Deep Learning Assignment-1

Team Members:

Dev Kulkarni – 2021A7PS2430P

Radhey Kanade – 2021A7PS2534P

Shantanu Ambekar – 2021A7PS2540P

The 2 architectures used for the assignment are as follows: MobileNet and Resnet-18. These models and their hyperparameters were chosen after thorough experimentation as described below in this report.

MobileNet:

MobileNets are designed to be efficient**,** which means they can run on mobile and embedded devices with limited computational resources. Due to their efficiency, MobileNets are typically much faster than traditional convolutional neural networks and they have a small memory footprint, which makes them easy to deploy and use. MobileNets are widely used for image classification applications and can classify images with high accuracy. All these features of this architecture lead us to choose this as one of the architectures.

It employs depthwise separable convolutions, which significantly reduces the number of parameters and computations compared to traditional convolutions while still capturing spatial and channel-wise dependencies. We have used Batch normalization to stabilize the training process by normalizing the activations and ReLU activation is used to introduce non-linearity allowing the model to learn complex patterns. Instead of traditional fully connected layers at the end, this architecture opts for global average pooling, which reduces the number of parameters and serves as a form of regularization, preventing overfitting.

Description or the MobileNet architecture used:

- **Input Layer**: Accepts input images with a shape of (224, 224, 3), which is a common size for many image classification tasks.
- **Convolutional Layers**: The initial convolutional layer applies a 3x3 convolution with 32 filters and a stride of 2, followed by batch normalization and ReLU activation. This layer extracts basic features from the input image.
- **Depthwise Separable Convolutions**: This is the core building block of MobileNet. It consists of a depthwise convolution followed by a pointwise convolution. The depthwise convolution operates independently on each input channel, followed by a 1x1 convolution to combine channels. These convolutions are repeated multiple times to extract hierarchical features while maintaining computational efficiency.

- **Downsampling Layers**: Certain depthwise separable convolutions use a stride of 2 to downsample the spatial dimensions, reducing the size of feature maps and increasing receptive field.
- **Global Average Pooling**: After the convolutional layers, global average pooling is applied to reduce the spatial dimensions to a single vector for each feature map. This operation helps in reducing the number of parameters and provides a fixed-size representation regardless of the input size.
- **Output Layer**: The model ends with a dense layer with 13 units and softmax activation, producing probabilities for each class.

Resnet-18:

ResNet is designed to address the challenges of training very deep neural networks. With the incorporation of skip connections, it alleviates the vanishing gradient problem, enabling effective training of networks with many layers. Despite its depth, ResNet-18 maintains simplicity and efficiency compared to even deeper variants like ResNet-50 or ResNet-101. This makes it more accessible for training on standard hardware and suitable for applications where computational resources are limited. The skip connections in ResNet-18 facilitate the flow of gradients during backpropagation, which helps in training deeper networks more effectively. ResNet-18 often exhibits superior generalization performance compared to shallower networks or networks without skip connections. The ability to effectively capture both low-level and high-level features enables ResNet-18 to learn more discriminative representations, leading to better generalization to unseen data.

Description or the Resnet architecture used:

- **Input Layer**: The model accepts input images with a shape of (224, 224, 3), which is a standard size for many image classification tasks.
- **Preprocessing**: The input images are rescaled to values between 0 and 1 using rescaling layer.
- **Initial Convolutional Layer**: The model begins with a 2D convolutional layer with 64 filters, a kernel size of (7, 7), and a stride of (2, 2). This layer extracts basic features from the input images.
- **Batch Normalization and ReLU Activation**: Batch normalization and ReLU activation are applied after the initial convolutional layer to stabilize and introduce non-linearity to the features.
- **Max Pooling**: Max pooling with a pool size of (3, 3) and a stride of (2, 2) is performed to reduce the spatial dimensions of the feature maps while retaining important information.
- **Residual Blocks**: The core of the ResNet architecture consists of residual blocks. In the architecture used, there are four stages, each containing multiple residual blocks. Each residual block consists of convolutional layers with batch normalization and ReLU activation, along with skip connections to facilitate gradient flow.
- **Final Layers**: After the residual blocks, the feature maps are pooled using average pooling with a pool size of (2, 2). Then, the feature maps are flattened to a 1D vector. Two fully

connected dense layers with ReLU activation are applied for further feature extraction and transformation. Finally, a dense layer with softmax activation produces the output probabilities for the classification task.

- **Output Layer**: The output layer consists of a dense layer with a number of units equal to the number of classes in the classification task (in this case, 13), using softmax activation to produce the final class probabilities.

Experimentation:

Initially we tried using small general models as learned in class but their training accuracies were very low and so the model was underfitting by a huge margin. In Resnet architecture, we tried using an architecture inspired by Resnet-50. However, ResNet-50 has some limitations. One disadvantage is that it requires a large amount of computational resources and memory due to its deep architecture. ResNet-50. During our training, this architecture was overfitting, this could be due to several reasons such as less training data or a huge number of parameters, it had close to 90 million parameters. Due to this reason, we changed our architecture to one with few layers resulting in lesser number of parameters, we chose ResNet-18. This architecture was not overfitting and had good validation accuracy(close to 90%). We also tried many different combinations of batch size and number of epochs with both the architectures, i.e. MobileNet and ResNet to get an optimal fit so that it will generalize well to new data. After multiple trainings of resnet, we noticed that validation accuracy did not go above 88% so we set the early stopping threshold to 87%. Similarly, for MobileNet we have set the threshold to 90%

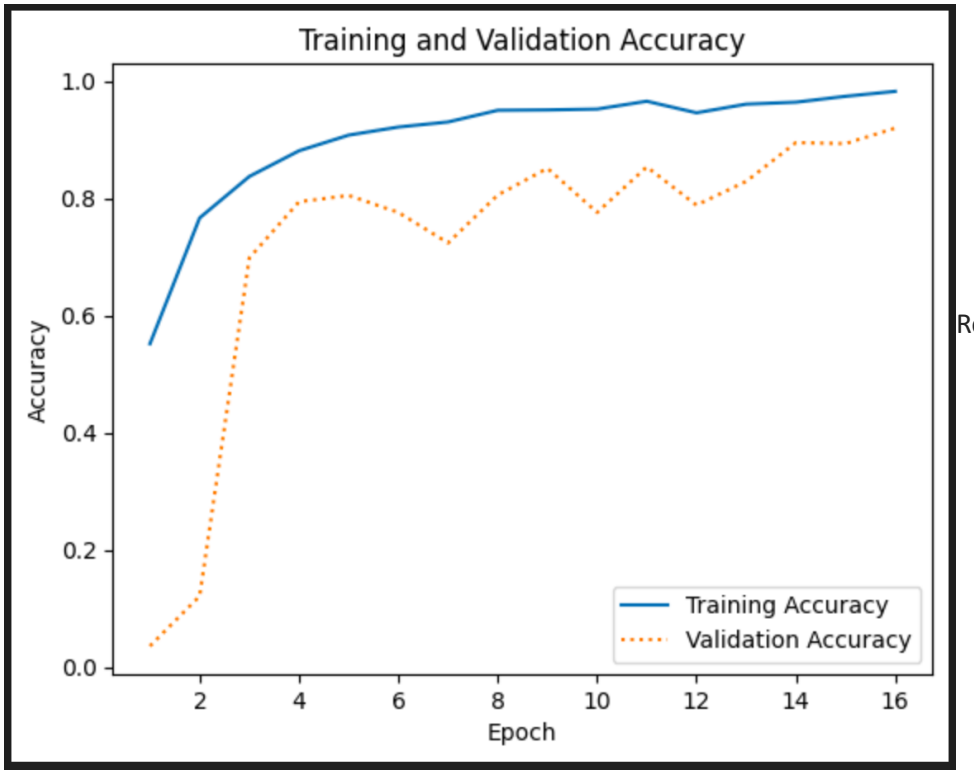
Observations:

MobileNet:

1. Accuracy: Training =98.25% and Validation=91.98%
2. Precision: :array([0.90217391, 0.90062112, 0.95652174, 0.980:76923, 0.97080292, 0.88617886, 0.90243902, 0.925 , 0.82051282, 0.95833333,0.93706294, 0.95575221, 0.76744186]),

3. Recall::array([0.97647059, 0.90625 , 0.94285714, 0.70833333, 0.97794118,0.97321429, 0.88095238, 0.90243902, 0.87671233, 0.86792453,0.95714286, 0.96428571, 0.76744186])

4. Training time: 5 minutes 39 seconds
5. No. of parameters: 2185101
6. Hyperparameters:3
7. Accuracy Curve:

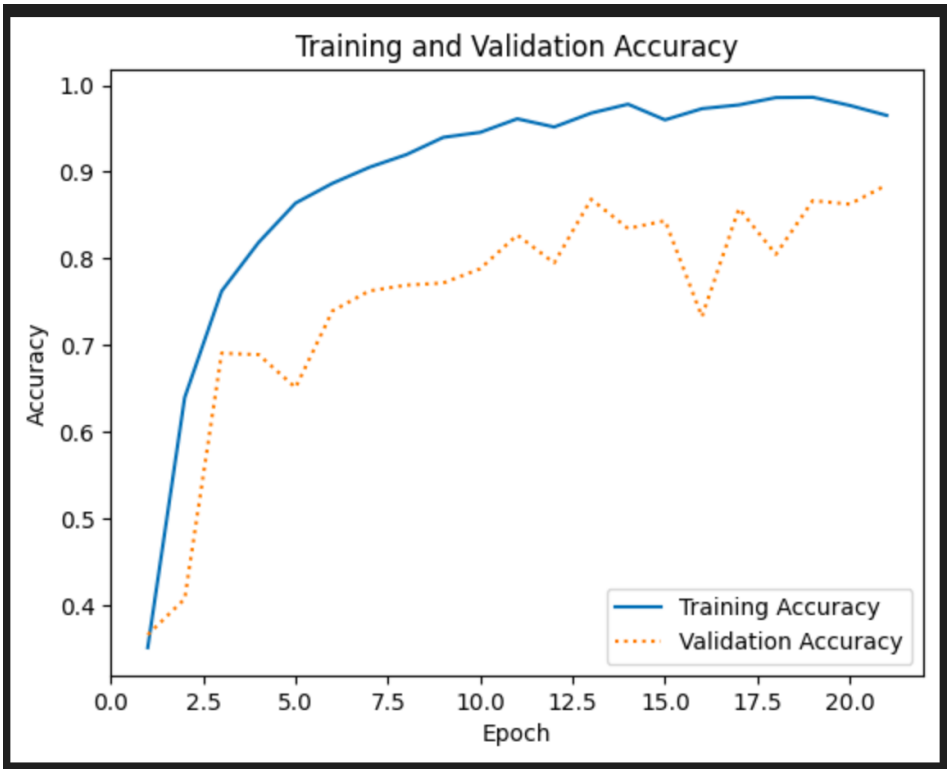


ResNet:

1. Accuracy: Training:96.52 and Validation:88.5%
2. Precision: array([0.9010989 , 0.79781421, 0.97560976, 0.88679245, 0.96875 ,0.89380531, 0.83333333, 0.90697674, 0.90322581, 0.89795918, 0.875 , 0.9375 , 0.63461538])

3. Recall: array([0.96470588, 0.9125 , 0.85714286, 0.65277778, 0.91176471,0.90178571, 0.95238095, 0.95121951, 0.76712329, 0.83018868, 0.95 , 0.9375 , 0.76744186])

4. Training time: 10m 52s
5. No. of parameters:19590285
6. Hyperparameters:3
7. Accuracy Curve:



Preprocessing: In preprocessing we use librosa to convert audio files into Mel spectrogram images. Since librosa cannot process .m4a files, all m4a files are first converted to .wav files using pydub.Audiosegment. Additionally, Laughter_284 is deleted from training since it is corrupted and was not getting transformed using librosa. Finally all images are resized to 224x224 with 3 channels (RGB) for uniformity.