A Survey of Sound Classification Using Classic Methods and Deep Learning

Filippo Ranalli
franalli@stanford.edu

Hiroshi Mendoza
hmendoza@stanford.edu

Abstract

Approximately 5% of the world population is afflicted by a form of hearing loss, which can lead to difficulties in identifying nearby dangers or meaningful occurrences around them. Such problem motivates research aimed at building systems that can recognize surrounding sounds in real time, and provide rapid feedback to the user. In this paper we attempt to tackle sound recognition by analyzing predictive models, using shallow and deep AI techniques to classify environmental and urban sound sources. It is within our goals to analyze their performance, comparing their advantages and shortcomings. After building an ensemble of methods of increasing complexity on a small but representative dataset, we were thus far able to achieve a classification accuracy of 70% using a four-layer fully-connected neural network.

Introduction

Sound is registered as a wave in the time domain. However, in order to extract salient information from this signal, it needs to be converted to the frequency domain. To do so, we make use of an open source music information retrieval library package -Librosa that is able to take a .wav file and provides different feature representations of the sound wave in terms of spectral, pitch, and contrast information. A combination of these different features will serve as the input to our model, while the output will consist of the class that the data point is most likely to belong to, according to a cross entropy distribution. The particular feature disposition may vary from model to model, as it may be appropriate to horizontally concatenate them for a shallow classifier or a feed-forward network, while a CNN may require a three-dimensional image representation, and an RNN a time series.

Classifying sounds is not new. Commercially, Shazam[18] has allowed millions of users to classify music via their application by converting sound into the frequency domain and matching the signature to their music database to identify the song. A similar approach will not work in our case since it's impossible to capture a unique signature among different scenarios of crying babies or glass shattering. In the biomedical domain, research groups in PARC have used deep learning to recognize abnormal heart sounds, where they transformed the one dimensional time series into a two-dimensional time-frequency representations to feed into their deep neural network. They were able to reach a high accuracy[14]. In [15], Salamon et al. explore an array of shallow models, which yield satisfactory performance, but could clearly be outperformed by deeper architectures. Researchers also have used convolutional neural networks to classify environmental sounds. They represented sound as segmented spectrograms and used 2 convolutional layers. Although the experiments showed that the convolutional model performed better than common approaches based on manually-engineered features the results were not impressive given the gap to human accuracy. They blamed their limited datasets and perhaps suggested for future work to combine convolutional neural networks with other less complex models[12]. In SCAPER[16], a new feature extractor is implemented, and a CNN with an AlexNet architecture is applied to it for learning. The performance of these models will serve as our initial target to replicate when building our own models. It is furthermore important to mention Google's recent investigation on sound classification through a collection of CNN architectures[7]. This paper presents a performance comparison of FC, AlexNet, VGG, and ResNet on the AudioSet database, where ResNet[6] is the clear winner with an accuracy as high as 0.926. Such results suggest that ResNet is the most successful network for visual recognition thus far, and it would be of interest for us to compare it with an RNN on a smaller dataset.

Dataset

The dataset we chose for our first experimentation is the UrbanSound8K[2], which provides 8732 labeled and evenly distributed data points from 10 different classes: air conditioner (AC), car horn (CH), children playing (CP), dog bark (DB), drilling (DR), engine idling (EI), gunshot (GS), jackhammer (JH), siren (SI), and street music (SM). Each example is a 4 second .wav file. This is a good dataset to start working on due to the moderate training time, but definitely not large enough for training state-of-the-art models. For training, we have divided the data into buckets such that 80% is the training set, 10% is the development set and another 10% is the hidden test set, which we will run our models on at the end of our project. In the future we hope to work on Google's 2.1 million-datapoint-strong AudioSet, given adequate computational resources and time. AudioSet has 527 classes of annotated sounds and over 5 thousand hours of audio.

Method

Feature Extraction

Each data point in UrbanSound8K is composed of a wave in the time-amplitude domain. In order to extract salient information about each sound, it is necessary to convert the waves to the time-frequency domain. With the help of the Librosa library and the features selected in a similar study by [5], we convert the wave to the frequency domain using the Short-Time Fourier Transform (STFT) with a window size of 2048. We then use the obtained spectrogram of shape [168,133] to extract the following features, the most expressive of which are shown in fig.1:

- Mel-Scaled Spectrogram (mel): scaled STFT based on a range of perceptual pitches.
- Chromagram (chroma): pitch class bucketing of STFT.
- Mel-frequency Cepstral Coefficients (mfccs): STFT linear cosine transform of mel.
- Spectral Contrast (contrast): relative distribution of harmonic and non-harmonic components of STFT[9]
- Tonal Centroid Features (tonnetz): chromagram-based transformation of STFT.
- Zero Crossing Rate (zcr): rate of sign change across time signal.
- Root-Mean-Square Energy (rmse): measure of energy given the STFT signal.
- Spectral Bandwidth (bw): wavelength interval in which a radiated spectral quantity is not less than half its maximum value [3]

The features described above present varying shapes in the time-frequency domain, hence a unified embedding summarizing their values needs to be defined. As all the features are discretized along a given number of time steps, a first approach is to average each of them along the time axis, and concatenate them into a feature vector of shape $[1, 193]$. While this is a good initial approach, it may lead to over-representing the feature space, as some features are strongly correlated with one another. To curb this issue, we can run dimensionality-reduction algorithms and learn a new lower-dimensional embedding.

The first algorithm of choice is PCA[4], which learns k-dimensional embeddings by performing singular value decomposition on the data covariance matrix. Such matrix is defined as $\Sigma = \frac{1}{m}\sum_i^m x_i x_i^T$, where x_i are the mean-centered and variance-normalized data points, and the new features are its k eigenvalues. The lower-dimensional embeddings that yield the best performance are of dimensions $[1, 185]$.

Oracle

As seen in Piczak's oracle[13], even humans make mistakes in identifying the source of sounds in the urban environment, but on average can classify 98% percent of the sounds. A working oracle would hence consist of a person with normal hearing capabilities.

kNN Baseline

K Nearest Neighbor (kNN) is a great baseline algorithm that looks for similarities between each data point and the entire training set. Each data point x_j it compared to the entire training set by a point-wise distance $L2_i = \sqrt{\sum_d (x_i - x_j)^2}$, and a majority vote of the K training data points determines the classification label. In the training phase, we learn which K yields the best results, and then use it for the validation set.

SVM

A support vector machine (SVM) is a classifier that uses a hinge loss and can use a nonlinear kernel to transform the feature domain for improved separability. The loss function and the kernel are described by the following equations, respectively:

$$\begin{gathered} L = \frac{1}{N}\sum_i \sum_{j\neq y_i} H(x_i; W; K) + \lambda W^T K W \\ K(x, z) = exp(-\frac{1}{2\tau^2}||x - z||_2^2) \end{gathered} \tag{1}$$

The hinge function $H(x_i; W; K)$ is the maximum between 0 and the largest gap between the correct class score and the predicted class score, minus a predefined margin Δ. The second component of the loss is the L2 regularization term, which is controlled by the strength λ. The Kernel function indicated above is the radial basis function, which performed best in the task of projecting the data into a separable space. The solver deployed is the SVC function in the Sklearn package which uses a "one-vs-rest" approach to learn the parameter matrix W. Because the Kernel is a greater-dimensional feature representation of the data [11],

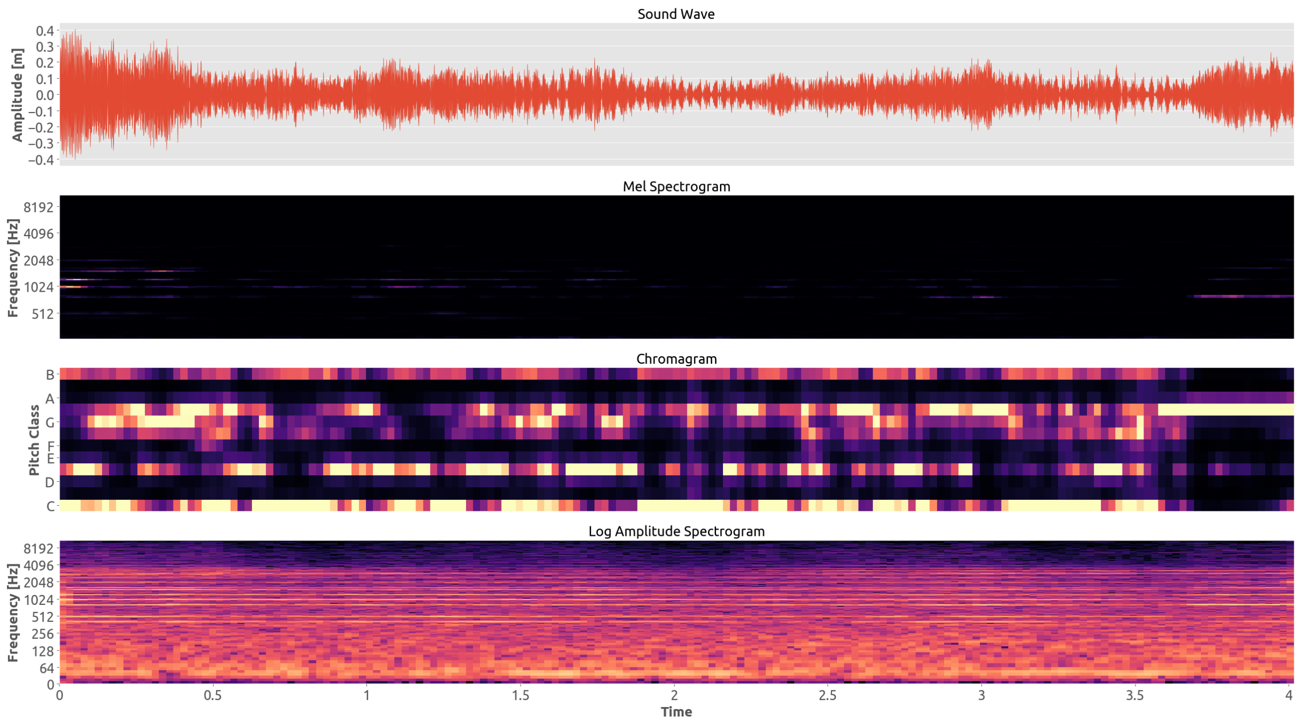


Figure 1. Car Horn Feature Visualization

the number of parameters dramatically scales up with the dataset size.

Random Forest

The Random Forest classifier is an ensemble method based on decision tree modules. In such model, all observed information is encoded in the edges, and all predictions in the leaves. A tree can be learned by recursively splitting the source set into subsets based on an attribute value test[3]. Analogously to SVM, in this classifier the number of parameters is a function of the dataset size. However, this number can be pruned by tuning the maximum depth of the tree, the minimum number of samples to split an internal node, the minimum number of samples at each leaf node, and the maximum number of total nodes[1]

One interesting aspect about this classifier is that the edge weights that it learns are proportional to the feature importance in the classification decisions. These weights can help us distinguish the most impactful features, and they are shown in the results section. To show the relative feature importance, PCA was turned off.

Feed-Forward Neural Network

The models described and implemented in the above succession provide significant insight on the importance of the features and the overall behavior of shallow classifier, but they do not scale well with larger datasets and present drastic overfitting of the training set. A simple feed-forward neural network is our first attempt at a classifier that can learn many more parameters and be much more expressive. This model, as well as all the other models that will follow, are implemented in PyTorch from scratch. The most successful architecture we implemented is as follows:

- Input[N,193] $\rightarrow$ Affine $\rightarrow$ Batch Norm $\rightarrow$ ReLU
- Affine $\rightarrow$ Batch Norm $\rightarrow$ ReLU
- Affine $\rightarrow$ Batch Norm $\rightarrow$ ReLU
- Affine $\rightarrow$ Batch Norm $\rightarrow$ ReLU $\rightarrow$ score[10,1]

Where the predicted score is the maximum over the ten classes. The loss function adopted is the cross entropy loss $L_i = -score_{correct} + log \sum_{classes} e^{score_{class}}$. All parameters are initialized with the Xavier initialization. To help curb overfitting issues on the training set, a dropout[17] probability of 0.75 is adopted alongside an L2 regularization on the loss function. Batch normalization[8] after each affine layer is deployed to speed up the training and also help with overfitting. Because both dropout and batch normalization behave differently at test and training time, it is important to specify the behavior in the model. The dataset is divided into randomly sub-sampled batches, on which the optimizer performs a gradient update for each epoch, defined as a single run of the whole mini-batched dataset. The optimizer of choice is Adam, which uses adaptive estimates of lower-order moments[10]. Furthermore, the four affine layers are all of hidden dimension 200, the learning rate starting at $1e^{-3}$ is decayed exponentially.

Results

The first model implemented is the kNN baseline, which achieved an accuracy of 53% on the development set with K = 6. This model is our lower bound on the performance, which is slightly over five times better than random. Moving onto SVM, the RBF kernel function performed far better than the other kernels, and achieved an accuracy of 61% on the development set. Moreover, the Random Forest classifier achieved 64% on the development set, while the four-layer feed-forward net performed as high as 70%. The number of parameters, as well as the performance on the training and test sets are summarized in Table1. We can clearly see how the neural network presents the best trade-off in terms of number of parameters, scalability and performance. Because the parameter space of SVM and Random Forest is directly proportional to the dataset size, it becomes clear that these algorithms would not be tractable on a much larger dataset.

Table 1. Model Overview

Model	Parameters	Train Acc	Val Acc
kNN	1	0.92	0.53
SVM	64000000	0.89	0.61
Random Forest	2408240	0.94	0.64
Four Layer NN	158608	0.88	0.7

It is clear that all the models attempted heavily overfit on the training set, but this is to be expected due to the small size of the dataset. Even on a heavily-regularized neural network, the learning curves in Fig.2 show the increasing gap towards the end of training. However, we can also notice how the neural network is the model that overfits the least, due to the deployment of dropout and L2 regularization.

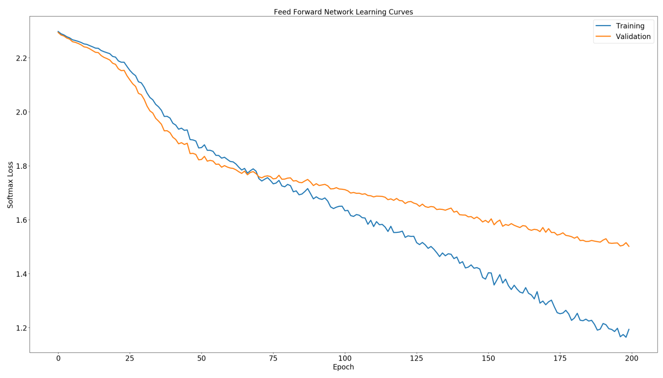


Figure 2. Neural Network Training Curves

An interesting aspect of the Random Forest classifier is the possibility to view the edge weights for each feature, the importance of which can be seen in the histogram plot in Fig.3. Is is hereby clear that the mel-scaled spectrogram has the largest impact on classification, which reinforces the intuition that the most salient learning can be done on more expressive representations of the spectrogram itself.

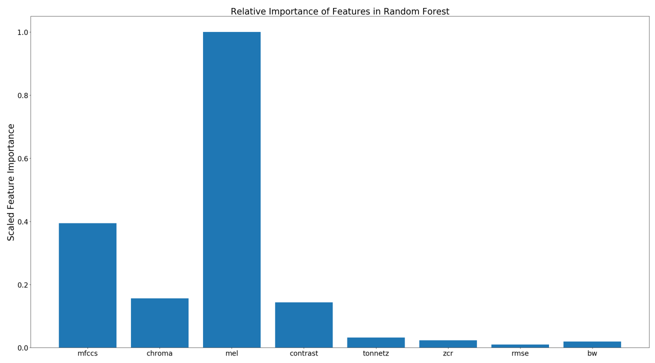
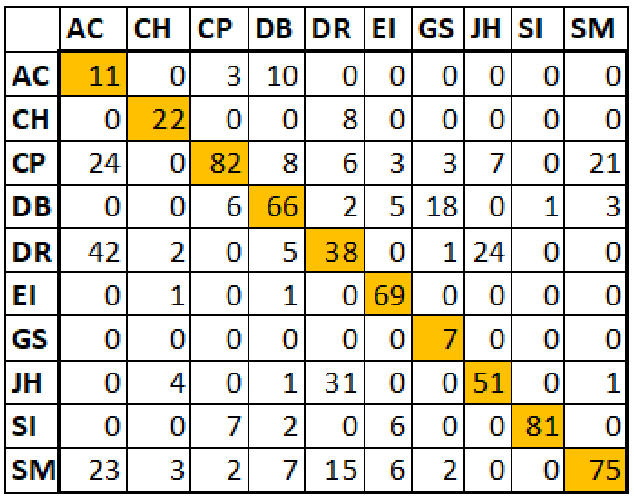


Figure 3. Scaled Random Forest Feature Importance

Finally, another useful metric for evaluating model performance is the confusion matrix over the development set, which is displayed in Table.4 with respect to the neural net classifier. By examining the matrix, we notice that the two most common mis-classifications are "Drilling" mistaken as "Air Conditioning", and "Jack hammer" mistaken as "Drilling". By taking a closer look in the time-wave domain, the amplitudes of these signals look very different. However, their spectra look similar, especially in the lower frequencies which dominate over the higher ones.



	AC	CH	CP	DB	DR	EI	GS	JH	SI	SM
AC	11	0	3	10	0	0	0	0	0	0
CH	0	22	0	0	8	0	0	0	0	0
CP	24	0	82	8	6	3	3	7	0	21
DB	0	0	6	66	2	5	18	0	1	3
DR	42	2	0	5	38	0	1	24	0	0
EI	0	1	0	1	0	69	0	0	0	0
GS	0	0	0	0	0	0	7	0	0	0
JH	0	4	0	1	31	0	0	51	0	1
SI	0	0	7	2	0	6	0	0	81	0
SM	23	3	2	7	15	6	2	0	0	75

Figure 4. NN Confusion Matrix

Conclusion and Looking Ahead

The models evaluated thus far present a collection of models with increasing complexity, each with its drawbacks and advantages. The future steps we envision taking from here revolve around more complex classifiers such as CNN and RNN, possibly around a larger dataset. It would furthermore be of interest to analyze the intermediate layer representations of the CNN, showing what features each filter focuses on, as well as saliency maps to understand which areas of the spectrograms are critical to the class scores. Another possible extension is to try auto encoders instead of PCA for dimensionality reduction, as these are non-linear and hence potentially more expressive. Finally, it would be interesting to compare the performance of the deep models explored using space-saving techniques for real-time applications, such as converting the parameters from 32-bit floats to 8-bit floats.

References

[1] Skearn.

[2] Urbansound8k dataset.

[3] Wikipedia.

[4] H. Abdi and L. Williams. Principal component analysis.

[5] aqibsaeed. Urban sound classification.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[7] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. W. Wilson. CNN architectures for large-scale audio classification. *CoRR*, abs/1609.09430, 2016.

[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[9] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai. Music type classification by spectral contrast feature.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] H. Q. Minh, P. Niyogi, and Y. Yao. Mercers theorem, feature maps, and smoothing.

[12] K. J. Piczak. Environmental sound classification with convolutional neural networks.

[13] K. J. Piczak. Esc: Dataset for environmental sound classification.

[14] J. Rubin, R. Abreu, A. Ganguli, S. Nelaturi, I. Matei, and K. Sricharan. Recognizing abnormal heart sounds using deep learning.

[15] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research.

[16] J. Salamon, D. MacConnell, M. Cartwright, P. Li, and J. P. Bello. Scaper: a library for soundscape synthesis and augmentation.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutseskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

[18] A. L.-C. Wang. An industrial-strength audio search algorithm.