EECE5644 Machine Learning Final Project

Urban Sound Classification

Zachary Neveu, Tianyi Zhou, Christian Grenier

Northeastern University

04-16-2019

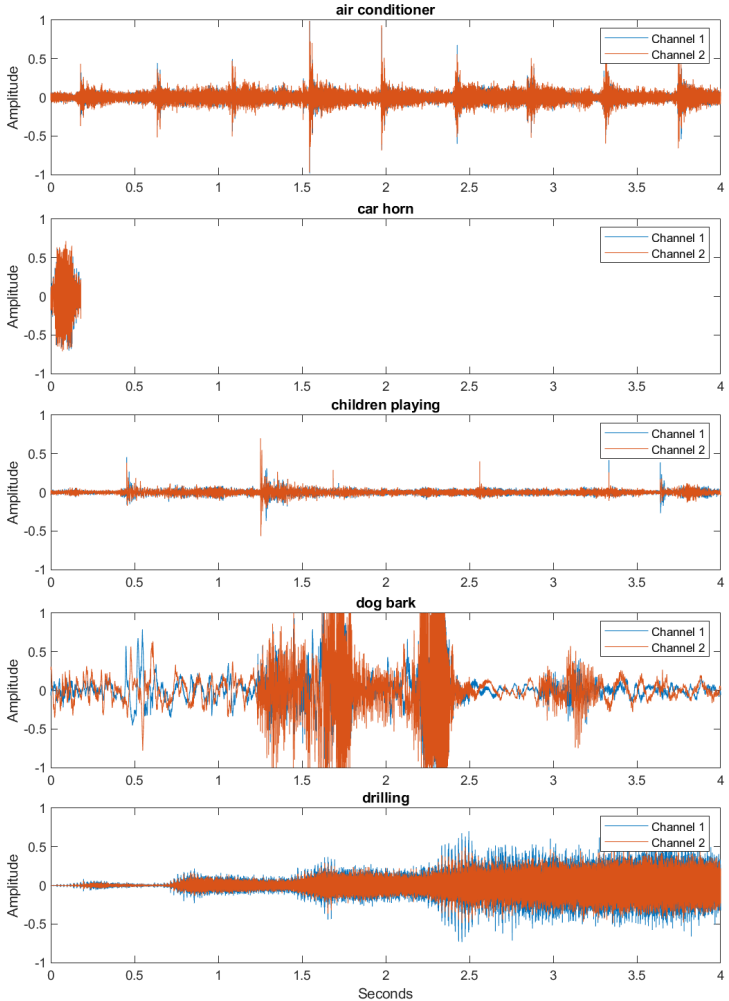
Task: Classify Urban Sounds into 10 Categories

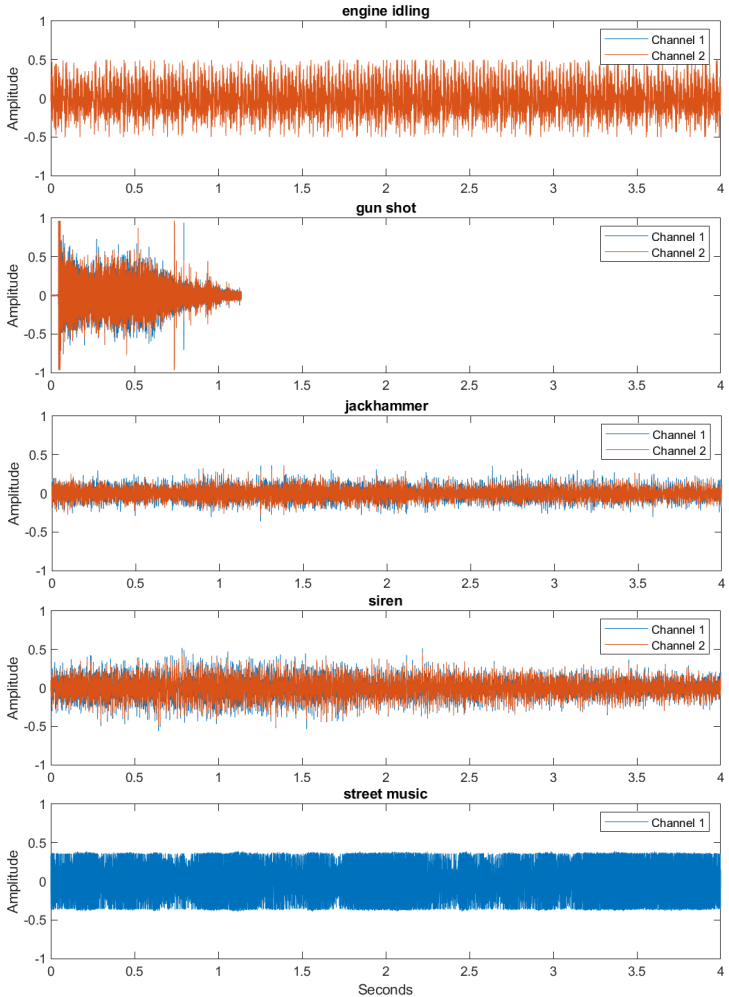
Dataset

- UrbanSound8k from Kaggle
- **5435** Audio .wav Files with 10 classes (shown below)
- Duration: **0.05 ~ 4s**
- Sample Rates: **44.1 kHz, 48 kHz, 96 kHz**

Potential Uses

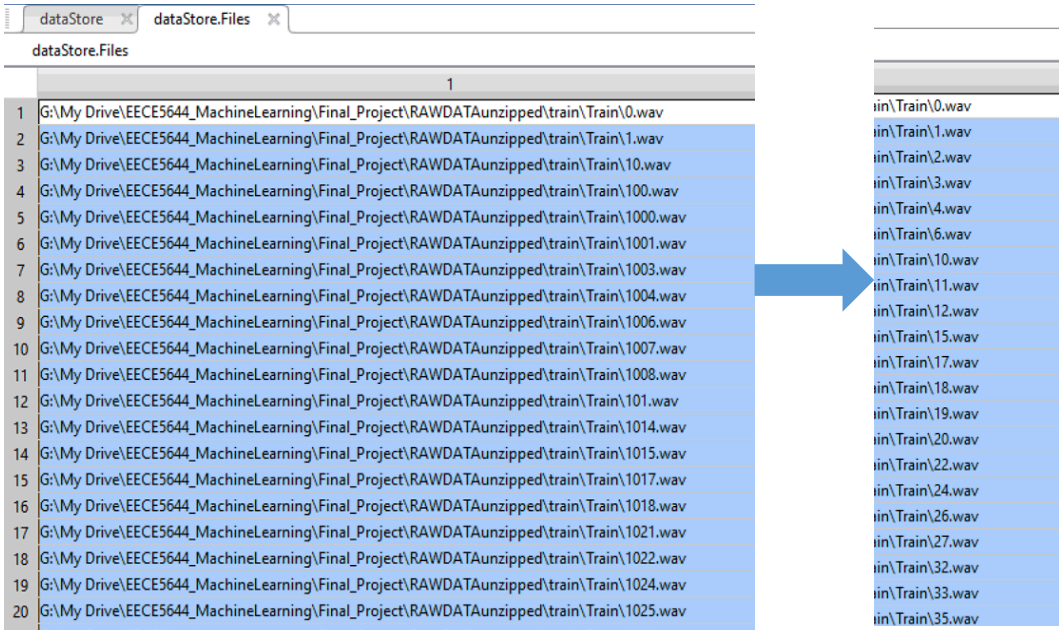
- Gun violence detection
- Predicting travel times based on construction
- Predicting energy usage based on air conditioner levels
- Acoustic situational awareness for machines





Pre-Processing

- UrbanSound8k Dataset from Kaggle is split:
  - Training Set: **5435** labeled audio files → our data set
  - Test Set: **3297** un-labled audio files → only for competition on Kaggle
- The `audioDatastore` object is used to manage the collection of audio files, and the `audioread` function extracts each file from the object.
- Files are sorted in numerical order instead of alphabetical,… as shown below. This way the labels match

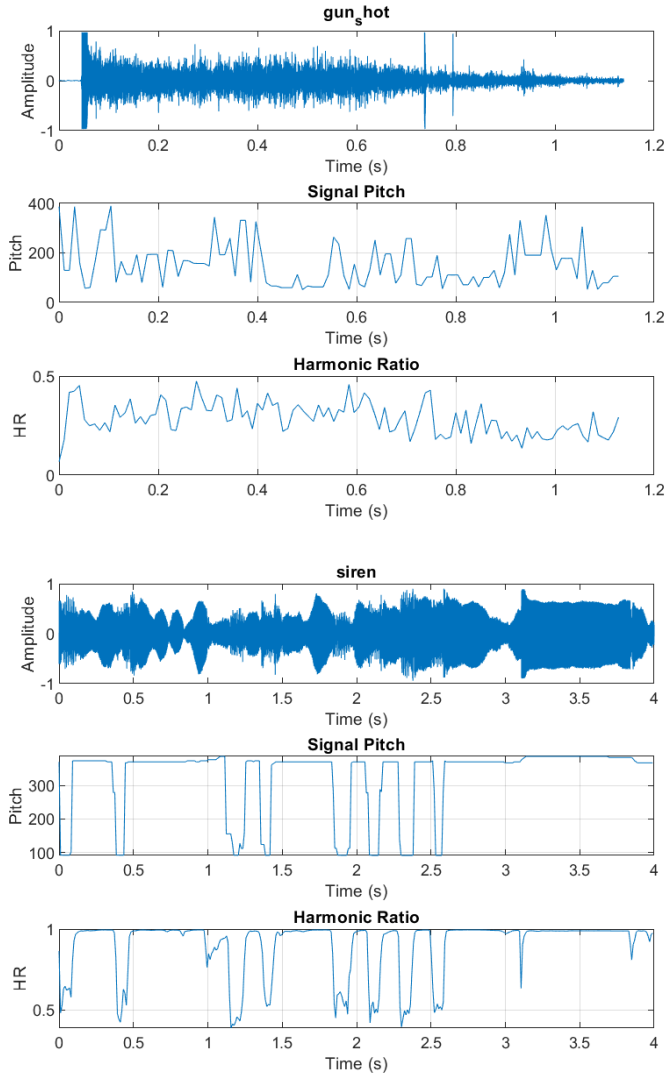


- Second channel of audio data is ignored for simplification and uniformity

Feature Extraction

- **Basic Info (# of features: 2)**
  - Sample Rate
  - Duration
- **Loudness (2)**
  - Integrated Loudness
  - Loudness Range
    - Max loudness – Min loudness
- **Pitch (2)**
  - Average Pitch
  - Average Magnitude of Derivative of Pitch
- **Harmonic Ratio (3)**
  - Average HR
  - Variance of HR
  - Average Magnitude Derivative of HR
- **Mel Frequency Cepstral Coefficients (98)**
  - 7 time windows for each audio signal
  - 1 log energy value and 13 Mel frequency cepstral coefficients (MFCC) for each time window
  - See more details about MFCC in next slide

Example plots of Pitch and HR for gunshot and siren audio.

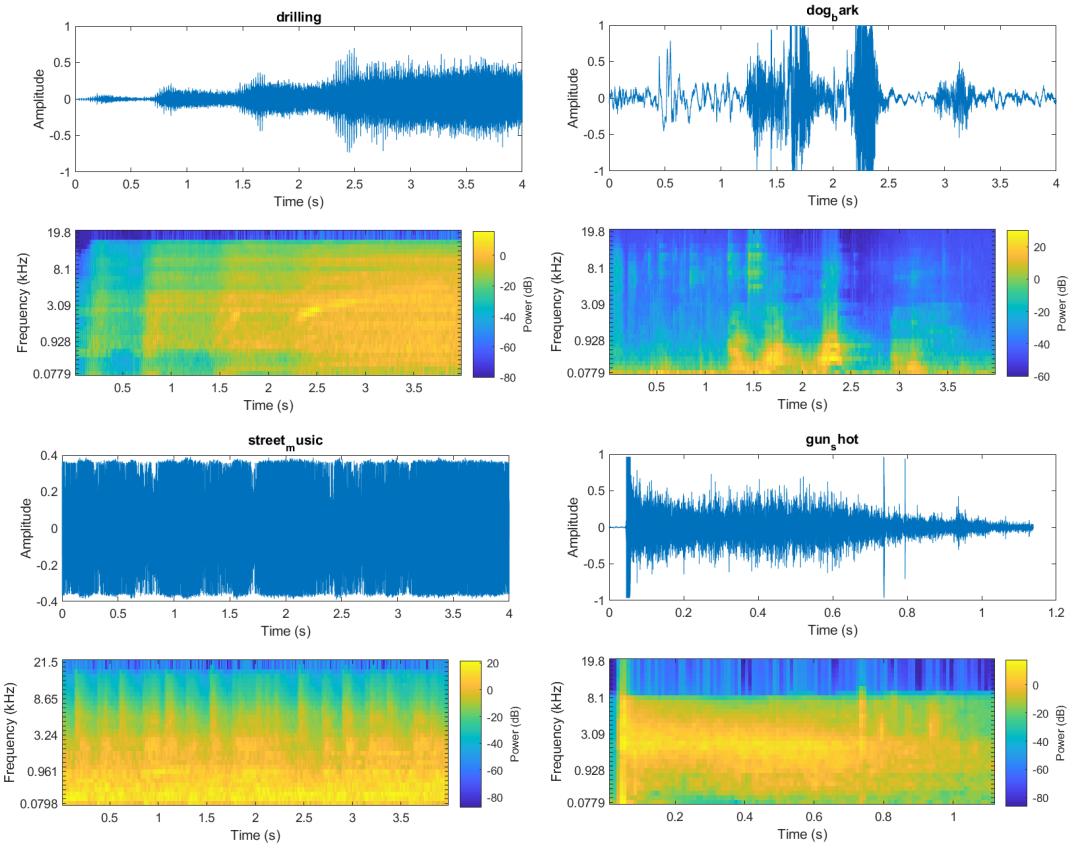


NOTE:

1. Audio Toolbox in MATLAB provides: `integratedLoudness`, `pitch`, `harmonicRatio`, `mfcc` functions to extract features from audios.
2. We padded the short audio to 4s, and added small SNR gaussian white noise so that the mfcc function doesn't return –inf when taking the log of the DFT. Then we manually set the window length and overlap length so that the number of window is 7.

Mel Frequency Cepstral Coefficients

Mel Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of the audio. It is derived by dividing the signal into time windows, taking Fourier Transform within each window, then mapping the powers of the spectrum onto the mel scale. MFC for selected audio samples are shown below.



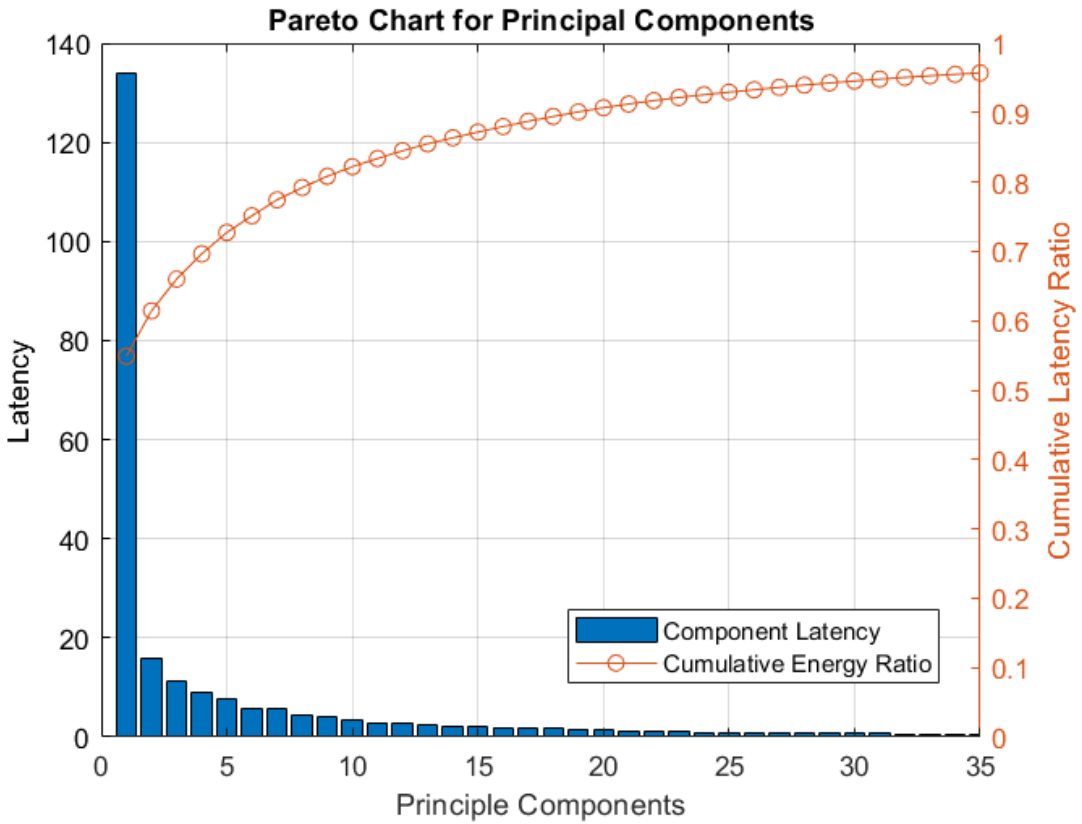
- Mel Frequency Cepstral Coefficients (MFCCs) are coefficients that collectively make up an MFC.
- As shown, **the spectrograms accurately capture time and frequency data for each audio class.**

To classify an audio sample, we could either:

1. Take MFCCs as features and perform **Supervised Classification**
2. Take MFCs to do **Image Classification using Convolutional Neural Network**

Dimensionality Reduction - PCA

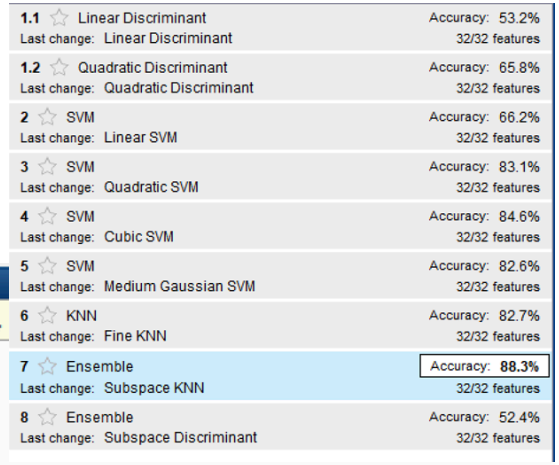
- A customized zscore function is used to deal with missing data in feature array
- Principal Component Analysis is performed with a threshold of 95% cumulative latency ratio
  - 107 → 32 features

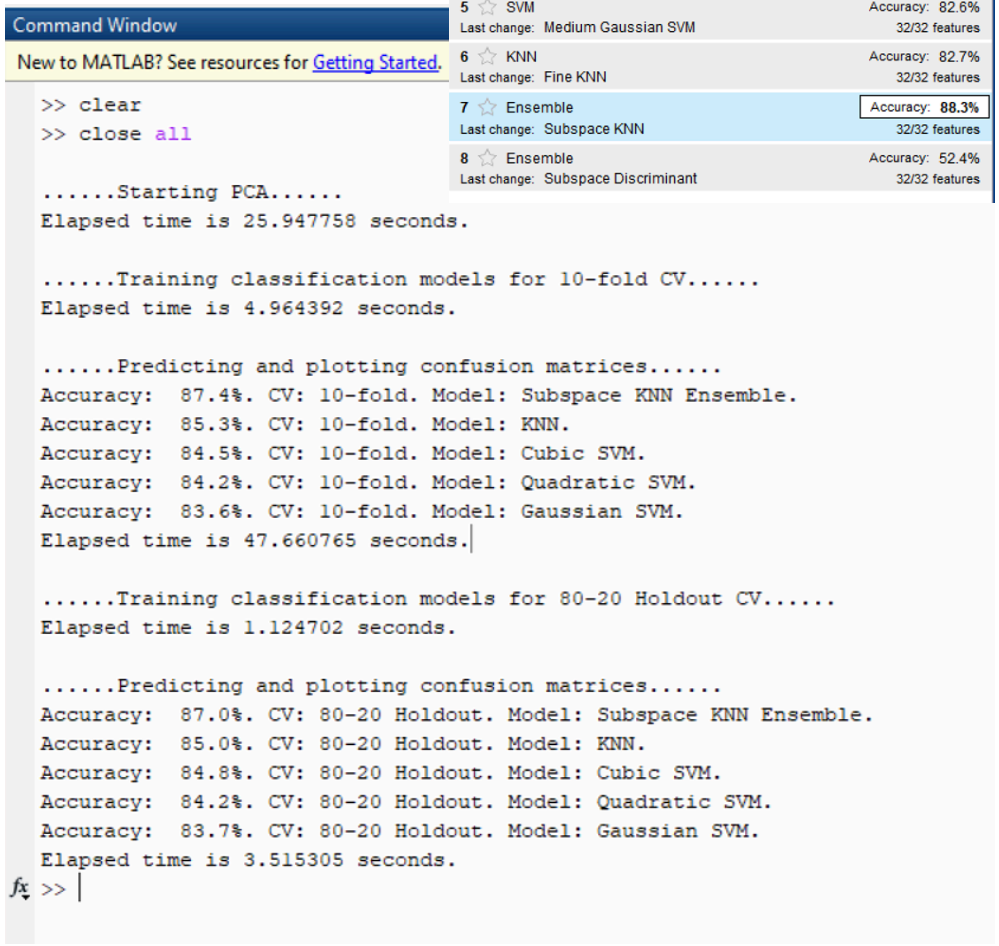


Supervised Classification

Model Selection

- The Classification Learner in MATLAB allows us to explore various classifiers and search for the best classification model type.
- We select 5 models with the best performances, manually tune the parameters for the classification model. These models are:
  - Ensemble Subspace KNN
  - K-Nearest Neighbor
  - Cubic Kernel SVM
  - Quadratic Kernel SVM
  - Gaussian Kernel SVM

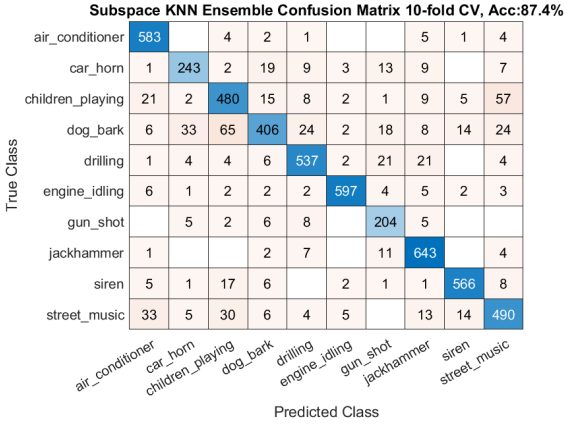


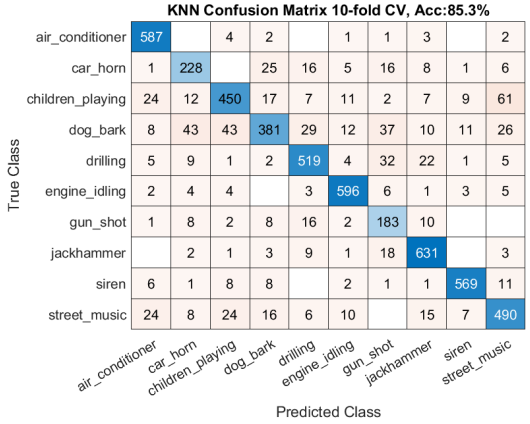


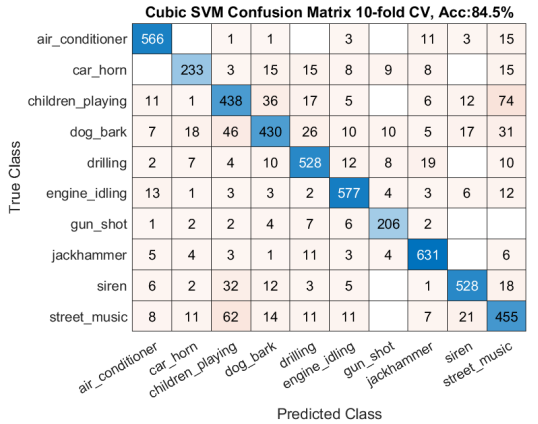
Supervised Classification

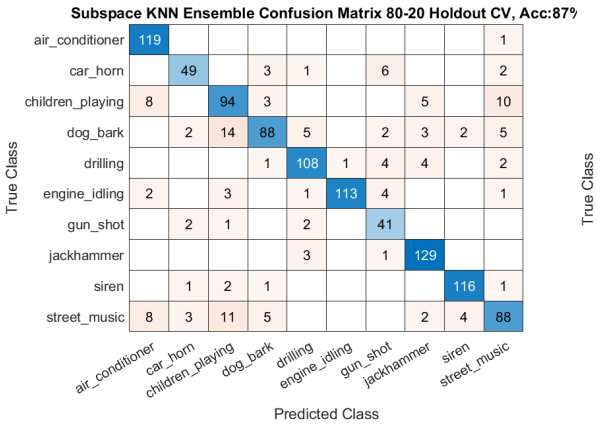
Cross Validation and Model Evaluation

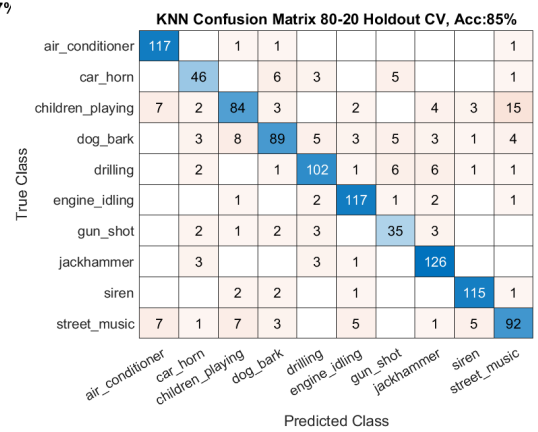
- Both K-fold (with 10 folds) and Holdout (80-20 split) methods of CV are used
- Parallel computing improves the speed of training and validation for 80-20 holdout case.
- Selected confusion matrices are shown below.

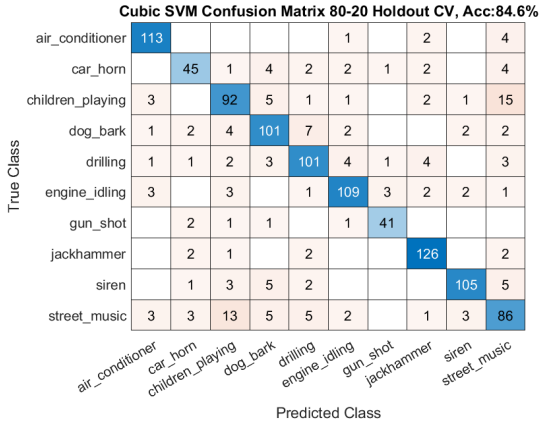






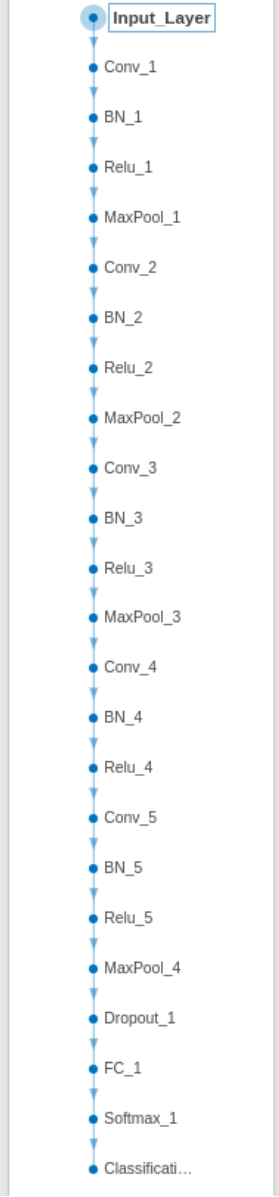






Neural Network Architecture

- Convolutional
- Repeated Blocks of:
  - 3x3 Convolution
  - Batch Normalization
  - RELU
  - Max Pool
- Dropout layer to avoid overfitting
- Softmax layer to create one-class predictions
- 24 Layers Deep

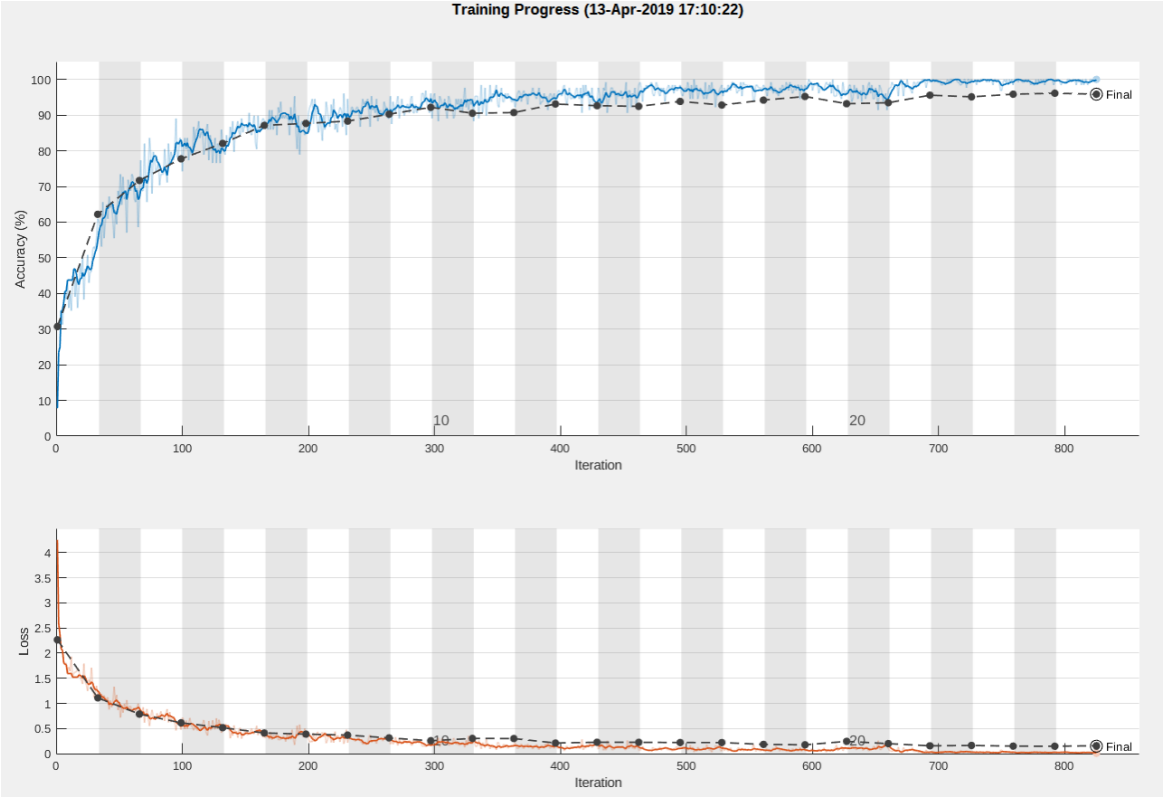


ANALYSIS RESULT

	Name	Type	Activations
1	Input_Layer 40x396x1 images with 'zerocenter' normalization	Image Input	40×396×1
2	Conv_1 12 3x3x1 convolutions with stride [1 1] and padding 'same'	Convolution	40×396×12
3	BN_1 Batch normalization with 12 channels	Batch Normalization	40×396×12
4	Relu_1 ReLU	ReLU	40×396×12
5	MaxPool_1 3x3 max pooling with stride [2 2] and padding 'same'	Max Pooling	20×198×12
6	Conv_2 24 3x3x12 convolutions with stride [1 1] and padding 'same'	Convolution	20×198×24
7	BN_2 Batch normalization with 24 channels	Batch Normalization	20×198×24
8	Relu_2 ReLU	ReLU	20×198×24
9	MaxPool_2 3x3 max pooling with stride [2 2] and padding 'same'	Max Pooling	10×99×24
10	Conv_3 48 3x3x24 convolutions with stride [1 1] and padding 'same'	Convolution	10×99×48
11	BN_3 Batch normalization with 48 channels	Batch Normalization	10×99×48
12	Relu_3 ReLU	ReLU	10×99×48
13	MaxPool_3 3x3 max pooling with stride [2 2] and padding 'same'	Max Pooling	5×50×48
14	Conv_4 48 3x3x48 convolutions with stride [1 1] and padding 'same'	Convolution	5×50×48
15	BN_4 Batch normalization with 48 channels	Batch Normalization	5×50×48
16	Relu_4 ReLU	ReLU	5×50×48
17	Conv_5 48 3x3x48 convolutions with stride [1 1] and padding 'same'	Convolution	5×50×48
18	BN_5 Batch normalization with 48 channels	Batch Normalization	5×50×48
19	Relu_5 ReLU	ReLU	5×50×48
20	MaxPool_4 1x50 max pooling with stride [1 1] and padding [0 0 0 0]	Max Pooling	5×1×48
21	Dropout_1 20% dropout	Dropout	5×1×48
22	FC_1 10 fully connected layer	Fully Connected	1×1×10
23	Softmax_1 softmax	Softmax	1×1×10
24	Classification crossentropyex with 'air_conditioner' and 9 other classes	Classification Output	-

Neural Network Training

- 80%/20% training/validation split in data
- ADAM Optimizer (see https://arxiv.org/abs/1412.6980)
  - Faster than SGD with momentum
- 25 Epochs to converge
- Maximum Validation Accuracy: 95.86%



Results and Conclusion

- The best classification model is Ensemble Subspace KNN, with 10-fold CV accuracy of **~87%.**
- To improve results:
  - Include second channel of audio files
  - Increase frequency and time resolution for MFCCs
  - Better pre-processing that includes pitch values for short duration files

- A Convolutional Neural Network achieves a maximum accuracy of **95.86%** on a 20% hold out validation set