Project Report
CNN-based Audio Classification Models

Data Preprocessing and Augmentation:

1. Feature Extraction: Relevant features are extracted from audio samples using Mel-frequency cepstral coefficients (MFCCs). Both were tested but MFCCs were preferred over mel spectrograms due to their superior accuracy. Additionally, padding is applied based on the largest spectrogram generated to ensure uniformity in feature dimensions.
2. Data Augmentation: Initially, data augmentation techniques such as time stretching, pitch shifting, and random noise addition were explored. However, these augmentations did not lead to a significant increase in accuracy and were computationally heavy for the google collab notebook. Therefore, data augmentation was omitted.

Architecture 1: 5-Layer DenseNet

This architecture is based on DenseNet-121 comprising dense blocks and transition layers, totaling five layers. Each dense block consists of two convolutional layers. The dimensions of the feature maps change as follows:

1. Initial Convolutional Layer:
    a. Input Dimensions: 20x365x1 (Height x Width x Channels)
    b. Output Dimensions: Same as input due to 'same' padding.
2. Dense Blocks:
    a. Within each dense block:
        i. Output Dimensions: Same as input.
        ii. The output of each convolutional layer is concatenated with the input, leading to an increase in the number of channels.
3. Transition Layers:
    a. After each dense block, a transition layer is introduced.
    b. These layers reduce the spatial dimensions (height and width) of the feature maps while maintaining the number of channels.
    c. Output Dimensions: Half the height and width of the input, with the same number of channels.
4. Global Average Pooling Layer:

    a. After the final dense block, global average pooling is applied.
    b. This operation reduces the spatial dimensions to 1x1 while preserving the number of channels.
5. Output Layer:
    a. Fully connected layer with softmax activation.
    b. Output Dimensions: Correspond to the number of classes (13 in this case).
6. Hyperparameters:
    a. growth_rate = 32 and compression_factor = 0.5. These values determine the number of channels generated by each layer within the dense blocks and control the model's growth and computational complexity
    b. Network has 5 layers because having more than that was computationally infeasible.
    c. Adam optimizer is used with sparse categorical cross-entropy loss for training.
    d. Training is performed for multiple epochs with batch sizes of 16,32 and 64 and no improvement was observed so a value of 32 was used.

Training: It was trained for 50 iterations but significant change stopped after 25

Architecture 2: Modified AlexNet (LiteNet)

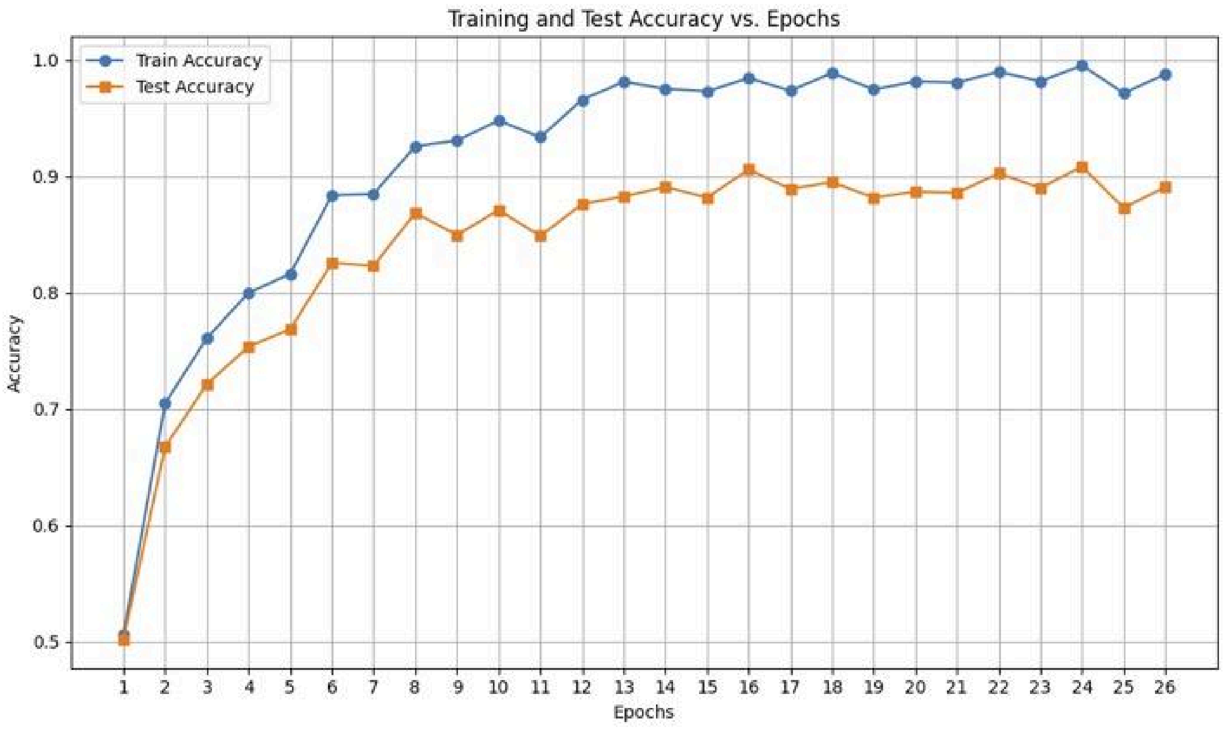
The architecture presented is based on AlexNet which was taught in class lectures.

1. Initial Convolutional Layers:
    a. Initial convolutional layer with 64 filters of size (3, 3) and ReLU activation.
    b. Output Dimensions: Same as input due to 'same' padding.
2. Pooling Layers:
    a. Max-pooling layers with a pool size of (2, 2) are employed to downsample the feature maps spatially.
3. Intermediate Convolutional Layers:
    a. Additional convolutional layers with increasing numbers of filters (128, 256, and 512) to capture more complex and abstract features.
    b. Each convolutional layer is followed by max-pooling to further downsample the feature maps.
4. Fully Connected Layers:
    a. Feature maps are flattened and fed into fully connected layers for classification.
    b. The model includes three dense layers with 512, 256, and 128 units, respectively, followed by ReLU activation functions.

    c. Dropout layers with a dropout rate of 0.5 are inserted after the first two dense layers.

5. Output Layer:
    a. Fully connected layer with softmax activation.
    b. Output Dimensions: Correspond to the number of classes (13 in this case).
6. Hyperparameters:
    a. Batch size: 128 performed slightly better than smaller batch sizes of 32, 64.
7. Comparison:
    a. The kernel sizes, number of filters, and pooling strategies may differ compared to AlexNet architectures.
    b. The architecture is scaled down to handle the specific input dimensions and task requirements, while maintaining computational feasibility.

Training: It was trained for 26 iterations but significant change stopped after 13 iterations



Results and Conclusion:

- Upon Testing we observed that the 5-Layer DenseNet achieves better performance. However LiteNet was much more computationally efficient with each epoch executing more than 5 times faster.
- 5-LayerDenseNet : (Mention Parameters)
- LiteNet : (Mention Parameters)