Environment Sound Classification: Team Project

Kipp McAdam Freud
Interactive Artificial Intelligence CDT
University of Bristol
km14740
km14740@bristol.ac.uk

Stoil Ganev
Interactive Artificial Intelligence CDT
University of Bristol
sg13286
stoil.ganev@bristol.ac.uk

I. Introduction

Environmental sounds were first defined by Vanderveer as "any possible audible acoustic event which is caused by motions in the ordinary human environment" [2]. These sounds are often more chaotic and noise like than standard speech audio; they also lack the underlying phonetic structure characteristic of speech, which has been successfully modelled using methods such as hidden Markov models [4].

The classification of environmental sounds has numerous possible applications in areas such as audio surveillance systems, hearing aids, smart room monitoring, and video content highlight generation [3]. Techniques which have been applied to this problem include matrix factorization, dictionary learning, wavelet filterbanks, and, most successfully, deep neural networks [5].

Most deep learning approaches to this problem augment the input audio data in some way, then utilize a deep convolutional neural network to classify environmental noises. One such approach was presented by Su, Zhang, Wang and Madani in their 2019 paper *Environment Sound Classification using a Two-Stream CNN Based on Decision-Level Fusion'* [1]. This paper claimed their approach achieved the highest classification accuracy of all existing models when tested on a publicly available environmental sound dataset, the *'UrbanSound8K dataset'* [10]. This paper documents our attempt to replicate the methods and reported accuracy of these authors. Using their publication as a guide, we have reconstructed their described network and examined its performance.

II. Related Work

A. Hidden Markov Models

Before 2015, the general approach to environmental sound classification was to use classifiers such as support vector machines, Gaussian mixture models, or hidden Markov models, to classify noises based on manually extracted features [7].

One example of this approach is taken from Chum, Habshush, Rahman, and Sang's 2013 paper *'Scene Classification Challenge using Hidden Markov Models and Frame Based Classification'* [8]. This paper modelled environmental noise as a Markov process, where each audio scene was assumed to have states that it transitioned between.

This approach split audio clips into frames, each containing 512 samples. For each frame, manually specified features were extracted, these features included a magnitude response (defined as the magnitude, in decibels, of the first 256 coefficients of a 512-point FFT of the frame), a loudness measure, a spectral sparsity measure, and a temporal sparsity measure.

Using these features, a hidden Markov model learned the transition matrix between states - a transition matrix was learnt for every class of environmental noise. For each new audio sample, the model calculated the features, and computed the likelihood that those features came from a specific class. The system classified the audio sample as having come from the class with the highest likelihood.

B. Deep CNN Approaches

The first publication which used a deep CNN architecture to tackle the problem of environmental sound classification was published by K. Piczak in 2015 [6].

The architecture used was relatively simple; the model consisted of 2 convolution layers with max-pooling applied, followed by 2 fully connected layers. ReLU activation functions were utilized, as was a dropout rate of 0.5 (during training) and a 0.001 L2 weight decay value to prevent overfitting. The network was trained using segmented log-scaled mel-spectrogram representations of audio data.

After training, it was shown that the convolutional model was able to outperform many of the past approaches, particularly those which relied on manually engineered features (for example the hidden Markov model approach described above). However, the author notes that because of the far greater training times, these results were "far from groundbreaking". One particular issue faced with this approach was the scarcity of training data - it was assumed that a larger dataset would yield higher accuracy.

The paper concluded that CNNs were a viable solution to the problem of environmental sound classification.

C. Deep CNNs with Data Augmentation

In 2017, the deep convolutional neural network approach to environmental sound classification was improved dramatically. Salamon and Bello's paper *Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification* [9] incorporated the use of data augmentation to overcome the problem of data scarcity.

The architecture used in this paper had three convolutional layers, with two pooling operations, followed by two

fully connected layers. Similar to the paper described above, these networks were trained using segmented log-scaled mel-spectrogram representations of audio data.

This paper experimented with four different augmentation techniques:

- Time Stretching - speeding up or slowing down the audio without changing the pitch).
- Pitch Shifting - changing the pitch of the audio without changing the duration.
- Dynamic range compression - compressing the dynamic range of the audio.
- Background noise - Mixing the audio with another recording containing only background noise.

This network was trained using the same dataset as was used in the Piczak paper [6], and the networks were trained both with and without the use of data augmentation.

It was found that the proposed architecture performed comparably to Piczak's architecture without data augmentatiom, with a mean accuracy of 74% and 73% respectively. However, when trained using the larger dataset generated by data augmentation, the performance of the new model increased significantly, yielding a mean accuracy of 79%.

Though, unfortunately, the authors do not note the change in accuracy of Piczak's architecture when used with data augmentation, it is clear from this paper that data augmentation provides a significant boost to the environmental sound classification accuracy of deep CNNs; this point is validated by the fact that this approach reportedly achieved state-of-the-art results.

III. Dataset

The UrbanSound8K dataset [10] contains 8732 sound clips of a variety of environmental sound sources. The dataset contains ten classes of environmental sound: air conditioner (ac), car horn (ch), children playing (cp), dog bark (db), drilling (dr), engine idling (ei), gun shot (gs), jackhammer (jh), siren (si), and street music (sm).

In the version of the dataset we are using, the 8732 sounds clips have been split into 55964 sound segments total. Out of these, 50569 are used for training and 5395 for validation. The class distribution can be seen in figure 1. It is identical between the training and validation splits.

The data is provided as a pair of `.pkl` [11] files - one for the training samples and one for the validation samples. These files contain the data in the form of a Python list whose elements are Python dictionaries, one for each segment. In these dictionaries we have the features, the name of the file it originated from, the full class name and a class ID which is just and integer representation of the class label.

IV. Input

Each of the samples in the dataset contains 5 different preprocessed audio features derived from the sound wave of the segment. These are combined in different ways in order to construct the input of the network variants we explored. In each segment we are given 41 time steps of the recording.

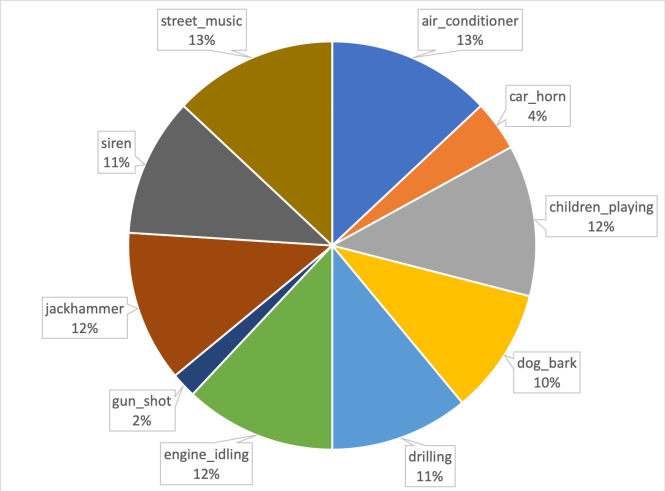


Fig. 1: Class distribution of the samples in the provided UrbanSound8K dataset

The *chroma* feature represents the tonal content of a musical audio signal in a condensed form [15].

The *tonnetz* is a conceptual lattice diagram representing tonal space - first described by Euler in [13].

The *spectral contrast* feature represents the spectral characteristics of an audio clip [14].

The *log mel spectogram* is a standard log scaled spectogram with the frequency axis scaled according to the mel scale - which transforms the frequency space to be more representative of how a human percieves sound (i.e two equally spaced points on the mel scale will sound 'equally spaced' to a human.) [16].

The *mel frequency cepstral coefficients (MFCC)* features are commonly used in speech recognition - they are harder to conceptualize, but to simplify them hugely, they represent the 'amount' of each type of phoneme used in an audio clip [16].

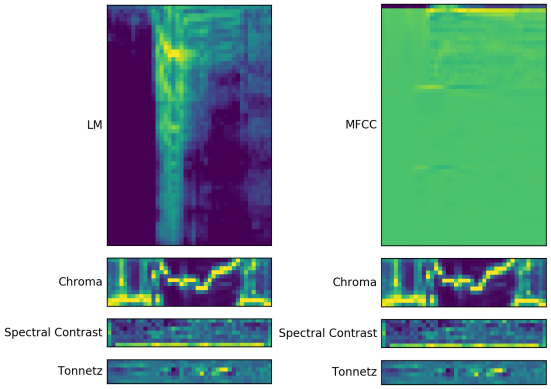
A. Feature Sets

For each audio sample a feature set is generated by combining some of the features described above. Separate networks are trained using each feature set, and results will be compared. We see examples of MC and LMC feature set in figure 2.

- The *LMC* feature set is made by concatenating the log-mel spectrogram, chroma, spectral contrast and tonnetz features.
- The *MC* feature set is made by concatenating the MFCC, chroma, spectral contrast and tonnetz features.
- The *MLMC* feature set is made by concatenating the log-mel spectrogram, MFCC and tonnetz features.

V. Architecture

The replicated paper contains inconsistencies, and lacks the information required to perfectly replicate the authors network without significant trial and error. These inconsistencies will be discussed in section VI - here we describe the architecture we have used.



(a) LMC input (b) MC input

Fig. 2: A training sample of the class dog_bark

1) Images are given to the network and passed through a convolutional layer consisting of 32 kernels, each having a 3×3 receptive field. 1×1 padding is applied to the input.
2) The Rectified Linear Unit (ReLU) activation function is applied.
3) Batch normalization is applied.
4) The representations are passed through a second convolutional layer consisting of 32 kernels, each having a 3×3 receptive field. 1×1 padding is applied to the input.
5) The ReLU activation function is applied.
6) Batch normalization is applied.
7) A max pooling is applied with a 2×2 kernel and a stride step of 2×2 - halving the height and width of each layer of the representation.
8) At training stage, a dropout is applied with 0.5 dropout probability.
9) The representations are passed through a third convolutional layer consisting of 64 kernels, each having a 3×3 receptive field. 1×1 padding is applied to the input.
10) The ReLU activation function is applied.
11) Batch normalization is applied.
12) The representations are passed through a fourth convolutional layer consisting of 64 kernels, each having a 3×3 receptive field. 1×1 padding is applied to the input.
13) The ReLU activation function is applied.
14) Batch normalization is applied.
15) A max pooling is applied with a 2×2 kernel and a stride step of 2×2 - halving the height and width of each layer of the representation.
16) At training stage, a dropout is applied with 0.5 dropout probability.
17) The representations are flattened into 1-dimensional vector representations of the input images.
18) The representations are passed through a fully connected linear layer with 1024 outputs.
19) A sigmoid activation function is applied.
20) At training stage, a dropout is applied with 0.5 dropout probability.
21) The representation is passed through a second fully connected layer with 10 outputs, one for each class in the data set.
22) A softmax activation function is applied to these outputs of length 10.
23) The length 10 outputs are returned by the network.

The output of the network can be thought of as the networks 'scores' for each class of environmental noise, with a higher score for a particular class indicating higher confidence that the input image was generated from audio of that class, with the class with the highest score being used as the given classification of an input image.

During training, Adam is used as an optimizer with a learning rate of 0.001 and L_2 regularization is used with a lambda of 1e-5. A batch size of 32 is used. Cross-entropy loss has been used as the loss function.

A. Late Fusion Model

The TSCNN model is formed using the networks trained on the LMC and MC feature sets. Class 'scores' from this network are generated by averaging the length 10 outputs of these networks - the predictions correspond to the class with the highest 'score', as with the standard networks.

VI. Implementation Details

Implementing the network described by Su, Zhang, Wang and Madani required significant guesswork, as their paper contains many inconsistencies which had to be resolved. The architecture described in the section above is the best recreation possible based on the information provided. The system was implemented using PyTorch.

A. Stride Step

The first of these inconsistencies was in the stride step used in the convolutional layers. The authors claimed a stride step of 2×2 for every convolutional layer in section *3.2*, this would reduce the height and width of the representations passing through this layer by half. However, we see in figure 4 of their paper that the height and width of the representations do not change, except when a max pooling step is applied.

We have assumed that the authors claimed a stride step of 2×2 erroneously. This is validated by looking at table 1 in the paper; which shows that the first 2 convolutional layers take the same amount of memory, implying no reduction in representation size. It is possible that the authors meant to report that the max pooling applications have a stride step of 2×2, which is indeed the case.

B. Dimensionality Reduction

There is an unexplained dimensionality reduction between the last convolutional layer and the first fully connected layer in the paper; this is briefly mentioned in section 4 of their paper. Since this dimensionality reduction seems to half the height and width of the representations, we have chosen to

apply a max pooling with a 2×2 kernel and a stride step of 2×2 - this achieves the desired results.

C. Optimization

The authors claim they are using a 'variant of stochastic gradient descent', and cite the 2014 paper describing Adam [12], a stochastic optimizer. However, they also claim they are using a momentum value of 0.9 - Adam doesn't use momentum. The authors also claim to use L_2 regularization, but do not give a value of the regularization parameter.

We believe that the momentum value was given erroneously, and we have chosen to ignore it and use Adam as our optimizer. We set the L_2 regularization parameter to 1e-5.

VII. Replicating Quantitative Results

We trained networks on each of the feature sets described in section IV-A, and used the LMC and MC networks to form the TSCNN late fusion network. Each network was trained for 50 epochs, which took approximately 1 hour using BlueCrystal. Our class-wise accuracy of the four networks on our validation set is presented in figure 3.

Class	LMC	MC	MLMC	TSCNN
ac	61.00	69.00	46.00	72.00
ch	90.91	69.70	87.88	81.82
cp	76.00	89.00	90.00	90.00
db	72.00	72.00	71.00	75.00
dr	84.00	77.00	86.00	82.00
ei	67.74	68.82	82.80	73.12
gs	100.00	81.25	96.88	96.88
jh	100.00	88.54	98.96	96.88
si	55.42	72.29	55.42	68.67
sm	96.00	81.00	82.00	90.00
Avg	78.38	77.18	78.02	81.84

Fig. 3: Replicated class-wise accuracy of four models with four-layer CNN evaluated on test set of UrbanSound8K test dataset

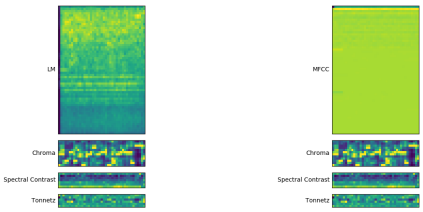
We note that we have achieved average accuracy over 10% below those that we have attempted to replicate. These results are, however, similar to other replications, being within around 5% of the accuracy claimed by Damen, Burghardt et al. in [17]. We see that our late fusion network significantly outperforms the other networks, all of which have similar accuracies.

VIII. Training Curves

In figure 4a we see the training loss against batch number. We note that the unsmoothed curve (seen lightly in the figure) contains significant noise. In figure 5a we can see the validation loss against batch number. Validation data was only taken once every 5 epochs, thus data is sparse, however we can see a downward trend with no obvious signs of overfitting. Interestingly, the validation loss does not decrease monotonically. We note that the first validation step was undertaken after the first 5 epochs, when the network had mostly converged. We observe complementary patterns in the average accuracy curves shown in figures 5b and 4b.

IX. Qualitative Results

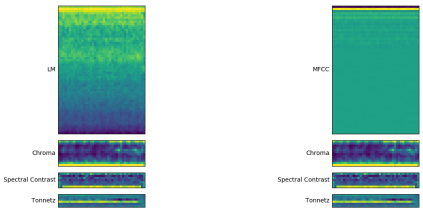
In this section we detail sample successes and failures of our models. Our first example is the 3rd audio sample in our UrbanSound8K test dataset. In this case, the audio contained the environmental sound of car horns - both the network trained on the LMC feature sets and on MC feature sets classified this audio sample correctly.



(a) LMC input (b) MC input

Fig. 6: The LMC and MC images generated by a correctly classified car horn noise.

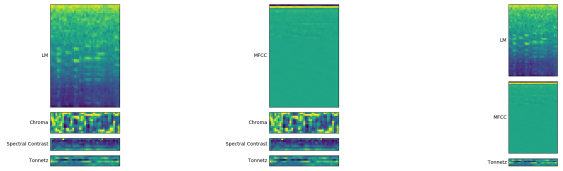
Secondly, we will look at the 340th audio sample in our UrbanSound8K test dataset. In this case, the audio contained the environmental sound of an idling engine - while the network trained on the LMC feature sets was able to classify this sample correctly, the network trained on MC feature sets classified this audio sample incorrectly as containing air conditioning noise.



(a) LMC input (b) MC input

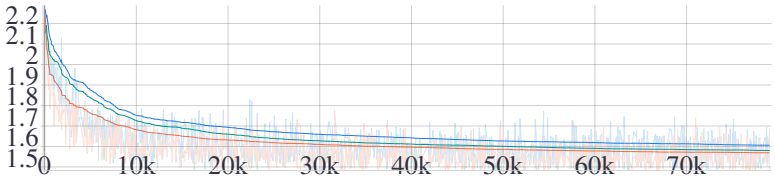
Fig. 7: The LMC and MC images generated by an incorrectly classified engine idling noise.

We will also look at the 1008th audio sample in our UrbanSound8K test dataset. In this case, all four models classified this noise incorrectly as coming from the 'children playing' class, however the noise was actually a recording of street music.

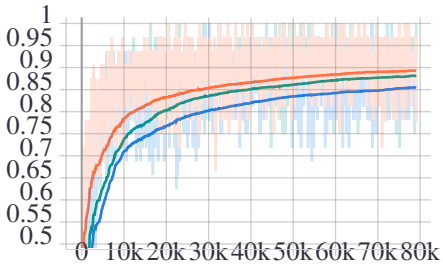


(a) LMC input (b) MC input (c) MLMC input

Fig. 8: The LMC, MC, and MLMC images generated by a universally incorrectly classified street music noise.

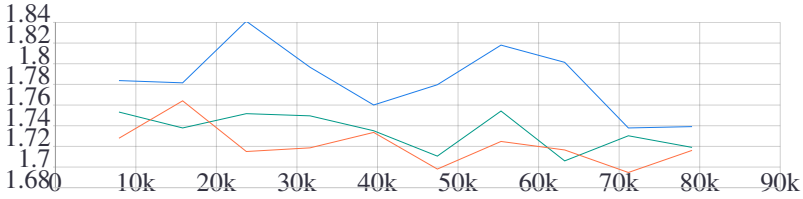


(a) Smoothed training loss of networks.

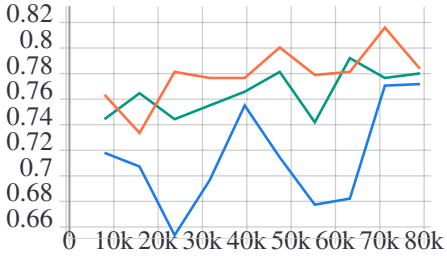


(b) Smoothed training average accuracy of networks.

Fig. 4: Smoothed training loss (left) and average accuracy (right) curves for trained networks; MC (blue), MLMC (green), and LMC (orange). Non smoothed training loss and accuracy is lightly shown behind.



(a) Validation loss of trained networks.



(b) Validation accuracy of trained networks.

Fig. 5: Validation loss (left) and average accuracy (right) curves for trained networks; MC (blue), MLMC (green), and LMC (orange).

Interestingly, there were no cases where the TSCNN 'late fusion' network was correct when all other networks were incorrect, despite the significantly higher accuracy. As this network makes predictions by evaluating the scores of two other networks, we believe that the increase in accuracy accounts for cases where one network is highly confident that a noise has been generated by a particular class of noises, while the other network is unsure. In this case, any correct predictions made by the late fusion model would also be made by the 'confident' network.

X. Conclusion and Future Work

We conclude that we have been unable to replicate the accuracies claimed by Su, Zhang, Wang and Madani in their 2019 paper *Environment Sound Classification using a Two-Stream CNN Based on Decision-Level Fusion'* [1], thus we are unable to verify their findings, or their claim to have achieved state of the art results.

In future, we would suggest performing data augmentation of the UrbanSound8K dataset to increase the dataset size. Methods for performing these augmentations were described in section II-C - these augmentation techniques have proved effective in the past, and would almost certainly increase the accuracy of the system described in this paper.

References

[1] Y Su, K Zhang, J Wang and K Madani. Environment Sound Classification using a Two-Stream CNN Based on Decision-Level Fusion. 2019.

[2] NJ Vanderveer. Ecological acoustics: human perception of environmental sounds. 1979.

[3] KJ Piczak. ESC: Dataset for environmental sound classification. 2015.

[4] M Huzaifah. Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks. 2017.

[5] S Sigtia, A Stark, S Krstulovic and M Plumbley. Automatic Environmental Sound Recognition: Performance Versus Computational Cost. 2016.

[6] K. J. Piczak, Environmental sound classification with convolutional neural networks. 2015.

[7] D. Barchiesi, D. Giannoulis, D. Stowell and M. Plumbley. Acoustic scene classification: Classifying environments from the sounds they produce. 2015.

[8] Chum, May and Habshush, Ariel and Rahman, Abrar and Sang, Christopher, IEEE AASP scene classification challenge using hidden Markov models and frame based classification. 2013.

[9] J Salamon, JP Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. 2017.

[10] J. Salamon, C. Jacoby and J. P. Bello, A Dataset and Taxonomy for Urban Sound Research. 2014.

[11] Pickle Python Module, https://docs.python.org/3.7/library/pickle.html, retieved 13/01/2020

[12] Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. 2014.

[13] L. Euler. Tentamen novae theoriae musicae ex certissismis harmoniae principiis dilucide expositae (in Latin). 1739.

[14] Jiang, Dan-Ning, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. 2002.

[15] Kattel, Manasi & Nepal, Araju & Shah, Ayush & Shrestha, D. Chroma Feature Extraction. 2019

[16] Logan, Beth and others. Mel Frequency Cepstral Coefficients for Music Modeling. 2000.

[17] Unit Web Page: Applied Deep Learning, University of Bristol. https://comsm0018-applied-deep-learning.github.io/. 2020.