# Applied Deep Learning Report

David Sharp (ds16797), Nashe Mncube (nm15042), Miklos Borsi (mb16240)

## I. Introduction

Within this report we will outline our solution for the assigned coursework. This will involve detailing our implementation of one of the detailed neural nets outlined in the assigned paper [1]. We will also discuss related work and results we obtained from our implementation.

The problem that by Su et.al [1] seeks to solve is to provide a deep-neural-network architecture that can be used for effective environmental sound classification. This is called intelligent sound recognition (ISR). A large amount of work is focused on sound recognition but primarily in the areas of music and speech classification. Due to how environmental, commonly "background" sound often has less clear identifying features and stationary characteristics solutions for the previous problems fail to achieve good accuracy on environmental sound problems. In the context of human beings, this can be recognised as the way in which people can characterise sound events. The provided dataset, used to determine the effectiveness of proposed models for environment sound classification (ESC), consists of audio files in which certain features are extracted. The goal of the proposed solution is to use two identical neural nets which are provided different sound features and then fuse the output of these two neural nets to classify the input. There are a number of existing solutions to the problem, an analysis of which follows in the next section of relevant work.

## II. Relevant work

The paper to replicate builds on the work of Karol J. Piczak on the usage of CNNs for environmental sound classification [2] - it uses the same feature extraction method from the raw data. As the first CNN approach this paper already achieved notable accuracy gains (5.6%) over more traditional methods. This paper uses a total of two convolutional layers with the ReLu activation function, the first having 80 filters of (57*6) size and (1*1) stride with a max-pooling of (4*3) size and (1*3) stride. The second layer also had 80 filters of (1*3) size and (1*1) stride with a max-pooling of (1*3) size and (1*3) stride. Two fully-connected layers were added, both of 5000 ReLu units then a final softmax output layer. It was also trained using a 0.5 dropout probability (but with other differing hyperparameters).

The authors of the UrbanSound dataset proposed a deep CNN architecture for recognizing sound as well, with the important addition of complex data augmentation techniques to overcome the scarcity of labeled data for training [3]. Previous work (including that by Piczak) reported that "simple augmentation techniques proved to be unsatisfactory for the UrbanSound8K dataset - given the considerable increase in training time they generated and negligible impact on model accuracy".

The proposed network architecture has three convolutional layers with two max-pooling operations between them, followed by a fully connected layer and a softmax output layer. As an input it uses the log-mel spectogram representation of audio. The first conv layer has 24 filters with a field of (5*5) and a (4*2) strided max pool. The second conv layer is the same except with 48 filters and the third is the same as the second but lacks a final pooling. All conv layers and the first fully connected layer (having 64 units) use ReLu (Rectified Linear Unit) activation functions. Training uses dropout with 0.5 probability.

This architecture does not differ significantly from the previous one, save for the addition of one more convolutional layer. Its more significant contributions are in effective data augmentation through 4 techniques:

1) Time stretching - speeding up or slowing down the sample by certain factors, leaving the pitch invariant
2) Pitch shifting - raising or lowering the pitch but leaving the duration unchanged
3) Dynamic range compression - compressing the dynamic range using parametrizations from the Dolby E standard and the icecast online radio streaming server
4) Background noise addition - mixing with a background noise sample from four differing origin acoustic scenes

While performing comparably on the original dataset to the best version of a shallow dictionary learning approach referenced in the paper, the CNN model shines in the augmented and expanded data set, gaining almost 6% more accuracy with the shallow model having negligible gains from the augmented set.

As an example of the dictionary techniques we would like to give mention to Selina Chu's work on environmental sound recognition [4]. The problem statement has a minor difference - it aims to recognize the environment the background sound comes from, rather than classify the sound itself, but essentially aims to accomplish a very similar task.

This earlier paper from 2009 recognizes the fact noted in the paper we are attempting to replicate, namely that the commonly used MFCC as audio features are poorly suited for environmental sound - due to the presence of noise, the importance of the temporal spectrum and the lack of predictable repetitions or harmonic structure compared to music or human speech. The paper conducts a review of the available features and points out how adding more is not always beneficial as it can lead to making the data set more sparse and some features could negatively and disproportionately impact the result despite being irrelevant in a specific case.

The Matching Pursuit algorithm is used to decompose the signal into a set of atoms using an overcomplete dictionary of functions. The remarkable part is that the algorithm is proven to converge to a solution with a zero-energy residual signal (i.e. the information is fully extracted) as long as the dictionary of atoms is overcomplete. Loosely described at each step the algorithm greedily selects the atom function that best correlates with the current residual signal. The used atoms are primarily Gabor dictionaries, sine-modulated, scaled and translated Gaussian functions. This model was able to distinguish environments in a pairwise decision scenario with over 0.9 accuracy in 70 of the 78 total pairs (13 distinct environments with full pairing).

## III. Dataset

The paper uses the UrbanSound8K dataset is freely available on the internet for research use, see [5]. It consists of over 8 thousand samples of sounds from an urban environment, each sample being at most 4 seconds long. The excerpts fit into 10 classes, fitting into the urban sound taxonomy described in the same paper as the dataset:

1) Air conditioner
2) Car horn
3) Children playing
4) Dog bark
5) Drilling
6) Engine idling
7) Gunshot
8) Jackhammer
9) Siren
10) Street music

The dataset was compiled from field samples uploaded to www.freesound.org. The 10 classes used for the dataset were selected for their prevalence in urban noise complaints with the exception of "children playing" and "gunshot", included to provide variety. The dataset was fully manually annotated by the authors, using some metadata from freesound.org but providing additional metadata in the labeling process such as whether the sound in question was in the "background" or "foreground".

The 8K dataset is drawn as a subset of the original dataset, which consists of a total of 18.5 hours of labeled audio with 3075 labeled occurrences. In a referenced paper the authors found that a 4 second sample was enough for subjects to identify environmental sounds with 82% accuracy, leading to a useful simplification of the problem with respect to sample duration/size and standardization of sample processing. These 4 second clips were extracted with a hop size 2s sliding window and a maximum of 1000 samples per class to prevent imbalances. These could have been the result of continuous clips of jackhammering (up to 30s) splitting into far more samples than short single occurrences like a gunshot. This 8K dataset is a total of 8.75 hours of audio.

## IV. Input

The model uses two different feature sets, each combined from four total auditory features. Three of these features are present in both datasets:

1) Chroma
   Typically used for analysis of music, chroma features describe the harmonic characteristics of sound. Generally divided into the 12 classes of $C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B$ a particular chroma is invariant to tone height - which octave the sound resides in.
2) Spectral contrast
   This feature highlights the differences in frequency strengths when a sound is translated into a power spectrum decomposition through the use of Fourier analysis. Musically, this would translate to the ability to tell the timbre and the pitch of a sound sample.
3) Tonnetz
   The Tonnetz (tone-network in German, from a work of Leonhard Euler) is a way of describing tonal relationships between sounds. Humans typically interpret sounds with primary frequencies having a rational ratio as pleasant. The Tonnetz data captures these relationships in the data, differing from the chroma in being relative as opposed to the division into octaves.

The two data features only present in one sets are as follows:

1) Log-mel spectogram
   A spectrogram is a way of visualizing a soundwave in 2D space - a log-mel spectrogram is a specific way of representing a spectrogram that presents auditory information in a way most useful to humans with respect to both visual clarity and the range of the data fitting that of human hearing. The y scale of the mel spectrogram is spaced in "mels", intervals perceived by listeners to be an equal distance from each other in pitch, mostly resembling the curve of the logarithm function.
2) Mel-frequency cepstrum coefficients
   MFCC for short are a result of mapping the power spectrum of an audio clip to the mel scale with the Fourier transform (mel scale described in more detail above). Individual coefficients from this are obtained by taking the logarithm of these mel frequency bands and interpreting them as a signal through a discrete cosine transform, measuring the final amplitudes of the produced spectrum.

The two data sets used by the paper are assembled from the first three and Log-mel spectrogram to form the LMC set as well as the first three and MFCC to form the MC set.

## V. Architecture

The paper's inconsistencies show as a difference between the stated convolutional layer structure and the stated number of parameters. The convolutional layers should not reduce naturally - the given memory usage details and table of parameters are evidence that the size of the layers decreasing

should only correlate with the size reduction of the max pooling. Considering how two pieces of evidence support this no reduction property (memory usage and parameter numbers) as opposed to the one diagram of layer structure the paper's results should be replicated based on them.

This necessitated a (1x1) padding and a reduction of the stride in the convolutional layers to (1x1), to maintain height and width across layers. There is however an additional issue with the architecture, by the listed parameter numbers the flattening of the network from the fourth convolutional layer to the fully connected hidden layer has 15.9 million parameters when in reality this operation requires no parameters. We believe that there is intended to be an additional fully connected layer at the end of the network. We added an additional max pooling operation after the fourth convolutional layer that cuts width and height in half, rounding up. This reduces to a shape of (11x22) with 64 kernels, giving 15488 parameters. That multiplied with the 1024 nodes of the fully connected layer yields a total of 15.85 million parameters, bringing us to the intended figures of the paper and presumably a good approximation of the process they followed.

So overall we have four convolutional layers, each with batch normalisation, and layers two and four have a dropout component and are followed by a max pooling operation. The fourth layer is flattened after the pool and is fully connected with 1024 output units, this layer also has batch normalisation and dropout. This then feeds into a final fully connected layer from 1024 units to our output layer of 10 units - one for each class. This final layer has no dropout or batch normalisation component.

When using the combined MLMC feature set this first fully connected layer had to be resized since the input to the model is then of size 41x145 which flattens to a 26048 linear layer which consequently has a higher parameter count of 26.7million parameters.

## VI. Implementation Details

The result was replicated with the architecture described in the section above, aligning with the concrete figures and the intentions of the original authors. Training was performed in a faster manner, using a single train/test fold instead of 10. The approach to result fusion is also different, using a simplified late fusion with softmax in place of the Dempster-Shafer Evidence information fusion.

We added a weight decay parameter to the Stocahastic Gradient Descent implementation of value 1e-4, and when trained with dropout set to 50% as described in the paper we trained the models over 100 epochs - chosen as we continued to see improvement in validation scores between 50 and 85 epochs so we wanted to ensure we had hit the validation limits.

### A. TSCNN

Unfortunately we ran out of time implementing the late fusion method for the TSCNN model. The model exists as displayed in the code, which runs a copy of MCnet and LMCnet through the validation set (with models loaded from previously created files), getting the logits from each. It then applies the softmax algorithm to get weighted values and then takes an argmax of the average between the two to get the prediction for the late fusion. This however doesn't work as intended in our final code, with the averaged logits losing the vast majority of the data, likely due to some incorrect softmax or argmax dimension in our code.

## VII. Replicating Quantitative Results

| Class | LMCnet | MCnet | MLMC |
|---|---|---|---|
| ac | 62.3 | 60.6 | 55.3 |
| ch | 87.2 | 72.0 | 81.7 |
| cp | 73.9 | 78.0 | 84.3 |
| db | 68.9 | 69.6 | 69.4 |
| dr | 71.5 | 77.6 | 73.2 |
| ei | 54.4 | 77.1 | 63.5 |
| gs | 95.1 | 90.2 | 91.2 |
| jh | 97.6 | 91.5 | 96.1 |
| si | 50.4 | 58.2 | 64.2 |
| sm | 68.7 | 73.7 | 81.9 |
| Avg | 73.0 | 74.8 | 76.1 |

## VIII. Training Curves

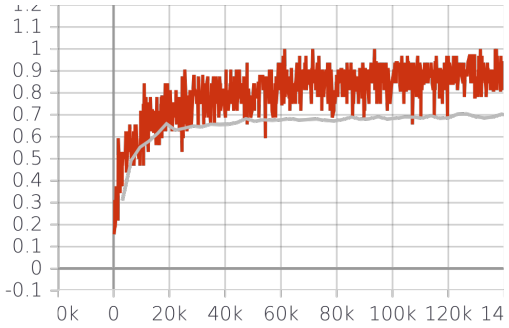


Fig. 1. LMCnet Train/Test Accuracy

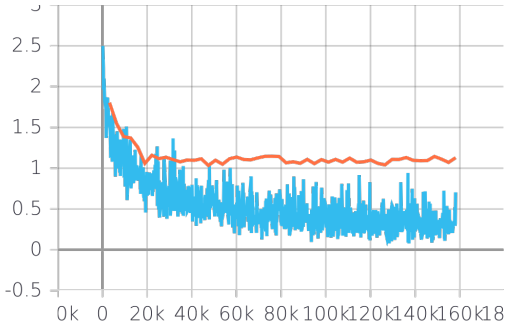


Fig. 2. LMCnet Train/Test Loss

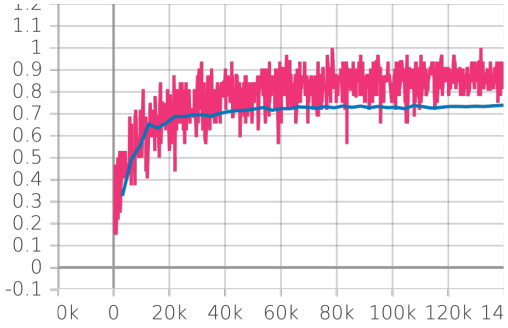


Fig. 3. MCnet Train/Test Accuracy

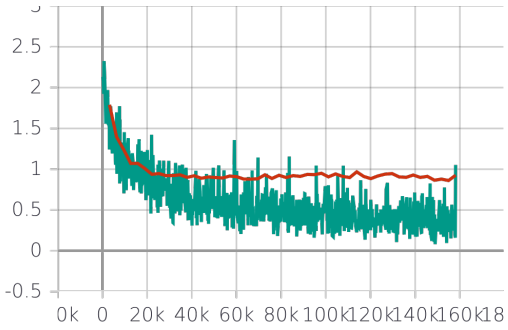


Fig. 4. MCnet Train/Test Loss

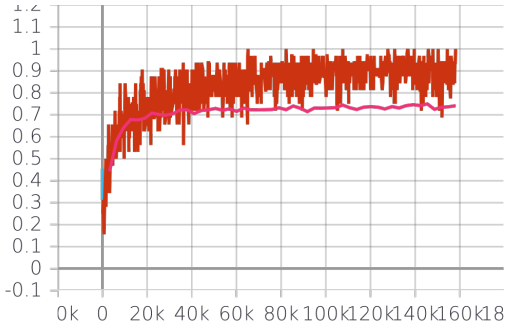


Fig. 5. MLMCnet Train/Test Accuracy

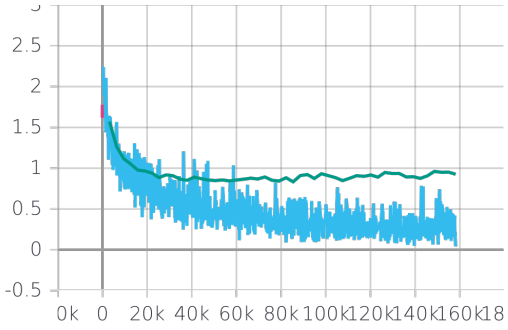


Fig. 6. MLMCnet Train/Test Loss

### A. Analysis

Looking at each of the above graphs, the loss and accuracy for each network plateaus fairly nicely indicating little overfitting if any. There is also an approximately 10% gap between the training accuracy and the test accuracy which potentially with some data augmentation or perhaps additional dropout on the final fully connected layer we may be able to convert to validation accuracy.

## IX. Improvements

We considered adding in a brightness transform to the dataset but ran out of time to implement it. We thought that the improvement should be dedicated to trying to improve the overlap between training performance and validation performance and hence should aid the model's generalisation. Since the input is in the form of several spectrograms, a random brightness jitter should be the most applicable noise to add since the data is primarily encoded in the brightness of the image already.

## X. Conclusion and Future Work

In this report we have performed a detailed review of the scientific paper given by the assignment. This included a review of other recent works with comparison to competing solutions. We provided an overview of the used dataset and an explanation of how said data was processed and how it can be used for a categorization task.

We replicated the neural network described by the authors in detail, highlighting the inconsistencies inherent in the paper. We made selections based on the original paper's architecture in an attempt to best recreate their results following in particular the reported numbers of parameters. Despite these choices made, the reproduced model has not managed to replicate the reported accuracies.

## References

[1] Y. Su, K. Zhang, J. Wang, and K. Madani, "Environment sound classification using a two-stream cnn based on decision-level fusion," *Sensors*, vol. 19, p. 1733, 04 2019.

[2] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, (Boston, MA, USA), pp. 1–6, Sept. 2015.

[3] J. P. B. Justin Salamon, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Singla Processing Letters*, vol. 24, pp. 279–283, 01 2017.

[4] C.-C. J. K. Selina Chu, Shrikanth Narayanan, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, pp. 1142–1158, 06 2009.

[5] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, (Orlando, FL, USA), pp. 1041–1044, Nov. 2014.