EMBEDDED SYSTEMS

DL model optimization for Lightweight Gesture Recognition

Rahul Barodia B20CS047
Aman Mithoriya B20CS004

Dataset used: Gesture Recognition

The LeapGestRecog dataset available on Kaggle is a collection of hand gesture images captured using the Leap Motion Controller. It contains a total of 20,000 images of 10 different hand gestures, performed by 10 different subjects.

The 10 different hand gestures included in the dataset are:

- Fist
- Five
- Point
- Grab
- Pinch
- Ok
- C
- Down
- L
- Peace

The images are captured in varying lighting conditions, backgrounds, and angles, making the dataset challenging and diverse. The Leap Motion Controller captures hand movements and gestures in three dimensions, making it a useful tool for gesture recognition research.

Model used

```
model = keras.models.Sequential()

model.add(Conv2D(filters = 32, kernel_size = (3,3), input_shape = (IMG_SIZE, IMG_SIZE, 1)))
model.add(Activation('relu'))


model.add(Conv2D(filters = 32, kernel_size = (3,3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(filters = 64, kernel_size = (3,3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer = 'rmsprop',
              metrics = ['accuracy'])
```

Accuracy: 0.993

We use convolutional neural network (CNN) models for image classification using the Keras deep learning library. Here is a breakdown of the layers:

- The first layer is a Conv2D layer with 32 filters of size 3x3 and an input shape of (IMG_SIZE, IMG_SIZE, 1)An activation function ReLU (Rectified Linear Unit) is applied after the first Conv2D layer to introduce non-linearity into the model.
- The second layer is another Conv2D layer with 32 filters of size 3x3 followed by a ReLU activation function.Then, a MaxPool2D layer is applied with a pooling window size of 2x2 to downsample the output of the second convolutional layer.
- A Dropout layer with a dropout rate of 0.3 is added to reduce overfitting during training.
- The next two layers are similar to the previous layers, with another Conv2D layer with 64 filters, followed by a ReLU activation function, MaxPool2D layer, and

Dropout layer with the same rate of 0.3.

- Two Dense layers with 256 and 10 units respectively are added with ReLU and Softmax activation functions respectively. The final layer outputs a probability distribution over the 10 classes.
- The model is compiled with the categorical_crossentropy loss function, the rmsprop optimizer, and the accuracy metric for evaluation during training.

Robustness check

The accuracy of a CNN model is often quite sensitive to the weights of its CNN layer, which determine how the model analyzes input data and generates predictions. When the weight of an object is altered, its accuracy might suffer.the model's fine-tuning, which may create noise or bias and impair its ability to gather relevant data. Because they often capture more complex and abstract information, the weights of starting layers may have a larger effect on accuracy than the weights of ending layers.

Why model works

- The model acquires non-linearity from the ReLU activation function, hence it is employed in most of the layers. This allows the model to learn more complex and non-linear representations of the input data, which is especially helpful for applications like hand gesture recognition where the input data may be rather complex.
- The softmax activation function is used in the output layer because it maps the layer's output into a probability distribution across the model's 10 possible classes. The Softmax function normalizes the input so that the probability at the output sum up to 1.
- Several factors, including the complexity of the task and the size of the input dataset, determine how many convolution and pooling layers are used. Based on experimental data, our model consists of three convolutional layers and three max pooling layers. For image classification jobs, this architecture is ideal.
- Extracting high-level features from the picture and shrinking the image without losing crucial details is possible because of the combination of the number of filters and the size of the kernel utilized. A single dense layer may effectively classify data and learn useful characteristics from the input.

Our Approach

We adopted different approaches like pruning , changing the weights in hidden layers and changing the different parameters like make number of hidden layers or convolution layers or may be increase in filter size etc.

Basically our approaches are -:

- changing the weights in hidden layers
- pruning
- changing the different parameters
- Quantization

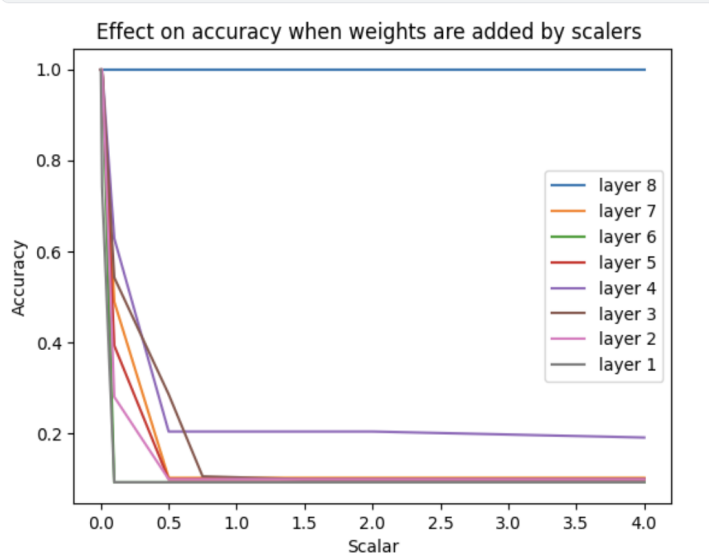
Changing the weights in hidden layers

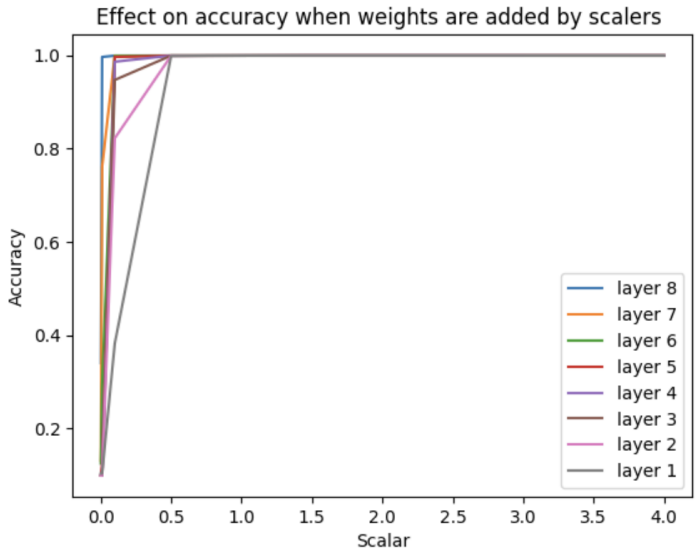
Our approach is to change the weights of the entire model, which involves multiplying or adding a scalar value to each weight in the model. This technique is known as scaling or shifting the weights, respectively. Scaling the weights involves multiplying each weight by a scalar value, while shifting the weights involves adding a scalar value to each weight.

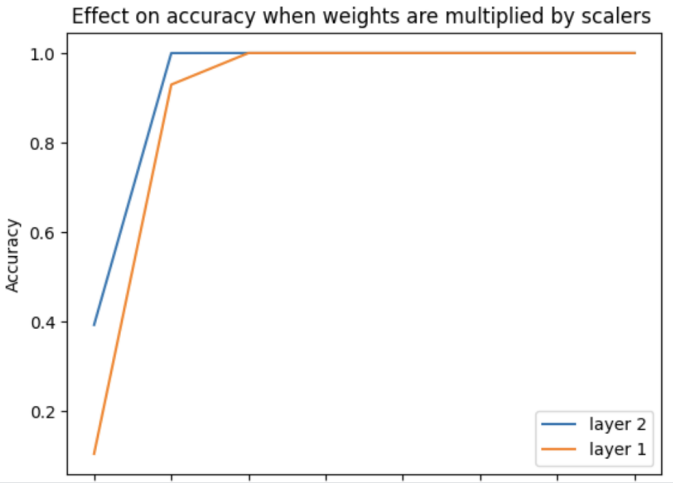
Another approach is to change the weights layer by layer. This technique involves modifying the weights of each layer in the model independently. This can be achieved by either adding a scalar value or multiplying a scalar value to each weight within a given layer.

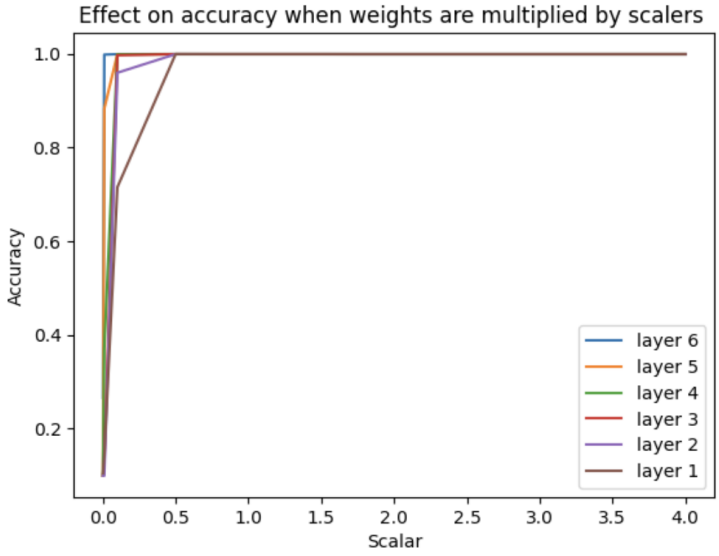
Both of these techniques can be used to improve the performance of a model. Changing the weights of the entire model can be useful when the overall performance of the model needs to be improved, while modifying the weights layer by layer can be more effective when certain layers in the model are not performing well.

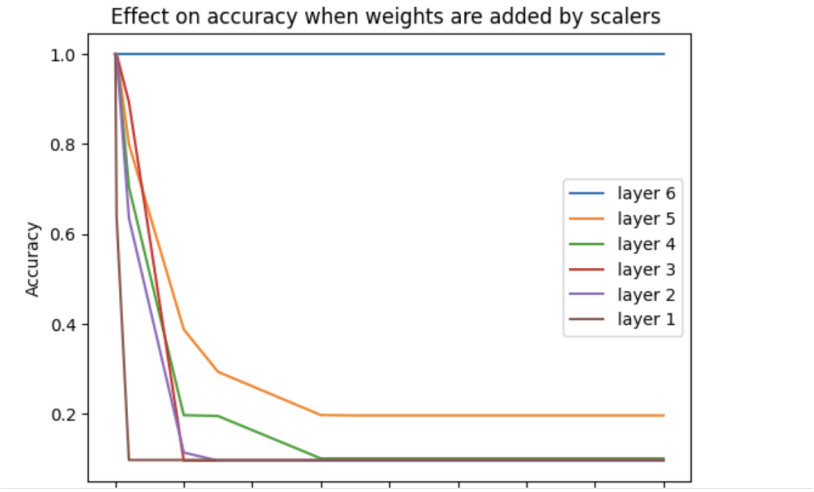
When weights of model are changed layer by layer



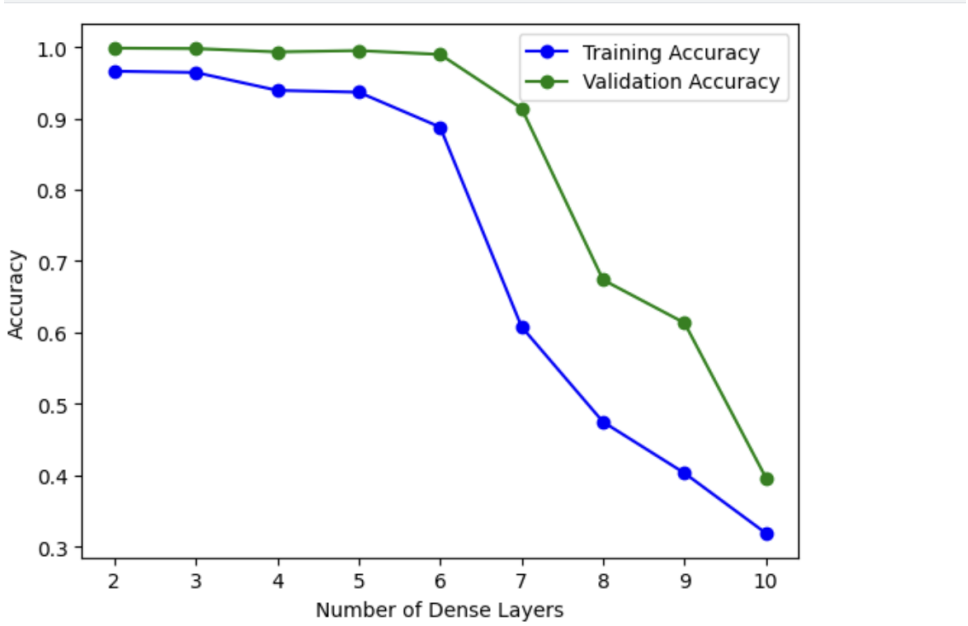








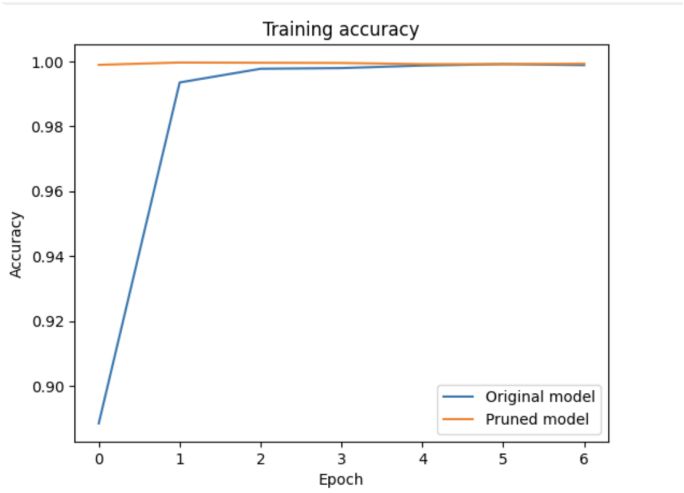
When number of dense layers are changed

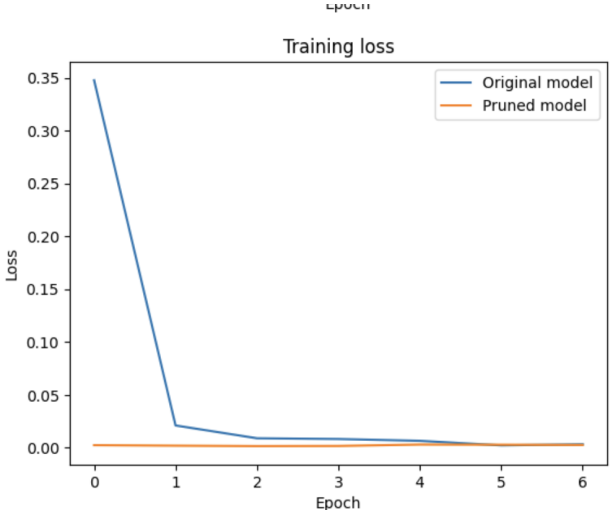


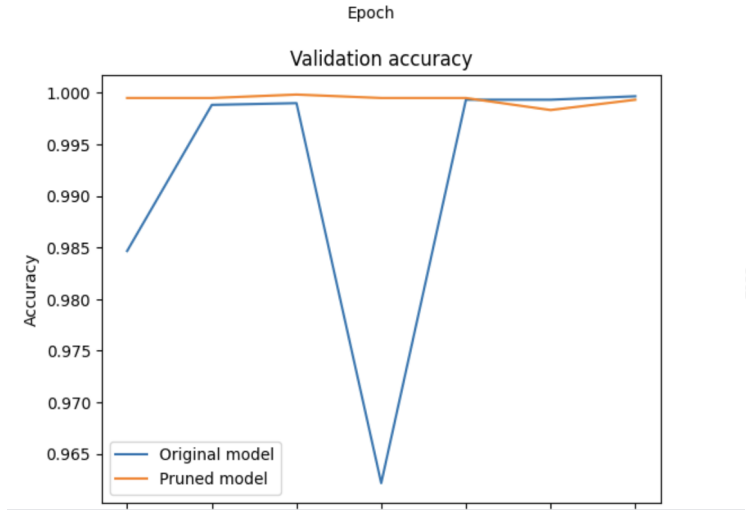
Accuracy of the model (both training accuracy as well as validation accuracy) decreases gradually as we increase the number of dense layers as observed from the graph above.On the other hand, adding more layers to a CNN can also make the model more difficult to train and lead to overfitting. In practice, the optimal number of layers in a CNN depends on the specific task and dataset, and is typically determined through experimentation and tuning.

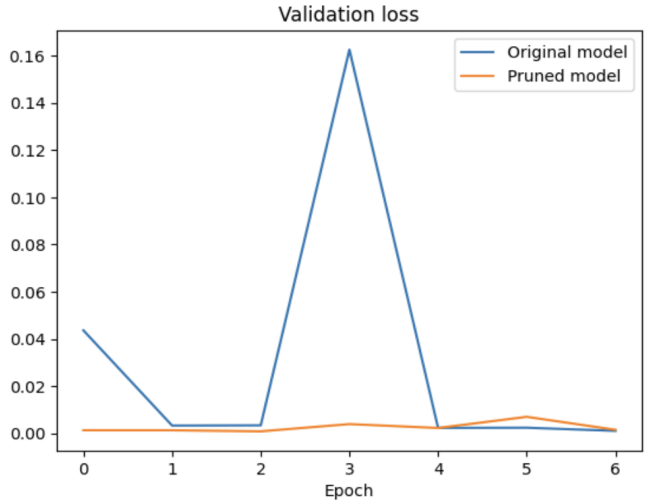
Pruning

Pruning can significantly reduce the size of the model without significantly affecting its performance.We create a pruning schedule, which specifies the pruning rate at different epochs during training. We define a pruning schedule that prunes 50% of the weights after the first 5 epochs, and 80% of the weights after the next 7 epochs.









Here we can conclude that after pruning accuracy increases because typically we are setting small-weight connections to zero or removing them entirely.

One of the main benefits of pruning is that it can lead to an increase in model accuracy. This is because a smaller, more efficient model is less likely to overfit to the training data, which can improve its ability to generalize to new, unseen data.

Changing different parameters

The following parameters can be changed in our neural network:

- Number of convolution layers
- Number of Maxpool layers
- Number of Dense layers
- Number of Filters in convolutional layers
- The optimizer used
- Size of filter in convolution layer

Model no.	Model	No. of filters used in convolution layers	Optimizer	Accuracy	No. of Parameters	Inference Time
1	2 Conv + 1 Maxpool + 2 dense layer	(8,16)	Adam	99.52%	642314	221.26 ms
2	2 Conv + 1 Maxpool + 2 dense layer	(16,32)	Adam	100%	2566922	152.92 ms
3	3 Conv + 3 Maxpool + 2 dense layer	(16,32,64)	Adam	99.95%	1208586	182.47 ms
4	3 Conv + 2 Maxpool + 8 dense layer	(32,32,64)	Rmsprop	99.93%	8651050	148.56 ms
5	3 Conv + 2 Maxpool + 6 dense layer	(32,64,64)	Rmsprop	99.95%	7338794	166.09 ms

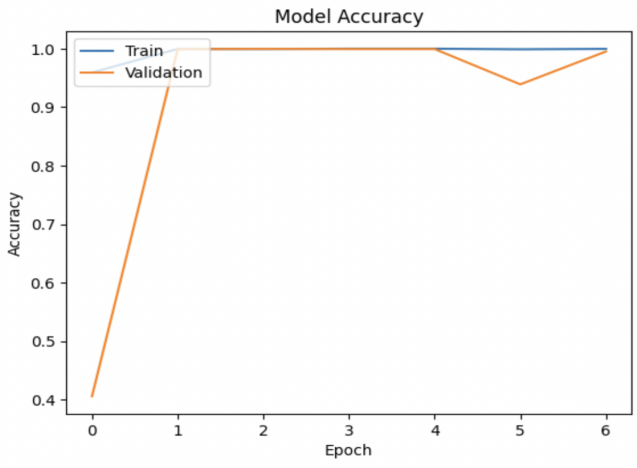
Number of parameters: The number of parameters in a model refers to the total number of learnable parameters that are used to calculate the output of the model for a given input.

Number of layers in a model: The number of layers in a model refers to the total number of layers that are used to transform the input data into the desired output. This includes layers such as convolutional layers, pooling layers, fully connected layers, and activation layers.

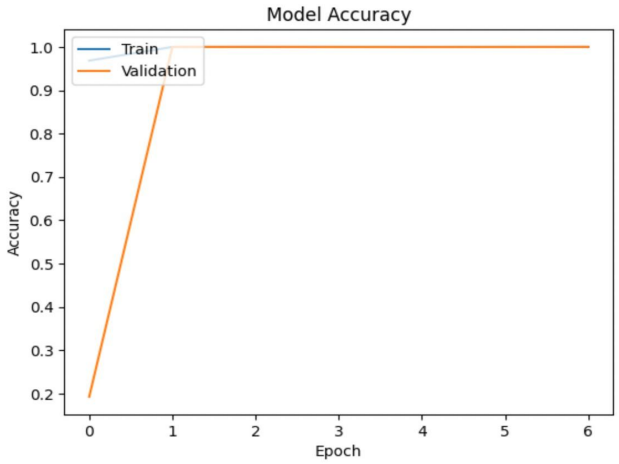
Number of filters used in convolutional layers: The number of filters used in a convolutional layer refers to the number of feature maps or channels that are generated by that layer. Each filter is responsible for detecting a specific feature in the input data.

Inference time: Inference time refers to the time it takes for a trained model to make predictions on new, unseen data.

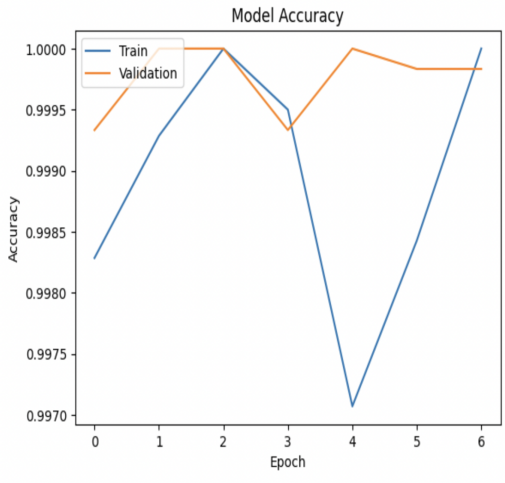
1)



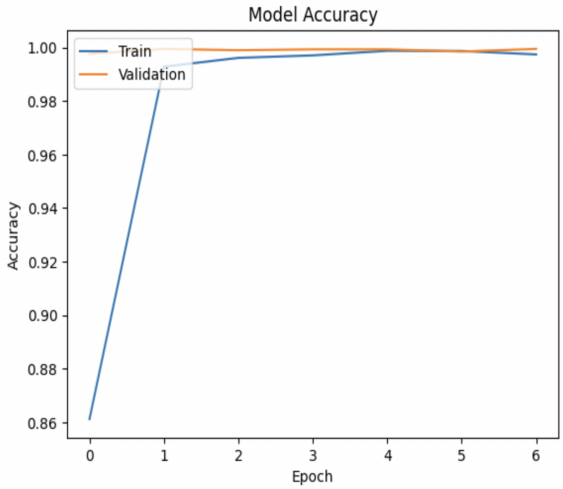
2)



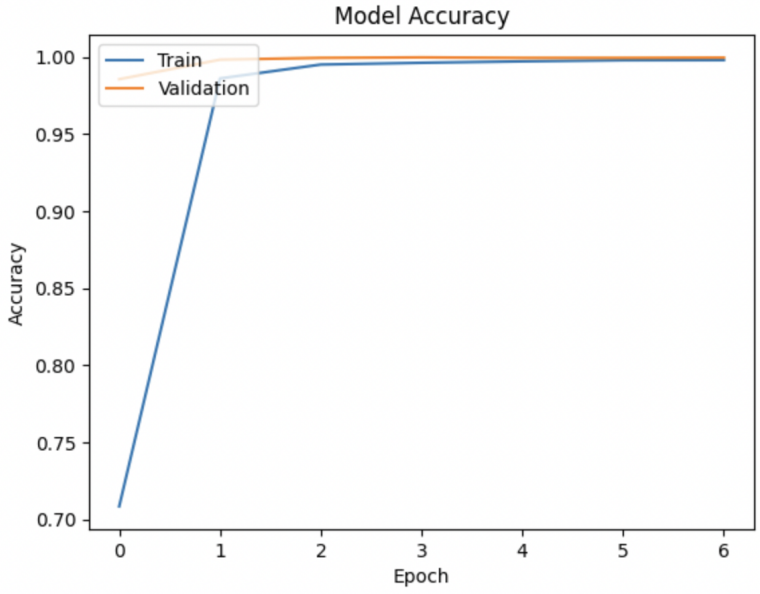
3)


Model Accuracy
Train
Validation
Accuracy
Epoch
1.0000
0.9995
0.9990
0.9985
0.9980
0.9975
0.9970
0 1 2 3 4 5 6

4)


Model Accuracy
Train
Validation
Accuracy
Epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0 1 2 3 4 5 6

5


Model Accuracy
Train
Validation
Accuracy
Epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0 1 2 3 4 5 6

Quantization

Quantization refers to reducing the accuracy of the DL model's weights and activations. In a typical DL model, the weights and activations are 32-bit floating-point numbers. However, devices with limited resources, such as smartphones and Internet of Things (IoT) devices, typically do not require this level of accuracy. By decreasing the accuracy of the weights and activations, we may potentially reduce the size and memory requirements of the model and make it more suitable for deployment on these devices. Post-training quantization has been utilized since it is straightforward and occurs after the model has been trained.

We have quantized the model by converting the 32-bit floating point values to 16-bit ones, and the result is shown below.here we can observe that before Quantization accuracy is more, this is basically depends of no of parameters like hidden layers, how the model is