**Teacher: Dr. Qiaohong Li**

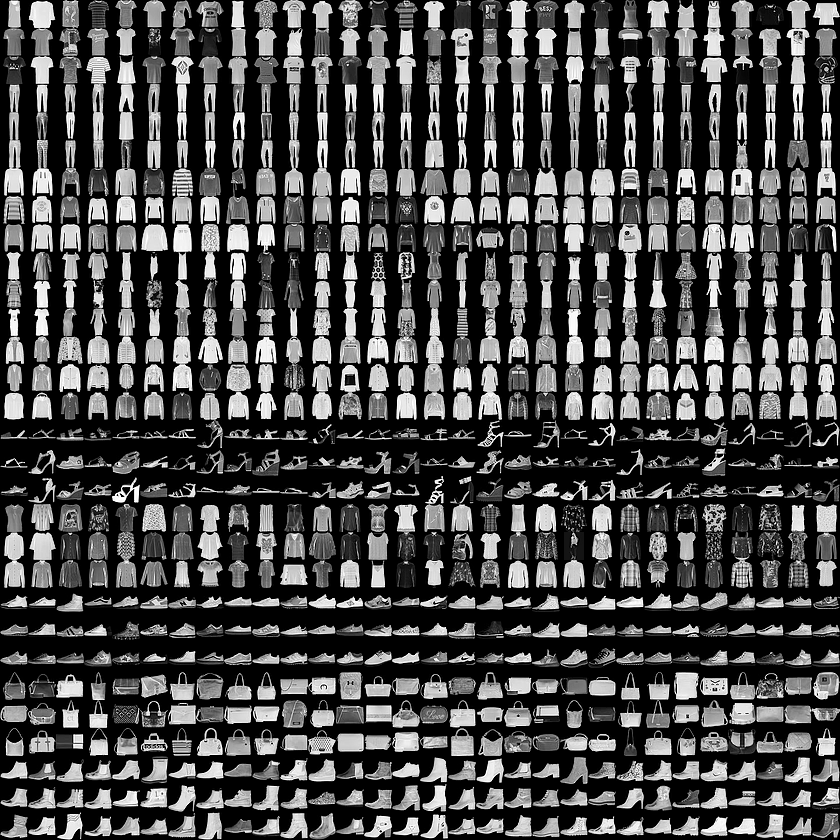
**Email: qhli@nus.edu.sg**

Exercise 1  
Explore MLPs

\_\_\_

## Introduction: Train the MLP models on Fashion-MNIST dataset.

**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. Each example is a 28x28 grayscale image, associated with a label from 10 classes. We intend **Fashion-MNIST** to serve as a direct drop-in replacement for the original **MNIST** dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.



**MLPs: Multilayer perceptrons** are a fully-connected network. They are also called deep feedforward networks or feedforward neural networks. MLPs are not optimal for processing sequential and multi-dimensional data patterns. By design, MLPs struggle to remember patterns in sequential data and requires a substantial number of parameters to process multi-dimensional data. Here, we use this as an example to explore how to build MLPs with different architectures and how to tune the MLPs hyperparameters, to observe the performance in different scenarios.

The following hyperparameters will be explored in this exercise:

* + The number of hidden layers or the number of layers.
  + The number of nodes at each layer.
  + The dropout tricks with different dropout rates.
  + The optimizer. (For each optimizer, the default parameters in tf.keras are used.)

## Notes

* While Fashion- MNIST data are 2D tensors, they should be reshaped accordingly depending on the type of input layer. For MLPs, it should be reshaped to 1D tensors.
* This is an image classification problem with 10 classes, thus the output layer should be a dense layer with 10 nodes, and **softmax** as the activation of the last layers.
* For the dense layers except the last layer, **relu** is used as the activation function.

**Optimizers in tf.keras**

'sgd': optimizers.SGD(),

'momentum': optimizers.SGD(momentum=0.9),

'nag': optimizers.SGD(momentum=0.9, nesterov=True),

'adagrad': optimizers.Adagrad(),

'adadelta': optimizers.Adadelta(),

'rmsprop': optimizers.RMSprop(),

'adam': optimizers.Adam()

**Please Fill this Table. And to conclude what do you find in this hyperparameter tuning exercise.**

Table: Different MLP network configurations and performance measures

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Layers | Regularizer | Optimizer | ReLU | Train Acc,% | Test Acc,% | #Trainable paras |
| 256-256-256 | None | SGD | None |  |  |  |
| 256-256-256 | L2(0.001) | SGD | Yes |  |  |  |
| 256-256-256 | L2(0.01) | SGD | Yes |  |  |  |
| 256-256-256 | None | SGD | Yes |  |  |  |
| 256-256-256 | Dropout(0.4) | SGD | Yes |  |  |  |
| 256-256-256 | Dropout(0.45) | SGD | Yes |  |  |  |
| 256-256-256 | Dropout(0.5) | SGD | Yes |  |  |  |
| 256-256-256 | Dropout(0.6) | SGD | Yes |  |  |  |
| 256-512-256 | Dropout(0.45) | SGD | Yes |  |  |  |
| 512-512-512 | Dropout(0.2) | SGD | Yes |  |  |  |
| 512-512-512 | Dropout(0.4) | SGD | Yes |  |  |  |
| 512-1024-512 | Dropout(0.45) | SGD | Yes |  |  |  |
| 1024-1024-1024 | Dropout(0.4) | SGD | Yes |  |  |  |
| 256-256-256 | Dropout(0.6) | Adam | Yes |  |  |  |
| 256-256-256 | Dropout(0.55) | Adam | Yes |  |  |  |
| 256-256-256 | Dropout(0.45) | Adam | Yes |  |  |  |
| 256-256-256 | Dropout(0.45) | RMSprop | Yes |  |  |  |
| 128-128-128 | Dropout(0.45) | Adam | Yes |  |  |  |
| 256-256 | Dropout(0.45) | Adam | Yes |  |  |  |
| 128-128 | Dropout(0.45) | Adam | Yes |  |  |  |

**Observations:**

1. Bigger networks do not necessarily translate to better performance.
2. Generally, we'll find that the Dropout layer has better performance than l2.
3. ...