Table of Contents

[I. Abstract 1](#_Toc11242599)

[II. Introduction 1](#_Toc11242600)

[III. Dataset – Fashion MNIST 1](#_Toc11242601)

[1. CNN Dataset Preparation 1](#_Toc11242602)

[2. KNN Dataset Preparation 1](#_Toc11242603)

[IV. Methodology 1](#_Toc11242604)

[1. Convolutional Neural Network 1](#_Toc11242605)

[a. Convolution Layer 2](#_Toc11242606)

[b. ReLU Activation Function 3](#_Toc11242607)

[c. Pooling Layer 4](#_Toc11242608)

[d. Dropout Layer 4](#_Toc11242609)

[e. Fully Connected Layer 5](#_Toc11242610)

[f. Softmax Activation Function 5](#_Toc11242611)

[g. CNN Model Implementation 6](#_Toc11242612)

[2. K Nearest Neighbor 7](#_Toc11242613)

[a. How K-NN works 7](#_Toc11242614)

[b. K-NN Fundamentals 7](#_Toc11242615)

[c. KNN Implementation 8](#_Toc11242616)

[d. Similarity Metric 8](#_Toc11242617)

[e. K-Value Insight 8](#_Toc11242618)

[f. Euclidean Distance Classification 9](#_Toc11242619)

[g. L0 Distance Classification 9](#_Toc11242620)

[V. Result and Discussion 9](#_Toc11242621)

[1. Convolution Neural Network 9](#_Toc11242622)

[2. K-Nearest Neighbor 10](#_Toc11242623)

[VI. References 13](#_Toc11242624)

CNN and KNN Implementation Using TensorFlow Keras

Trung Quan Lai, Stefano Felau

# Abstract

Machine learning concept has become one of the most common terms in the modern world. It provides computers with the ability to predict and detect an image throughout a training progress with a huge of input examples. It can learn and improve by repeating the same tasks until it reaches a particular result. Nowadays, many machine learning-based applications can predict or detect a specific item with the accuracy of more than 90 percent and it has contributed benefits into essential areas which help people more than ever.

In this paper, we are going to apply CNN (Convolution Neural Network) and KNN (Key Nearest Neighbor) into image recognition with supporting by TensorFlow Keras library. These state-of-art algorithms are going to use Fashion MNIST and try to archive high accuracy as much as possible over some configurations and testing. We also discuss in detail how to implement by using Python. Training and Testing environment would be Google Colaboratory which support to run a Machine Learning project with a huge dataset.

# Introduction

The visual cortex is one inspiration that usually prefers to when talking about Neural Network [1]. It is a part of the brain that interprets the visual of the eyes getting from the retina. The information from retina will pass to the first region which has neurons that are sensitive to particular simple visual forms such as edges and corners then the data of this neurons influence the activity of another part with more complicated ways such as faces or specific objects. The processing of information through these layers will be implemented by using an artificial neural network called Convolutional Neural Network (CNN) to solve the problem.

The KNN learning algorithm, on the other hand, is an instance-based algorithm which classifies/label each object based on the object(s) that is closest to, using a distance metric. It is easy to understand because it simply just stores and learns from the training data. The distance metric the will be used is the Euclidean distance and L0 distance metric to help with the classification.

# Dataset – Fashion MNIST

Fashion MNIST dataset contains 60,000 examples of the training set and 10,000 examples of the testing set that is stored in CSV files [2]. Each row has all information on the grayscale image. The first column associated with labels of 10 classes in different fashion items and the rest has 784 pixels which represent the color from darkness to lightness between 0 to 255.

## CNN Dataset Preparation

Firstly, the raw dataset should be reshaped into a 28x28 image by taking 784-pixel values and separated the label column into another vector. This step applies for both training and testing dataset. After that, the training dataset would be split into two, one for training and another for validation purpose. The ratio would be 80% for training (48,000 examples) and 20% for validation (12,000 examples).

## KNN Dataset Preparation

Splitting dataset is kept the same structure as raw Fashion MNIST dataset with 60,000 and 10,000 examples for training and testing respectively.

# Methodology

## Convolutional Neural Network

It is kind of deep neural network which was designed from the biologically driven models and inspires from how a human perceives an image into the brain in different layers. It is commonly used in image processing pattern recognition. There are some advantages and disadvantages of CNN compared with the typical neural network.

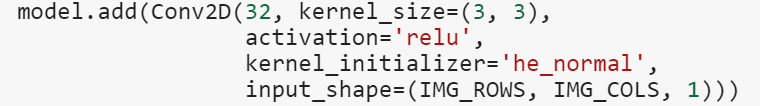
As a typical feed-forward neural Nets, every neuron will be connected together throughout the network [3] which causes a big challenge to deal with a large number of parameters that they need to train and learn, for example, an image with 256 pixel width, 256 pixels high and 3 channels then stretching out to get 196, 608 parameters. In the other hand, it also comes up with overfitting model. In CNN, this problem can be resolved by using a filter to interact with a specific location of the image then slide over the entire image until it reaches to the end to create a feature or activation map.

Shared weight and bias are also a big advantage of CNN which contribute to reducing the number of parameters significantly. Differently, from the typical neural network, the weight would be learned and updated during the training progress. In CNN, the output activation map is generated by the same filter in the same layer [4] such as simple edges in the first layer and turn into more complex when taking through the depth network. Another feature of CNN is pooling layer, it is also an important aspect in term of cutting down the dimension of the image.

A weakness of CNN might come from the amount of data that it requires. It can cause an overfitting problem if the training example has a small dataset as well as the memory is consumed when training the network. It cannot run directly on the normal computer with a large database which can cause some limitations on doing experiments. Besides that, the CNN still depend on the human invention to gain knowledge of its architecture which helps to understand and build an effective network such as a number of layers or input volume need to be trained in term of achieving the best performance.

### Convolution Layer

Taking the Fashion MNIST image (28x28). Instead of stretching it out into one long vector as a typical neural network, the structure of the image is going to be kept with two-dimensional input. And then the weights are going to be a small filter, in this case, a three by three filter.



The filter always has the same channel as the input volume, for example, in the digital image with three channels (RGB), the filter would be 3x3x3. In this paper, the Fashion MNIST is a grayscale image, so the depth always is one. The filter is going to take and slide over the image partially and compute dot products by multiplication between the filter with the spatial location in the image then plus bias term. Basically, doing W transpose times x plus bias.

Taking detail into how to slide the filter over all the spatial locations. The filter starts at the upper left-hand corner and basically center the filter on top of every pixel in the input volume. At every position, the dot product would be calculated by multiplying two matrices between the filter and spatial location from the input image then it will produce one value in the output activation map. The output dimensions are not the same as the input dimension depends on the size of the filter and how the filter is going, such as going through every pixel or two pixels over at a time.

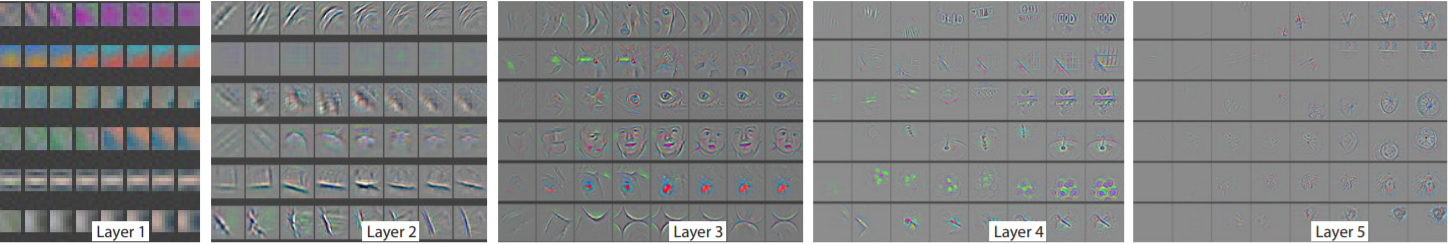


Activation map process.

In convolutional layer, multiple filters are going to be used because each filter represent to a specific edge which may contain from the image. Then the output will generate many activation maps stacking together because each filter will create one feature by itself. The second and other filters are going to slide as the same method as the first filter. People can do this for as many filters as they want to have in this layer, for example, if we have six filters then we are going to get six activation maps out.

Zero padding is common used in Convolution Neural Network as it will cover outside the image with 0 values to make the size of the image bigger as what we expect. In the result, it can place a filter centered at the upper right-hand pixel location of the input image. Another reason for that is due to information losing because the filter strides over the upper-left corner only one time.

Following formulas can explain the size of the activation map:

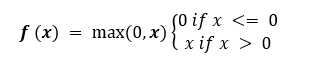


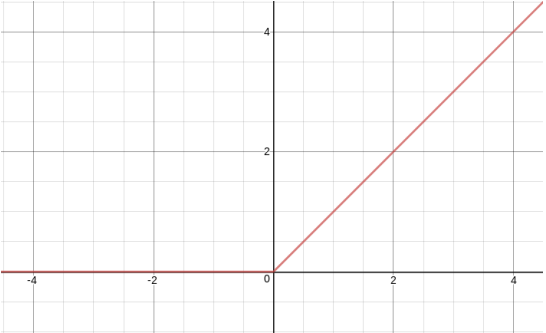
Different features through the Neuron Network [5]

The example above shows different edges going depth entire the network. It ends up with multiple layers of filters with basic edges on the first layer and then in the middle lever, it is going to get more complex kind of features, for example, corners, blobs and so on until getting to the high-level features which look more resemble concepts.

### ReLU Activation Function

ReLU which is short for Rectified Linear Unit. The input goes through this function is going to get the result either 0 or itself depended on its value but always greater or equal to 0. For example, if the input value is less than or equal 0, the ReLU function will turn it into 0. On the other hand, the output will be remained itself if the input is greater than 0.

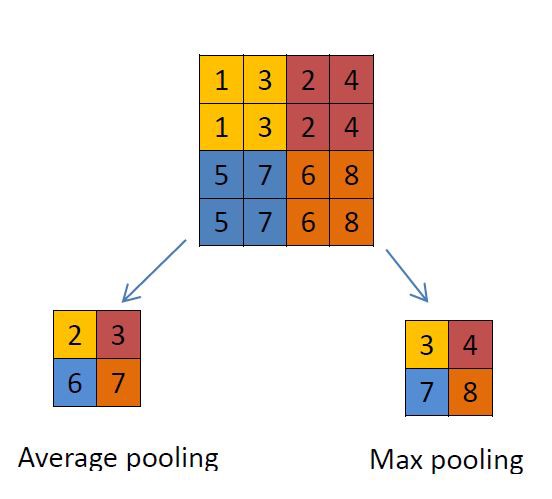
****



The Rectified Linear Unit (ReLU) [6]

### **Pooling Layer**

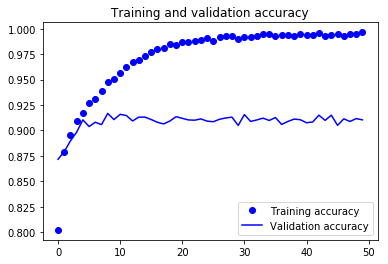
The idea of pooling layer is to reduce the size of the imzge, make it smaller, another word is to have fewer parameters at the end. It does precisely downsamples, for example, 28x28 image and then reduce to get 14x14 output. It is vital that this does not do anything in depth, the input depth is going to be the same as output depth. There are two ways to do pooing that are average and max pooling. However, the common way is max pooling and used in this paper. The pooling layer also has a filter size and it is going to be the region that we pool over, for example, we have a filter 2x2 and stride is 2, then it is going to stride along the input volume in the same way as the convolution layer did. But here, instead of doing these dot products, it takes the maximum value of the input volume in that region.



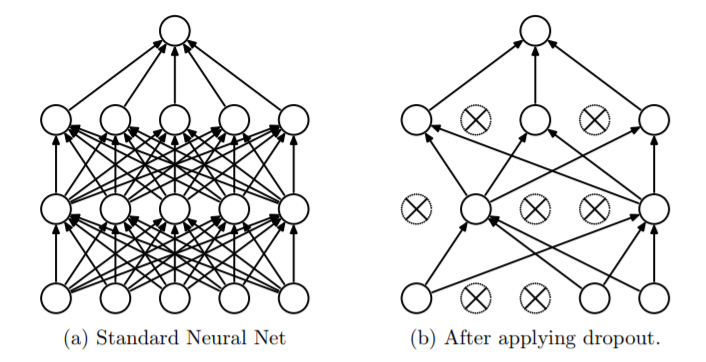
Max and Average Pooling

### Dropout Layer

This is a method of randomly ignoring the units from certain neurons during training [7]. It is also a form of regularization which is used in another type of machine learning. It is applied in this project to prevent over-fitting during training stage when using a typical CNN with ConvNet + ReLU, Pooling, Flatten and Dense throughout the network. The accuracy and loss between training and validation do not match at all.



Accuracy and Loss without Dropout



Dropout diagram [8]

In typical feed-forward neural Nets, neurons in current layer is connected to every node in the next level. Compared to the dropout network, neurons of the dropout layer are blocked randomly so that the output node cannot rely on any one feature because any one feature or its own input could go away at random.

### Fully Connected Layer

It is the same as the fully connected layer in the typical neuron network. In this case, taking the convolutional network output, which, it might have the width x height x depth with depth equals 1 in this paper. Then we take all of them and mainly stretch out as basically 1D input as used in a typical neuron network. Now, these neurons are going to be connected to every other in the next layer. In this layer, it aggregates all of this information together to get a score for matching which class relevant to them.

### Softmax Activation Function

The softmax activation function is implemented into a fully connected layer. It is very similar to Sigmoid activation function which taking a probability in between 0 and 1 that determines how likely something happens. With Sigmoid function in term of probability, the output is also in between 0 and 1, for example, we have five output and each output has a probability of 0.8. It does not make sense because the probability of each class needs to sum up to 1. It means that the Sigmoid function returns a value in between 0 and 1 but it does not represent the probability of something happens. However, with Softmax function, the output sums up to 1, the smallest value has the smallest probability and the highest value has the highest probability with it can be represented by:

[9]

### CNN Model Implementation

We are going to define a network as a sequence of the layer using the sequential class supporting by Tensorflow-Keras. Once the instance of the sequential is created, we can add new layers.

Con2D is the first layer of the model. In this layer, we need to define parameters such as:

* filters: the number of filters in the layer. The sequence would be 32, 64, 128.
* kernel\_size: the dimension of the filter (3x3).
* activation: define the activation function, in this case, ReLU.
* kernel\_initializer - the function used for initializing the kernel.
* input\_shape: define the input shape or dimension of the input image (28x28).

MaxPooling2D is the pooling layer with parameters

* pool\_size: the dimension of filter (2x2)

Dropout is the dropout function with the ratio is 0.25, 0.25, 0.4 and 0.4 respectively.

Flatten is the function which use to stretch the matrix input into a vector, so it has no parameter.

Dense is the fully connected layer with parameters

* units: the dimension of the output, the sequence would be 128 and 10.
* activation: activation function – ReLu and Softmax

Then compiling the model with parameters:

* loss.
* optimizer.
* metrics.



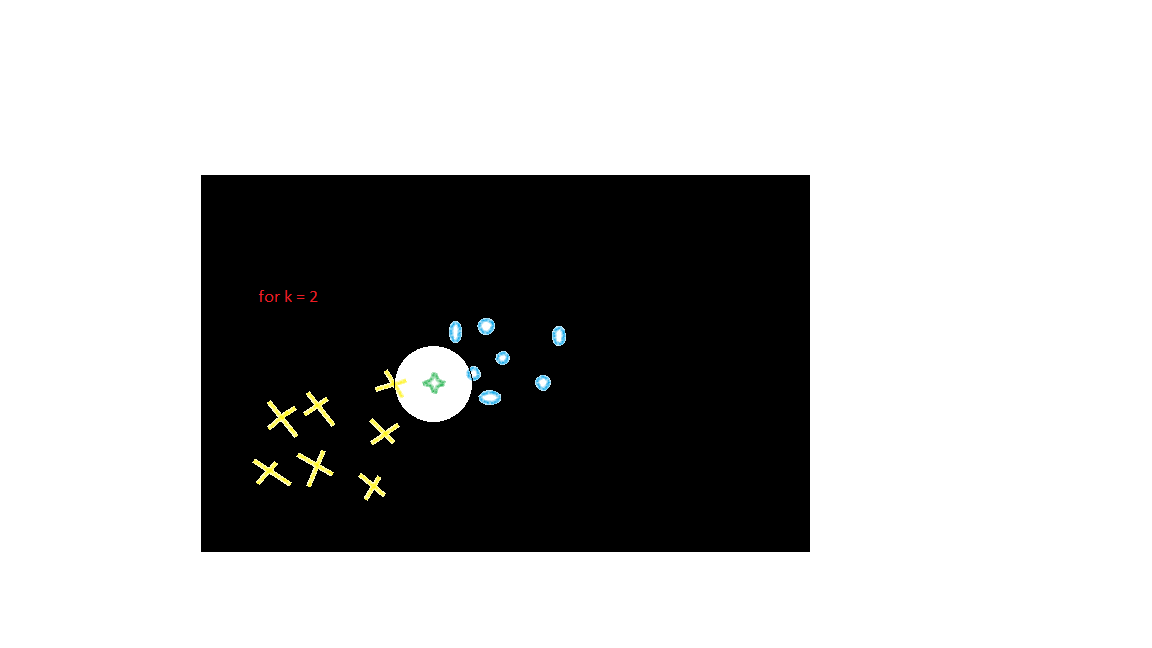
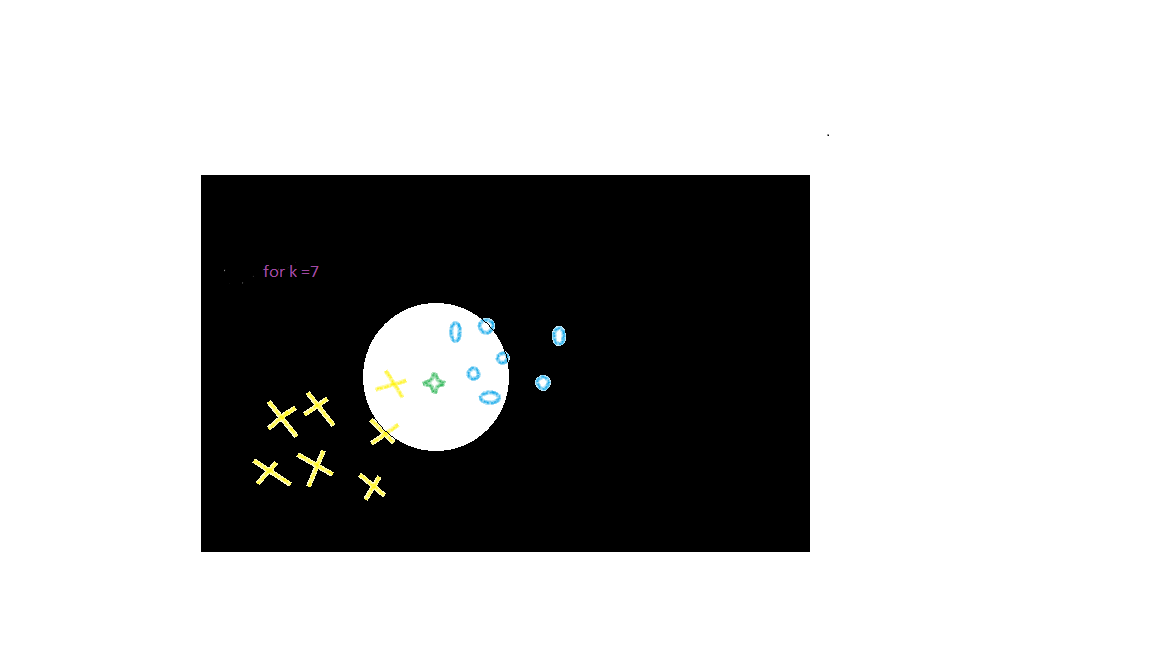
## K Nearest Neighbor

It is one of the well-known instance based-learning techniques that falls under the supervised machine learning domain. It only stocks up the necessary classification from the training examples and any assumptions outside these examples are put on hold for the purpose of awaiting a new instance to be classified [10]. K-nearest neighbour is most often referred to as a lazy learner because it simply does not have a training phase. That is, it does not obtain a genuine distinction from the training data, meaning it only memorizes the dataset. Ideally, this learning method has been used to solve numerous problems in classification [11]. According to [12] there must be a designated origin of/for similarities, description for the distances between the similarities and the decision for the k-value.

### How K-NN works

Suppose that there is a new problem where the only 2 classes are that of “yellow-x images” and “blue-o images”. Now a new instance “green-star” needs to be classified and with K-NN this can be done by assigning a k-value. For k = 2, i.e. the nearest 2 points to the provided instance are as shown below, at this point we cannot yet decide which class the “green-star” belongs to. As a result, will have to use choose the value of k to be an *odd number* so that we can obtain a more sensible classification.

**Figure1.1**:Instance Space, k = 2**Figure1.2**:Instance Space, k= 7

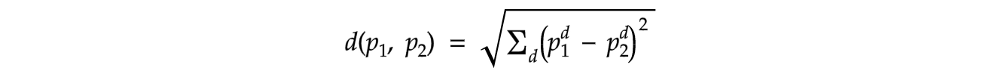


**Source:** Adapted by the researcher (Setefano. Finau.)

For k = 7, there is a clear implication that our assigned instance is closer to more “blue-o images” than that of the “yellow-x images”. Hence the “green-star image” falls under the “blue-o images” class.

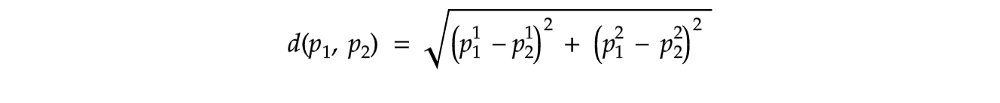
### K-NN Fundamentals

For the computers to understand the information under these circumstances, there is the need to clearly state the nature of what is required and sends it off to the program. In order to do so, we use the Euclidean distance between two points on an XY-plane as shown below:

****

**Source:** Intro to image classification with KNN [13]

Generalizing this equation needs to be done for all parts of the dimensions. In the case of 2D space, the equation in its simplest form is as follows”:

****

**Source:** Intro to image classification with KNN [13]

Similarly;

**https://cdn-images-1.medium.com/max/1000/1*2_UHWSD4gNMoRK827l8f9g.png**

**Source:** Intro to image classification with KNN [13]

Therefore the last equation is normally for calculating how far one point is from the other on a Cartesian plane. Knowing the results generated by these equations will immensely benefit our computer program, i.e. possessing such tool will provide us with much-needed assistance as we proceed with the algorithm.

* Compute the nearest neighbours for the new instance
* Arrange all the nearest neighbours (distances)
* Choose the most relevant k nearest neighbour distances
* Implement the voting process, the majority will decide which class the new image belongs to

Now that we have some idea about the K-Nearest Neighbor algorithm, we will now try to implement it on our chosen dataset; Fashion MNIST.

### KNN Implementation

As mentioned earlier, KNN implementation is done in Python with the assistance of the Tensorflow libraries using the Fashion MNIST dataset. The dataset can be obtained from Kaggle.com.

Note: Implementation was done using the Google collaboration platform

### Similarity Metric

The Euclidean metric as stated above is the most commonly used closeness metric when it comes to calculating the value of the nearest neighbors to any new assigned instance. However, we may need to consider another distance metric as a backup just in case our classifier may encounter technical difficulties due to our training data being very large (60,000 images of 28 \*28) and our 10,000 images in the test set.

Hence, L0 distance metric is deployed next on a set of modified images where 1 is assigned to each image with non-zero pixels and 0 for each image with zero pixels. i.e we then deploy the emetrics on this and the results are as shown. This approach may not be as popular but it does, however, falls in line with the purpose of this project. Normalization is done by generating 255 random values between 0 and 1with respect to the lightness of each pixel.

### K-Value Insight

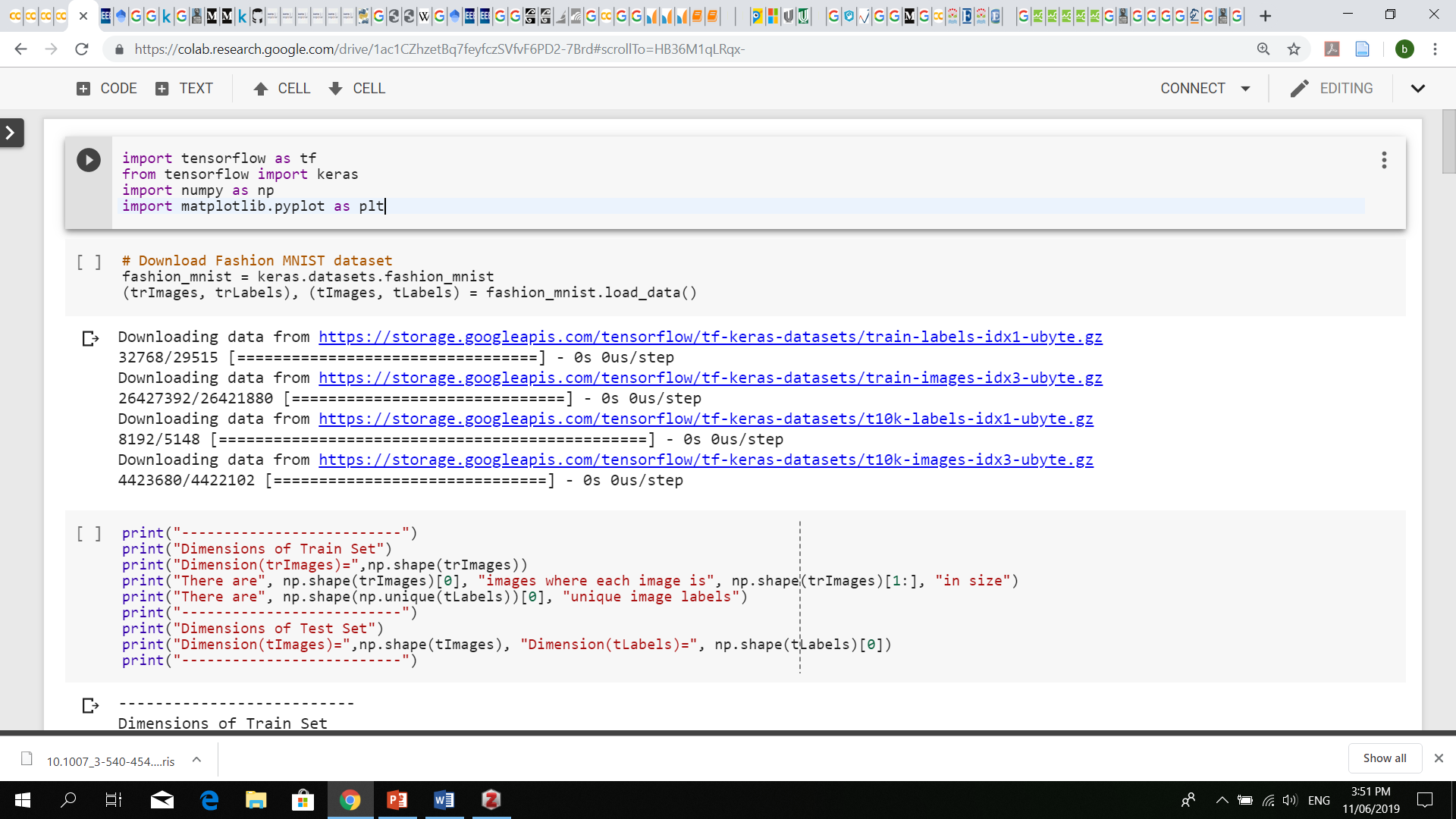
When choosing the value of k, consider:

* For k = 1, this means that the test instance will be labeled based on a single, closest training image.
* For k = 60,000, this means that the test instance will be labeled based on our whole dataset and this raises concerns such as the possibility that the labeling will be biased towards a certain training data that is most dominant.
* Setting an uneven number for the k-value as this will eliminate the possibility of a tie when counting the votes.

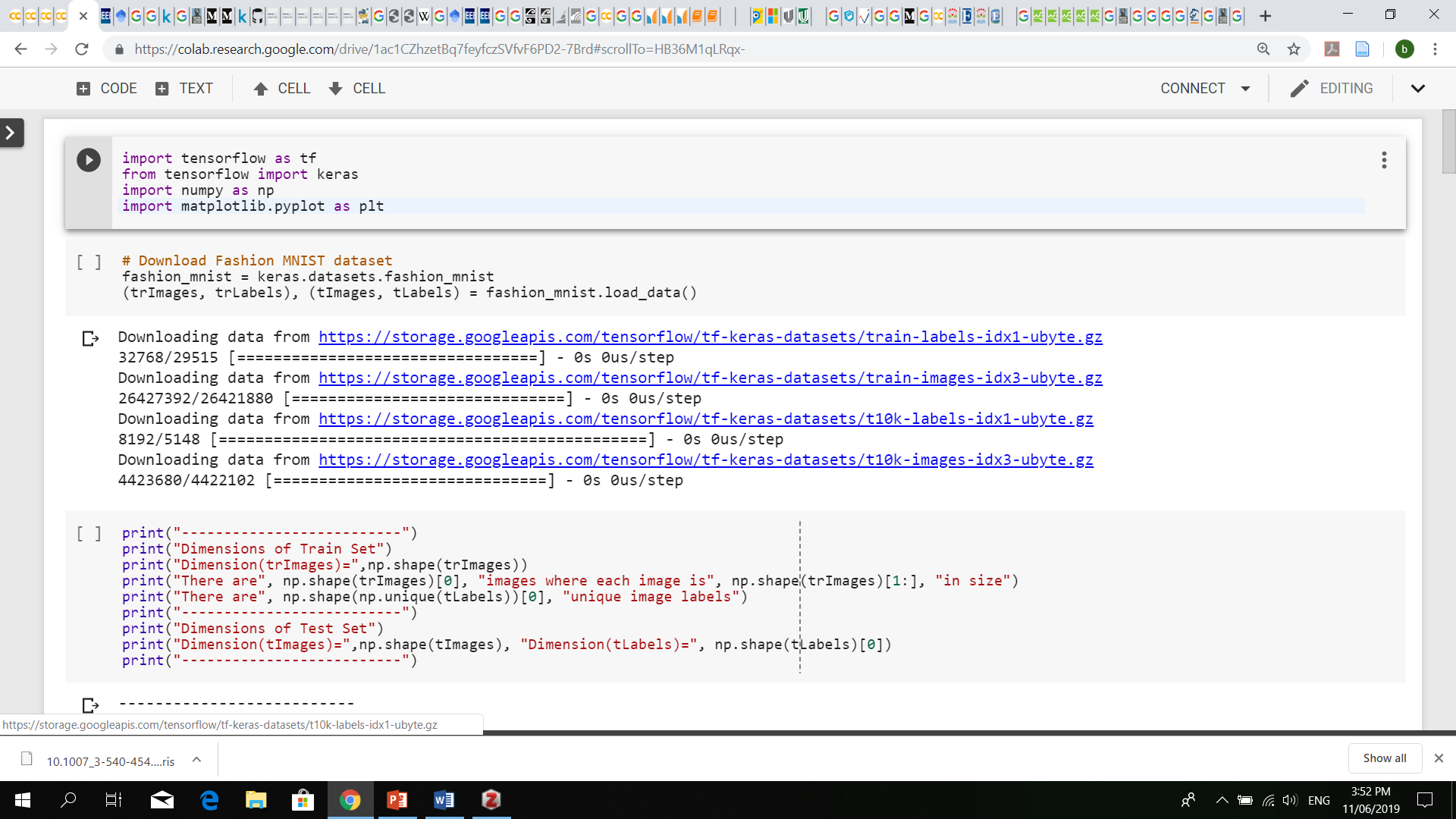
### Euclidean Distance Classification

For each of the 10,000 test images, 60,000 metrics has to be generated and that is a lot of computation which also takes up memory and it is a known drawback of this algorithm itself. Implementation is as follows;

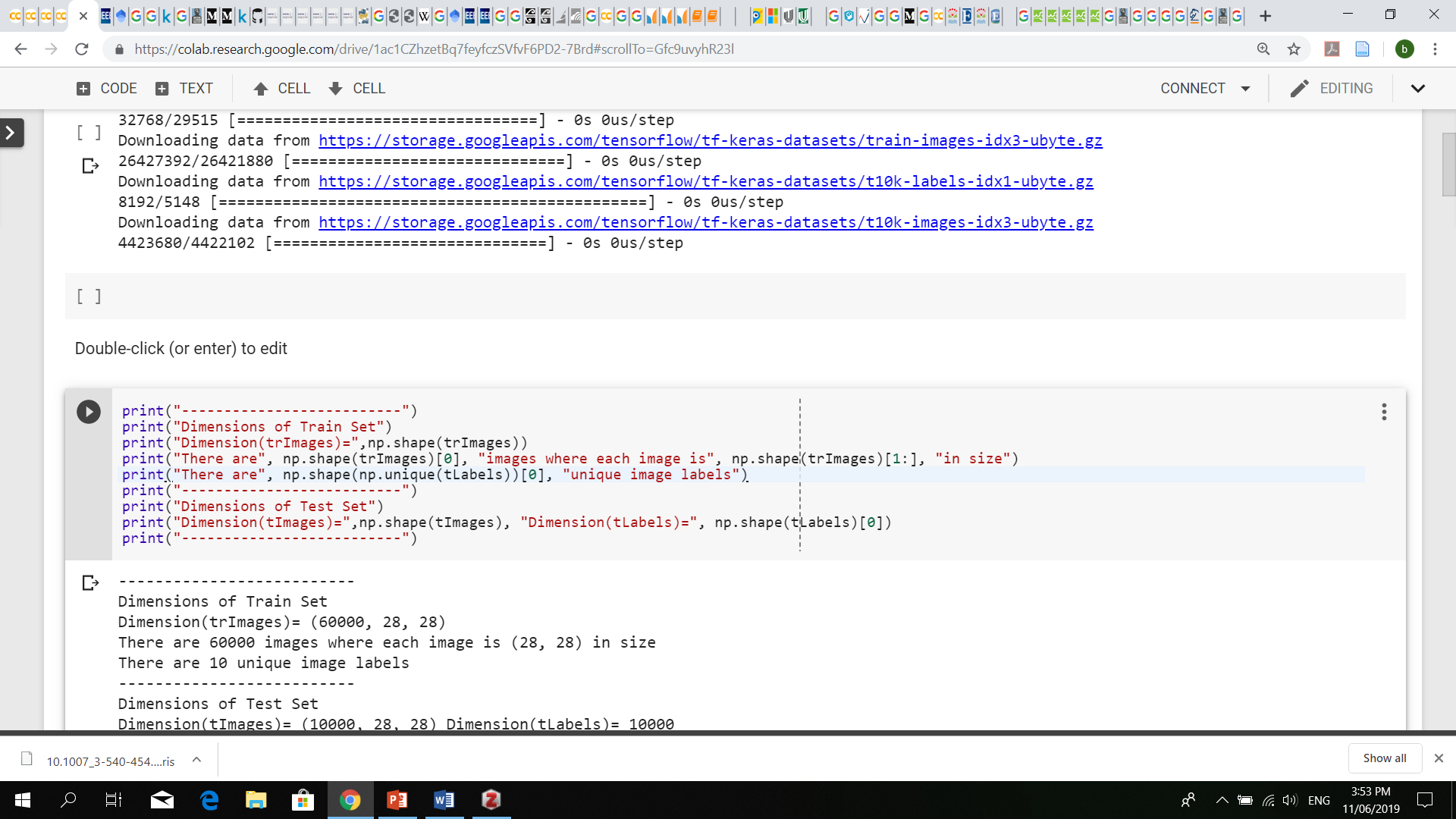
#### Importing the libraries



#### Loading the dataset



#### Viewing the dataset



#### Rest of the code here

Carefully set a value for **k,** load in both all of our train and testing images. Put all the **k** labels in an array for each of the test images (for loop 1). For distance (L2Norm), put them in an array for each training images (for loop 2). Deploy the metric and then append the array. Sort the distance and return the index and then along, indicating the first **k**-value and returning the count and label accordingly.

### L0 Distance Classification

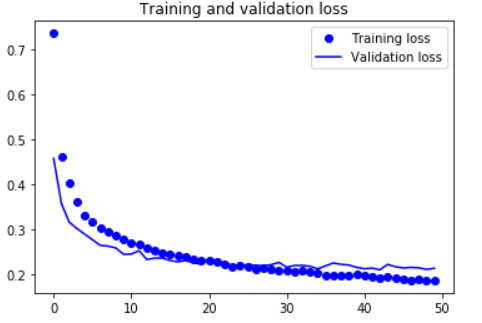
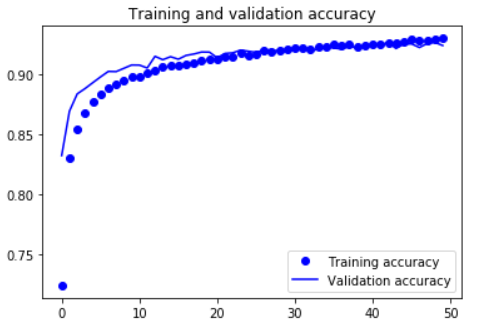
Here, the KNN implementation is still the same; a slight difference due to the images being modified i.e. each modifies images now consists pixels of zeros (0) and ones (1).

# Result and Discussion

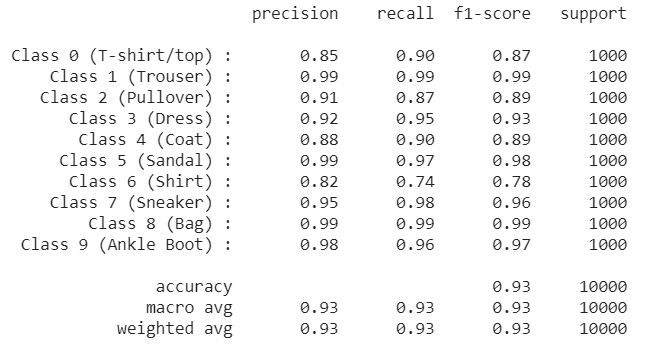
## Convolution Neural Network

At the first time, the network with multiple layers without Dropout layer, the test accuracy is around 0.91 with 50 epochs. However, when doing plotting between training vs validation accuracy and training vs validation loss. The validation accuracy does not improve after a few epochs and the validation lost increases steadily because of overfitting.

The result is improved significantly after adding some dropout layers into the training network with an accuracy of 0.93.

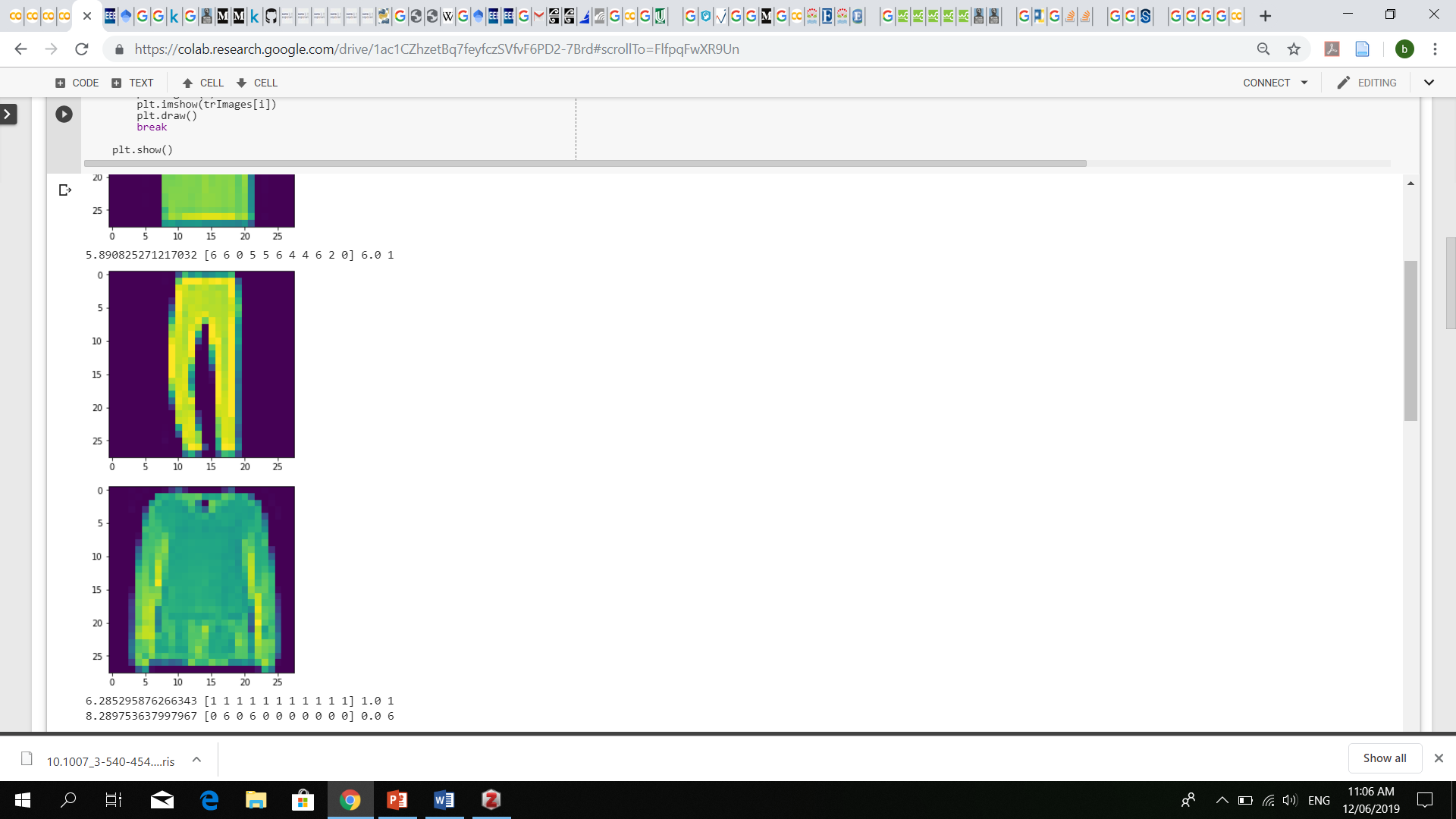
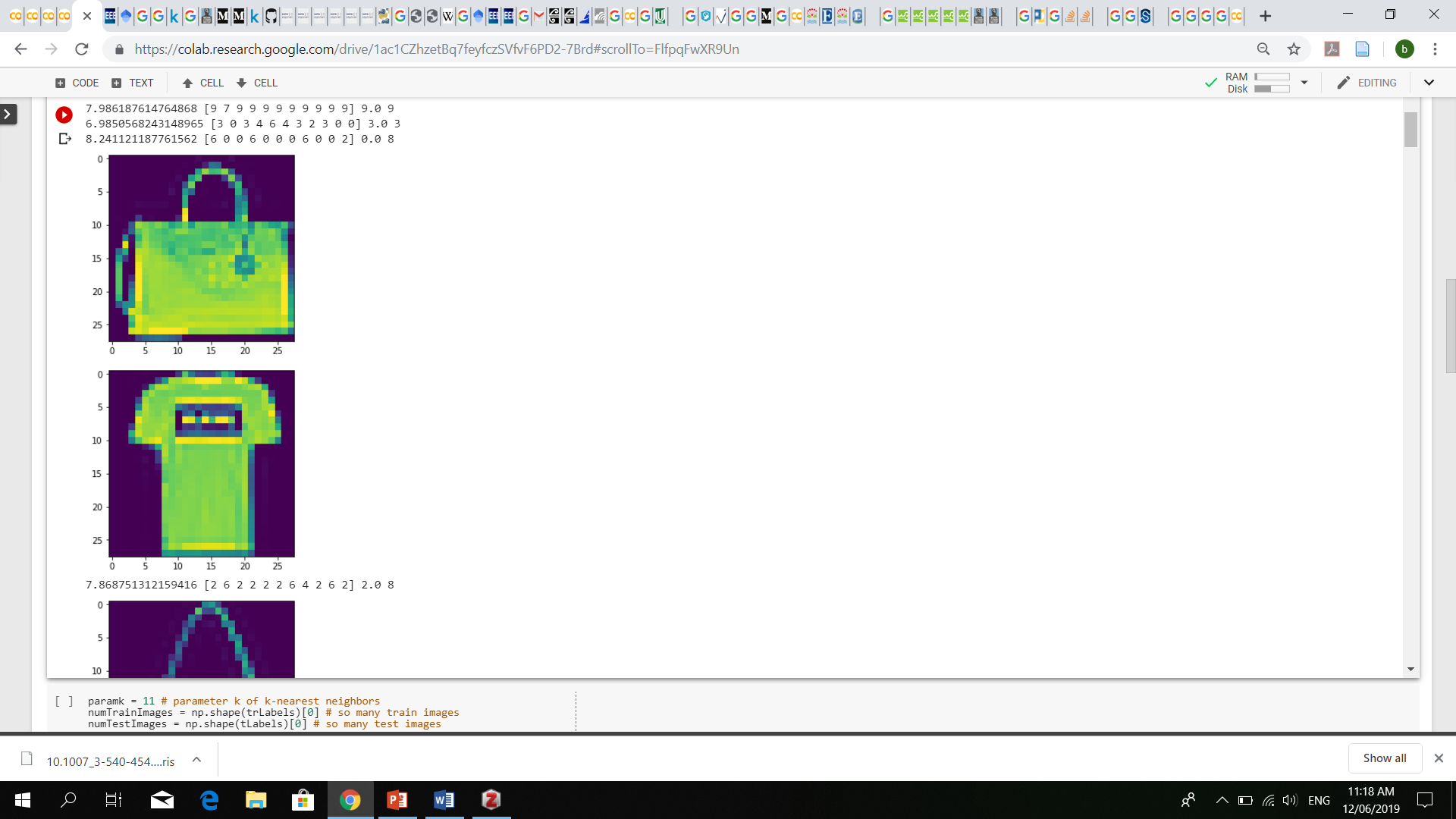


Based on the table of plotting accuracy of 10 classes. Class 1 (Trouser), Class 5(Sandal) and Class 8 (Bag) have the highest score of 0.99. Following that, other classes have a percentage of more than 85 percent except Class 6 (Shirt) is 0.82 as the lowest prediction. These incorrect items are more often between Shirt vs Pullover, Pullover vs Coat and Coat vs Shirt.



## K-Nearest Neighbor

For the *Euclidean distance* *classification metric*, the results are as shown below;

* Here, a bag is being labelled as a t-shirt
* Here we can see that a trouser has been classified as a pull over

Based on the images shown above, it is obvious that the Euclidean distance metric did a very poor job as a classifier.

For the *L0 distance classification metric*, the results are as shown below;





Both of the images above have been labeled correctly. The L0 distance metric (modified images being used) has done good work in labeling with a classification accuracy of 83.6% which also means that our classification is 16.4% not accurate.

A few take away from KNN algorithm is that the reason why the Euclidean distance metric did not work is because of the complexity of the computations i.e. all of the 10,000 test images needed to have 60,000 metrics for each one. Also which proves that KNN is indeed a lazy learning algorithm. The L0 distance metric did a better job because the complexity of the KNN was somehow reduced when the pixels for each image (both train and test) were modified. Although it may have been done in an indirect way but it did the job for the classification. In order for the KNN algorithm to work, it needs to be given a metric that will do the job based on the given problem.

# References

|  |  |
| --- | --- |
| [1] | Hongping Fu, Zhendong Niu, Chunxia Zhang, Jing Ma and Jie Chen, „Visual Cortex Inspired CNN Model for Feature Construction in Text Analysis,“ Front. Comput. Neurosci., 14 July 2016. |
| [2] | Z. Research, „Fashion MNIST,“ [Online]. Available: https://www.kaggle.com/zalando-research/fashionmnist. |
| [3] | Daniel Svozil, Vladimir KvasniEka, JiE Pospichal, „Introduction to multi-layer feed-forward neural networks,“ Department of Analytical Chemistry, Faculty of Science, Charles University, Albertov 2030, Prague, (72-12840, Czech Republic, 6 June 1997 . |
| [4] | Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, „Gradient-Based Learning Applied to Document Recognition,“ Proc. of the IEEE, November 1998. |
| [5] | Matthew D. Zeiler, Rob Fergus, „Visualizing and Understanding Convolutional Networks,“ arXiv:1311.2901v3 [cs.CV] 28 Nov 2013. |
| [6] | A. F. M. Agarap, „Deep Learning using Rectified Linear Units (ReLU),“ arXiv:1803.08375v2 [cs.NE] 7 Feb 2019. |
| [7] | Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, „ImageNet Classification with Deep Convolutional,“ Advances in Neural Information Processing Systems 25 (NIPS 2012). |
| [8] | Nitish Srivastava, Geoffrey Hinton,Alex Krizhevsky, Alex Krizhevsky,Ruslan Salakhutdinov, „Dropout: A Simple Way to Prevent Neural Networks from,“ 15(Jun):1929−1958, 2014. . |
| [9] | Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, „Activation Functions: Comparison of Trends in Practice and Research for Deep Learning,“ arXiv:1811.03378v1 [cs.LG] 8 Nov 2018. |
| [10] | T. M. Mitchell, „Machine Learning,“ McGraw-Hill PubCo, New York, 1997. |
| [11] | A. K. Nandi, „Automatic Modulation Classification Principles, Algorithms, and Applications,“ Hoboken: Wiley, 2015. |
| [12] | Praveen S Thakur, Sushila Madan, Mamta Madan, „Trends in Automatic Modulation Classification,“ International Journal of Advanced Research in Computer Engineering & Technology, February 2015. |
| [13] | A. Goswami, „Intro to image classification with KNN,“ Akash Goswami, 09-Feb-2018. |
| [14] | Karel Lenc and Andrea Vedaldi, „Understanding image representations by measuring their equivariance and equivalence,“ arXiv:1411.5908v2 [cs.CV], 20 Jun 2015. |