COMP 432 Intro. to Machine Learning (Fall 2024)

Major Assignment #2

Due: 11:59PM, October 20th, 2024

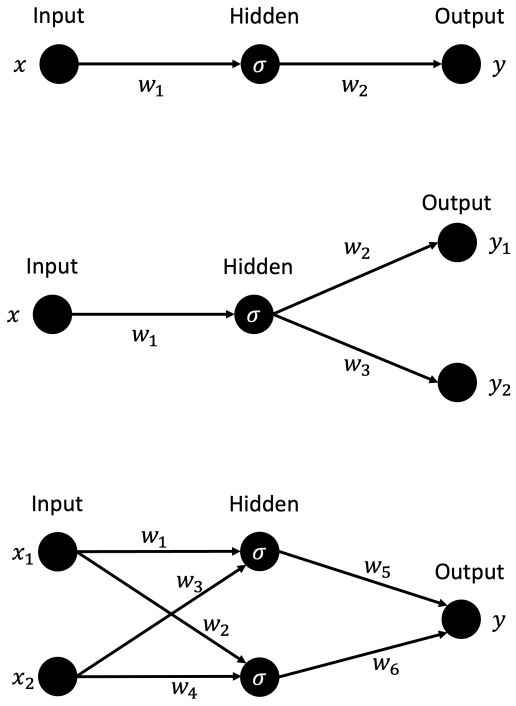
Note Your will be submitting two separate files from this assignment as follows:

(a) **One(1) .pdf file:** containing answers to Question as well as reported results from coding you develop. Include snapshots of the pieces of code you developed in the appendix.

(b) **One(1) .zip folder:** containing all developed Python codes including a README.txt file on explaining how to run your code.

Theoretical Questions

Question 1 Consider the following three neural networks:



where σ is the sigmoid activation function.

(a) For each neural network, derive an expression for the output of the neural network in terms of the input and the weights.

(b) Assume that you are using the following loss functions:

$$l = \sum_{i}^{n} \frac{1}{2}(y_i - t_i)^2 \tag{1}$$

where t is the target and n is the number of outputs for a given input. For each neural network, compute $\frac{\partial l}{\partial w_i}$ for each w_i (i.e. $\frac{\partial l}{\partial w_1}$, $\frac{\partial l}{\partial w_2}$, etc ...). (hint: use the chain rule)

(c) Given the table below of inputs and weights, compute the output of each network. Show your work.

	Input	Weights
Network 1	$\mathbf{x} = 0.7$	$\mathbf{w} = [0.4, 0.5]$
Network 2	$\mathbf{x} = 0.3$	$\mathbf{w} = [0.1, 0.2, 0]$
Network 3	$\mathbf{x} = [0.4, 0.8]$	$\mathbf{w} = [1, 0.2, 0.2, 0, 0.5, 0.1]$

Question 2 Consider the 3rd network and the loss given the previous question, with the additional assumption that the output has a sigmoid activation function. The following are steps resulting from deriving the loss for that specific case (Reference). The aim in this question is to perform Backpropagation. Assume that the weights are initialized as ($w_1 = 0.2$, $w_2 = 0.1$, $w_3 = 1.0$, $w_4 = -1.5$, $w_5 = 0.7$, $w_6 = -0.3$), and that you are using sigmoid as the activation function at each neuron, with a learning rate of $\eta = 0.1$. Use the data point $\mathbf{x} = [0.2, 0.9]$ with the target $y = 0.7$.

(a) Perform a forward pass through the network, showing the final output as well as the output at each hidden unit.

(b) For each output unit k, the error term δ_k is calculated as $\delta_k \leftarrow g'(x_k) \times Err_k$, where $g(x)$ is the activation function, x_k is the input to that unit, and Err_k is the error between the output O_k and the target T_k, given as $Err_k = O_k - T_k$. Derive an expression for δ_k in terms of O_k and T_k, and use this expression to compute δ_k for the output of your network.

(c) For each hidden unit h, the error term δ_h is calculated as $\delta_h \leftarrow g'(x_h) \times Err_h$, where $g(x)$ is the activation function, x_h is the input to that unit, and Err_h is the error of the output O_h, given as $Err_h = \sum_{k \in \text{outputs}} w_{hk}\delta_k$. Derive an expression for δ_h in terms of O_h, w_{hk}, and δ_k, and use this expression to compute δ_h for each of the hidden units.

(d) Each weight w_{ij} connecting nodes i and j is updated as $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$, where $\Delta w_{ij} = -\eta \delta_j O_i$. Update all the weights in your network.

(e) Perform a forward pass and compute the error at the output. Compare the error with that from the initial forward pass and comment on the results.

(f) Explain, in your own words, what would change in this process above if you are to use a ReLU activation function instead of sigmoid.

Question 3 For each of the following equations, draw the architecture of the neural network. You need to label each of the nodes, edges, and activation functions.

(a)

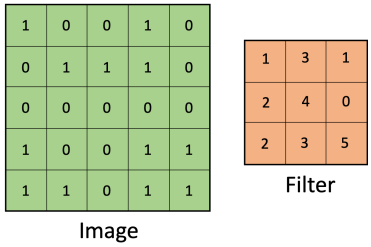
$$\begin{aligned} y = \sigma\Big(& w_{41} \cdot \text{ReLU}(w_{11}x_1 + w_{12}x_2 + b_1) \\ & + w_{42} \cdot \text{ReLU}(w_{21}x_1 + w_{22}x_2 + b_2) \\ & + w_{43} \cdot \text{ReLU}(w_{31}x_1 + w_{32}x_2 + b_3) + b_4 \Big) \end{aligned}$$

(b)

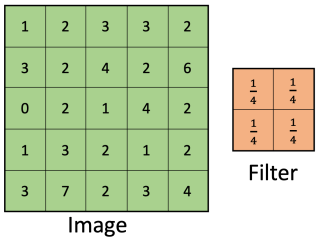
$$\begin{aligned} y = \sigma\Big(& w_6 \cdot \text{ReLU}\big(w_2 \cdot \text{ReLU}(w_1 x + b_1) + b_2\big) \\ & + w_7 \cdot \tanh\big(w_4 \cdot \text{ReLU}(w_2 \cdot \text{ReLU}(w_1 x + b_1) + b_2) \\ & + w_5 \cdot \tanh(w_3 \cdot \text{ReLU}(w_1 x + b_1) + b_3) + b_4\big) + b_5 \Big) \end{aligned}$$

Question 4 For each of the following 2D maps (images), apply convolution using the given filter with the specified parameters (assume no padding):

(a) Stride = 1



(b) Stride = 2

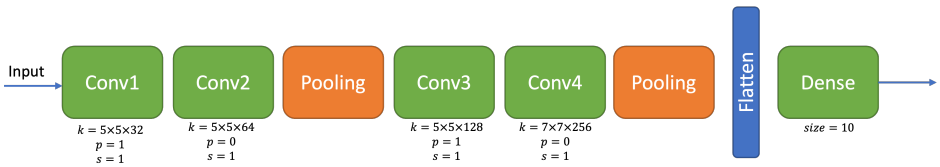


(c) MaxPooling with kernel size = 2

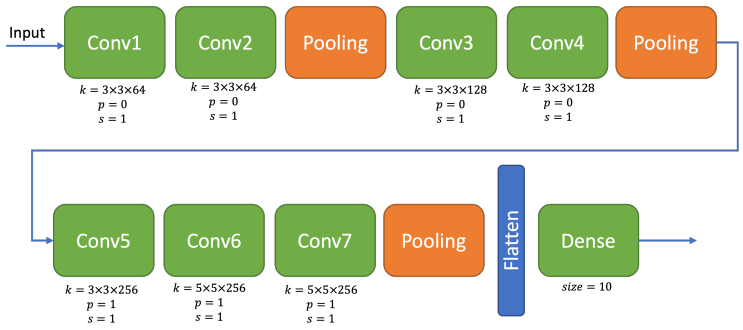
4	2	4	4	3	4
4	3	5	2	6	1
2	4	3	4	2	4
1	6	4	1	2	3
3	7	2	5	6	1
5	4	1	2	4	5

Question 5 For each of the architectures given below, write down the output dimensions (in $H \times W \times N$ format) of each layer. Refer to PyTorch documentations on how to handle cases with fractions. (note: in this question, pooling refers to Max Pooling layer with a 2x2 kernel and a stride of 2). The input image size is $224 \times 224 \times 3$.

(a)



(b)



Question 6 Design a Convolutional Neural Network (CNN) for image classification with the following requirements:

- Input image size is $224 \times 224 \times 3$.
- The CNN encoder contains exactly four(4) convolutional layers.
- You are allowed a maximum of two MaxPool layers, each with a 2×2 kernel and a stride of 2.
- The output dimensions before flattening is $7 \times 7 \times 256$.
- For the convolutional layers, filters (kernels) have to be of odd dimensions with a maximum size of 7×7. The maximum padding allowed is 2 and the maximum stride allowed is 2.
- The classifier contains one Fully-Connected (FC) layer with 5 output prediction classes.

You are required to draw/sketch the whole CNN pipeline describing the details of each layer in the network (kernel size, padding, stride, input/output dimensions).

Implementation Questions

Question 1 Develop a neural network using PyTorch for BMI estimation. Given information about the user, such as their gender, height, and weight, your model should **classify** the user into one of the BMI indices. Refer to Kaggle for access to the dataset.

(a) Load and split your dataset into train and test sets with a 7:3 ratio using methods from PyTorch.

(b) Implement and train your neural network. Note and plot the performance in terms of batch vs loss/accuracy across a sufficient number epochs (until convergence if possible).

(c) Analyze the performance on the test set using common classification metrics.

(d) It is time to fine-tune your model with the aim of enhancing its performance. Implement a grid search considering the learning rate and batch size parameters. Try variations of learning rates (i.e. 0.1, 0.01, 0.001, 0.0001, etc) and batch sizes (8, 16, 32, 64, etc). Compare the results in terms of training curves and testing performance.

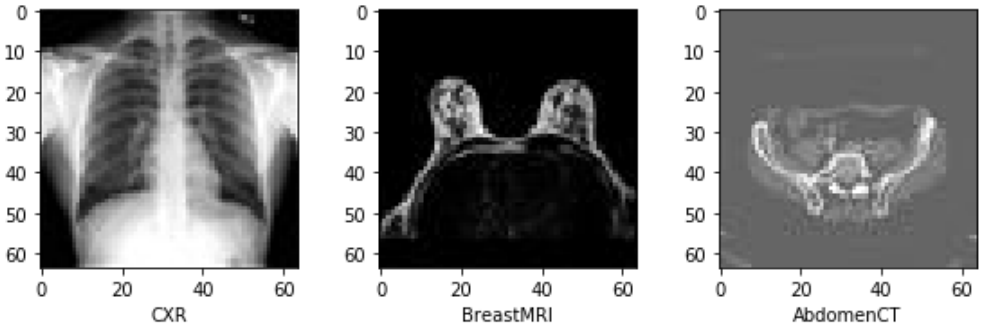
Question 2 Develop a neural network using PyTorch for Concrete Strength estimation. Given information about the concrete, such as the amount of cement, water, superplasticizer, etc, your model should **estimate** the strength of the concrete. Refer to Kaggle for access to the dataset.

(a) Load and split your dataset into train and test sets with a 7:3 ratio using methods from PyTorch.

(b) Implement and train your neural network. Note and plot the performance in terms of batch vs loss across 100 epochs. Analyze the Mean Squared Error (MSE) and Mean Average Error (MAE) on the test set.

(c) It is time to fine-tune your model with the aim of enhancing its performance. Implement a grid search considering the learning rate and batch size parameters. Try variations of learning rates (i.e. 0.1, 0.01, 0.001, 0.0001, etc) and batch sizes (8, 16, 32, 64, etc). Compare the results in terms of training curves and testing performance.

Question 3 Design and implement a CNN to be used in a task of Medical Image Classification. Given an image of an MRI/CT/X-ray scan, your model is to predict the body part being scanned. Some sample images are shown in the figure below. You should use and download the dataset from Kaggle, which you can reduce to 1000 data points per class. You are required to split the data into train and test sets with 7:3 ratio. You should design and build a simple CNN using PyTorch, and train it on the given dataset for 10 epochs. Report on your design, your choice of hyperparameters, as well as the training accuracy/loss plots. You can use scikit-learn's classification report tool for numerical analysis.



You can use the code segments below to guide you in loading the data (documentations: ImageFolder, ToTensor, torch.utils.data). Make sure that your

dataset contains folders corresponding to the different classes. Refer to PyTorch documentations for the required libraries to be imported, and for examples on training CNN.

```
================================================
dataset=ImageFolder(path,ToTensor()) # loads dataset from path
train_set,test_set=torch.utils.data.random_split(dataset,[0.7,0.3]) # splits dataset into specified ratios
train_loader=DataLoader(train_set,shuffle=True,batch_size=16) # create train loader
test_loader=DataLoader(test_set,batch_size=16) # create test loader
================================================
```