Laurence Pettitt

Tram sound classifier

Overview

Windows → Feature extraction [STFT] → CNN Classifier → Post-Processing → **Event predictions**

I chose to transform the audio signal into the time-frequency domain using the Short-Term Fourier Transform (STFT) algorithm and use the result as input to a Convolutional Neural Network. The CNN was trained on a data set of short examples of various models of tram's accelerating and decelerating, as well as background noise samples. I was able to use the CNN classifier to segment longer audio files by running a sliding window over the file, and use the Window at each hop as input. Finally, I wrote a simple algorithm based on inherent knowledge of the inputs and some assumptions of the task to reduce the series of predictions (one for each Window) into a shorter series of predictions (one for each Tram Event).

Dataset

The dataset was imbalanced so I used class weighting to reduce bias and improve generalisation. I also augmented the dataset threefold by shifting each sample a random amount less than a short time, in both directions. This helped to make the classifier less brittle to time shifting and therefore generalise better. Data augmentation also provides more training examples, which should result in an increased accuracy.

Feature Extraction

Transforming the audio signal time series into the time-frequency domain via STFT resulted in more suitable features for training. From inherent knowledge about the nature of noises from a tram, it is reasonable to assume that a well 'tuned' transformation like this would result in features which are consistent with their classes. Furthermore, the transformation gives a significant reduction in the dimensionality of the feature space. For example a 5 second audio sample with a sample rate of 22 050 would have 110 250 features, whereas the spectogram would have 27 frames and 200 frequency bins giving a total of 5 400 features. To choose the best parameters for STFT, I looked at typical values for tasks related to music (and speech) recognition and optimised for this task. The main intuition used here was that since the noises produced by a tram do not change significantly in short periods of time, I could get away with a spectogram with a low temporal resolution which allowed a high frequency resolution in a reasonably small feature space.

CNN Classifier

In designing the classifier, I was aware of the limitations due to the small size of the available dataset. I considered using a very powerful pretrained CNN such as Inception or AlexNet and only retraining the last few layers. However, these are all incredibly large networks designed to recognise a wide range of classes which have diverse features and tram recognition does not have such diversity. This intuition proved correct as in the end I showed that is possible for a smaller custom network to accurately discriminate the classes.

The final layer of the CNN classifier is a softmax layer with nine units; two units for each of the four tram models, either accelerating or braking, which make eight units for the positive event classes, plus one unit for the negative event class (i.e. no tram present). These units show the relative confidence of the classifier in the presence of the unit's corresponding class.

Post-Processing

In the segmentation of a long audio sample task, I had to reduce the nine probabilites produced by the CNN classifier from each window step of the sliding window into a shorter series of single class events. To do this I firstly took a rolling average in the time domain to smooth out volatily within each class. Then I took the maximum class in each window, removed all sequential duplicates and Class-8 (background noise), and finally removed all predictions which have another prediction nearby with a greater confidence. In this last step is where the earlier rolling average becomes particularly important; the rolling average reduces the favouribility towards predictions which have very high peaks even if this confidence lasts a very short time, and instead demands that a prediction should have a high probability and last a reasonably long time, in order for that prediction to be considered strong.

Summary

It is difficult to predict how the whole architecture would perform in the wild without analysis on more test files, however I believe this approach has the capability of performing quite well with a bit of adjustment. Mainly due to the appropriateness of the chosen STFT features and the proven power (in object recognition) of CNNs to extract high level features for classification. With more test files and more time, it could also be possible to replace the Post-Processing layer with a supervised sequence to sequence classifier such as a Hidden Markov Model, Recurrent Neural Network or Connectionist Temporal Classifier.