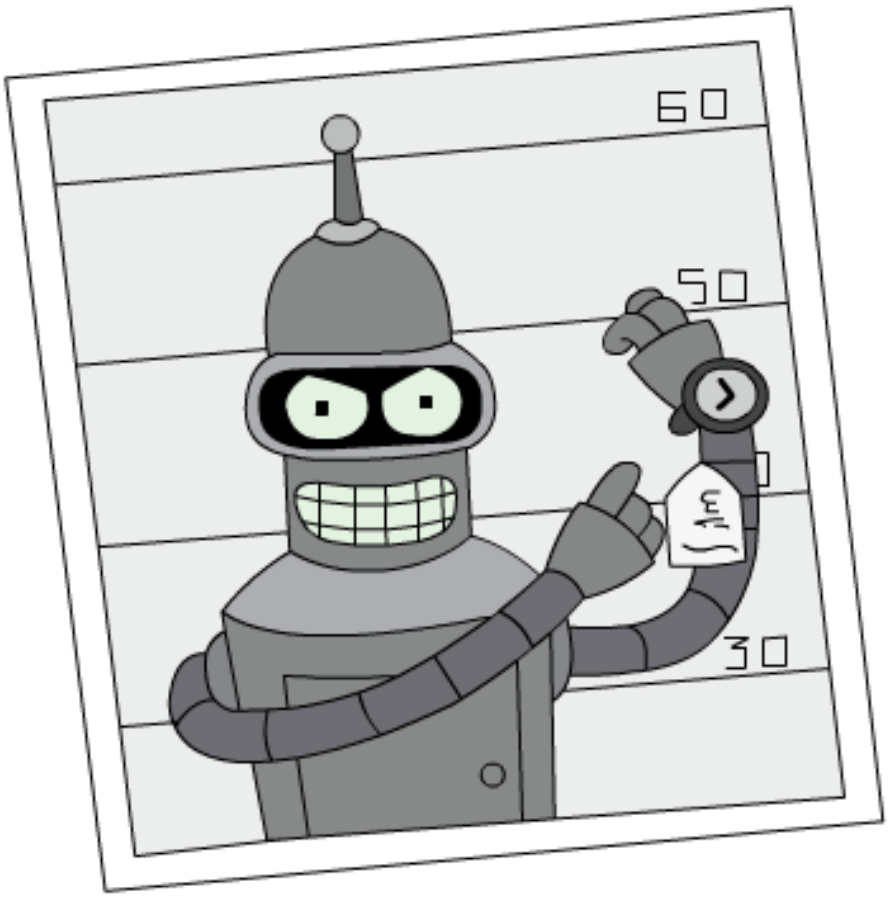
ECE4179/5179 - Deep Learning and Neural Networks

Assignment 2

Due: 6/9/2021, 6:00pm



General Comments.

- Please submit your report along with the code (Jupyter notebook is preferred) as a single zip file.
- Include your name and student number in the filename for both the zip file and PDF. Do not send doc/docx files.
- Include your name and email address on the report and use a single column format when you prepare your report.
- Please ensure all of your results are included in your report. Marking is based on what is in the report. This includes plots, tables, code screen-shots etc.
- Make sure you answer questions in full and support any discussions with relevant additional information/experiments if necessary.

Late submission. Late submission of the assignment will incur a penalty of 10% for each day late. That is with one day delay, the maximum mark you can get from the assignment is 90 out of 100, so if you score 99, we will (sadly) give you 90. Assignments submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness).

Note from ECE4179/5179 Team. The nature of assignments in ECE4179/5179 is different from many other courses. Here, we may not have a single solution to a problem. Be creative in your work and feel free to explore beyond questions. Creativity will be awarded by bonus points.

Good Luck

Question:	1	2	3	4	Total
Points:	35	15	35	15	100
Score:					

1. Using "hw2_Q1.ipynb" as starter code, implement the following network:

$$\mathbb{R}^{4096} \ni \boldsymbol{x} \rightarrow \text{fc1} : \text{Linear}(4096 \times n) \rightarrow \text{ReLU} \rightarrow \text{fc2} : \text{Linear}(n \times 4096) = \hat{\boldsymbol{x}} \, .$$

That is, the input $\boldsymbol{x}$ which is a 4096D vector is first mapped to an n dimensional vector using the fc1 layer (which is a linear layer). The resulting vector goes through the ReLU nonlinearity followed by another linear layer (fc2) to create a 4096 dimensional vector (which is called $\hat{\boldsymbol{x}}$). For the sample $\boldsymbol{x}$, consider the loss of the network to be $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2$ (check MSELoss in the PyTorch documents).

1.1. [15 points] Train the network described above for $n = 16$ and 64. Let $W \in \mathbb{R}^{4096 \times n}$ be the weight matrix of fc1. For each network, save the **weights** of fc1 by reshaping *every column* to a 64×64 image during the course of training. For example and when $n = 16$, this should result in 16 images. If you need 100 epochs to fully train your network, store the weights as images at regular intervals (say every 10 epochs) starting from the **random initialization** (epoch 0) till the final epoch. In your report, discuss the following points;

- state how you identified the value of the learning rate for the experiment.
- plot the final output for some samples for each network. Note that the output $\hat{\boldsymbol{x}}$ can be reshaped to an image of size 64×64.
- plot the stored weights as images (*e.g.*, every 10 epochs). Is this what you expected?

1.2. [10 points] Read the Wiki article on Eigenfaces at https://en.wikipedia.org/wiki/Eigenface. Use the SVD method described under **Computing the eigenvectors** to obtain n Eigenfaces from your data (remember each column of matrix $\boldsymbol{X}$ should be a face here). Discuss, if any, links between the Eigenface method and your network.

1.3. [10 points] The network you have implemented in the above parts is a simple form of an AutoEncoders (AEs). Do you think such a neural network can be used to denoise/recover contaminated face images? What will happen if you randomly zero out some pixels and then feed the image to your network? Empirically justify your answer.

Hint. You may want to use ADAM optimizer instead of SGD (see PyTorch documents). If you opt to use ADAM optimizer, remember that usually the learning rate for the ADAM is smaller than that of SGD.

2. **Shallow MLP.** Use "hw2_Q2_and_Q3.ipynb" as the starter code for this question. We are interested in designing an image classifier for Q2. The data file "hw2_Q2_and_Q3_data.npz" contains images and labels for training, validation and test purposes. In all the parts below, you should only use the **training** images and their labels to train your model. You may use the **validation** set to pick a trained model. For example, during training, you can test the accuracy of your model using the validation set every epoch and pick the model that achieves the highest validation accuracy. You should then report your results on the **test** set once you choose your model. Train a shallow MLP, consisting of just one hidden layer to classify the images according to the following design;

$$\mathbb{R}^{784} \ni \boldsymbol{x} \rightarrow \text{fc1} : \text{Linear}(784 \times n) \rightarrow \text{ReLU} \rightarrow \text{fc2} : \text{Linear}(n \times 10) = \hat{\boldsymbol{y}} \ ,$$

where $n = 32$. In your report, study the following factors,

2.1. [10 points] Train your network exclusively with SGD optimizer and make sure that the network has completely converged (*i.e.*, till the training loss flattens out). Discuss the followings in your report;

- Plot the training loss, validation loss, and validation accuracy per epoch.
- As a data scientist, after training, you need to pick a model for deployment. To identify the best model after training, you can choose a model that achieves the lowest training loss, the lowest validation loss, or the highest validation accuracy. Use the test set and evaluate the trained model for each of the aforementioned choices. Comment on how you would pick a model in general based on your observations in this experiment.

2.2. [5 points] **Learning Rate.** When using SGD as an optimizer, it can be beneficial to reduce the learning rate after a few epochs. For the network with 32 hidden units from the previous part, complete the following study. First, use a fixed learning rate throughout training. Then by studying the behaviour of the loss curve, try to identify when loss stops changing noticeably (hence network stops learning) and decrease the learning rate accordingly. For example, if the loss becomes steady after epoch 55, decrease the learning rate by a factor of 10 from that epoch and observe the behaviour of the network. In your report, plot the training and validation accuracy and loss and discuss the impact of decreasing the learning rate during training in your report.

3. **Deep MLP.** Use "hw2_Q2_and_Q3.ipynb" as the starter code again. In this question, we are interested in designing deep MLPs with three hidden layers to classify images.

3.1. [15 points] Design and train the following deep MLP.

$$\mathbb{R}^{784} \ni \boldsymbol{x} \to \text{fc1} : \text{Linear}(784 \times 32) \to \text{ReLU} \to \text{fc2} : \text{Linear}(32 \times 64)$$
$$\hat{\boldsymbol{y}} \leftarrow \text{fc4} : \text{Linear}(32 \times 10) \leftarrow \text{ReLU} \leftarrow \text{fc3} : \text{Linear}(64 \times 32) \leftarrow \text{ReLU}$$

In your report, plot the training and validation accuracies and losses per epoch. Compare the performance of the deep MLP with the shallow MLP you trained in question 2.

3.2. [10 points] For this part, you train a wider deep MLP. Design and train the following deep MLP.

$$\mathbb{R}^{784} \ni \boldsymbol{x} \to \text{fc1} : \text{Linear}(784 \times 128) \to \text{ReLU} \to \text{fc2} : \text{Linear}(128 \times 64)$$
$$\hat{\boldsymbol{y}} \leftarrow \text{fc4} : \text{Linear}(128 \times 10) \leftarrow \text{ReLU} \leftarrow \text{fc3} : \text{Linear}(64 \times 128) \leftarrow \text{ReLU}$$

In your report, plot the training and validation accuracies and losses per epoch. Compare the performance of the wide deep MLP with the one you trained in the part 3.1.

3.3. [5 points] Calculate the number of parameters for the MLPs you trained in part 3.1 and part 3.2.

3.4. [5 points] AI can benefit society in many ways but, given the energy needed to support the computing behind AI, these benefits can come at a high environmental price. Packages such as CarbonTracker can be used as tools for tracking and predicting the energy consumption and carbon footprint of training deep learning models. Measure the carbon footprint of question 3.2.

Hint. If you do not have access to an Nvidia GPU, use Google Colab for this purpose.

4. **When Training data can mislead you.** Use "hw2_Q4.ipynb" as a starter code for this question. We are interested in designing an image classifier for Q3 as well. The data file "hw2_Q4_data.npz" contains images and labels for training and test purposes. You should only use the training images and their labels to train your model. You do not have a validation set, so you pick your model once training is completed.

   4.1. [15 points] Based on your expertise from Q2, design a deep model with at least two hidden layers and use ADAM optimizer to fully train your model. During the training, plot the test accuracy after each epoch. If you observe a strange behaviour, discuss what might be the source of that.