Domestic Audio Classification

Taqiya Ehsan
RUID: 194000857

Rutgers University, New Brunswick

Fall 2021

Abstract

In this paper, we build a pipeline architecture for an audio classifier that can classify different domestic sounds. This framework is created to facilitate activity recognition by MAESTRO sensors deployed inside buildings with a view to constructing smart, energy-efficient buildings. The novelty of the part of the project outlined in this paper lies in the fact that there does not yet exist an audio classifier for acoustic scene classification within domestic environments. This paper establishes the pipeline to build a CNN model for this purpose using the most well-suited audio feature extraction techniques and audio preprocessing. This architecture would be an essential building block for configuring sensors to operate in the most energy-efficient way by identifying human activities based on audio data generated from them.

Keywords
feature extraction, CNN, melspectrogram

1 Introduction

In this paper, we build a pipeline architecture for an audio classifier that can classify different domestic sounds. This pipeline consists of all the pieces required to train and establish a Convolution Neural Network model to read in, preprocess, and classify audio data based on specific domestic activity labels. Through this paper, we aim to address three fundamental aspects for building an audio classifier:

- What dataset would be best suited to train and evaluate a model?
- What are the features that need to be extracted from the audio files to input into our model for the most accurate results?
- Analysis of already existing CNN models for audio classification.

In future work, we will be able to build on the simplistic audio event classifier to identify activities and environments in which the event is taking place.
We begin this paper by introducing labeled audio datasets available to build and train audio classifiers. We present the details of the datasets, the number of class labels and sample size each has and the potential challenges with each dataset. This paper also discusses a simple method to read in .WAV files and remove background noise from the audio file as a part of audio preprocessing.
Once the audio data has been cleaned and processed, it is still not ready to be fed into a model. Some of the most accurate CNN classifiers are trained on image data that have specific features. To replicate the same accuracy in audio classification, it is essential to extract features from the audio files as well. Feature extraction from audio files can be implemented through different techniques. This paper explores the four most common audio feature extraction techniques – Melspectrogram, Mel-frequency Cepstral Coefficients (MFCC), Spectral Contrast, and Chromagram and elaborates on which input would work best for a CNN model.
Lastly, this paper presents two potential CNN models that can be trained using the domestic audio datasets introduced earlier. Both the CNN models were built for audio classification – the first for urban sound classification and the second for animal sound classification. The paper outlines the technical details of each model and how each might be beneficial for the overarching aim of the project – to classify sensor audio data collected from domestic environments.
We conclude this paper with the architecture of this audio classification model and an elaboration of the future work to establish the complete model for application on MAESTRO sensor audio data.

2 Datasets

Audio event classification is a challenging machine learning task. The most challenging aspect was to find a dataset with labels relevant to the domestic sound classification that we want to achieve. Some additional challenges with the datasets we were able to find included:

- Audio files muffled with background noise
- Limited number of samples per label
- Unbalanced train and test datasets
- Inconsistent samples with varying loudness and time durations
- Incorrectly labeled files

After exploring different datasets, we settled on two potential datasets that contained class labels relevant to domestic sounds and data we might be able to work with.

2.1 AudioSet

AudioSet is a large scale weakly labelled 2.3 GB audio dataset published in 2017 and is maintained by Google. It contains an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labeled 10-second sound clips drawn from YouTube videos with 527 labels. The dataset is divided into three disjoint sets: a balanced evaluation set, a balanced training set, and an unbalanced training set. The balanced evaluation and balanced training datasets have the same number of samples for each class.
Of the 632 event classes in AudioSet, 50 classes are domestic sounds relevant to our target audio classification. The 50 classes contain almost 3.3K samples for each of the training and evaluation datasets.

Many sophisticated GitHub projects have been built using Google's every expansive AudioSet data. Notable among these projects are Google's very own VGGish classifier/feature extractor Tensorflow package as well as its successor YouTube-8m. IBM improved upon Google's work by proposing an attention neural network for AudioSet that can achieve a mean average precision (mAP) of 0.360[3]. However, this dataset is difficult to use. Some of the challenges with Google's Audioset dataset are as follows:

- The dataset has multi-labelled samples so correlating class label with sample becomes complicated when training

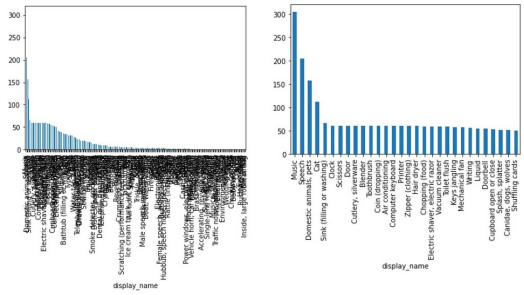

Figure 1: (L) Number of samples for all 50 classes (R) Classes with at least 50 samples each.

- The audio clips are of varying quality that require attention-based feature extractors and audio clip splitting for real-time sound event recognition

2.2 Kaggle DASEE-dataset

Kaggle's Domestic Acoustic Sounds and Events in the Environment (DASEE) is an audio database collected from a typical one-bedroom apartment at Hebrew SeniorLife Facility for dementia patients. It is an 11-class database containing excerpts of clean and noisy signals at 5-seconds duration each. The audio files are uniformly sampled at 16 kHz and yielded a weighted F1-score of 86.24'% using the baseline models using Continues Wavelet Transform Scalograms and AlexNet.The train-test split for this dataset is approximately 92-8[1].
The only challenges with this dataset are that

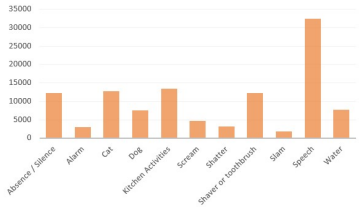

Figure 2: Sample size per class

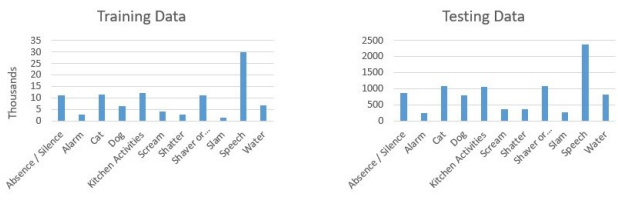

Figure 3: Train-test split

it has a limited number of classes and the classes are not as specific as the ones in the AudioSet dataset.

3 Audio Preprocessing

LibrROSA is a python package for audio analysis which can be used to read in .wav files. Through LibROSA audio files are read in as numpy arrays consisting of amplitudes sampled at some arbitrary sampling rate. The audio clip can also be visualized using librosa numpy arrays.[4]

```
audio_data, sampling_rate = librosa.load(“audio.wav”)
```

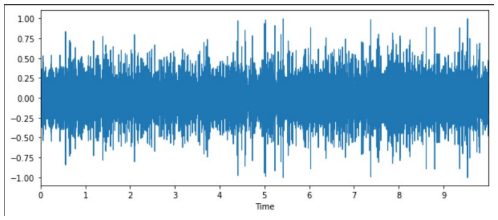

Figure 4: Visualization of the sound of a faucet running obtained from Audioset.

For datasets with noisy samples, a noise reduction algorithm outlined by Audacity, called “spectral noise gating” can be used. This can be implemented using a package called noisereduce can be used for this purpose.[7]

```
noisy_part = audio_data[start_time:end_time]

reduced_noise = nr.reduce_noise(audio_clip=audio_data,
noise_clip=noisy_part,verbose=True)
```

The challenge here is that the noisy_part was deduced from careful inspection of an audio clip which is difficult to perform on large datasets whose audio samples have been recorded using different equipment. This is why using an already clean dataset like Kaggle’s DASEE-dataset might be a simpler route for building an efficient audio classifier.

4 Feature Extraction

Feature extraction is an essential step in audio signal processing and audio event classification. In a broader sense, feature extraction involves reducing the number of resources required to describe a large dataset. Analysis of data with a large number of variables would require a large amount of memory and computation power, while running the risk of causing the classification algorithm to overfit to the training samples and generalize poorly to new samples. So feature extraction is used to construct combinations of variables to get around these issues and to describe the data with sufficient accuracy. Some Machine Learning scientists believe that properly optimized feature extraction is the key to effective model construction.

[6] In case of audio classification, some of the most notable feature extraction techniques are presented in the following sections. The Python package LibROSA introduced in the preceding section offers useful methods to implement these techniques. Using these methods from LibROSA, four types of feature extraction have been conducted on the running faucet audio clip presented earlier:

- `melspectrogram:`

  A Mel-scaled power spectrogram first takes a regular spectrogram of the sample, then converts the frequencies to mel frequencies. The features are frequency bins of the sample, which are averaged over time. This is useful for obtaining frequency information versus time.[7, 4]

- `mfcc:`

  Mel-frequency cepstral coefficients provide information about the rate of change of the frequency spectrum bands using the power cepstrum to eliminate phase. This is achieved by performing a Fourier transform on the input waveform, converting the resulting spectrum to the mel frequency band, then taking the logs of the powers of each frequency, and performing a discrete cosine transform. The resulting values are the coefficients evolving over time, the average of which are used as features.[7, 4]

- `spectral_contrast:`

  A spectral contrast is used to classify broad versus narrow frequency band signals using a short time Fourier transform. Contrast is calculated by taking the difference between the spectral peak and valley within a sub band. A high contrast translates to narrow band signals, while a low contrast translates to wide band signals. The signal is evenly split into frequency sub bands to calculate contrast over time. These then become additional features when all time values are averaged.[7, 4]

- `chorma-stft:`

  Computing a chromagram from a waveform or power spectrogram uses short time Fourier transforms. A chromagram is based on the 12 pitch classes – the Fourier transform aggregates information about each pitch within the sample as a feature at each FT window. The data can be averaged for additional features out of this. This technique is useful for obtaining pitch information irrespective of timbre.[7, 4]

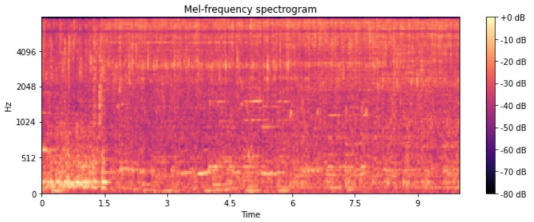

Figure 5: Melspectrogram

5 Convolution Neural Network Models

A very powerful deep learning neural network algorithm called Convolution Neural Network (CNN) has been developed through years of research on vision and how humans and animals

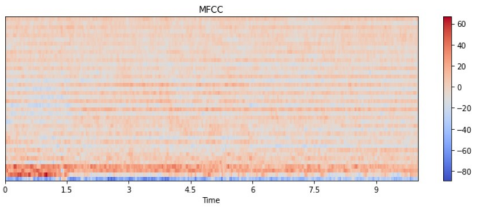

Figure 6: MFCC

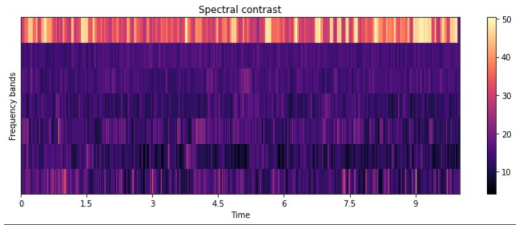

Figure 7: Spectral Contrast

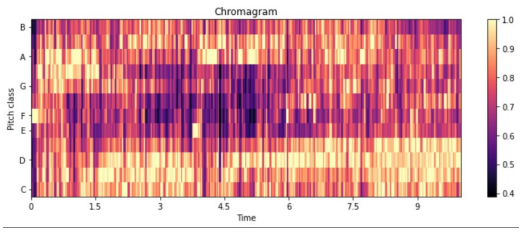

Figure 8: Chroma STFT

perceive images. CNN models can classify images with high accuracy. This is because CNNs are fully connected feed forward neural networks that are very effective in reducing the number of parameters without losing on the quality of models. Images have high dimensionality (as each pixel is considered as a feature) which suits the above described abilities of CNNs. Moreover, CNNs were primarily developed for image classification. CNNs are trained to identify the edges of objects in any image.[5]
By exploiting the CNN's ability to classify images with high accuracy, many different types of classifiers have been developed using it throughout the years. The benefit of using CNN for classification, besides its high accuracy, is that it can be generalized for a wide range of classification. Many examples of trained models for audio classification using CNN can be found out there. However, for the purposes of this project for domestic sound classification, the following two models could be used.

5.1 Model 1: Urban Sound Classification

This model has been adapted from a paper that aims to classify different urban sounds using a Convolution Neural Network. This specific model was trained and evaluated using the UrbanSound8K dataset incorporating its 10-folds cross validation. The paper uses the Sequential model from Keras and the Adam optimizer to minimize a sparse categorical cross-entropy objective for training the network. The CNN is built with four layers, each followed by batch normalization and max-pooling. After the last convolutional layer, a global pooling is added to extract embedding, which is further processed by a fully-connected layer. Lastly, a Dense layer with Softmax activation is added to output probabilities of the sound classes.[6]

```
model = keras.models.Sequential()
model.add(keras.layers.Conv2D(24, kernel_size, padding="same",
input_shape=input_shape))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation("relu"))
model.add(keras.layers.MaxPooling2D(pool_size=pool_size))
model.add(keras.layers.Conv2D(32, kernel_size, padding="same"))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation("relu"))
model.add(keras.layers.MaxPooling2D(pool_size=pool_size))
model.add(keras.layers.Conv2D(64, kernel_size, padding="same"))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation("relu"))
model.add(keras.layers.MaxPooling2D(pool_size=pool_size))
model.add(keras.layers.Conv2D(128, kernel_size, padding="same"))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation("relu"))
model.add(keras.layers.GlobalMaxPooling2D())
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(num_classes, activation="softmax"))
model.compile(optimizer=keras.optimizers.Adam(1e-4),
loss=keras.losses.SparseCategoricalCrossentropy(), metrics=["accuracy"])
```

5.2 Model 2: Animal Sound Classification

This CNN model was conceptualized to perform classification on animal sounds over multiple levels. However, to start off, the authors trained it to classify audio files into two labels – cats and dogs, trained using UrbanSound8K and Kaggle Cats Dogs and ESC datasets.
The model is made of four Dense layers that have 256, 128, 64, and n_classes number of outputs. The input of the first layer is set dynamically to 187 (the number of features pulled). Each layer is paired with the ReLu activation function, except for the final layer that uses a Softmax activation function. It compiles with the sparse_categorical_crossentropy loss function with an Adam optimizer. The learning rate of this model was discovered to be most efficient at 0.001. The model was trained in sets of 10 audio files per batch, and training it for about 60 epochs was found to be the most optimal.

```
model = Sequential()
model.add(Dense(256, activation='relu', input_dim=x_train.shape[1]))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(n_classes, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
optimizer=optimizer, metrics=['accuracy'])
```

This two-label CNN classifier was able to achieve approximately an average of 91.38% testing accuracy – 83.67% accuracy for dog labels and 100% accuracy for cat labels.
The authors then moved to generalizing the

model a bit more to build a 12-label animal sound classifier. They used the Kaggle Cats vs Dogs dataset and a subset of the animal-only data from the Kaggle Environmental Sounds dataset to train and test the model. They received a training accuracy of 100% and a test accuracy of 97.22% with a training cross entropy loss of 1% and a test cross entropy loss of 13% when the data was split 80–20 for training and testing. Using various random seeds and train-test splits, the model consistently achieved an accuracy of approximately 95%.[7]

6 Architecture & Assumptions

The audio classification architecture that would result from this machine learning model has been visualized here.

The feature extraction technique that would

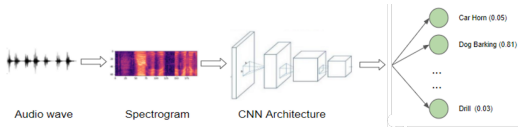

Figure 9: Model architecture including CNN model

work most effectively as the input of this model would be the Melspectrogram. Broadly spectrograms are essentially concise snapshots of the audio wave and therefore are well-suited for CNN-based architectures, such as this one, which were built for handling image data. In Mel Spectrograms the frequency is accounted for in the Mel Scale and amplitude in Decibels. As a result the difference of varying amplitudes at different points in an audio wave become even more vivid and so easier for the CNN model to perceive and derive more information from. In other words, the Mel Spectrogram captures the most essential features of any audio file and therefore is often the most suitable input for audio data in deep learning models.[2]

The contrast between figures a and b make

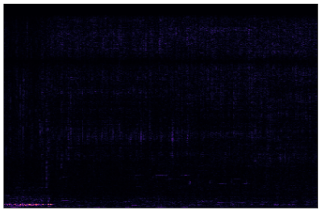

Figure 10: Regular spectrogram

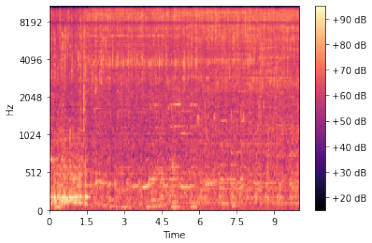

Figure 11: Mel spectrogram

clear the difference between a regular spectrogram and the mel-spectrogram. In the latter, it is easier to distinguish between higher and lower amplitudes because of using the Mel scale to express frequencies instead of Hertz.

7 Conclusion

This report presents a detailed description of the different pieces required to build an audio classifier to identify various household/domestic sounds. Once built and trained, this audio classifier will be able to successfully read in audio files collected from the MAESTRO sensors and classify the different types of sounds that sensors are able to pick up. Further work on the model would include grouping what types of sounds can be associated with what types of activities, for example - cooking, showering, exercising, etc. and what environment such activities can be performed in, such as - kitchen, bathroom, home gym, etc.

Although a lot of research is being done on audio classification and there are many pre-trained models that can accurately classify sound data, there has not yet been built a model that can specifically classify sounds based on domestic activities. On successfully training this CNN model for audio classification, it could prove to be a novel and noteworthy contribution to Machine Learning in the Internet-of-Things.

References

[1] Abigail Copiaco. *Domestic Acoustic Sounds and Events in the Environment of dementia patients*. 2021. URL: https://www.kaggle.com/abigailcopiaco/daseedataset/metadata.

[2] Ketan Doshi. "Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better". In: *Towards Data Science* (2021). URL: https://towardsdatascience.com/audio-deep-learning-made-simple-part-

2-why-mel-spectrograms-perform-better-aad889a93505.

[3] Google. *AudioSet*. 2021. URL: https://research.google.com/audioset/download.html.

[4] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *SciPy* (2015). DOI: https://zenodo.org/badge/latestdoi/6309729.

[5] Prafful Mishra. "Why are Convolutional Neural Networks good for image classification?" In: *The Medium* (2019). URL: https://medium.datadriveninvestor.com/why-are-convolutional-neural-networks-good-for-image-classification-146ec6e865e8.

[6] Aaqib Saeed. "Urban Sound Classification Part 2". In: (2020). URL: https://aqibsaeed.github.io/2016-09-24-urban-sound-classification-part-2/.

[7] Colton Saska, Mark DiValerio, and Matthew Molter. "Building an Audio Classifier". In: *The Medium* (2019). URL: https://medium.com/@anonyomous.ut.grad.student/building-an-audio-classifier-f7c4603aa989.