Final Report of Team 3

Yinghao Ma **Reese Grimsley** **James Merrill** **Shuqing Li**

Abstract

In this report, we describe our MLSP course project based on the DCASE 2020 Challenge 5: *Urban Sound Tagging with Spatiotemporal Context*. This challenge uses machine learning and signal processing techniques to detect noise sources in New York City, such as shouting or car alarms, as part of the SONYC dataset. We perform feature extraction on the provided audio and use time and location annotations to query open data sources like OpenNYC to generate spatiotemporal context (STC) features. We use those features to train traditional ML classifiers, evaluating our performance with the area under precision-recall curve (AUPRC). Our best results came from a Random Forest classifier, with an AUPRC of 0.402 and 0.395 when training with and without STC features, suggesting our use of STC is only marginally useful for this dataset.

1 Introduction

For this project, we approach DCASE 2020 Challenge 5: *Urban Sound Tagging with Spatiotemporal Context (STC)* [1]. We use traditional machine learning techniques to detect a variety of bothersome urban noises, such as jackhammers, vehicle signals, barking dogs, human voices, etc. using audio collected in New York City (NYC). In addition to each audio sample, we are provided with spatiotemporal context, such as date, location, etc. to improve the model. We combine STC with audio to solve this classification problem, testing various classifier types and ensemble topologies to build a model that effectively uses external information to reduce the search space and boost performance. This project shows that multiple types of information can assist prediction of an audio clip's sound taxonomy.

2 Related Work

The formal challenge has ended, giving us the opportunity to view other challengers' approaches to solving this problem — convolutional neural nets are vastly popular within this challenge [2][3] and several similar ones. In our project, we will not use these techniques, as they are not central topics to the class. Instead, we will focus on other aspects of signal processing, feature extraction, and classification. However, those submissions still include useful techniques regarding preprocessing, dataset augmentation, and STC inclusion.

Common methods on audio feature extraction are Mel-frequency cepstral coefficients (MFCC), Spectral Centroid, log-Mel spectrum and Mel spectrum. Besides, the robustness of common feature extraction approaches are poor and Denoising AutoEncoders (DAE) is believed a efficient way to solve that. DAE uses a non-linear transform to reduce the dimension of input x with noise and drive x to y, and uses the gradient descent to decrease the distance (or value of other loss function) between the input without noise and the inverse of y [6].

Widely used classifiers include Support Vector Machine (SVM) [8], Logistic Regression, Gaussian Mixture Models (GMMs) [9], Hidden Markov Models (HMMs) [10] and Non-Negative Matrix Factorization (NMF) [11]. Using STC may also help to reduce the set of noise types (*the classes*),

Machine Learning for Signal Processing (Fall, 2020), Carnegie Mellon University.

enabling classifier ensembles, such as Random Forests [12], that train many similar classifiers on subsets of the training samples and/or feature-set.

3 Dataset

DCASES 2020 Challenge 5 uses the SONYC Dataset [2], which is constructed from over 100M 10-second audio clips collected over a 3 year period across 50 sensors in NYC. This dataset has been labelled with both coarse and fine grain labels, for example, coarse label 'machinery impact' includes fine labels like 'rock drill' and 'pile driver'. The set of positively labelled samples is around 18000 samples (much less than the entire set of collected audio). The entire set of classes is in Figure 2, of which we focus on the coarse-grain labels (*i.e.*, the larger boxes) to make the problem more tractable, especially because some of the classes are imbalanced, making generalized models difficult to build without resampling abd augmentation techniques.

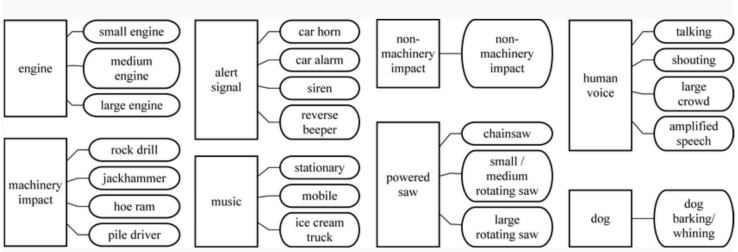

Figure 1: Set of coarse and fine grain classes labelled in the dataset.

4 Evaluation metric

DCASE 2020 Challenge 5 is evaluated based on the Area Under Precision Recall Curve (AUPRC). The challenge website describes their exact calculation of this for multiple coarse- and fine-grain classes by calculating the macro-average and micro-average of true positive class, false positive class and false negative class. False negatives are not included in the calculation of precision or recall.

5 Approach

5.1 Audio Preprocessing

We applied minimal audio preprocessing due to limited scope of the class. Since each audio file had different variations of noise, we were unable to use deep learning methods such as denoising autoencoders to learn the noise present in each files. Instead, the audio was sampled at 16 kHz yielding a 8 kHz low-pass cutoff to remove any high frequency content and reduce the frequency bandwidth of each mel-frequency cepstral coefficients (MFCC) and log-mel spectrum frequency band. With smaller frequency bandwidth, more concentrated information can be extracted.

5.2 Audio Feature Extraction

Each 10 second sample is broken into windows for short time Fourier transform (STFT). We tried window sizes of 25ms and 100ms. Long windows provide less resolution, but more-detailed features in frequency domain, therefore, may keep better result one tasks focusing on "timbre", such as sound event detection and speaker recognition etc.

The log-Mel spectrum is generated after filter banks and log scale from the result of STFT. To decrease the dimensionality and reduce noise, the spectrum is implemented with discrete cosine transform (DCT). The first 13 dimensions are used as the well-known MFCC features.

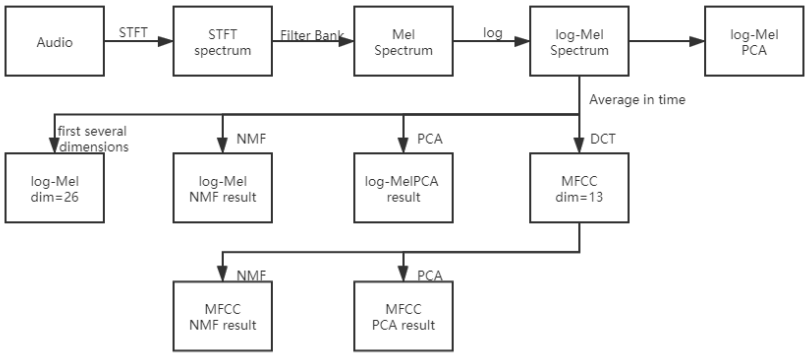

Figure 2: Pipeline of audio features extraction and dimension reduction.

5.3 Spatiotemporal Context Feature Generation

The annotations provided with each sample include a timestamp, accurate to the hour, and location, accurate to the nearest block, to help generate spatiotemporal context. Unmodified, these values are not particularly useful as they do not provide any generalizable information. We used these annotations to generate a set of features alongside those from audio. The set of STC features we generated generically fit into two buckets: temporal and queried.

The temporal features were generated from the timestamp. These included the time of day, the day of the week, a boolean indicator of the weekend, and the month. To make the set of values more continuous (and to avoid enumerated values), we applied to sine and cosine to the time of day and month to mimic the changes over time. We include a representation using a sine and cosine separately so that any time is not susceptible to being confused with another by virtue of phase repetitions (*e.g.*, $cos(\pi/4) = cos(-\pi/4)$ but $sin(\pi/4) \neq sin(-\pi/4)$). Sinusoids also wrap around, consistent with the periodicity we experience with the passing days and months.

The queried features were generated by referencing open data sources for New York City: OpenNYC [4] and PLUTO [5]. OpenNYC contains many databases, most of which are not useful or interesting. Since many noise types are based on construction or road-work, we generate several features to indicate ongoing construction within 2 blocks at the time the sample was taken. We also generate a feature to indicate the presence of a nearby subway, as this may point to increased pedestrians and traffic nearby. To further characterize the area, we generate three features using the PLUTO dataset to represent the composition of the neighborhood based on tax lots, categorized into 'residential', commercial', and 'manufacturing'.

5.4 Dimensionality Reduction

To reduce dimensions, the audio features (from MFCC and log-mel spectrum) were averaged across the time domain. Besides the averaged results themselves, the MATLAB implementations of principal components analysis (PCA) [15] and non-negative matrix factorization (NMF) [16] were applied to the averaged data to further reduce dimensionality. However, we found that this did not improve any classifier's performance. If allotted more time, we would apply PCA and NMF to the audio features before being averaged and reduced. This wasn't tried due to multiple MATLAB crashes and the long run time needed to apply PCA and NMF to each data point (2.386E8 data points for MFCC and 6.147E17 data points for log-mel spectrum).

5.5 Classifier Training and Testing

We trained and tested a variety of classifiers using audio feature with and without STC features. We tried the following classifiers: Support Vector Machine, Hidden Markov Model, Gaussian Mixture Model, AdaBoosting, Random Forest, K Nearest Neighbors, Logistic Regression, and Decision Tree. We used Scikit Learn [13] and hmm-learn[7] Python implementations for these classifiers.

We trained one-vs-all classifiers for each class; part of our reasoning is that multiple noise types can be present in the same sample, making it inappropriate to train a single classifier to detect a single type of noise as a multi-class problem. For each model, we tune hyperparameters to find those that give the best weighted average AUPRC.

6 Results

Models	Macro-averaged AUPRC w/ STC Features	Macro-averaged AUPRC w/o STC Features
HMM	0.2970	0.2970
GMM	0.3406	0.3307
Logistic Regression	0.3416	0.3276
Decision Tree	0.3486	0.3399
SVM	0.3498	0.3212
KNN	0.3537	0.3575
AdaBoost	0.3753	0.3701
Random Forest	0.4019	0.3952

Table 1: Macro-averaged AUPRC of all the classifiers we trained with the best parameters and feature sets.

Coarse Tag Name	AUPRC for Random Model	AUPRC of Our Best ML Model (Random Forest)
engine	0.3238	0.4332
machinery-impact	0.0211	0.0434
non-machinery-impact	0.0720	0.0768
powered-saw	0.0420	0.0061
alert-signal	0.1551	0.2147
music	0.0617	0.1304
human-voice	0.3961	0.5124
dog	0.0316	0.0475

Table 2: Macro-averaged AUPRC of random classifiers and our Random Forest Model

	MFCC	Log Mel-filterbank
long_window_size	0.3903	0.4019
long_window_size + PCA	0.3847	0.3870
long_window_size + NMF	0.3667	0.3901
short_window_size	0.3949	0.4019
short_window_size + PCA	0.3764	0.3939
short_window_size + NMF	0.3739	0.3952

Table 3: Macro-averaged AUPRC for different feature combinations using Random Forest

In the table 1, Adaboost and random forest generate better classification results. It makes sense because ensemble methods combine multiple weak classifiers to produce improved results comparing to a single model. Moreover, in terms of the macro-weighted AUPRC, adding STC did improve the result slightly by informing what sounds we may expect to hear in a recording. However, based on 3, STC features don't improve the performance of every class.

In the table 2, we compare the AUPRC of random classifiers and Random Forest. For the metric AUPRC, a random classfier would give a value of $P/(P+N)$, where P is the number of positive data points and N is the number of Negative data points. Our model is indeed better than random classifiers since our AUPRC score for most of the class is above random classifiers. In table 3, Log Mel-filterbank provide a better result. The reason might be that to obtain MFCCs, a Discrete Cosine Transform (DCT) is applied to the filter banks, which could hurt future training. Lastly, we noticed that change the window size and using dimension reduction did not improve the performance.

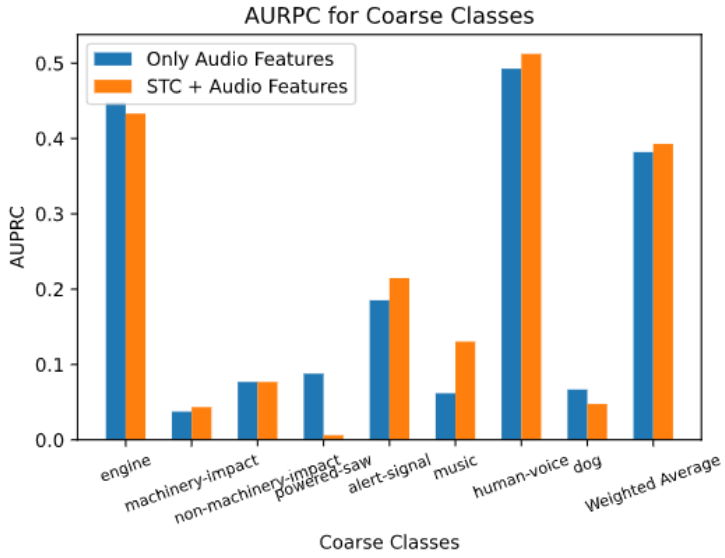

Figure 3: AUPRC for Random Forest trained with and without Spatio-Temporal Context Features

7 Analysis and Discussion

We find that several classes are especially difficult to distinguish, such as powered-saw and machinery-impact. We find the dataset is poorly balanced, which may influence this. However, we still find our results to be worse than the baseline. This may be related to our evaluation metric, AUPRC. Our classifiers produced binary outputs, so the PR curve is a line; we cannot vary our threshold for measuring positive detection of a class, making the 'area under' the curve coarse. We believe our results would improve if our classifiers output a continuous-valued confidence.

We also find that STC is not especially helpful, which is unfortunate as it was a large motivating factor in selecting this challenge. Context can be very broad and general, and it is likely that our dataset was not large and varied enough to capture the generality with the set of features we used. We note that it is most helpful with the class containing the most samples, human-voice. It likely that using STC directly as features is improper, as it glosses over context as a high-level representation of the surroundings. Perhaps we could have used something like a Bayesian inference model on the outputs, provided they were continuous valued, to better select likely types of noise *given* knowledge of the surroundings. In general, most submissions to the challenge had limited improvement from STC.

In addition, due to the limitations of this course, we did not implement the deep learning methods (ex. Convolution Neural Networks (CNNs) or Convolution Recurrent Neural Networks (CRNNs)) used by the most effective teams in the original challenge. We were also unable to use denoising autoencoders due to limitations to improve quality of audio files. These additions would've significantly improved our results.

8 Conclusion

In our course project, we took on the DCASE challenge for detecting urban noise using spatiotemporal context. We build an audio-processing pipeline to generate feature sets of many variants to test on a variety of classifiers. Using extraneous annotations in the dataset, we generate features intended to describe the surroundings when the audio was collected, and perform feature-level fusion to aid classification. We find that STC generally boosts the AUPRC by 0.005-0.02, depending on the classifier and hyperparameters; our best performance uses a Random Forest to produce 0.4019 AUPRC, which is unforunately below the baseline model provided with the dataset. We believe vast improvements would come from more careful usage of STC, a larger and more balanced dataset, and probabilistic classifier outputs. Please find our code on GitHub [14].

References

[1] Mark Cartwright, Ana Elisa Mendez Mendez, Jason Cramer, Vincent Lostanlen, Graham Dove, Ho-Hsiang Wu, Justin Salamon, Oded Nov, and Juan Bello. SONYC urban sound tagging (SONYC-UST): a multilabel dataset from an urban acoustic sensor network. In Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), 35–39. October 2019. URL: http://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Cartwright_4.pdf.

[2] Cartwright, M., Mendez, A.E.M., Cramer, J., Lostanlen, V., Dove, G., Wu, H., Salamon, J., Nov, O., Bello, J.P. "SONYC Urban Sound Tagging (SONYC-UST): A Multilabel Dataset from an Urban Acoustic Sensor Network", Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE) , 2019.

[3] Shikha Gupta1, Jafreezal Jaafar, Wan Fatimah wan Ahmad, and Arpit Bansal, Feature Extraction Using MFCC, Signal Image Processing : An International Journal (SIPIJ) Vol.4, No.4, August 2013, https://www.researchgate.net/profile/Shikha_Gupta35/publication/276200395_Feature_Extraction_Using_Mfcc/links/56f053a208ae0dcdafd6a8f8.pdf

[4] NYC Open Data. Retrieved November 9, 2020 from https://opendata.cityofnewyork.us/.

[5] Bytes of the Big Apple: PLUTO Tax Lot Data. Retrieved November 9, 2020 from https://www1.nyc.gov/site/planning/data-maps/open-data.page

[6] Vincent P, Larochelle H, Bengio Y, et al. Extracting and composing robust features with denoising autoencoders[C]. Proceedings of the 25th international conference on Machine learning. ACM, 2008: 1096-1103

[7] Lebedev, S. "HMMlearn." GitHub.(https://github. com/hmmlearn/hmmlearn) (2016).

[8] Zhu Y., Ming Z., Huang Q. (2007) SVM-Based Audio Classification for Content- Based Multimedia Retrieval. In: Sebe N., Liu Y., Zhuang Y., Huang T.S. (eds) Multimedia Content Analysis and Mining. MCAM 2007. Lecture Notes in Computer Science, vol 4577. Springer, Berlin, Heidelberg.

[9], G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in IEEE Conference on Advanced Video and Signal Based Surveillance, London, UK, 2007, pp. 21–26.

[10] T. Butko, F. G. Pla, C. Segura, C. Nadeu, and J. Hernando, "Two source acoustic event detection and localization: Online implementation in a smart-room," in 19th European Signal Processing Conference.

[11] Komatsu, T., Toizumi, T., Kondo, R., Senda, Y. (2016, September). Acoustic event detection method using semi-supervised non-negative matrix factorization with a mixture of local dictionaries. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016) (pp. 45-49).

[12] Criminisi, A., Shotton, J., Konukoglu, E. (2012). Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Foundations and Trends® in Computer Graphics and Vision, 7(2–3), 81-227.

[13] SK Learn: Supervised Learning. Retrieved December 10, 2020 from https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

[14] MLSP Team 3 Code Repository. Retrieved December 12, 2020 from https://github.com/reese-grimsley/mlsp-f20-team3

[15] Principal Component Analysis of Raw Data. Retrieved December 12, 2020 from https://www.mathworks.com/help/stats/pca.html

[16] Nonnegative Matrix Factorization. Retrieved December 12, 2020 from https://www.mathworks.com/help/stats/nnmf.html