Project 4 Questions

Instructions

- 7 questions (Q6 and Q7 with code components).
- *This will take two weeks! Parts are tricky—start early!*.
- Write code where appropriate; feel free to include images or equations.
- Please make this document anonymous.
- This assignment is **fixed length**, and the pages have been assigned for you in Gradescope. As a result, **please do NOT add any new pages**. We will provide ample room for you to answer the questions. If you *really* wish for more space, please add a page *at the end of the document*.
- **We do NOT expect you to fill up each page with your answer.** Some answers will only be a few sentences long, and that is okay.
- Question 6 has a coding component that must be submitted to Gradescope under a separate assignment (Project 4 Written Code (Numpy)) and is due a week after the rest of the written questions.

Questions

Q1: Many traditional computer vision algorithms use convolutional filters to extract feature representations, e.g., in SIFT, to which we then often apply machine learning classification techniques. Convolutional neural networks also use filters within a machine learning algorithm.

(a) What is different about the construction of the filters in each of these approaches?

(b) Please declare and explain the advantages and disadvantages of these two approaches.

Please answer on next page.

A1: Your answer here.

(a) In traditional computer vision algorithms, such as SIFT, the construction of filters are defined manually. For example, SIFT descriptor are constructed based upon the number in every orientation bins based upon the gradients of each pixel. However, the filters in convolutional neural networks are learned during the training process, we only need to define the shape and the number of the filters and the parameters in filters will be tuned during the optimization procedure.

(b) Filters in traditional computer vision algorithms:

- Advantages: it is fast as the feature extraction is predefined and required no training process. It is simpler design and less parameters to set compared to CNN.
- Disadvantages: it is poor generalization and not robust to nonlinear transformations.

Filters in convolutional neural network:

- Advantages: it can provide great performance in classification tasks and stongly bio-inspired and very good at generalization.
- Disadvantages: it requires large training set for better performance and demand high memory to store the trained weights of the network. Thus, it is slower than SIFT.

(source: Object Recognition SIFT vs Convolutional Neural Networks)

Q2: Many CNNs have a fully connected multi-layer perceptron (MLP) after the convolutional layers as a general purpose 'decision-making' subnetwork. What effects might a *locally-connected* MLP have on computer vision applications, and why?

Please give your answer in terms of the learned convolution feature maps, their connections, and the perceptrons in the MLP.

A2: The difference between the CNN and the *locally-connected* MLP is that CNNs are locally connected layers with shared weights while for the *locally-connected* MLP each pixel positions will have its own weights.Thus, the *locally-connected* MLP needs fewer overall filters that need to be trained as every pixel positions gets its own filter.

Source: Understanding Locally Connected Layers In Convolutional Neural Networks

Q3: Given a neural network and the stochastic gradient descent training approach for that classifier, discuss how the *learning rate*, *batch size*, and *training time* hyperparameters might affect the training process and outcome.

A3: Your answer here.

(a) *learning rate*: the learning rate controls how fast the model is adpated to the problem. Smaller learning rate requires more training epochs as the smaller changes are made to tune the weights of the model, whereas larger learning rates result in rapid change and require fewer epochs. For the outcome, if the learning rate is too large it can cause the model converge too quickly to a suboptimal solution or diverged, whereas a too small learning rate might cause the process to get stuck.

(b) *batch size*: the batch size impacts how quickly a model learns and the stability of the learning process. A smaller batch size are noisy, offering a regularizaing effect and lower generalization error while require less memory for training model. A larger batch size will lead to poor generalization but using a batch equal to the entire dataset guarantees convergence to the global optima of the objective function.

(How to control the stability of training neural networks with the batch size; Effect of batch size on training dynamics)

(c) *training time*: generally the training epochs will decide the training time of the model. If the number of training epochs is too small, the model might not reach the optima and give a bad performance. If the number of training epochs is too large, the model might become overfitting to the training data and start to lose performance in terms of generalization to test data.

Q4: What effects does adding a max pooling layer have for a single convolutional layer, where the output with max pooling is some size larger than 1 x 1 x d?

Notes: 'Global' here means whole image; 'local' means only in some image region.

LaTeX: To fill in boxes, replace '\square' with '\blacksquare' for your answer.

A4: Multiple choice. Choose all that apply.

Increases computational cost of training	☐
Decreases computational cost of training	■
Increases computational cost of testing	☐
Decreases computational cost of testing	■
Increases overfitting	☐
Decreases overfitting	■
Increases underfitting	■
Decreases underfitting	☐
Increases the nonlinearity of the decision function	☐
Decreases the nonlinearity of the decision function	■
Provides local rotational invariance	■
Provides global rotational invariance	☐
Provides local scale invariance	☐
Provides global scale invariance	☐
Provides local translational invariance	■
Provides global translational invariance	☐

Something to think about (ungraded): Given some input to a convolutional layer with stride 2×2 and kernel size 3×3, and ignoring the boundary, what is the minimum number of convolutional filters required to preserve all input information in the output feature map?

Multiple choice. *LaTeX:* Use command '\bullet' (•) to fill in the dots.

0.5	☐
1	•
2	☐
4	☐
It's impossible	☐

Here: Feel free to leave optional short written justification if desired.

Q5: In medical imaging, hand designing features with biological experts might lead to a generalizable and explainable method for physicians, regulators, or patients. However, deep learning methods can show higher test accuracy, even if their results may be less explainable. Please read this short article.

(a) In the article, physician Dr. Nicholas Price expresses the view that a lack of explainability should not hamper advances in health care, pointing out that Aspirin's mechanism was not understood for 70 years and that Lithium's mechanism to treat bipolar disorder is still unknown. Do you agree with Dr. Price? Would you feel comfortable allowing your doctor to rely on a 'black box' algorithm for medical diagnoses? (3-4 sentences)

(b) When deciding whether a medical algorithm is suitable for clinical use, what accuracy would you consider to be 'good enough'? In which different ways could we measure this? (3-4 sentences)

(c) In the event that a medical imaging algorithm makes an error and causes a patient harm, who should be held responsible? *Think about everyone involved: the algorithm engineer, the company selling the algorithm, the FDA approving it, the hospital or clinic purchasing it, the physician using it, etc.* (3-4 sentences)

(d) Nicholas Price is actually a legal scholar who specializes in health law at the University of Michigan. Does this information change your responses? (3-4 sentences)

A5: Your answer here.

(a) No, I don't agree with Dr. Price. Although the mechanisms of Aspirin and Lithium are not understood yet, they both show consistent positive results to patients, or at least no negative effects to the patients. However, for DL algorithms there still exists possibilities for the false predictions. And once it gives a false prediction, it will lead to some irreparable results.

I don't feel comfortable using a 'black box' algorithm for my medical diagnoses. I would not know the reason why it gives such diagnoses and I don't feel that I could trust it.

(b) To be 'good enough', the accuracy of the algorithm should reach at least 99.99%. That is 1 false prediction out of 10000 results. These is still really large if we take the number of patients into consideration. There are 7 billion people around the world and if they all use this algorithm to diagonize the disease, there will be 70,000 false predictions.

We could measure the percentage of false negatives as this is actually the worst case for the patients when the model gives predictions. And it would be easier to lower this percentage than the whole accuracy.

Extra page. Please remember to keep your page length the same.

(c) First of all, the physicians using the medical imaging algorithm will remain ultimately responsible for patient care. They need to acquire new skills to do their best for patients before they decide whether or not to adopt the outcome of the algorithms. Second, the company selling the algorithm and the hospital or clinic purchasing it share the same responsibilities as they are the ones that write the code and introduce them to physicians. Lastly, the FDA has to reexamine the safety of this algorithm and redesign their procedure for such AI algorithms approving.

(d) This information doesn't change my responses. Although Dr. Nicholas is knowledgable of the field of public health, his opionion can be affected by his personal emotions. There are so many cases where experts hold contradictory views to the same things and some of them should be wrong. Before the wide application of deep learning in health field, I would prefer it has gone through careful and meticulous testing.

Q6 background: Let us consider using a neural network (non-convolutional) to perform classification on the MNIST dataset of handwritten digits, with 10 classes covering the digits 0–9. Each image is 28×28 pixels, and so the network input is a 784-dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_i, \ldots, x_{784})$. The output of the neural network is probability distribution $\mathbf{p} = (p_1, \ldots, p_j, \ldots, p_{10})$ over the 10 classes. Suppose our network has one fully-connected layer with 10 neurons—one for each class. Each neuron has a weight for each input $\mathbf{w} = (w_1, \ldots, w_i, \ldots, w_{784})$, plus a bias b. As we only have one layer with no multi-layer composition, there's no need to use an activation function.

When we pass in a vector $\mathbf{x}$ to this layer, we will compute a 10-dimensional output vector $\mathbf{l} = (l_1, l_2, ..., l_j, ..., l_{10})$ as:

$$l_j = \mathbf{w}_j \cdot \mathbf{x} + b_j = \sum_{i=1}^{784} w_{ij} x_i + b_j \tag{1}$$

These distances from the hyperplane are sometimes called 'logits' (hence l) when they are the output of the last layer of a network. In our case, we only have *one* layer, so our single layer is the last layer.

Often we want to talk about the confidence of a classification. So, to turn our logits into a probability distribution $\mathbf{p}$ for our ten classes, we apply the *softmax* function:

$$p_j = \frac{e^{l_j}}{\sum_j e^{l_j}} \tag{2}$$

Each p_j will be positive, and $\sum_j p_j = 1$, and so softmax is guaranteed to output a probability distribution. Picking the most probable class provides our network's prediction.

If our weights and biases were trained, Eq. **??** would classify a new test example.

We have two probability distributions: the true distribution of answers from our training labels $\mathbf{y}$, and the predicted distribution produced by our current classifier $\mathbf{p}$. To train our network, our goal is to define a loss to reduce the distance between these distributions.

Let $y_j = 1$ if class j is the true label for x, and $y_j = 0$ if j is not the true label. Then, we define the *cross-entropy loss*:

$$L(w, b, x) = -\sum_{j=1}^{10} y_j \ln(p_j), \tag{3}$$

which, after substitution of Eqs. 2 and 1, lets us compute an error between the labeled ground truth distribution and our predicted distribution.

So...why does this loss L work? Using the cross-entropy loss exploits concepts from information theory—Aurélien Géron has produced a video with a succinct explanation of this loss. Briefly, the loss minimizes the difference in the amounts of information

needed to represent the two distributions. Other losses are also applicable, with their own interpretations, but these details are beyond the scope of this course.

Onto the training algorithm. The loss is computed once for every different training example. When every training example has been presented to the training process, we call this an *epoch*. Typically we will train for many epochs until our loss over all training examples is minimized.

Neural networks are usually optimized using gradient descent. For each training example in each epoch, we compute gradients via backpropagation (an application of the chain rule in differentiation) to update the classifier parameters via a learning rate λ:

$$w_{ij} = w_{ij} - \lambda \frac{\partial L}{\partial w_{ij}}, \tag{4}$$

$$b_j = b_j - \lambda \frac{\partial L}{\partial b_j}. \tag{5}$$

We must deduce $\frac{\partial L}{\partial w_{ij}}$ and $\frac{\partial L}{\partial b_j}$ as expressions in terms of x_i and p_j.

Intuition: Let's just consider the weights. Recall that our network has one layer of neurons followed by a softmax function. To compute the change in the cross-entropy loss with respect to neuron weights $\frac{\partial L}{\partial w_{ij}}$, we will need to compute and chain together three different terms:

1. the change in the loss with respect to the softmax output $\frac{\delta L}{\delta p_j}$,
2. the change in the softmax output with respect to the neuron output $\frac{\delta p_j}{\delta l_j}$, and
3. the change in the neuron output with respect to the neuron weights $\frac{\delta l_j}{w_{ij}}$.

We must derive each individually, and then formulate the final term via the chain rule. The biases follow in a similar fashion.

The derivation is beyond the scope of this class, and so we provide them here:

$$\frac{\delta L}{\delta w_{ij}} = \frac{\delta L}{\delta p_a} \frac{\delta p_a}{\delta l_j} \frac{\delta l_j}{w_{ij}} = \begin{cases} x_i(p_j - 1), a = j \\ x_i p_j, a \neq j \end{cases} \tag{6}$$

$$\frac{\delta L}{\delta b_j} = \frac{\delta L}{\delta p_a} \frac{\delta p_a}{\delta l_j} \frac{\delta l_j}{b_j} = \begin{cases} (p_j - 1), a = j \\ p_j, a \neq j \end{cases} \tag{7}$$

Here, a is the predicted class label and j is the true class label. An alternative form you might see shows $\frac{\delta L}{\delta w_{ij}} = x_i(p_j - y_j)$ and $\frac{\delta L}{\delta b_j} = p_j - y_j$ where $y_j = 1$ if class j is the true label for x, and $y_j = 0$ if j is not the true label.

So...after all of that, our gradient update rules are surprisingly simple!

Further details: For interested students, we refer students to Chapter 1 of Prof. Charniak's deep learning notes, which derives these gradient update rules. We also refer to Prof. Sadowski's notes on backpropagation: Section 1 derives terms first for cross-entropy loss with logistic (sigmoid) activation, and then Section 2 derives terms for cross-entropy loss with softmax. The Wikipedia article on the backpropagation algorithm likewise represents a derivation walkthrough of the general case with many hidden layers each with sigmoid activation functions, and a final layer with a softmax function.

Q6: We will implement these steps in code using numpy. We provide a code stencil `main.py` which loads one of two datasets: MINST and the scene recognition dataset from Project 3. We also provide two models: a neural network, and then a neural network whose logits are used as input to an SVM classifier. Please look at the comments in `main.py` for the arguments to pass in to the program for each condition. The neural network model is defined in `model.py`, and the parts we must implement are marked with TODO comments.

Tasks: Please follow the steps to implement the forward model evaluation and backward gradient update steps. Then, run your model on all four conditions and report training loss and accuracy as the number of training epochs increases.

Please upload your completed `model.py` to Gradescope under the assignment titled "Project 4 Written Code (Numpy)". The autograder will check that you correctly implemented parts of your model. Please also include a `writeup.pdf` with your responses to the following questions.

1. What do these numbers tell us about the capacity of the network, the complexity of the two problems, the value of training, and the value of the two different classification approaches?
2. How well did each model perform on each dataset? Please use this table to structure your response.
    - NN on MNIST: xx% (highest accuracy)
        - Epoch 0 loss: xx Accuracy: xx%
        - Epoch 9 loss: xx Accuracy: xx%
    - NN+SVM on MNIST: xx% (highest accuracy)
        - Epoch 0 loss: xx Accuracy: xx%
        - Epoch 9 loss: xx Accuracy: xx%
    - NN on SceneRec: xx% (highest accuracy)
        - Epoch 0 loss: xx Accuracy: xx%
        - Epoch 9 loss: xx Accuracy: xx%
    - NN+SVM on SceneRec: xx% (highest accuracy)
        - Epoch 0 loss: xx Accuracy: xx%
        - Epoch 9 loss: xx Accuracy: xx% s

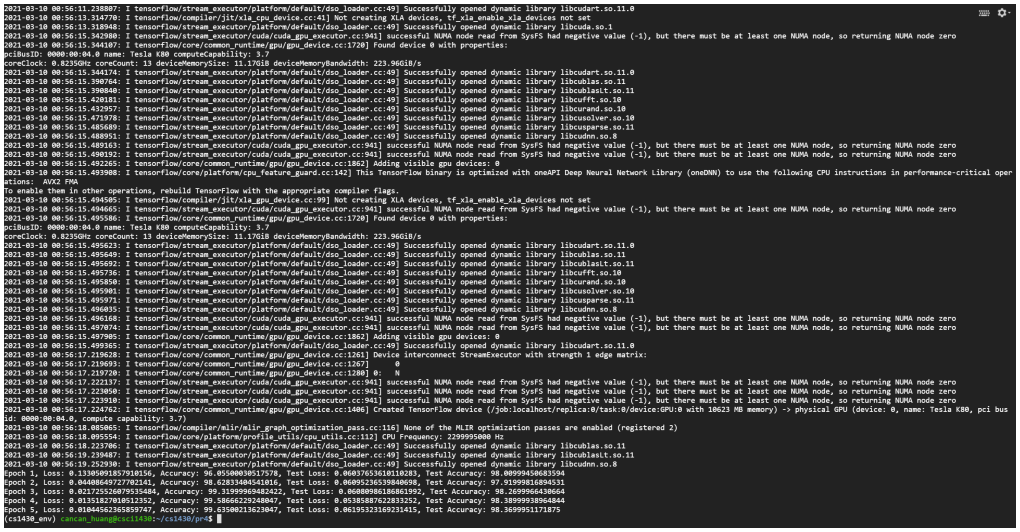
Q7: Follow the GCP guide to set up GCP. *Note:* This is tricky, and it will take you some time to become familiar with the system! Once finished, complete one of these two Tensorflow MNIST tutorials on GCP:

- TensorFlow 2 quickstart for beginners
- TensorFlow 2 quickstart for experts

The work for the tutorial can be done in a completely new Python file inside or outside of your project code (you won't have to turn it in).

A7: Please insert a screenshot of the tutorial code output, running on GCP:

I choose to finish the second one TensorFlow 2 quickstart for experts. From the screenshot, we can check that GPU is K80 and accuracy results meets our expectation.



Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!