MSIA 432 Deep Learning
Spring 2020

Project Midterm Form

Due 5/11/2020

Project title	**Urban Sound Classification**
Team members	1. Shreyashi Ganguly 2. Aakanksha Sah 3. Aditya Gudal 4. Srividya Ganapathi 5. Bhavya Kaushik 6. Aditya Tyagi 7. Manish Kumar

Problem statement

A quick recap of the problem statement

The objective of this project is to use deep learning to classify urban sounds from the UrbanSound8k dataset. We have 10 classes of urban sounds and we predict probabilities for a given audio clip belonging to each of the classes. The model performance evaluation metric is classification accuracy.

Specific action items in the Minimum Viable Product (MVP)

1. Preprocess the audio files and normalize their audio properties for consistency
2. Extract MFCC (Mel-Frequency Cepstral Coefficients) from the preprocessed audio data
3. Explore different simple approaches like Logistic regression, XGBoost, Random Forest, KNN, and Neural Network and train them over the dataset
4. Train an initial CNN model over MFCC spectrogram representations of the audio data
5. Perform 10-fold CV on these simple models to compare their mean CV accuracies
6. Deliver a trained model that can classify sounds into one of the 10 classes with mean CV accuracy of more than 50%
7. Identify the pros and cons of simple models and pave the way for more complex models and prepare a plan to iterate over more complicated CNN models

Dataset

Dataset description

The UrbanSound8K dataset consists of 8732 labeled sound excerpts (=<4s) of urban sounds. There are 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. Along with the individual files in .wav format corresponding to audio clips, a metadata(.csv) file is provided which tags each file to its class labels. The total size of the data in a compressed format is 5.7GB.

Example:

The Urbansound8K dataset is compiled from freesound.org. Follow this link for an example of a car horn sound from the dataset: https://freesound.org/people/ceberation/sounds/235506/ .

Did we have any difficulty obtaining the dataset?

We did not face any difficulty in obtaining the data. The dataset is open source and can be downloaded from here: urbansounddataset.weebly.com.

The UrbanSound8K dataset is offered free of charge for non-commercial use only under the terms of the Creative Commons Attribution Noncommercial License. There are no restrictions on the use of this dataset for academic purposes.

Were there any unexpected issues in the data? Too small? Too noisy?
The dataset is adequate for machine learning algorithms. It is neither small nor noisy. One unexpected issue that arose when exploring the dataset was that the audio file properties like the sampling rate, the number of channels, and the bit depth weren't uniform over all the observations. This difference in the digital quality of the audio files can obscure the feature extraction process and lead to inconsistent features. To deal with this, we had to normalize these properties over the entire dataset.

If the data doesn't work out, is there a backup dataset we can use?
Although we know now that the dataset works fine, there are several alternative datasets that can be used in place of UrbanSound8K for audio classification tasks, like UrbanSound (non-8K version), AudioSet, Bird Audio Detection challenge, etc.

Technical Approach/Initial Results

Description of the technical approach

1. We explored and visualized the data using the *Librosa* package that allowed us to load audio in our notebook as a *NumPy* array for analysis and manipulation.
2. We used *Matplotlib* to display the waveform as a *waveplot*. e.g. Dog Bark

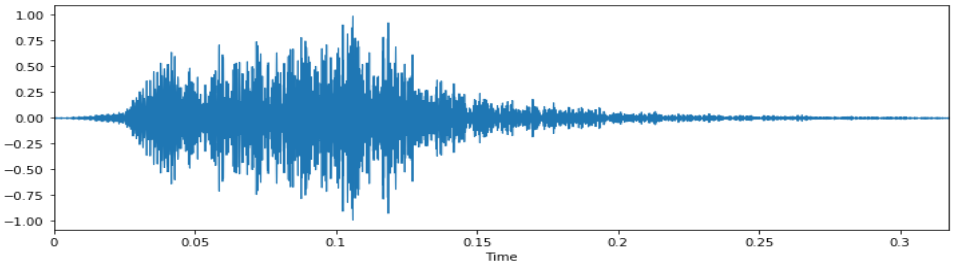

Figure 1: Example of a waveform

3. We iterated through each of the audio sample files and extracted the number of audio channels, sample rate, and bit-depth. The number of audio channels was converted to 1 by merging signals, the sample rate was converted to 22.05 kHz and the bit depth was normalized so that the values were in the range of -1 to 1.
4. We extracted the Mel-Frequency Cepstral Coefficients (MFCC) from the audio samples. These were 40 coefficients over 173 frames in each audio file.
5. The mean of each coefficient over the 173 frames is currently being considered as input. Thus, we have 40 features for each of the 8732 audio files for training.
6. We performed a comparative analysis on different simple predictive models along with a CNN model (trained over MFCC spectrograms) using the mean 10 fold CV accuracy.

Simple approaches explored for MVP
We trained the following models under 10-fold CV (the folds were pre-defined in the metadata) - Logistic Regression, XGBoost, Random Forest, KNN, and Neural Network. We also trained a demo Convolutional Neural Network over MFCC spectrogram images generated from the audio data.

The following table summarizes the results of the 10 fold cross-validation:

S.No.	Model	Parameters	Mean CV accuracy
1	Logistic Regression	penalty='l1', solver= SAGA , tolerance=0.1	45.83%
2	Random Forest	split criterion = gini criterion, sqrt(p) features at each split (p = # of features), min_samples_split = 2, n_estimators = 100, min_impurity_decrease = 0.0 {Trees are grown deep}	54.01%
3	XGBoost	base_score: 0.5, booster:gbtree,gamma: 0, importance_type:gain,learning_rate: 0.1, max_delta_step: 0,max_depth: 3, min_child_weight: 1	46.69%
4	KNN	nearest_neighbors=19, weights='distance', algorithm='ball_tree', metric='minkowski'	43.47%
5	Neural Network	Optimizer = adam, batch size = 32, learning rate = 0.0001, epochs = 250 2 hidden layers each having 256 nodes with relu activation and 0.5 dropout, softmax activation on the outer layer	54.43%
6	Demo Deep Neural Network (CNN)	Architecture: Conv2d->Maxpooling2d->Conv2d->Maxpooling2d->Conv2d->Maxpooling2d->Dense->Dense(softmax) Optimizer = adam, batch size = 256, num epochs = 20, activations = 'relu', 'softmax' (final layer), loss = 'categorical_crossentropy'	60.33%
7	Human	Explained below	Accuracy: ~98%

We tuned the hyper parameters of all the models except the CNN to get an optimal fit. Still the models failed to perform well in the audio classification task.
Limitations of simple approaches considered:

- They are unable to capture the relationships between the complex temporal patterns and the sound categories. These have been captured in the MFCC spectrograms which are images.
- They easily overfit if given a higher number of parameters.

We then constructed a trivial CNN architecture to just provide a demo of a CNN's potential to deal with this particular task of audio classification. CNN by far shows the best results.
Advantages of CNN over the other models:

- Inherent capability of dealing with images which turns out very helpful in this scenario as MFCC spectrograms are very efficient representations of audio. CNN exploits the virtue of local connectivity in these images and connects neurons to smaller local regions of the input volume.
- Parameter sharing scheme is used in convolutional layers to control the number of parameters. This greatly reduces the risk of overfitting unlike densely connected neural networks.

What is the human accuracy rate on a small sample of the data (around 100 examples)? What was easy to identify, and what was hard? What features did we look for? Did all our team members agree, or did one domain expert provide extra input?

To calculate this, we created a stratified sample of 100 records and listened to them individually. We already knew what the 10 different categories are and had heard a few labelled examples before performing the test. The human correct classification (accuracy) rate based on that sample was ~98%. The reason for this high accuracy are:

- The dataset consists of audio files of common/familiar sounds. A human with a decent hearing ability can easily identify the corresponding class of sound
- The audio files do not have any significant background noise
- The sample size of 100 is too small to quantify human error properly

The classes 'jackhammer' and 'engine_idling' created confusion and were the primary reason for misclassifications by humans. The classes 'gunshot', 'air_conditioner' and 'drilling' were also problematic for humans. The other classes like 'dog bark', 'car horn' etc. were easier to identify. Considering that this was a small sample, in general the human accuracy rate can be estimated to be in the range 95% to 98%. The features humans look at while performing the classification task are very intrinsic in the human auditory perception system and all the team members were in agreement regarding those. The need for a domain expert for this task is not necessary.

Visualizing Intermediate Layer Activations of the CNN model *(using Keras)*

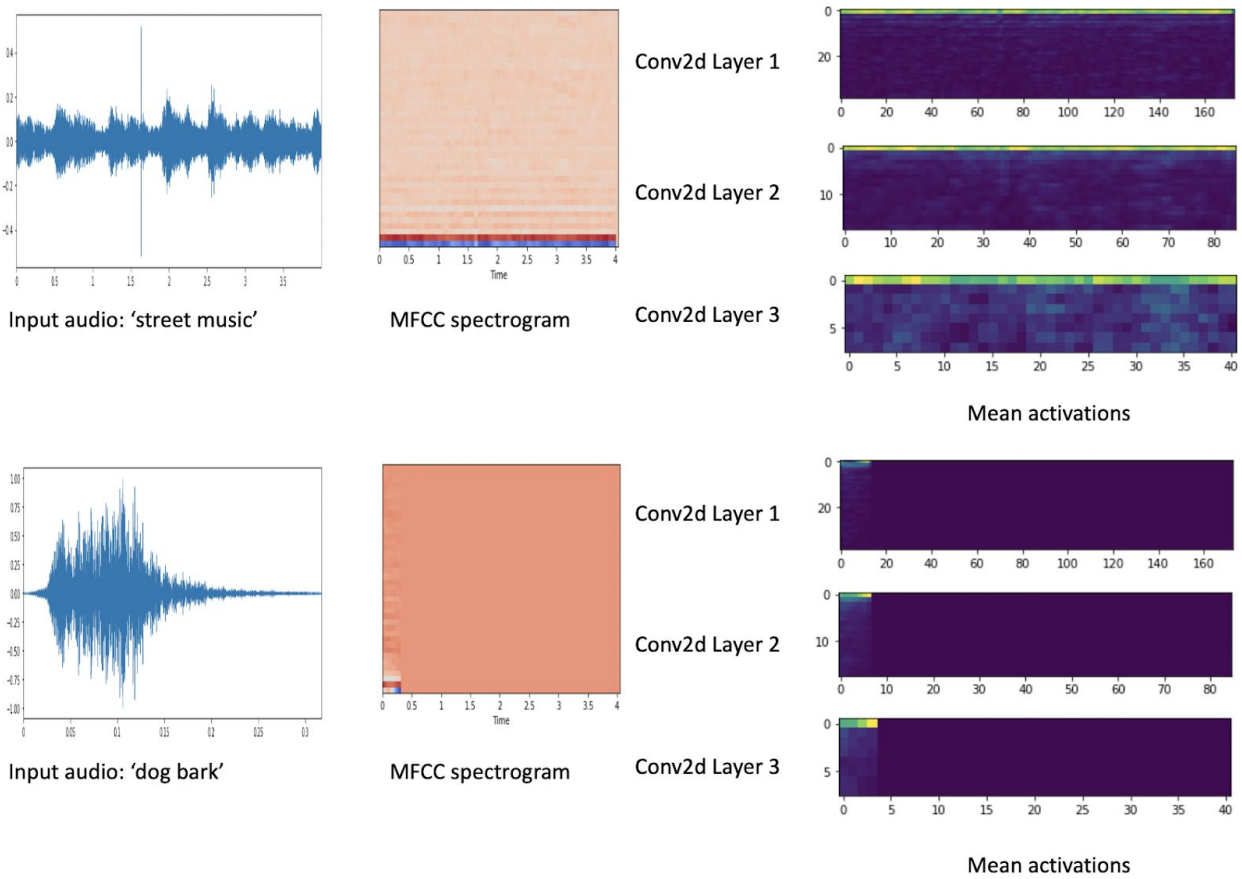

Figure 2: Sample from the dataset (Left), MFCC spectrogram(Center), Mean of the activations(Right)

The above visualization shows the images of the mean activations (mean over the number of filters) from three of the intermediate convolutional layers of the CNN. The first example is of the class 'street music' and has an MFCC that gives parallel horizontal color bands. The activations show similar bands. The second example was of a 'dog bark' and the MFCC only showed a stimulation in the first 0.3s of the 4s snippet, i.e. a few columns of pixels on the left. The visualizations show that the activations are focusing on the right area. Hence we feel that the model is focusing on the right areas.
The model needs to focus more on the foreground for categories like 'car horn', 'dog bark' etc. while it needs to focus more on the background for classes like 'air conditioner', 'engine idling' etc. Foreground and background are very difficult to identify while visually examining the MFCC spectrograms. Hence we won't be providing insights on how the model is dealing with foreground and background at this point.

MVP shortcomings and next steps
As part of the MVP, we tried various linear and tree based classification algorithms. Depending on their performance, it seems the data needs to be modeled with more complicated functions. Here are a few of the major shortcomings of the simple models explored for MVP:

- The models didn't capture the complex temporal patterns of different sound categories.
- They can't work with images, which means MFCC spectrograms can't be utilized.
- They overfit very easily.

There is a lot of literature on how to classify sounds using image (spectrogram) representations. Hence we tried a deep CNN model on MFCC spectrogram images generated from the audio data. CNN performed better than other models.
This encourages us to next improve upon the architecture of our CNN. We can add/remove layers, test different activation functions and optimizers, put adaptive learning rates. We can also implement regularization techniques like dropout, do data augmentation of audio files, tweak background noise.

Project Risks

A few possible project risks that we encountered are listed below:

- There is a risk of overfitting for densely connected neural network layers, if we include all the 40*173 = 6920 features.
- If we take mean over the frames we lose the temporal information completely.
- Training is taking very long, this is aggravated further if we do a 10 fold CV.
- The classes are not uniformly distributed, making training for classification of classes like 'car horn' and 'gunshot' difficult.
- Lack of knowledge about what MFCC spectrograms actually mean can result in problems while inspecting the intermediate layer activations of the CNN for diagnosis of signs of poor training.
- There can be background noise in a few classes of the dataset, which can hamper training and can be difficult to diagnose.

We plan to do the following to mitigate these risks:

- Implement regularization techniques like dropout in CNN.
- We will use GPUs to train the CNNs when we have a lot of parameters and training is taking long.
- We will explore data augmentation and oversampling of minority classes.
- We will implement version control using Github to coordinate better in a large team.
- We will try to get a good background about MFCCs and do feature engineering if required.
- We will play around with techniques like noise injection to test the robustness of the model.