Group 12 Proton: Audio Classification using ensemble of 1D and 2D CNN

Abhinav Dudeja 21111001 Arpan Kapoor 21111016 Piyush Gangle 21111046
{abhinavd21, arpank21, piyushg21}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

Sound classification is one of the most widely used applications in Audio Deep Learning. It involves learning to classify sounds and to predict the category of that sound. he voice. Our motivation was to build an audio classifier which will be better than existing state of the art classifiers. In order achieve this goal we applied ensemble method to combine 1D and 2D convolution neural network. 1D CNN considers amplitude based audio features whereas 2D CNN captures spectral features, hence final classification decision will be made considering both the features which will effectively increase overall accuracy of classifier.

1 Introduction

In this project, we have developed an audio classifier that takes advantage of two deep learning models (1D and 2D CNN) to predict labels for audio samples. We have proposed an ensemble method which helps improve classification accuracy of the stand alone 1D and 2D convolution models.

Audio signal is an analog signal having varying amplitude over a period of time. The height shows the intensity of the sound and is known as the amplitude.

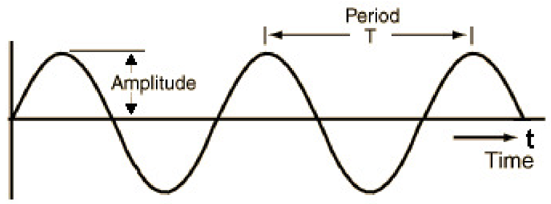


Figure 1: Analog audio signal

Spectrogram is a visual representation of the spectrum of frequencies of an audio signal as it varies with time. A spectrogram is usually depicted as a heat map, i.e., as an image with the intensity shown by varying the colour or brightness.

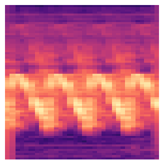


Figure 2: Spectrogram of a siren

2 Related Work

Literature survey helped us to explore the current state of the art solutions that are already available. The approaches that we found relevant to our topic are mentioned below:

- (**Base paper**) End-to-end environmental sound classification using a 1D convolutional neural network ([1]): It uses 1D CNN in order to classify the UrbanSound8K sound dataset. As mentioned it uses 1D CNN therefore it uses only amplitude features of audio sample. The authors of this paper claims to achieve an accuracy of 87%.
- Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification ([2]): Authors have proposed a deep convolutional neural network (2D CNN) architecture for environmental sound classification with use of audio data augmentation for overcoming the problem of data scarcity and explore the influence of different augmentations on the performance of the proposed CNN architecture.
- 1D/2D deep CNNs vs. temporal feature integration for general audio classification ([3]): Authors aimed to optimize a lightweight configuration for convolutional network topologies on a Speech/Music/Other classification scheme that can be deployed on various audio information retrieval tasks, such as voice activity detection, speaker diarization or speech emotion recognition. The outmost target of their work was to establish an optimized protocol for constructing deep convolutional topologies on general audio detection classification schemes, minimizing complexity and computational needs.
- Our Approach: In order to build a better audio classifier than existing ones we tried to use combination of 1D and 2D CNN because 1D CNN extracts amplitude features and 2D CNN extracts spectral features, therefore classification will be more fair if we consider both features during the time of decision making.

3 Proposed Idea

Proposed idea is to build an audio classifier that will have better prediction accuracy than the existing models. We are using amplitude and spectral features to classify audio for which we combine the 1D and 2D CNN model.

Classifying audio based on a single feature can cause misclassification of audio samples with similar distribution for those features. Comparing audio using different features can help resolve this issue. Since 1D and 2D CNN require different inputs, they take into consideration different features thus helping in improving the overall accuracy of the system.

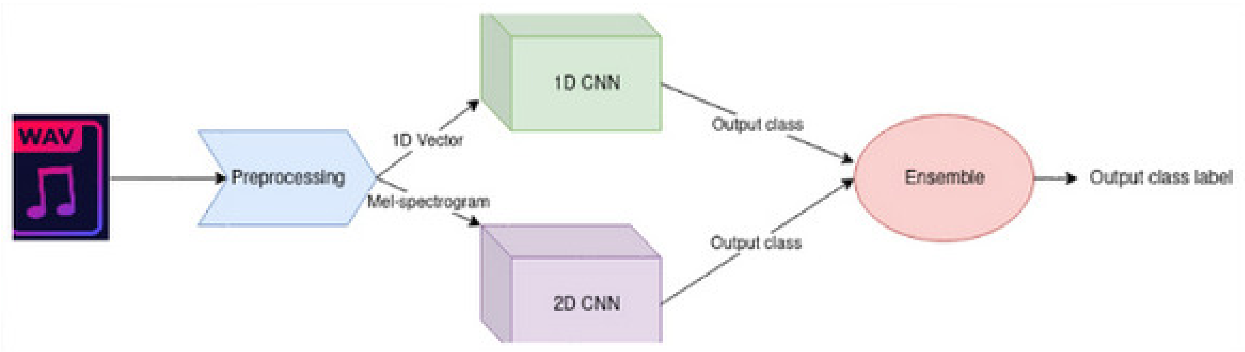


Figure 3: Work flow of proposed audio classifier

4 Methodology

4.1 Dataset

We have used UrbanSound8K dataset [4]. It contains 8732 samples of varying lengths ($\leq$ 4s) distributed among 10 classes (air conditioner, car horn, children playing, dog bark, drilling, enginge idling, gun shot, jackhammer, siren, and street music). The dataset comes with 10 pre-sorted folds, which are to be used for 10-fold cross validation to get the evaluation of the model.

4.2 Preprocesssing

Data from the dataset couldn't directly be used and had to be preprocessed. We loaded the data using sampling rate as 16KHz, which means we are reading 16K data points for each second of audio. Next we divided these audio signals into overlapping equi-sized chunks, as the CNN requires a fixed size as an input. Mel-spectrogram images were also generated from the chunks to be fed to the 2D CNN as input.

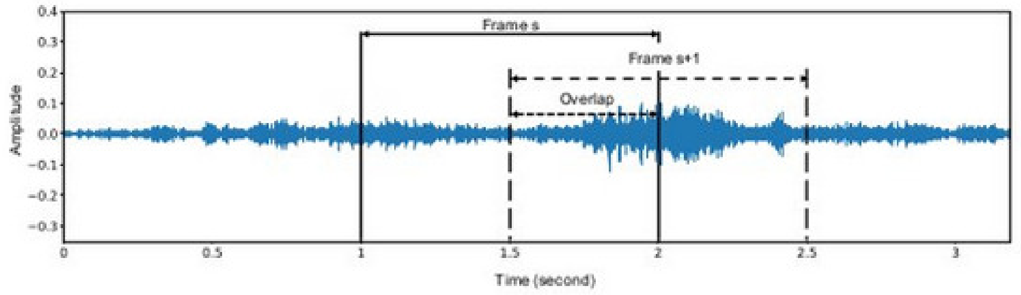
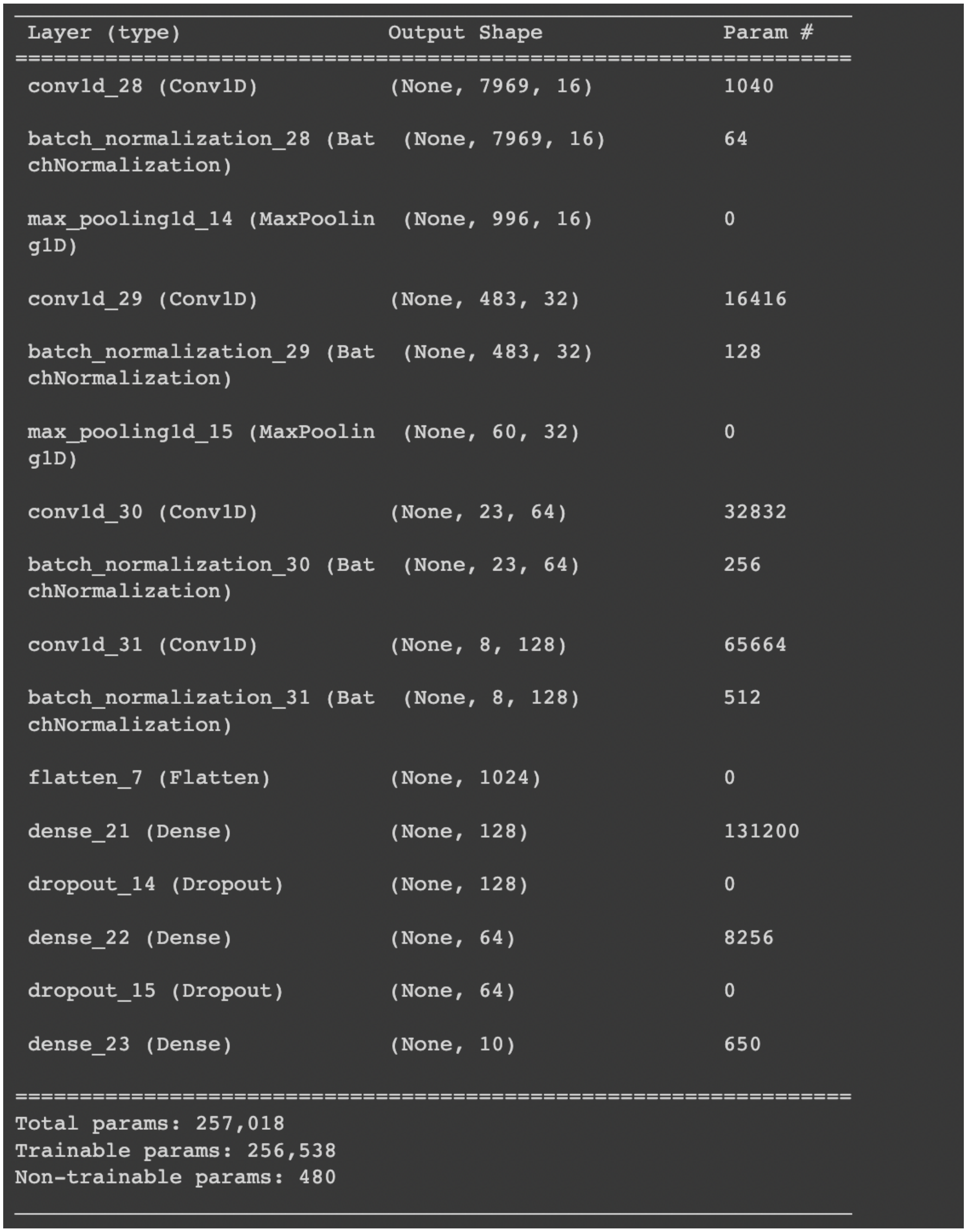


Figure 4: Illustration for overlapping chunks of audio signal

4.3 Computer vision models used

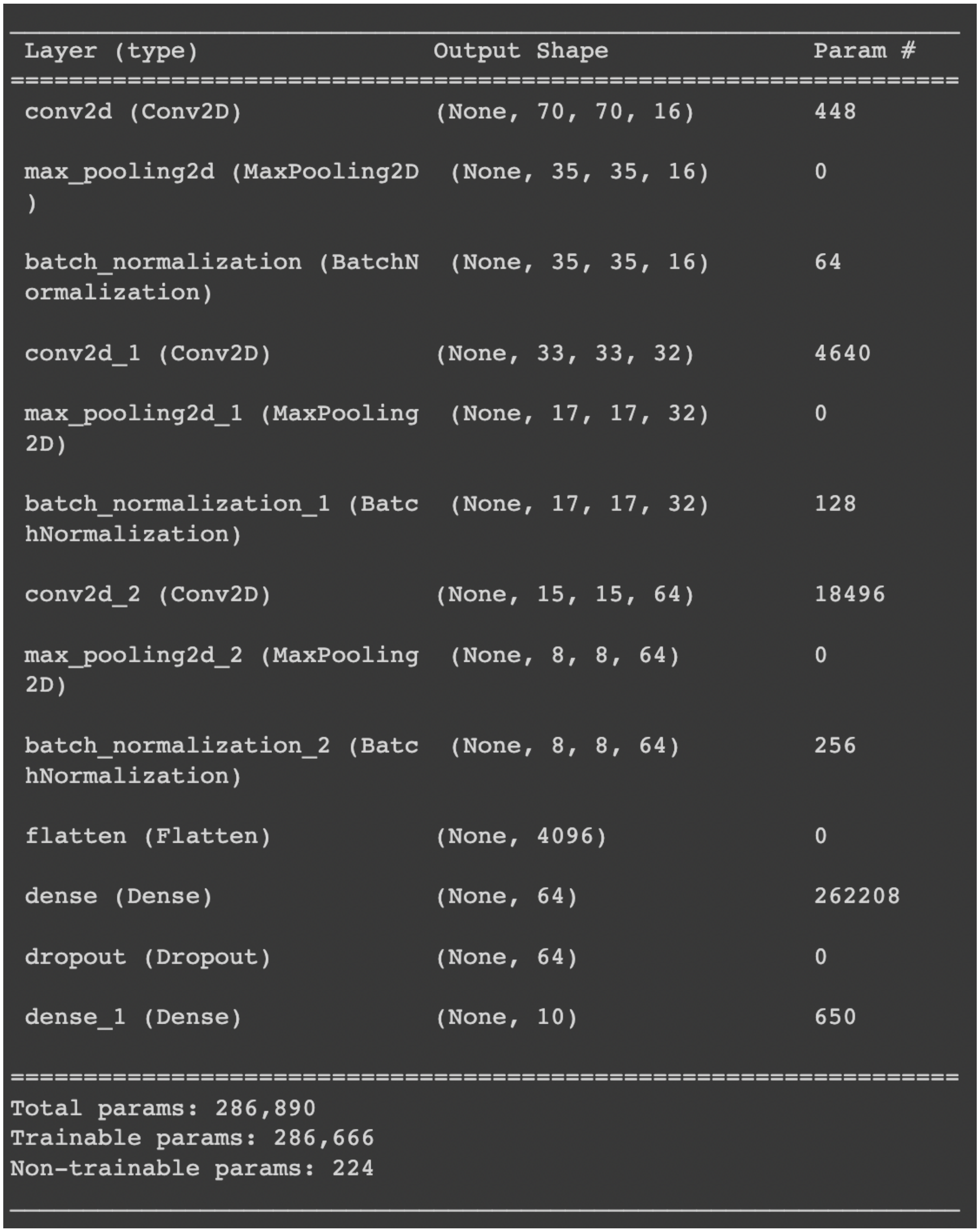
Computer Vision model used are mentioned as follows :

1. **1D CNN :** We have used the same architecture for the 1D CNN as was used by the authors of [1]. The architecture contains 4 convolution layers and 2 pooling layers, with 2 fully connected layers followed by a softmax layer for final classification. Detailed view of the architecture can be seen in figure 5
2. **2D CNN-1 (Proposed by us) :** We have proposed a 2D CNN model with 3 convolution layers and 2 pooling layers followed by 1 fully connected layer and a final softmax classification layer. Figure 6 shows the complete architecture
3. **2D CNN-2 (using transfer learning on VGG16):** We have used VGG16 architecture pre-trained on ImageNet with all except the last CONV layer frozen. We have added a fully connected layer with 64 neurons and a final softmax layer for classification.



```
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_28 (Conv1D)          (None, 7969, 16)          1040

 batch_normalization_28 (Bat  (None, 7969, 16)         64
 chNormalization)

 max_pooling1d_14 (MaxPoolin  (None, 996, 16)          0
 g1D)

 conv1d_29 (Conv1D)          (None, 483, 32)           16416

 batch_normalization_29 (Bat  (None, 483, 32)          128
 chNormalization)

 max_pooling1d_15 (MaxPoolin  (None, 60, 32)           0
 g1D)

 conv1d_30 (Conv1D)          (None, 23, 64)            32832

 batch_normalization_30 (Bat  (None, 23, 64)           256
 chNormalization)

 conv1d_31 (Conv1D)          (None, 8, 128)            65664

 batch_normalization_31 (Bat  (None, 8, 128)           512
 chNormalization)

 flatten_7 (Flatten)         (None, 1024)              0

 dense_21 (Dense)            (None, 128)               131200

 dropout_14 (Dropout)        (None, 128)               0

 dense_22 (Dense)            (None, 64)                8256

 dropout_15 (Dropout)        (None, 64)                0

 dense_23 (Dense)            (None, 10)                650

=================================================================
Total params: 257,018
Trainable params: 256,538
Non-trainable params: 480
_________________________________________________________________
```

Figure 5: 1D CNN Model Architecture



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 70, 70, 16)        448

 max_pooling2d (MaxPooling2D  (None, 35, 35, 16)       0
 )

 batch_normalization (BatchN  (None, 35, 35, 16)       64
 ormalization)

 conv2d_1 (Conv2D)           (None, 33, 33, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 17, 17, 32)       0
 2D)

 batch_normalization_1 (Batc  (None, 17, 17, 32)       128
 hNormalization)

 conv2d_2 (Conv2D)           (None, 15, 15, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 batch_normalization_2 (Batc  (None, 8, 8, 64)         256
 hNormalization)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 64)                262208

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 286,890
Trainable params: 286,666
Non-trainable params: 224
_________________________________________________________________
```

Figure 6: 2D CNN Model Architecture

4.4 Implementation details

`1d.ipynb` contains code to train and test the accuracy for the 1D as well as ensemble models. `2d.ipynb` has code to train both our custom 2D CNN models including our custom CNN and the VGG16 finetuned model. `demo.ipynb` uses the weights from the trained model to predict the class of any given audio file. Further instructions and details are provided in `README.md`

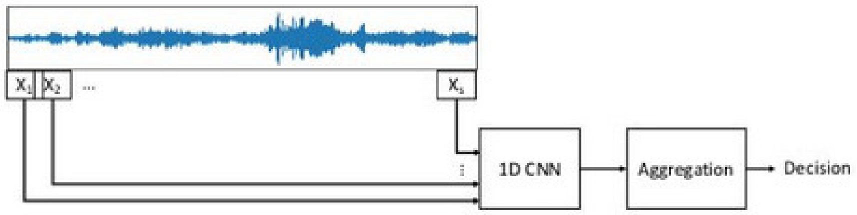


Figure 7: Aggregation function illustration

5 Results

Following table shows the average and best accuracy that was achieved by our models across all the folds of the dataset. As we can see, ensemble method helps in improving the accuracy of the stand alone 1D and 2D CNN models. 1D and 2D CNN achieves an average accuracy of 60.98% and 69.74% respectively. This accuracy value jumps up to 72.54% when we use the two models in ensemble. Results also improve when using a pre-trained VGG16 model, ensembled with 1D CNN.

5.1 Accuracies

Model	Average Accuracy	Standard Deviation	Best Accuracy
1D CNN	60.98	7.17	70.81
2D CNN	69.74	5.39	76.02
2D CNN (VGG16)	73.23	5.29	82.44
Ensemble (1D + 2D CNN-1)	72.54	6.47	80.32
Ensemble (1D + 2D CNN-2(VGG16))	73.96	6.46	83.94

5.2 Comparison with SOTA

Parameter	Base-Paper([1])	1D/2D Deep CNN([3])	Our Approach
Amplitude Features	✓	✓	✓
Spectral Features	✗	✓	✓
Emsemble method	✗	✗	✓
Best accuracy	87%	88.3%	83.94%

6 Discussion and Future Work

6.1 Experiments

We have fine-tuned the following hyperparameters to get the best accuracy:

1. Percentage overlap between chunks: As figure-1 depicts that during generation of spectrogram images there is an overlap between two consecutive chunks. We tried multiple values of percentage overlap like 0%, 25%, 50%, 75% and we got best results on 75% overlap.
2. Loss function: We tried two loss functions in our approach
   (a) Mean squared logarithmic error: It can be interpreted as a measure of the ratio between the true and predicted values.
   $L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2$
   (b) Categorical cross-entropy : It calculates the loss of an example by computing the following sum:
   $Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i$
   where $\hat{y}_i$ is the i-th scalar value in the model output, y_i is the corresponding target value, and n is the number of scalar values in the model output.
   (c) Categorical cross-entropy gave us best results.
3. Batch size : We tried experimenting various batch size ranging from 32,34,128,256 and at the end we concluded that 128 batch is giving best results.
4. Learning rate: We fine tuned our model by trying various learning rates like 0.001, 0.0001, 0.0005, 0.00001, 0.00005. We finalized 0.0001.
5. Spectrogram dimensions : During spectrogram images generation we tuned the image resolution of spectrogram by trying various sizes size 72x72, 224x224 and 72x72 gave us best results.

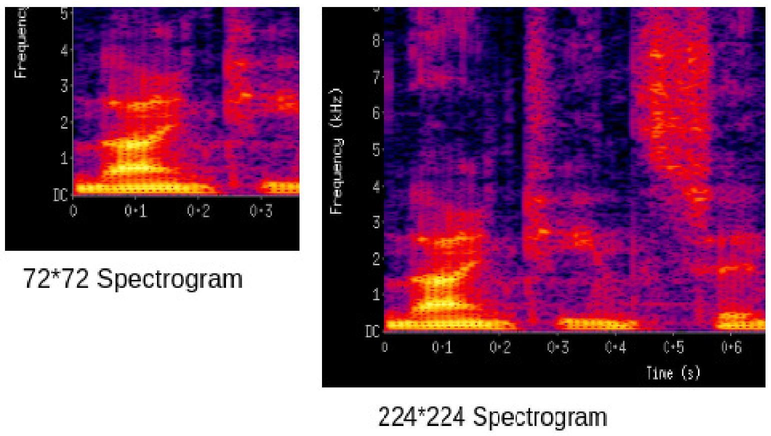


Figure 8: Various Spectrogram sizes

6. Ensemble model weights: During generation of final output class we have final output class equation as follows
   $\longrightarrow$ Weight formula: W * 1DModel + (1-W) * 2DModel
   $\longrightarrow$ Where W is weight ranging from (0-1), which depict how much preference is give to 1D CNN and (1-W) depicts weight of 2D CNN output.We tried various values and W=0.4 gave us best result. Figures 9 and 10 drawn below makes this concept more clear.

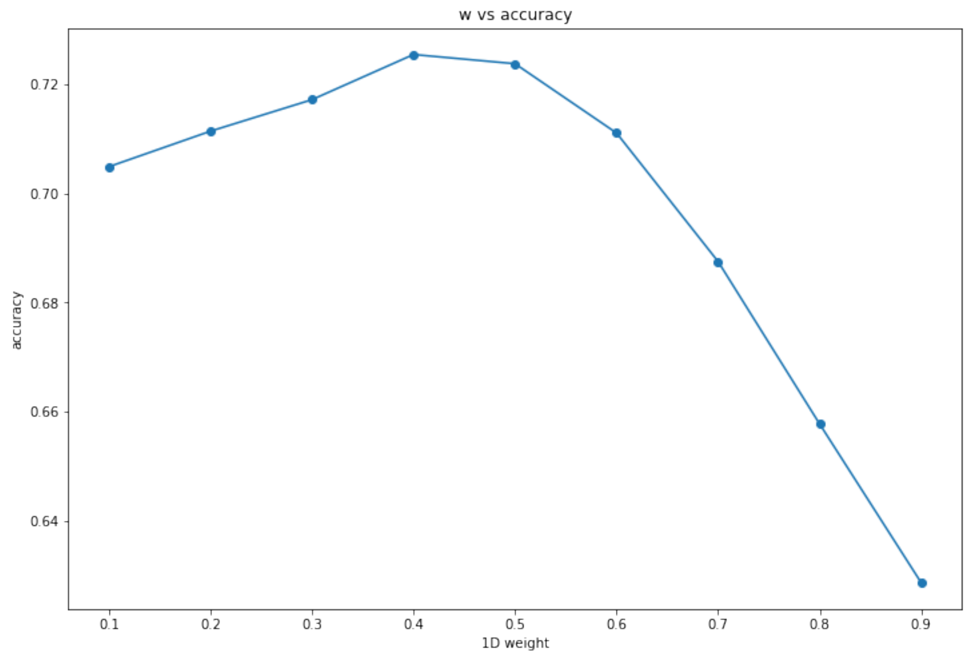


Figure 9: Weight v/s accuracy plot for ensemble with normal CNN

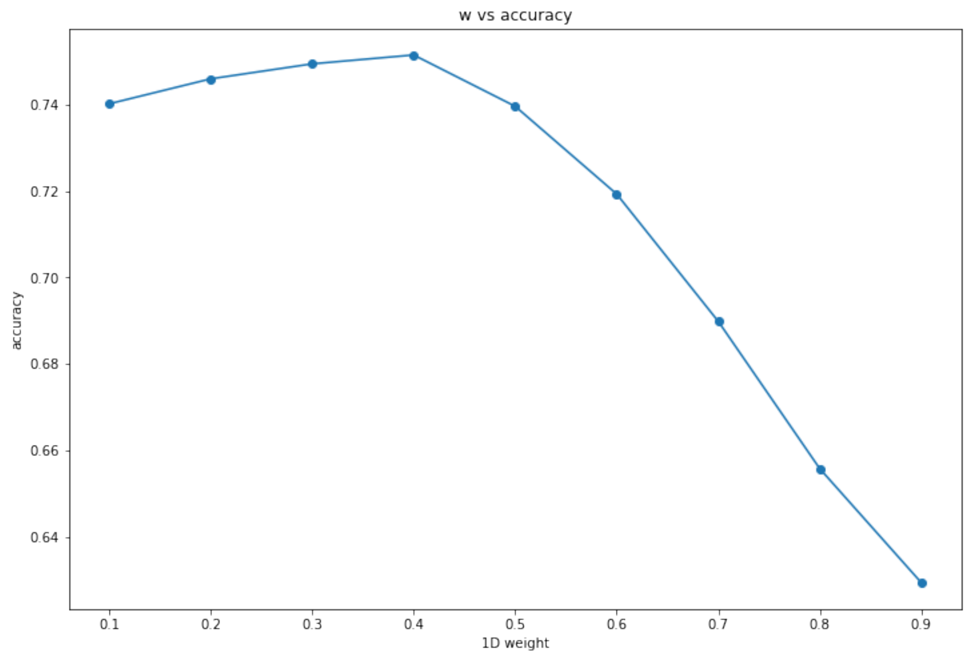


Figure 10: Weight v/s accuracy plot for ensemble with VGG CNN

6.2 Future Work

1. Collecting more data samples to make model robust.
2. Use of CRNN (Convolutional Recurrent Neural Network) for spectrogram images classification.
3. Data augmentation using time streching, pitch shifting, dynamic range compression or adding background noise.

7 Conclusion

We have trained an audio classifier using an ensemble method, where we combined two Convolution networks to generate a more accurate classifier. We combined 1D and 2D CNN to take advantage of two different features of audio signals (amplitude and spectral features). We were able to significantly improve the prediction accuracy by joining the two models. We can clearly conclude from the results that extracting information using different features boosts the accuracy. We were able to improve the prediction by around 13% for 1D CNN and around 3% for 2D CNN.

8 Individual Contributions

Team member	**Abhinav Dudeja**	**Arpan Kapoor**	**Piyush Gangle**
Literature Survey	✓	✓	✓
1D CNN	✓	✗	✗
2D CNN	✗	✓	✗
VGG transfer learning	✗	✗	✓
Ensemble	✓	✓	✓
Hyperparameter tuning	✓	✓	✓
PPT and report	✓	✓	✓

References

[1] S. Abdoli1, P. Cardinal, and A. L. Koerich, "End-to-end environmental sound classification using a 1d convolutional neural network," *arXiv*, vol. abs/1512.03385, 2019.

[2] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," arXiv, 2016.

[3] L. Vrysis, N. Tsipas, I. Thoidis, and C. Dimoulas, "1d/2d deep cnns vs. temporal feature integration for general audio classification," *Audio Engineering Society*, vol. 10.17743/jaes.2019.0058, 2020.

[4] J. Salamon*, C. Jacoby*, and J. P. Bello*, "Urban Sound Datasets," ACM International Conference on Multimedia, 2014.