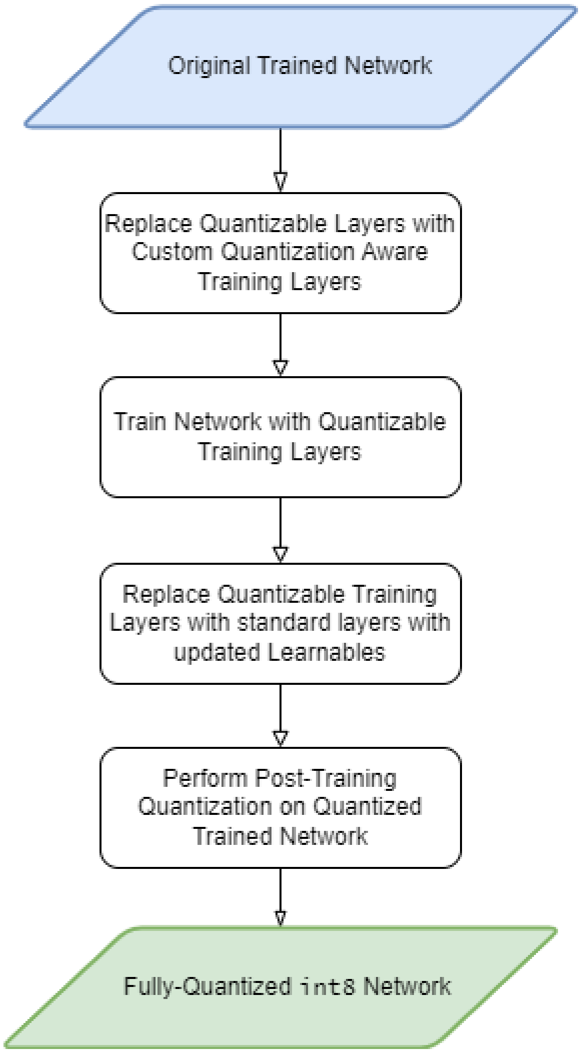
Quantization Aware Training for Transfer Learned MobileNet-v2

This example shows how to perform quantization aware training for transfer learned MobileNet-v2 network.

Low precision types like `int8` propagate quantization error that may degrade the accuracy of the network. Quantization aware training is a method that introduces quantization error at training, thus giving the network the ability to adapt and ultimately produce a network more robust to quantization. In most cases, a fully quantized network or integer-arithmetic only network constructed after quantization aware training can produce accuracy on par with the original floating point network.

This example takes you through the quantization workflow of a transfer learned MobileNet-v2 network. MobileNet-v2 was chosen for this example because it contains depthwise-separable convolution layers that are especially sensitive to post-training quantization.

The flowchart below highlights the steps necessary to convert a trained network into a fully quantized one via quantization aware training.


Original Trained Network
Replace Quantizable Layers with Custom Quantization Aware Training Layers
Train Network with Quantizable Training Layers
Replace Quantizable Training Layers with standard layers with updated Learnables
Perform Post-Training Quantization on Quantized Trained Network
Fully-Quantized int8 Network

Load Flower Dataset

Download the flower dataset [1] using the supporting function `downloadFlowerDataset`.

```
imageFolder = downloadFlowerDataset;
```

```
Downloading Flower Dataset (218 MB)...
```

```
imds = imageDatastore(imageFolder, ...
        IncludeSubfolders=true, ...
        LabelSource="foldernames");
```

Inspect the classes of the data.

```
classes = string(categories(imds.Labels))
```

```
classes = 5×1 string
"daisy"
"dandelion"
"roses"
"sunflowers"
"tulips"
```

Perform Transfer Learning on MobileNet-v2

MobileNet-v2 is a convolutional netural network 53 layers deep. The pretrained version of the network is trained on more than a million images from the ImageNet database.

```
net = mobilenetv2;
```

Split the data into training and validation sets and create augmented image datastores that automatically resize the images to the input size of the network.

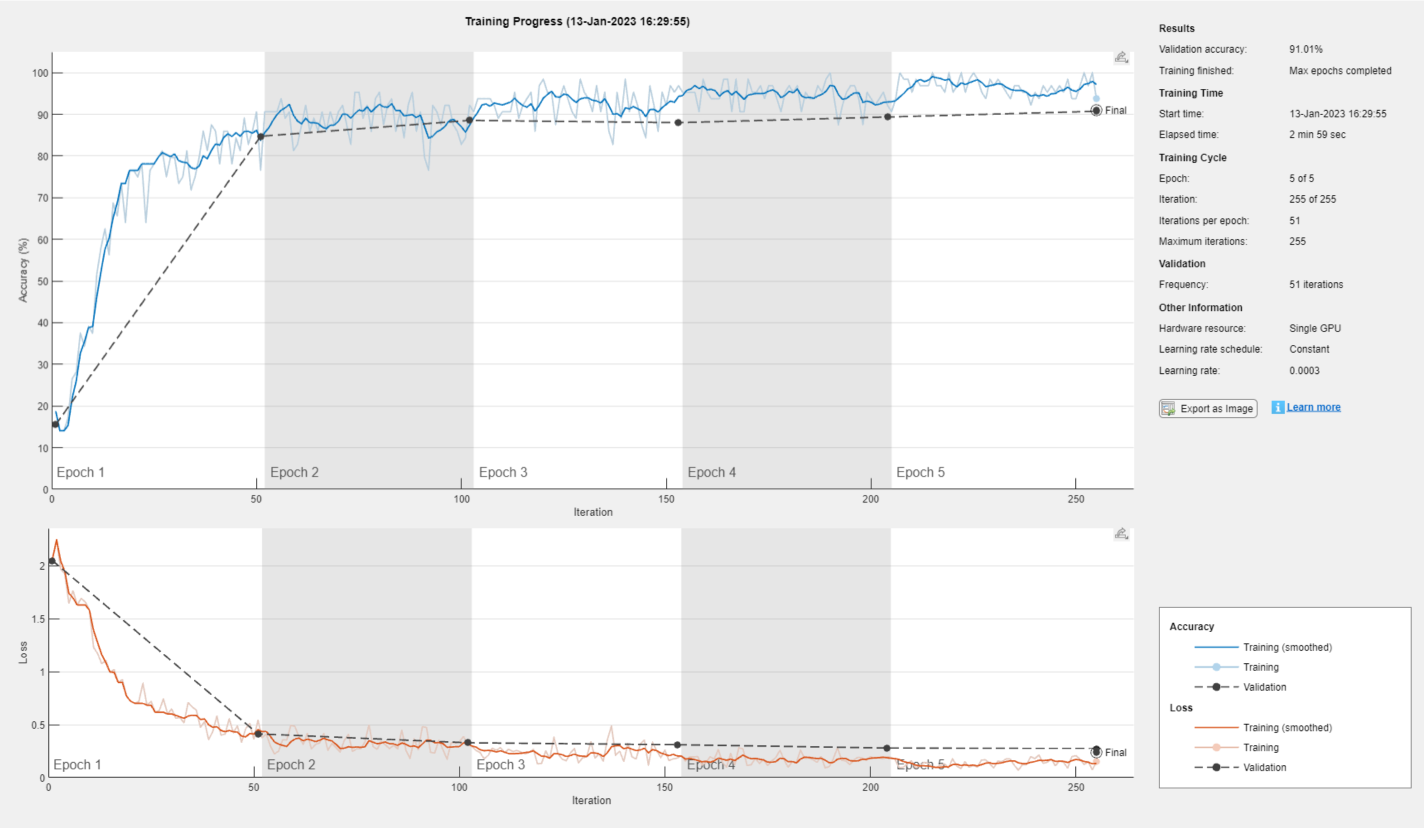
```
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.9);

inputSize = net.Layers(1).InputSize;
augimdsTrain = augmentedImageDatastore(inputSize,imdsTrain);
augimdsValidation = augmentedImageDatastore(inputSize,imdsValidation);
validationActualLabels = imdsValidation.Labels;
```

Set aside a portion of the training dataset to use during the calibration step of quantization. This datastore should be representative of the data used for training but ideally separate from the one used to validate.

```
augimdsCalibration = subset(shuffle(augimdsTrain),1:320);
```

Perform transfer learning on the network on the flowers image dataset. The learnable parameters of the trained network `transferNet` are in `single` precision.

```
transferNet = createFlowerNetwork(net,augimdsTrain,augimdsValidation,classes);
```



Results
Validation accuracy: 91.01%
Training finished: Max epochs completed
Training Time
Start time: 13-Jan-2023 16:29:55
Elapsed time: 2 min 59 sec
Training Cycle
Epoch: 5 of 5
Iteration: 255 of 255
Iterations per epoch: 51
Maximum iterations: 255
Validation
Frequency: 51 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0003

Evaluate Baseline Network Performance

Evaluate the performance of the `single` precision network. Performance in this case is defined as the correct classification rate.

```
netCCR = 
evaluateModelAccuracy(transferNet,augimdsValidation,validationActualLabels)
```

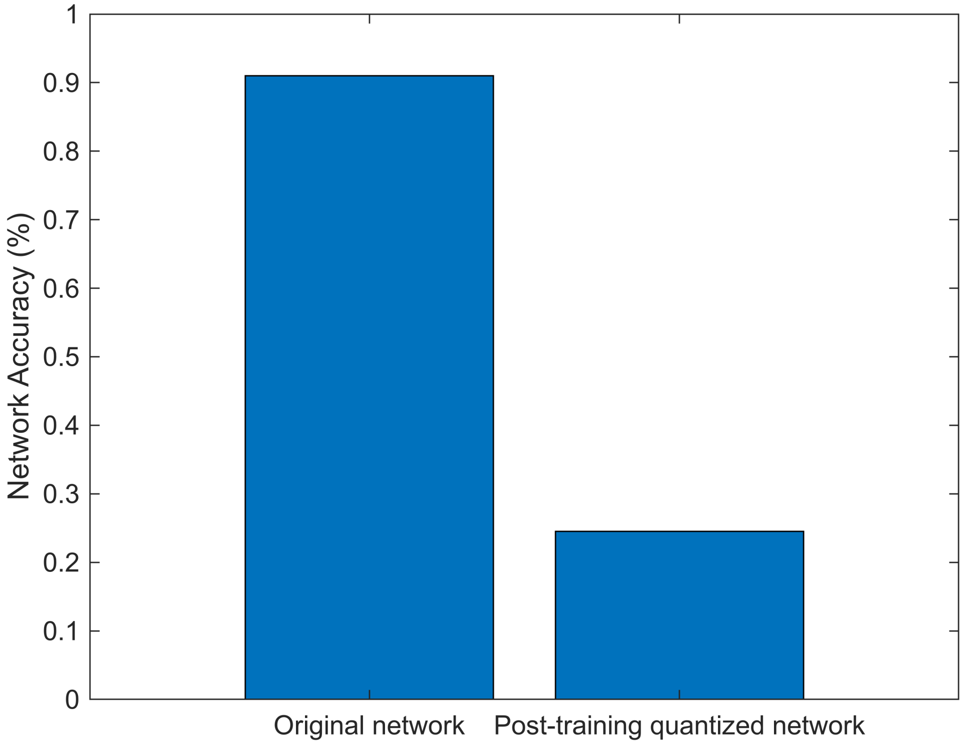
```
netCCR = 0.9101
```

Quantize the network using the `createQuantizedNetwork` function provided at the end of this example and evaluate the performance of the quantized network. Post-training quantization of the original network yields poor performance due to the range of learnable values in the depthwise separable convolution layers. An accuracy of roughly 20% is the equivalent to guessing one of the 5 possible labels for each image.

```
originalQuantizedNet = createQuantizedNetwork(transferNet,augimdsCalibration);
originalQuantizedCCR = 
evaluateModelAccuracy(originalQuantizedNet,augimdsValidation,validationActualLabels)
```

```
originalQuantizedCCR = 0.2452
```

```
bar( ...
    categorical(["Original network","Post-training quantized network"]), ...
    [netCCR,originalQuantizedCCR] ...
    )
ylabel("Network Accuracy (%)")
```



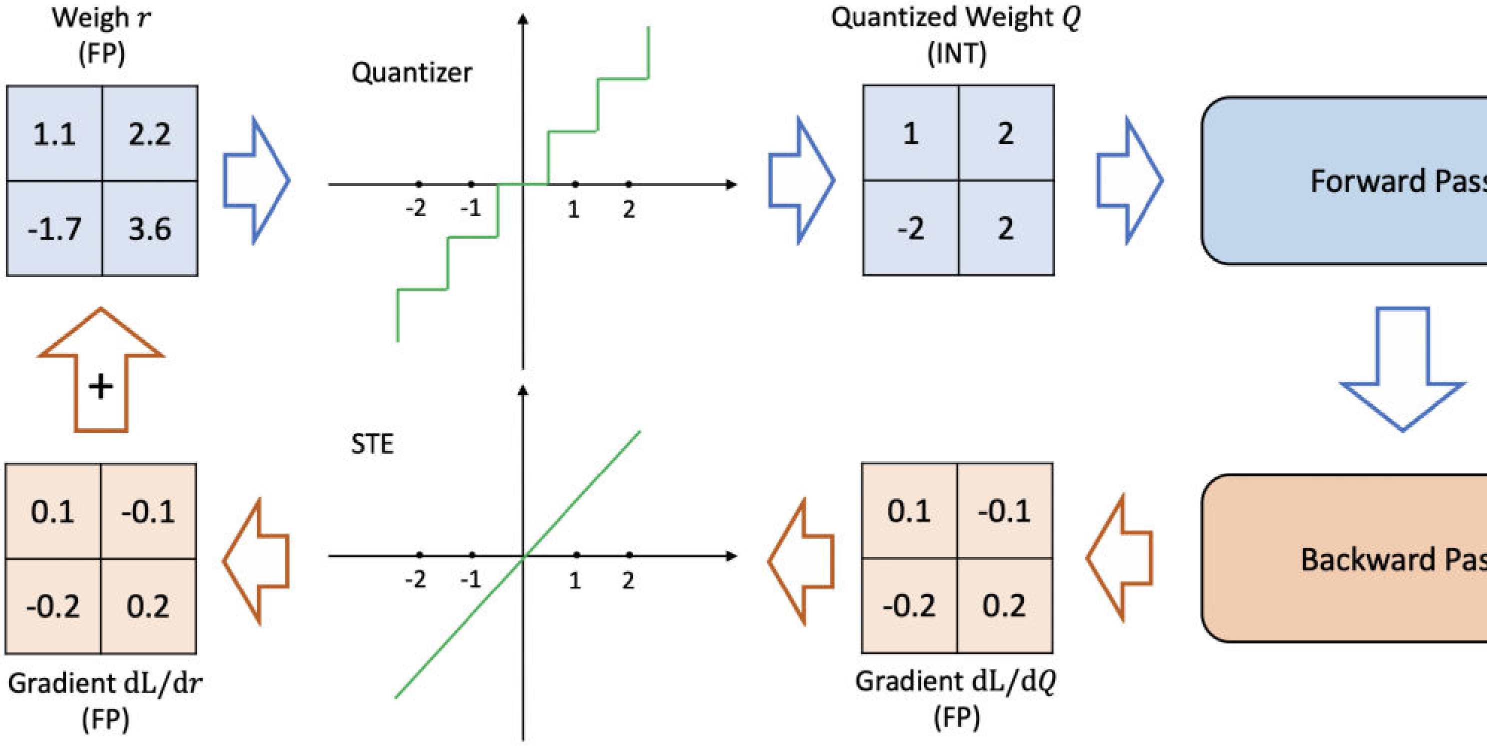
Replace Network Layers with Quantization Aware Training Layers

Replace the `Convolution2D` and `GroupedConvolution2D` layers along with their adjacent `BatchNormalization` layers with custom layers that are quantization aware using the `makeQuantizationAwareLayers` function provided with this example. The quantization aware layers are custom layers that have modified `forward` and `predict` behavior that inject quantization error similar to that of post-training quantization. The quantization error comes from the quantizeToFloat function that quantizes, then unquantizes a given value using best-precision scaling to `int8` precision.

Quantization to float can be expressed as follows.

$$\begin{aligned}\hat{x} &= \mathrm{quantizeToFloat}(x)\\ &= \mathrm{unquantize}(\mathrm{quantize}(x))\\ &= \mathrm{rescale}\cdot\mathrm{saturate}\left(\mathrm{round}\left(\frac{x}{\mathrm{scale}}\right)\right)\end{aligned}$$

The quantization step uses a non-differentiable operation `round` that would normally break the training workflow by zeroing out the gradients. During quantization aware training, bypass the gradient calculations for non-differentiable operations using an identity function. The diagram below [2] shows how the custom layer calculates the gradients for non-differentiable operations with the identity function via straight-through estimation.



For 2-D convolution layers, the weights and biases of the replacement layers include the batch normalization layer statistics. Convolution operations furing training use the adjusted and quantized weights [3].

As the batch normalization layer statistics are incorporated into the convolution layers, the `makeQuantizationAwareLayers` replaces each batch normalization layer with an identity layer that returns its input as its output.

```
quantizationAwareLayerGraph = makeQuantizationAwareLayers(transferNet);
```

Inspect the layers of the network.

```
quantizationAwareLayerGraph.Layers
```

```
ans = 
  154×1 Layer array with layers:

     1   'input_1'                         Image Input                         224×224×3 images with 'zscore' normali
     2   'Conv1'                           Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
     3   'bn_Conv1'                        Identity Training Layer             No operation to forward behavior
     4   'Conv1_relu'                      Clipped ReLU                        Clipped ReLU with ceiling 6
     5   'expanded_conv_depthwise'         Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
     6   'expanded_conv_depthwise_BN'      Identity Training Layer             No operation to forward behavior
     7   'expanded_conv_depthwise_relu'    Clipped ReLU                        Clipped ReLU with ceiling 6
     8   'expanded_conv_project'           Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
     9   'expanded_conv_project_BN'        Identity Training Layer             No operation to forward behavior
    10   'block_1_expand'                  Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
    11   'block_1_expand_BN'               Identity Training Layer             No operation to forward behavior
```

```
12   'block_1_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
13   'block_1_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
14   'block_1_depthwise_BN'           Identity Training Layer             No operation to forward behavior
15   'block_1_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
16   'block_1_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
17   'block_1_project_BN'             Identity Training Layer             No operation to forward behavior
18   'block_2_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
19   'block_2_expand_BN'              Identity Training Layer             No operation to forward behavior
20   'block_2_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
21   'block_2_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
22   'block_2_depthwise_BN'           Identity Training Layer             No operation to forward behavior
23   'block_2_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
24   'block_2_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
25   'block_2_project_BN'             Identity Training Layer             No operation to forward behavior
26   'block_2_add'                    Addition                            Element-wise addition of 2 inputs
27   'block_3_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
28   'block_3_expand_BN'              Identity Training Layer             No operation to forward behavior
29   'block_3_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
30   'block_3_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
31   'block_3_depthwise_BN'           Identity Training Layer             No operation to forward behavior
32   'block_3_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
33   'block_3_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
34   'block_3_project_BN'             Identity Training Layer             No operation to forward behavior
35   'block_4_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
36   'block_4_expand_BN'              Identity Training Layer             No operation to forward behavior
37   'block_4_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
38   'block_4_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
39   'block_4_depthwise_BN'           Identity Training Layer             No operation to forward behavior
40   'block_4_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
41   'block_4_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
42   'block_4_project_BN'             Identity Training Layer             No operation to forward behavior
43   'block_4_add'                    Addition                            Element-wise addition of 2 inputs
44   'block_5_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
45   'block_5_expand_BN'              Identity Training Layer             No operation to forward behavior
46   'block_5_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
47   'block_5_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
48   'block_5_depthwise_BN'           Identity Training Layer             No operation to forward behavior
49   'block_5_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
50   'block_5_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
51   'block_5_project_BN'             Identity Training Layer             No operation to forward behavior
52   'block_5_add'                    Addition                            Element-wise addition of 2 inputs
53   'block_6_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
54   'block_6_expand_BN'              Identity Training Layer             No operation to forward behavior
55   'block_6_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
56   'block_6_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
57   'block_6_depthwise_BN'           Identity Training Layer             No operation to forward behavior
58   'block_6_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
59   'block_6_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
60   'block_6_project_BN'             Identity Training Layer             No operation to forward behavior
61   'block_7_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
62   'block_7_expand_BN'              Identity Training Layer             No operation to forward behavior
63   'block_7_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
64   'block_7_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
65   'block_7_depthwise_BN'           Identity Training Layer             No operation to forward behavior
66   'block_7_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
67   'block_7_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
68   'block_7_project_BN'             Identity Training Layer             No operation to forward behavior
69   'block_7_add'                    Addition                            Element-wise addition of 2 inputs
70   'block_8_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
71   'block_8_expand_BN'              Identity Training Layer             No operation to forward behavior
72   'block_8_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
73   'block_8_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
74   'block_8_depthwise_BN'           Identity Training Layer             No operation to forward behavior
75   'block_8_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
```

```
76   'block_8_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
77   'block_8_project_BN'             Identity Training Layer             No operation to forward behavior
78   'block_8_add'                    Addition                            Element-wise addition of 2 inputs
79   'block_9_expand'                 Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
80   'block_9_expand_BN'              Identity Training Layer             No operation to forward behavior
81   'block_9_expand_relu'            Clipped ReLU                        Clipped ReLU with ceiling 6
82   'block_9_depthwise'              Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
83   'block_9_depthwise_BN'           Identity Training Layer             No operation to forward behavior
84   'block_9_depthwise_relu'         Clipped ReLU                        Clipped ReLU with ceiling 6
85   'block_9_project'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
86   'block_9_project_BN'             Identity Training Layer             No operation to forward behavior
87   'block_9_add'                    Addition                            Element-wise addition of 2 inputs
88   'block_10_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
89   'block_10_expand_BN'             Identity Training Layer             No operation to forward behavior
90   'block_10_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
91   'block_10_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
92   'block_10_depthwise_BN'          Identity Training Layer             No operation to forward behavior
93   'block_10_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
94   'block_10_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
95   'block_10_project_BN'            Identity Training Layer             No operation to forward behavior
96   'block_11_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
97   'block_11_expand_BN'             Identity Training Layer             No operation to forward behavior
98   'block_11_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
99   'block_11_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Group
100   'block_11_depthwise_BN'          Identity Training Layer             No operation to forward behavior
101   'block_11_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
102   'block_11_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
103   'block_11_project_BN'            Identity Training Layer             No operation to forward behavior
104   'block_11_add'                   Addition                            Element-wise addition of 2 inputs
105   'block_12_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
106   'block_12_expand_BN'             Identity Training Layer             No operation to forward behavior
107   'block_12_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
108   'block_12_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
109   'block_12_depthwise_BN'          Identity Training Layer             No operation to forward behavior
110   'block_12_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
111   'block_12_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
112   'block_12_project_BN'            Identity Training Layer             No operation to forward behavior
113   'block_12_add'                   Addition                            Element-wise addition of 2 inputs
114   'block_13_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
115   'block_13_expand_BN'             Identity Training Layer             No operation to forward behavior
116   'block_13_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
117   'block_13_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
118   'block_13_depthwise_BN'          Identity Training Layer             No operation to forward behavior
119   'block_13_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
120   'block_13_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
121   'block_13_project_BN'            Identity Training Layer             No operation to forward behavior
122   'block_14_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
123   'block_14_expand_BN'             Identity Training Layer             No operation to forward behavior
124   'block_14_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
125   'block_14_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
126   'block_14_depthwise_BN'          Identity Training Layer             No operation to forward behavior
127   'block_14_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
128   'block_14_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
129   'block_14_project_BN'            Identity Training Layer             No operation to forward behavior
130   'block_14_add'                   Addition                            Element-wise addition of 2 inputs
131   'block_15_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
132   'block_15_expand_BN'             Identity Training Layer             No operation to forward behavior
133   'block_15_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
134   'block_15_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
135   'block_15_depthwise_BN'          Identity Training Layer             No operation to forward behavior
136   'block_15_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
137   'block_15_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
138   'block_15_project_BN'            Identity Training Layer             No operation to forward behavior
139   'block_15_add'                   Addition                            Element-wise addition of 2 inputs
```

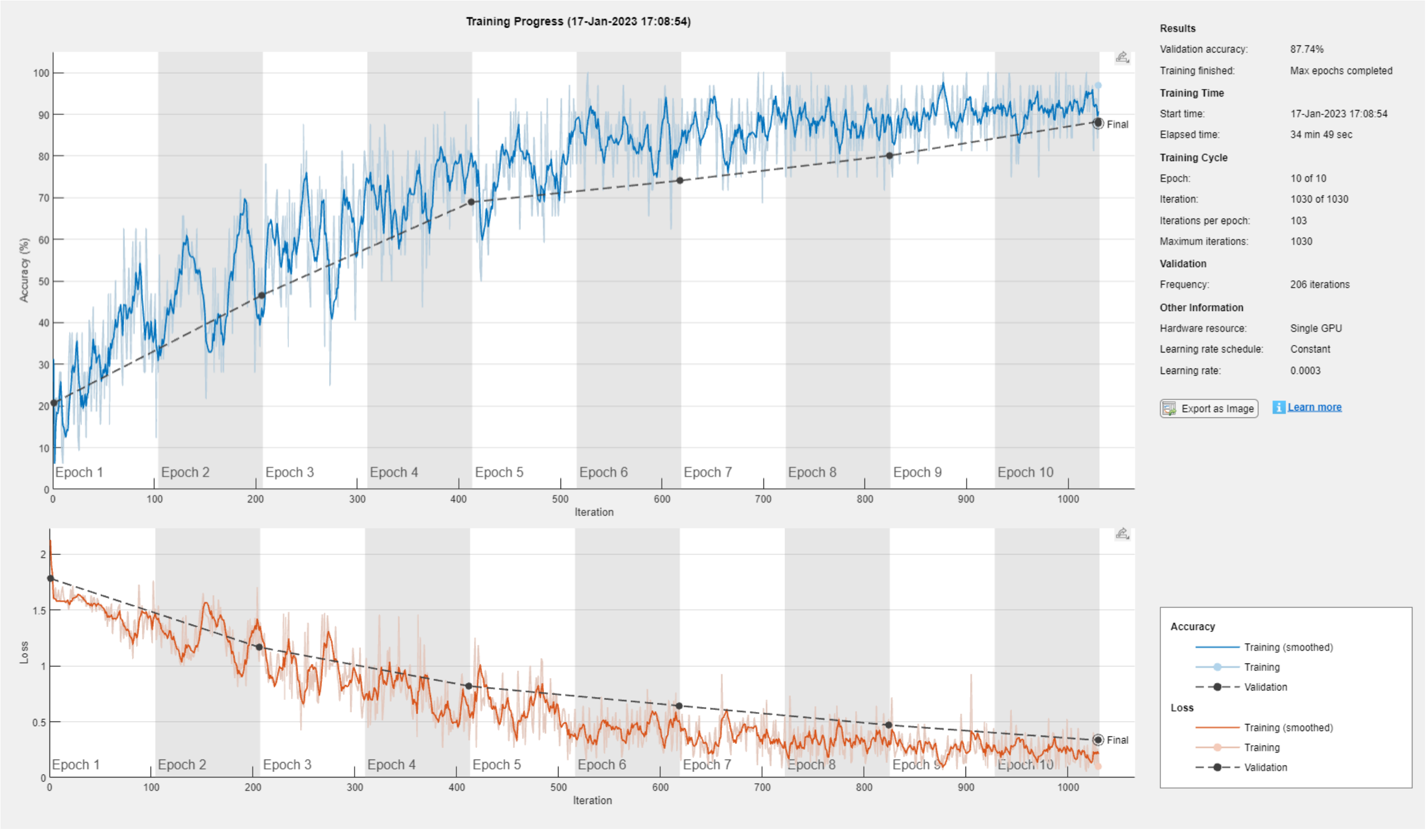
```
140   'block_16_expand'                Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
141   'block_16_expand_BN'             Identity Training Layer             No operation to forward behavior
142   'block_16_expand_relu'           Clipped ReLU                        Clipped ReLU with ceiling 6
143   'block_16_depthwise'             Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
144   'block_16_depthwise_BN'          Identity Training Layer             No operation to forward behavior
145   'block_16_depthwise_relu'        Clipped ReLU                        Clipped ReLU with ceiling 6
146   'block_16_project'               Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
147   'block_16_project_BN'            Identity Training Layer             No operation to forward behavior
148   'Conv_1'                         Quantized Fused Convolution Layer   Quantization Aware Conv-BN Layer Grou
149   'Conv_1_bn'                      Identity Training Layer             No operation to forward behavior
150   'out_relu'                       Clipped ReLU                        Clipped ReLU with ceiling 6
151   'global_average_pooling2d_1'     2-D Global Average Pooling          2-D global average pooling
152   'new_fc'                         Fully Connected                     5 fully connected layer
153   'Logits_softmax'                 Softmax                             softmax
154   'new_classoutput'                Classification Output               crossentropyex with 'daisy' and 4 oth
```

To apply quantization aware training to a network that contains convolution layers without an adjacent bach normalization layer, use the `QuantizedConvolutionTrainingLayer` provided with this example instead of `QuantizedConvolutionBatchNormTrainingLayer`.

Do Quantization Aware Training

Using the layer graph with quantization aware training layers, train the network. Compared to the training of the original network, the training options have been modified to increase the number of `MaxEpochs` to 10 and the `ValidationFrequency` to every 2 epochs.

```
miniBatchSize = 32;
validationFrequencyEpochs = 2;
numObservations = augimdsTrain.NumObservations;
numIterationsPerEpoch = floor(numObservations/miniBatchSize);
validationFrequency = validationFrequencyEpochs*numIterationsPerEpoch;

options = trainingOptions("sgdm", ...
        MaxEpochs=10, ...
        MiniBatchSize=miniBatchSize, ...
        InitialLearnRate=3e-4, ...
        Shuffle="every-epoch", ...
        ValidationData=augimdsValidation, ...
        ValidationFrequency=validationFrequency, ...
        Plots="training-progress", ...
        Verbose=false);

quantizationAwareTrainedNet =
trainNetwork(augimdsTrain,quantizationAwareLayerGraph,options);
```



Results
Validation accuracy: 87.74%
Training finished: Max epochs completed

Training Time
Start time: 17-Jan-2023 17:08:54
Elapsed time: 34 min 49 sec

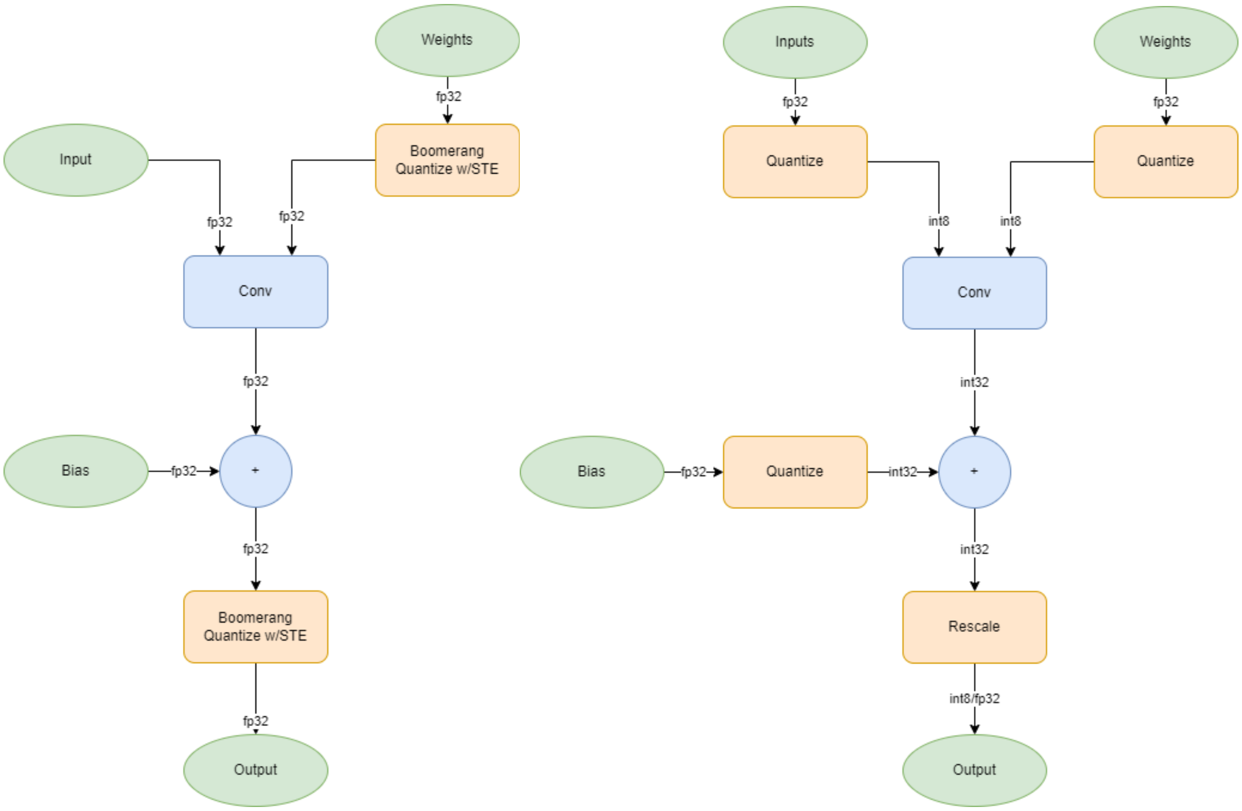
Training Cycle
Epoch: 10 of 10
Iteration: 1030 of 1030
Iterations per epoch: 103
Maximum iterations: 1030

Validation
Frequency: 206 iterations

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0003

Quantize the Network

The network returned by the `trainNetwork` function still has quantization aware training layers. The quantization aware training operators need to be replaced with operators that are specific to inference. Whereas training was performed using 32-bit floating-point values, the quantized network must perform inference using 8-bit integer inputs and weights.



Remove the quantization aware layers and replace with underlying learned convolution layers using the `removeQuantizationAwareLayers` function, provided at the end of this example.

```
preQuantizedNetwork = removeQuantizationAwareLayers(quantizationAwareTrainedNet);
```

Perform post-training quantization on the network as normal using the function `createQuantizedNetwork`, provided at the end of this example.

```
quantizationAwareQuantizedNet =
createQuantizedNetwork(preQuantizedNetwork,augimdsCalibration);
quantizedNetworkDetails = quantizationDetails(quantizationAwareQuantizedNet)
```

```
quantizedNetworkDetails = struct with fields:
            IsQuantized: 1
          TargetLibrary: "cudnn"
    QuantizedLayerNames: [105×1 string]
    QuantizedLearnables: [60×3 table]
```

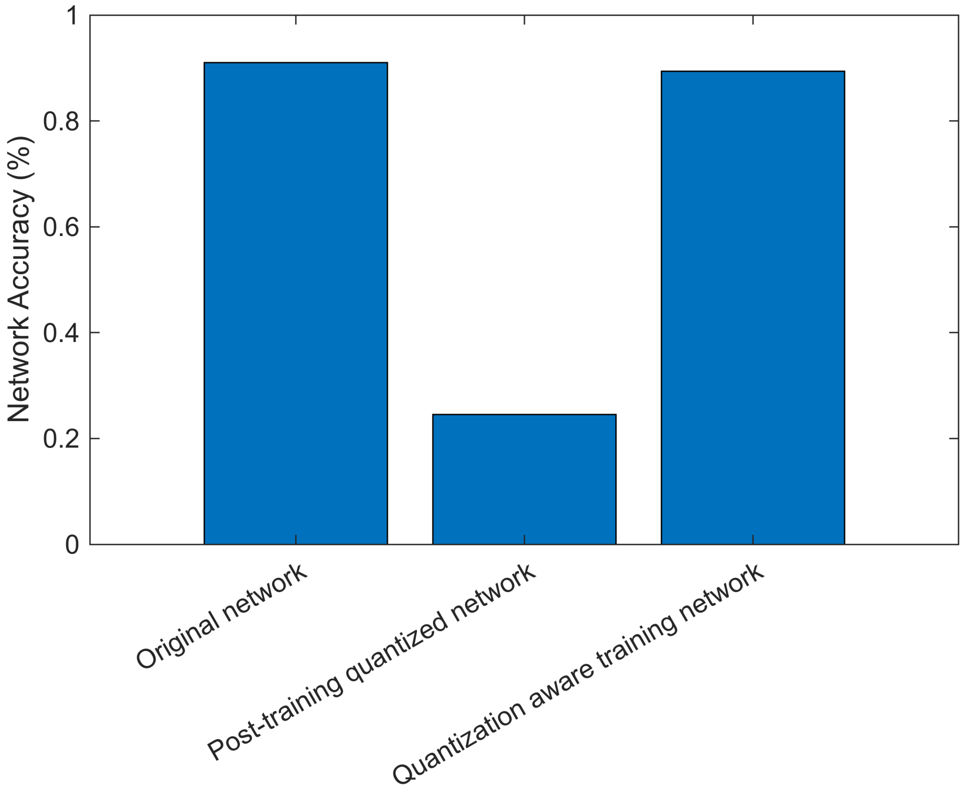
Evaluate the Quantized Network

Evaluate the performance of the quantized network.

```
quantizedNetworkCCR =
evaluateModelAccuracy(quantizationAwareQuantizedNet,augimdsValidation,validationActu
alLabels)
```

```
quantizedNetworkCCR = 0.8937
```

```
bar( ...
    categorical(["Original network","Post-training quantized network","Quantization
aware training network"]), ...
    [netCCR,originalQuantizedCCR,quantizedNetworkCCR] ...
    )
ylabel("Network Accuracy (%)")
```



The accuracy for the quantized network after quantization aware training is on par with the accuracy of that from the original floating point network.

References

1. The TensorFlow Team. *Flowers* http://download.tensorflow.org/example_images/flower_photos.tgz
2. Gholami, A., Kim, S., Dong, Z., Mahoney, M., & Keutzer, K. (2021). A Survey of Quantization Methods for Efficient Neural Network Inference. Retrieved from https://arxiv.org/abs/2103.13630
3. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. Retrieved from https://arxiv.org/abs/1712.05877

Supporting Functions

Download Flower Dataset

The `downloadFlowerDataset` function downloads and extracts the flowers dataset, if it is not yet in the current folder.

```
function imageFolder = downloadFlowerDataset

    url = "http://download.tensorflow.org/example_images/flower_photos.tgz";
    downloadFolder = pwd;
    filename = fullfile(downloadFolder,"flower_dataset.tgz");
    imageFolder = fullfile(downloadFolder,"flower_photos");

    if ~exist(imageFolder,"dir")
        disp("Downloading Flower Dataset (218 MB)...")
        websave(filename,url);
        untar(filename,downloadFolder)
    end

end
```

Perform Transfer Learning

The `createFlowerNetwork` function replaces the final fully connected and classification layer of the MobileNet-v2 network and retrains the nework to classify flowers.

```
function transfer_net = 
createFlowerNetwork(net,augimdsTrain,augimdsValidation,classes)

    % Define network architecture.
    % Find and replace layers to perform transfer learning.
    lgraph = layerGraph(net);

    % Replace the learnable layer with a new one.
    learnableLayer = lgraph.Layers(end-2);
    numClasses = numel(classes);
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        Name="new_fc", ...
        WeightLearnRateFactor=10, ...
        BiasLearnRateFactor=10);
    lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);

    % Replace the classification layer with a new one specific to the type
    % classes seen in the flowers dataset.
    classLayer = lgraph.Layers(end);
    newClassLayer = classificationLayer(Name="new_classoutput");
    lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);

    % Specify training options.
    miniBatchSize = 64;
    validationFrequencyEpochs = 1;
    numObservations = augimdsTrain.NumObservations;
    numIterationsPerEpoch = floor(numObservations/miniBatchSize);
    validationFrequency = validationFrequencyEpochs * numIterationsPerEpoch;

    options = trainingOptions("sgdm", ...
        MaxEpochs=5, ...
        MiniBatchSize=miniBatchSize, ...
        InitialLearnRate=3e-4, ...
        Shuffle="every-epoch", ...
        ValidationData=augimdsValidation, ...
        ValidationFrequency=validationFrequency, ...
        Plots="training-progress", ...
        Verbose=false);

    % Train the network.
    transfer_net = trainNetwork(augimdsTrain,lgraph,options);
end
```

Evaluate Mode Accuracy

The `evaluateModelAccuracy` function compares the classify output of the network with the actual labels.

```
function ccr = evaluateModelAccuracy(net,valDS,labels)
    ypred = classify(net,valDS);
    ccr = mean(ypred == labels);
end
```

Create Quantized Network

The `createQuantizedNetwork` constructs a `dlquantizer` object for `GPU` target, simulates and collects ranges of the network with a representative datastore using the `calibrate` function, then quantizes the network using the `quantize` function.

```
function qNet = createQuantizedNetwork(net,calDS)
    dq = dlquantizer(net,ExecutionEnvironment="GPU");
    calResults = calibrate(dq,calDS);
    qNet = quantize(dq);
end
```

Make Quantization Aware LayerGraph

The `makeQuantizationAwareLayers` function takes a `DAGNetwork` object as input and replaces 2-D convolution, grouped 2-D convolution and batch normalization layers with quantization aware versions.

```
function lg = makeQuantizationAwareLayers(net)
    lg = layerGraph(net);

    for idx = 1:numel(lg.Layers) - 1
        currentLayer = lg.Layers(idx);
        nextLayer = lg.Layers(idx + 1);

        % Find 2-D convolution layers or 2-D grouped convolution layers.
        if (isa(currentLayer,"nnet.cnn.layer.Convolution2DLayer") ...
                || isa(currentLayer,"nnet.cnn.layer.GroupedConvolution2DLayer"))

            if isa(nextLayer,"nnet.cnn.layer.BatchNormalizationLayer")

                % Replace convolution layer with quantization aware layer.
                qLayer = 
QuantizedConvolutionBatchNormTrainingLayer(currentLayer,nextLayer);
                lg = replaceLayer(lg,currentLayer.Name,qLayer);

                % Replace batchNormalizationLayer with identity training
                % layer.
                qLayer = IdentityTrainingLayer(nextLayer);
                lg = replaceLayer(lg,nextLayer.Name,qLayer);

            else
                % Replace convolution layer with quantization aware layer.
                qLayer = QuantizedConvolutionTrainingLayer(currentLayer);
                lg = replaceLayer(lg,currentLayer.Name,qLayer);
            end

        end
    end

end
```

Remove Quantization Aware Layers from Network

The `removeQuantizationAwareLayers` function extract the original layers from the quantization aware network and replaces the quantization aware layers with the original underlying layers.

```
function net = removeQuantizationAwareLayers(qatNet)
    lg = layerGraph(qatNet);

    % Find quantization aware training layers and replace with the
    % underlying layers.
    for idx = 1:numel(lg.Layers)
        currentLayer = lg.Layers(idx);

        if isa(currentLayer,"QuantizedConvolutionBatchNormTrainingLayer")

            cLayer = currentLayer.Network.Layers(1);
            lg = replaceLayer(lg,cLayer.Name,cLayer);

            bLayer = currentLayer.Network.Layers(2);
            lg = replaceLayer(lg,bLayer.Name,bLayer);
        end
    end

    net = assembleNetwork(lg);
```

```
end
```