Environment Sound Classification Using a Two-Stream CNN

COMSM0018: Applied Deep Learning

Mark Ergus Nicholl
mn16660@bristol.ac.uk

Faizaan Sakib
ss16161@bristol.ac.uk

I. Introduction

The task of identifying the class of an environmental sound, such as the noise of a siren, is accordingly known as environmental sound classification (ESC). This task falls beyond the scope of solutions produced for other domains of sound such as speech or music. This is due to the difference in characteristics found in environmental sound, where there tends to be a lack of structure and consistency of length.

Su, Zhang et al. [7] recently proposed a solution to address this problem, involving a two-stream CNN using a combination of auditory features to train itself.

This report aims to document the implementation of, specifically, their 4-layer CNN. This will be followed by presentation of the results obtained from the replicated network, its evaluation, before finally attempting to make an improvement to the network.

II. Related Work

There has undoubtedly been a rise of deep learning in recent years within various domains, due to its superior performance compared to other learning methods. It is no different for sound classification, spawning a prevalence of CNNs similar to the one seen in Su et al. [7] to tackle the task of ESC.

Amongst the first of its kind was Piczak's [4] implementation, involving a shallow two-layer CNN with max-pooling, followed by two fully connected layers. The network was trained and tested on three datasets, including UrbanSound8K [6] which is also used by Su et al. [7]. The conclusion was that CNNs offered better performance than manually engineered features used previously. This naturally ushered in a motivation to further explore the effectiveness of CNNs in ESC, with Piczak's implementation being used as a baseline benchmark to this day.

Salamon and Bello [5] proposed the use of a 3-layer CNN, but mainly, a comprehensive set of audio augmentation techniques including: time stretching, pitch shifting, dynamic range compression and mixed in background noise. Although the CNN performs only as well as Piczak's [4] implementation on its own, it performs significantly better with the data augmentation. Furthermore, the varied effectiveness of augmentation for each class prompts further investigation into class-wise augmentation.

More recently in 2019, Li et al. [2] outlined a three-stream deep CNN using a late fusion method. Each stream takes a different type of input: the raw audio waveform, short-term Fourier transform (STFT) coefficients, and the delta spectrogram. A novel attention function is integrated into the network, which allows it to take into consideration the fine-grained temporal structures in audio. Thus, the network can identify dynamic changes in energy, enabling it to overcome silences and background noise in audio clips. The network achieves state-of-the-art performance in a number of datasets with its unique appoach.

From the works referenced above, it can be seen that CNN-based ESC systems still require further exploration. For instance, there is a variety of audio-based features and network architecture characteristics (e.g. number of streams) to consider. Meanwhile, the need to encode fine-grained temporality [2] has only recently emerged. As such, Su, Zhang et al. [7] aimed to make some headway in ESC with regards to the use of combined feature sets and decision-level fusion of two streams.

III. Dataset

The UrbanSound8K [6] dataset will be used to train and test the network. It is made up of WAV-format audio clips consisting of "urban sounds", taken from field recordings. Each of the clips are up to 4 seconds long, totalling up to 9.7 hours of audio. These clips are then split further into multiple segments which are used as samples for the network.

Su et al. [7] use a ten-fold split across the dataset to perform cross-validation. However, for the purposes of this replication, a singular train and test fold will be used. The training split consists of 50,560 segments obtained from 7891 audio clips. Meanwhile, the smaller testing split consists of 5,376 segments obtained from 837 clips.

Every audio clip/segment is labelled as one of ten 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). These classes have been drawn in accordance to the "Urban Sound Taxonomy" [6], which aims to achieve unambiguity and relevancy in urban sound categorisation.

IV. Input

As stated in Section II, one of the main objectives presented by Su et al. [7] is to utilise feature combination to achieve superior results. Within sound classification, the "log-mel spectrogram" and "mel frequency cepstral coefficent" (MFCC) are two of the most used auditory features. These two features form the basis of the two main feature combined inputs, called LMC and MC respectively.

Next, we consider three other auditory features. These include a sample's "chroma", "spectral contrast" and "tonnetz". To obtain the final combined input, these features are

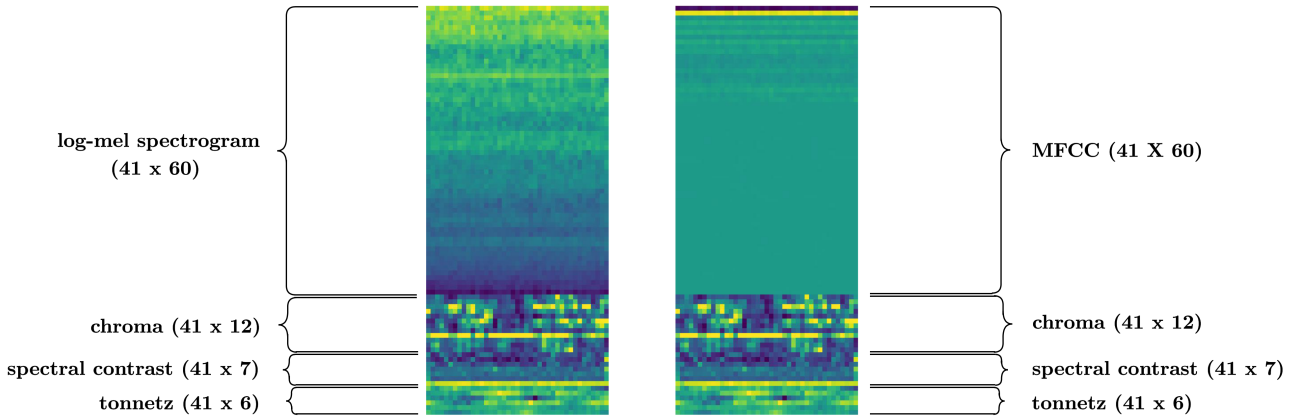

Fig. 1: A visualisation of the spectrograms produced by the LMC and MC inputs for a particular audio segment, including the breakdown of their features.

appended to the log-mel spectrogram to obtain the LMC. Otherwise, they are appended to the MFCC to obtain the MC input. The networks are accordingly named "LMCNet" and "MCNet", based on the type of input they are given.

Every feature is two-dimensional with a constant width of 41. The log-mel spectrogram and MFCC are both made up of 60 channels, giving them a size of 41 x 60. Meanwhile, the other three features combined have a size of 41 x 25. This means the final size of the input is 41 x 85. Furthermore, to ensure consistency across input types, the order of the features is maintained.

V. Architecture

The architecture of the network was replicated as found in the work presented by Su et al. [7]. An illustration of the layers and their properties, including dimensions and the number of channels, can be seen in Figure 2.

Although a replication was attempted, there were a couple of inconsistencies found in the referenced work. This led to some design choices that had to be made in order for the network to give sufficient results.

A. Stride

The first inconsistency is the use of a stride of 2 x 2 for every convolutional layer in the network for LMCNet and MCNet. The use of such a stride would downsample the input to each layer, roughly halving the dimension sizes. This is in direct contrast to the architecture diagram and experiment discussion seen in later parts of the paper.

Outside of the network descriptions given, Su et al. [7] state in the same paper, that the first and second layer have an input size of 41 x 85, followed by a 2×2 max-pool to obtain a feature map of size 21×43 used for the third and fourth convolutional layer. Finally, a feature map of size 11×22 is "derived" from the fourth and final convolutional layer.

As such, if a stride of 2×2 were to be applied to all four convolutional layers, including the max-pool required after the second layer, the size of the feature map would be $\approx (41 \times 85)/2^5$ which would be much smaller than the 11×22 feature map required before the first fully connected layer.

Due to the lack of declaration of how the size of the feature map is maintained, we resolve this inconsistency as such: the network is max-pooled after the second layer as originally required, and a stride of 2×2 is only used after the fourth layer, so that a feature map of size 11×22 is obtained in preparation for the fully connected layer.

It is important to note that preceding our final approach, we attempted to maintain the size of the feature map using two different methods. Firstly, by upsampling to the required size. Secondly, by applying the appropriate amount of padding (which would tend to be quite large). Both of these methods, used independently, actually caused a significant decrease in performance.

B. Fully Connected Layers

Another irregularity found involved the 10-value vector output from the network. Although the network mentions the use of one fully connected layer after the fourth convolutional layer, this is in fact not sufficient to get from a size of 15488 units $(11 \times 22 \times 64$ channels) to 1024 and then to 10 units.

Instead, two fully connected layers that are used. The first layer takes an input of 15488 from the fourth convolutional layer, outputting a vector of size 1024, before applying the Sigmoid activation function as stated [7]. Then, a second fully connected layer which takes an input of 1024, produces the output of size 10. Finally, softmax is applied to this vector as part of the loss function, to obtain the class-wise predictions for the sample.

VI. Implementation

For the purposes of this report, only the 4-layer CNN will be replicated. Meanwhile, the two-stream implementation will follow a method different to the one produced by Su et al [7].

A. Dataloading

As described in Section IV, the inputs LMC and MC are each made up of several features combined together. The creation of the inputs occur before training the network. The feature map is created according to the type of input requested, picking the required features and concatenating them together. It is important ensure that the order of the concatenation is maintained and the input has a final size of 41×85.

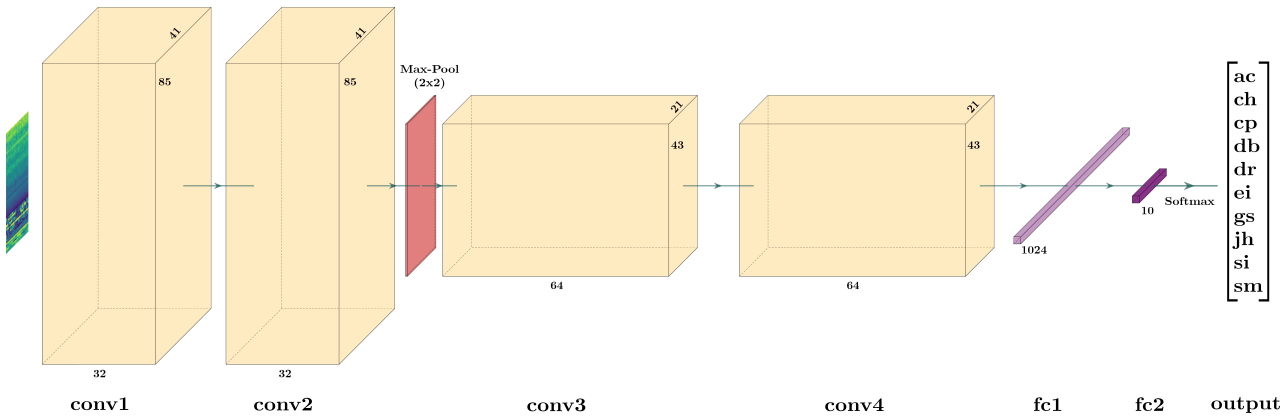

Fig. 2: A diagram of the network architecture used to classify LMC, MC and MLMC inputs.

B. LMCNet and MCNet

1) Architecture & Training: Apart from the decisions made in Section V, all other aspects of the network are replicated as found in the paper [7].

There are four convolutional layers that are defined, where there are 32 convolutional kernels for the first two layers and 64 kernels for the last two layers. Each layer uses a receptive field of size 3×3. The choice of stride as discussed Section V-A, and the kernel size used, means a zero padding of size 1 is sufficient to ensure no change in size to the input after convolution. Each layer is followed by batch normalisation and then the activation function ReLU. These layers are followed by two fully connected layers as described in Section V-B.

It is important to note that bias values within the convolutional filters are redundant, as they are already included due to batch normalisation. Therefore, the bias parameters within the layers can be excluded. This gives the network the same number of parameters across all layers as listed by Su et al. [7].

Max-pooling of size 2×2 with stride 2×2 is applied after the second layer. However, by default, the max-pool operation would downsample to the floor, producing an output of 20×41 which would be too small for the third layer. To rectify this, the max-pool layer was specified to round to the ceiling so that the output would have the correct dimensions. Meanwhile, a dropout probability of 0.5 is applied to the second and fourth convolution layers, and the first fully connected layer. Softmax cross-entropy is applied as the loss function.

The Adam optimiser is used to train the network as specified by Su et al. [7] with β_1, equivalent to momentum, set to 0.9. However, they do not mention the value of the L_2 regularisation used. In this network, we have chosen to use a value of 0.0005, which has proved to perform best for our network. The L2 regularisation is added as a 'weight decay' parameter to the Adam optimiser, as instructed according to Pytorch's documentation. It is important to note, however, that the weight decay and L2 regularisation term, although mathematically equivalent for stochastic gradient descent(SGD) optimisers, is not equivalent when applied on adaptive optimisation models such as Adam [3].

The batch size is 32, the learning rate is set to 0.001, and the number of epochs the network is trained for is 50.

2) Validation: Recall from Section III, that the audio clips found in the dataset are split into segments for use by the network. During training, these segments are treated as independent samples. However, in testing, the performance of the network is measured against the audio clips.

This is achieved during testing, by collecting all of the sets of logits acquired for each segment and attributing them to the clip they have come from. Then, the logits across all of the segments for each file are averaged class-wise to obtain a 10 value vector which holds the final prediction values for each class. The class with the maximum value is chosen to be the predicted class for the audio clip.

The average class-wise accuracy (mean of all class accuracies) is used as the main metric of a model's performance.

C. MLMCNet

A third input type, named "MLMC" is also considered. This input is simply the feature set of MFCC, log-mel spectrogram, chroma, spectral contrast and tonnetz combined in that order.

With the addition of an extra feature, the input is now larger with a size of 41×145. Thus, the size of the output from the 4th convolutional layer will now also be larger, specifically 11×37. Therefore, the input size of the first fully connected layer has to be changed from 15488 to 26048 ($11 \times 37 \times 64$ channels), while the output remains the same.

D. TSCNN

For the two stream network (TSCNN), a simpler late fusion method is used compared to the one seen in Su et al. [7]. LMCNet and MCNet are independently trained and then saved to disk. Then, to obtain predictions, the softmax-applied logits of each network are averaged class-wise, where the highest value indicates the class predicted.

VII. Quantitative Results

The results achieved by the replicated network for the inputs LMC, MC, MLMC, and the two-stream method TSCNN, are shown in Table II.

They seem to indicate that with MC as the input, the network performs least favourably. Meanwhile, TSCNN, fusing LMCNet and MCNet, produces the best-performing average class-wise accuracy of 81.8%. Amongst the classes found in the dataset, the network performs consistently poor on sounds classed as "siren", with an average accuracy of

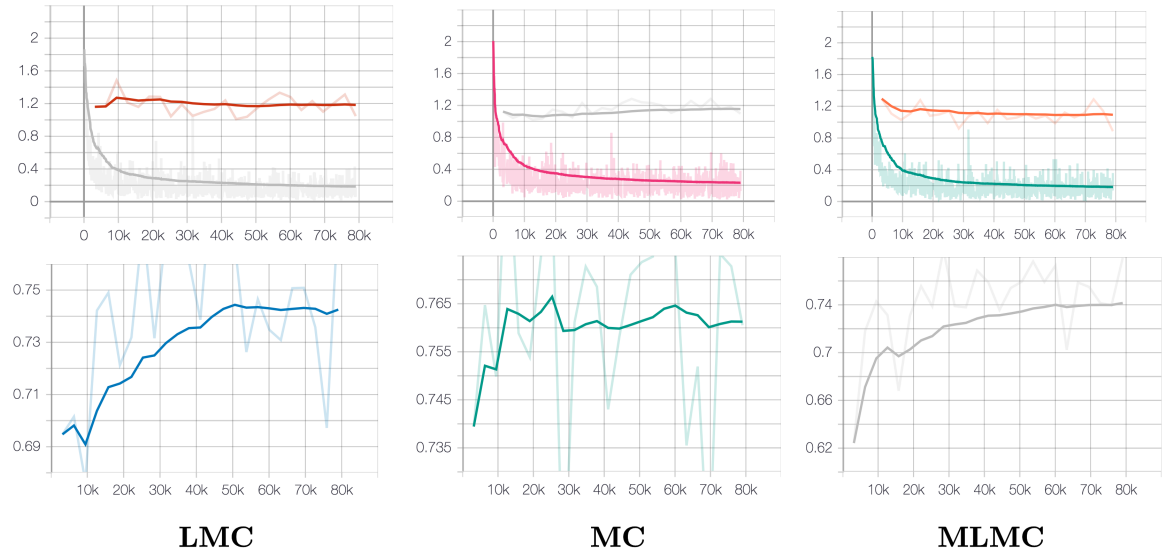

Fig. 3: A grid of plots showing the curves for the training/test loss (top row) and the average class-wise accuracy (bottom row) for the network across inputs LMC, MC and MLMC. Smoothing of 1.0 is applied.

Class	LMC	MC	MLMC	TSCNN
air conditioner	63.0%	70.0%	57.0%	73.0%
car horn	94.0%	42.4%	87.9%	84.8%
children playing	78.0%	86.0%	87.0%	86.0%
dog bark	77.0%	77.0%	74.0%	76.0%
drilling	86.0%	70.0%	90.0%	87.0%
engine idling	59.1%	76.3%	64.5%	72.0%
gunshot	87.5%	96.9%	93.8%	93.8%
jackhammer	100%	91.7%	92.7%	99.0%
siren	53.0%	54.2%	49.4%	56.7%
street music	83.0%	96.0%	83.0%	90.0%
Average	78.1%	76.1%	77.9%	81.8%

TABLE I: Class-wise accuracies of four models based on a 4-layer CNN evaluated on UrbanSound8K.

53.3% across all four models. Meanwhile, the network does best for "gunshot" and "jackhammer" sounds with across model average accuracies of 92.9% and 95.6% respectively.

VIII. Training Curves

Figure 3 show the curves for the training loss, test loss and the average class-wise accuracy over the 50 epochs of training.

The general trend is that the network is prone to overfitting for all three input types. This can be said due to the test loss curve being consistently higher than the training loss curve. Furthermore, the difference between the two curves is significant, sometimes reaching up to 1.0. This seems to suggest that even with a majority of correct predictions, the network is less certain about the predictions it is making. Also, there is not much improvement seen for the test loss, as there is in training loss. This may point the fact that the current use of dropout is ineffective.

The average class-wise accuracy curves have an overall pattern of dramatic increase to start off with, when the rate of change plateauing near the end of the 50 epochs. This is expected, as a model learns the most near the start of its training. However, the non-smooth curves show that there is a high level of volatility in the performance of the network. A reason for this may be a sub-optimal learning rate which is too large, overshooting past the local minima causing it to oscillate.

IX. Qualitative Results

Figure IX shows four different scenarios found in the performance of the network on the test split.

a): Both LMCNet and MCNet successfully predict the class of the audio clip, "jackhammer". This class has the the highest accuracy overall, as there is plenty of variation and contrast spread throughout each of the features, even in MFCC, which tends to have less variation. This leads to more features that the network can learn well.

b): Here, LMCNet correctly predicts the clip to be of a "dog bark", while MC predicts it to be of "street music". The chroma feature represents the pitch and tonal content of a sound. [1]. In this case, the chroma feature seems to be notably characteristic. As such, the confusion may have been caused by samples of street music resembling similar tonal qualities. Furthermore, "harmonics" tend to be more commonly found in voices and instruments, which the model struggles to choose between here.

c): This clip is only correctly classed by TSCNN as a "siren", whereas LMC classes it as "engine idling" and MC as "drilling". The incorrect predicted classes are interesting to see. An analysis of the UrbanSound8K dataset [6] shows that audio clips of sirens originate from the presence of motorised transport, and tend to have the most interference from background sound. Therefore, it makes sense the models' predictions could legitimately resemble the background interference found within these clips. The reduction in uncertainty with the fusion helps here.

d): Finally, this clip of class "engine idling" is incorrectly classed by all networks. MCNet predicts it to be of class "air conditioner". The similar nature of both sounds may have caused this incorrect prediction. However, LMCNet and causally TSCNN, predicted it to be of class "siren". On Figure 5 we can see the input is very similar to the siren clip (**c**). These incorrect predictions, along with the findings from **c**, helps us to argue that sounds need to be distinguished from background interference. If not, cases such as these will hinder a network's performance on ESC.

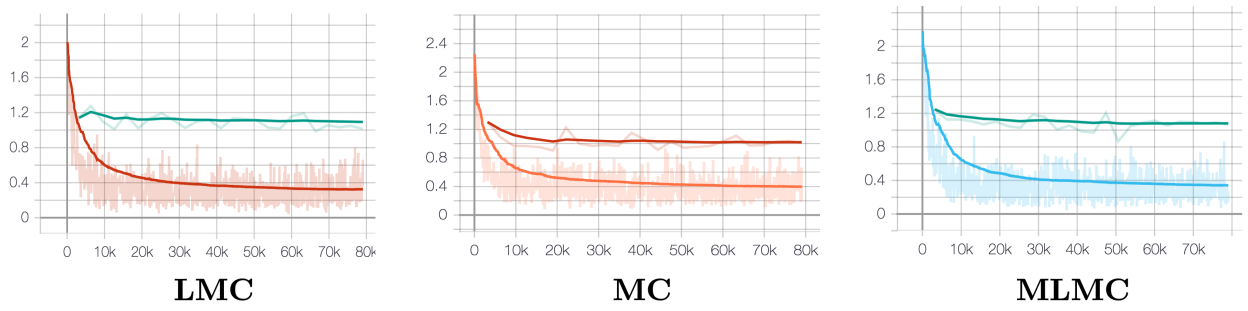

Fig. 4: Training/test loss curves produced by the improved network for LMC, MC and MLMC. Smoothing of 1.0 is applied.

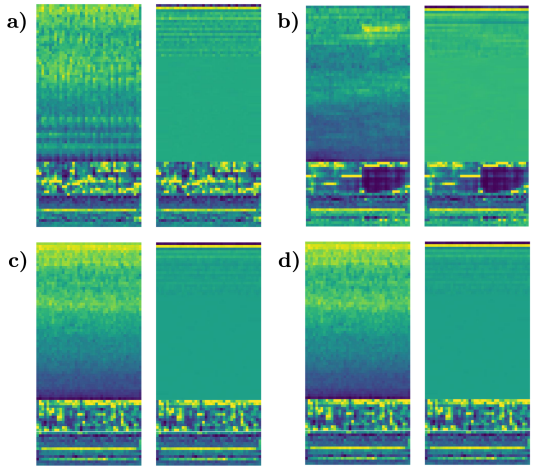

Fig. 5: Visualisations of samples (LMC left, MC right) for four different success/failure cases produced by the network.

Class	LMC	MC	MLMC	TSCNN
air conditioner	52.0%	84.0%	48.0%	86.0%
car horn	93.9%	75.8%	90.9%	87.9%
children playing	69.0%	67.0%	76.0%	74.0%
dog bark	79.0%	58.0%	79.0%	74.0%
drilling	89.0%	82.0%	83.0%	84.0%
engine idling	67.7%	80.6%	48.4%	75.3%
gunshot	100.0%	90.6%	93.8%	96.9%
jackhammer	99.0%	86.5%	94.8%	96.9%
siren	51.8%	49.4%	51.8%	55.4%
street music	76.0%	96.0%	86.0%	92.0%
Average	77.7%	77.0%	75.2%	82.2%

TABLE II: Class-wise accuracies after adding dropout layers to all convolutional and fully connected layers

X. Improvements

In order to implement improvements to the reproduced network, the loss graphs from Figure 3 are analysed. Stated in Section VIII, since the validation loss is significantly higher than the training loss, the model overfits the training data. As dropout is only implemented on the second, fourth, and first fully connected layer, the decision was made to add dropout layers to the third layer and the second fully connected layer with the hypothesis that this would increase regularisation and reduce overfitting on the training data.

In terms of average class wise accuracy after 50 epochs alone, this method produced mixed results, producing a marginal increase for the MC and TSCNN models while for the LMC and MLMC models, there was a marginal decrease. Despite these mixed results, the changes in the test loss curves for the LMC and MC models are noticeable. As shown in Figure 4, when smoothed, the average test loss is lower for both the LMC and MC model. In the case of the MLMC model, it remains relatively unchanged. As the cost function and training set are unchanged, this signifies that the model could be overfitting less and generalising better over the test data. This could mean that given more test data, it can perform better than the baseline implementation.

XI. Conclusion

In this paper, we have attempted to reproduce a CNN according to a published research paper [7]. In this process, we identified inconsistencies and in order to construct a working neural network, we had to make conscious design decisions to ensure that the network performs well to replicate the reported results while keeping to the original architecture as much as possible. Furthermore, we analysed the performance of the CNN to find an area where improvements could be made. We concluded that the network created through our reproduction was overfitting the test samples and hence, we attempted to improve the network by adding more dropout layers to the architecture. Finally, we gathered data from the results to discuss and analyse them critically.

A. Future Work

Throughout this process, we have stumbled upon literature that signifies how most implementations of weight decay in Adam optimisers are dysfunctional and do not regularise the weights well [3]. As a result, newer AdamW optimisers that decouple the weight decay has been shown to regularise the weights in a network better and could contribute to an increase in test accuracy. We would be interested in comparing the results of using the AdamW optimiser with adjusted hyperparameters on this architecture and dataset to compare the difference in performance.

References

[1] Kattel, M., Nepal, A., Shah, A., and Shrestha, D. Chroma feature extraction.

[2] Li, X., Chebbiyam, V., and Kirchhoff, K. Multi-stream network with temporal attention for environmental sound classification. *Amazon AI* (01 2019).

[3] Loshchilov, I., and Hutter, F. Fixing weight decay regularization in adam. *CoRR abs/1711.05101* (2017).

[4] Piczak, K. Environmental sound classification with convolutional neural networks. *IEEE* (2015).

[5] Salamon, J., and Blanes, Pablo Bello, J. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing* (03 2017).

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

[7] Su, Y., Zhang, Wang, J., and Madani, K. Environment sound classification using a two-stream cnn based on decision-level fusion. *Sensors* (2019).