Audio Event Detection

Sahil Saini*

Koyel Pramanick†

M. Sc. Statistics,Department of Mathematics & Statistics, Indian Institute of Technology, Kanpur

Abstract

In our baseline model, we have trained our model model using traditional Convolutional Neural Network (CNN) by a sample of size 1600 (1500 for training and 100 for testing) from total 8732 observations of UrbanSound8K dataset. We are achieving training and testing accuracy of around 52% and 57%, both are not so good. Our main project aims at training a model with few shot learning which consists of k samples per class where $k = 1(1)5$. We use Siamese Network approach of few shot learning. Finally we get 91% as training accuracy and 75% as testing accuracy.

Index Terms- Audio Event Detection, Few shot learning, CNN, Siamese Network.

1 Introduction

After vision, sound is the second thing which helps humans and any most other animals to understand nature around us. When we listen some sound, we can immediately identify the sound, get their duration and cause of occurance. Automatic Sound Event Detection also called as Acoustic Event Detection (AED) is task of processing the audio to make computers understand sound like human brain understand audio signals that is to identify particular sound and their start and end time in audio clip. Our project aims at identify particular sound by a few sample of it in training data.

For these reasons, the thought came infront of us that if we can train machines to process and understand audio data as they have started to do for image data, a lot of works will be easier, applealing for less human effort and less time consuming.

Audio Event Detection is used in real life application like bio acoustic monitoring, Ambient Assisted Living Programme, self driving cars, Amazon Alexa and many more things.

But in real life data, it is not always possible to get a large number of data to train model. So training model with few shot learning method is becoming popular day by day and a very hot topic for machine learning and deep learning applications. We have shown here one such few shot learning method called siamese network.

Here in our project, we have trained our model to identify whether a particular sound is present in our given audio data by Siamese Network approach of Few Shot Learning Model.

2 Convolutional Neural Network

CNN was first used for image classification. Recently for varitiessound related works like speech recognition, audio event detection etc CNN is also using.

CNN is special neural network for processing data with grid like topology or sequential data. It has three features for reducing parameters- Sparse Interactions, Parameter Sharing and Equivalent Representation. CNN has four layers. Output from each layer is input for next layer except the first layer where input is an audio or image data:**Convolution Layer** provides meaningful, low dimensional, invariant feature space. This layer consists of set of learnable kernels. In **Activation Layer** some non-linear activation functions are applied on output of previous layer. **Pooling Layer**helps in downsampling features. **Fully Connected Layer** helps to run non-linear activation function like RELU.

Recently in CNN batch normalization is using to speed up and stabilize training data. Non-linear activation functions are applied after batch normalization.

*191118
†201333

3 Few Shot Learning

3.1 What is Few Shot Learning?

Few Shot Learning is 'Learning' or understanding some new image or audio data by a limited number of given sample in the training data. If we have to set face recognition lock system in our mobiles then we need Few Shot Learning model because to provide a lot of different photos of our face is somewhat time consuming.

Generally, any traditional machine learning model needs a huge amount of data for training to get high accuracy from testing data. But the more training data, the more dimension of input data and the more resource cost like time cost and computational cost of model training. Here, FSL can reduce cost of training.

To make machine understand a rare image or rare audio, or in cases where there are not sufficient number of people to collect sample, Few Shot Learning (FSL) is very much useful.

Actually FSL helps to get high accuracy during testing time with very less amount of training data.

We are using here C way K shot learning, where C is the number of classes which is fixed and K is number of samples per class. As K is very small number (K lies between 1 to 5) our goal will be to correctly classify a given unseen sample from any of these classes.

4 Siamese Network

4.1 Siamese Network for Few Shot Learning

Here siamese means 'twin'. There are two inputs in this network. They use same CNN (i.e. same set of weights). It gives output by with the help of Similarity Measure, numerical value indicating the extent of alikeness between two objects. It ranges between 0 to 1. 0 indicates the corresponding two objects are not alike. 1 indicates they are same. In practice, this measure takes non-negative values.In our report, we have calculated similarity measure between two audio data using C way K shot method.

4.2 Method

First one audio data from training set is taken and feature is extracted by applying convolution layer, pooling layer and flatten layer over this input.Let the input data be x and formed CNN be f. So, after extraction of features, the feature vector will be f(x) = h, say.

The same CNN (f) is used for each audio for feature extraction task from it. Here, suppose two inputs are x_1 and x_2. Then the feature vector will be $f(x_1)$ and $f(x_2)$.Therefore,the z-score is calculated as $|f(x_1)$-$f(x_2)| = z$ (say). Now we have applied dense layer over this z-score. Next we have applied sigmoid activation function over the scalar output coming from it. Now we get a number which lies in [0,1], which is the similarity measure between two given inputs, denoted as $sim(x_1, x_2)$. Now after getting the similarity measure, we have calculated loss function, use optimizer during backpropagation, perform a particular number of epochs to minimize the loss function and get the final model.

5 Data Formulation

5.1 Datasets

For experimentation we use UrbanSound8k dataset.It contains 8732 labeled sound excerpts ($\leq$ 4 sec) of urban sounds from 10 classes: air-conditioner,car horn, children playing, dog-bark, drilling, engine idling,gun-shot, jackhammer, siren, and street music.It also includes a CSV file containing meta-data information about every audio file in the dataset.This includes slice_file_ name, start, end, salience, fold, classID, class.We consider only 4 variables namely: slice_file_name, fold (location), classID and class.

For training purpose, we are making our own dataset by choosing 3 observations from each of 10 classes from UrbanSound8K dataset.

5.1.1 Positive and Negative Samples

Now here we will assign target value for each pair from our prior knowledge. Each sample from one class along with the other sample from same class forms a positive sample and we have assigned value 1 as target value. Each sample from a class with sample from another class is forming a negative sample. We have assigned value 0 as target value for this pair.

As testing data, we will take some unseen audio from remainnig audios of UrbanSound8K dataset.

Sample Pair For	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	Class 10
Same Class	+	+	+	+	+	+	+	+	+	+
Different Class	-	-	-	-	-	-	-	-	-	-

Table 1: Table showing each Class has one positive and one Negative sample

6 Model Architecture and Training

6.1 Feature Extraction Using CNN

First one audio data from training set is taken and feature is extracted by applying Convolution layer with 24 output nodes, Maxpooling layer with strides (4,2), Convolution layer with 48 output nodes, Maxpooling layer with strides (4,2), Convolution layer with 48 output nodes followed by flatten layer followed by dense layer with 64 output nodes. For convolution layers and dense layer Relu activation function is used.

Next lambda layer is used to get the absolute diifference between two extracted feature vectors. Finally one dense layer with **sigmoid** activation function is applied to get the final similarity measure.

6.2 Model Training

We trained the parameters of sister layers used for feature extraction and the Dense layer. Here binary cross entropy loss function is used along with Stochastic Gradient Descent optimizer in keras with learning rate 0.001 to update fully connected dense layer and CNN structure both, so as to minimize the loss function. We have done 50 epochs and metric 'accuracy' is used to get information about model in each epoch.

Our model architecture in detail is shown below:

```
Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)           [(None, 128, 128, 1) 0
__________________________________________________________________________________________________
input_2 (InputLayer)           [(None, 128, 128, 1) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                (None, 124, 124, 24) 624         input_1[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)              (None, 124, 124, 24) 624         input_2[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)   (None, 31, 62, 24)   0           conv2d[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D) (None, 31, 62, 24)   0           conv2d_3[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)              (None, 27, 58, 48)   28848       max_pooling2d[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)              (None, 27, 58, 48)   28848       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D) (None, 6, 29, 48)    0           conv2d_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D) (None, 6, 29, 48)    0           conv2d_4[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)              (None, 2, 25, 48)    57648       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)              (None, 2, 25, 48)    57648       max_pooling2d_3[0][0]
__________________________________________________________________________________________________
```

Figure 1: Model Architecture: Part 1

```
__________________________________________________________________________________________________
flatten (Flatten)              (None, 2400)         0           conv2d_2[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)            (None, 2400)         0           conv2d_5[0][0]
__________________________________________________________________________________________________
dense (Dense)                  (None, 64)           153664      flatten[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                (None, 64)           153664      flatten_1[0][0]
__________________________________________________________________________________________________
lambda (Lambda)                (None, 64)           0           dense[0][0]
                                                                dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                (None, 1)            65          lambda[0][0]
==================================================================================================
Total params: 481,633
Trainable params: 481,633
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 2: Model Architecture: Part 2

6.3 Model Testing

We will test the query audio by calculating similarity score between query audio and and example of training class. We will classify the audio to a particular class which has maximum similarity score.

7 Conclusion

Here we have shown use of Siamese network with only 30 samples (3 samples for each of 10 classes) and both training and testing accuracies are increased compared to our baseline model. Here our model is getting 91% accuracy in training and 75% accuracy in testing time.

8 Future Work

We have used here few shot learning without meta learning. To increase testing accuracy our future work will contain episodic training in meta learning.

References

https://dl.acm.org/doi/pdf/10.1145/3338472.3338476?casa_token=1HAQmZvRpecAAAAA:G6SFzkFLt65eiDG_9JKRpseTte--_hbs8Sz15S3Im8ewK73IIntEx-ilwSi6eug-hZk8gu6jJN73Hw