EMBEDDED SYSTEMS

DL model optimization for Lightweight Gesture Recognition

(Abhishek Jilowa (B20CS001) , Gojiya Piyush D (B20CS015))

Dataset used: Gesture Recognition
The dataset contains 20,000 images of 10 different classes { Palm, fist, fist moved, thumb, ok, palm moved, down, index, l, c }, each class having 2000 images each. We split the dataset into ratios of 80-10-10 to train, test and validate respectively.

Model used:
A CNN model is used. There are 3 sets of a Conv2D layer followed by a MaxPooling2D layer with different number of filters and kernel size. One hidden layer is present and then the final layer with 10 nodes for 10 different classes.

```
model=models.Sequential()
model.add(layers.Conv2D(32, (5, 5), strides=(2, 2), activation='relu', input_shape=(120, 320,
1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

Accuracy: 0.9995

Working of the model

Steps:

1. The first layer of the model is Conv2D layer with ReLU activation function and 32 5x5 Conv2D filters with a stride of 2. Using a 5x5 filter, this layer collects 32 features from the input image. A stride of (2,2) is then used to condense the feature maps' spatial dimensions.

2. The second layer is MaxPooling2D layer that uses a pool size of 2x2. The feature maps' spatial dimensions become two times smaller due to this layer. By doing so, the model's parameter count is decreased and overfitting is prevented.

3. The next set of Conv2D and MaxPooling2D layers function similarly to the previous two layers to extract features from the image and reduce spatial dimensions respectively.

4. The output of the last layer is then flattened into a 1D list so that it can be sent as input to fully connected layers which further perform classification based on the extracted features.

5. One hidden layer is present with 128 nodes and a ReLU activation function is used.

6. Finally there is a dense layer with a softmax activation function and 10 neurons. The result is normalized by the softmax function so that it may be understood as a probability distribution across the 10 potential classes.

Why the model works:

- The **ReLU** activation function is used in most of the layers because the model gains non-linearity from the ReLU activation function. For tasks like hand gesture recognition, where the input data can be fairly complicated, this enables the model to learn more intricate and non-linear representations of the input data.

- The output layer uses **softmax** activation function because it transforms the output of the previous layer into a probability distribution over the 10 potential classes in the model's output layer. To make sure that the output probabilities add up to 1, the Softmax function normalizes the input.

- The number of convolution and pooling layers depends on several factors like how complex the task is and size of the input dataset. In our model, three convolutional layers and three max pooling layers are present which comes from experimental data. This architecture works well for image classification tasks.

- The number of filters and the size of kernel used is able to extract high level features from the image and reduce the size of image without losing important features. Single dense layer is able to learn relevant features from the input data and performs classification with good accuracy.

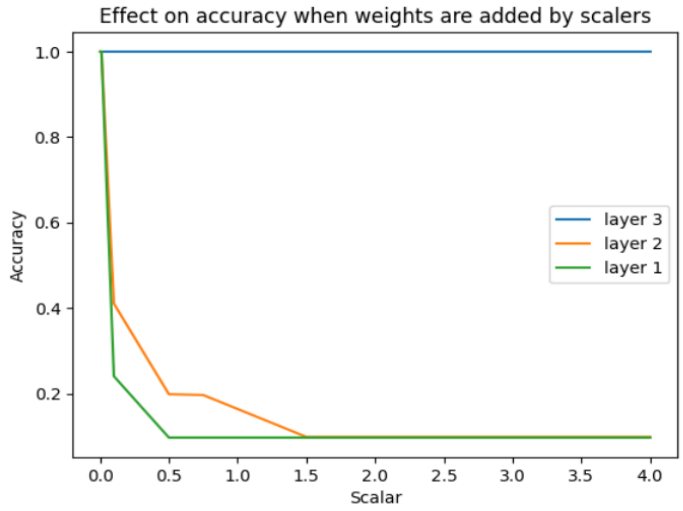
Robustness check

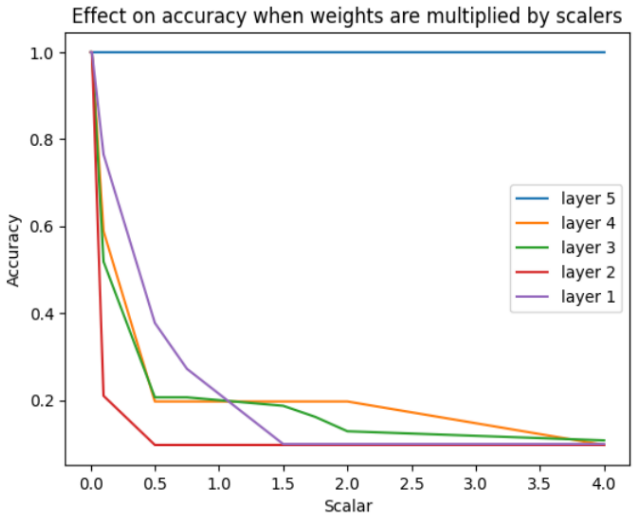
The weights of a CNN layer, which control how the model processes input data and produces predictions, can generally have a considerable impact on the model's accuracy. The accuracy may decline as a result of the weight changes in the fine-tuned model, which reduce its capacity to capture pertinent data or introduce noise or bias. Altering the weights of starting layers, which frequently

capture more complicated and abstract information, may have a bigger effect on accuracy than altering the weights of ending layers.

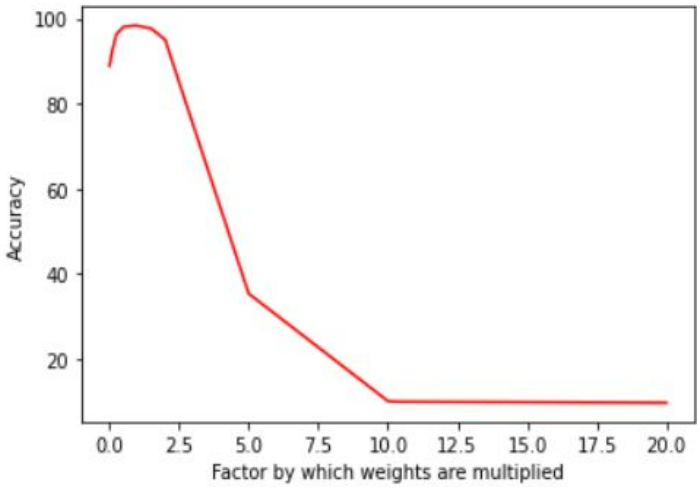
Our Approach : We have changed the weights of the model by adding or multiplying weights by certain scalars. Change in weights can be done in the whole model or layer by layer. We have shown results of both methods below:

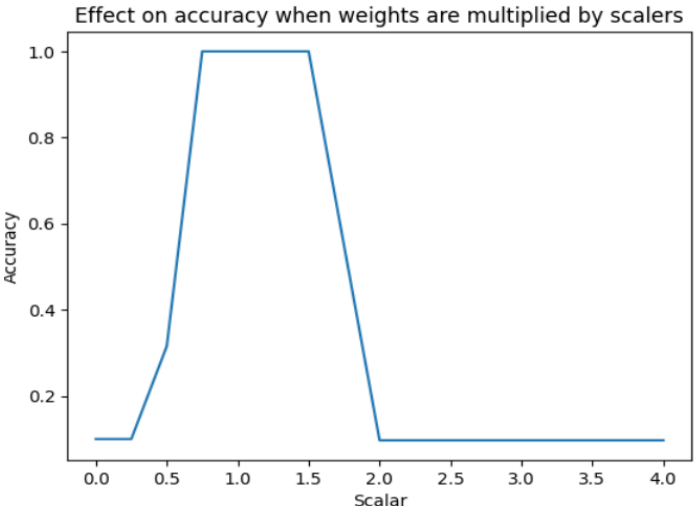
When weights of model are changed layer by layer :



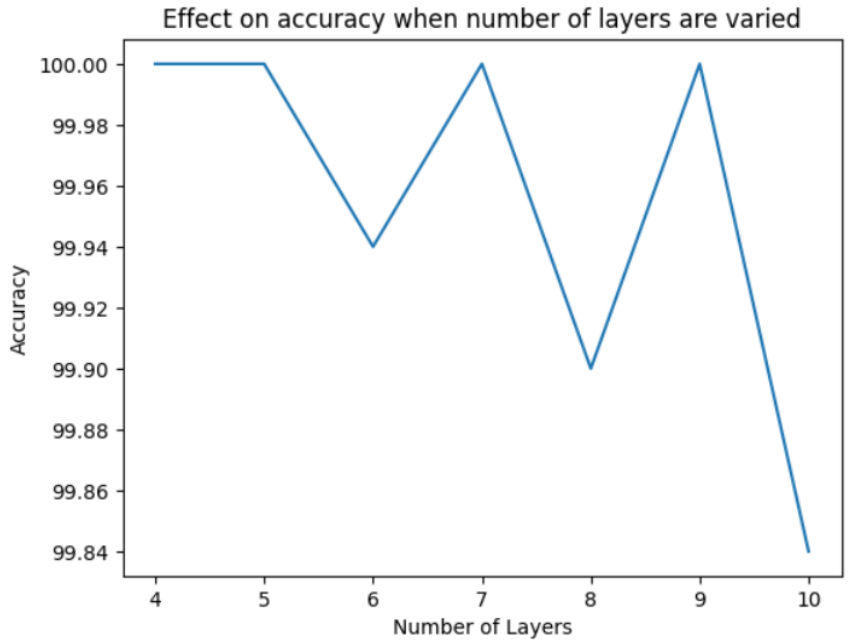


When all the weights are changed at same time:





Increasing the number of layers in a convolutional neural network (CNN) can have both positive and negative effects on the accuracy of the model, depending on several factors, including the architecture of the network, the size and complexity of the input data, and the quality and quantity of the training data. On one hand, adding more layers to a CNN can improve the model's ability to learn complex features and patterns in the data. 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.



Reducing the size of the neural network design

Size of the neural network can be reduced by changing :

- **Depth of the network :** number of layers
- **Width of the network :** number of filters
- **Kernel size :** size of the convolutional kernel
- **Input size :** size of the input data
- **Dropout, weight decay, and batch normalization**

When size of neural network is changed then the accuracy change is shown in the table below:

Model	No. of params	Accuracy
3 Conv+ 3 Maxpool+ 2dense	12,06,666	100
4 Conv+ 4 Maxpool+ 2dense	1,84,714	100
2 Conv+ 2 Maxpool+ 2dense	55,35,018	99.94
2 Conv+ 2 Maxpool+ 2dense	18,33,290	100
2 Conv+ 2 Maxpool+ 2dense	18,59,898	99.9
2 Conv+ 2 Maxpool+ 5dense	79,80,283	100
2 Conv+ 2 Maxpool+ 7dense	79,82,042	99.84

When the number of filters in convolutional layers are changed keeping other parameters the same. The number of convolutional layers, maxpool layer and dense layers are 2,2 and 3 respectively.

No. of filters	Accuracy
(16,16)	99.9
(16,32)	100
(32,96)	99.94

Optimization for lightweight gesture recognition

Optimizing a Deep Learning (DL) model for lightweight gesture recognition usually includes reducing the size of the model, the number of parameters, and the amount of computation needed to make a prediction, without sacrificing too much accuracy. This is especially important for real-time applications that need to run the model on devices with limited resources, such as smartphones, smartwatches, and Internet of Things (IoT) devices. Here we have used two techniques for optimization, quantization and pruning.

Quantization

In a DL model, quantization means lowering the precision of the weights and activations. Weights and activations are commonly expressed in a standard DL model using 32-bit floating-point values. However, this level of accuracy is often not required for devices with limited resources like smartphones and Internet of Things (IoT) devices. We may greatly lower the model size and memory needs and increase the model's suitability for deployment on these devices by reducing the precision of the weights and activations. We have used post-training quantization, which is simple and done after training the model.

We have performed quantization by changing the 32 bit floating point values into 16 bit floating point values and the quantized model is saved in another file.

Pruning

When a DL model is pruned, unimportant links or neurons are taken away. In a DL model, some of the weights or neurons may be redundant or not add much to the end output. By getting rid of these connections or neurons that are not important, we can lower the number of parameters and make the model work better. There are different ways to prune, such as magnitude-based pruning, weight-based pruning, and channel-based pruning. In magnitude-based pruning, the weights are ranked by their exact values and the least important weights are taken away. We have used magnitude based pruning for our model.