**Teacher: Dr. Qiaohong Li**

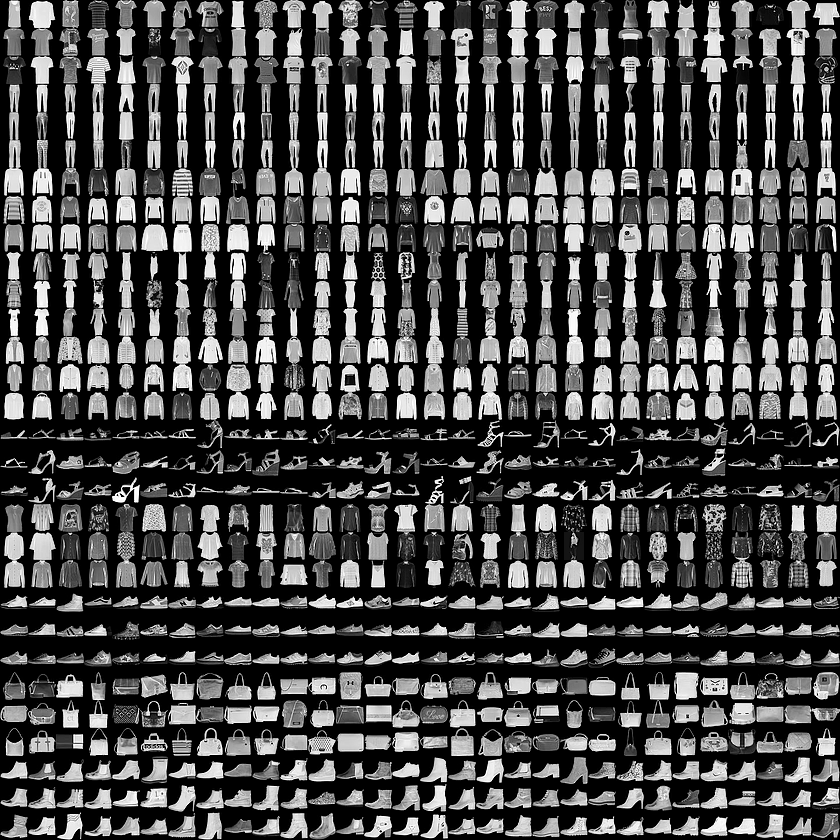
**Email: qhli@nus.edu.sg**

Exercise 2  
Explore CNNs

\_\_\_

## Introduction: Train the CNN 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.



**CNNs: Convolutional neural networks:** For multi-dimensional data like images and videos, a CNN excels in extracting feature maps for classification, segmentation, generation, and other purposes. In some cases, CNN in the form of a 1D convolution is also used for networks with sequential input data. However, in most deep learning models, MLPs, RNNs, and CNNs are combined to make the most out of each network. For example, when we handle with image classification problem, generally we need to add the dense layer as the last layer which has the nodes corresponding to the number of classes of the dataset.

The following hyperparameters will be explored in this exercise:

* + The dropout tricks with different dropout rates.
  + The optimizer. (For each optimizer, the default parameters in tf.keras are used.)

## Notes

* Convolutions without padding reduces the feature map sizes.
* The input is padded with zeros around its borders to keep the dimensions unchanged after the convolution.
* The relu activation function is already an argument of **Conv2D**. The relu function can be brought out as an Activation layer when the batch normalization layer is included in the model.
* **MaxPooling2D** compresses each feature map. Every patch of size pool\_size × pool\_size is reduced to one pixel.
* The role of **Flatten** is to convert the stack of feature maps into a vector format that is suitable for either Dropout or Dense layers, similar to the MLP model output layer.
* Batch normalization is used in deep CNNs so that large learning rates can be used without causing instability during training. (not used in this exercise)

**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 CNN network configurations and performance measures

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layers | Optimizer | Regularizer | Train Acc,% | Test Acc,% | #Trainable paras |
| 64-64-64 | SGD | Dropout(0.2) |  |  |  |
| 64-64-64 | RMSprop | Dropout(0.2) |  |  |  |
| 64-64-64 | Adam | Dropout(0.2) |  |  |  |
| 64-64-64 | Adam | Dropout(0.4) |  |  |  |

**Observations:**

1. CNN model requires a smaller number of parameters compared to MLP model.
2. ...