**PROJECT REPORT**

**CONVOLUTIONAL NEURAL NETWORK MODEL TO CLASSIFY IMAGES OF ZALANDO’S ARTICLES**

**MOUNIKA KANAKANTI**

**Problem Statement:**

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. Build a CNN model to classify the images.

**Steps Involved in Building a CNN Classifier:**

* Data Preparation
  + Reshaping the train and test data
  + Separating labels from data and checking for its distribution
  + Label Encoding
  + Normalizing the data
* Building CNN Architecture
* Evaluating the CNN Model

**Building CNN Architecture:**

I used the Keras Sequential API where each layer can be added one by one.

* The first layer is Conv2D which convolves with the layer inputs to produce output tensors. The input for this layer are:

Filters (Dimensionality of output space) = 32

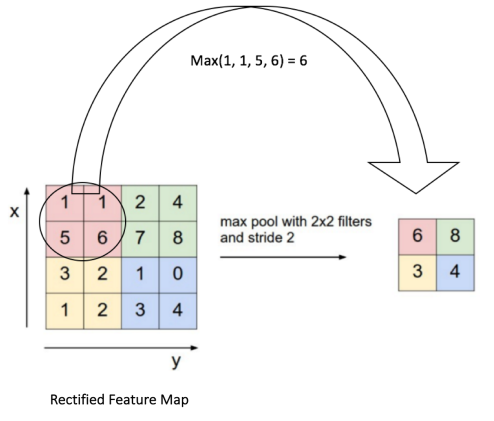
Kernel size = (3,3) (The height and width of a 2D Convolution window or in other words, 3x3 matrix that slides over the input image to form a feature map)

Strides = 1 (Specifies the number of pixels by which the kernel can slide through the image)

‘Relu’(Rectified Linear Unit) is an activation function used to add non – linearity in our CNN.

The output of this Conv2D layer passes to the next Conv2D and acts as the input to this layer.

* The next important layer is MaxPooling2D which helps in dimensionality reduction of each feature map. It also makes the network invariant to small transformations. (a small change in input will not affect the output of Pooling).



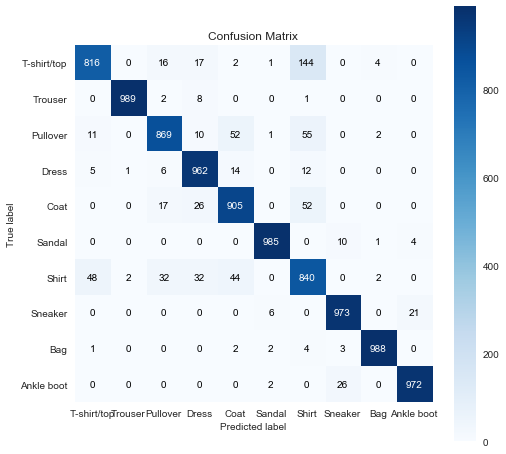
* The Dropout layer is used for regularization where few of the neurons are turned off by setting their weights to zero. This helps the network to learn in a distributed way and prevent overfitting.
* Finally the output feature maps need to be sent to a fully connected layer to calculate probabilties and classify them. For this purpose the output feature maps should first be converted to a single 1D vector using Flatten().

**Evaluating the CNN Model:**

Train Accuracy : 95.2

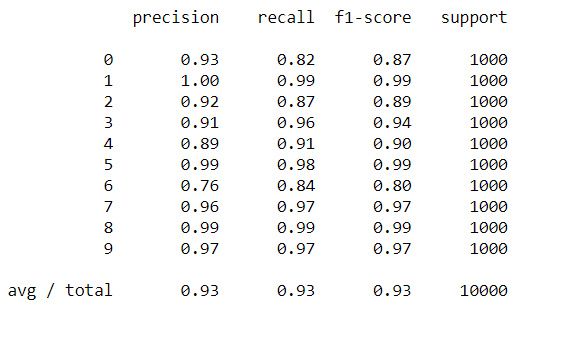
Test Accuracy : 93

As the accuracies of train and tests are quite good, let’s check the confusion matrix for more details.



**Fig L Confusion Matrix**

From the above confusion matrix, we observe that maximum errors were made while predicting shirts which were confused to be T-shirt/top which is quite common even to the human eye.



**Fig : Classification Report**

The highest scores were observed while predicting trousers and sneakers (may be because there were no other similar things) and the lowest were observed while predicting shirts because they were mostly confused with tops or pullovers.