Replicating Published Work on Sound Classification Using Fused Acoustic Features and Stacked CNN

Kheeran Naidu
Department of Computer Science
University of Bristol
kn16063@bristol.ac.uk

Adam Pluck
Department of Computer Science
University of Bristol
ap16894@bristol.ac.uk

Farrel Zulkarnaen
Department of Engineering Mathematics
University of Bristol
fz16336@bristol.ac.uk

***Abstract*—Environmental Sound Classification (ESC) is a major task in the growing field of Intelligent Sound Recognition technology. The acoustic features of environmental sounds are notably different to those of speech or music, which is well-studied in the context of speech or music Recognition. These differences pose two main challenges associated to ESC: (1) composing an appropriate input feature set; and (2) developing a relevant and well-performing model to process the features. So far, existing models are still unsatisfactory; however, due to advent of deep learning models, some progress has been made. A notable example is the published work of Su et al. 2019, which claims the current state of the art solution. In this report, we will attempt to replicate such results using the proposed architecture, and see if further improvements can still be made.**

I. Introduction

The work done on '*Environmental Sound Classification (ESC) Using Two-streamed CNN with Decision Level Fusion*' [1] by Su et al., is an attempt to produce a state of the art model for solving ESC tasks, more specifically, classifying sounds commonly heard in background noises. One of the challenges of ESC is that commonly known acoustic features, such as those typically used in speech or music recognition, are insufficient representation of environmental sounds, that is, they are typically either non-stationary or have higher temporal variance. Another known issue is the unsatisfactory history of classifiers for ESC. Existing models are either lacking in temporal invariance or failing to reach high accuracy predictions.

Therefore, to solve both of these problems, Su et al. claim that (1) a combination of acoustic features used in speech or music recognition via late fusion methods would make a suitable feature set for ESC; and (2) a stacked neural network of two-streamed CNNs (with late-fusion during testing) can outperform other models of the time on known datasets such as the *UrbanSound8K*.

II. Related Work

Within the domain of sound classification, and more specifically environmental sound, a wide scope of methods have been explored, ranging from complex to simplistic.

Piczak et al. [2] first attempted to capitalise on the success of convolutional neural networks in other domains. Their architecture consisted of two convolutional layers each with max-pooling leading into two fully connected layers. All bar the 2nd convolutional layer were equipped with dropout. They achieved an accuracy of 73.1% (LP) when trained and evaluated on a common dataset for this problem (*UrbanSound8K*). This success set off a flurry of CNN related methods, with the majority focusing on features. Similarly used by Su et al., Meyer et al. [3] successfully incorporated the mel-spectrogram (log for Su) as a feature of the CNN. This was done by using mel-spectrogram as the transformer of the raw audio signal into a time-frequency representation (referred to as the "front-end").

Other methods took a different approach, attempting to negate the ill-effects of CNNs arising in ESC setting. Chen at al. [4] saw the negative effects of information loss as a result of max-pooling and so implemented dilated convolution: dilating or contracting convolutions (increasing/decreasing the receptive field) for the higher layers. This allows for an equal amount of a parameters compared to a conventional model, but with much lower levels of information loss. The success of this is clear. A dilation based model achieved an accuracy of 78%, beating Piczak et al.'s model by a good margin. This model was also trained and evaluated on (*UrbanSound8K*) which will now be explained.

III. Dataset

The dataset used by Su et al is the *UrbanSound8K* dataset which consist of 10 classes of environmental sounds. The data set is comprised of 8732 audio clips, each labelled as one of the 10 following 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). In our adaptation of this we divide the clips into multiple parts, treating each part as an independent training sample. We note that the class split of the dataset is uneven and has the proportions 7:2:7:6:6:7:1:6:6:7 in order of the listed classes. For the validation aspect of the data set, we test on each part and then recombine to assess the final score and ultimately final classification. The overall proportion of test samples was just below 10% (50569:5395).

To neatly organise the divisions and other key features, the structure of the data set is simply a list of dictionaries with multiple keys (filename, class, classId, features). By making use of a dataloader we un-pickled (de-serialized) the data,

ready to be modified into the two distinct inputs for each "stream" of the CNN.

IV. Input

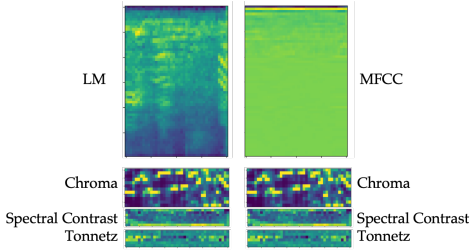


Fig. 1: Both inputs for a single data point: LMC (left) and MFCC (right)

Although it was stated that acoustic features used for speech or music recognition are insufficient for ESC, Su et al states that the fusion of several said features could give a more complete representation of environmental sounds. Therefore, two acoustic features originally used for speech recognition, which has been proven to work well for ESC, are Log-Mel spectrogram (LM) and Mel Frequency Cepstral Coefficients (MFCC). LM is a representation of frequencies with the mel-scale (a logarithmic transformation) applied, whereas MFCC is a representation of the envelope of short term power spectrum of the audioclips - typically used to represent phonemes of speech ref.

Both LM and MFCC will be concatenated separately with three other acoustic features called Chroma, Spectral Contrast and Tonnetz (abbreviated as CST). Together, LM and CST will make up the LMC input training set, whilst MFCC and CST will then make the MFC input training set. This is better illustrated in Figure 1. Both of these training sets give rise to the two-stream nature of the proposed architecture. Each stream would then be a complete network but with different training sets. At the end of learning process, the output values given of the softmax layer of both stream will be combine using late-fusion method to give the final classification results.

V. Architecture (Su et Al)

The main architecture of the model proposed by Su et al is a stacked CNN of two stream identical network with four convolution layers, and two fully connected layers with a softmax layer at the end, as seen in Figure 2.

However, there are some inconsistencies in the details of the architecture written in the paper; namely there are misspecified values of parameters, and inconsistent dimension reports.

A. Inconsistencies/errors:

It was found that when detailing the filter dimensions in the model, the authors have left some ambiguity, leaving readers to make certain assumptions. Specifically, the number of paddings for each filter was never specified, so in order to achieve the specified layer dimensions either the stride value stated in the paper is incorrect, or readers must assume the correct padding values on their own.

To elaborate, for a matrix of size $n \times m$, let the output size after applying a kernel/filter of size $f_n \times f_m$ be $n' \times m'$, which can be found by the following set of equations:

$$n' = \lfloor \frac{n + 2p_n - f_n}{s_n} + 1 \rfloor, \qquad m' = \lfloor \frac{m + 2p_m - f_m}{s_m} + 1 \rfloor$$

where $p_n \times p_m$ are the number of padding for each filter dimension, $s_n \times s_m$ are the stride values, and $\lfloor . \rfloor$ denotes the floor function.

Using the equations above we can deduce the correct values for padding and the stride to achieve the specified output dimension in the paper. In the paper it was stated that they use convolution filters of size 3×3 with stride 2×2; however, these will not lead to the specified output shape corresponding to the next layer's. Therefore, to revise this mistake we believe that a padding of 1×1 with stride 1×1 would be the correct parameter values. Or alternatively if we believe that the stride value is correctly stated as 2×2, then the required padding value would be 21×43.

Furthermore, note that the inconsistencies mentioned above are only those for the convolution filters. The dimension details of the pooling filter were correct; a max-pooling filter of size 2×2 with strides 2×2 would indeed reduce the dimension of the layers by half (transition between Conv2 to Conv3), that is provided that the padding of the pooling filter is 1×1.

B. Individual models

As a form of experiment validity, Su et al proposes four variations of their architecture to compare and contrast performance results. As explained previously, there are two main separate features set: LMC and MC. Each will be fed separately into an identical CNN, making up a single stream. The stream trained with the LMC feature set is named LMCNet, whilst the stream trained with just MC is called MCNet.

Furthermore, the main architecture of the paper would be denoted as TCSNNNet; in which both the LMCNet and MCNet streams were trained independently, and their predictions combined during testing i.e. late-fusion. Additionally, to confirm late-fusion effectiveness, another variation of the architecture, denoted as MLMCNet, is created where all five features are combined linearly into one feature set named MLMC, and then passed into a CNN identical to either of the streams.

The crux of the authors' claim is that predictions made with late-fusion (TCSNNNet) would outperform other models such as MLMCNet, LMCNet, MCNet.

VI. Implementation Details

The implementation of the replicated results was based on the skeleton code provided for the course lab work.

The implementation uses 3 main classes; the CNN class, the Trainer class and the *UrbanSound8K* dataset class. The CNN class defines the architecture described in section V. The Trainer class contains the actions taken at each of the epochs

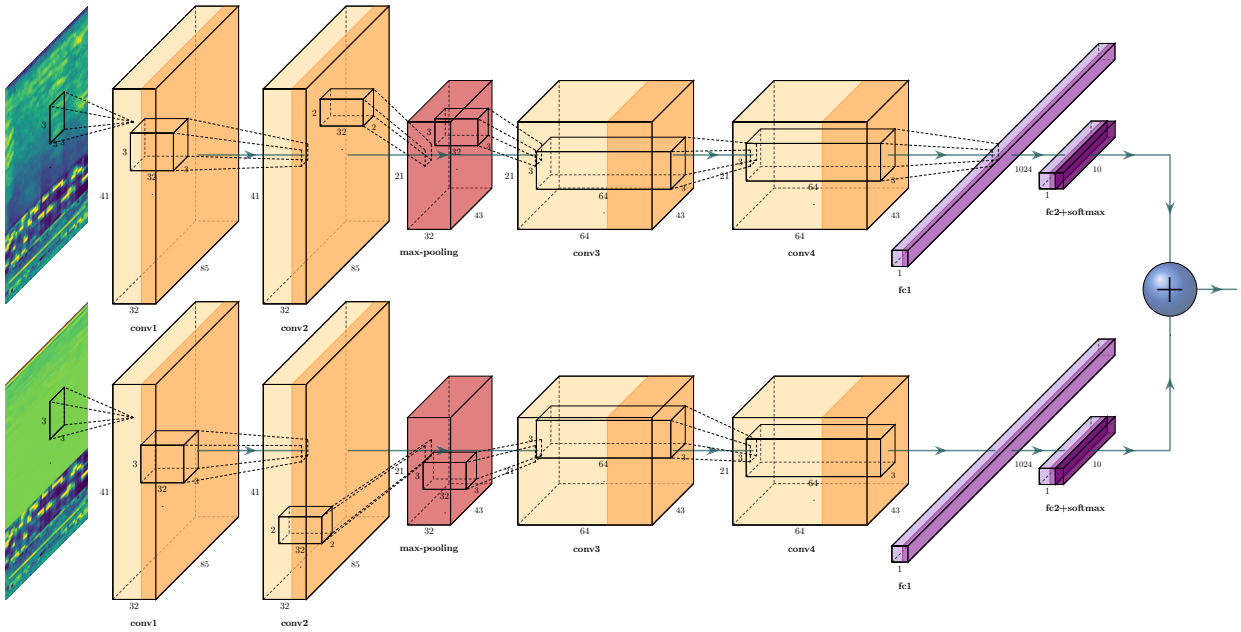


Fig. 2: TSCNNNet architecture proposed by Su et al. Top stream is LMCNet, bottom stream is MCNet. Yellow layers indicate convolution layers, whilst purple ones are fully-connected networks. The red layer represents a pooling layer after max-pooling, to add clarity to the confusion in the paper. Filters of size $3 \times 3 \times d$ are convolution filters, whilst $2 \times 2 \times d$ are pooling filters, and $d \in \{32, 64\}$ is the hyperparameter indicating the number of filters used in that specific layer.

associated with training. Finally, the *UrbanSound8K* dataset class provides the data in a standard format for PyTorch's DataLoader. Based on the format of the data, described in section III, we used PyTorch's 2D Dropout, BatchNorm, MaxPool and Conv functions, with all weights initialised using Kaiming, to define the neural network.

The output logits of the CNN were further processed specifically for evaluation. Each sample in a batch was a segment from a full audio file of a particular environment sound being classified, called its label. For each batch before classification, the logits of each sample from the same audio file was averaged and the new averaged logits with their new respective labels were used for evaluation.

Replicating the results required careful fine tuning of the hyperparameters not defined in the given paper. These included the weight decay associated with the $L2$ regularisation, catering for the unbalanced class distribution of the data set and the type of variant of the SGD optimiser. The final replication results used the SGD optimiser with $L2$ regularisation and a weight decay of 10^{-2}.

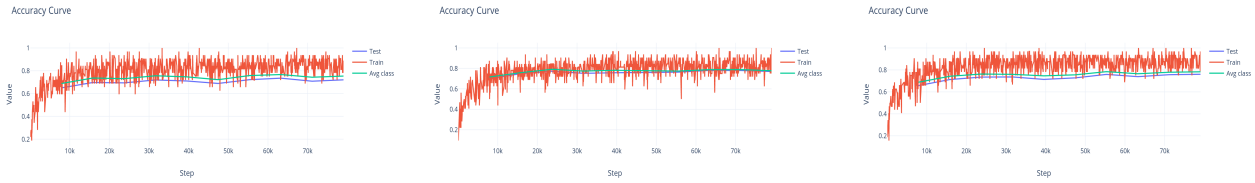
The full code used is provided with this report and is fully commented for ease of understanding of what has been done. When running the code, the following file arguments are available as adjustable hyperparameters; *--epochs (int), --batch-size (int), --dropout (float), --optimiser (str), --momentum (float), --weight-decay (float), --learning-rate (float), --mode (str) and --TSCNN*. The optimisers available are Stochastic Gradient Decent (*'SGD'*), Adaptive Learning Rate Optimisation (*'Adam'*) or Adam with Decoupled Weight Decay Regularisation (*'AdamW'*). The type of network used, either LMCNet (*'LMC'*), MCNet (*'MC'*) or MLMCNet (*'MLMC'*), is determined using *--mode* however using *--TSCNN* overwrites this for TSCNNNet and loads previously saved LMCNet and MCNet results. The remaining arguments are self-explanatory.

VII. Replicating Quantitative Results

Class	LMC (LMCNet)	MC (MCNet)	MLMC	TSCNN
ac	48.3	78.0	55.1	56.8
ch	97.4	71.1	92.1	94.7
cp	74.4	84.6	94.0	82.1
db	68.1	54.0	71.7	71.4
dr	75.4	80.5	81.4	78.0
ei	60.5	75.2	65.1	73.4
gs	100.0	100.0	97.1	100.0
jh	99.1	95.6	94.7	97.3
si	50.0	54.1	57.1	58.2
sm	80.5	84.7	74.6	78.8
Avg.	**75.4**	**77.8**	**78.3**	**79.1**

Table I: Replicated class-wise accuracy of four models with four-layer CNN evaluated on our split of *UrbanSound8K*.

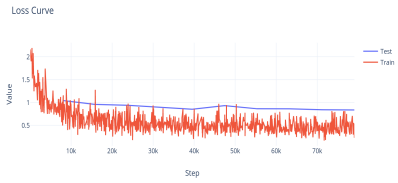
Table I shows our results of replicating the work of Su et al, using an amendment of their architecture, on the *UrbanSound8k* dataset. We managed to achieve accuracy results that were within the 5% range of the stated average accuracy for MCNet and MLMCNet, whereas our results for LMCNet and TSCNNNet sits in the 10% range of the (coursework) reported average. Nonetheless, TSCNNNet outperforms the other networks as presented by Su et al.



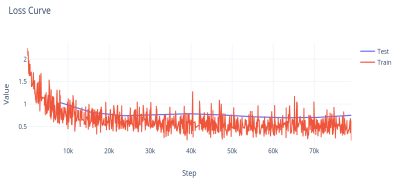
(a) LMC accuracy

(b) MC accuracy

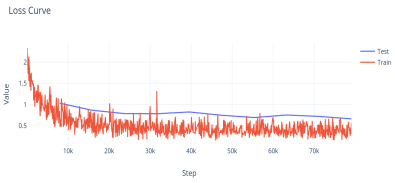
(c) MLMC accuracy



(d) LMC loss



(e) MC loss



(f) MLMC loss

Fig. 3: Training curves of the three networks without fusion

VIII. Training Curves

To further elaborate the performance of our replication result, below are the learning curves of the four networks as referenced in Table I.

A. LMC

Figure 3a and Figure 3e shows the accuracy and loss curves of the LMCNet stream respectively. Overall, out of the 4 variations of the architecture, LMCNet by itself performed the least desirable. Our results indicate a slight overfitting in LMCNet's sole performance compared. From both the accuracy and loss curves it can be seen that the test results still have a considerable gap to the training results, implying that model fails to generalise to the LMC training feature set.

B. MC

Figure 3d and Figure 3c are accuracy and loss curves of the MCNet stream. Our results indicate favourable and consistent performance to those of the recorded ones. The test results in both accuracy and loss show little divergence from the training curves, implying better generalisation than that of the LMC.

C. MLMC

MLMCNet's accuracy and loss curves are reported in Figure 3b and 3f. Although we managed to successfully replicate the reported average for MLMC within a 5% range, we still met a similar overfitting problem as in LMCNet. This could in fact imply that the MLMC feature-set itself is not a good representation for the task, further supporting Su et al claims that late fusion is more suitable.

D. TSCNN

Figures 5 show only the test accuracy/loss and average class accuracy. The choice to apply late fusion of both LMCNet and MCNet during testing only, provides us with insufficient data to comment on the results with respect to training. However, the overall accuracy of TSCNNNet is significantly better than that of LMCNet and is an improvement on MCNet, which is consistent with the author's claim.

IX. Qualitative Results

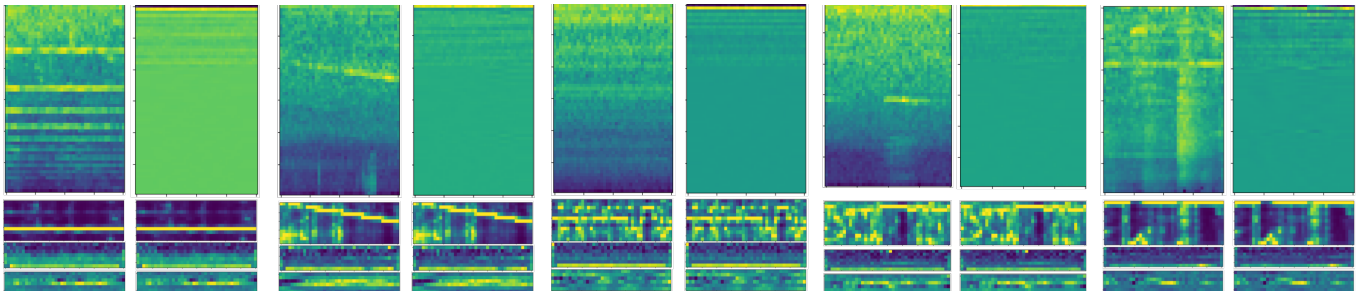
In our implementation, described in section VI, the 5395 test samples given was reduced by averaging the logits of respective files per batch. This gave us a total of 977 samples of averaged logits spanning all batches. Out of those 977 samples, 641 were classified correctly by both LMCNet and MCNet (Figure 4c), 65 were classified correctly by LMCNet but not MCNet (Figure 4a), 122 were classified correctly by MCNet but not LMCNet (Figure 4b), and 151 were classified incorrectly for both (Figure 4d). We then used TSCNNNet, which does better overall, but doesn't do much better on the cases where both LMCNet and MCNet fail. In fact, out of the 151 samples where both LMCNet and MCNet fail, TSCNNNet only correctly classifies a single one (figure 4e).

We conclude that the areas where generalisation is lacking in LMCNet and MCNet isn't catered for by late fusion to create TSCNNNet. However, it does produce an overall increase in generalisation, shown by the overall better class accuracy.

Overall, Figure 4 represents the spectograms of some of the 977 samples used. We only represent a single segment of the sample, which in practice is made up of several segments. However, we note the indices of all the segments in the sample, as a reference to the test data split of the given pre-prepared dataset.

X. Improvements

From the analysis of our results, we concluded that LMCNet and MLMCNet overfit to the training data, however MCNet provides good generalisation. In order to improve generalisation, we propose augmentation of the input data. By applying random scaling of the data by integers 1 to 4, the network will do better for generalising sound waves of varying amplitudes. We increased the number of epochs from



(a) (b) (c) (d) (e)

Fig. 4: (a) Correct classification by LMCNet but not MCNet (segment of 115241-9-0-9.wav (sm), indices {416 - 421}), (b) Correct classification by MCNet but not LMCNet (segment of 93567-8-0-6.wav (si), indices {5175 - 5181}), (c) Correct classification by both LMCNet and MCNet (segment of 189982-0-0-23.wav (ac), indices {3064 - 3070}), (d) Incorrect classification by both LMCNet and MCNet (segment of 167464-0-0-6.wav (ac), indices {2244 - 2250}), (e) Incorrect classification by both LMCNet and MCNet but correctly classified by TSCNNNet (segment of 77901-9-0-6.wav (sm), indices {4690 - 4696})

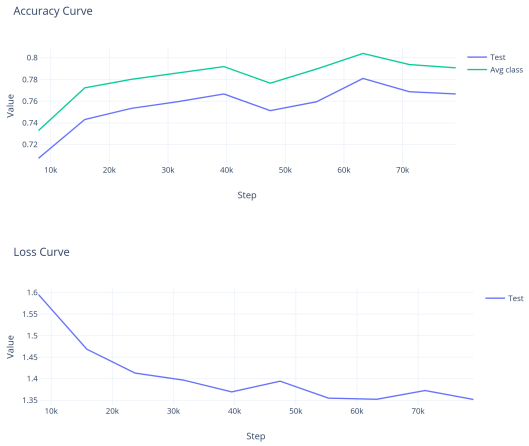


Fig. 5: TCSNN training curves

Class	LMC (LMCNet)	MC (MCNet)	MLMC	TSCNN
ac	54.4	83.0	59.3	67.3
ch	93.1	76.3	94.7	89.4
cp	78.1	76.0	78.6	77.2
db	65.6	69.0	77.0	68.0
dr	72.5	73.7	78.8	73.0
ei	56.0	84.4	70.6	81.9
gs	100.0	97.1	97.1	99.2
jh	99.1	92.9	96.4	98.9
si	57.6	60.2	60.2	63.7
sm	73.2	78.0	77.1	77.8
Avg.	**75.0**	**79.1**	**79.0**	**80.0**

Table II: Class-wise accuracy of four models with four-layer CNN evaluated on our split of *UrbanSound8K* using data-augmentation during training.

50 to 100 as the network trains slower with more varying data; all other, however, hyperparameters are kept the same.

The results in table II show improvements in the class-wise accuracy for MCNet, MLMCNet and TSCNNNet, with LMC-Net showing no significant change. The random augmentation of the input data provide more variance to the networks which reduce overfitting and allow training for more epochs.

XI. Conclusion and Future Works

To summarise, the work done by Su et al is a notable example of recent progress made in environmental sound classification using deep learning models. Our own attempt at replicating their work shows favourable and consistent results in support of their claim. The performance of TSCNNNet achieves the highest accuracy out of other tested models; indicating that late fusion using the proposed architecture does perform better than input level fusion (like in MLMCNet) in terms of accuracy. Furthermore, the training curves of our MLMCNet implementation shows signs of overfitting. Having shown architecture's promise capability, this could imply that the fault of overfitting in MLMCNet is due to the feature set instead, which again supports the authors assertion that simple aggregation of multiple features is not enough.

Having tried various combinations of hyperparameters we concluded that the hyperparameter values we used give the optimal result as referenced in Table I. Therefore, we approached generalisation by augmenting the input data. This showed better generalisation of the networks. Other methods to attempt generalisation would be in the form of changes to the proposed architecture such as more layers or more streams.

References

[1] Y. Sung, K. Zhang, J Wang and K Madani. "Environment sound classification using a two-Stream CNN based on decision-level fusion." In Sensors 19(7), 2019.

[2] Piczak, Karol J. "Environmental sound classification with convolutional neural networks." 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2015.

[3] Meyer, Matthias, Lukas Cavigelli, and Lothar Thiele. "Efficient convolutional neural network for audio event detection." arXiv preprint arXiv:1709.09888 (2017).

[4] Chen, Yan, et al. "Environmental sound classification with dilated convolutions." Applied Acoustics 148 (2019): 123-132.