlayer1'):
        W_conv2 = weight_variable([2, 2, 6, 12],0.01)
        b_conv2 = bias_variable([12],0.01)
        h_conv2 = tf.nn.elu(conv2d(h_pool1, W_conv2) + b_conv2)
        h_pool2 = max_pool_2x2(h_conv2)

        tf.summary.histogram('W_conv2',W_conv2)
        tf.summary.histogram('b_conv2',b_conv2)

    with tf.name_scope('hiddenlayer1_1'):
        W_conv3 = weight_variable([2, 2, 12, 12],0.01)
        b_conv3 = bias_variable([12],0.01)
        h_conv3 = tf.nn.elu(conv2d(h_pool2, W_conv3) + b_conv3)
        h_pool3 = max_pool_2x2(h_conv2)

        tf.summary.histogram('W_conv3',W_conv3)
        tf.summary.histogram('b_conv3',b_conv3)


    with tf.name_scope('hiddenlayer2'):
        W_fc1 = weight_variable([1*1*12, 24],0.01)
        b_fc1 = bias_variable([24],0.01)
        h_pool2_flat = tf.reshape(h_pool3, [-1, 1*1*12])
        h_fc1 = tf.nn.elu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

        tf.summary.histogram('W_fc1',W_fc1)
        tf.summary.histogram('b_fc1',b_fc1)

    with tf.name_scope('dropout'):
        keep_prob = tf.placeholder(tf.float32)
        h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

        tf.summary.scalar('dropout_keep_probability', keep_prob)


    with tf.name_scope('outputlayer'):
        W_fc2 = weight_variable([24, 2],0.01)
        b_fc2 = bias_variable([2],0.01)

        tf.summary.histogram('W_fc2',W_fc2)
        tf.summary.histogram('b_fc2',b_fc2)
```

In the fourth part, we used Softmax as the classification activation function, cross entropy as the loss function , AdmaOptimizer as the gradient estimation function and built a function to record the accuracy of our training model.



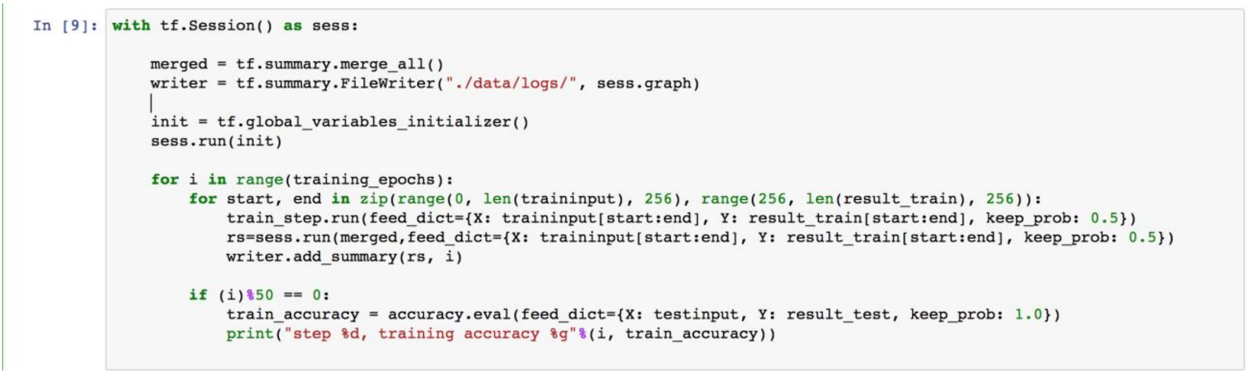
```
with tf.name_scope('activations'):
    y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```



```
with tf.name_scope('cross_entropy'):
    cross_entropy = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(y_conv), reduction_indices=[1]))
    tf.summary.scalar('cross_entropy', cross_entropy)

with tf.name_scope('train'):
    train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)

with tf.name_scope('accuracy'):
    with tf.name_scope('correct_prediction'):
        correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(Y,1))
    with tf.name_scope('accuracy'):
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
        tf.summary.scalar('accuracy', accuracy)
```

3.3 Model Accuracy Test

Finally , the main function is to launch the project work and get the output of our program, accuracy reflects the fact that when test voice data go through the data set, whether our model can tell if it is male's voice or not.



```
In [9]: with tf.Session() as sess:

            merged = tf.summary.merge_all()
            writer = tf.summary.FileWriter("./data/logs/", sess.graph)

            init = tf.global_variables_initializer()
            sess.run(init)

            for i in range(training_epochs):
                for start, end in zip(range(0, len(traininput), 256), range(256, len(result_train), 256)):
                    train_step.run(feed_dict={X: traininput[start:end], Y: result_train[start:end], keep_prob: 0.5})
                    rs=sess.run(merged,feed_dict={X: traininput[start:end], Y: result_train[start:end], keep_prob: 0.5})
                    writer.add_summary(rs, i)

                if (i)%50 == 0:
                    train_accuracy = accuracy.eval(feed_dict={X: testinput, Y: result_test, keep_prob: 1.0})
                    print("step %d, training accuracy %g"%(i, train_accuracy))
```

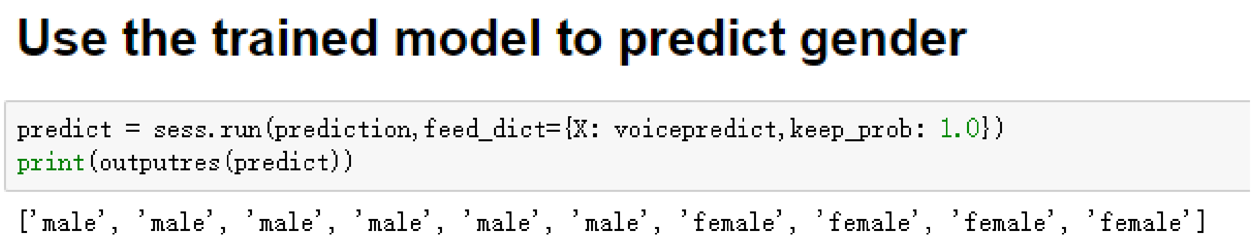
4. Result

For model train, the accuracy performance

Training epochs	Accuracy
Step 0	0.411028
Step 50	0.421053
Step 100	0.451128
Step 150	0.513784
Step 200	0.513784
Step 250	0.528822
Step 300	0.585466
Step 350	0.681704
Step 400	0.726817
Step 450	0.789474
Step 500	0.837093
Step 550	0.892331
Step 600	0.909774

Step 650	0.932331
Step 700	0.937343
Step 750	0.93985
Step 800	0.944862
Step 850	0.949875
Step 900	0.949875
Step 950	0.954887
Step 1000	0.954887
Step 1050	0.952381
Step 1100	0.954887
Step 1150	0.954887
Step 1200	0.957393
Step 1250	0.957393
Step 1300	0.957393
Step 1350	0.954887
Step 1400	0.954887
Step 1450	0.954887
Step 1500	0.954887
Step 1550	0.954887
Step 1600	0.954887
Step 1650	0.954887
Step 1700	0.954887
Step 1750	0.954887

To identify the gender of voice we choose 10 WAV sound files(1-6 male, 7-10 female) randomly to convert to our input data. The model performance is excellent.



Use the trained model to predict gender

```
predict = sess.run(prediction,feed_dict={X: voicepredict,keep_prob: 1.0})
print(outputres(predict))
```

```
['male', 'male', 'male', 'male', 'male', 'male', 'female', 'female', 'female', 'female']
```

5.Discussion

The whole process of this machine learning project is truly interesting. It help us go through the neural network model and many details. Convolutional neural network is suitable for the case that input contains too many dimensions, after the implement of convolution layer and max pool layer, the model would become more concise and readable.

During our implementation of this project, we put a lot of effort into the improvement of model accuracy. We tried to reshape image in different size, change the network architecture, loss function, activation function, gradient estimation and parameter initialization, as a result, 4-5

hidden layers, cross entropy and Relu function, AdamOptimizer, Gaussian initialization serves as the best combinations in our test.
In the end, our record has no null value and outliers, which means the dataset we processed is a ‘cleaned’ dataset, so we didn’t spend much time on EDA, dataset that has null value and outliers would be more challenging.

6.Reference

[1] Gender Recognition by Voice: https://www.kaggle.com/primaryobjects/voicegender
[2] Convolutional Neural Network for Visual Recognition: http://cs231n.github.io/convolutional-networks/
[3] Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks: https://arxiv.org/abs/1701.02720
[4] Improving End-to-End Models For Speech Recognition: https://research.googleblog.com/2017/12/improving-end-to-end-models-for-speech.html
[5] Shinji WatanabeMarc. New Era for Robust Speech Recognition. 2017