Genre-Based Audio Classification: Analyzing Audio Content Categorization

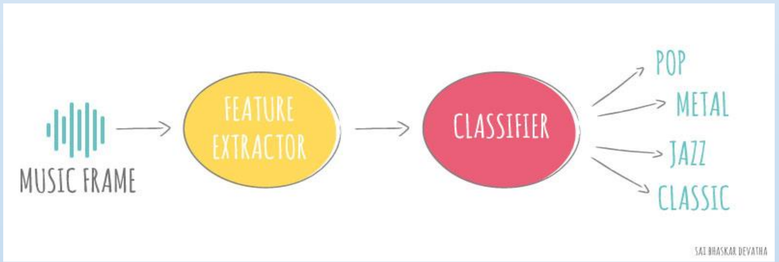
Ankur Kumar (Dept: EE)
ankurk21@iitk.ac.in

Dr. Vipul Arora (Dept: EE)
vipular@iitk.ac.in

1. Problem Statement

The lack of efficient genre-based classification of audio content in **All India Radio (AIR)** and India hinders effective content management and user experience. We use the **Prasar Bharti Hindi dataset** for the training.

Aimed at improving content organization and recommendation systems in AIR, this research project envisions to create a **Genre-based Recommendation System** that can be made into a model that keeps evolving with every new addition to the Radio archives.



Classification Model Approach

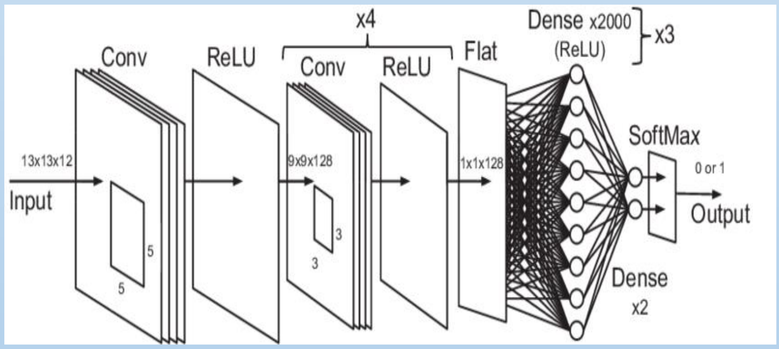
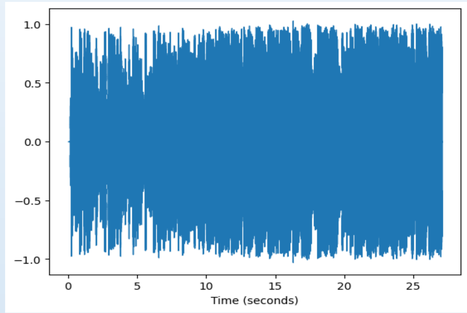
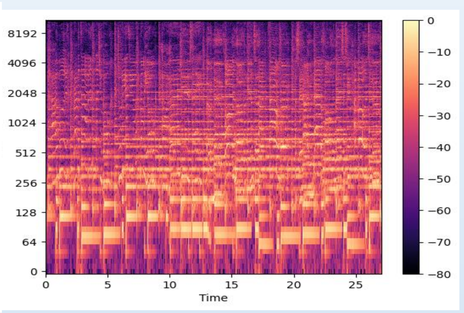


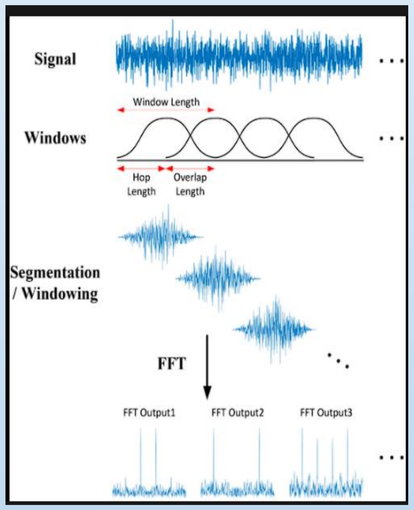
Diagram: Convolutional Neural Network Classifier

1. The CNN model architecture takes as input the tensor matrix of *spectrogram.jpg* image file which is first sent through series of five **Conv2D** layers (with **ReLU** activation) with filters of exponentially increasing size.
2. The output of final conv2D layer is flattened and passed through **"Dense" layers** and finally a **Softmax** activation to produce output layer with Genre probabilities.

Implementation and Analysis







1. Created the MEL spectrogram via **Fourier Transform** (STFT/FFT) for the *audio.wav* files using *Librosa* library.
2. The spectrogram is plotted, converted into an image and passed into **CNN** architecture as (256x256x3) tensor of RGB pixel values in the plot.
3. We use **ReLU** and **Softmax** activation layers to introduce non- linearity.
4. The **Cross Entropy** Loss Function is used since this is a case of Multi-class categorization involving probabilities.

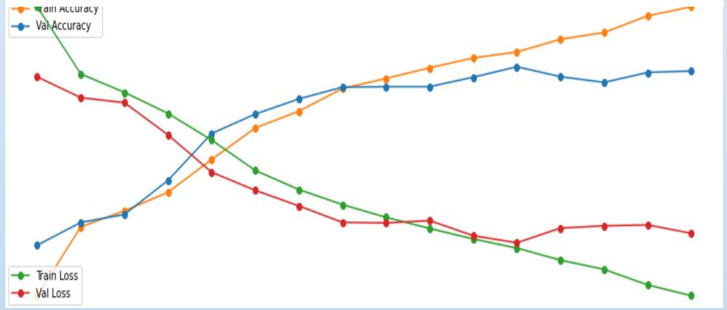
```
model = tf.keras.models.Sequential(
    [
        tf.keras.layers.Rescaling(1./255, input_shape = (config['img_size'], config['img_size'], 3)),
        tf.keras.layers.Conv2D(16, kernel_size=5,  activation='relu'), #16 filters, 5x5 kernel
        tf.keras.layers.Conv2D(32, kernel_size=3, activation='relu'),
        tf.keras.layers.Conv2D(64, kernel_size=3, strides= 2, activation='relu'),
        tf.keras.layers.Conv2D(128, kernel_size=3,  activation='relu'),
        tf.keras.layers.Conv2D(256, kernel_size=3, strides= 2,  activation='relu'),
        tf.keras.layers.Conv2D(512, kernel_size=3, strides= 2, activation='relu'),
        tf.keras.layers.GlobalAveragePooling2D(), #to reduce the number of parameters
        tf.keras.layers.Flatten(), #to flatten the input into 1D array
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(64, activation='relu'), # 64 neurons connected to alla neurons in the
        tf.keras.layers.Dropout(0.2), #to prevent overfitting 20% of the neurons are dropped
        tf.keras.layers.Dense(config['num_labels'], activation='softmax')
    ]
)
```

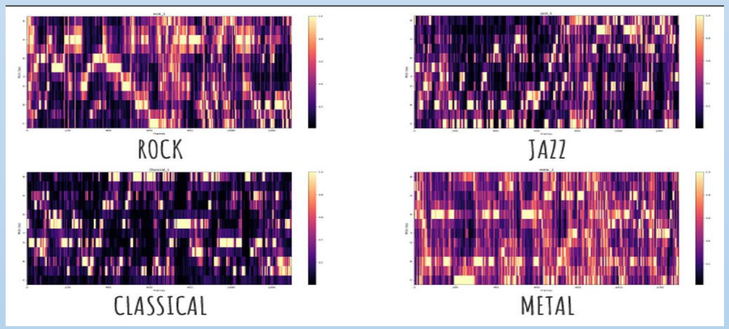
Code: Convolutional Neural Network model & Loss Function

Observations and Results

```
63/63 [==============================] - 21s 314ms/step
              precision    recall  f1-score   support

           0       0.35      0.73      0.47       299
           1       0.37      0.40      0.39       284
           2       0.40      0.48      0.51       279
           3       0.44      0.30      0.35       191
           4       0.43      0.67      0.65       171
           5       0.47      0.50      0.46       117
           6       0.52      0.36      0.43       113
           7       0.56      0.30      0.32        99
           8       0.60      0.00      0.00        86
           9       0.63      0.37      0.39        83
          10       0.67      0.02      0.04        97
          11       0.65      0.61      0.57        51
          12       0.68      0.91      0.92        43
          13       0.70      0.00      0.00        31
          14       0.67      0.00      0.00        20
          15       0.66      0.00      0.00        10
          16       0.69      0.25      0.40         8
          17       0.69      0.00      0.00         9
```





1. After 40 epochs, the model plateaued at an **efficiency** of **69%**.
2. Above is example spectrogram for songs from their respective genres. This after passing through CNN layers produces patterns that is picked up by neuron Dense Layers.
3. We also observed in ***[1]*** that the accuracy is inversely dependent on the pitch resolution of audio signals. We can extend this to a **MC-DNN** and incorporate multiple features like beats, pitch, chroma etc. in parallel CNN channels and combine them in final step.

References:

[1]. Sangeun Kum, Melody Extraction on Vocal Segments using MC-DNN
[2]. MDan-Ning hang, Lie Lu, Music type classification by Spectral Contrast
[3]. George Tzanetakis, Automatic Musical Genre Classification of Audio Signal