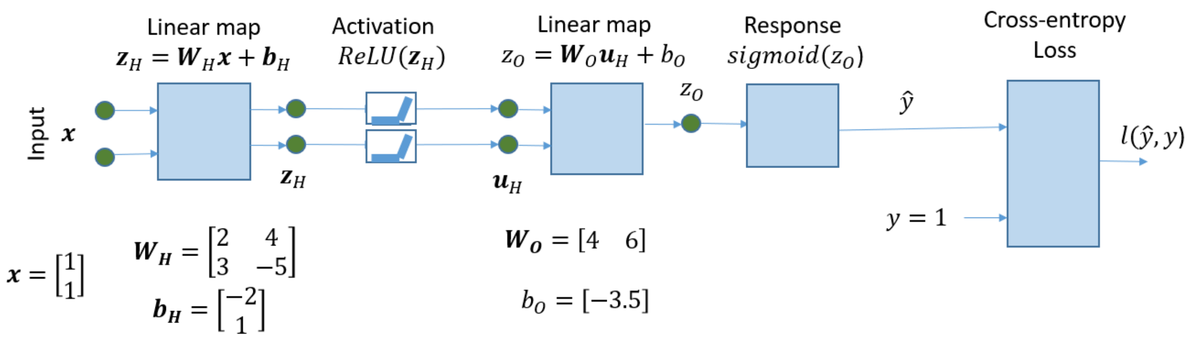
Deep Learning (Spring 2022) Homework 1
Assigned: Feb 15th 2022
Due: **Midnight, Feb 28, 2022**

This homework is to be done individually. Please submit your solutions to the theory questions in a single PDF document. Please submit your solutions to the coding questions in a single Python notebook (in ipynb) format. All code must use PyTorch.

Q1) (Backpropagation, 25 Points) Consider the two layer neural network shown below. Your goal is to compute the gradients of the loss function for a training input x, given the current values of weights and biases as shown in the figure. You will do this in two steps.



1. Forward pass: compute $\hat{y}$ and $l(\hat{y}, y)$, i.e., the network prediction and cross-entropy loss. The values of the input ***x*** and ground-truth response *y* are shown in the figure.
2. Backward pass: compute $\partial l / \partial W_O$, $\partial l / \partial b_O$, $\partial l / \partial W_H$ and $\partial l / \partial b_h$. Note that $\partial l / \partial W_O$ will have the same dimensions as W_O, and so on for the other weight and bias matrices/vectors.
3. Were some derivatives zero? If so, comment briefly on why this happened.

Q2) (ADAM Optimizer, 25 Points) In class we discussed the ADAM optimizer first proposed by Kingma and Ba (https://arxiv.org/abs/1412.6980). The optimizer includes two bias correction terms, $\widehat{m_t}$ and $\widehat{v_t}$. Read the paper and try and understand the role of bias correction in ADAM.

1. Read the paper and try and explain, briefly in 2-3 sentences, the role of bias correction in the ADAM optimizer.
2. Assume that the g_t (the gradient computed in each time step) is a stationary process; for this question, it will be sufficient to assume that the mean of the gradient does not change with time, i.e., $E(g_t) = E(g) = G$. Assuming $\beta_1 = 0.9$ and $G = 2$, compute $E(\widehat{m_t})$ for t={2,10,100}.

3. Repeat the question above, but this time assume that we are not using bias-correction, i.e., $\widehat{m_t} = m_t$.

Q3) (Multi-layer CIFAR-10 Classifier Coding, 25 Points) Train a (dense) neural network with three hidden layers with 256, 128, and 64 neurons respectively, all with ReLU activations (note, by layer we mean a linear transform followed by ReLUs), to classify ***grayscale*** images from the CIFAR-10 dataset available via `torchvision.datasets.CIFAR10`. You can use `torchvision.transforms.Grayscale` to convert CIFAR10 images to grayscale. Use a batch size of 64 and SGD optimizer with fixed lr=0.01. Display train- and test- loss curves, and report test accuracies of your final model. We leave the choice of how many epochs of training to run up to you; ideally you want to observe the training loss curve and stop when the training loss has flattened out.