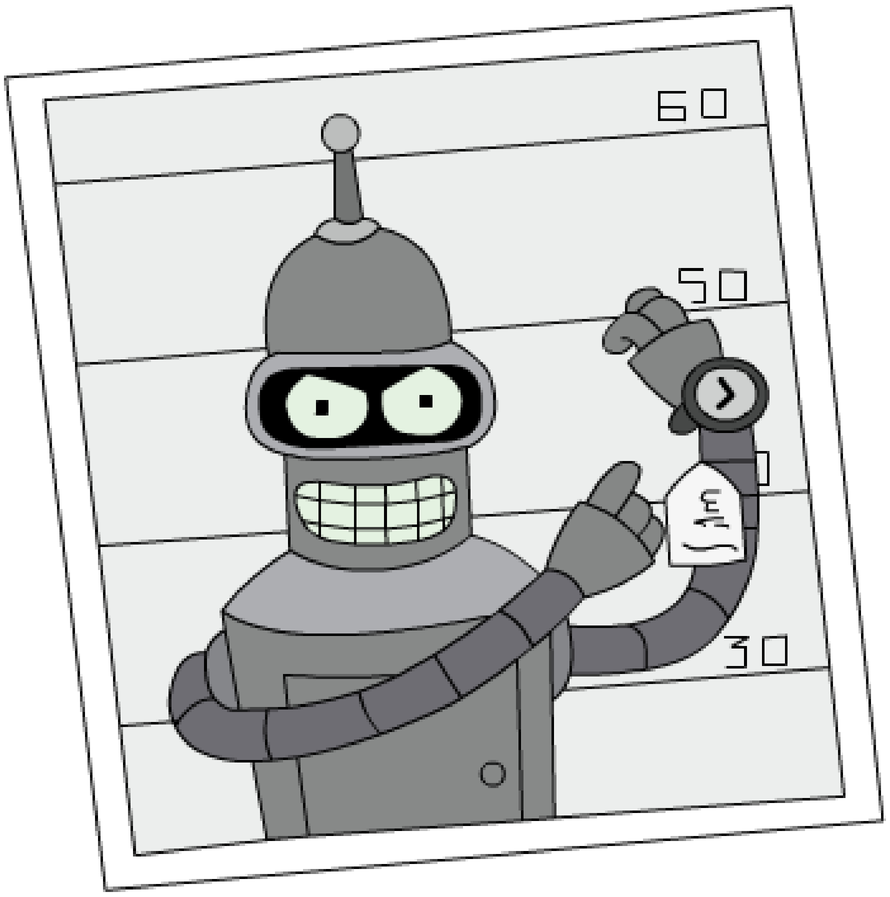
ECE4179/5179 - Deep Learning and Neural Networks

Assignment 3

Due: 26/9/2021, 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	Total
Points:	40	45	15	100
Score:				

Data. In this assignment, we work with CNNs. The dataset we use is STL10. The dataset represents 10 object classes, namely "airplane", "bird", "car", "cat", "deer", "dog", "horse", "monkey, "ship", and "truck". Download the data for this assignment from this link (file is 350MB). Our dataset has 13,000 color images of size 96×96. In this assignment, we use 8,000, 2,000, and 3,000 images for training, validation and testing, respectively. You will use the training images to train your model. You can use validation set to tune the hyper-parameters of your model (*e.g.*, learning rate) or pick a model (*e.g.*, after training for 100 epochs, the model that has the best accuracy on both training and validation images is chosen as the best model). You will use testing images to report the accuracy of your trained model once you have picked a model according to the above.

1. **Shallow CNN.** For this question, you design a shallow CNN and use it to classify STL images. Use "hw3.ipynb" as a starter code and implement the following CNN.

   image → **conv1** → ReLU → **conv2** → ReLU → **conv3** → ReLU → **maxpool** → **fc1** → ReLU → **fc2**

   Table 1 below provides details of each layer. Please note that you need to flatten the output of the maxpooling layer (using the view function in PyTorch) to connect the fc layers. Also, note that the output of fc2 is fed to a softmax layer but since in PyTorch, the CrossEntropyLoss has an inbuilt softmax function, the softmax layer is not shown here.

Table 1: Q1 - network structure

Name	Type	in	out	kernel size	padding	stride
conv1	Conv2D	3	96	7×7	0	2
conv2	Conv2D	96	64	5×5	0	2
conv3	Conv2D	64	128	3×3	0	2
fc1	Linear	1152	128	NA	NA	NA
fc2	Linear	128	10	NA	NA	NA
maxpool	Pooling	NA	NA	3×3	0	3

1.1. [20 points] Train the network described above. In your report, plot the training and validation loss (per epoch). Also, plot training, validation and test accuracy per epoch. Detail out the value of hyperparameters, the optimizer used and all other relevant information in your report.

1.2. [5 points] Once training is done, pass all your validation images through the network and plot the top five images correctly classified per class. That is, for each class, pick five images that are correctly classified by your network and have the maximum softmax scores.

1.3. [5 points] Repeat the above but this time plot the top five images that are misclassified for each class (*i.e.*, your network is very confident about its decision but the decision is totally wrong).

1.4. [10 points] A confusion matrix is a table used to describe the performance of a classifier. It allows easy identification of class confusions (*e.g.*, one class might be mislabeled as the other more often). Read more about the confusion matrix from the encyclopedia of machine learning (see the pdf file Ting2010_ConfusionMatrix.pdf). Compute the confusion matrix of your training, validation, and test data. Are they following a similar pattern?

2. **Deep CNN.** Use "hw3.ipynb" as a starter code for this question and develop a deep CNN to classify STL images. Your network has 4 convolutional blocks. We denote a convolutional block by *conv-blk* hereafter. The structure of a conv-blk is as follows;

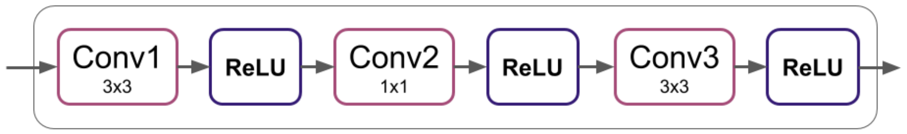


Figure 1: Structure of the *conv-blk*.

The details of layers inside the conv-blk are depicted in Table 2. In essence, a convolutional block receives an input $\boldsymbol{x}$ of size $c_i \times H_i \times W_i$ and processes it with 3 convolutional layers followed by ReLU non-linearity. The first convolutional layer has c_o filters of size 3×3. With an stride of two and padding of one, the first convolutional layer creates a feature map of size $c_o \times H_i/2 \times W_i/2$. This is further processed by 1×1 and 3×3 convolutions (and non-linearity).

Table 2: Q2 - Details of the convolutional block

Name	Type	in	out	kernel size	padding	stride
Conv1	Conv2D	c_i	c_o	3×3	1	2
Conv2	Conv2D	c_o	c_o	1×1	0	1
Conv3	Conv2D	c_o	c_o	3×3	1	1

Our deep CNN uses a stack of four of the aforementioned blocks. This will create a feature map of spatial resolution 6×6 ($96 \xrightarrow{blk1} 48 \xrightarrow{blk2} 24 \xrightarrow{blk3} 12 \xrightarrow{blk4} 6$). The network then uses a Global Average Pooling (GAP) layer followed by a linear layer. Put all together, the structure of the network reads as:

image → **conv-blk1** → **conv-blk2** → **conv-blk3** → **conv-blk4** → **GAP** → **fc**

The details of the conv-blks are depicted in Table 3.

Table 3: Q2 - Block structure

Name	in	out
conv-blk1	3	32
conv-blk2	32	64
conv-blk3	64	128
conv-blk3	128	192
fc1	192	10

2.1. [25 points] Implement and train the network described above. In your report, plot the training and validation loss (per epoch). Also, plot training, validation and test accuracy per epoch. Detail out the value of hyperparameters, the optimizer used and all other relevant information in your report.

2.2. [10 points] Use data augmentation and normalization techniques to improve the performance of your network. In particular, check "torchvision.transforms". Commonly used augmentation techniques include random flip and random crop, while channel normalization (*e.g.*, RGB) can be beneficial.

2.3. [10 points] You are allowed to add or decrease the number of blocks or their structures. Can you design a better network as compared to the original structure suggested in question 2? Detail out your design. A better network may have **1.** less parameters, **2.** faster convergence behaviour, **3.** better accuracy or all of them. It would be useful to pay attention to the confusion matrices over the validation set, as it may suggest models that can outperform others for some particular classes.

3. [15 points] **Occlusion Sensitivity.** A way of visualizing a CNN is called "Occlusion Sensitivity". Basically, one hides a rectangular patch of the input of a CNN and measures the softmax score for occluded input. If the patch occluds essential information about the input, the softmax score will drop significantly. By sliding the patch over the input, one can identify and visualize which parts of the image are more important for classification. To give an example, if your CNN classifies dogs, you expect by applying the occlusion sensitivity, softmax scores of the class dog drop when the occlusion slides over the face of the dog. This shows that your CNN is not picking up clues from the background to recognize dogs. For more details, check part 4.2 of the paper "Visualizing and Understanding Convolutional Networks" by Zeiler and Fergus. Implement the occlusion sensitivity and show the sensitivity heatmap for the images from parts