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 used traditional Convolutional Neural Network (CNN) for 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 = 3. We use Siamese Network approach of few shot learning. Finally we get 94% as training accuracy and 83% as testing accuracy. This model gets 60% accuracy in our demo data.

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.

Now, we know that human can identify sound, their time duration in an audio clip by hearing a few examples. But machine needs a lot of data to learn how to do task of Audio Event Detection. But in real life data, it is not always possible to get a large number of data to train model. For this reason, training a model with few shot learning method is becoming popular day by day and a very hot topic for machine learning and deep learning applications as it is able to learn from very little amount of data. Here in our project we have shown here one such few shot learning approach called **Siamese Network** by using **Keras** library. Our **project aims at to identify whether a particular sound is present in our given audio data by Siamese Network approach of Few Shot Learning Model.**

From the very much progression of Few Shot Learning in image data, the thought came in front of us that if we can train machines to process and understand audio data by Few Shot Learning approaches as they have started to do for image data, a lot of works will be easier, appealing for less human effort and less time consumption.

Some real life use cases of Audio Event Detection is applications like **Bio Acoustic Monitoring, Ambient Assisted Living Programme, Self Driving Cars, Amazon Alexa** and many more things.

In our report, we have discussed about Convolutional Neural Network framework in Section 2, about Few Shot Learning in Section 3, about Siamese Network in Section 4, about our required dataset in Section 5. Then in Section 6 Feature Extraction Using CNN, how we trained and tested our model with a few amount of data are discussed. In Section 7 we have discussed about showing model performance on a demo dataset followed by conclusion Section 8.

2 Convolutional Neural Network

CNN was first used for image classification. Recently for varities sound 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

*191118
†201333

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.

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 if our mobile requires a lot of pictures of our face to recognize us every time, it will be somewhat irritating and 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, or it may be very much costly to get many samples for each class, Few Shot Learning (FSL) is very much useful.Actually FSL helps to perform well 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 = 3) 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

Siamese Twin means **Monozygotic Twins**, as per medical terminology they are also called **Conjoined Twins**. In 1811, after **Chang and Eng**, the celebrated conjoined **Chinese twins** borns in **Siam (Thailand)**, conjoined twins are popularly known as **Siamese Twins**. This type of twins born due to incomplete division of one fertilized ovum. They are not completely separated from one another.

In Fig:1 we have shown the structure of Siamese Network. Here it is clearly shown that, initially there are two inputs in this network. They use same CNN configuration (i.e. same parameters and same set of weights) to extract feature from each input audio. But after that, model gives output with the help of Similarity Measure. For which from two different audio files there is one absolute difference followed by the Similarity Measure. This structure is similar as the whole model was performing separately over different audio inputs upto feature extraction and after that model works on both inputs together. That is why, this approach of Few Shot Learning is called as Siamese Network.

Siamese Network works by calculating Similarity Measure which is a 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.

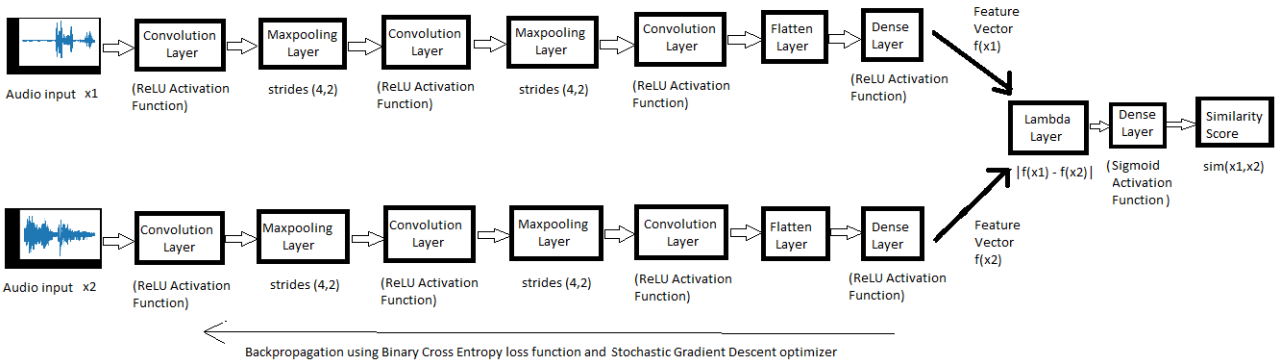

Figure 1: Siamese Network for Few Shot Learning (The two audio data used in the diagram are generated two random audios, just to make understand the model framework)

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 $|f(x_1)\text{-}f(x_2)| = z$, 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 (≤ 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.

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

As **testing data**, we will take some audios from remainig audios of UrbanSound8K dataset.

6 Model Architecture and Training

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]

(a) 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

(b) Model Architecture: Part 2

Figure 2: Model Architecture Obtained from Google Colab

6.1 Feature Extraction Using CNN

First one audio data from training set is taken and feature is extracted by applying **3 Convolution Layers, 2 Maxpooling Layers, Flatten Layer and Dense Layer. The details of these layers are shown in Fig:2.** For **Convolution Layers** and **Dense Layer ReLU** activation function is used.

Next **Lambda Layer** is used to get the absolute difference 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 twin CNN structures 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.

The curve of loss over time is shown below:

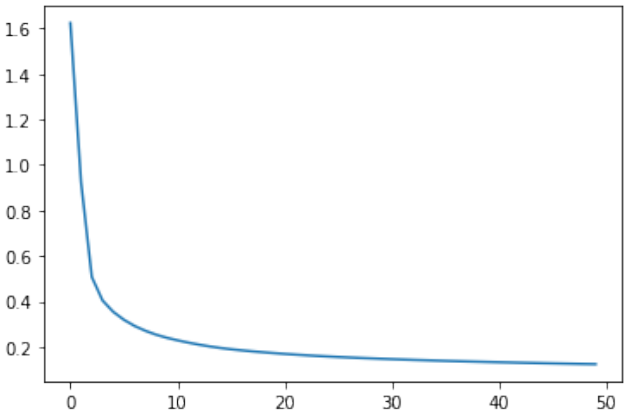

Figure 3: Loss Over Time during Model Training

6.3 Model Testing

We are performing test on the testing dataset that we made by taking some audios from the audio which remains in the UrbanSound8K dataset after making training dataset.

Here, we perform training and testing in each epoch. Starting from 19th epoch upto 50th epoch, we get the same training and testing accuracy. Finally we are stopping at 50th epoch with 94% accuracy from training data and 83% accuracy from training data.

7 Demo

To show our model is actually working, we have recorded some audios from movies and random youtube videos, converted them to .wav file using **Switch Sound File Converter** Application and perform during our presentation. Model performs quite well and gets 60% accuracy for demo dataset.

8 Conclusion

In this report we have shown use of **Siamese Network** with only 30 samples (3 samples for each of 10 classes) from UrbanSound8K dataset for training model and both training and testing accuracies are increased compared to our baseline model. Here our model is getting 94% accuracy in training and 83% accuracy in testing time, 60% accuracy in our demo data which we have shown during presentation of this project.

References

1. Snake Image Classification using Siamese Networks

(Click on the paper name to see the paper)

2. Few-shot learning(2/3): Siamese network— Shusen Wang—Online video, dated- Dec 3, 2020
(Click on the video name to see the paper)