[HW5_prob1] Resnet20 Quantization-aware Training (15pts)

1. Modify your “[HW_Code5]_VGG16_Quantization_aware_train.ipynb” to train Resnet20 with 4bit Quantization-aware training, with 4bits for both weight and input (activation) quantization
2. You do not need to quantize the first convolution layer (as there are only 3 RGB channels, we do not want to lose important information here) and last fully connected layer (as it is the decision making layer).
3. For 4 bit quantization:
   a. Find x_int and w_int for the 2nd convolution layer
   b. Check the recovered psum has similar value to the un-quantized original psum
   c. Try to have ~ 90% accuracy
4. Iterate the process for 2 bit quantization as follows:
   a. Please quantize the first conv layer as well.
   b. Try to get ~ 90% accuracy (not necessarily need to achieve, but prefered)
   c. No need to do steps 3-a and 3-b from above point. Just perform quantization aware training after quantizing first layer as well.

[HW5_prob2] : VGGNet Customized Cost Function (15 pts)

- Train your VGGNet (without Quantization) to achieve preferably 90%.
- Then, check the first conv layer's weights' absolute sum
- Then, adjust the loss function to minimize above term by having loss2 as shown in "[Code13]_Customized_Cost_Function.ipynb". e.g., loss = loss1 + loss2
- Check your accuracy and the first conv layer's weights' absolute sum
- Try to run again with optimal regularization factor (gamma) to achieve >80% accuracy by having regularization factor, e.g., loss = loss1 + gamma * loss2.
- Now, check your new model's accuracy and conv layer's weights' absolute sum

[HW5_prob2] :FAQ

Q1. What accuracy are we expected to achieve?

There is no fixed accuracy range specified, but achieving *very low accuracy* (e.g., around 50%) even at the optimal gamma is not ideal. You should observe a clear **improvement trend** in both accuracy and weight sum as you move from γ = 1 toward the optimal γ value.

Q2. Will there be point deductions for low accuracy or poor weight convergence?

We will not grade based on absolute accuracy alone. However, you **must demonstrate a clear trend of improvement** in both accuracy and weight sum as gamma approaches its optimal value. A lack of this trend may affect your score.

Q3. How should I choose my learning rate and number of epochs?

You can experiment with different learning rates or more epochs to improve accuracy. However, for a fair comparison, **you must keep the learning rate and number of epochs consistent across all three cases**.

Q4. Can I change my technique or training setup between cases?

No. Whatever technique or number of epochs you use in your first case, use the same setup for the remaining ones. Only then will your accuracy and weight-sum comparisons be meaningful — otherwise, differences could be caused by training parameters rather than the customized loss function.

[HW5_prob2] :FAQ

Q5: In HW5 Q2, are we supposed to train two new VGG16 models using loss = loss1 + gamma * loss2, or just modify the existing VGG16 model (accuracy > 90%) and train for a few more epochs?

A: You should **modify the existing VGG16-trained model** (from HW3) and then **retrain it with the modified cost function**.

When applying the new cost function, **save models separately** for:

1. **γ = 1** → loss = loss1 + loss2
2. **Optimal γ value** → loss = loss1 + γ * loss2

Ideally, your submission should show the following three results:

- Original model's accuracy (**>90%**)
- Accuracy with γ = 1
- Accuracy with optimal γ

[HW5_prob2] :FAQ

Q6: How can we make an apples-to-apples comparison of loss, accuracy, and absolute weight sum if we train three separate models for the three scenarios? Won't each training run produce different parameters?

A: That's a good question. The key difference lies between **post-training quantization (HW4)** and **cost-function modification (HW5)**:

- **In HW4**, we did *not* retrain the model. We performed **post-training quantization**, meaning we took an already trained model and simply rounded its weights to the nearest quantized value (e.g., 8-bit or 4-bit). Since there's no backpropagation involved, the parameters remain nearly the same—so it makes sense to directly compare the accuracy drop across quantized versions.
- **In HW5**, however, we **modify and retrain** the model with different cost functions.

  For example:

  - Base case: loss = nn.CrossEntropyLoss() → proven to give >90% accuracy
  - Modified case: loss = nn.CrossEntropyLoss() + conv1.weight.abs().sum() → equal weightage may hurt accuracy
  - Tuned case: loss = nn.CrossEntropyLoss() + γ * conv1.weight.abs().sum() → reducing γ favors accuracy

Each training run has a *different cost function*, so the goal is **not identical parameters** but to **observe how accuracy and weight-sum change with different loss formulations**.

Even though models are trained separately, this comparison highlights how the **choice of cost function** affects model performance.

[HW5_prob2] :FAQ

Q7: Do we need to rewrite the code in the same notebook three times to train the models? Or can we run the code three times, save the models, and then load them in one notebook to show the test accuracy? Also, are training logs required?

A: You can **run the code three times** (once for each case) and **save the three trained models**. Then, in your final notebook, **load the saved models** and display their **test accuracies** — that's sufficient.

Please **remove all training logs** before submitting.

Make sure the **cost function with the optimal gamma value** is clearly reported or visible in the PDF you submit.

Q8: For the customized loss (loss2), should we minimize the weights of only the first convolution layer, all convolution layers, or all layers?

A: You should minimize the weights **only for the first convolution layer**.