SMARTBARK: AN AUTOMATED DOG BARKING DETECTION AND MONITORING SYSTEM USING AI AND DEEP LEARNING

Xiangjian Liu[1], Yishan Zou[2] and Yu Sun[3]

[1]Arnold O. Beckman High School
[2]University of Pennsylvania, Philadelphia, PA 19104
[3]California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Dogs have the tendency to bark at loud noises that they perceive as an intruder or a threat, and the hostile barking can often last up to hours depending on the duration of such noise. These barking sessions are unnecessary and negatively impact the quality of life of the others in your community, causing annoyance to your neighbors [1]. Having the rights to file noise complaints to the Home Owners Association, potentially resulting in fines or even the removal of the pet [2]. In this paper, we will discuss the development of an algorithm that takes in audio inputs through a microphone, then processes the audio and identifies that the audio clip is dog barks through machine learning, and ultimately sends the notification to the user. By implementing our application to the everyday life of dog owners, it allows them to accurately determine the status of their dog in real-time with minimal false reports.

KEYWORDS

Monitoring System, Deep Learning, AI, mobile platform

1. INTRODUCTION

Noise pollution, or excessive unwanted sound, is a problem worldwide, causing nuisances ranging from annoyance, mental breakdowns, to deafness [3]. In 1981, the Environmental Protection Agency estimated that nearly 50% of the people in the United States were exposed to harmful noise pollution [5]. Consequently, homeowners would contact local authorities to remove sources of noise pollution [2]. Without proper communication between homeowners, these disputes could often end up in unnecessary fines and penalizations [4] in preventable situations, especially against dog owners. Dogs, as tame, loyal pets, could be easily controlled by their owner's commands. Their barks, however, are purely instinctive and often unable to be trained.

Some of the pre-existing solutions have been proposed regarding sound event detection (SED), most of which allow the user to check in on their home environment in real-time. More advanced monitors offer the functionality of recording sound files and storing them in a database, allowing the user to access archived sound files. These proposals offer a decibel based detection method, meaning that sound will be recorded if the loudness of the sound is over a certain decibel. However, these solutions require the user to determine the sound and are incapable of identifying sound files alone.

David C. Wyld: CSITY, NWCOM, SIGPRO, ASOFT, AIFZ, BDIoT, ITCCMA, CLSB, DTMN, MLNLP - 2021
pp. 13-25, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111402

Our method follows the same premise in monitoring noise in real-time along with some additional features that are crucial to the user experience. First, our method allows for accurate and precise classification [6] of sound files of 10 categories, specifically dog barks. Though our method is exclusive for dog barks, it provides a high customizability so that it could be specialized for classification and notification of other sounds. Second, the implementation of cloud messaging [7] and app notifications lets the user know if the barking is ensuing in real-time. Each entry of identified dog bark is recorded with corresponding confidence score [8] and time recorded, and transmitted to the user's application, allowing the user to identify the need to go back.

With the goal of achieving real-time prediction and notification in mind, we also needed to minimize false predictions by producing an efficient model. In two application scenarios, we demonstrate how the variation of elements during model training substantially impacts the program's performance during testing. First, we prove the effectiveness of utilizing a larger set of training data by comparing the accuracy of models trained by datasets of different sizes. Second, we weigh the impact of including additional categories of prediction results on prediction accuracy. Through trials of experiment, we compare the mean prediction accuracy of each variation in these two experiments in order to find the most efficient arrangement of these elements.

The rest of the paper is organized as follows: Section 2 scrutinizes the challenges that we faced during the experiment and design process; Section 3 provides detail on our solutions to the corresponding challenges mentioned in Section 2; Section 4 presents relevant details regarding the experiment, followed by presenting related work in Section 5. Finally, Section 6 concludes this research paper, as well as outlining future work of this project.

2. CHALLENGES

2.1. Challenge 1: How to Build the Training Model Without Having the Available Data From the Specific Dogs

In order to build a model with high accuracy and precision, a vast amount of data would be needed while training the model. This poses a challenge to our team as not only do we have to record hundreds to thousands of sound files of different categories, these sound files would not provide enough diversity for the program to work accurately on the dog barks of different breeds

2.2. Challenge 2: How to Perform Real-Time Bark Sound Detection

With the premise of notifying users the current state of their dogs, our project must perform both data collection and prediction in real-time. With each sound file recorded, a corresponding prediction needs to be made without delay to ensure the resulting predictions reflect the condition of the environment at all times. At first glance, the usage of a cloud server becomes a possibility as it provides ample processing power to ensure low prediction run time, but local predictions also provide accurate predictions without server delay.

2.3. Challenge 3: How to Integrate Components From Different Platforms With a Low Latency

Data collection, sound processing, prediction, and notification constitute this system, and the integration of these components with low latency poses another challenge for us. Aside from creating a system with real-time bark detection and prediction, the user should also be notified in

real-time when their dogs are barking. To accomplish this goal, the usage of a database is necessary for the retrieval and storage of data, and the user's mobile device should be connected to the database for easy access. However, the efficiency of these methods are yet to be determined for this system to be declared as real-time.

3. SOLUTION

3.1. Overview of the Solution

In order to perform predictions on audio inputs and notify the user in real-time, a three-part system was developed. As shown in Figure 1, The Dog Bark Detector is a system that records audio inputs in 5-second intervals through a microphone on a Raspberry Pi, which then performs predictions locally through a trained model on every sound file. If the program predicts that the sound file signifies a dog bark, information such as the time of the event and the confidence score is then transmitted to the online database, as the Raspberry Pi is connected to the internet. Along with the storage of that information in the database, a notification is also sent to the user's mobile device, notifying them that their dog is likely barking at the moment. The user is also able to access all of the archived information in the database on their mobile application.

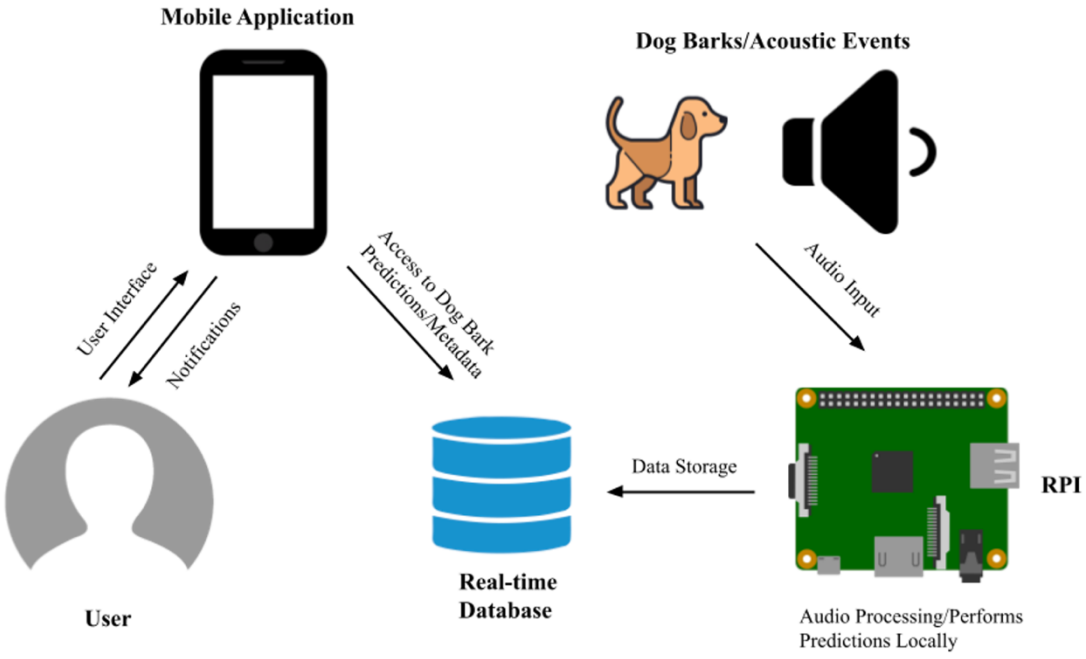

Figure 1. Graphic Representation of the solution

3.2. Machine Learning

a. Introduction

To get started with the core component of this system, we needed to find an available dataset to train our machine with, as recording thousands of sound files and manually labeling them is a time-consuming process. Furthermore, we needed to find an adequate library for predictions.

b. Dataset

In order to have enough training data to produce a model with both accuracy and precision, a copious amount of diverse, categorized sound files would be needed. However, achieving this goal would not be possible in a reasonable amount of time, so we utilized an online dataset. UrbanSound8K [9] provides a variety of training data of 10 categories, up to 8732 sound files.

c. Training Model

d. Attach Some Code

The following Figure 2 shows the utilization of the Neural Network during model training.

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

model = keras.Sequential()

input_layer = keras.layers.Dense(3, input_shape=[3], activation = 'tanh')
model.add(input_layer)
output_layer = keras.layers.Dense(1, activation = 'sigmoid')
model.add(output_layer)

gd = tf.train.GradientDescentOptimizer(0.01)

model.compile(optimizer = gd, loss = 'mse')

training_x = np.array([a
    [1,1,0],
    [1,1,1],
    [0,1,0],
    [-1,1,0],
    [-1,0,0],
    [-1,0,-1],
    [0,0,1],
    [1,1,0],
    [1,0,0],
    [-1,0,0],
    [1,0,1],
    [0,1,1],
    [0,0,0],
    [-1,1,1],
])
```

```
training_y = np.array([
    [0],
    [0],
    [1],
    [1],
    [1],
    [0],
    [1],
    [0],
    [1],
    [1],
    [1],
    [1],
    [1],
    [0]
])

model.fit(training_x, training_y, epochs = 1000, steps_per_epoch = 10)

test = np.array([[1,0,0]])

prediction = model.predict(test, verbose = 0, steps = 1)

print(prediction)A

model.save_weights('model1.h5')
```

Figure 2. Model Training and Neural Net

3.3. Raspberry PI

a. Installation

During the installation process of the Raspberry Pi, we decided to include the addition of an extended USB adapter cord. The reasoning behind this is that the Raspberry Pi 4 includes a cooling fan to prevent overheating over a prolonged period of runtime. If the microphone for audio input were to be connected to the Raspberry Pi IO directly, the fan-produced background noise interferes with the audio, and could potentially throw off the predictions.

b. Libraries

Figure 3 shows the library integrated into the program, while Figures 4 and 5 present snippets of the program.

```
import numpy
import time
import pyaudio
import datetime
import wave
from multiprocessing import Process
import tensorflow as tf
import librosa
import numpy as np
import requests
import json
import firebase_admin
from firebase_admin import credentialsS
from firebase_admin import firestore
```

Figure 3. Imported Libraries

c. Code

```
model = tf.keras.models.load_model('my_model.h5')

def get_prediction(model, wav_file):

    dat1, sample_rate = librosa.load(wav_file)
    mels = np.mean(librosa.feature.melspectrogram(y=dat1, sr=sample_rate)
    .T,axis=0)
    arr = mels.reshape(1,16,8,1)
    pred = model.predict(arr)
    pred_index = np.argmax(pred, axis = 1)[0]
    print(pred_index, pred[0][pred_index])
    return pred_index, pred[0][pred_index]
```

Figure 4. Prediction Function

```
while True:
    time.sleep(2)
    frames = []
    for i in range(0, int(sampleRate / streamChunk * seconds)):
        data = stream.read(streamChunk, exception_on_overflow = False)
        frames.append(data)
    fileName = saveToWav(frames)
    print("Done")
    rawsamps = stream.read(streamChunk, exception_on_overflow = False)
    samps = numpy.fromstring(rawsamps, dtype = numpy.int16)
    pred, confidence = get_prediction(model,fileName)
    record = {}
    print(sound_dict[pred])
    if pred == 3:
        record[str(int(time.time()))] = str(confidence)
        response = requests.put
        ('https://dog-bark-detector.firebaseio.com/device/mobile/timestam
        p.json', data=json.dumps(record))
        deviceTokens = get_device_tokens()
        for token in deviceTokens:
            body = {'notification': {'title': 'Notification from Dog
            Bark Detector','body': 'Your Dog Barked!'},'to': token,
            'priority': 'high'
            }
            response = requests.post
            ("https://fcm.googleapis.com/fcm/send",headers = headers,
            data=json.dumps(body))
            print(response)
```

Figure 5. Real-time Recording, Prediction, and Integration of Firebase

d. How the Code Works

The program was written in Python 3, ran on Raspberry Pi's built-in IDE Thonny, and employs machine learning library TensorFlow [10]. TensorFlow has a relatively higher training time compared to other libraries such as Pytorch [11] and Scikit-learn, but has lower prediction execution time and lower memory usage, therefore more adequate for Raspberry Pi. In our case, model training time is immaterial as all predictions are made with a pre-trained model, so TensorFlow stands out as the best choice.

3.4. Mobile App

a. Each Screen (Screenshots)

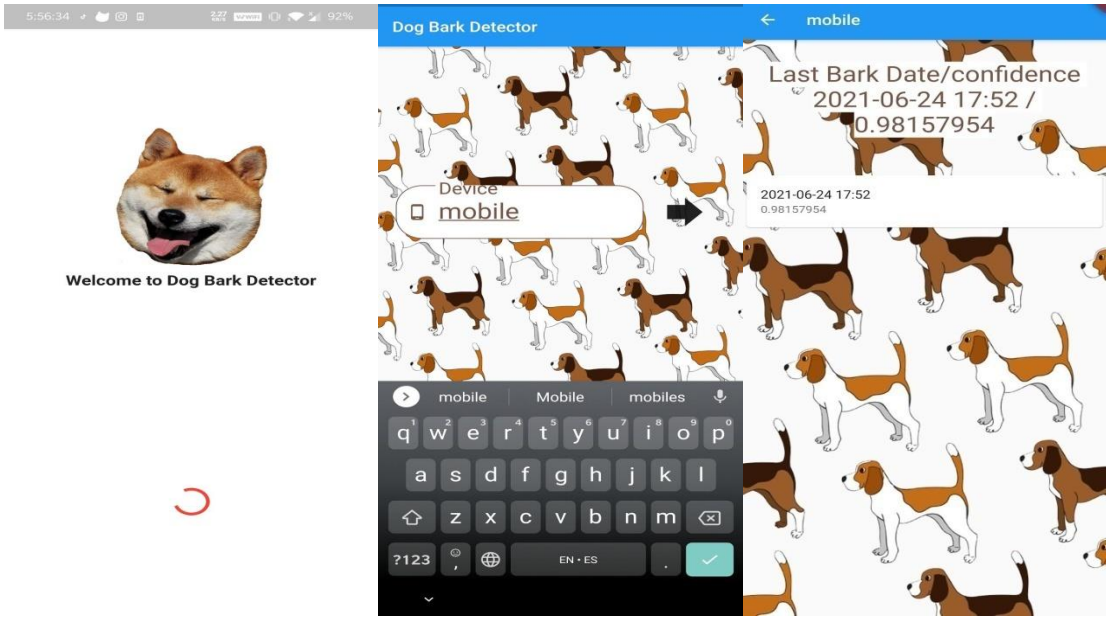

Figure 6. Screenshots of the screens

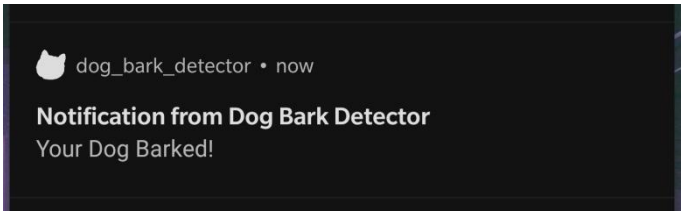

Figure 7. Notification Bar

a. How Each Screen Works

As the mobile application boots up either through the notification or home screen icon, a 5-second long loading screen would be shown to indicate that the application is booting up. The user is then required to input the device's name to access the data. A page of recent history of dog barks is then shown along with the exact time each entry was recorded and the corresponding confidence score.

b. Flutter Code

```
import 'dart:async';
import 'dart:io';
import 'package:firebase_database/firebase_database.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_core/firebase_core.dart';

class Server {
  FirebaseFirestore _db;
  final databaseReference = FirebaseDatabase();
  final FirebaseMessaging _firebaseMessaging = FirebaseMessaging();

  Server() {
    _init();
  }
  _init() async {
    await Firebase.initializeApp().then((value) {
      _db = FirebaseFirestore.instance;
    });
  }

  Future getTimestamps(String device) async {
    DataSnapshot ds = await databaseReference
        .reference()
        .child("device/" + device + "/timestamp")
        .once();
    print('********getting timestamps: ' + ds.value.toString());
    return ds.value;
  }

  Future<String> saveDeviceToken() async {
    await Future.delayed(Duration(seconds: 2));
    String fcmToken = await _firebaseMessaging.getToken();

    if (fcmToken != null) {
      var tokenRef =
          _db.collection('tokens').doc(fcmToken);
      await tokenRef.set({
        'token': fcmToken,
        'createAt': FieldValue.serverTimestamp(),
        'platform': Platform.operatingSystem
      });
    }
  }
```

4. EXPERIMENT

4.1. Experiment 1

With the aim of improving the accuracy and precision of the model, we performed an experiment on prediction accuracy with varying dataset sizes. 10 trials are performed on three different models trained with 200, 400, and 800 sound files each for 10 categories of sounds. With each trial, we played a two-minute long sound file of solely dog barks, and recorded the percentage of times the machine predicted correctly.

Table 1. Experiments 4.1 Results

	Mean Prediction Accuracy With 200 Files	Mean Confidence Score With 200 Files	Mean Prediction Accuracy With 400 Files	Mean Confidence Score With 400 Files	Mean Prediction Accuracy With 800 Files	Mean Confidence Score With 800 Files
Trial						
1	0.6898790124	0.8424729302	0.7559623831	0.8314935261	0.7631219106	0.7884614562
2	0.5912350912	0.7129731386	0.7066988174	0.8999849948	0.7921745436	0.8605691466
3	0.6581090928	0.8561823995	0.5864704471	0.8511170761	0.8597018772	0.7721071216
4	0.7362699784	0.7654463363	0.7819248494	0.7761451834	0.8241789856	0.8606500193
5	0.6383840589	0.9582189809	0.8404772978	0.7871010207	0.9029941285	0.8178276164
6	0.4983447521	0.8827262971	0.6831437616	0.8652125737	0.8908187336	0.8019187893
7	0.5379288024	0.6957023572	0.8202670667	0.8943896996	0.8373232305	0.9108755031
8	0.5818455893	0.8942314855	0.7184482631	0.7464330883	0.8736614863	0.8825037595
9	0.5860662396	0.7900477937	0.6779488965	0.9123124168	0.7590682019	0.7547074734
10	0.5147576467	0.8228336562	0.7335328985	0.8974635624	0.7108356222	0.9080990634

Though training the model with a larger dataset provides substantially higher accuracy, there was little to no change in the confidence score. By introducing more well-labeled data, we have significantly improved the accuracy of the model but with diminishing returns.

4.2. Experiment 2

After determining that a larger dataset provides better accuracy, we conducted another experiment to see if the number of categories of sounds affect the prediction accuracy and confidence. The setup of this experiment follows the first experiment, with the difference of varying categories instead of dataset sizes. We trained all three models with 800 sound files for each category but with varying 2, 5, 10 categories.

Table 2. Experiments 4.2 Results

	Mean Prediction Accuracy With 2 Categories	Mean Confidence Score With 2 Categories	Mean Prediction Accuracy With 5 Categories	Mean Confidence Score With 5 Categories	Mean Prediction Accuracy With 10 Categories	Mean Confidence Score With 10 Files
Trial						
1	0.8855852903	0.8885125829	0.8390956366	0.7043971628	0.8556732618	0.7331123179
2	0.9042318007	0.9562397262	0.8286099065	0.7183265602	0.7935607238	0.7191559346
3	0.7224606445	0.8487778692	0.8523785396	0.8008522239	0.8251472019	0.7119563546
4	0.8844939738	0.9451752134	0.7721262744	0.8124913883	0.8443776904	0.6221453087
5	0.8281247652	0.8292085486	0.8791382616	0.7974484484	0.7656544446	0.6704566329
6	0.8409141153	0.8064941977	0.7566873349	0.7993017992	0.7845480411	0.8042016261
7	0.8972567475	0.8709239151	0.8998339055	0.7089790011	0.8675134809	0.7083710241
8	0.7063986176	0.7768236345	0.8594534284	0.748812237	0.8251472019	0.6794634922
9	0.8023372542	0.8553321055	0.7638194263	0.7413257356	0.7707758818	0.7280253879
10	0.7796276999	0.8458663361	0.7305136769	0.7044689465	0.8567961605	0.6494426147

The introduction of new categories affected the mean confidence of predictions due to more similarities between more categories, but the similarities were not indistinguishable, and were unable to affect the prediction accuracy.

4.3. Analysis

Through trials of experimentation, we were able to assess the overall effects of varying dataset size and categories. The increase of data in the dataset resulted in an overall improvement in prediction accuracy, which is reasonable as all of the data we used were accurately labeled. With the increase of categories, however, was able to negatively affect the prediction confidence, though not able to significantly affect prediction accuracy. The decrease in prediction confidence could be caused by the similarities between each type of sound, but the sounds are not similar enough to affect the actual prediction results.

5. RELATED WORK

Along the same lines of work, Mesaros, A. et al [13] utilize TUT Acoustic Scenes database to train models with the ability of sound event detection, specifically environmental acoustic scenes. In comparison, their data training employs an unsupervised Gaussian mixture model, while we utilized a supervised neural network as our dataset was manually labeled. In order to produce an accurate model through their method, it is safe to assume that a larger dataset is needed. In a scenario where training data is not abundant, our method of supervised training would be preferable, though with the downside of time-consuming manual labeling.

Adavanne, S. et al [14] investigate the difference between monaural and binaural sound event detection by studying different binaural features. The paper explores the substantial improvement in prediction accuracy along with the ability of classifying polyphonic sound events. Their

method combines convolutional neural network [15] with recurrent neural network, producing a model that requires minimal supervision on pre-processing and secures high accuracy in polyphonic sound events. In our case, having the ability of polyphonic detection would be beneficial as the overlapping of sounds is a common occurrence in communities.

Wang, Y. [16] explores the advantages of transfer learning with a rather incomplete, poorly labeled dataset. Wang, Y. concluded that linear softmax pooling produces the best performing model through Convolutional Neural Network in the absence of a large, well-labeled dataset. Transfer learning, as discussed in the paper, would require the source task to have over 50 times the amount of data of the target task to achieve a successful application. The implementation of transfer learning, though proven to be useful, would not be realistic in our scenario as we have access to a rather large, well-labeled dataset.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed an operational system that encompasses efficiency, accuracy to a sizable problem regarding the well being of community residents. The Dog Bark Detector allows users to be notified of the current ambience of their home environment as a form of automated system. The integration of a real-time database, real-time predictions and cloud messaging allows the user to receive notifications with minimal delay. Through trials of experiment, we were able to prove the high prediction accuracy of our program by finding the most efficient combination of training features. We found that, though with diminishing returns, simply increasing the size of the dataset with well-labeled data increases the accuracy of predictions. We also explored the impact of adding additional categories during model training, and came to a conclusion that adding categories of prediction results decreases the confidence score of prediction and its effects towards prediction accuracy are negligible.

The system has much to improve on, such as its prediction accuracy and practicality. The current model was trained through a dataset provided by UrbanSound8K. Despite having over 8732 sound files, only a small fraction of the sound files were dog barks as the dataset covered 10 categories ranging from “Jackhammer” to “Siren”. This significantly limits the amount of soundfiles going towards the training of specifically dog barks. One way to improve the accuracy of the predictions is to include more dog barks sound files in the dataset.

These limitations could be solved through either searching for a larger labeled dataset to produce a model that predicts with minimal errors, or include a verification segment where the program only sends notifications to its user when multiple predictions appear to be dog barks in a short span of time.

In K-12 educational settings, the program can be modified to strengthen the connection between teachers and administration and solve the in-class emergencies. For example, when a physical altercation takes place in class, since teachers are not allowed to touch the students, they need to let the class administration know about the situation. In this case, teachers can label the assistance that are needed to stop a physical altercation as “help one”. When the program predicts that the sound file signifies “help one”, the information of classroom number and teacher’s name will be sent to the school office. When the class administration receives the notification, the school secretary or principal can come over and stop the fight. In the same way, medical emergencies can be labeled as “help two”. When it happens, the teacher can perform CPR while reaching out to the administration to help call the ambulance.

REFERENCES

[1] Jégh-Czinege, Nikolett, TamásFaragó, and PéterPongrácz. "A bark of its own kind–the acoustics of 'annoying' dog barks suggests a specific attention-evoking effect for humans." Bioacoustics 29.2 (2020): 210-225.

[2] "HOA Noise Rules: Complaining About Neighbor's Party Noise: HOAM." HOA Management, 10 Dec. 2020, www.hoamanagement.com/hoa-noise-rules/.

[3] Singh, Narendra, and Subhash C. Davar. "Noise pollution-sources, effects and control." Journal of Human Ecology 16.3 (2004): 181-187.

[4] Yang, Honggang. "The disputing process: An ethnographic study of a homeowners association." Mediation Quarterly 13.2 (1995): 99-113.

[5] Hammer, Monica S., Tracy K. Swinburn, and Richard L. Neitzel. "Environmental noise pollution in the United States: developing an effective public health response." Environmental health perspectives 122.2 (2014): 115-119.

[6] Casey, Michael. "General sound classification and similarity in MPEG-7." Organised Sound 6.2 (2001): 153-164.

[7] Moroney, Laurence. "Firebase cloud messaging." The Definitive Guide to Firebase. Apress, Berkeley, CA, 2017. 163-188.

[8] Mandelbaum, Amit, and Daphna Weinshall. "Distance-based confidence score for neural network classifiers." arXiv preprint arXiv:1709.09844 (2017).

[9] Garg, Srishti, et al. "Urban Sound Classification Using Convolutional Neural Network Model." IOP Conference Series: Materials Science and Engineering. Vol. 1099. No. 1. IOP Publishing, 2021.

[10] Dillon, Joshua V., et al. "Tensorflow distributions." arXiv preprint arXiv:1711.10604 (2017).

[11] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019): 8026-8037.

[12] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

[13] Mesaros, Annamaria, Toni Heittola, and Tuomas Virtanen. "TUT database for acoustic scene classification and sound event detection." 2016 24th European Signal Processing Conference (EUSIPCO). IEEE, 2016.

[14] Adavanne, Sharath, and Tuomas Virtanen. "A report on sound event detection with different binaural features." arXiv preprint arXiv:1710.02997 (2017).

[15] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." 2017 International Conference on Engineering and Technology (ICET). Ieee, 2017.

[16] Wang, Yun. "Polyphonic sound event detection with weak labeling." PhD thesis (2018).