Homework Assignment 4

COGS 181: Neural Networks and Deep Learning

Due: 11:59pm, Thursday, February 29th, 2024 (Pacific Time).

Instructions: Answer the questions below, attach your code, and insert figures to create a PDF file; submit your file via Gradescope. You may look up the information on the Internet, but you must write the final homework solutions by yourself.

Late Policy: 5% of the total points will be deducted on the first day past due. Every 10% of the total points will be deducted for every extra day past due.

System Setup: You can install Anaconda to setup the Jupyter Notebook environment. Most packages have been already installed in Anaconda. If some package is not installed, you can use `pip` to install the missing package, that is, just type `pip install PACKAGE_NAME` in the terminal.

Grade: ____ out of 100 points

1 (20 points) Conceptual Questions

1. (**5 points**) For image recognition problems (e.g. detecting a cat in a photo), which architecture of neural network would be best suited to solve the problem?

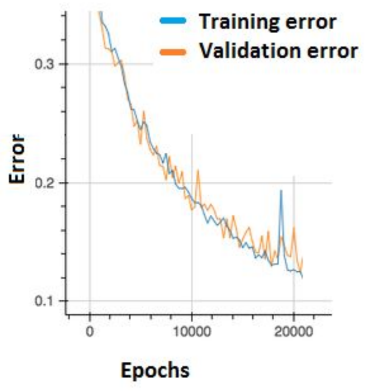
   A. 2-layer neural networks.
   B. Convolutional Neural Network (CNN).
   C. 4-layer neural networks.
   D. Perceptron.

2. (**5 points**) Which of the following gives non-linearity to a neural network?

   A. Stochastic Gradient Descent (SGD).
   B. Rectified Linear Unit (ReLU).
   C. Convolutional layer (Conv).
   D. Average Pooling layer (AvgPool).

3. (**5 points**) In the figure below, we observe that the error curves (both training and validation) have many "ups and downs":

Training and Validation Error Curves



Should we worry about the training process? Choose the **best** explanation below:

A. Yes, because the "ups and downs" means there is a problem with the learning rate of the neural network.
B. Yes, because the "ups and downs" means there is a problem with the model design of the neural network.
C. No, as long as there is a cumulative decrease in both training and validation errors, we do not need to worry.
D. No, since the error is too low (< 0.2 after 10,000 epochs), we do not need to worry about the oscillation of the error.

4. (**5 points**) Which of the following is **the most probable** about model capacity (model capacity means the ability of the neural network to approximate complex functions)?

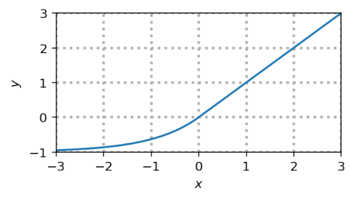
A. As the number of data points in a mini-batch increases, model capacity increases.
B. As the training loss after convergence increases, model capacity increases.
C. As the learning rate increases, model capacity increases.
D. As the number of learnable parameters increases, model capacity increases.

2 (15 points) Activation Functions

For **each** activation on the left-hand side, find and connect the corresponding plot of the activation function on the right-hand side.

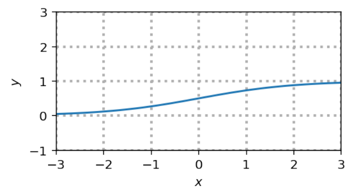
(a) ReLU

(1)



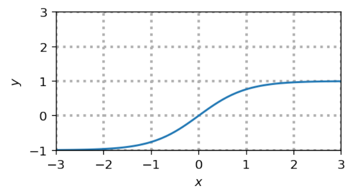
(b) LeakyReLU

(2)



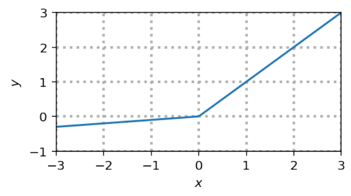
(c) Sigmoid

(3)



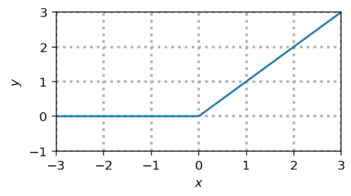
(d) Tanh

(4)



(e) ELU

(5)



3 (20 points) 2-D Convolution

Suppose we have a small network for image recognition. The input image $\mathbf{x} \in \mathbb{R}^{4\times 4}$. The network has three layers named Conv, MaxPool, and ReLU:

$$\text{Image } \mathbf{x} \rightarrow \text{Conv} \xrightarrow{\mathbf{z}_1} \text{MaxPool} \xrightarrow{\mathbf{z}_2} \text{ReLU} \rightarrow \mathbf{z}_3$$

where $\mathbf{z}_1, \mathbf{z}_2$ are intermediate output and $\mathbf{z}_3$ is the final output. The details of the layers are given below:

Conv: A single-channel convolution layer with 3×3 kernel, stride size 1 and no padding.

MaxPool: A max pooling layer with 2×2 kernel and stride size 2.

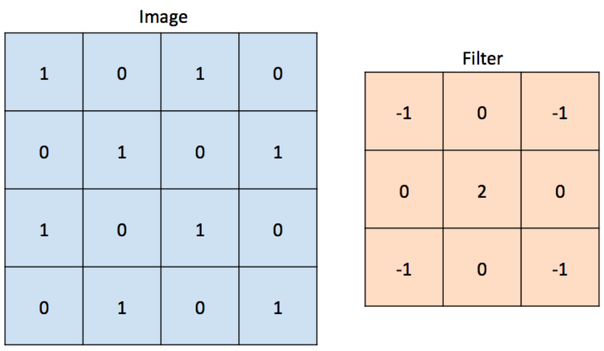
ReLU: A ReLU activation layer which has $\max(0, z)$ as the output of input z.

3.1 (6 points) Output Shape Computation

Please calculate the shape of the outputs and fill in the blanks below:

1. Shape of input $\mathbf{x}$: 4×4.
2. Shape of intermediate output $\mathbf{z}_1$: $\underline{\hspace{2em}} \times \underline{\hspace{2em}}$.
3. Shape of intermediate output $\mathbf{z}_2$: $\underline{\hspace{2em}} \times \underline{\hspace{2em}}$.
4. Shape of final output $\mathbf{z}_3$: $\underline{\hspace{2em}} \times \underline{\hspace{2em}}$.

3.2 (14 points) Output Value Computation



Image

1	0	1	0
0	1	0	1
1	0	1	0
0	1	0	1

Filter

-1	0	-1
0	2	0
-1	0	-1

Assume the values of input image $\mathbf{x}$ and the filter of the Conv layer are given in the figure above. Please compute the values of outputs $\mathbf{z}_1, \mathbf{z}_2$ and $\mathbf{z}_3$ by hand.

Hint: The values of $\mathbf{z}_1, \mathbf{z}_2$ and $\mathbf{z}_3$ should match the shapes in the previous sub-problem.

4 (10 points) 1x1 Convolution

1×1 convolution, also called 1-by-1 convolution, is a kind of convolutional layer which has kernel size 1×1. Although the kernel size is only 1×1, it does **not** mean that this convolution can **only** scale the input: When the input has multiple channels, the 1×1 convolution will linearly transform the input channels to output channels for each spatial location. Thus, you can observe that after the 1×1 convolution, the height and width of the output are the same as those of the input, but the channels have changed. More details about 1×1 convolution can be found from http://cs231n.github.io/convolutional-networks/#conv.

In this section, assume the input image $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ where C,H and W stand for channels, height, and width of the image. We use Conv-1x1(m, n) to represent the 1x1 convolutional layer with input channels m and output channels n. Essentially, conv-1x1(m, n) refers to n 1-by-1 convolution filters of size $m \times 1 \times 1$.

4.1 (6 points) Output Shape Inference

If the network is formed as below:

$$\text{Image } \mathbf{x} \rightarrow \text{Conv-1x1}(C, C_1) \xrightarrow{\mathbf{z}_1} \text{Conv-1x1}(C_1, C_2) \rightarrow \mathbf{z}_2$$

Please calculate the shape of outputs and fill in the blanks below:

1. Shape of input $\mathbf{x}$: $C \times H \times W$.
2. Shape of intermediate output $\mathbf{z}_1$: _____ $\times$ _____ $\times$ _____ .
3. Shape of final output $\mathbf{z}_2$: _____ $\times$ _____ $\times$ _____ .

4.2 (4 points) Model Capacity of Network

In this subsection, we define the **model capacity** as the ability of the neural network to approximate complex functions. For example, assume set S_A contains all the functions that can be represented by network A and set S_B contains all the functions that can be represented by network B. If $S_A = S_B$, it means network A and network B have the **same** model capacity. If $S_A \supset S_B$, it means network A has **higher** model capacity than network B.

Note: The definition of model capacity may vary under different contexts. Please stick to the definition above to solve the question below.

If we have two networks:

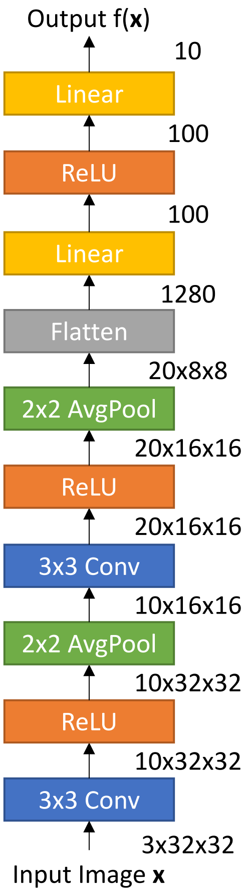
Network 1: Image $\mathbf{x} \rightarrow$ Conv-1x1$(C, C) \rightarrow$ Conv-1x1$(C, C) \rightarrow \mathbf{z}$.

Network 2: Image $\mathbf{x} \rightarrow$ Conv-1x1$(C, C) \rightarrow \mathbf{z}$.

Does the network 1 have **higher** capacity than network 2, or the **same** model capacity as network 2? Why?

5 (35 points) Train a Convolutional Neural Network

In this question, we are going to train a convolutional neural network (CNN) to classify the CIFAR-10 dataset $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ where $\mathbf{x} \in \mathbb{R}^{3\times32\times32}$ and $y \in \{0, 1, ..., 9\}$. Assume the images are already normalized. The network structure of the CNN is given on the right, with the shape of the intermediate output marked next to the arrow:



1. For an input image $\mathbf{x}$, we apply two convolutional blocks at first. Each convolutional block has one 3×3 convolutional layer, one ReLU activation layer, and one 2×2 average pooling layer. The two convolutional layers have stride 1 and padding size 1. The intermediate output after two convolutional blocks has the shape $20 \times 8 \times 8$.

2. Then, we flatten the previous output from shape $20\times8\times8$ into 1280-dimensional vectors. The flattened output will be passed through one linear layer, one ReLU activation layer, and another linear layer.

Besides, we use cross-entropy as the loss function and the stochastic gradient descent (SGD) as the optimizer.

Note:

1. For the other details of convolutional layers and linear layers (e.g. in-channels and out-channels), you can infer from the network structure on the right.

2. The shape mentioned above is for a single image. However, in the implementation, you need to deal with a mini-batch of images. Thus, you may find that the shape of tensors in the network has one more dimension (e.g. one input image has shape $3 \times 32 \times 32 \rightarrow$ a mini-batch containing 4 input images has shape $4 \times 3 \times 32 \times 32$).

3. You may notice that there is no softmax layer put on top of the network structure on the right. The reason is that we use `nn.CrossEntropyLoss()` as the loss function, which already has a built-in softmax function.

Download the skeleton code `hw4.ipynb` from the course website. Complete the skeleton code. In order to get full marks, your report should include:

1. Your code.

2. Training loss curve.

3. Test error (overall).

4. Test error (each class).

Hint: You may find the official PyTorch tutorial helpful before implementing the network.

6 (Extra Credit - 20 points) Improve the performance on the CIFAR-10 dataset.

The problem setting is the same as Problem 5. You can use the same skeleton code `hw4.ipynb`. Try to modify the architecture of the previous one and improve the performance. You can also make changes other than the architecture such as loss function, data preprocessing, etc. Your report should include:

1. Your code.
2. Training loss curve.
3. Test error (overall).
4. Test error (each class).
5. Your main modification. You're encouraged to provide a detailed description and analysis.

The grading will be based on your performance, the description, and the analysis of the modification you made.