EE5180

CNN based Music Instrument Classification

Instructor : Avhishek Chatterjee

Group 4

Anirudh Sriram, Arjun Menon V, Srinivas Mareddi, Nithin Varma
ee18b{073, 104, 057, 052}@smail.iitm.ac.in

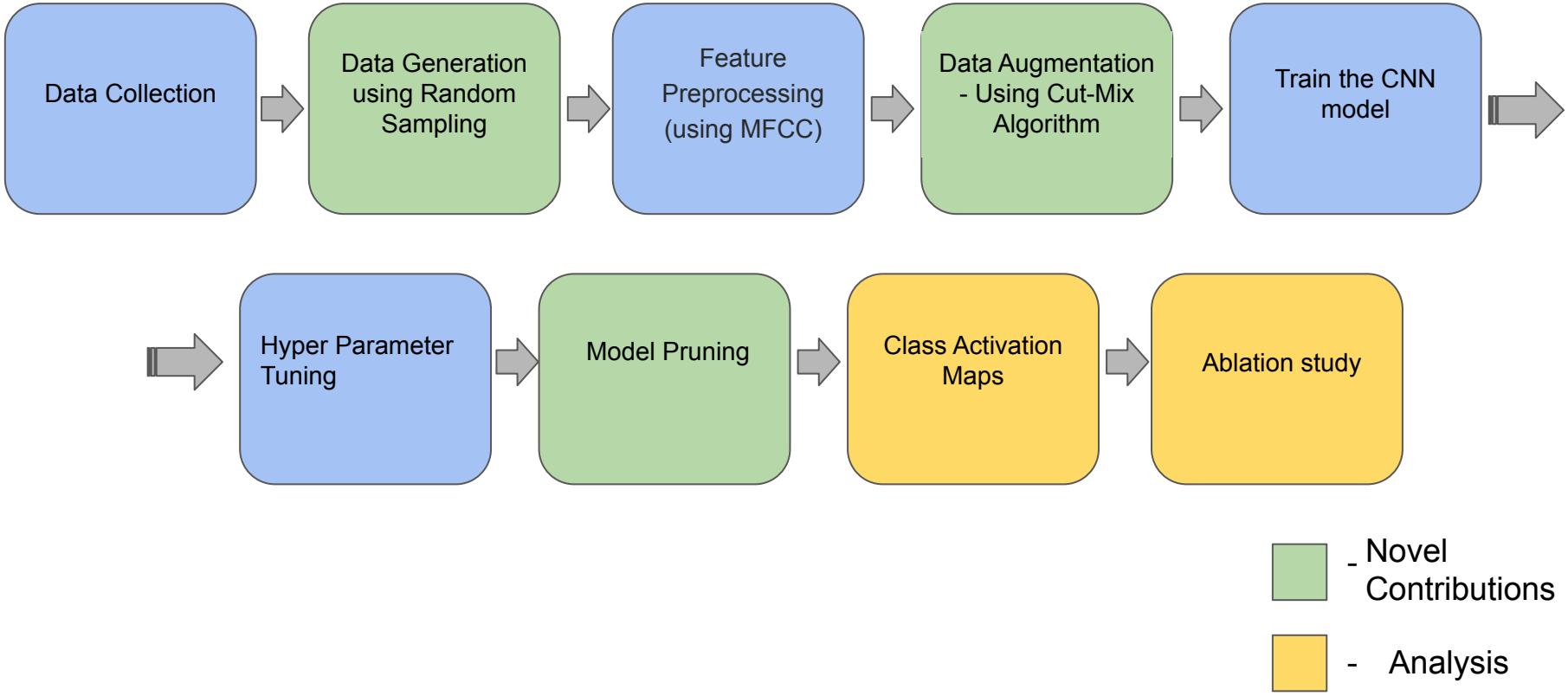
Problem

- Different music instruments exhibit different trends in their audio features, it is difficult to devise a standard approach to classify musical instruments using traditional DSP methods.
- Most of the variation in instrument sounds comes not from the range of notes that the instrument can play, but from the overtones produced by playing the instrument. The overtones present are determined by the physical characteristics of the instrument, which vary between instruments, and are not necessarily straightforward to model.
- While relevant features of the sounds generated by musical instruments can be captured by STFT based spectrograms, popular instrument classifiers have used learning based approaches (SVM, Hidden Markov Models, Neural Networks) for the classification task.

Motivation

- Music Instrument classification enables the extraction of valuable information that in turn could contribute to tasks like music classification by genre, music search by instrument, content based recommender systems and similar problems in other domains such as anomaly detection for mechanical systems.
- Music Information Retrieval (MIR) is the interdisciplinary research focused on retrieving information from music. Instrument classification, a polyphonic challenge, is one of the cornerstone problems of MIR.

Road Map

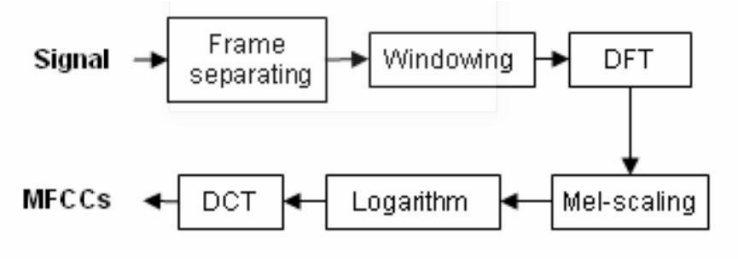

Data Collection
Data Generation using Random Sampling
Feature Preprocessing (using MFCC)
Data Augmentation - Using Cut-Mix Algorithm
Train the CNN model
Hyper Parameter Tuning
Model Pruning
Class Activation Maps
Ablation study
- Novel Contributions
- Analysis

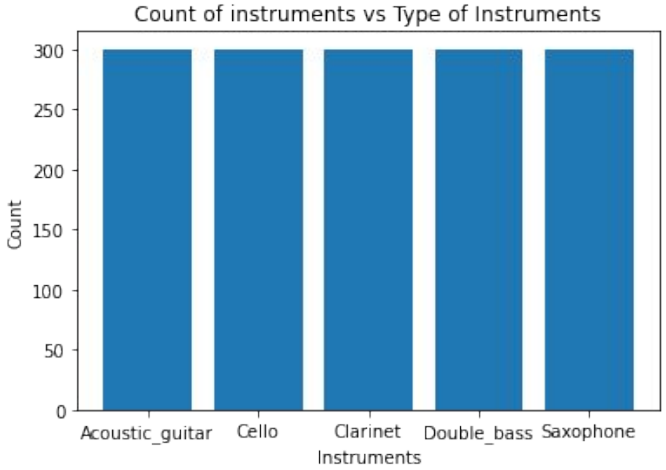
Dataset and Preparation

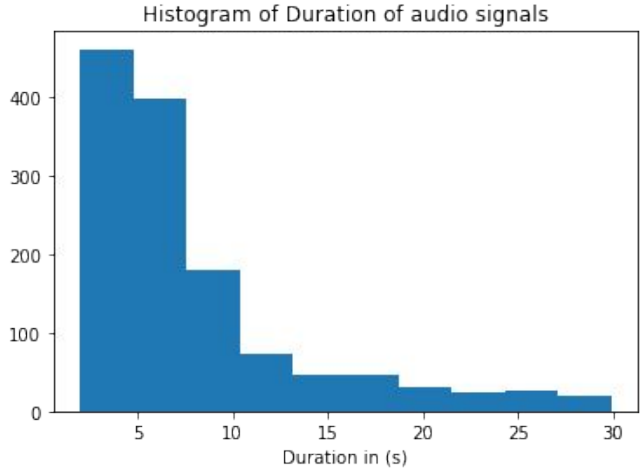
- We have taken an open source dataset called ‘Free Sound Audio tagging data’ for classification of musical instruments. We filter out the Top 5 most commonly occurring music instrument classes to make a dataset with 1500 samples.
- Next we remove the samples with audio content less than 2s.This can be reasoned due to the fact that there will not be sufficient data for the model to learn from audio files
- Next to make use of files with more than sufficient audio content we make use of random sampling algorithm to obtain random chunks of required threshold length from each data, and hence create 6000 more samples.

Feature Extraction

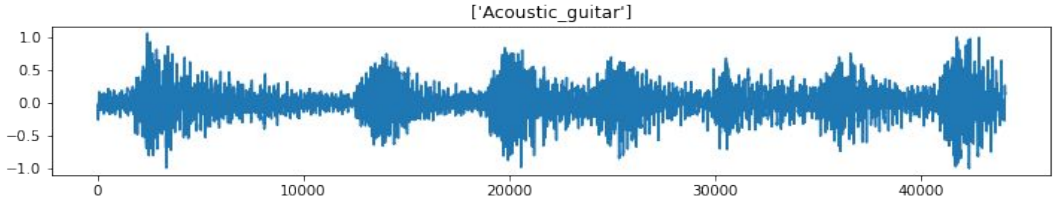
- The Mel Frequency Cepstral Coefficients (MFCCs) of each music piece from the generated data is extracted using Librosa.
- For each audio file, its MFCCs are averaged to produce the final feature vector in which only 13 of the 26 DCT coefficients are kept.
- Now the shape of mfcc_features represents the number of audio data and the heatmap image with the size of 275 times 13 produced using mfcc() function.
- The most time-consuming part of the system is the MFCC feature extraction. However, it still needs less than a second to process a one-second audio, which makes it feasible to do real-time instrument detection.
- Next we label-encode by one-hot-encoding the labels of each sample.



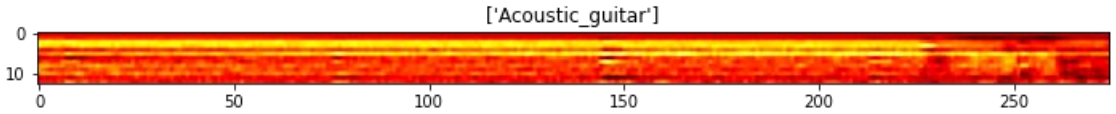




- Sample of Generated Audio



- Sample of MFCC Feature Plot



Proposal

- We proposed a **CNN** based classification model to classify the input spectral features into the class of instruments.
- We also have proposed a **data generation technique** in **Random sampling** of audio files and **data augmentation** technique in **Cut Mix algorithm.**
- We perform **Hyperparameter tuning** for the model parameters of the CNN model using **Keras-tuner** and **Random search algorithm**.
- We perform **Model Pruning** to determine the importance of connections and remove the unwanted connections.
- We also plot **Class Activation Maps** for visualising the significance of the features at the input to the model.
- We conduct an **Ablation Study** to evaluate the effects of the NN layers on the model's accuracy.

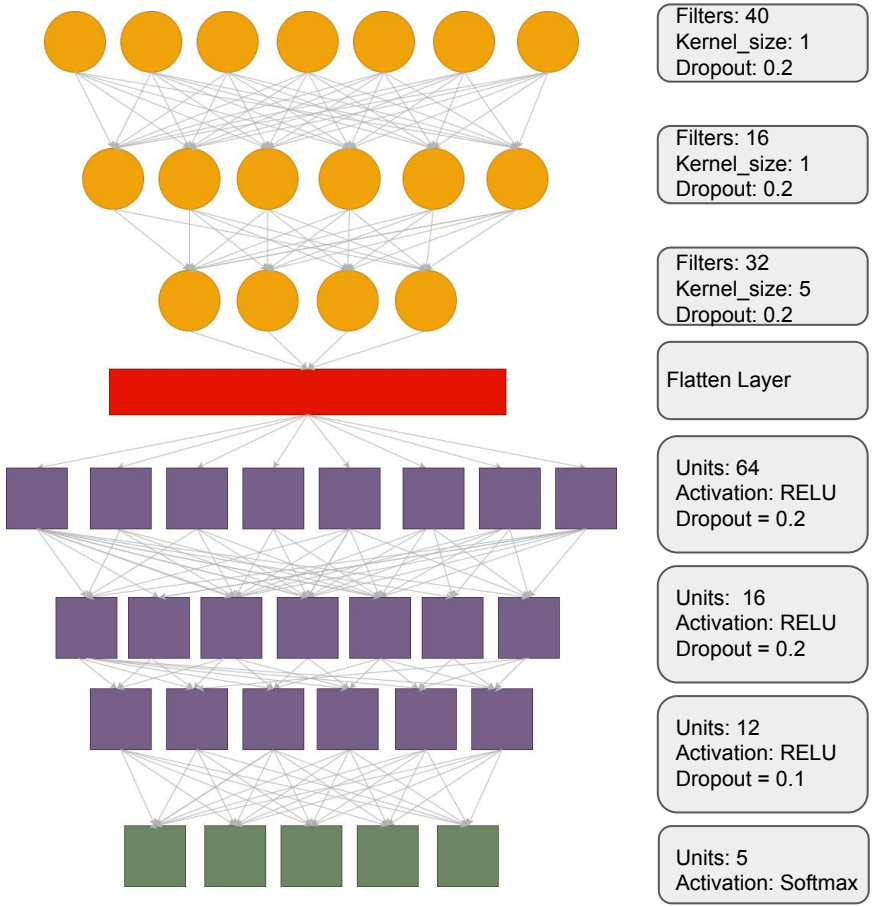
CNN Model

The proposed model comprises of 3 convolutional layers and 3 Dense layers,parameters of which are tuned with Keras tuner.

The model takes in a 275*13 dimensional array as input and outputs a 5*1 array predicting the probability of each class.

Model consists of Dropout in between all the layers to prevent overfitting. We also make use of Model checkpoints to revert back to point with max validation accuracy during training.

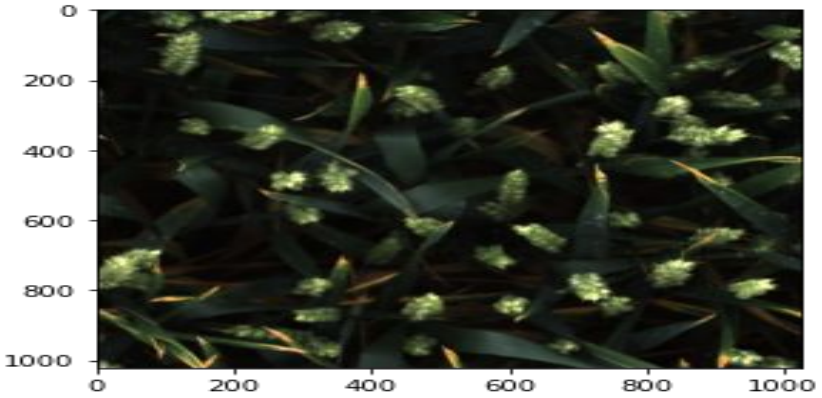
All the layers use RELU activation and Adam optimizer with initial learning rate of 0.001.

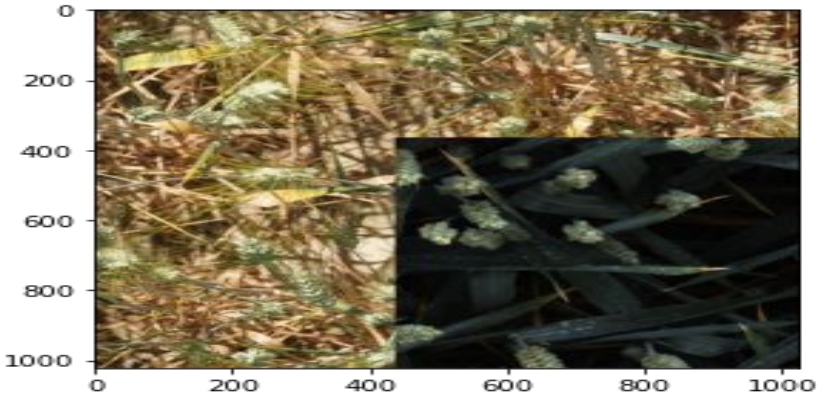


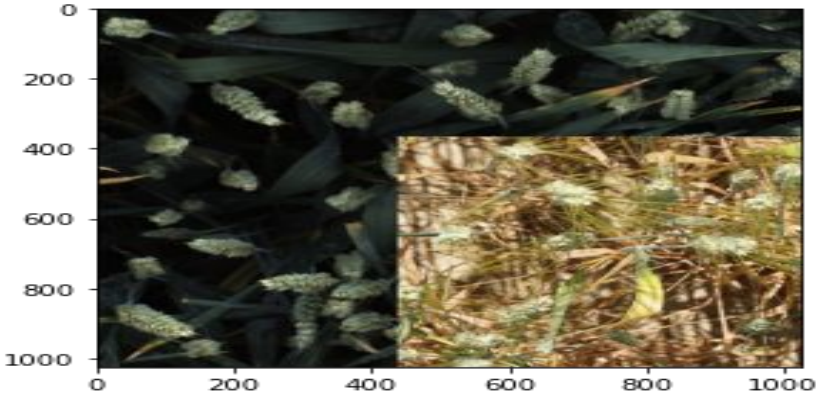
Data Augmentation- Cut Mix Algorithm

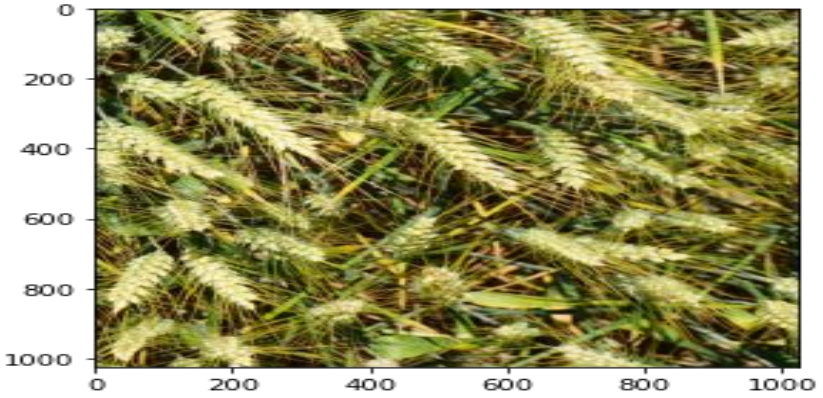
- The preprocessed data MFCC has a dimension of (275,13) , there the input data can be treated as an image.
- In Cut Mix augmentation we cut and paste random patches between the training samples. The ground truth labels are also mixed in proportion to the area of patches in the input arrays.
- Cut Mix increases localization ability by making the model to focus on less discriminative parts of the object being classified.
- By making efficient use of training data and retaining the regularization effect of regional dropout,this method outperforms existing methods. Therefore it increases the robustness of the model.

Samples from Cut Mix Image generation









Results and Inferences

Metrics Used : What are accuracy, precision, recall, F1 score ?

- **True Positives (TP)** - These are the correctly predicted positive values, that is the predicted class is the same as actual class.
- **True Negatives (TN)** - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

- **False Positives (FP)** – When actual class is no and predicted class is yes. These are the cases in which the input is misclassified as the class in consideration.
- **False Negatives (FN)** – When actual class is yes but predicted class in no. These are the inputs which are actually the class in consideration but are misclassified.

- **Accuracy** = (TP+TN)/(TP+FP+FN+TN)
- **Precision =** TP/(TP+FP)
- **Recall** = TP/(TP+FN)
- **F1 Score =** 2*(Recall * Precision) / (Recall + Precision)

Confusion Matrix for CNN + CutMix Model

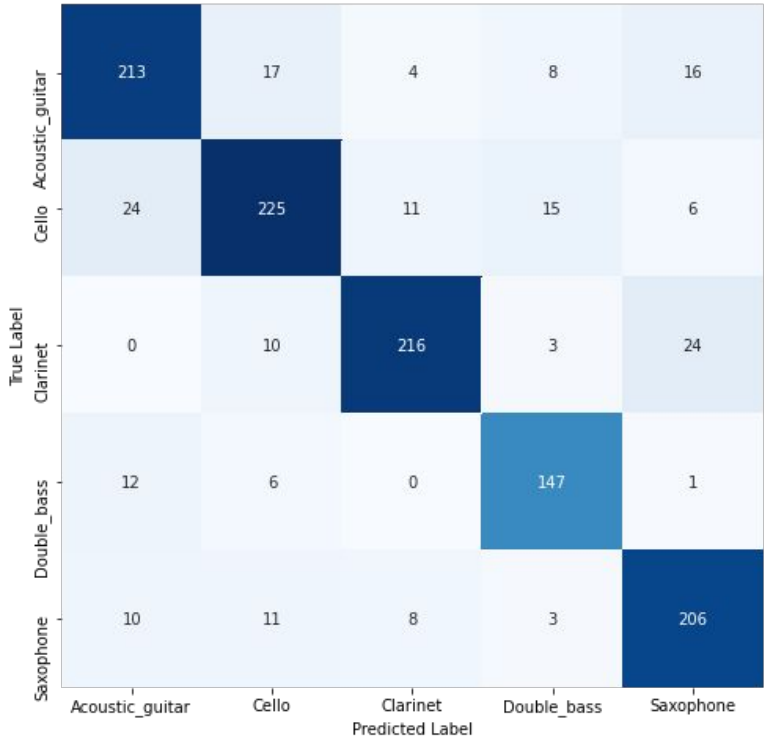
The figure to the right is the confusion matrix for the above mentioned model.

Instrument Classification Accuracy is about

- **85%** for **Guitar**
- **80%** for **Cello**
- **85%** for **Clarinet**
- **89%** for **Double Bass**
- **87%** for **Saxophone**

We can also have other inferences such as

- A **guitar** is **more likely** to be misclassified as **Cello** and vice versa.
- Also a **guitar** is very **less likely** to be classified as a **Clarinet** and vice versa or a **Saxophone** to be **confused** with a **Double Bass**.
- This indicates that some of these instruments are more closely related to others in terms of their spectral features.



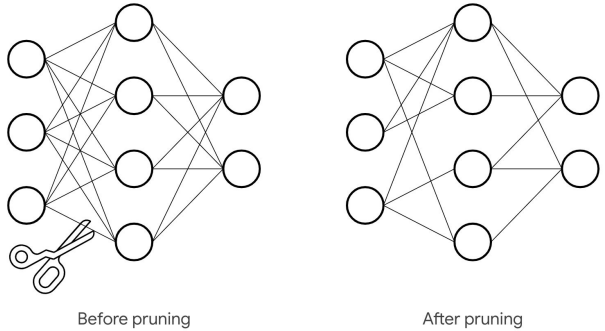
Hyperparameter Tuning

- Optimal hyperparameters for the model were chosen using the Keras Tuner API (Random Search algorithm)
- Hyperparameters that were tuned were the number of filters and kernel size for the convolution layers, and the number of units in FC layers and learning rate of optimizer.
- Comparison with other Search algorithms:
    - In contrast to sklearn.GridSearchCV, Random Search goes through only a fixed number of hyperparameter settings in a random fashion, reducing unnecessary computation
    - Keras Tuner also supports search algorithms such as Hyperband and Bayesian Search:
        - search iteratively, evaluating performances from previous iterations to improve subsequent iterations
        - approach the optimal hyperparameters faster

Hyperparameter	Parameter Space	Optimal Parameter
Conv_1_filter	16 to 64 ; step_size=8	40
Convo_1_kernels	[1,3,5]	1
Conv_2_filter	16 to 32 ; step_size=4	32
Conv_2_kernel	[1,3,5]	5
Conv_3_filter	8 to 16 ; step_size=2	16
Convo_3_kernels	[1,3,5]	1
Dense_1_units	16 to 64 ; step_size=8	64
Dense_2_units	8 to 16 ; step_size=4	16
Dense_3_units	4 to 8 ; step_size=2	12
Learning Rate	[1,0.1,0.01,0.001]	0.001

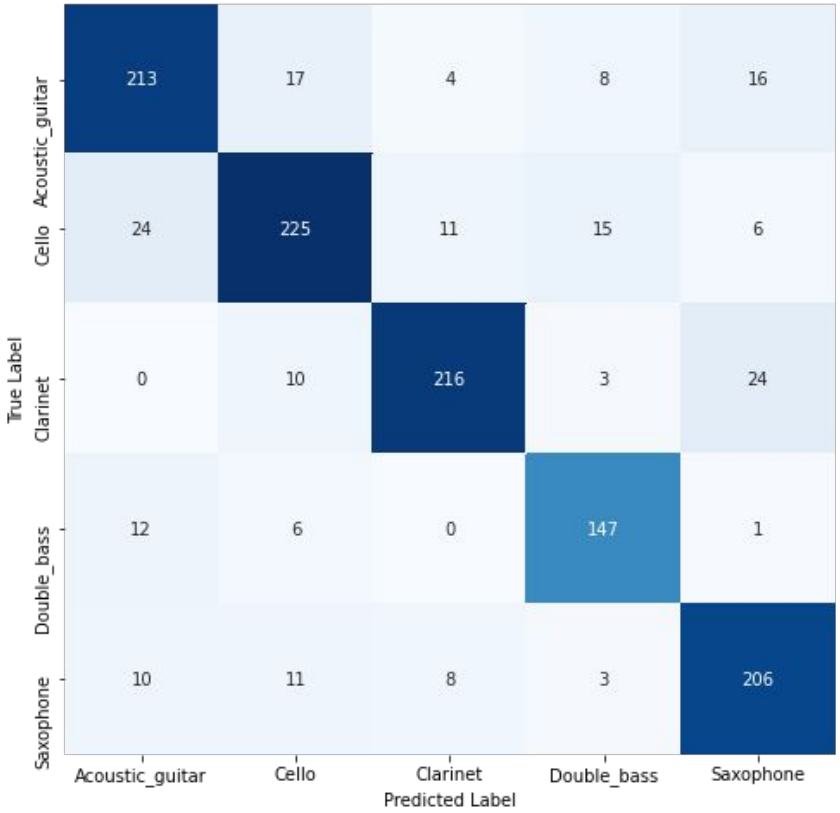
Pruning

- The bottleneck for Neural Networks is the number of parameters required to train.
- Pruning is used to reduce the number of parameters and operations involved in the computation by removing connections, and thus parameters, in between neural network layers.
- In weight pruning we are practically setting the neural network parameters' values to zero to remove what we estimate are unnecessary connections between the layers of a neural network. This is done during the training process to allow the neural network to adapt to the changes.
- We start off with a 50% sparse neural network and keep increasing the sparsity till we reach the minimum performance threshold.

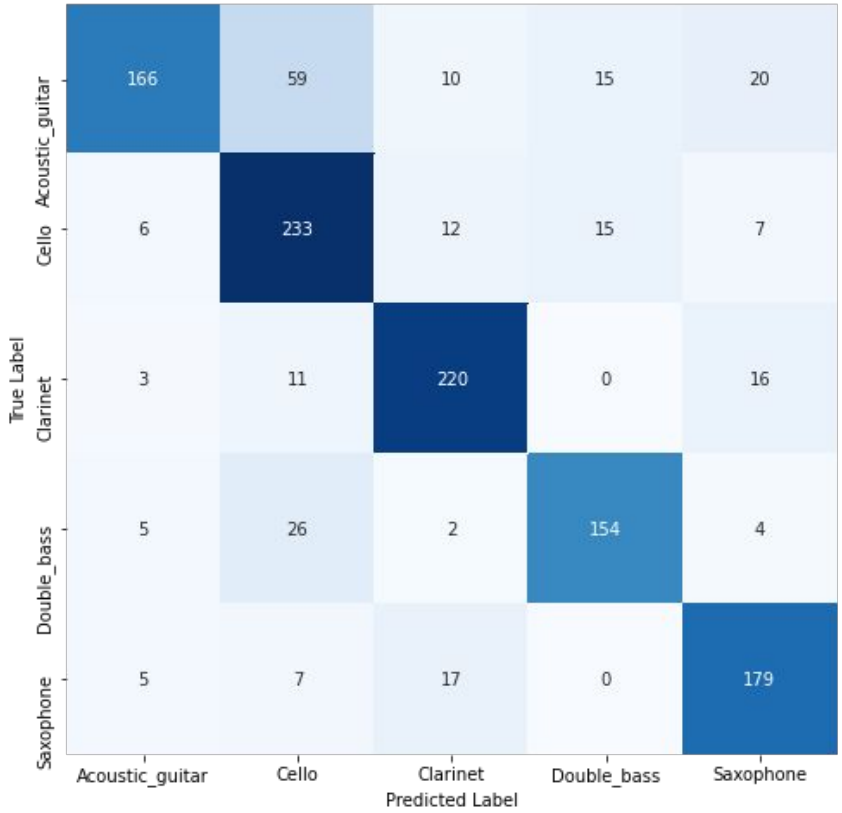


	Accuracy	**No. of Trainable parameters**
Model without pruning	0.86	3,671,892
Model with pruning	0.83	1,835,989

Confusion Matrix before Pruning



Confusion Matrix after Pruning

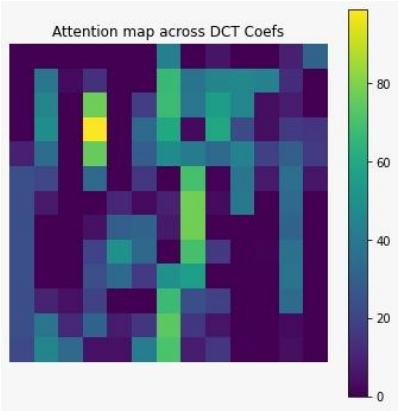


Comparison with Other Models

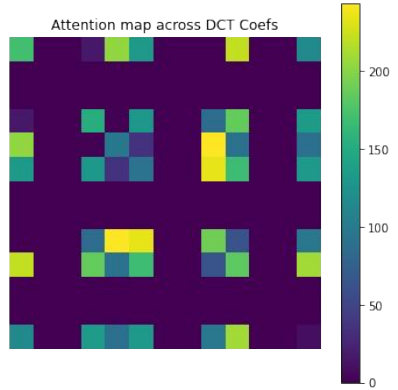
Model Name	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.68	0.68	0.68	0.68
Naive Bayes	0.58	0.62	0.6	0.58
SVM	0.71	0.71	0.71	0.71
Decision Tree Classifier	0.64	0.64	0.63	0.64
Random Forest	0.80	0.81	0.80	0.81
Adaboost	0.66	0.68	0.67	0.66
MLP Classifier	0.47	0.48	0.47	0.47
CNN (w/o CutMix)	0.79	0.79	0.80	0.79
CNN + CutMix	**0.86**	**0.86**	**0.85**	**0.85**
CNN + CutMix + Pruning	**0.83**	**0.83**	**0.82**	**0.82**

Class Activation Maps

- Class Activation Maps help us visualise the significance of the features at the input to the model.Grad-CAM is used, which can create heatmaps by visualizing the important of input pixels from the output of any convolutional layer in the network.
- The gradient of the output class value with regards to each channel in the feature map of a certain layer is calculated. This results in a gradient channel map with each channel in it representing the gradient of the corresponding channel in a feature map.
- The gradient channel map obtained is then global average pooled, and the values obtained henceforth are the weights of importance of each channel in the feature map. The weighted feature map is then used as a heatmap.



A sub matrix of the attention map across the DCT coefficients



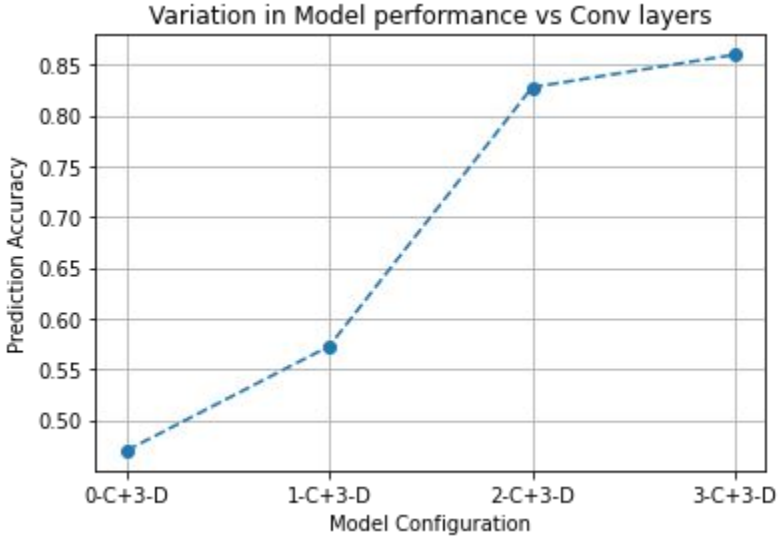
Inner Product of Columns (<col i, col j>) of the Heatmap

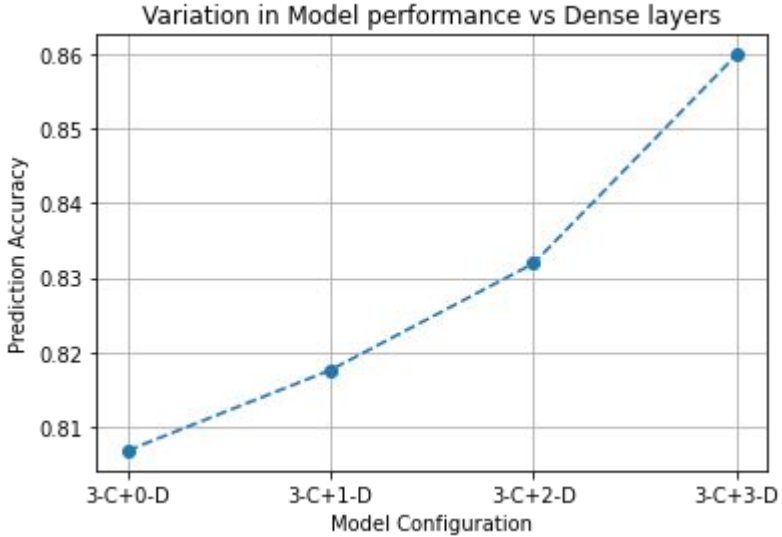
Ablation Study

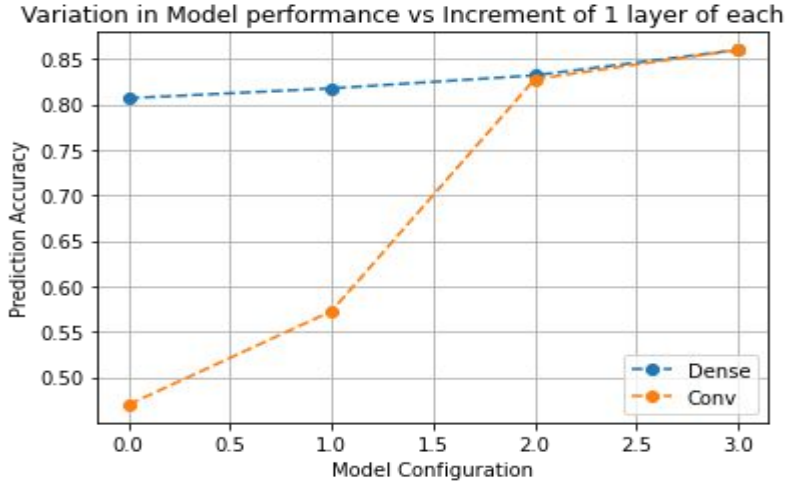
- How significant is each layer in predicting the class accurately?
- To estimate this qualitatively, we remove some intermediate layers from the model and train the network and observe the corresponding accuracy
- Fixing the number of dense layers, the number of conv layers is reduced from 3 to 0; and vice versa
- Insights from Ablation Study:
  - Removing convolutional layers takes a big hit on the accuracy of the model
  - This is as expected- conv layers identify the spatial information in the MFCC matrix.
  - Each dense layer improves the accuracy significantly- better approx. of the function mapping output of conv layers to softmax

Ablation study	Accuracy
No Conv Layer + 3 Dense Layers	0.4698
1 Conv Layers + 3 Dense layers	0.5727
2 Conv Layers + 3 Dense layers	0.8277
3 Conv Layers + 3 Dense Layer	0.8599
3 Conv Layers + No Dense Layer	0.8068
3 Conv Layers + 1 Dense Layer	0.8176
3 Conv Layers + 2 Dense Layers	0.8319

Ablation Study Results


Variation in Model performance vs Conv layers
Prediction Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0-C+3-D
1-C+3-D
2-C+3-D
3-C+3-D
Model Configuration


Variation in Model performance vs Dense layers
Prediction Accuracy
0.86
0.85
0.84
0.83
0.82
0.81
3-C+0-D
3-C+1-D
3-C+2-D
3-C+3-D
Model Configuration


Variation in Model performance vs Increment of 1 layer of each type
Prediction Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Model Configuration
Dense
Conv

Thank You

Highlights:

- Novel Data Augmentation Technique (Cut Mix)
- Model Pruning (Lightweight Model)
- Class Activation maps (Grad CAM) to identify Class specific important data
- Thorough Comparative Analysis, Ablation study and efficient Hyperparameter Tuning
- About 7% improvement over existing models.

Link to Repo: https://github.com/anirudhs123/EE5180-Group-4

Any Questions ?