The neural networks are exhaustive algorithms, they have high computation cost and consume huge amounts of memory. The computational resources are available in the market and are increasing day by day, the optimization for training models and neural networks interface is important. As to minimize the cost of infrastructure, networks are optimized to run models on the edges. There are various hardware limitations in running the neural networks on smartphones and embedded systems. When more models are introduced to servers and move towards edge reducing computation and size, quantization is one of the ways to do it. It replaces floating point with integers or binary inside the network. The idea for quantization is to convert weight and activation into binary or integer types, that consume less storage space and calculations are done faster but decrease the accuracy. The floating point were introduced for representation and forgo precision of real number, for example 0.1 is 0.166666... and never ending digits and it is not possible to represent it in memory. Therefore, floating point numbers will handle this situation. In recent years as the application of these networks are moving towards embedded applications, the need for quantization has come up again.

There are various ways of implementing Quantization, one of them is Post-training Quantization. In this project, we have implemented this technique to quantize the pre-trained networks at quantization levels such as int8 and int16.

**Methodology**

**Quantization**

It refers to the technique used to carry out computations to store tensors to less bandwidth then floating point precision. In deep learning the quantization means converting into fixed range in between 0 and 255/8 bit integers) from floating point in the range of 1x10^-38 to 1 x 10^38. For minimizing latency in GPU, the modern AI chip has small memories surrounding cores and one big memory so instead of using 32bit we use 8 bit will speed up the transfer 4 times. This will also have additional benefits such as small file size, less memory storage.

In this project we have implemented Post-training quantization of networks.

**1. Quantization Levels**

There are various levels of Quantization available to implement for the pre-trained networks. For this project we have implemented int8 and int16 levels of quantization.

1. **int16 Quantization:** Here the weights pre-trained networks are quantized to int16. There is loss in the information, however, the networks still retain the important information and should not affect the predictions.

2. **int8 Quantization:** In this level of quantization the weights are quantized to 8-bit data. This creates a significant loss in the information and the effects will be quite noticeable in the predictions.

**2. Datasets and Pretrained networks**

**2.1 Datasets**

For this project 2 popular datasets were used:-

1) MNIST dataset
2) Fashion-MNIST dataset

These are the standard datasets that can be used with any network to perform basic analysis.

**2.2 Pretrained networks**

The Pretrained networks used for this project are:-

1) VGG16: It has convolutional layers of 3x3 which are stacked on each other in such a way that it increases the depth.
2) ResNet: The Residual learning means we don’t learn all features, it will try to learn some features from input of the layer.
3) ResNet50v2: It has 50 layers of Residual Network.
4) ResNet101v2: This is an extension of the original ResNet50v2, with a total of 101 layers of ResNet.
5) Xception: It has replaced inception modules with depthwise separable convolution, basically extending the inception architecture.

**3. Performance Matrix**

For each model following Performance parameters were fetched from the confusion matrix to analyze the networks were:-

1) Accuracy: It is defined as the no. true prediction made divided by all the predictions. So this way we can calculate the confusion matrix
2) Precision: It is defined as the no. of true positives predictions returned for model divided by Predicted positives.

**Analysis and Results**

<table>
<thead>
<tr>
<th>F1 score</th>
<th>VGG16</th>
<th>ResNet</th>
<th>ResNet50v2</th>
<th>ResNet101v2</th>
<th>Xception</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.92</td>
<td>0.97</td>
<td>0.94</td>
<td>0.96</td>
<td>0.98</td>
</tr>
<tr>
<td>Fashion-MNIST</td>
<td>0.82</td>
<td>0.72</td>
<td>0.75</td>
<td>0.74</td>
<td>0.92</td>
</tr>
</tbody>
</table>

**Summary/Conclusions**

The quantization were implemented to see if the models behave the same way as the original pre-trained networks. However, not all the quantization levels were able to work as intended.

int16 level of quantization was effective enough for the most of the models. However, int8 quantized models lost a lot of accuracy. The TensorFlow library was used to quantize the models to various levels.

**Key References**


**Acknowledgments**

We would like to express our sincere gratitude to our professor Birsen Sirkeci for giving us the opportunity to do this project and guiding us on each step of the project. The professor’s vision and motivation has helped us to develop new interest in the field of machine learning. It is a great privilege to learn and work under her guidance. We would like to thank our friends for their encouragement and special thanks to Sophia Susan Raju for her support to complete this project.

**Project Advisor:** Birsen Sirkeci