Udacity-ML-Capstone Final Report
Dog Barking Detection

By Mike Lam
April 20th, 2018

Table of Contents

I. Definition

Project Overview

It is an Acoustic Event Detection (AED) classification task, but it can be turned into an image classification problem. Convolutional Neural Networks (CNN) have proven to be very effective in image classification and show promise for audio set. [1][2]

The transformation process requires knowledge of audio signal processing. Frequency of the audio is very important for identifying and classifying the source. Since in real life many signals are not stationary, the frequency domain will be changing through time. Turning the signal into time-frequency domain is a very popular technique. This is called time-frequency analysis and is very popular. [3]

Problem Statement

Puppies bark a lot for various reasons, for instance they might be scared, might be hungry, might hear someone walking past the house, etc. Owners would want to know if their dogs or puppies are barking, again for many reasons. The motivation of this project is for owners who cannot spend daytime with their puppies because of work and have to leave them in crate. Through detecting barking instances, a barking session can be defined as a period of consecutive barking. Then owners can know how many barking sessions happened during the day. This is a classification problem with 10-second audio inputs. The output is whether it is a dog barking or not (one versus all classification).

Evaluation Metrics

Since we are only interested in knowing whether a bark has occured, this image classification task is a binary classification problem, bark or not bark. This one-versus-all situation would result in an imbalanced dataset where majority is not bark. I have chosen two evaluation metrics for this project. The main one is cross entropy [6] and the supporting one is F1 Score [7].

Cross-entropy measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. The formula is as follow.

$$-(y \log(p) + (1 - y) \log(1 - p))$$

F1 score is the harmonic mean of recall and precision. Precision measures how many predictions made are correct, while recall measures how many positive test samples are picked up. F1 is a balanced measure of the two.

Since cross entropy focuses on probability, which is not affected by the decision threshold and the balance of dataset, I decided it is more appropriate to focus on cross entropy rather than F1 score as the metric as well as loss function. But F1 score is still a good supporting metric to review.

II. Analysis

Datasets and inputs

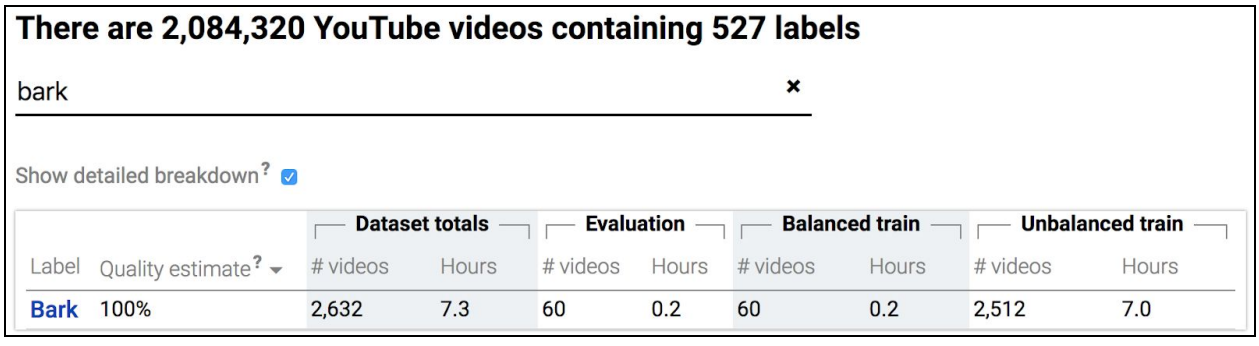
Google AudioSet [4] is a sound vocabulary. It consists of an expanding ontology of 632 audio event classes and a collection of over two-million human-labeled 10-second sound clips from YouTube videos. The ontology ranges from human and animal sounds to musical instruments and genres, and common everyday environmental sounds. Each video is labeled one or more classes. For this project, each label will be turned into bark or not bark.

The audios are stored on YouTube. A spreadsheet with the urls to the target videos is provided. Videos are uploaded by different people so there is no standard sampling rate. The whole dataset is imbalanced because only a small portion is dog barking videos. For barking videos, there are 0.2 hour of videos as training set and another 0.2 for evaluation set. The evaluation set will be split into validation and testing set evenly. There is another 7 hours of videos of which the quality is worse and imbalanced. In this project, several sizes of training data will be tried. Smaller and more balanced dataset will be tried, and the size is increased and more imbalanced dataset will be used. Two fixed size relatively balanced (50-50 or 40-60) dataset will be used for validation and testing.

The raw input will be a 10-second audio stored in wav file. The audio file is then pre-processed. The final input will be a series of frames of time-frequency signal in the form of images. The labels will be changed to bark/not bark. Then the problem will change from audio event detection to an image binary classification task. [1]

Exploratory Visualization

Google AudioSet contains 2M videos and only 2,632 videos contain barking. In these videos only 120 are balanced. And only these 120 are used for this project.



		Dataset totals		Evaluation		Balanced train		Unbalanced train	
Label	Quality estimate?	# videos	Hours	# videos	Hours	# videos	Hours	# videos	Hours
Bark	100%	2,632	7.3	60	0.2	60	0.2	2,512	7.0

Fig 1. Meta-data of all videos and barking videos in Google Audioset

Algorithms and Techniques

I intend to apply CNN to this problem. Barking of dogs and puppies should have a specific pattern which should be reflected in the time-frequency plot. Similar patterns should be observed in those plots when barking is occuring. As CNN is good at uncovering the abstract features that can describe the image, it is a good choice of algorithm to tackle this problem. Training CNN from scratch might not achieve good results because this dataset is not huge compared to YouTube-8M dataset [5]. Transfer learning [12] will be applied in an attempt to overcome this issue. Given spectrogram is quite different from the image datasets VGG is trained on and the training dataset is small, only early layers would be retained and the weights of them would be set to stay unchanged during training.

To discuss more about CNN [11], convolutional layer and pooling layer can be explained. Convolutional layer is a layer that convolutes the image with filter(s) and produces feature map(s). Each pixel of the feature map shows a weighted sum of the center pixel with the surrounding pixels of the image. The size of the filter specifies how big the window is. Usually in a convolutional layer, the image is convolved with multiple filters of different size. Each filter will be responsible at picking up different feature. Then each feature map contains information of whether the feature is present within that window.

Sometimes when dimensions get too large due to convolution, pooling layers are used to condense information in or across the feature maps. Common ways include averaging or getting the maximum value from a window. Averages try to bring in information within the window while maximum values pick up the most extreme attribute. In research papers there are researchers adopt some specific pooling layer, such as dynamic pooling.

Transfer learning is to take advantage of pre-trained models researchers have developed and trained. These models achieve state of the art results at different tasks, like image classification or captioning, etc. Since over the years researchers have found that the early layers from CNN usually extract relatively simple ideas, for instance edges and stripes, reusing the early layers would help save time to train.

Benchmark Model

A fully connected neural network will be used as the benchmark model. It will take flattened inputs from the images of time-frequency plot and output whether the given series of frames contain a bark. This model can contrast with the ability of CNN model that can retain and process spatial information.

III. Methodology

Google Audioset contains a total of 5.8 hours of video with manually annotated audio events. Barking dataset is about 0.2 hour long. For this project, the dataset will include barking and other events. Different size of training set will be tested. The smallest dataset will contain barking and 0.2 hour long of other audio events, so it will be balanced. Then the training set will include higher portion of other audio events, to make the dataset larger. Same principle will be followed when gathering dataset for training, validation and testing.

Preprocessing

Data preprocessing is as follows: Audio from every 10-second-long YouTube video is extracted. The audio is then being broken down into frames according to the chosen window size. After that each frame is being transformed into time-frequency plots (log-scaled mel-spectrograms). Librosa [8] will be used to create the plots. Each audio input becomes a stack of images with the same label.

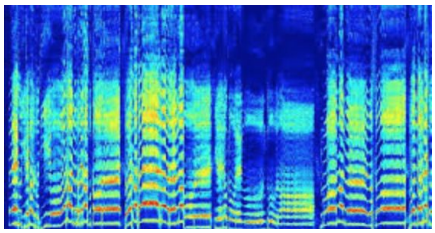


Fig 2. Example of a spectrogram

Implementation - Training

After preprocessing the data, the different models are going to be trained and evaluated. All models will have an output layer with one node (activation function is sigmoid) to show the probability of the input being a dog bark. As mentioned before, in this project, several different balance of training data will be tried. The ratio of positive to negative ratios tried are 1:2, 1:3, 1:5 and 1:10. Two fixed size, relatively balanced (1:1.5) dataset will be used for validation and testing.

a. Fully Connected Neural Network

First a fully connected neural network is fit to be the benchmark model. Two dense layers of size 10 and 5 are chosen as the baseline. To prevent overfitting given the small number of data versus the number of parameters, a dropout layer will follow each of the dense layer, with a dropout rate of 0.5.

Layer (type)	Output Shape	Param #
flatten_100 (Flatten)	(None, 6208)	0
dense_247 (Dense)	(None, 10)	62090
dropout_178 (Dropout)	(None, 10)	0
dense_248 (Dense)	(None, 5)	55
dropout_179 (Dropout)	(None, 5)	0
dense_249 (Dense)	(None, 1)	6

Total params: 62,151
Trainable params: 62,151
Non-trainable params: 0

Fig 3. Vaniila NN model summary

b. CNN from scratch

CNN models would be tried from scratch. The model will have six convolutional layers with ReLu activations to pick up relatively simple things like edges and shapes. They will have two fully connected layers at the end to learn representations. Dropout is added to each dense layer.

Layer (type)	Output Shape	Param #
conv2d_292 (Conv2D)	(None, 64, 97, 8)	104
max_pooling2d_291 (MaxPoolin	(None, 32, 49, 8)	0
conv2d_293 (Conv2D)	(None, 32, 49, 16)	528
max_pooling2d_292 (MaxPoolin	(None, 8, 13, 16)	0
conv2d_294 (Conv2D)	(None, 8, 13, 32)	2080
max_pooling2d_293 (MaxPoolin	(None, 1, 2, 32)	0
conv2d_295 (Conv2D)	(None, 1, 2, 32)	4128
max_pooling2d_294 (MaxPoolin	(None, 1, 1, 32)	0
conv2d_296 (Conv2D)	(None, 1, 1, 32)	4128
max_pooling2d_295 (MaxPoolin	(None, 1, 1, 32)	0
conv2d_297 (Conv2D)	(None, 1, 1, 32)	4128
max_pooling2d_296 (MaxPoolin	(None, 1, 1, 32)	0
flatten_126 (Flatten)	(None, 32)	0
dense_412 (Dense)	(None, 100)	3300
dropout_291 (Dropout)	(None, 100)	0
dense_413 (Dense)	(None, 100)	10100
dropout_292 (Dropout)	(None, 100)	0
dense_414 (Dense)	(None, 1)	101

Total params: 28,597
Trainable params: 28,597
Non-trainable params: 0

Fig 4. CNN model summary

c. Transfer Learning

Transfer learning would be tried attempting to get a lower cross entropy and higher F-1 score. For transfer learning, only the first few layers are kept because the images in this project's dataset are not similar to the original dataset and the training dataset is not large. VGG-16 [9] is implemented. After tuning, I decided to keep 10 layers from the original VGG model. Two dense layers are added after the retained layers. Dropout layer is added to each dense layer.

Layer (type)	Output Shape	Param #
input_71 (InputLayer)	(None, 64, 97, 3)	0
block1_conv1 (Conv2D)	(None, 64, 97, 64)	1792
block1_conv2 (Conv2D)	(None, 64, 97, 64)	36928
block1_pool (MaxPooling2D)	(None, 32, 48, 64)	0
block2_conv1 (Conv2D)	(None, 32, 48, 128)	73856
block2_conv2 (Conv2D)	(None, 32, 48, 128)	147584
block2_pool (MaxPooling2D)	(None, 16, 24, 128)	0
block3_conv1 (Conv2D)	(None, 16, 24, 256)	295168
block3_conv2 (Conv2D)	(None, 16, 24, 256)	590080
block3_conv3 (Conv2D)	(None, 16, 24, 256)	590080
block3_pool (MaxPooling2D)	(None, 8, 12, 256)	0
global_max_pooling2d_35 (Glo	(None, 256)	0
dense_379 (Dense)	(None, 100)	25700
dropout_269 (Dropout)	(None, 100)	0
dense_380 (Dense)	(None, 100)	10100
dropout_270 (Dropout)	(None, 100)	0
dense_381 (Dense)	(None, 1)	101

Total params: 1,771,389
Trainable params: 35,901
Non-trainable params: 1,735,488

Fig 5. Transfer Learning model summary

*ResNet50 [10] was in the plan, but it requires the size of image to be larger than the spectrogram I obtained.

Implementation - Testing

The goal of this project to predict whether a barking exists in a 10-second sound clip. Since the clip is decomposed into non-overlapping frames and turned into spectrogram, for all frames that belong to the same sound clip, their predicted probability will be averaged to compute log loss and give a final prediction.

The threshold to classify whether the sound clip contains a dog bark is changed in each setting. It is computed by the 1/(1+r), where r is the ratio negative to positive samples. The intuition is that with more and more imbalanced dataset, a random chance that a sound clip contains dog park will be lower and lower.

Refinement and Model Tuning

All models' output layer is one node with sigmoid activation, so they output the probability of the frame containing a bark or not. I try to keep the number of parameters lower since there is only 5180 training data. Vanilla NN has two hidden layers after the flatten layers, and about 62k parameters. CNN has 6 convolutional layers and max pooling layers, followed by the flatten layer, a total of about 15k parameters.

For transfer learning, one of the decisions to make is how many layers should be used from the pre-trained VGG16 model. I decided to try keeping layers before each max-pooling layer in the original model. Therefore grid search is performed on validation data and the layers to keep are 3, 6, 10, 14 and 18. Performance when positive to negative ratio is 1:2 is shown below. It is observed that the log-loss decreases with the layers to keep up to 10, then it starts to increase. This is observed over all ratios of positive to negative samples. Therefore 10 layers are retained in the final model and their weights are set to untrainable. Two dense hidden layers are connected to the retained model. The training would only train the two hidden layers before output, a total of 35k parameters.

First n layers to keep	Log-loss (LL)	F1 (threshold = 0.5)
n=3	0.5807	0.3750
n=6	0.5601	0.3750
n=10	0.4723	0.8696
n=14	0.5259	0.424
n=18	0.6080	0.207

Table 1. Effect of layers to keep from VGG16

IV. Results

Model Evaluation and Validation

As described before, different training set is used to train the three different models, and test on the fixed, relatively balanced (1:1.5) test set. The threshold for classification is changed for each balance setting of the training set. The results are shown below.

Pos-to-Neg	Vanilla NN	CNN	Transfer Learning (k=10)
1:2 (t=0.333)	LL: 0.5920, F1: 0.6977	LL: 0.5445, F1: 0.3530	LL: 0.5119, F1: 0.5641
1:3 (t=0.250)	LL: 0.6027, F1: 0.0	LL: 0.5190, F1: 0.3530	LL: 0.5293, F1: 0.6222
1:5 (t=0.167)	LL: 0.6448, F1: 0.0	LL: 0.5489, F1: 0.3530	LL: 0.4110, F1: 0.6222
1:10 (t=0.09)	LL: 0.7168, F1: 0.0	LL: 0.5456, F1: 0.3530	LL: 0.4644, F1: 0.6222
average	LL: 0.6390	LL: 0.5395	LL: 0.4792

Table 2. Model performance on testing sets under different training balance setting

Justification

I. Layers to keep from VGG16

The effect of number of layers to retain from the original VGG16 is displayed in table 1 in section 3. As observed the log loss decreases, then increases again after n= 10. I would interpret as the earliest 10 layers from the original model are picking up features that are useful in generalized tasks, maybe like picking up edges, shapes, etc. As the number of layer gets over 10, the features extracted become more abstract and more specific to the original task it is trained on.

II. Performance on testing sets under different training balance setting

A summary of table 2 is that transfer learning performs the best out of the three model under all balance setting of training dataset.

The baseline model, vanilla neural network performs the worst and as the dataset becomes more imbalanced the performance decreases further. F1 score immediately turns to 0 since there is no positive sample correctly detected. This is in line with expectation in general direction, although I did not expect F1 to be 0. CNN performs better than the vanilla NN with fewer parameters. That is matching the expectation since CNN could make use of spatial data instead of flattening the input directly. Results from transfer learning are even better than CNN. On average, it's transfer learning with 10 layers from VGG16 achieves log-loss 0.04 below CNN and 0.15 below baseline vanilla NN.

In addition to the average, it is observed that for vanilla NN, the performance decreases when the training set data becomes more imbalanced. This problem does not seem to show for CNN and transfer learning.

A performance of 0.48 as log loss can approximately correspond to around 62% of times the model is outputting the right class. I don't think in practice this is a good enough accuracy to develop a product around this. However, this approach shows potential and the model

performance can be improved a lot if more tuning is carried out about the detail of architecture. Even the problem setting and data preprocessing can be fine tuned.

V. Conclusion

Visualization

Here is a real spectrogram containing background noise and a dog bark. It is not obvious that when the events occured. That's why there is a need to rely on machine learning, and furthermore CNN that is designed for image tasks, to classify whether barking occured.

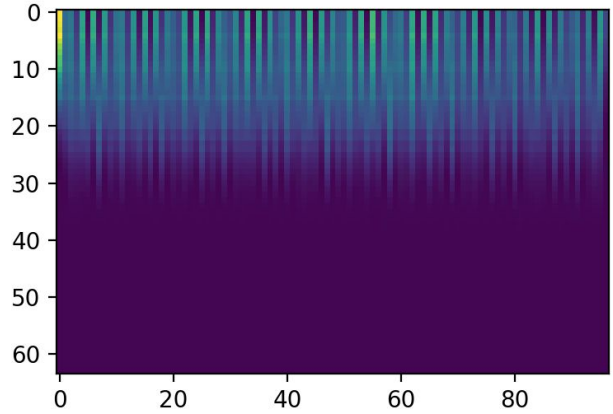


Fig 6. Spectrogram sample from real data used

Reflection

The workflow of this project can be broken down into 3 stages.

1. Data collection
    - First the positive videos are all downloaded
    - Then since the most imbalanced case I plan to explore is when negative samples are 10X of positive ones, I downloaded 10X of negative videos, sampled uniformly from all other tags available.
2. Data preprocessing
    - Videos are turned to sound clips.
    - The sound clips are then divided into non-overlapping window, and mel-spectrogram is obtained from each frame. All frames would carry the same label (positive or negative)
    - Google Audioset provides training set and evaluation set of videos, I used the whole training set as training set, and split their evaluation set in half to be the validation set and testing set.
3. Construct models and tune transfer learning model, testing
    - Baseline models, CNN and transfer learning model are constructed.
    - Tuning for transfer learning model refers to how many layers from the original VGG16 are retained.

    - Since the model spits out class probabilities for each frame, for each sound clip, the probabilities have to be averaged to get the final log-loss and the F1 score.

Difficulties of the project include the data downloading and the preprocessing work. Signal processing knowledge is required for this and it is something extra from the course materials. Building structured codes is a fun aspect of the project, functions are defined in different scripts and imported as needed, for example models are built from a script and the compiled models are imported when training. Having such structure makes testing and tuning easy.

Improvement

There are mainly two aspects to further work on to improve performance.

1. Preprocessing of data
    - Manual breakdown of the positive samples can be done to label the specific frames containing baking, to provide a more accurate representation of the spectrogram that the models can rely on.
    - Different window size to divide the sound clip can be chosen. Overlapping window can be explored. Different windowing function can be tried.
2. Cross validation
    - Breaking the dataset into training, validation, testing set is only done once, it could be done according different seeding to get more generalized results.
3. Details to model tuning
    - There are some model parameters that this project did not focus to tune since the focus is to explore the basic ability of the three models and the effect of imbalanced dataset. A simple structure is empirically chosen to give the general feeling of the model power.
    - Number of fully connected hidden layers and hidden nodes can be tuned by random search to get the best out of each model.

Reference

[1] Hershey, Shawn, et al. "CNN architectures for large-scale audio classification." Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE, 2017.
[2] https://www.skcript.com/svr/building-audio-classifier-nueral-network/
[3] https://www.mathworks.com/help/signal/examples/practical-introduction-to-time-frequency-analysis.html
[4] Gemmeke, Jort F., et al. "Audio set: An ontology and human-labeled dataset for audio events." Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE, 2017.
[5] Abu-El-Haija, Sami, et al. "Youtube-8m: A large-scale video classification benchmark." arXiv preprint arXiv:1609.08675 (2016).
[6] http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy
[7] https://en.wikipedia.org/wiki/F1_score
[8] https://librosa.github.io/librosa/
[9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
[10] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[11] http://cs231n.github.io/convolutional-networks/
[12] http://cs231n.github.io/transfer-learning/