Intro to Image Understanding (CSC420)
Assignment 2

Due Date: October 21st, 2022, 10:59 pm
Total: 200 marks

General Instructions:

- You are allowed to work directly with one other person to discuss the questions. However, you are still expected to write the solutions/code/report in your own words; i.e. no copying. If you choose to work with someone else, you must indicate this in your assignment submission. For example, on the first line of your report file (after your own name and information, and before starting your answer to Q1), you should have a sentence that says: *"In solving the questions in this assignment, I worked together with my classmate [name & student number]. I confirm that I have written the solutions/-code/report in my own words"*.
- Your submission should be in the form of an electronic report (PDF), with the answers to the specific questions (each question separately), and a presentation and discussion of your results. For this, please submit a file named **report.pdf** to MarkUs directly.
- Submit documented codes that you have written to generate your results separately. Please store all of those files in a folder called **assignment2**, zip the folder and then submit the file **assignment2.zip** to MarkUs. You should include a **README.txt** file (inside the folder) which details how to run the submitted codes.
- Do not worry if you realize you made a mistake after submitting your zip file; you can submit multiple times on MarkUs.

Part I: Theoretical Problems (80 marks)

[Question 1] Image Pyramids (10 marks)

In Gaussian pyramids, the image at each level I_k is constructed by blurring the image at the previous level I_{k-1} and downsampling it by a factor of 2. A Laplacian pyramid, on the other hand, consists of the difference between the image at each level (I_k) and the upsampled version of the image in the next level of the Gaussian pyramid (I_{k+1}).

Given an image of size $2^n \times 2^n$ denoted by I_0, and its Laplacian pyramid representation denoted by $L_0, ..., L_{n-1}$, show how we can reconstruct the original image, using the minimum information from the Gaussian pyramid. Specify the minimum information required from the Gaussian pyramid and a closed-form expression for reconstructing I_0.

Hint: The reconstruction follows a recursive process; What is the base case that contains the minimum information?

[Question 2] Activation Functions (10 marks)

Show that in a fully connected neural network with linear activation functions, the number of layers has effectively no impact on the network.

Hint: Express the output of a network as a function of its inputs and its weights of layers.

[Question 3] Back Propagation (10 marks)

Consider a neural network that represents the following function:

$$\hat{y} = \sigma(w_5\sigma(w_1x_1 + w_2x_2) + w_6\sigma(w_3x_3 + w_4x_4))$$

where x_i denotes input variables, $\hat{y}$ is the output variable, and σ is the logistic function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Suppose the loss function used for training this neural network is the **L2** loss, i.e. $L(y, \hat{y}) = (y - \hat{y})^2$. Assume that the network has its weights set as:

$$(w_1, w_2, w_3, w_4, w_5, w_6) = (-0.65,\ -0.55,\ 1.74,\ 0.79,\ -0.13,\ 0.93)$$

[3.a] (5 marks) Draw the computational graph for this function. Define appropriate intermediate variables on the computational graph.
[3.b] (5 marks) Given an input data point $(x_1, x_2, x_3, x_4) = (1.2,\ -1.1,\ 0.8,\ 0.7)$ with true label of 1.0, compute the partial derivative $\frac{\partial L}{w_3}$, by using the back-propagation algorithm. Indicate the partial derivatives of your intermediate variables on the computational graph. Round all your calculations to 4 decimal places.

Hint: For any vector (or scalar) $\mathbf{x}$, we have $\frac{\partial}{\partial \mathbf{x}}(||\mathbf{x}||_2^2) = 2\mathbf{x}$. Also, you do not need to write any code for this question! You can do it by hand.

[Question 4] Convolutional Neural Network (15 marks)

In this problem, our goal is to estimate the computation overhead of CNNs by counting the FLOPs (floating point operations). Consider a convolutional layer C followed by a max pooling layer P. The input of layer C has 50 channels, each of which is of size 12×12. Layer C has 20 filters, each of which is of size 4×4. The convolution padding is 1 and the stride is 2. Layer P performs max pooling over each of the C's output feature maps, with 3×3 local receptive fields, and stride 1.
Given scalar inputs x_1, x_2, ..., x_n, we assume:

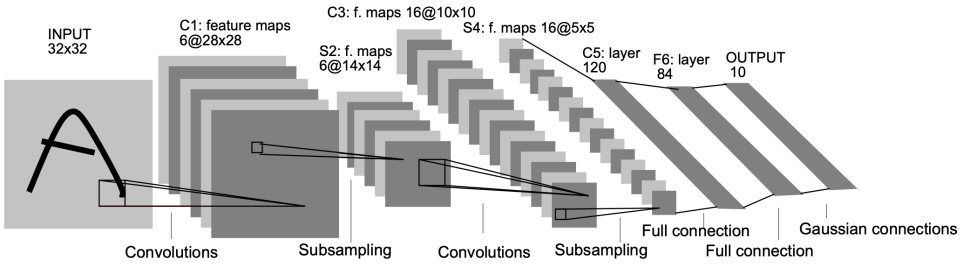
- A scalar multiplication $x_i.x_j$ accounts for one FLOP.
- A scalar addition $x_i + x_j$ accounts for one FLOP.

- A max operation $\max(x_1, x_2, ..., x_n)$ accounts for $n - 1$ FLOPs.
- All other operations do not account for FLOPs.

How many FLOPs layer C and P conduct in total during one forward pass, with and without accounting for bias?

[Question 5] Trainable Parameters (10 marks)

The following CNN architecture is one of the most influential architectures that was presented in the 90s. Count the total number of trainable parameters in this network. Note that the Gaussian connections in the output layer can be treated as a fully connected layer similar to $F6$.



[Question 6] Logistic Activation Function (10 marks)

For backpropagation in a neural network with logistic activation function, show that, in order to compute the gradients, as long as we have the outputs of the neurons, there is no need for the inputs.

Hint: Find the derivative of a neuron's output with respect to its inputs.

[Question 7] Hyperbolic Tangent Activation Function (15 marks)

One alternative to the logistic activation function is the hyperbolic tangent function:

$$\mathbf{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}.$$

- (a) What is the output range for this function, and how it differs from the output range of the logistic function?
- (b) Show that its gradient can be formulated as a function of logistic function.
- (c) When do we want to use each of these activation functions?

Part II: Implementation Tasks (120 marks)

In this question, we train (or fine-tune) a few different neural network models to classify dog breeds. We also investigate their dataset bias and cross-dataset performances. All the tasks should be implemented using Python with a deep learning package of your choice, e.g. PyTorch or TensorFlow.

We use two datasets in this assignment.

1. Stanford Dogs Dataset
2. Dog Breed Images

The Stanford Dogs Dataset (SDD) contains over 20,000 images of 120 different dog breeds. The annotations available for this dataset include class labels (i.e. dog breed name) and bounding boxes. In this assignment, we'll only be using the class labels. Further, we will only use a small portion of the dataset (as described below) so you can train your models on Colab. Dog Breed Images (DBI) is a smaller dataset containing images of 10 different dog breeds.

To prepare the data for the implementation tasks, follow these steps:

1- Download both datasets and unzip them. There are 7 dog breeds that appear in both datasets:

- Bernese mountain dog
- Border collie
- Chihuahua
- Golden retriever
- Labrador retriever
- Pug
- Siberian husky

2- Delete the folders associated with the remaining dog breeds in both datasets. You can also delete the folders associated with the bounding boxes in the SDD.

3- For the 7 breeds that are present in both datasets, the names might be written slightly differently (e.g. Labrador Retriever vs. Labrador). Manually rename the folders so the names match (e.g. make them both *labrador_retriever*).

4- Rename the folders to indicate that they are subsets of the original datasets (to avoid potential confusion if you later want to use them for another project). For example, *SDDsubset* and *DBIsubset*. Each of these should now contain 7 subfolders (e.g. *border_collie*, *pug*, etc.) and the names should match.

5- Zip the two folders (e.g. *SDDsubset.zip* and *DBIsubset.zip*) and upload them to your Google Drive.

You can find sample code working with the SDD on the internet. If you want, you are welcome to look at these examples (particularly the one linked here) and use them as your starting code or use code snippets from them. You will need to modify the code as our questions are asking you to do different tasks, which are not the same as the ones in these online examples. But using and copying code snippets from these resources is fine. If you choose to use this (or any other online example) as your starting code, please acknowledge them in your submission. We also suggest that before starting to modify the starting code, you run them as is on your data (e.g. DBIsubset) to 1) make sure your dataset setup is correct and 2) to make sure you fully understand the starter code before you start modifying it.

Task I - Inspection (10 marks):
Look at the images in both datasets, and briefly explain if you observe any systematic differences between images in one dataset vs. the other.

Task II - simple CNN Training on the DBI (25 marks):
Construct a simple convolutional neural network (CNN) for classifying the images in SDD. For example, you can construct a network as follow:

- convolutional layer - 16 filters of size 3×3
- batch normalization
- convolutional layer - 16 filters of size 3×3
- max pooling (2×2)
- convolutional layer - 8 filters of size 3×3
- batch normalization
- convolutional layer - 8 filters of size 3×3
- max pooling (2×2)
- dropout (e.g. 0.5)
- fully connected (32)
- dropout (0.5)
- softmax

If you want, you can change these specifications; but if you do so, please specify them in your submission. Use RELU as your activation function, and cross-entropy as your cost function. Train the model with the optimizer of your choice, *e.g.*, SGD, Adam, RMSProp, etc. Use random cropping, random horizontal flipping, and random rotations for augmentation. Make sure to tune the parameters of your optimizer for getting the best performance on the validation set.

Plot the training, and test accuracy over the first 10 epochs. Note that the accuracy is

different from the loss function; the accuracy is defined as the percentage of images classified correctly.

Train the same CNN model again; this time, with dropout. Plot the training and test accuracy over the first 10 epochs; and compare them with the model trained without dropout. Report the impact of dropout on the training and its generalization to the test set.

Task III - ResNet Training on the DBI (25 marks):

[III.a] (15 marks) ResNet models were proposed in the "Deep Residual Learning for Image Recognition" paper. These models have had great success in image recognition on benchmark datasets. In this task, we use the ResNet-18 model for the classification of the images in the DBI dataset. To do so, use the ResNet-18 model from PyTorch, modify the input/output layers to match your dataset, and train the model from scratch; *i.e.*, do not use the pre-trained ResNet. Plot the training, validation, and testing accuracy, and compare those with the results of your CNN model.

[III.b] (10 marks) Run the trained model on the entire SDD dataset and report the accuracy. Compare the accuracy obtained on the (test set of) DBI, vs. the accuracy obtained on the SDD. Which is higher? Why do you think that might be? Explain very briefly, in one or two sentences.

Task IV - Fine-tuning on the DBI (30 marks):
Similar to the previous task, use the following three models from PyTorch: ResNet18, ResNet34, and ResNeXt32. This time you are supposed to use the pre-trained models and fine-tune the input/output layers on DBI training data. Report the accuracy of these fine-tuned models on DBI test dataset, and also the entire SDD dataset. Discuss the cross-performance of these trained models. For example, are there cases in which two different models perform equally well on the test portion of the DBI but have significant performance differences when evaluated on the SDD?

Task IV - Dataset detection (30 marks):
Train a model that – instead of classifying dog breeds – can distinguish whether a given image is more likely to belong to SDD or DBI. To do so, first, you need to divide your data into training and test data (and possibly validation if you need those for tuning the hyperparameters of your model). You need to either reorganize the datasets (to load the images using *torchvision.datasets.ImageFolder*) or write your own data loader function. Train your model on the training portion of the dataset. Include your network model specifications in the report, and make sure to include your justifications for that choice. Report your model's accuracy on the test portion of the dataset.