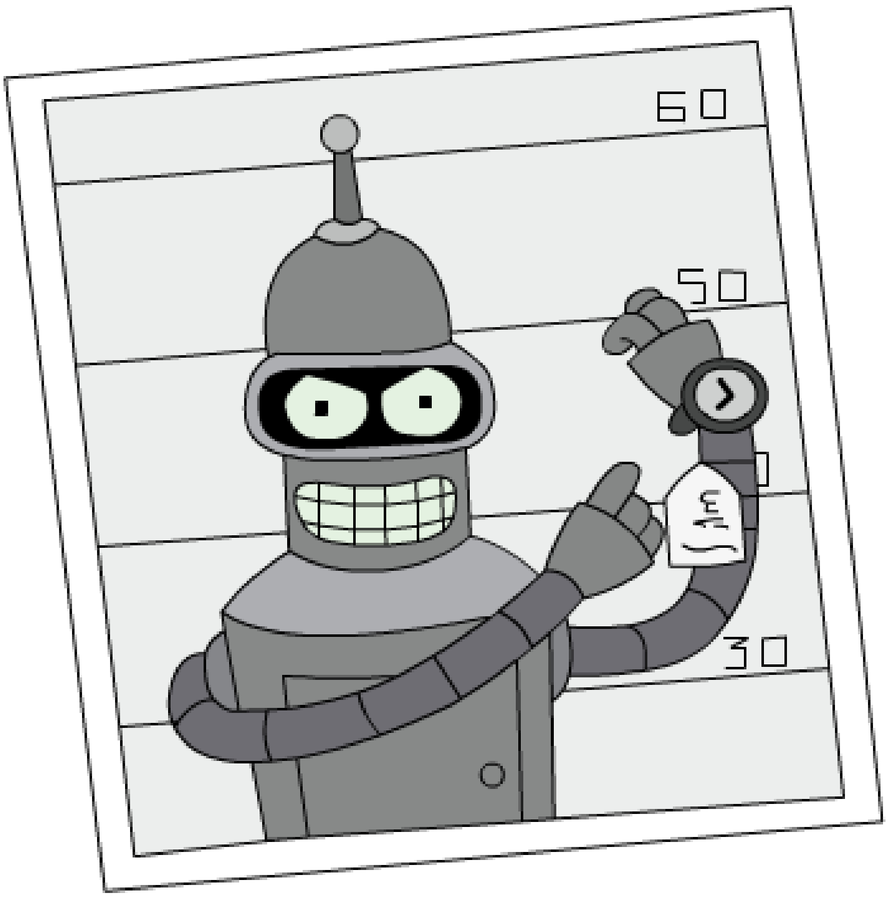
Convolutional Neural Network (CNN)

ECE4179/5179/6179: Neural networks and deep learning
Lab4 (Weeks 8,9)

With all the code completed, this can take anything between approx. 30 min - 1 hour to run on Colab, so ensure to start early! Note: optimisation in section 3 may take awhile (ie. a few hours) to run



Academic integrity Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment along with interviews with the plagiarism officers at Monash University.

Late submission. Late submission of the lab 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 9 out of 10, so if you score 9.5, we will (sadly) give you 9. Labs 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).

Each lab is worth 5% and there are a number of sections and tasks with their own weighting. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

This lab is about applying what you have learnt so far from PyTorch and PyTorch-Lightning to a new model architecture. In this case, we will be using convolutional layers within the neural network (CNN) to build features that help to identify the shapes of objects in images. Although there are other uses for CNNs (such as for 1-dimensional data like analysis of audio recordings), this lab will focus on using CNNs for images in both gray-scale and RGB space. At the end of the lab, you should have a solid understanding of applying CNNs for classification.

- Section 1: Design a shallow CNN to classify samples form the Fashion-MNIST dataset. In this section, you can reuse a lot of the old code from the previous lab to build up your dataloaders. You will be implementing a shallow CNN and also learn data augmentation techniques to train a more robust model. Lastly, you will discuss the implications of using CNNs as opposed to MLPs for this dataset.
- Section 2: Design a shallow CNN and use it to classify STL images. The STL10 dataset we are using contains 96×96 RGB images, and you will be developing a shallow CNN to classify them into 10 classes. You will also be analysing your classification results to understand the intricacies of your model's performance.
- Section 3: Design and analysis of a deep CNN. In this section, you will improve on the performance by utilising methods you have already learnt (hyper-parameter tuning, data augmentation etc.) while extending the network size to contain additional convolutional layers. Additional analysis such as the confusion matrix will be introduced, and this will be applied to reinforce your understanding of the model's performance relative to predicting each class.

The learning outcomes for this lab are:

- Furthering your knowledge of PyTorch and PyTorch-Lightning
- Performing additional analysis and describing the results
- Learning new techniques to improve robustness of the model
- Learning new analysis tools, namely confusion matrix and showing predictions

It is **highly** recommended that you start on this as soon as you can. If you leave it to the second week of the lab, you may find yourself not having time to ask questions. **Remember**, if we get a lot of questions only in the last few labs sessions, the TAs will not stay back because it is unfair for them. Please have your questions ready from the first week of the lab. **This lab also takes a while to complete running the actual code! You want to ensure your models are trained ahead of time, so you can add in any necessary discussions and have all the outputs ready for submission by the deadline.**

Note: Most of the lab does not require you to use the Numpy library. You will be using PyTorch/PyTorch-Lightning in-built methods to create your tensors instead of Numpy. Follow the PyTorch/PyTorch-Lightning videos for more information.

Introduction

The prominence of deep learning models erupted with the ability to train deep neural networks more efficiently. The ability to split data up to different GPUs and run simultaneous training has propelled the research of neural networks forward. With CNNs, we are not required to handcraft the kernels to detect features (such as edges), but rather allow these kernels to be learnt automatically. Being able to train and test CNNs will be your entrance to hands-on deep learning, and you will re-use a lot of past concepts in order to complete this lab. The submission for lab 4 is due 25th of September (Sunday) 11:55 PM AEST.

It is recommended that you go through the following videos/documents prior to attending your lab 4 sessions:

1. Begin by reading through this document. This document contains all the relevant information for lab 4.
2. Watch the introductory video for this lab.
3. Follow the lecture series by Luke Ditria on CNNs with PyTorch/PyTorch-Lightning.
4. You may choose to use Google Colab for this lab so that you can utilise the free GPU provided. [1]

In the lab and in your own time, you will be completing the lab 4 notebook by going through this document and the provided notebook.

Please note that the provided checkpoint results are there for guidance and do not have to exactly match your outputs. If your results deviate too much though, it may be due to errors in your code.

[1]There is a lab 3 video detailing on how to use Google Colab. **Remember, Google Colab is NOT compulsory to use, but may help if you do not have your own GPU**. If you do use Google Colab, ensure the data is stored on Google Drive and accessed via mounting GDrive to your Google Colab. You will have to find other resources showing you how to mount Google drive.

Section 1: Design a shallow CNN to classify samples from the Fashion-MNIST dataset. [40%]

This section follows on from the previous lab, but we will be using CNNs instead of MLPs. You can reuse the Fashion-MNIST dataset code (*i.e.*, setting up dataloaders). **Note that we have arranged the order of data as: train, validation, and test.** In this section, you will also be introduced to the concept of data augmentation, and you will be performing additional analysis for this lab. Ensure that you add discussions where necessary.

- 1.1 Create and test a shallow CNN
- 1.2 Optimise the model's performance via data augmentation
- 1.3 Train a shallow CNN based on the augmented data
- 1.4 Compare validation accuracies obtained with the original data vs. data augmentation
- 1.5 Visualize 5 predictions of the test set along with ground-truth labels and input images for the best performing CNN Model
- 1.6 Describe differences in CNNs and MLPs

You can follow the shallow CNN architecture as described:

$$\text{image } (\boldsymbol{x} \in \mathbb{R}^{B\times C\times H\times W}) \rightarrow \textbf{conv1} \rightarrow \text{ReLU} \rightarrow \text{pool} \rightarrow \textbf{conv2} \rightarrow \text{ReLU} \rightarrow \text{pool} \rightarrow \textbf{fc1} \rightarrow \text{ReLU} \rightarrow \textbf{fc2} = \hat{\boldsymbol{y}}$$

End-to-end from input image to predicted class for a shallow CNN on MNIST data

Name	Type	in	out	kernel size	padding	stride
conv1	Conv2D	1	16	5×5	0	1
conv2	Conv2D	16	32	5×5	0	1
fc1	Linear	512	32	NA	NA	NA
fc2	Linear	32	10	NA	NA	NA
pool	Max Pooling	NA	NA	2×2	0	2

Table 1: Network structure

Table 1 provides details of each layer. Please note that you need to flatten the output of the maxpooling layer by using view() in PyTorch to connect to the FC layers. Also, note that the output of fc2 is fed to a softmax layer but in PyTorch, the CrossEntropyLoss has an inbuilt softmax function. Hyperparameters can be found in Table 2.

Parameter	Value	Description
λ	1e-1	Learning rate
Loss function	CE	*Use* ***"nn" module***
Optimizer	SGD	*Use* ***"torch" module***

Table 2: Hyperparameters for Section 1

Section 2: Design a shallow CNN and use it to classify STL images [40%]

In this section, you are writing the code foundations for Sections 2 and 3. You will be developing a shallow CNN model for the STL10 dataset. The dataset represents 10 object classes, namely {'airplane', 'bird', 'car', 'cat', 'deer', 'dog', 'horse', 'monkey', 'ship', 'truck'} in that order. The dataset has 13,000 color images of size 96×96 . We divide 8,000, 2,000, and 3,000 images for training, validation and testing, respectively.

- 2.1 Creating a PyTorch dataset for STL10
- 2.2 Visualize the dataset
- 2.3 Design, train, and test a Shallow CNN
- 2.4 Visualize experimental results
- 2.5 Visualize 'Top Classified' and 'Top Misclassified Images' from the test set

Remember to always visualize your data first before tackling the task! Understand the structure of the data before building your augmentations and training your model.

Train a shallow CNN for with the following layers:

$$\text{image } (\boldsymbol{x} \in \mathbb{R}^{B\times C\times H\times W}) \rightarrow \textbf{conv1} \rightarrow \text{ReLU} \rightarrow \textbf{conv2} \rightarrow \text{ReLU} \rightarrow \text{AvgPool} \rightarrow \textbf{fc1} \rightarrow \text{ReLU} \rightarrow \textbf{fc2} = \hat{\boldsymbol{y}}$$

End-to-end from input image to predicted class for a shallow CNN

You can refer to this documentation for AdaptiveAvgPooling.
The parameters of each layer within the CNN architecture can be found in Table 3:

Name	Type	in	out	kernel size	padding	stride
conv1	Conv2D	3	32	7×7	0	1
conv2	Conv2D	32	64	5×5	0	1
fc1	Linear	256	128	NA	NA	NA
fc2	Linear	128	10	NA	NA	NA
AvgPool (GAP)	AdaptiveAvgPool2D	NA	2×2	NA	NA	NA

Table 3: Network structure

The hyperparameters for this section can be found in Table 4
Ensure you add any discussion wherever it is required, and where you think is necessary.

Parameter	Value	Description
λ	1e-4	Learning rate
Loss function	CE	*Use **"nn" module***
Optimizer	ADAM	*Use **"torch" module***

Table 4: Hyperparameters for Section 1

Section 3: Design and analysis of a deep CNN [20%]

In this section, we will be further improving the CNN model that you have built in the previous section. You will implement a deeper version of the CNN with modular CNN blocks. We will provide you with the CNN architecture, but you will have to optimize the model yourself. You can compare the validation accuracy between Sections 2 and 3 to gauge the performance of the model. Note, you can use other techniques such as data augmentation to improve your model.

For Section 3, it is broken down into the following tasks:

- 3.1 Design a deep CNN model
- 3.2 Train and visualize results for your deep CNN
- 3.3 Define a 'Confusion Matrix' function and visualize the Confusion Matrix for the test set

Each modular convolutional block has this flow in inputs and outputs as seen below. The parameters for each convolutional layer in each block can be found in Table 5.

Conv1 $\rightarrow$ **ReLU** $\rightarrow$ **Conv2** $\rightarrow$ **ReLU** $\rightarrow$ **Conv3** $\rightarrow$ **ReLU**

The convolutional layers and activation functions within one convolutional block.

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

Table 5: Details of the convolutional block

Note: ReLU is applied after every convolution

Our deep CNN uses a stack of four of the aforementioned blocks. The network then uses a Global Average Pooling (GAP) layer followed by a linear layer [2]. All together, the structure of the network reads as:

image ($\boldsymbol{x} \in \mathbb{R}^{B\times C\times H\times W}$) $\rightarrow$ **conv-blk1** $\rightarrow$ **conv-blk2** $\rightarrow$ **conv-blk3** $\rightarrow$ **conv-blk4** $\rightarrow$ AvgPool $\rightarrow$ **fc** $=$ $\hat{\boldsymbol{y}}$

End-to-end from input image to predicted class for a deep CNN.

Name	in	out
conv-blk1	3	32
conv-blk2	32	64
conv-blk3	64	128
conv-blk3	128	192
fc1	1728	10

Table 6: Block structure

The details of the conv-blks are depicted in Table 6. Choose and optimize your own set of hyper parameters.You should be able to outperform the shallow CNN from section 2.

Remember to use view() to flatten your PyTorch tensor before passing it to the fully connected layer after GAP. Add in a discussion along with plots or tabulated results to display the optimal hyperparameters that you have found in improving the model's performance. Analyse the performance of the deep CNN using the confusion matrix, and discuss why it performs poorly on some classes.

Additional information.

- For understanding the confusion matrix: You can look at this website

Hopefully you have enjoyed this lab as much as we did when we made it! This lab aims to give you insight into training a simple CNN via the PyTorch-Lightning framework. Additional robustness and analysis techniques were also introduced in this lab and we hope that has been an interesting learning experience!

[2]For GAP, you can use (3×3) as the parameters for your GAP module