Introduction to Data Science Tools & Techniques

## Group Assignment - 4

# Classification of Grayscale Images

### Group Members:

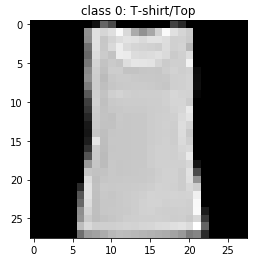
Ali Nauman – 18L1863  
Hammad Akram – 18L1808  
Ammar Hussain – 18L1834

## Data-set Description

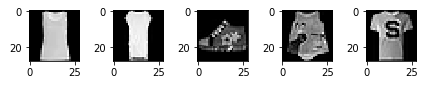
The dataset consists of a training set of 30,000 examples and a test set of 5,000 examples. They belong to 10 different categories. The validation set is not provided, but you can randomly pick a subset of the training set for validation. The labels of the first 2,000 test examples are given, we will analyze the performance of our proposed method by exploiting the 2,000 test examples. We have not utilized any examples from the test set for training.

There are 10 classes in total:  
0 T-shirt/Top  
1 Trouser  
2 Pullover  
3 Dress  
4 Coat  
5 Sandal  
6 Shirt  
7 Sneaker  
8 Bag  
9 Ankle boot

Showing a sample data. The first example belongs to class 0: T-Shirt/Top



We shall see a 5 example and see how these images look.



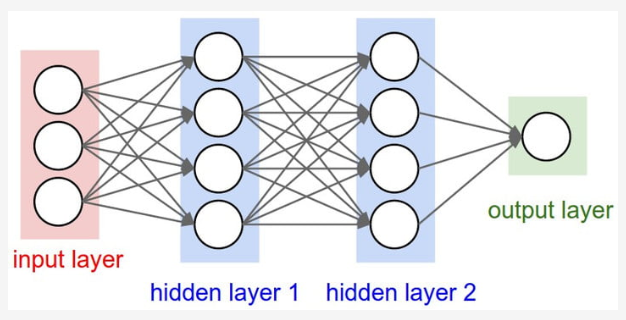
## Classifier Description

We will be building a *Neural Network* from scratch. We’ll train it to recognize the above displayed images using the data set provided.

### Artificial Neural Network

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

For a basic idea of how a deep learning neural network learns, imagine a factory line. After the raw materials (the data set) are input, they are then passed down the conveyer belt, with each subsequent stop or layer extracting a different set of high-level features. If the network is intended to recognize an object, the first layer might analyze the brightness of its pixels.



The next layer could then identify any edges in the image, based on lines of similar pixels. After this, another layer may recognize textures and shapes, and so on. By the time the fourth or fifth layer is reached, the deep learning net will have created complex feature detectors.

#### Types of Neural Networks:

There are multiple types of neural network, each of which come with their own specific use cases and levels of complexity. The most basic type of neural net is something called a feed-forward neural network, in which information travels in only one direction from input to output.

A more widely used type of network is the recurrent neural network, in which data can flow in multiple directions. These neural networks possess greater learning abilities and are widely employed for more complex tasks such as learning handwriting or language recognition.

#### Main Idea:

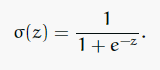
We’ll use just basic Python with NumPy to build our feed-forward network. We will split our data in *Training Set* and *Validation Set*. Using the *Training Set*, we will build our model and experiment it on the *Validation Set*, before we predict the labels for the *Test Set*.

#### Sigmoid Function:

The forward pass on a single example *‘x’* executes the following computation,



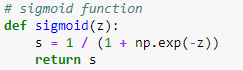
Here is the sigmoid function,



We will vectorize by stacking examples side-by-side, so that our input matrix ‘*X’* has an example in each column. The vectorized form of the forward pass is then:

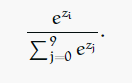


We shall define the sigmoid function as following,



#### Forward Propagation:

Only the last layer of our network is changing. To add the softmax, we have to replace our lone, final node with a 10-unit layer. Its final activations are the exponentials of its z-values, normalized across all ten such exponentials. So instead of just computing σ(z), we compute the activation for each unit i:



So, in our vectorized code, the last line of forward propagation will be as following,

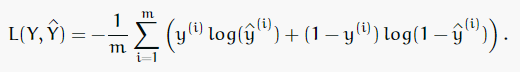


#### Cost Function:

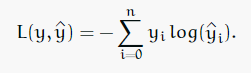
We’ll use cross-entropy for our cost function. The formula for a single training example is:



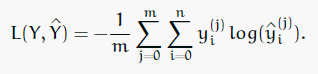
Averaging over a training set of *‘m’* examples we then have:



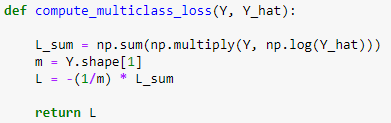
Our cost function now will have to generalize to more than two classes. The general formula for *‘n’* classes is following,



Averaging over *‘m’* training examples this becomes:



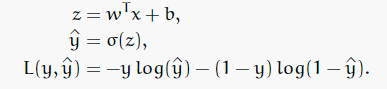
Thus, we have defined the cost function as following,



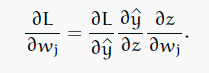
#### Backward Propagation:

For back propagation, we’ll need to know how *L* changes with respect to each component *wj* of *w*. That is, we must compute each *∂L/∂wj*.

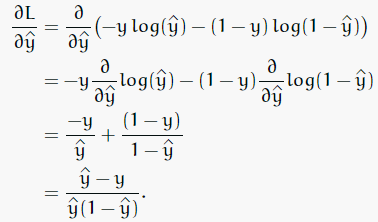
Focusing on a single example will make it easier to derive the formulas we need. Holding all values except *wj* fixed, we can think of *L* as being computed in three steps: *wj→z→ŷ→L*. The formulas for these steps are:



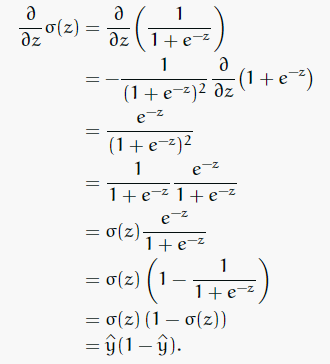
And the chain rule tells us:



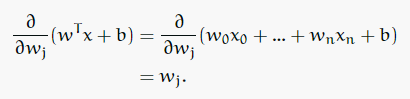
Looking at *∂L/∂ŷ* first:



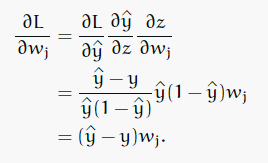
Next we want *∂ŷ/∂z*:



Lastly we tackle *∂z/∂wj*:



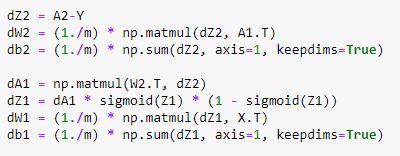
Finally we can substitute into the chain rule to find:



In vectorized form with *‘m’* training examples this gives us:



A softmax generalizes the sigmoid activation we’ve been using. In the code, we have defined the gradients as following,



## Results on Test Set

We will be evaluating the labels of the first 2,000 test examples. We will analyze the performance of our proposed method by exploiting the 2,000 test examples.

**Testing Accuracy**: 84%

Furthermore, we have compiled the complete set of test examples predictions in the file mentioned in the report, “*predicted\_labels.h5*”.

## Run-time

We run the model for 2000 epochs and computed the cost and the run-time on how long did it take for the model to run.

**Run-time**: 1385.5266199111938 seconds

## Hardware and Software Specifications of the Computer used for Performance Evaluation

Please find below the hardware details of the computer on which the model was run,

RAM: 8.00 GB  
Processor: Intel® Core™ i3-7100U CPU @ 2.40GHz  
System Type: 64-bit OS

Please find below the software specifications used in this,

IDE: Ananconda – Jupyter Notebook  
Programming Language: Python  
Libraries: NumPy, Pandas, Matplotlib