Towards Low Complexity Model for Audio Event Detection

Muhammad Saleem[1], Syed Muhammad Shehram Shah[2], Erum Saba[3], Nasrullah Pirzada[4], Masood Ahmed[5]
saleemjamali600@gmail.com shehram.shah@faculty.muet.edu.pk nasrullah.pirzada@faculty.muet.edu.pk
Department of Telecommunication Engineering [1,4]
Department of Software Engineering [2]
Information Technology Centre[3]
Department of Computer System[5]
Mehran University of Engineering & Technology, Jamshoro, Sindh, Pakistan[1,2,4,5]
Sindh Agriculture University, Tandojam, Sindh, Pakistan[3]

Abstract
In our daily life, we come across different types of information, for example in the format of multimedia and text. We all need different types of information for our common routines as watching/reading the news, listening to the radio, and watching different types of videos. However, sometimes we could run into problems when a certain type of information is required. For example, someone is listening to the radio and wants to listen to jazz, and unfortunately, all the radio channels play pop music mixed with advertisements. The listener gets stuck with pop music and gives up searching for jazz. So, the above example can be solved with an automatic audio classification system. Deep Learning (DL) models could make human life easy by using audio classifications, but it is expensive and difficult to deploy such models at edge devices like nano BLE sense raspberry pi, because these models require huge computational power like graphics processing unit (G.P.U), to solve the problem, we proposed DL model. In our proposed work, we had gone for a low complexity model for Audio Event Detection (AED), we extracted Mel-spectrograms of dimension 128x431x1 from audio signals and applied normalization. A total of 3 data augmentation methods were applied as follows: frequency masking, time masking, and mixup. In addition, we designed Convolutional Neural Network (CNN) with spatial dropout, batch normalization, and separable 2D inspired by VGGnet [1]. In addition, we reduced the model size by using model quantization of float16 to the trained model. Experiments were conducted on the updated dataset provided by the Detection and Classification of Acoustic Events and Scenes (DCASE) 2020 challenge. We confirm that our model achieved a val_loss of 0.33 and an accuracy of 90.34% within the 132.50KB model size.

Keywords:
Audio Event Detection, Spatial Drop, Separable 2D, Batch Normalization, VGG, CNN, Quantization, and Mel_Spectrogram.

1. Introduction

Sound Classification or Audio Classification can be referred to as the process of analyzing audio recordings. Audio classification can be categorized in multiple forms such as Audio Event Detection (AED), Acoustic Data Classification, Music Classification, Natural Language Classification, and Environmental Sound Classification. Sound classification has multiple applications in the field of Artificial Intelligence (AI) and Data Science such as chatbots, automated voice translators, music genre classification, text-to-speech applications, and virtual assistants. Audio Classification is considered the heart of modern AI technology, audio classification could also be found in multimedia indexing and retrieval, and smart home security systems. Audio classification projects as we described above start with annotated audio data.

In many applications, Convolutional Neural Network (CNN) achieved success in applications like image recognition tasks [2], [3] and has recently been proven useful in tasks related to 1-Dimensional (1-D) type of data such as natural language processing [4] and speech recognition [5]. This depends on the quality and total amount of data used for training and the way data is fed to the Deep Learning (DL) model is important. In [6], *Sampathkumar et al.* It has been proven that by using different types of approaches, we can improve the performance of speech recognition. Many researchers proposed that by increasing additional data for training with a variation or by altering the shape of the audio recording or by adding additional background noise [7], [8], we can improve the performance DL model for audio classification. In [9], *Tokozume et al.* proposed an architecture such as SoundNet, that describes the different approaches to utilizing external data or knowledge.

The Audio classification DL models learn a very good representation of sound and sounds, and images included in a large number of without labeled video datasets. It is done by transferring the learned knowledge of pre-trained image recognition networks into sound recognition networks with the help of minimizing Kullback-Leibler (KL) divergence between the sound networks and outputs predictions of the image recognition network and that of the sound network. After that sound recognition's hidden layer's output is used as a feature, when it is applied to a sound classification

Manuscript received September 5, 2022
Manuscript revised September 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.9.26

problem and this classification is performed with a linear Support Vector Machine [10].

We had 4 different types of audio classification and related use cases for each. (i) Acoustic Data Classification: it is also known as acoustic event detection, in this type of classification, audios are identified from where those audios were recorded. It means, this distinguishes among environments including streets, homes, schools, restaurants, offices, etc. The acoustic data classification is also used for maintaining sound libraries for audio multimedia. We can use it to check the presence of fish in a different part of the ocean based on their acoustic type of data. (ii) Environmental Sound Classification: this name shows clearly that it is a classification of sounds within different types of environments. In this type of classification, we identify environments including roadwork, human voices, sirens, car horns, etc. (iii) Music Classification: music can be classified, in this type of classification, the application is based on factors such as instruments played and genre. This type of classification plays important role in improving recommendation systems algorithms, organizing and maintaining audio libraries by genre, and discovering trends and listeners' preferences through data analysis. (iv) Natural Language Utterance Classification: This type of classification is based on dialect, languages spoken, semantics, or other types of features.

Automatic Sound Classification is a growing field in AI, many researchers had already presented their research work on audio classification and speech reorganization. As DL models require huge computational power like graphics processing unit (G.P.U), individual users can't afford such computational powers in their machines or these DL models can't be implemented on edge devices such as mobile phones, nano BLE sense, and raspberry pi. To overcome this problem researchers had worked to reduce model complexity to fit in low constraint devise like IoT, Mobile phones, BLE sense, and Raspberry pi.

Our proposed work is mainly focused on low-complexity models as well as understanding various approaches for building low-complexity machine learning models (based on DL models), training deep learning neural models for audio classification, and applying quantization of float16 to reduce model complexity/model size of the trained model and compare the differences to (i) model's classification, and (ii) model size, which could be deployed at edge devices such as in nano BLE sense and raspberry pi. We used several methods to reduce the model size as well as computational cost. In our proposed work, we presented 2 different neural network architectures, (i) with depthwise separable 2D and (ii) without depth-wise separable 2D with different architectures to reduce the computational cost and memory footprint of models. We also compared our models with the baseline model and the other two models from previous work. We had used batch normalization, spatial dropout, and separable 2D trained all the models of our own and as well as a baseline model for 50 epochs on our updated dataset and we have seen that our models had outperformed with 9% accuracy model size is reduced to 4 times lesser as compared to baseline models on the updated dataset.

2. Literature Review

Acoustic Scene Classification (ASC) is nowadays a field of interest, many researchers had published their work on ASC application. But due to the huge computational power requirement, it has seemed that these ASC models should be reduced to smaller model sizes with fewer memory footprints as compared to the baseline one. Many researchers applied different audio feature extraction during audio processing like spectrograms, Mel-spectrograms, MFCCs, Constant Q-transform [11], and different methods of data augmentation to generate more samples from the available dataset. They used Time Masking, Frequency Masking, Mixup [12], Cutmix, Time Stretching, and Pitch Shift and used different neural architectures like CNN [13], VGGnet [14], RESNET [15], and different types of 2D filters. They used convolution 2D and separable 2D after training models for AED, they used different methods to optimize the model like model pruning, knowledge distillation (KD), and quantization [16], in model quantization, they used float16 [17]. Most recent work and their achievements are written in detail here. In [18], *Lopez-Meyer et al.* proposed work for DCASE task 1b, they experimented with different low-memory implementations of CNN challenge guidelines. Pruning, KD, and quantization as input features, also mixup data augmentation were used to reduce the model size, to reach the highest accuracy, and deleting the total number of parameters in the context of the model trainable parameters as well as model size. They investigated different methods and their experimented results yielded achieved accuracy higher than the baseline model, which is more than 90%, where accuracy of baseline model is 87.30%. Their submissions managed to achieve more than 90.00% accuracy using CNN models with less than 500 KB model size.

The limitation in their work is that they did not achieve smaller model size as the model should have, their model size is below 500KB. In [19], *Ngo et al.* participated in the DCASE challenge used spectrograms and mix-up data augmentation and achieved very competitive results of 7.20%, compared to the baseline model in DCASE 2020 task 1b challenges their achieved accuracy is 94.50% and the model size with 445KB and baseline model had an accuracy of 87.30% and model size of 450KB. Limitation in their work, they had achieved better model accuracy 94.50% but their model size is larger which is 445KB, the model size still could be reduced, while accuracy could also be increased. In [20], *Pajusco et al*. proposed a method to remove the least important convolutional kernels by using iterative structured parameter pruning. They used weight quantization of float16 to reduce weights in half precision. They trained two different types of network architecture: (i) 1 Dimensional-CNN based VGG-like blocks, and (ii) ResNet architecture with 1-Dimensional convolutions. Their experiments dictate that they could train, quantify, and prune a VGG model to reduce it 20 times smaller than the 500KB restricted limitation by DCASE challenge with an accuracy of the baseline model (87.60%), as well as their larger proposed model achieved 91% accuracy, with reduction of 8 times smaller than the challenge limit. They used ResNet model architecture could successfully be trained, quantify, and pruned to be below 500KB, achieving up to 91.20% accuracy. Limitations in their work is model size, which is 466.60 KB with higher accuracy of 91.20 %, which is below 500KB, it could still be reduced to smaller memory footprint and accuracy could also be increased. In [21], *Singh et al*. authors presented a network comprises of three convolutional layer, which are followed by a full connected layer after training the model authors opted filter pruning method and quantization of float16, authors achieved an unpruned model size of the network of 90.30KB with 46246 number parameters with the accuracy of 48.59% and with pruning model achieved an accuracy of 48.65% with model size 71.44 KB. Their model size is small, which is good but their model accuracy is poor at 48.59%. It means below 50.00% accuracy, which is not good accuracy, it should be improved. In [22], *Pham et al.* proposed experiments on the DCASE 2022 task 1 development dataset and, achieved their 60.10% highest classification accuracy of improving the DCASE baseline by 17.2% with a model size of 128 KB.

3. Methods

The idea behind this research is to reduce model complexity to fit Deep Neural Network (DNN) on constraint devices like nano BLE sense and raspberry pi, we are very inspired by the capabilities of DNN. Many authors had proposed different methods to reduce the model size of AED like quantization [23], model pruning, and KD to fit AED models in constraint devices in nano BLE sense and raspberry pi, they had trained those models on different datasets. Our main aim of this research is to design DNN with a smaller number of parameters with the least cost of the number of calculations. For doing that we proposed work, that reduced model size, and maintained model performance approximately the same as the accuracy of the baseline model. We used Mel-spectrograms as an input feature and used batch normalization after each hidden layer to make outputs of all neurons to the same scale, between 0 to 1 and we divided the dataset in Train-Test split using data augmentation and applied post-training quantization as shown in the flow chart given in figure 5.

3.1. Audio Preprocessing

We had trained our model with the dataset basically provided by a challenge known as Detection and Classification of Acoustic Scenes and Events (DCASE) 2020. There are many different types of audio feature extraction methods, but we had converted the whole dataset of pre-recorded audios into Mel-spectrograms of dimension 128x431x1, Mel-spectrogram is the pictorial representation of audios in logarithmic scale in the time-frequency domain, this dataset is consisting of 3 classes, Mel-spectrogram of 3 classes, which is shown in figure 1.

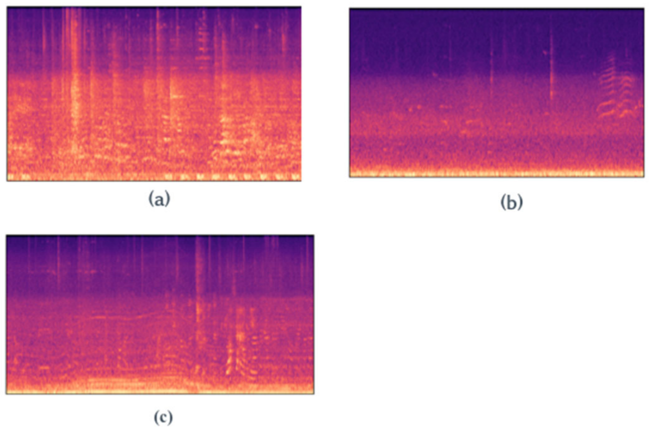


Figure 1: Mel-spectrogram of Class-0, Class-1, Class-2 in (a), (b), and (c) respectively

3.2. Data Augmentation

Many data augmentation methods are frequently used like Pitch Shitting, Time Stretching, Additive Noise, Frequency Masking, Time Masking, and Mixup. We used three different types of data augmentation methods, Time Masking, Frequency Masking, and Mixup to overcome the overfitting problem. These methods had been used to create more data samples from the available dataset to make our model more reliable during the training process as shown in figure 2, figure 3, and figure 4 respectively.

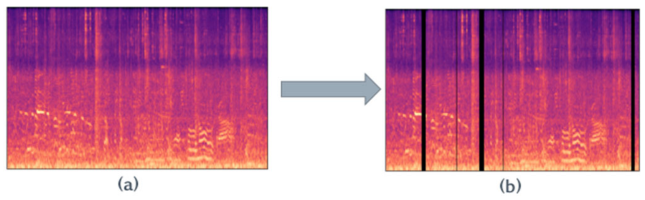


Figure 2: Time Masking

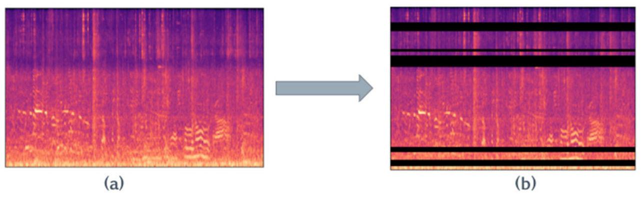


Figure 3: Frequency Masking

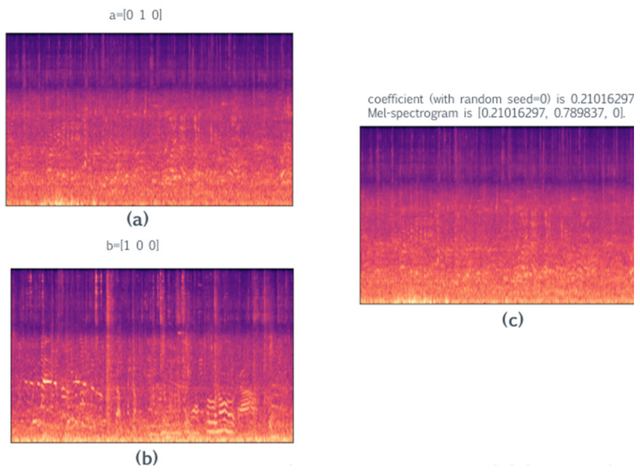


Figure 4. Indoor Mel-spectrogram, outdoor Mel-spectrogram, and mixup of indoor and outdoor are shown in (a), (b), and (c) respectively

In mixup, we combine Mel-spectrograms from indoor and outdoor. Indoor sample has one-hot encoded label as [1,0,0] whereas Outdoor sample has label of [0,1,0]. With a randomly generated MixUp coefficient (with random seed=0) is 0.21016297, the label for mixup Mel-spectrogram is [0.21016297, 0.789837, 0].

3.3 Proposed Methodology

In this section of the paper, we will discuss the flow chart of the proposed methodology step by step in figure 5. In step (i) we used pre-recorded audios to feed the input layer, step (ii) those pre-recorded audios had been converted to Mel-spectrogram, step (iii) Mel-spectrograms are normalized to the same scale from 0 to 1, step (iv) those all normalized values of Mel-spectrograms are split into training and testing, step (v) after splitting data into training and testing we used data augmentation to the available dataset to increase the number of samples, step (vi) after data augmentation those all data had been fed to DL model, step (vii) after training model, pre-trained model had been quantized by using post-training quantization of float16 shown in figure 5.

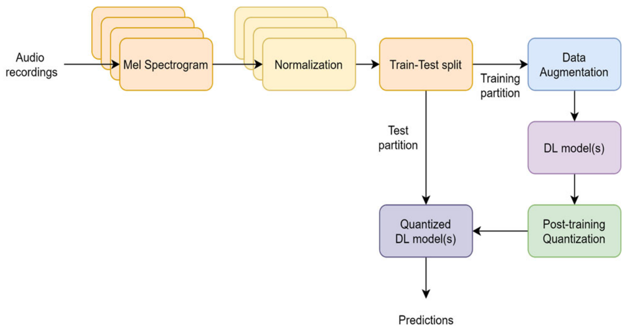


Figure 5. Flow chart of the proposed model.

3.4 Dataset

The details for the dataset used in this work are given in table 1:

Table 1. The total number of segments from each class and each location is shown he

	Number of Segments		
Location	**Indoor**	**Outdoor**	**Transportation**
Barcelona	4320	5760	4320
Helsinki	416	577	428
Lisbon	437	577	433
London	432	576	432
Lyon	433	577	434
Milan	432	576	432
Paris	444	576	432
Prague	432	576	432
Stockholm	446	577	433
Vienna	416	572	432
Total	8208	10944	8208

The dataset is divided into three classes indoor, outdoor, and transportation. In indoor class we have ["airport", "metro_station", "shopping_mall"]. In outdoor class we have outdoor ["park","public_square","street_pedestrian","street_traffic"]. In transportation class we have transportation ["bus", "metro", "tram"]. We used a subset of datset due to limited storage and processing power, from indoor class ["airport", "shopping_mall"], from outdoor ["park", "public_square"], from transportation class we have ["bus", "metro", "tram"] i.e., we removed from indoor class ["metro_station"], from outdoor class ["street_pedestrian","street_traffic"], and kept transportation class as it was.

Data partitioning into train and test partitions is like the recommended partitions from DCASE 2020.

3.5 Depthwise Separable 2D vs Convolution 2D

Depthwise separable 2D reduces the cost of calculation during training of the model, in depthwise separable 2D we use filters of depth 1 for each channel of input data, once features are extracted then all features get passed through the filter of 1x1xM, and all features are added to make the depth of feature map same as the depth of input data. While convolution 2D uses a filter of the same depth as the depth of our input data. Depthwise separable 2D and convulsion 2D are given in figure 6 and figure 7 respectively, where Df x Df x M shows the width, height, and depth of the input image as well as filters used for feature extraction.

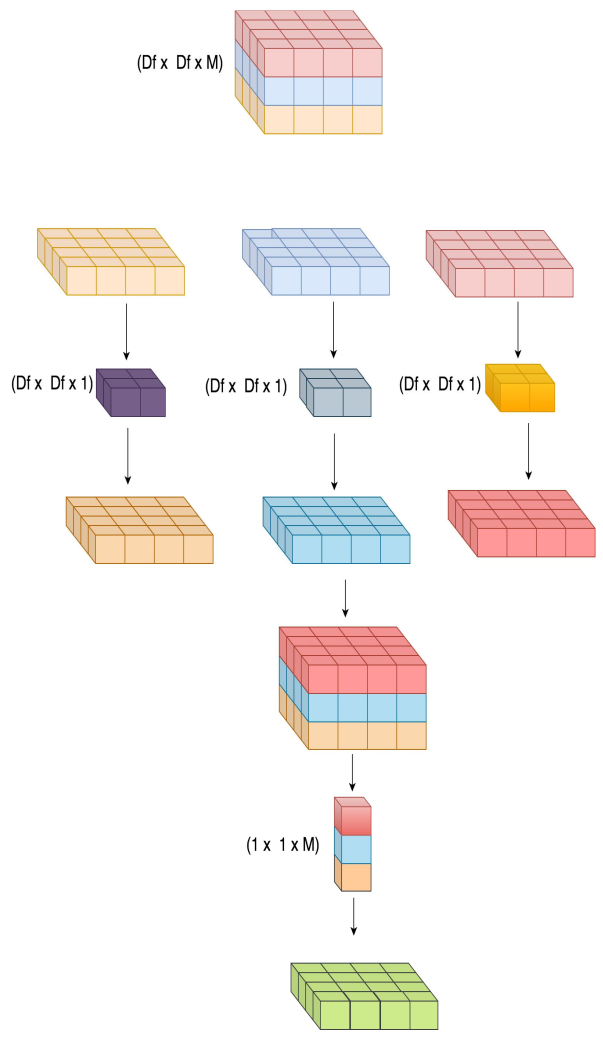


Figure 6: Depthwise separable convolution

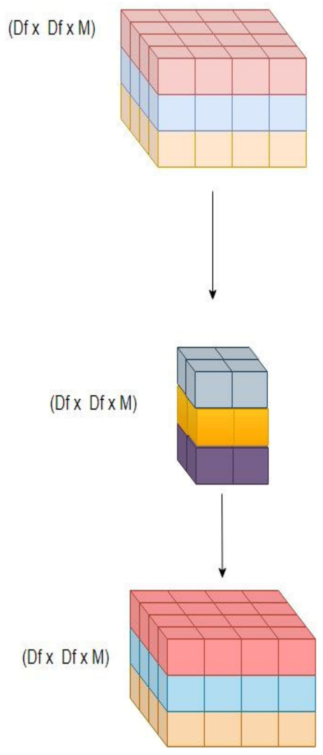


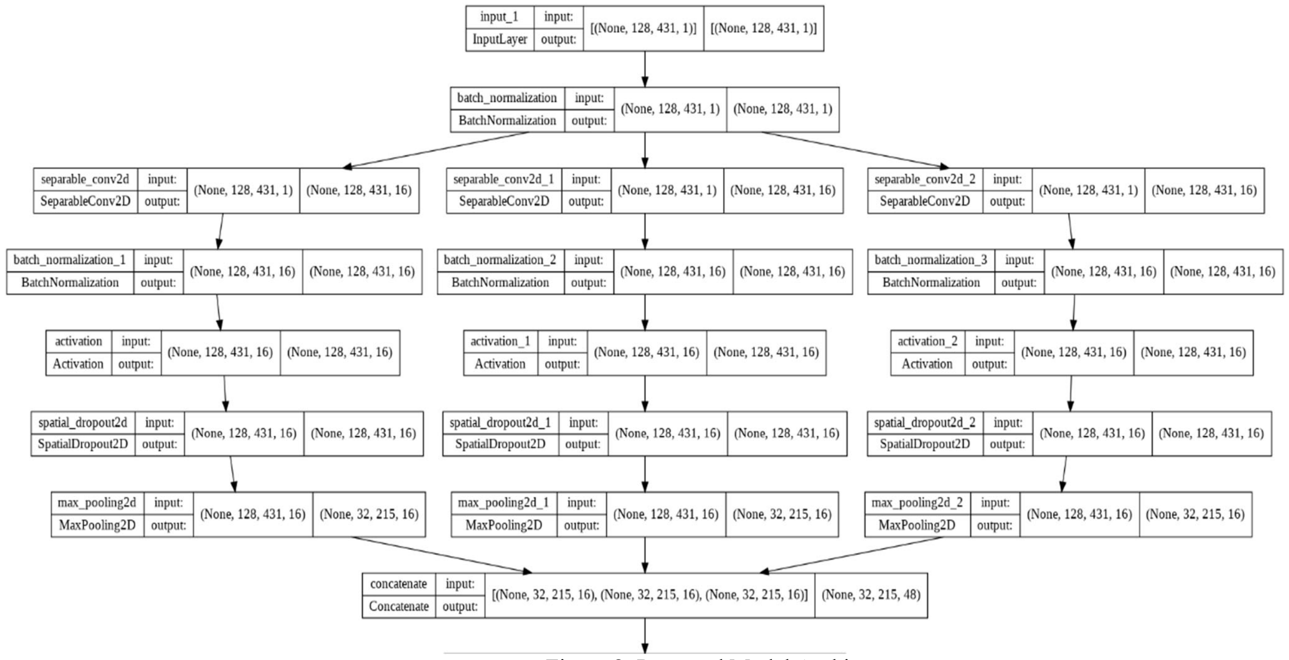
Figure 7. Diagram of Convolution 2D

3.5 Deep Learning Model CNN (VGG)

We proposed DL architecture based on VGGnet, and CNN with some extra layers, batch normalization, spatial drop-out, and separable convolutions.

We used a spatial drop to reduce the overfitting problem and used three parallel separable 2D layers to reduce the cost of calculations, we extracted features by using three different dimensions of the kernel, for extracting features at the time axis we used the kernel of (10,1), for frequency axis kernel of (1,10), and for both frequency axis and time axis, we had to use the kernel of (3,3).

In the end, we had concatenated all the features extracted during the training process whole model architecture is shown in figure 8. This architecture provided outstanding results.



input_1 InputLayer input: output: [(None, 128, 431, 1)] [(None, 128, 431, 1)]
batch_normalization BatchNormalization input: output: (None, 128, 431, 1) (None, 128, 431, 1)
separable_conv2d SeparableConv2D input: output: (None, 128, 431, 1) (None, 128, 431, 16)
separable_conv2d_1 SeparableConv2D input: output: (None, 128, 431, 1) (None, 128, 431, 16)
separable_conv2d_2 SeparableConv2D input: output: (None, 128, 431, 1) (None, 128, 431, 16)
batch_normalization_1 BatchNormalization input: output: (None, 128, 431, 16) (None, 128, 431, 16)
batch_normalization_2 BatchNormalization input: output: (None, 128, 431, 16) (None, 128, 431, 16)
batch_normalization_3 BatchNormalization input: output: (None, 128, 431, 16) (None, 128, 431, 16)
activation Activation input: output: (None, 128, 431, 16) (None, 128, 431, 16)
activation_1 Activation input: output: (None, 128, 431, 16) (None, 128, 431, 16)
activation_2 Activation input: output: (None, 128, 431, 16) (None, 128, 431, 16)
spatial_dropout2d SpatialDropout2D input: output: (None, 128, 431, 16) (None, 128, 431, 16)
spatial_dropout2d_1 SpatialDropout2D input: output: (None, 128, 431, 16) (None, 128, 431, 16)
spatial_dropout2d_2 SpatialDropout2D input: output: (None, 128, 431, 16) (None, 128, 431, 16)
max_pooling2d MaxPooling2D input: output: (None, 128, 431, 16) (None, 32, 215, 16)
max_pooling2d_1 MaxPooling2D input: output: (None, 128, 431, 16) (None, 32, 215, 16)
max_pooling2d_2 MaxPooling2D input: output: (None, 128, 431, 16) (None, 32, 215, 16)
concatenate Concatenate input: output: [(None, 32, 215, 16), (None, 32, 215, 16), (None, 32, 215, 16)] (None, 32, 215, 48)

Figure 8: Proposed Model Architecture

4. Experiments and results

From our experiments we had come across many best results, we applied different methods as well as investigated total 2 models with different architectures and used spatial drop and normal drop out, used convolutional 2D and separable 2D, and also had trained baseline model architectures and other two model architectures from previous work, all models are trained for 50 epochs on the updated dataset. Results of the baseline model, previous work, and proposed model are given in table 2, and training accuracy, training loss, validation accuracy, and validation loss are given in figure 9 and figure 10, respectively. From these results, we can find that our proposed model is much better compared to other models.

Table 2: Results of baseline, previous work, and our proposed model

M_ID	**Baseline**	**Sampath**	**Vilouras**	**M_2_3**	**M_2_0**
Accuracy	80.85 %	88.35 %	87.06 %	90.34 %	90.85%
Val_loss	0.66	0.40	0.53	0.33	0.59
Val_accuracy	77.36%	86.%59	85.17%	87.93 %	83.05%
Parameters	108,915	28,275	64,949	7,968	86,387
Non-Trainable parameters	192	96	354	718	718
Memory	665.60KB	196.50KB	429.50KB	132.50 KB	563.20 KB

In table 2 we can notice the performance and model size of our proposed model. In our proposed work, we selected Model M_2_3 with separable 2D and M_2_0 without separable 2D to fit in constraint devices such as mobile phones, nano BLE sense, and raspberry pi. These models have low complexity and reduced memory footprint and with a little bit of degradation of accuracy.

In figure 9, we can notice how training accuracy is increased with each epoch and we also analyzed 3 moving averages of training accuracy. In addition, we ran it for 50 epochs. In addition to that, we also analyzed the validation accuracy of the model for 50 epochs, and 3 moving average validation accuracy is also analyzed for 50 epochs, here we can observe that training accuracy is increased from 55.00% to 90.34%, and in addition to that, we can also observe validation accuracy for 50 epochs that how validation accuracy is increased from 35.00% to 87.93%

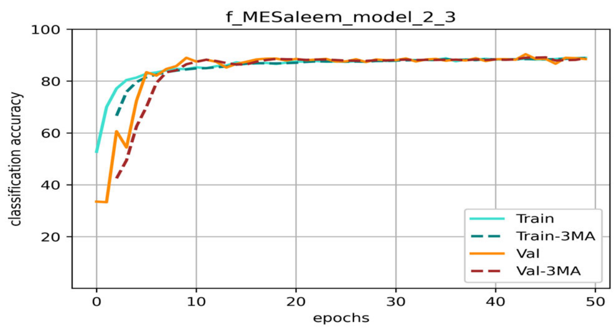


Figure 9: Training accuracy in continuous and 3 moving averages for 50 epochs and validation accuracy in continuous and 3 moving averages for 50 epochs.

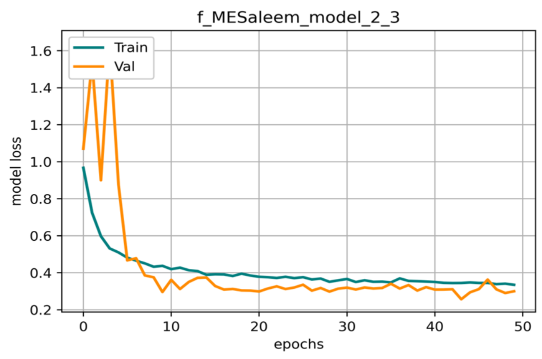


Figure 10 shows training loss and validation loss for 50 epochs.

In figure 10 we observe how training and validation losses are decreased from epoch to epoch as we can observe that training loss is decreased from 0.90 to 0.33.

5. Conclusions

As we had experimented with different methods to design CNN-based models, we added drop out, spatial drop, batch normalization, and separable 2D and trained all models for 50 epochs on the updated dataset. In addition to that, we also trained the baseline model's architecture and two other model architectures from previous work on the same updated dataset and observed the results of baseline models and proposed models. We compared proposed models in the context of (i) model performance and (ii) model size. We concluded that our model's performance is increased as compared to baseline models with reduced model size. A comparison is given in table 2. From table 2, we can observe that our 1st model with Model_ID M_2_3 has a 132KB model size with an accuracy of 90.34%, and the baseline model has 665.60KB with an accuracy of 80.85%. Our model has less memory footprint and an accuracy of 90.34%, and 2nd proposed model with Model_ID M_2_0 has a 563.20 KB model size with an accuracy of 90.85%.

References

[1] K. Ooi, S. Peksi, and G. Woon-Seng, "Ensemble of Pruned Models for Low-Complexity Acoustic Scene Classification," Jun. 2020", in *Detection and Classification of Acoustic Scenes and Events 2020*,

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, DOI: 10.1109/CVPR.2016.90.

[3] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Lecture Notes in Computer Science*, 1999, pp. 1–28.

[4] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489. DOI: 10.18653/v1/n16-1174.

[5] [5] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio, "ReNet: A Recurrent Neural Network Based Alternative to Convolutional Networks," *arXiv:1505.00393*, pp. 1–9, 2015.

[6] A. Sampathkumar and D. Kowerko, "Low complexity acoustic scene classification using aalnet-94," *Detect. Classif. Acoust. Scenes Event*, pp. 1–5, 2020.

[7] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[8] S. Kahl *et al.*, "Large-Scale Bird Sound Classification using Convolutional Neural Networks," in *CLEF 2017 Working Notes*, 2017, pp. 1–14.

[9] Y. Tokozume, Y. Ushiku, and T. Harada, "Learning from between-class examples for deep sound recognition," in *International Conference on Learning Representations*, 2018, pp. 1–13.

[10] J. Salamon, C. Jacoby, and J. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, 11 2014.

[11] B. McFee *et al.*, "librosa: Audio and music signal analysis in python," in *14th Python in Science Conference*, 2015, pp. 18–24.

[12] J. A. Lopez *et al.*, "Low-memory Convolutional neural networks for acoustic scene classification," in *Detection and Classification of Acoustic Scenes and Events 2020*, 2020, no. November, pp. 96–99

[13] J. A. Lopez *et al.*, "low-memory convolutional neural networks for acoustic scene classification," in *Detection and Classification of Acoustic Scenes and Events 2020*, 2020, no. November, pp. 96–99.

[14] N. Pajusco, R. Huang, and N. Farrugia, "Lightweight Convolutional Neural Networks on Binaural Waveforms for

Low Complexity Acoustic Scene Classification," Jun. 2020.

[15] N. Pajusco, R. Huang, and N. Farrugia, "Lightweight Convolutional Neural Networks on Binaural Waveforms for Low Complexity Acoustic Scene Classification," Jun. 2020.

[16] J. Zhang, C. Ren, and S. Li, "Bupt submissions to dcase 2020: low-complexity acoustic scene classification with post-training static quantization and prune," *Detect. Classif. Acoust. Scenes Events*, pp. 1–5, 2020.

[17] A. S. Singh1 and P. R. Devalraju2, Dhanunjaya Varma, Rajan3, "pruning and quantization for low-complexity acoustic scene classification," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 2–3.

[18] J. A. Lopez *et al.*, "low-memory convolutional neural networks for acoustic scene classification," in *Detection and Classification of Acoustic Scenes and Events 2020*, 2020, no. November, pp. 96–99.

[19] D. Ngo, L. Pham, A. Nguyen, and H. Hoang, "Cnn-based framework for dcase 2020 task 1b challenge," *Detect. Classif. Acoust. Scenes Event*, pp. 1–5, 2020.

[20] N. Pajusco, R. Huang, and N. Farrugia, "Lightweight Convolutional Neural Networks on Binaural Waveforms for Low Complexity Acoustic Scene Classification," Jun. 2020.

[21] A. S. Singh1 and P. R. Devalraju2, Dhanunjaya Varma, Rajan3, "pruning and quantization for low-complexity acoustic scene classification," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015, pp. 2–3.

[22] L. Pham, H. Tang, A. Jalali, A. Schindler, R. King, and I. McLoughlin, "A Low-Complexity Deep Learning Framework For Acoustic Scene Classification," *Data Sci. – Anal. Appl.*, pp. 26–32, 2022, DOI: 10.1007/978-3-658-36295-9_4.

[23] A. Sampathkumar and D. Kowerko, "Low complexity acoustic scene classification using aalnet-94," *Detect. Classif. Acoust. Scenes Event*, pp. 1–5, 2020.