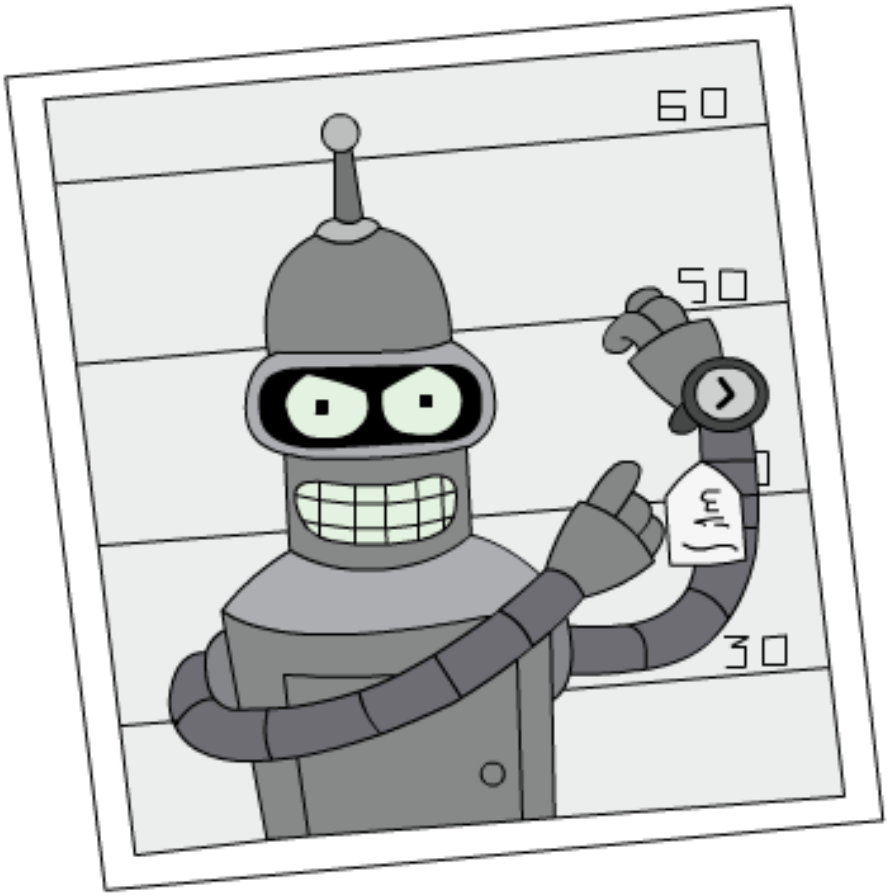
Multi-Layer Perceptrons (MLP)

ECE4179/5179/6179: Neural networks and deep learning
Lab3 (Weeks 6,7)



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 understanding PyTorch-Lightning basics and their primary functionalities. At the end of the lab you will know how to apply PyTorch functionalities for developing simple MLPs for model learning tasks like classification. The following are the three sections that you should complete.

- Section 1: Approximating a region of the sine function. This section will explore PyTorch Neural Network Structures based on MLPs, PyTorch datasets, data-loaders and PyTorch-Lightning functionalities. This section includes approximating a region of the sine function with a MLP to get a sense of how architecture and hyperparameters affect neural network performance.
- Section 2: Setting up a shalloe MLP for a simple classification problem. In this section you will be applying functionalities learned during section 1 to train a MLP for a grayscale image.
- Section 3: Train several Deep MLPs. In this section you will investigate the performance differences between shallow MLPs and Deep MLPs for the same dataset in section 2.

The learning outcomes for this lab are:

- Familiarising yourself with PyTorch and PyTorch-Lightning
- Understanding implementation of MLPs and network training.
- Using other essential libraries such as Numpy and Matplotlib.
- Analysing and describing results.

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.

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 fundamentals of deep learning models start off with understanding multilayer perceptrons and the training processes required. There are additional hyper-parameters that we can choose and tune, such as the type of optimizers, learning rate, activation functions, and model architecture to name a few. The ability of activation functions to model non-linearities is what allows MLPs to be able to generalise better than traditional machine learning models. In this lab, you will learn how to code MLPs with PyTorch and PyTorch-lightning frameworks and tune hyper parameters of the model to achieve optimal performance. Each section follows a similar set of tasks in terms of model training, with other section 3 having additional The submission for lab 3 is 11th of September (Sunday) 4:30 PM AEST.

It is recommended that you go through the following videos/documents prior to attending your lab 3:

1. Begin by reading through this document. This document contains all the relevant information for lab 3
2. Watch the introductory video for this lab
3. Follow the lecture series by Luke Ditria on PyTorch/PyTorch-Lightning
4. You may choose to use GoogleColab for this lab so that you can utilise the free GPU provided. There is a lab 3 video detailing on how to use GoogleColab.

In the lab and in your own time, you will be completing the lab 3 notebook by going through this document and the provided notebook.

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

Before you begin, make sure to follow the installation instructions in the notebook for PyTorch and PyTorch-Lightning. We have given you the set of necessary libraries and details on PyTorch functionalities that you need to use when implementing algorithm for Section 1.

Section 1: Approximate the sine function [30%]

This section includes how to develop and evaluate a small neural network for function approximation. You will approximate the sine function with a MLP. The following tasks need to be completed:

- 1.1 Creating a Pytorch dataset
- 1.2 Create instances from defined custom dataset and visualize the training dataset
- 1.3 Design the Shallow Linear MLP model using PyTorch Lightning Module.
- 1.4 Define training and testing methods for models
- 1.5 Generate scatter plots for above two trained models to compare noisy datapoints and denoised predictions
- 1.6 Visualize experimental results
- 1.7 Analyse the different optimizer results
- 1.8 Discuss why the simple MLP is able to denoise noisy data and approximate sine function

The first few tasks in this section aims to setup a dataloader class for the noisy sine data. Task 1.3 requires you to design a shallow linear MLP model with the following structure/hyper-parameters:

Parameter	Value	Description
λ	1e-2 (SGD), 1e-3 (ADAM)	Learning rate
# Hidden layers	1	The layers you will be coding
# Neurons	{64}	Number of neurons in each layer
Activation function	tanh	For each hidden layer
Loss function	MSE	*Use* ***"nn" module***
Optimizer	SGD, ADAM	*Use* ***"nn" module***

Table 1: Hyper parameters for section 1

After constructing the MLP model, you can train the model using the *Trainer* constructor as it has useful built-in methods such as tensorboard logging, model checkpointing, training and validation loop, early-stopping. [1] Ensure you add a discussion on the modelling results for each optimizer and add a discussion for the why the sine approximation works in this case.

[1] Make sure to check out the lecture videos to understand these concepts

Section 2: Training shallow MLP for custom dataset [40%]

In this section, you are writing the code foundations for sections 2 and 3. You will be developing a simple MLP model for the custom dataset that is provided for you.

- 2.1 Create custom dataloader
- 2.2 Visualize dataset
- 2.3 Design Shallow MLP
- 2.4 Train and evaluate model's performance for following scenarios
- 2.5 Plot Training losses for the different shallow networks
- 2.6 Plot Validation accuracies for the different shallow networks
- 2.7 Visualize 5 predictions of test set along with groundtruths and input images for 3rd Shallow MLP
- 2.8 Optimize hyper-parameters to improve accuracy of shallow network

For section 2, you will be comparing the performance of shallow and deep MLPs for the Fashion-MNIST dataset. The Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. The images are 28×28 grayscale images (so only one colour channel). Section 2 contains the following tasks:

Train a shallow MLP for the following 3 scenarios, consisting of just one hidden layer to classify the images according to the following design:

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

where n = 32, n= 128 and n = 512 scenarios.

You may pick your own hyper parameters for the model training in sections 2 and 3. The rest of the tasks in section 2 is similar to section 1. Ensure that you add a discussion on the best scenario for 2.3. Task 2.6 requires you to dive deeper in optimizing the hyper-parameters of the model and produce your own optimal results. Choose a few to optimize for, and produce it in an understandable format.

Section 3: Deep MLP [30%]

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

- 3.1 Design Deep MLP
- 3.2 Train and evaluate model's performance for following scenarios
- 3.3 Plot Training loss vs Validation loss for the different deep networks
- 3.4 Visualize 5 predictions of test set along with groundtruths and input images for 3rd Deep MLP
- 3.5 Discussion

Design and train the following deep MLPs for following scenarios in 3.2:

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

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

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

Visualize plots for the training and validation losses per epoch and add a discussion to compare the different scenarios for 3.2 accompanied with the results. Furthermore, discuss the performance differences between the best shallow and deep MLPs. You may discuss the MLP models where they had the same hyper parameters such as λ etc. What are its implications? For the final task, please discuss how well the deep MLP does and how does it compare with the shallow MLP for this dataset.

Additional information.

- You may use tensorboard instead of using csv to log and display the results
- You may find that for sections 2 and 3, deeper model may or may not be better than the shallow model for this particular dataset. Make sure to add a discussion for your reasoning.
- Understand the signs of overfitting in machine learning problems. You can have a look at this website

Hopefully this lab has given you insight in training a simple model via the PyTorch-Lightning framework, and provided realistic scenarios in which optimizing several hyper-parameters is required for any deep learning problem!