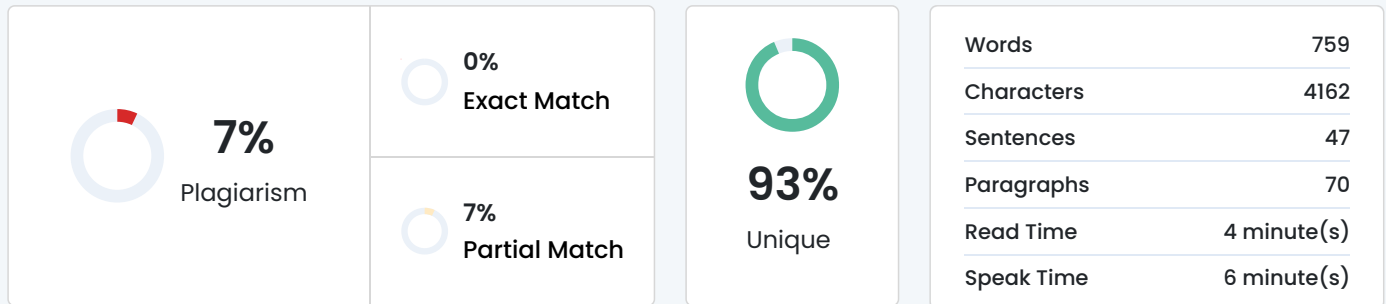


## Plagiarism Scan Report



## Content Checked For Plagiarism

Where,  $W$  is the current value of the weight, and  $\alpha$  is the learning rate. Learning rate is the amount in which direction we move the weight. Choosing the correct learning rate is necessary for getting the minimum cost efficiently and correctly. If it is too high the optimization will take a very long time, having to do this for every weight will get very computationally expensive. Having it too high will cause the weights to jump around a lot, this causes it to never settle for the minima. This is show in Figure 7

This project uses Adaptive Gradient Descent (AdaGrad) which adaptively changes learning rate. This means that rather than being constant like we saw previously they are updated based sum of previous gradients squared.

The project uses TensorFlow to create and train the neural network. The model is trained on the Google Colab T4 (Tesla T4) compute unit. The model is trained on cloud and is then downloaded to be used in offline project. The following imports are used to train the model

We process the dataset using sobel filter and save the results in a csv file in the our drive, this is then loaded into the cloud. We also **split the dataset into training and testing data.**

After the dataset is loaded, we will make our model. This model has three layers, input layer has 16,384 neurons and uses the ReLU activation function. The second layer has 8,174 neurons and also uses the ReLU activation function. This hidden layer has the number of neurons chosen to be between neurons in input layer and those in output layer. The output layer is a layer with 36 neurons. This is built in tensorflow as

To train this model we need to call the compile and fit methods as shown

Now, we can check the accuracy of our model using the testing data. The output for this is shown in Figure 8.

The common way to show the accuracy of a model is to use a confusion matrix. It has the true labels in the Y-axis and predicted label in X-axis and data is shown in form of a matrix. Each cell shows the percent of time the model makes the prediction for the given label corresponding true label. Thus if model is has high accuracy, the diagonal will have should be high. The confusion matrix of our model is shown in Figure 9. We can see that our model shows confusion for 'U', '7', '6' and '4' but otherwise has very high accuracy.

The final component of this project is the OpenCV application which will take the input from the user. It takes the video from webcam as the input. It will then pass the input through our sobel filter. It will finally use the model which we trained to give a prediction. We can predict using the `$model.predict$` method. The result of this function is shown in Figure 10. This returns an array of predictions, where every index represents a different label. The index with the largest signal is the label which our model predicted.

The first step is to setup the openCV loop. This loop is called every frame of the video

Now, we need to get a square region in the middle of the frame where we can get the input. Thus we will need some global variables that will define the region

Now, in our main loop we will add the following line to put the region in our frame after the `cv2.putText`

Finally, in our we can get the region after the `cv2.imshow` call to get the region and write it to an output.jpg

After we have written our output image, we can then give it to the sobel filter. To run other processes, we use the python subprocess module.

Since we now have data in form of numpy array, we can pass the values to the model to get our predictions and change the current predictions accordingly.

In this chapter, we will present the results we got in this project in brief. The sobel filter is the first major component of this project. Since this is written in C++, this is very fast and computationally efficient. It is also very portable because the main logic is written in pure C++ and library to read and write images are the stb single file libraries (6) which are very small and cross platform. The filter being implemented in a system programming language also means it is easy to integrate with other languages.

## Matched Source

**Similarity 2%**

**Title:**[Model Complexity & Overfitting in Machine Learning | GeeksforGeeks](#)

Split the dataset into training and testing Data: Splitting your dataset is crucial because it ensures the model doesn't simply memorize the training data and can generalize to unseen examples.Feb 28, 2024

<https://www.geeksforgeeks.org/model-complexity-overfitting-in-machine-learning>

---

**Similarity 2%**

**Title:**[Confusion matrix \(X-axis represents the predicted label and Y-axis...](#)

Confusion matrix (X-axis represents the predicted label and Y-axis denotes true labels): (a) ResNet-50; (b) DenseNet-121; and (c) EfficientNet-B6.  Schematic...

[https://www.researchgate.net/figure/Confusion-matrix-X-axis-represents-the-predicted-label-and-Y-axis-denotes-true-labels\\_fig7\\_383713607](https://www.researchgate.net/figure/Confusion-matrix-X-axis-represents-the-predicted-label-and-Y-axis-denotes-true-labels_fig7_383713607)

---

**Similarity 2%**

**Title:**[Loop for adding user input to creating objects \(C#\)](#)


Jun 5, 2019 · I'm setting up a console application that will take the input from the user and make a new object with my class "Items". But I want a loop there as well so the user can do multiple inputs, what kind of loop should I use and how does it works if I want the variable "foo" to change through the loop?

<https://stackoverflow.com/questions/56459397/loop-for-adding-user-input-to-creating-objectsc>

---

**Similarity 2%**

**Title:**[CSCI 250 Lab #2: Introducing OpenCV - Simon D. Levy](#)

The goal of this lab is to get familiar with OpenCV, the popular computer-vision package that is the second major technology that you will use in your final...

<https://simondlevy.academic.wlu.edu/csci-250-introducing-opencv>

---