Ronald Pacheco

Final Report:
Urban Sound Classification

Problem Statement

Executive Summary

In this project, we are looking to create a deep learning supervised multiclass classification model to successfully classify common sounds of an urban environment. This can be useful in home security systems, where sounds can be identified and classified as burglary sounds (such as glass or door breaking, gunshot, etc), and common urban sounds (such as a car horn or ambulance siren).

Background

This project will involve 5,435 classified audio files distributed amongst 10 classes.
Principal stakeholders in this project are Ronald Pacheco and Blake Arensdorf.

Success Criteria

This model will be evaluated with the accuracy metric, and will be successful once a model can accurately classify each sound in the test set in their respective class.

Data Source in Appendix A

Data Wrangling

The data consisted of 5,435 observations. The data was separated into audio files, identified by an ID, and a csv file that contained the ID of the audio file and its class.

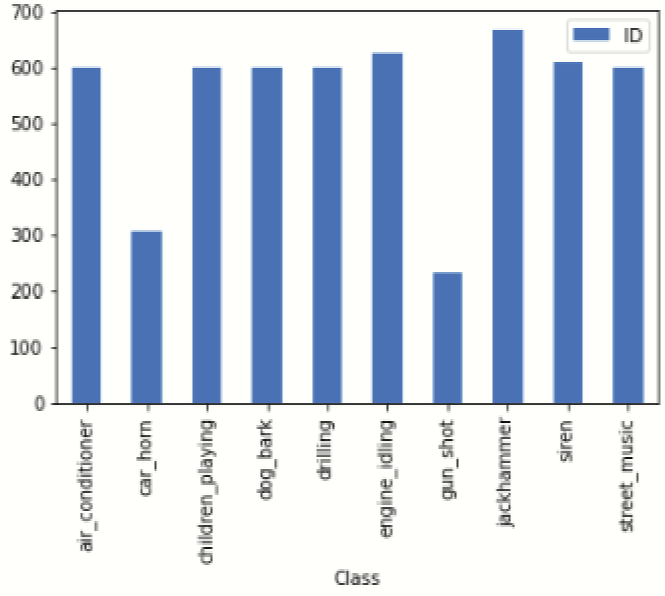
From each audio file, the nine following features were extracted:

- zcr: measures how many times the signal crosses 0
- tonnetz: Computes the tonal centroid features
- melspectrogram: Compute a Mel-scaled power spectrogram
- mfcc: Mel-frequency cepstral coefficients
- chorma_stft: Compute a chromagram from a waveform or power spectrogram
- spectral_contrast: Compute spectral contrast
- spectral_centroid: Indicates at which frequency the energy of a spectrum is centered upon
- spectral_rolloff: Represents the frequency at which high frequencies decline to 0
- spectral_bandwidth: The width of the band of light at one-half the peak maximum

Resulting in a new dataset of 5,435 rows and 185 rows.

Modeling

This dataset presented class imbalance for the classes ‘car_horn’ and ‘gun_shot’.

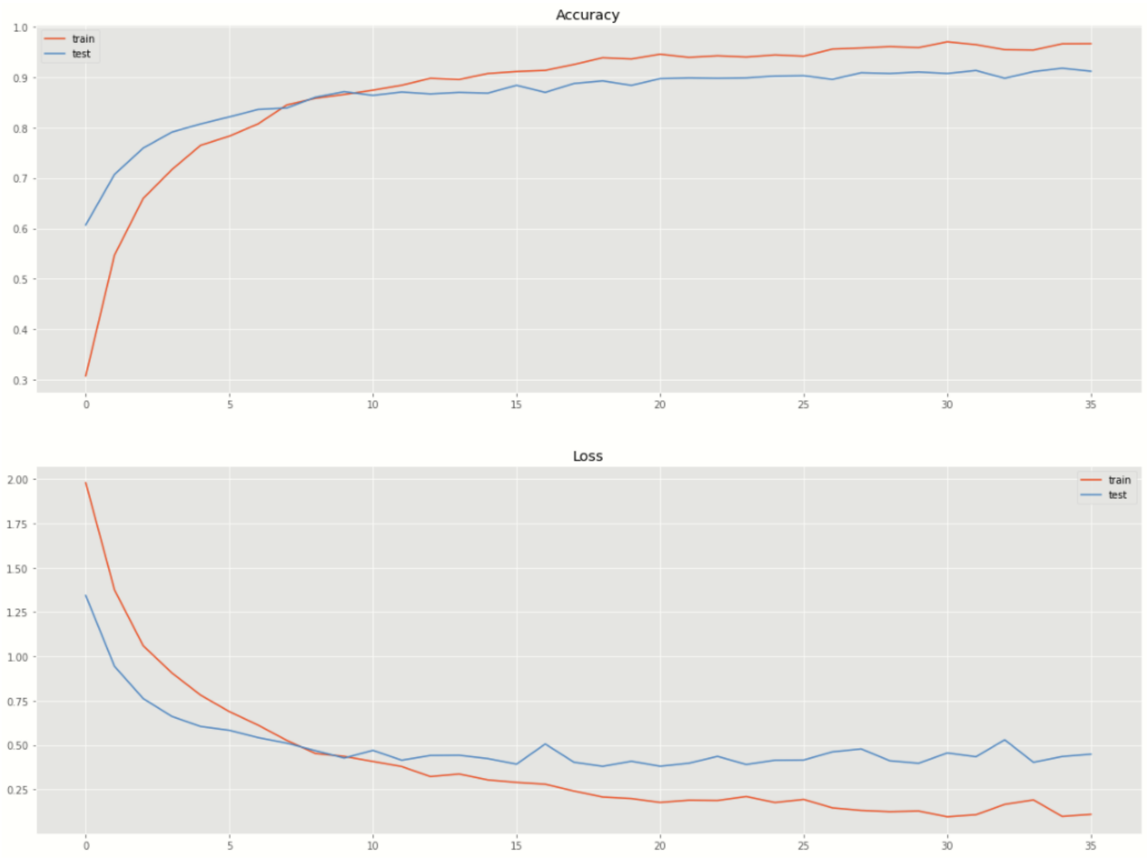


Unbalanced data

However, we decided to run the neural network with this imbalance to have a benchmark for future improvements.

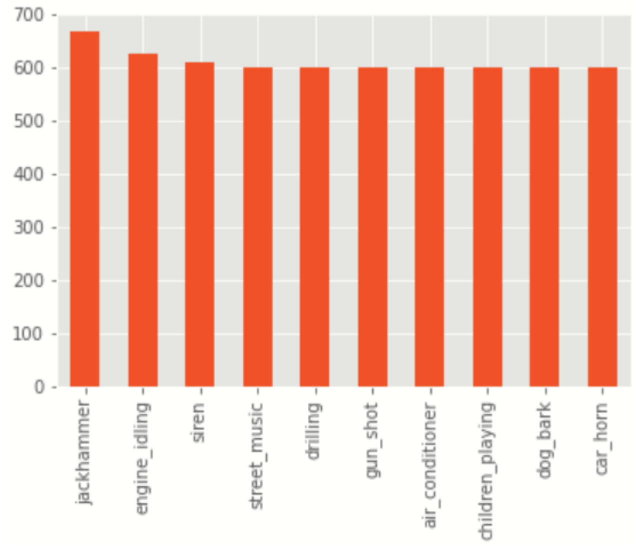
For this neural network model, we first needed to standardize our features, and create our test and train sets.

We created a Sequential model with 4 Dense layers and 2 Dropout layers (See Appendix B) which gave us an initial accuracy of 91.83%:

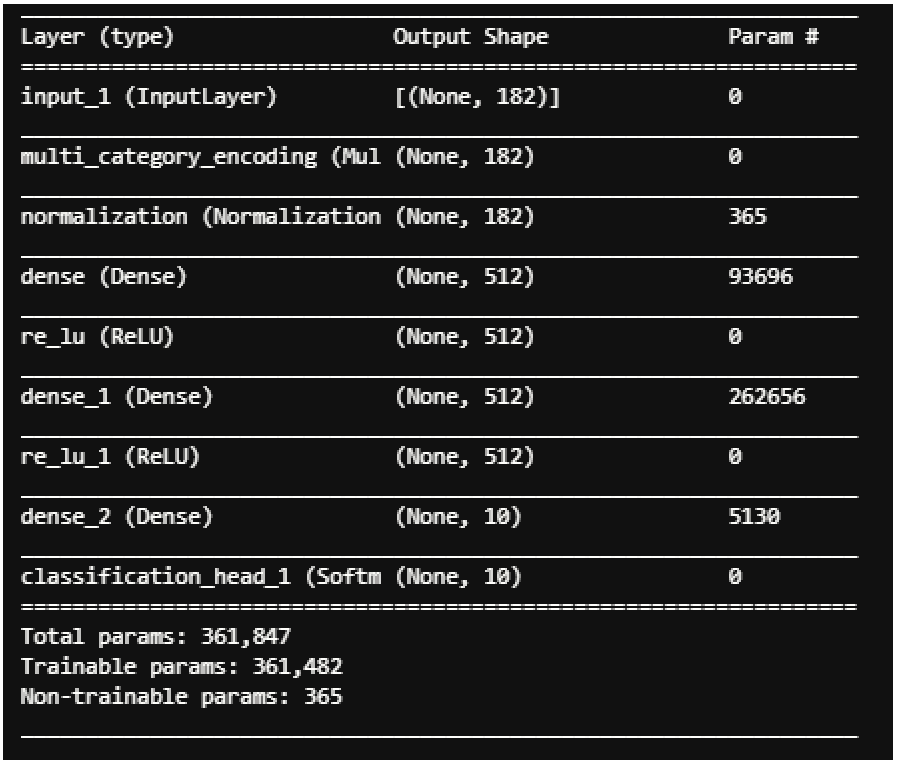


Accuracy/Loss graph for test and train data

We then proceeded to balance the data and ran it through an AutoKeras model, to which we got an accuracy of 95.28%, an improvement of over 3.4%:



Balanced data

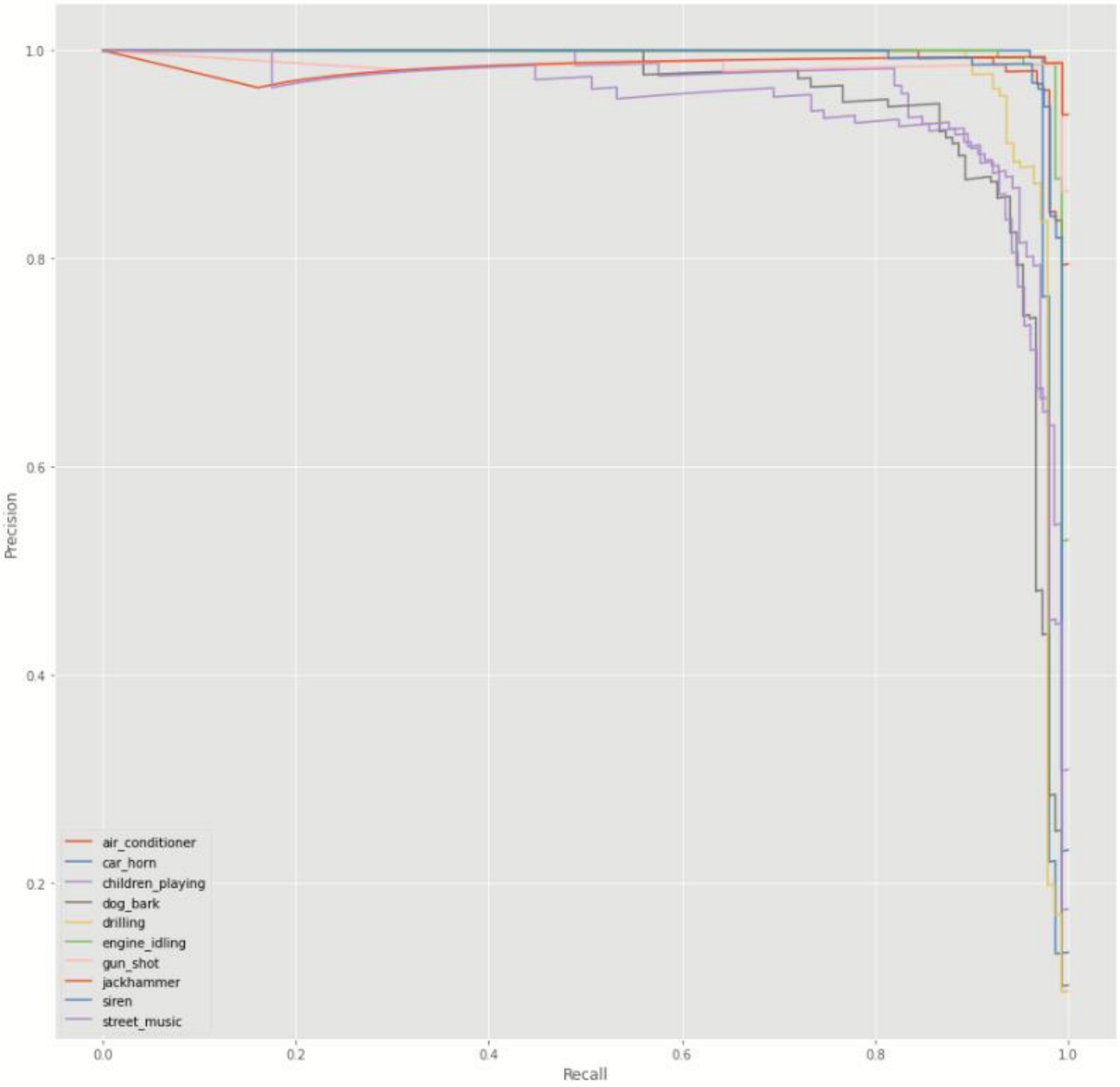


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 182)]             0
_________________________________________________________________
multi_category_encoding (Mul (None, 182)               0
_________________________________________________________________
normalization (Normalization (None, 182)               365
_________________________________________________________________
dense (Dense)                (None, 512)               93696
_________________________________________________________________
re_lu (ReLU)                 (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
re_lu_1 (ReLU)               (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
_________________________________________________________________
classification_head_1 (Softm (None, 10)                0
=================================================================
Total params: 361,847
Trainable params: 361,482
Non-trainable params: 365
_________________________________________________________________
```

AutoKeras model summary

Selecting Class Threshold

By maximizing our precision/recall scores, we are able to select different thresholds for each class(see Appendix C), which allows the model to predict, not only the most likely class, but the other possible classes, one recording might have two sounds (i.e. gun_shot and dog_barking).



Precision/Recall curves

Future Work

Filters can be applied to the sound files to try and reduce the amount of noise. This would give us a clearer audio that might be easier to correctly classify, improving the model's accuracy.

Recommendations

These classes can be grouped into different categories, which will represent the type of action required.

Based on the predicted class, a decision can be made regarding the type of activity happening in said residence. This can take into account other factors, such as time of day, crime rate in the area, type of crimes, etc.

Conclusions

This project serves as proof of concept for the business case. More audio files (such as glass breaking, door breaking, etc.) need to be gather in order to have a successful model than can be useful in this application.

Appendix

A. Data source links

https://datahack.analyticsvidhya.com/contest/practice-problem-urban-sound-classification/#ProblemStatement

B. Models Hyperparameters

```
model = Sequential()

model.add(Dense(182, input_shape=(X_train.shape[1],), activation = 'relu'))

model.add(Dense(256, activation = 'relu'))

model.add(Dropout(0.6))

model.add(Dense(128, activation = 'relu'))

model.add(Dropout(0.5))

model.add(Dense(len(y.columns), activation = 'softmax'))

model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

C. Classes' thresholds

```
'air_conditioner': 0.70582217,
'car_horn': 0.878067,
'children_playing': 0.29421836,
'dog_bark': 0.68255407, '
drilling': 0.69227,
'engine_idling': 0.8521776,
'gun_shot': 0.8978123,
'jackhammer': 0.32630515,
'siren': 0.89426386,
'street_music': 0.5781726
```