This homework contains 3 problems. There are no star problems in this homework. The topics covered in this homework include: DNN training with quantized gradients, derivation of weight initialization, BatchNorm absorption. This homework is due on Oct. 9 via Gradescope. No extension will be provided, so make sure to get started as soon as possible.

Problem 1: Training Neural Networks using Python with Quantized Gradients

In this problem, you will need the MNIST dataset from `http://yann.lecun.com/exdb/mnist/`. You can initiate the dataset directly using the PyTorch API. Your job is to write a Python code to train a neural network with an MLP architecture of 784-512-256-256-10 that performs the classification task on the MNIST dataset. You are to use the vanilla version of SGD. You are encouraged to read through all questions in this problem before starting to code.

1. You can find tutorials on how to adjust learrning rate in PyTorch at `https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate`. In this problem, train the network with **a**: constant learning rate, **b**: a cosine annealing learning rate, and **c**: a step-decaying learning rate (learning rate decreases every x epochs). Plot the convergence curve of the test error of your network as a function of time (measured in epochs) for each learning rate scheduler.

2. For the model trained using scheduler **a**, plot the per-tensor variance of each weight gradient as a function of time (measured in epochs), and record the maximum and minimum values of recorded variances during training.

3. You are asked to retrain this network with scheduler **a**, but using close-to-minimal (CTM) quantized weight gradients. The challenge is to determine a suitable clipping level and quantization step-size (Hint: use your answers above). Show convergence curves of training with CTM quantized gradients and report the number of bits used for each tensor.

4. Compare the test error convergence curve obtained using CTM quantized weight gradients in Part 2, with those obtained by quantizing the weight gradients to: a) 6-b, and b)7-b, for all layers.

Problem 2: Deriving Weight Initialization Formulas

In this problem, your task is to derive the two key equations utilized in the He initialization scheme (`https://arxiv.org/abs/1502.01852`). In what follows, we assume weights are zero-mean and independent. Further, activations are independent but non-zero mean

(because of their rectifying nature).

1. Show that during for the forward propagation we have:

$$Var(y_L) = Var(y_1)\left(\prod_{l=2}^{L}\frac{1}{2}n_l Var(w_l)\right)$$

where y_l, w_l, and n_l are the pre-activations, weights, and forward dot-product length at layer l and, L is the number of layers.

2. Show that during the backward propagation we have:

$$Var(\Delta x_2) = Var(\Delta x_{L+1})\left(\prod_{l=2}^{L}\frac{1}{2}\hat{n}_l Var(w_l)\right)$$

where Δx_{l+1} and $\hat{n}_l$ are the activation gradients and backward dot-product length at layer l.

3. In a convolutional layer, how are the forward dot-product length n_l and backward dot-product length $\hat{n}_l$ related?

4. Explain how the He initialization prevents the vanishing or explosion of activations in the forward propagation.

5. Explain how the He initialization prevents the vanishing or explosion of gradients in the backward propagation (HINT: the answer to this question is not the same as that of the previous one).

Problem 3: BatchNorm Absorption

Recall in Batch Normalization, the output $\langle \mathbf{w}, \mathbf{x} \rangle$ of a layer with activation or feature map $\mathbf{x}$ with a weight or filter $\mathbf{w}$ is transformed as:

$$\gamma \frac{\langle \mathbf{w}, \mathbf{x} \rangle - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta$$

where γ, β, μ, and σ are the BatchNorm (BN) parameters, ϵ is a numerical stability constant, and $< \cdot, \cdot >$ denotes the dot product or convolution depending on the context.

If only inference needs to be performed, it is possible to eliminate the extra computations required by BN by absorbing its parameters into the weights. This can be rewriting the above equation as:

$$\langle \hat{\mathbf{w}}, \mathbf{x} \rangle + b$$

where $\hat{\mathbf{w}}$ and b are a new weight/filter and a bias, respectively.

1. Derive expressions for $\hat{\mathbf{w}}$ and b as a function of $\mathbf{w}$, γ, β, μ, σ, and ϵ?

2. For the case of convolutions, write a Python script that returns the BN 4D tensor $\hat{\mathbf{w}}$ and 1D vector b for standard 2D convolution given a original weight tensor $\mathbf{w}$ and BN parameters γ, β, μ, σ, and ϵ. You only need to submit a script for this question (Hint: the solution is not trivial and requires *broadcasting*).

3. For the MLP model in Problem 1, add a BN layer after each of the first three fully connected layers and retrain the network from scratch. Plot the convergence of test error for the MLP before and after adding the BN layer. Pay attention to the order between the BN layer and the ReLU unit. How many parameters does this new MLP have compared to the old one?