682: Midterm Exam

PUT YOUR NAME AND ID HERE:

1 Warm-up: Presidential Images

Donald Trump wants every image to look like him. He has decided to replace every piece of art in the White House (where the President of the United States lives) with a similar image in which his face appears throughout the image.

To do this, he first trains a classifier on 100 different classes, where one of the classes is "images of Donald Trump". For an image I that he wants to modify, he first puts it through the trained network and gets the probabilities of each category. Let p_T be the probability assigned to the Donald Trump class. He then uses backpropagation to obtain the derivative of p_T with respect to the image pixels:

$$\frac{\partial p_T}{\partial I}.$$

1. (3 points) What should he do with this derivative to modify the image?

2 Non-linearities

Huaizu is designing a deep network in which the only layers are linear layers of the form $Y = WX$, where W is a matrix of weights, X is the input to the layer and Y is the output of the layer. He says his network will be great because the code is so simple. Be builds a 10-layer network with only linear layers and trains it using the multi-class SVM loss on the final layer outputs (the scores). He finds it doesn't work that great for his image classification problem.

1. (3 points) Discuss a limitation of a network built out of only linear layers.

Consider a two class classification problem with classes A and B. Let the input to the network be just two values represented by the variables x_1 and x_2. Now suppose we train a network like the one above with only linear layers.

2. (3 points). Draw a scatter plot showing a distribution of points from class A (use small +'s) and class B (use small circles) that the network would not be able to separate using a network with only linear layers.

Vishna tells Huaizu he needs to use non-linearities in a neural network, and she chooses to use the sigmoid function. She builds a shallow, *two-layer* neural net using sigmoids in the first hidden layer. She finds that her training algorithm works, but it is quite slow. She thinks this has something to do with the fact that the sigmoid function outputs strictly positive values.

3. (3 points) Can you suggest a reason why Vishna's net might be training slowly? You can draw a picture to illustrate if you wish.

SouYoung suggests to Vishna that she should use the tanh non-linearity instead of a sigmoid non-linearity. This seems to work better for the two-layer network, but when she tries using a tanh on an 8-layer network, the training does not work very well. She finds that the weight derivatives after the second layer are very close to zero.

4. (3 points) Explain what might be causing this, and at least one potential solution.

3 Convolutional Neural Networks

Mbewe and Steve are having a discussion about the relationship between standard neural networks and CNNs. Steve is trying to understand the motivation for CNNs. Mbewe explains that, when learning about images, the following are often true:

- Having a unit that is connected *locally* to a small set of features in the previous layer is often effective, and that there may not be a need to have a unit connected to *all* of the features in the previous layer,
- A feature that is detected locally in one part of the image (say, the upper left corner), may also be useful in another part of the image (say, the lower right corner).

"Fine", says Steve, "But we don't need a CNN to implement these ideas." In fact, he argues,

- First, if it is beneficial for a network to have local connections, can't the network itself simply learn to have non-zero weights in a small region, with near-zero weights far from the center of that region?
- Second, if it is useful for the network to have the same features at multiple different regions, why can't a regular neural network simply learn to reproduce patterns at different points in the image?

Mbewe responds to Steve that while, "yes, these ideas are possible", in general it is better to implement these ideas using a CNN.

1. (8 points). Discuss why CNNs are generally a better way to solve these problems in the following aspects.

- Memory

- Parameter counts

- Training set size

- Training time

4 ReLU

1. (1 point) What does ReLU stand for?

2. (3 points) Write the mathematical definition of the ReLU as a function. Give the derivative of the ReLU with respect to its input x. You can do this by specifying the derivative in two different regions.

3. (3 points) What is a dead ReLU? Why does it happen?

4. (3 points) Describe the leaky ReLU. How many parameters does it have?

5. (3 points) Describe the max-out neuron. Name one advantage and one disadvantage over the ReLU.

5 Tips for training

1. (3 points) What is the problem, even in a very shallow network, with initializing weights to all zeros?

2. (3 points) In a network without batch normalization, what can happen if I initialize weights to random Gaussian values that are too small?

3. (3 points) Omar trains a network and gets a slightly higher accuracy on his validation data than on his training data. What might be going on, and what can he do about it?

6 Gradient Descent

A full *gradient descent* algorithm uses the entire training set to calculate the gradient of the total loss with respect to the parameters of the network. It then takes a step in the direction of the *negative* gradient, multiplied by a scalar called the learning rate.

Stochastic gradient descent (SGD) uses a subset of the training data (a mini-batch) to *estimate* the gradient of the total loss with respect to the network parameters. Of course we can choose the mini-batch size to be anywhere from a single training example to almost the entire training set.

1. (5 points) Discuss trade-offs to consider when choosing mini-batch size in SGD. Discuss as many factors as you can think of.

7 Semantic Segmentation

Jong-Chyi is interested in the image segmentation task, where the prediction should have the same size as the input and each pixel belongs to one class. Now suppose the dataset has 10 handwritten digits like the MNIST dataset, but the size of the input images are 9×9×1 (gray scale images). The goal is to segment the foreground and the background of the images. That is, for every pixel $x_{i,j}$ we have output label $y_{i,j} \in \{0, 1\}$ where 1 means the pixel is in the foreground and 0 means background.

1. (6 points) Jong-Chyi trained a small network for classification task on this dataset. The model is shown below. What?s the output dimension of layer {a,c,d,f,h,j} with one 9×9×1 image as the input? Fill in the table. Consider only one image and the first dimension of batch size can be ignored. The input dimension of `Conv2d` layer is $W \times H \times C$, and $F_{in} \times F_{out}$ for `Linear` layer.

a - `Conv2d(in_channels=1,out_channels=10,kernel_size=(3,3),padding=(1,1),stride=(1,1))`
b - `ReLU()`
c - `MaxPool2d(kernel_size=(3,3),padding=(0,0),stride=(3,3))`
d - `Conv2d(in_channels=10,out_channels=30,kernel_size=(3,3),padding=(1,1),stride=(1,1))`
e - `ReLU()`
f - `MaxPool2d(kernel_size=(3,3),padding=(0,0),stride=(1,1))`
g - `Flatten()` *This layer vectorizes the feature of each channel and squeezes the array.
h - `Linear(in_features=30,out_features=20)`
i - `ReLU()`
j - `Linear(in_features=20,out_features=10)`
k - `SoftMax()`

Input	9×9×1		
After layer a		After layer f	
After layer c		After layer h	
After layer d		After layer j	

2. (3 points) Modify this network for the segmentation task. Specify the layers you want to keep and add, and write down the model architecture in the same format. You can use the `Upsampling` layer with the argument either `output size` or `scale factor`. You can also use `ConvTranspose2d` layer for this problem.

3. (3 points) Do you think the weights from the pre-trained model of the classification task should be kept, and reused for the segmentation task? Why or why not?

4. (2 points) The Fully-Connected Network paper [Long et al. CVPR 2015] is also doing the segmentation task. There are several variants of the models in the paper, FCN-32s, FCN-16s, and FCN-8s. What's the differences between them? Why does FCN-8s give better results?

8 Back propagation

1. (3 points) Consider a convolutional layer in a neural network. The input to the convolutional layer is X (for simplicity, assume batch size is 1 and there is only 1 channel, so the shape of X is 16×16). The convolutional layer has a 3×3 window, with stride 1, and 1 output channel. The weight in the convolutional layer is W (with shape 3×3). Zero padding with size 1 is used in the convolutional layer, so the shape of Y is 16×16. Assume we have the gradients for Y as $dY(= \partial Loss/\partial Y)$, which has the same shape as Y. Elements in X are indexed as $X_{1-16,1-16}$ Calculate the gradient of $X_{1,2}$. Write it as a expression of dY and W.