Justin Lam (77333383)

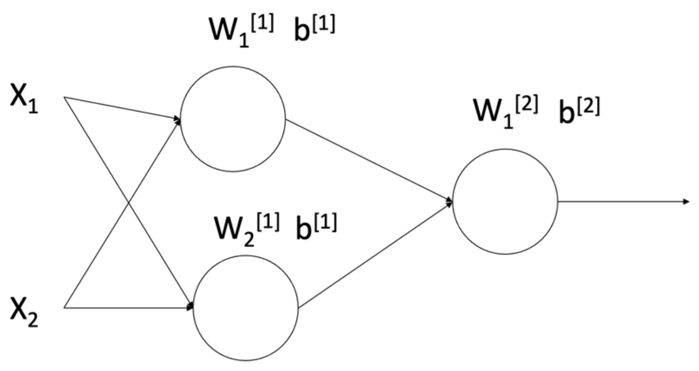
CIVL 498A HW-4: MLP, CNN.

Note: For hand calculation problems, you are supposed to write down your answers on a piece of paper and then submit the scanned version of the paper(s). Show your work by writing out derivations if there is any. For coding problems, you can either use Google Colab or Jupyter Notebook (on your own machine) to write and run the code. When submitting, submit only the ipynb file (you should use relative file directory for input files e.g., "./data.csv" in your code, so when I run the file on my local computer, files would load normally), and make sure when "run all" is clicked, all required results from the questions are shown (this is how I will grade your answer. If when I "run all" and your code crashes, it will be deemed as wrong).

1. **(Hand calculation)** Use the following table, and the given neural network architecture to solve a regression problem.

Note there are 5 parameters in this network, as $W_1^{[1]}$, $W_2^{[1]}$, $W_1^{[2]}$, $b^{[1]}$, and $b^{[2]}$.

Observations	**x_1**	**x_2**	**y**
1	2	2	20
2	3	4	28



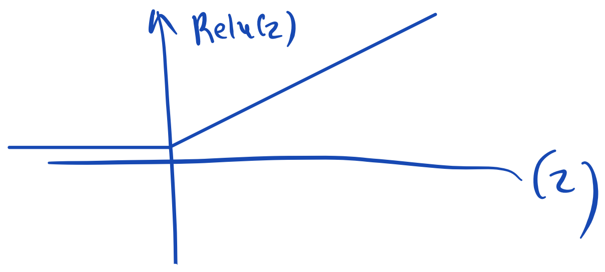
1.1. Assume all neurons use the ReLU activation function. Write out the forward propagation equations.

1.2. Using the squared L_2 loss without continuous sum (i.e., $J(\theta)$ in linear regression without the continuous sum), write out the derivatives for each parameter using the chain rule.

1.3. Use the table above to perform SGD for one epoch, and report the 5 parameters after each training example.

1.1

$$\text{Relu}(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$



Forward prop:

$$Z = W^T x + b \quad , \alpha = \text{Relu}(z)$$

$$Z_1^{[1]} = W_1^{[1]T} X + b^{[1]} \quad , \alpha_1^{[1]} = \text{Relu}\left(Z_1^{[1]}\right)$$

$$Z_2^{[1]} = W_2^{[1]T} X + b^{[1]} \quad , \alpha_2^{[1]} = \text{Relu}\left(Z_2^{[1]}\right)$$

$$Z_1^{[2]} = W_1^{[2]T} \alpha^{[1]} + b^{[2]} \quad , \alpha_1^{[2]} = \text{Relu}\left(Z_1^{[2]}\right) = \hat{y}$$

1.2

$$L_2 \text{ LOSS} = \sum_{i=1}^{n} \left(y_{\text{true}} - y_{\text{pred}}\right)^2$$

$$\frac{dL}{dW_1^{[2]}} = \frac{dL}{da_1^{[2]}} \times \frac{da_1^{[2]}}{dz_1^{[2]}} \cdot \frac{dz_1^{[2]}}{dW_1^{[2]}}$$

$$\frac{dL}{dW_1^{[1]}} = \frac{dL}{dz_1^{[2]}} \times \frac{dz_1^{[2]}}{d\alpha_1^{[1]}} \times \frac{d\alpha_1^{[1]}}{dZ_1^{[1]}} \times \frac{dZ_1^{[1]}}{dW_1^{[1]}}$$

$$\frac{dL}{dW_2^{[1]}} = \frac{dL}{dZ_1^{[2]}} \times \frac{dZ_1^{[2]}}{d\alpha_2^{[1]}} \times \frac{d\alpha_2^{[1]}}{dZ_2^{[1]}} \times \frac{dZ_2^{[1]}}{dW_2^{[1]}}$$

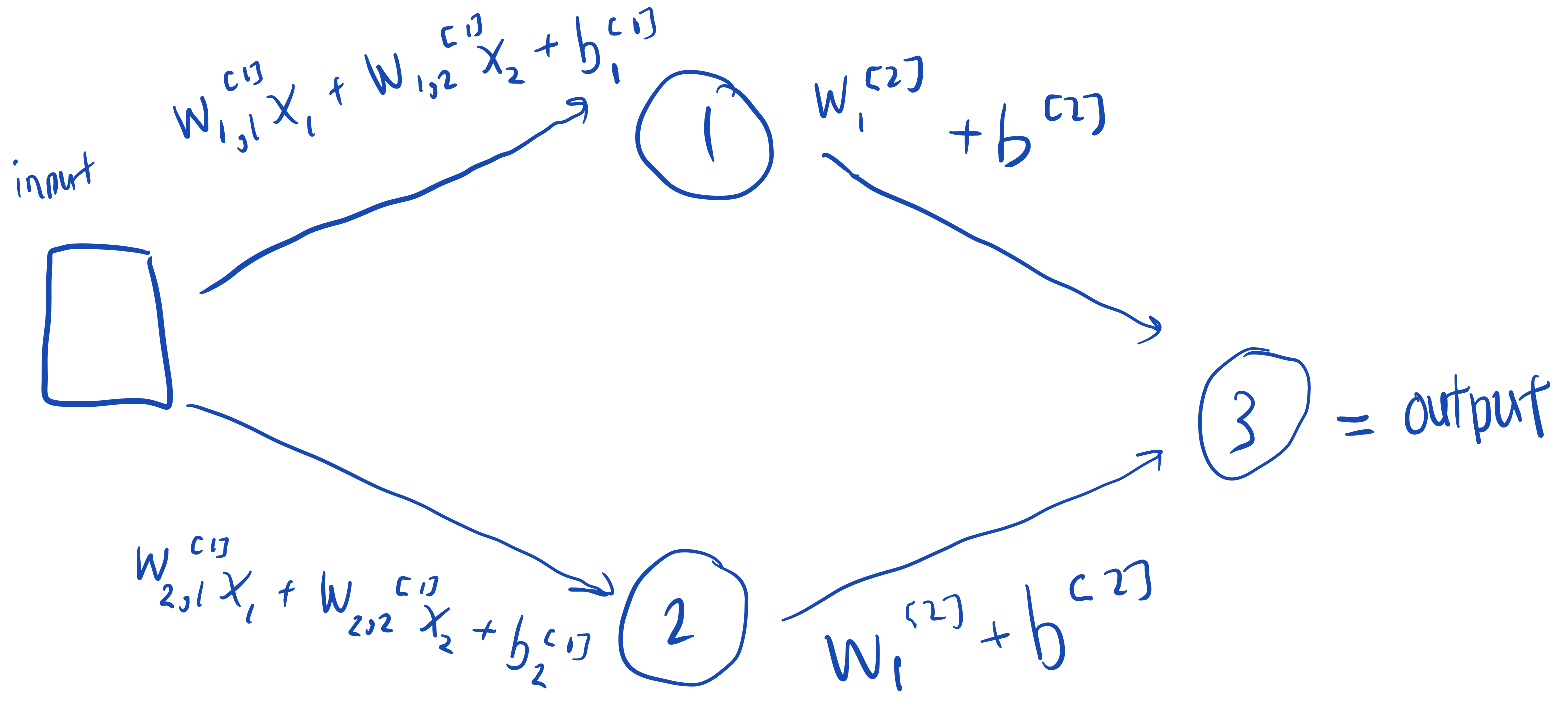
$$\frac{dL}{db^{[2]}} = \frac{dL}{d\alpha_1^{[2]}} \times \frac{d\alpha_1^{[2]}}{dZ_1^{[2]}} \times \frac{dZ_1^{[2]}}{db^{[2]}}$$

$$\frac{dL}{db^{[1]}} = \frac{dL}{dZ_1^{[2]}} \times \frac{dZ_1^{[2]}}{d\alpha_2^{[1]}} \times \frac{d\alpha_2^{[1]}}{dZ_2^{[1]}} \times \frac{dZ_2^{[1]}}{db^{[1]}}$$

1.3

Assume $b^{[1]} = b^{[2]} = 0$, $W^{[1]} = W^{[2]} = 1$

Assumed network architecture:



input
$W_{1,1}^{[1]}x_1 + W_{1,2}^{[1]}x_2 + b_1^{[1]}$
1
$W_1^{[2]} + b^{[2]}$
3
= output
$W_{2,1}^{[1]}x_1 + W_{2,2}^{[1]}x_2 + b_2^{[1]}$
2
$W_1^{[2]} + b^{[2]}$

① Try (2,2,20)

Node 1

(i) $(2\times1)+(2\times1)+0=4 \rightarrow \text{relu} \rightarrow z=4 \rightarrow f(z)=4$

(ii) $4\times1+0=4$

Node 2

(i) $(2\times1)+(2\times1)+0=4 \rightarrow \text{relu} \rightarrow z=4 \rightarrow f(z)=4$

(ii) $4\times1+0=4$

For (2,2,20) predicted value is 8

$$\frac{dL}{d\alpha_1^{[2]}} = \sum_{i=1}^{n} -2(y_{true}-y_{pred})$$

$$\frac{dL}{dw_1^{[2]}} = \sum_{i=1}^{n} -2(y_{true} - y_{pred}) \times 1 \times a^{[1]}$$

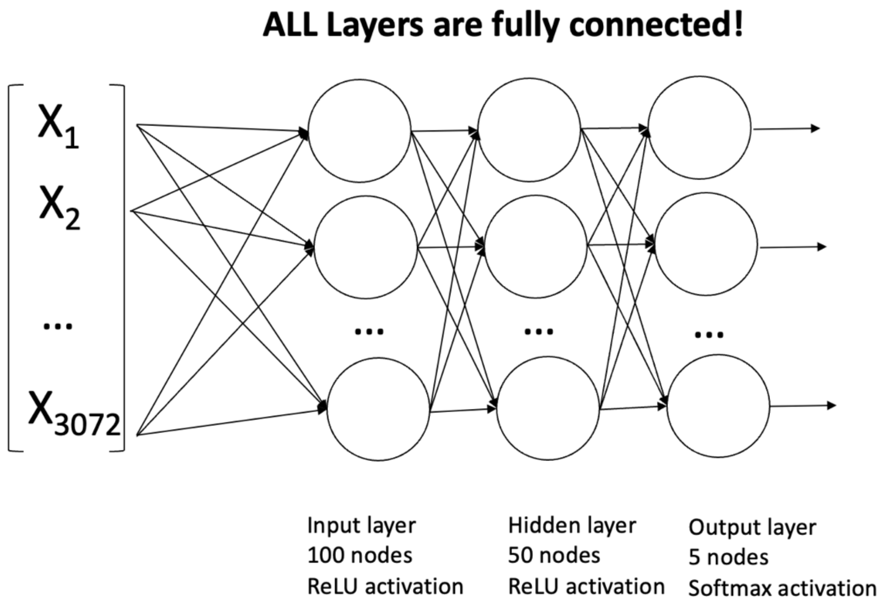
$$\frac{dL}{dw_1^{[1]}} = \sum_{i=1}^{n} -2(y_{true} - y_{pred}) \times W_1^{[2]} \times 1 \times X$$

$$\frac{dL}{dw_2^{[1]}} = \sum_{i=1}^{n} -2(y_{true} - y_{pred}) \times W_2^{[2]} \times 1 \times X$$

$$\frac{dL}{db^{[2]}} = \sum_{i=1}^{n} -2(y_{true} - y_{pred}) \times 1 \times 0 = 0$$

2. **(Coding Problem)** Using the CIFAR-100 dataset available on Pytorch (import this dataset from Pytorch in your code using cifardataset.py instead of downloading from the website yourself), conduct a multi-class classification problem for the superclass "large man-made outdoor things" using neural networks (i.e., train your neural networks to classify an image into one of the five classes).

Use multi-layer perceptron with the following structure to classify input image into one of the five classes:

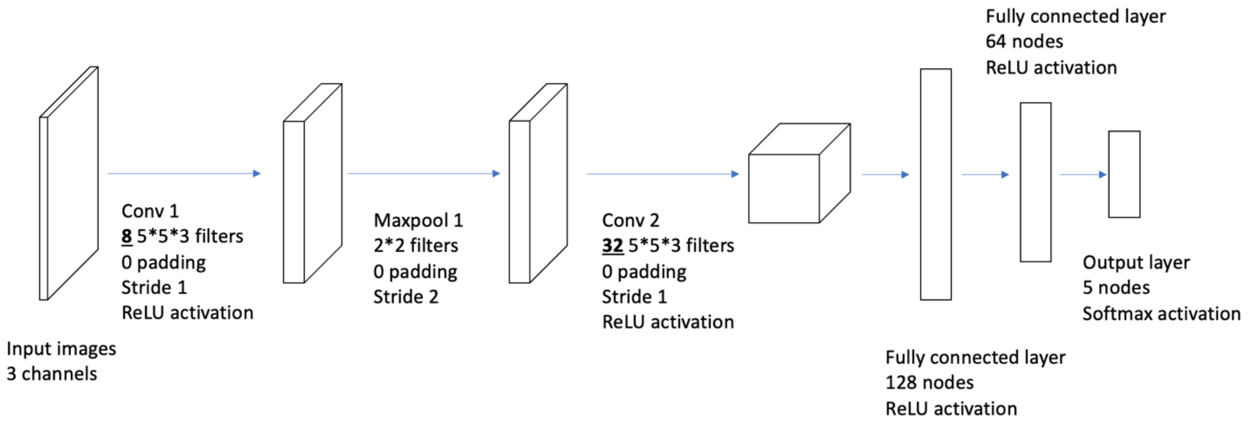


Training hyperparameters: number of epochs is 10, batch size is 100, and learning rate is 0.001, cross entropy as loss, Adam as the optimizer.

2.1. Report the precision and accuracy of the network on the test set images.

2.2. Draw the figure of average loss per epoch during training (i.e., x axis is the number of epoch, y axis is the average cross entropy loss of that epoch).

3. **(Coding Problem)** Same as Q2, but with the new network structure as below (Note, this structure is not optimal in any sense except that it will not run forever on your computer):



Training hyperparameters: number of epochs is 10, batch size is 4, and learning rate is 0.001, cross entropy as loss, Adam as the optimizer.

3.1. Report the precision and accuracy of the network on the test set images.

3.2. Draw the figure of average loss per epoch during training (i.e., x axis is the number of epoch, y axis is the average cross entropy loss of that epoch).