Pei-Chi Pan, Raunak Sarbajna and Christoph F. Eick

COSC 3337: Data Science I

Fall 2025

TASK 2: Build a Classical Neural Network Model

for the Fashion MNIST Dataset (Individual Taks)

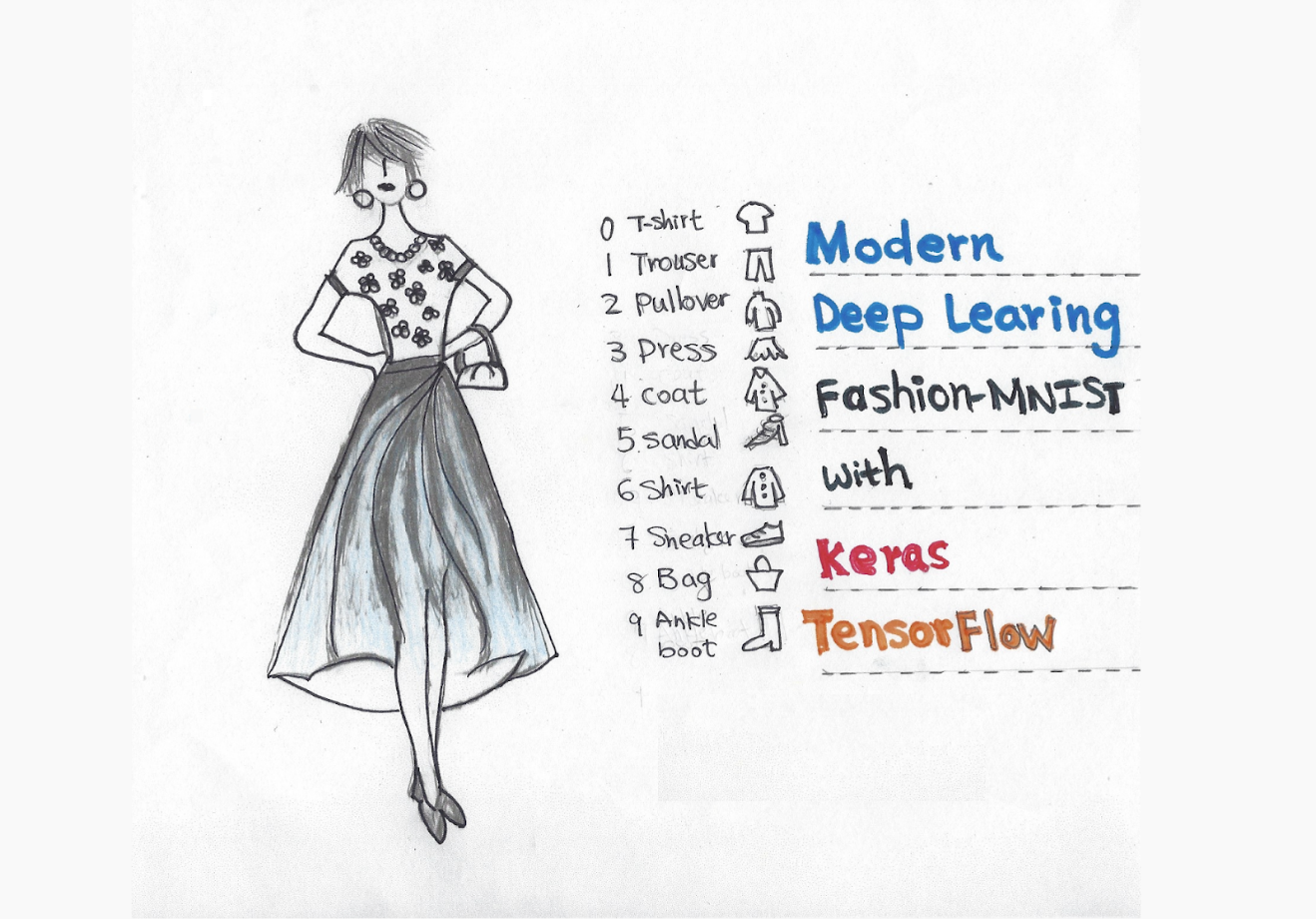


Fig. 1: Training a model to classify images is the archetypal neural network task

Submission Deadlines Task2: Mo., Oct. 6, 11:59p

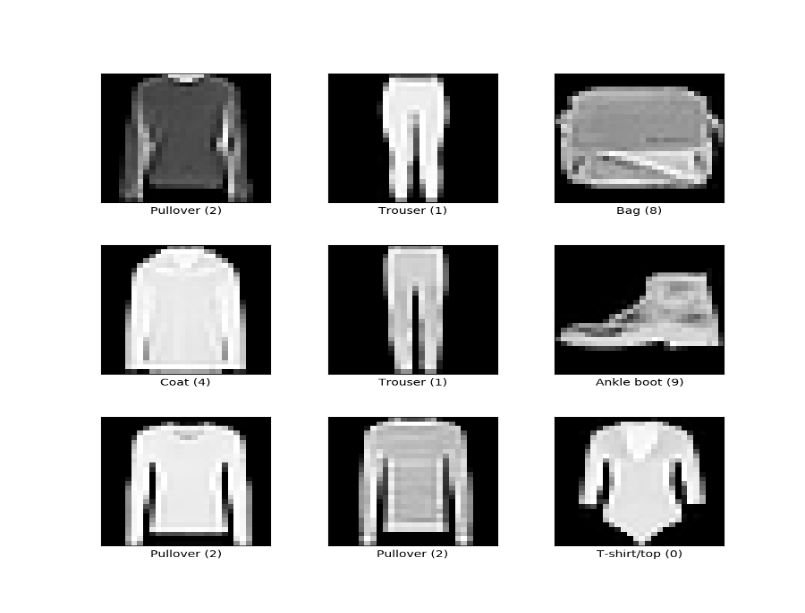
Responsible TA: Pei-Chi Pan

Last Updated: September 22

In this task, you will explore the world of deep learning by building a neural network to classify clothing items in the Fashion MNIST dataset. You will learn how to build, train, and evaluate a model, gaining valuable insights into neural network capabilities.

**Dataset:**

Fashion MNIST is a popular image classification dataset containing 70,000 grayscale images of 10 different clothing items (T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot). We will use this dataset to train our neural network to identify these clothing items. Below some samples are given from Fashion MNIST dataset:



**Tasks:**

**1. Preprocessing**

1. Download the Fashion MNIST dataset using TensorFlow or another suitable library.
2. Preprocess the data: normalize pixel values, split the data into training and testing sets.

**2. Building the Neural Network (5 pts)**

1. Choose a neural network architecture suitable for image classification. Consider using a Convolutional Neural Network (CNN) due to its effectiveness in this domain.
2. Define the network architecture in your chosen library (e.g., TensorFlow, PyTorch). Use layers like convolutional, pooling, and dense layers.
3. Compile the model, specifying the optimizer (e.g., Adam) and the loss function (e.g., categorical cross-entropy for one-hot encoding or sparse cross-entropy).

**3. Training and Evaluation (5 pts)**

1. Train the model on the training set with 3 different epochs (e.g., 10/20/50).
2. Monitor the training process by logging metrics like accuracy and loss after each epoch.
3. Evaluate the model's performance on the test set using metrics like accuracy, precision, and recall for each class.
4. Experiment with at least 3 different hyperparameters to improve the model's performance. Your model needs to have **at least 90%** accuracy on the testing data.
   * learning rate (e.g., 0.1/0.25/0.30),
   * number of layers (e.g., 2/3/5),
   * neurons per layer (e.g., 10/20/30)

**4. Analysis (10 pts)**

1. Only analyse for the best model from your experiments.
2. Analyse the training and evaluation results. Are there any classes with particularly low accuracy?
3. Display the model's confusion matrix. Can you identify any interesting patterns or mistakes? Visualize the lowest performing class (e.g., its accuracy compared to other classes in a bar graph, etc.)
4. Determine which class is the most difficult to predict by seeing which has the lowest F1. Try to explain why this class is challenging to predict.

**Submission:**

* Submit a report including:

1. PDF report

* + Talk about your CNN architecture design that gives the best result. What hyperparameters (e.g., number of epochs, learning rate) did you set? How many convolution layers? Did you include padding? What pooling method did you use? Also, calculate how many features are in the final fully connected layer.
  + Visualization of training and evaluation metrics (graphs of training loss and accuracy at each epoch, confusion matrices).
  + Your answers to the analysis questions.
  + Briefly discuss the limitations of your model and potential improvements for future exploration.

2. Code, either a .py or .ipynb file

**Bonus: (up to 5 pts)**

* Implement data augmentation techniques to improve model generalizability and accuracy.
* Fine-tune the CNN model to reach 95% accuracy.
* Apply your model to classify images from a different dataset of your choice. You can try to use CIFAR-100, which consists of 32×32 color images of 100 categories (e.g., apple, bear, clock, dolphin) from TensorFlow.

**Resources:**

* Fashion MNIST dataset: <http://pytorch.org/vision/stable/generated/torchvision.datasets.FashionMNIST.html>
* TensorFlow tutorials: <https://www.tensorflow.org/tutorials>
* PyTorch tutorials: <https://pytorch.org/tutorials/>

**Pre-Processing Example:**

**A screen shot of a computer program

Description automatically generated**