Arab Open University- Egypt

Faculty of Computer Studies

Information Technology and Computing Department

SIGN LANGUAGE GESTURE RECOGNITION

Shady Mahmoud Ahmed Elkholy – 1751710378

TM471: Final Year Project, December 2021

Supervisor: DR. Sara Nabil

Faculty of Computer Studies

TM471 Project

Sign Language gesture recognition

Abstract:

One of the fastest-growing areas of study is sign language gesture recognition. Many techniques and tools have been developed in this area and a lot of them are getting polished. Sign language is mostly used by the hearing impaired.

There’s always a challenge for non-sign language speakers to be able to communicate with deaf and mute people. This project is underdevelopment to help translate American Sign Language to text through hand gesture recognition. This project aims to make it easier for the hearing impaired to communicate with a wider range of people especially those who don’t know American Sign Language. In this project we aim to start converting sign language gestures to text to make it easier to communicate.

Acknowledgements:

I would like to thank my mentor **Dr. Sara Nabil** for giving me the opportunity to undertake this project. I would like to thank her for her immense guidance and appreciate her timely engagement.

I would also like to give my thanks to friends and family for helping me complete this project.

Special thanks to my friends: Hazem Salah, Mahmoud Gamal, Mohaned Dawood, Zyad Samy who I kept up some nights with me trying to de-bug different parts of the code and trying to find out where the problem might be, although most of the time we didn’t find any problems together and I ended up finding it a few days later multiples I still would like to thank them for mentally being there for me because it’s the thought that counts.

I would also like to thank my mother who always made and brought me food while I was working on the project so I would use all the time I can to make this happen. I would also like to thank my dog for accompanying me while using the bathroom and making it truly a stress-relief time.

Contents:

Figures Table:

Chapter 1: Introduction:

For people with hearing impairment, sign language is the most common and expressive way to communicate with other people. Those who are not deaf almost never attempt to learn sign language just in order to communicate with people with hearing impairment.

And as a result of this hearing-impaired people become isolated. That's why trained sign language interpreters are hired for legal, medical appointments and other sensitive meetings. (Figure 1).

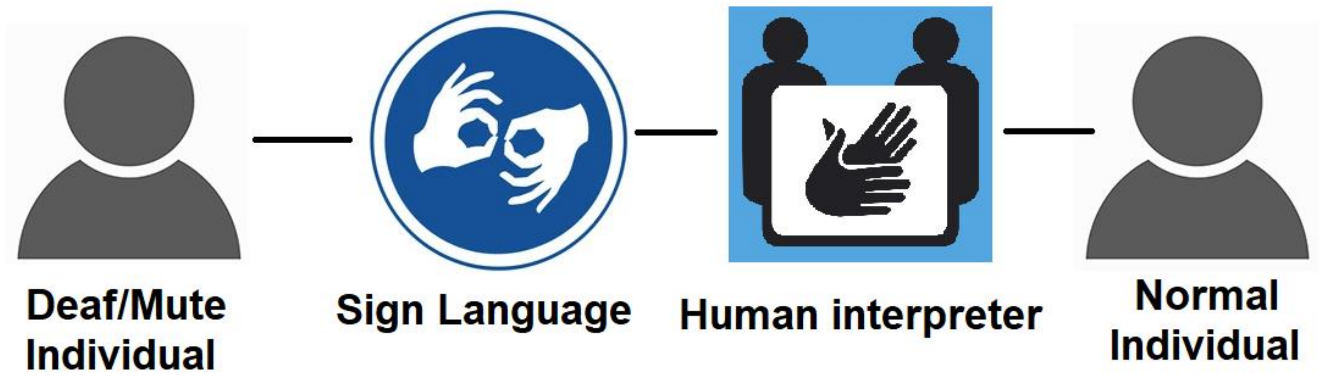


Figure 1: Present Scenario

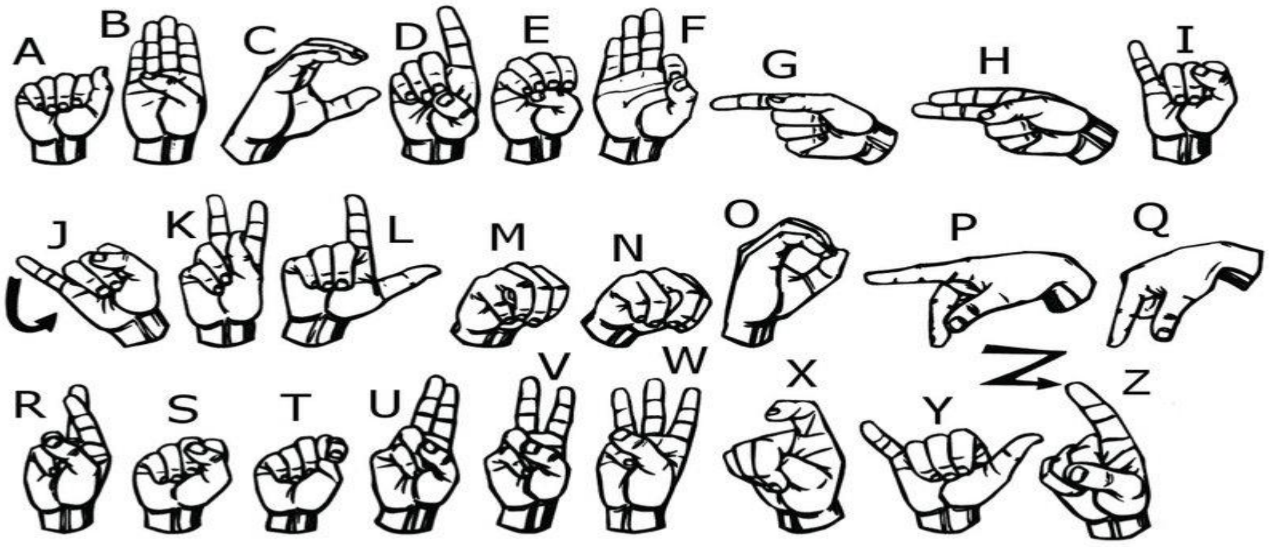
But if a computer can be used as a translator that can translate sign language to text or even speech it will be cheaper than having to hire an interpreter and also make it easier for the hearing impaired to communicate with more people overall.

To address this, we can use a computer program that takes a camera feed and translates the sign language to text. A real time sign language recognition system is hard to achieve but possible and can be implemented for usage. There are different approaches to sign language recognition the most common two are:

-Vision based approach:
Uses a video capture device to capture images and feed it to the program and capture the gestures from the images.

-Glove based approach:
Uses sensors on a glove to capture the hand gestures physically and send the data to the program to decode the movement and capture the gestures.

Since this project uses the vision-based approach, we will go into its details in latter chapters. The system should capture the sign language through a video capture device like a webcam and then the images are processed and when the computer recognizes a gesture it outputs as a text on a monitor and where there is a full sentence or a certain gesture was made it will output it as text-to-speech.



Motivation:

My motivation for doing this project is helping people struggling every day to conduct their normal daily lives by implementing a system that helps them communicate easier with people who don't use sign language. And since there is not many options out there available to help them, I think this is a great project to be conducting especially since good interpreters may not usually be available and if they were it wouldn't be cheap or efficient for individuals or corporations to hire them.

Especially considering the other option is pen and paper which is not a great idea, it is not comfortable, it is messy and it is time consuming both for the deaf person and to the hearing person. With technologies taking over all the industries, the demand for a machine learning application that solves this issue is getting higher every year.

Although the accuracy and efficiency of the algorithm and the model developed in the app must be pretty high to make it useable, I think that early prototypes are not bad and we are on the right direction.

The main point is that I would like to shorten the gap between the hearing person and the hearing-impaired person by using a sign language translation system.

Chapter 2: Related Work:

Sign language recognition is not a new computer vision problem There were many papers talking about the subject for the last 2 decades. Huge advancements and research have been done on hand gesture recognition in the past few years.

There are a few apps available on the market that translate sign language into text, but almost all of them are somewhat slow. The deaf and mute can use texting apps that read out their text to communicate but by using those apps the problem is still not solved, that's why some developers developed apps that can recognize hand gestures and translate it to text.

And in the University of California, San Diego, July 2017, a glove which uses sensors to recognize hand gestures have successfully converted the 26 letters of American Sign Language (ASL) into text. But it could only interpret those 26 letters of the alphabet. (O'Connor, et al., 2017)

There is also a smartphone app called GnoSys made by a start-up based in the Netherlands called Evalk that uses computer vision (smartphone's camera) and neural networks to recognize the sign language gestures being made in the video feed and translates it into speech. But the app is still not available for public use. (NewzHook, 2018)

In a paper by (Huang, et al., 2018), the authors recognized that the problem in sign language recognition is when the signs are broken down to words and that's an issue with continuous sign language recognition.

The way they decided to solve the problem is by using a new framework called Latent Space Hierarchical Attention Network (LS-HAN) which gets rid of the need to preprocess the temporal segmentation.

The framework uses a two-stream CNN for video feature representation generation which is very complicated and very hard to accomplish.

The main objective of this project is to design a software that will help the deaf and mute to be less alienated in the workplace by removing the communication barriers between the disabled and the abled.

The system is intended to work as an interpreter. Which can be used in businesses or for personal use.

This project aims to interpret continuous sign language and make the output as smooth as it can and make it as user friendly as possible so it can be used in businesses to make it cheaper to hire deaf or mute individual instead of using an expensive human interpreter. (Figure 2)

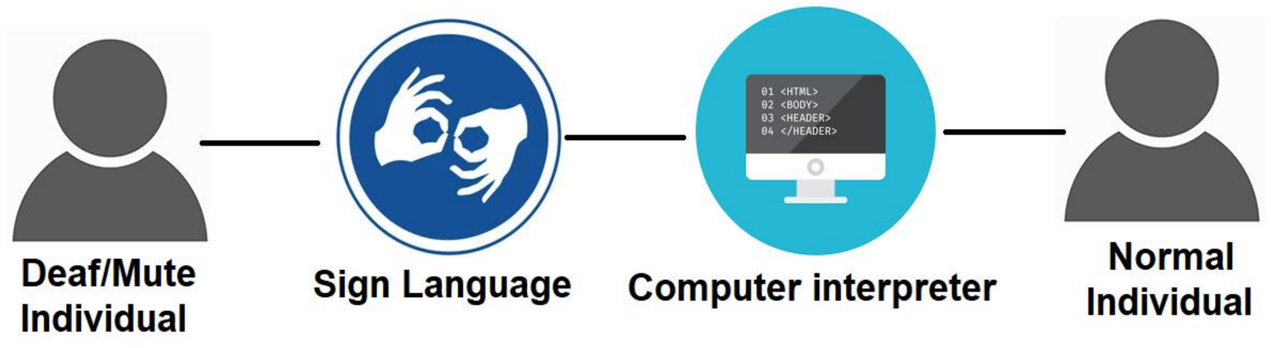


Figure 2: Proposed Situation

Chapter 3: Requirements and analysis:

The vision-based system takes images from a camera in a specific frame then makes an image out of that frame and reduces its resolution to 200 by 200.The image is then used in the model. When the model is trained using the dataset formed from these images, it is then used to make predictions on the gestures and it predicts which gesture is on screen right now and then it puts the output as text on the screen. If we found that the predictions aren't as accurate as one might want, we then might retrain the model using a diverse dataset using the gestures of multiple people and multiple lighting conditions and redo the predictions again to and see if the accuracy of the predictions improved, this can be done multiple time until the desired output is achieved. It is also possible to do complete words not only individual characters. After the user has done the same gesture for 3 continuous seconds the system will save the letter to be put in a word. (Figure 3)

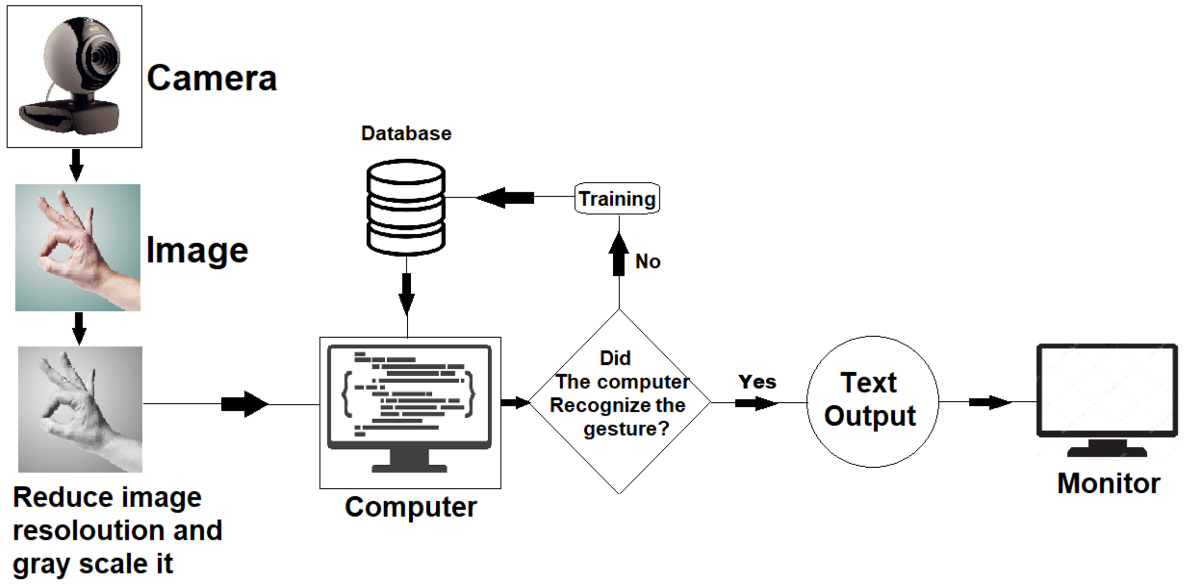

Camera
Image
Reduce image resoloution and gray scale it
Database
Training
No
Computer
Did The computer Recognize the gesture?
Yes
Text Output
Monitor

Figure 3: Block Diagram

The user's video feed should be picked up by the webcam and entered into the system to get the features of the image and make out a gesture, then we use neural networks to train that gesture into the database of the system.

We can then retry the same gesture and see if it's picked up and recognized accurately and correctly. If it's not the system is then retrained until it's recognized accurately and correctly.

Repeat that until we have trained all the gestures we need or going to use. Then when a user makes that gesture, it should get detected by the system and outputted as a text on screen after holding the same gesture for a few seconds. (Figure 4)

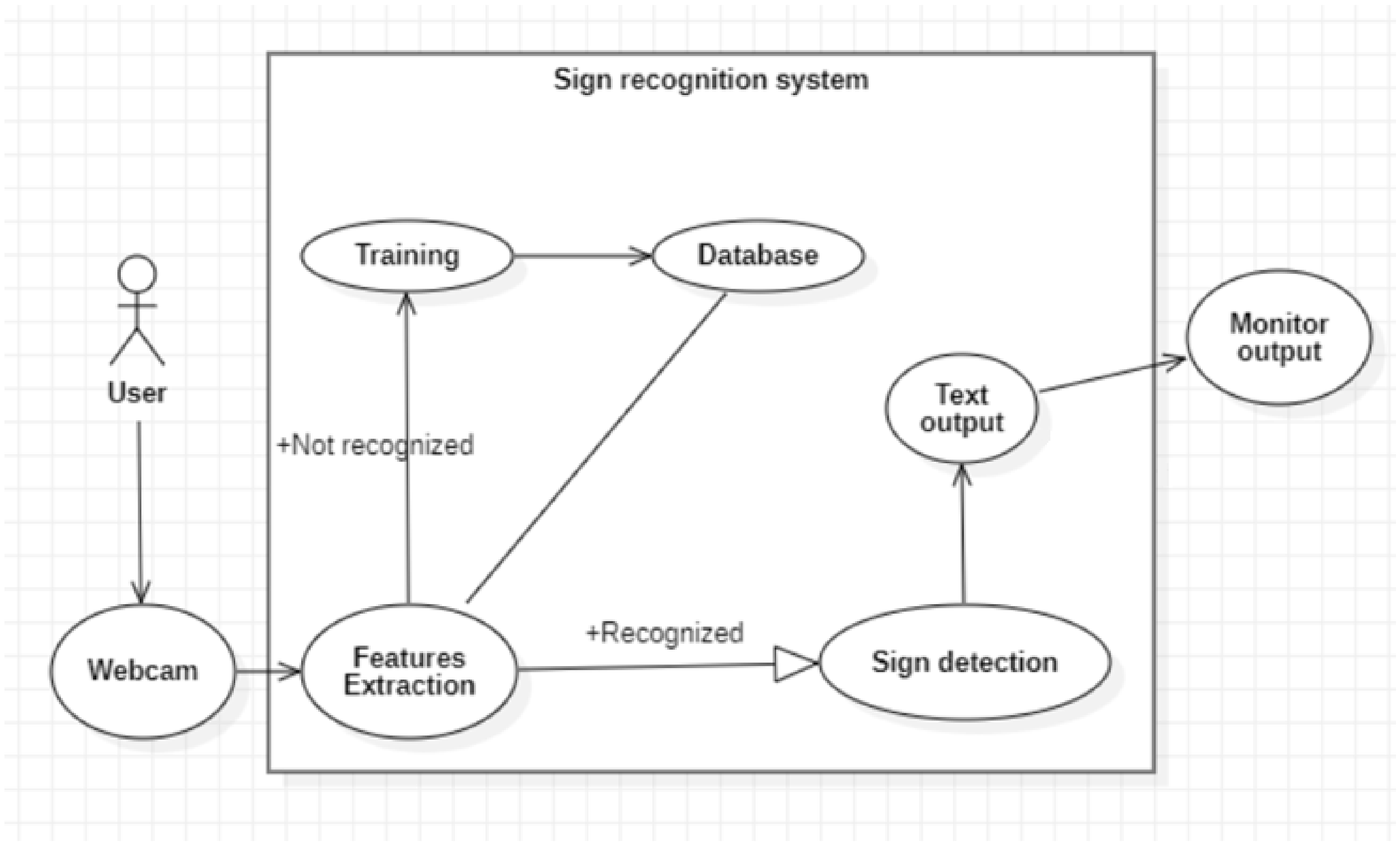


Figure 4: Use case diagram

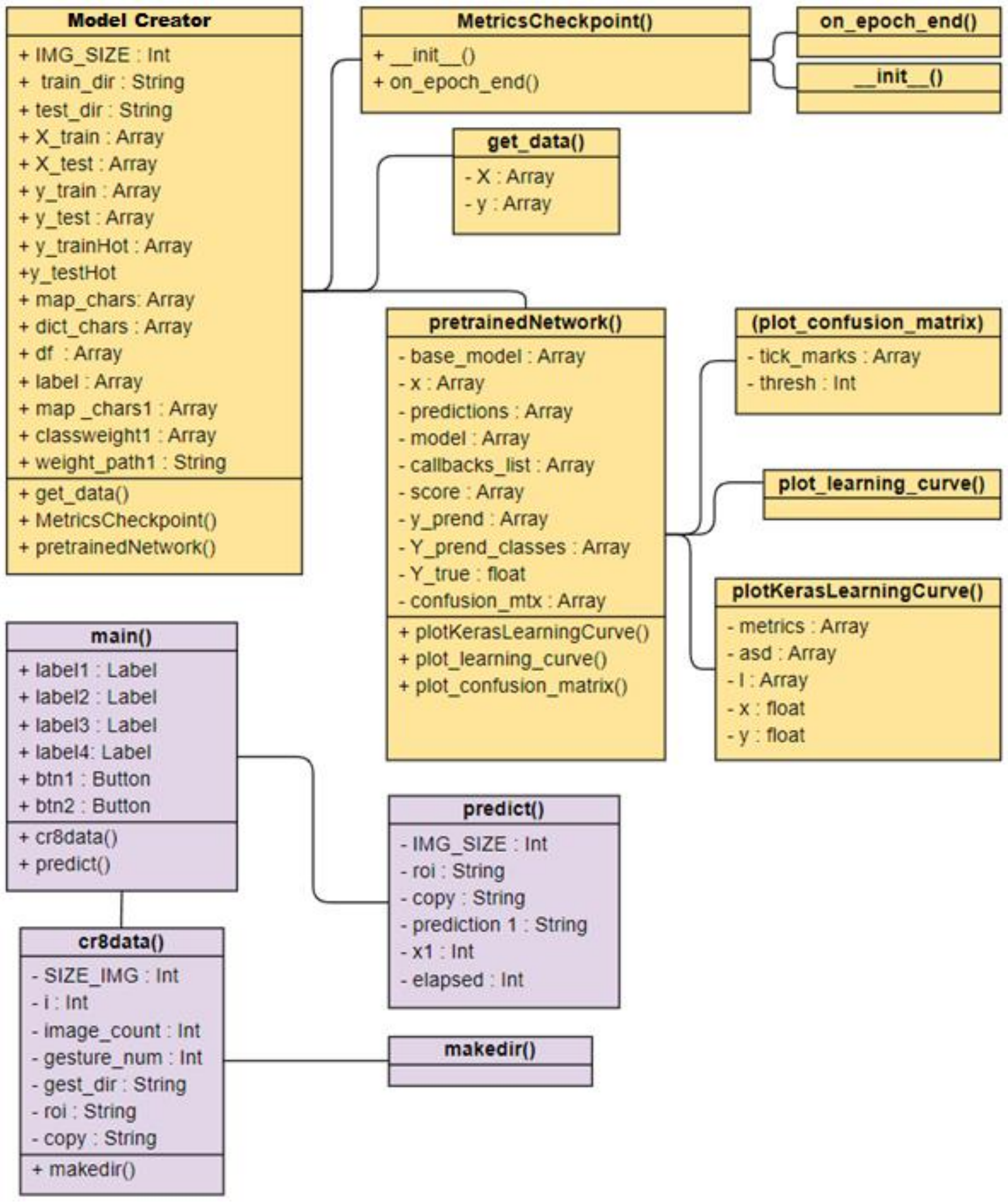
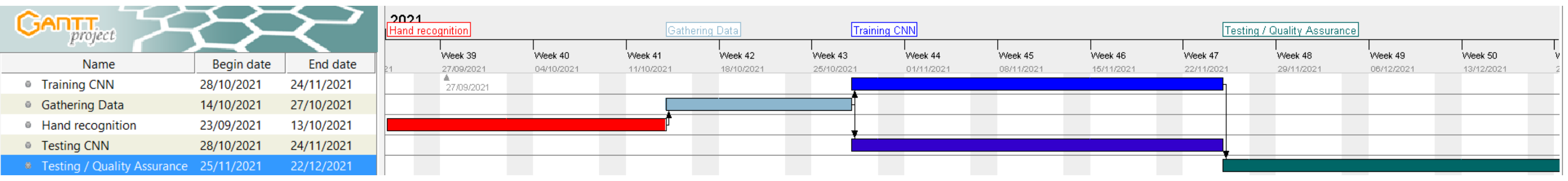


Figure 5: Class Diagram

This project should take around 12 – 14 weeks from start to finish including all the training, data gathering and testing. (Figure 6)



Name	Begin date	End date
Training CNN	28/10/2021	24/11/2021
Gathering Data	14/10/2021	27/10/2021
Hand recognition	23/09/2021	13/10/2021
Testing CNN	28/10/2021	24/11/2021
Testing / Quality Assurance	25/11/2021	22/12/2021

Figure 6: Timetable

Chapter 4: Design, Implementation and testing:

4.1: Model design:

Artificial Neural Networks (A. N. N.s) is consisted of units called neurons and these neurons are inspired by the biological design of neurons.

The neurons work together to model and translate input data and classify them into different classifications or labels. The weights associated with each node can be changed to make sure to minimize the error factor in training samples in the supervised training environment. Any neural network consists of layers that work on the input data.

(Figure 7)

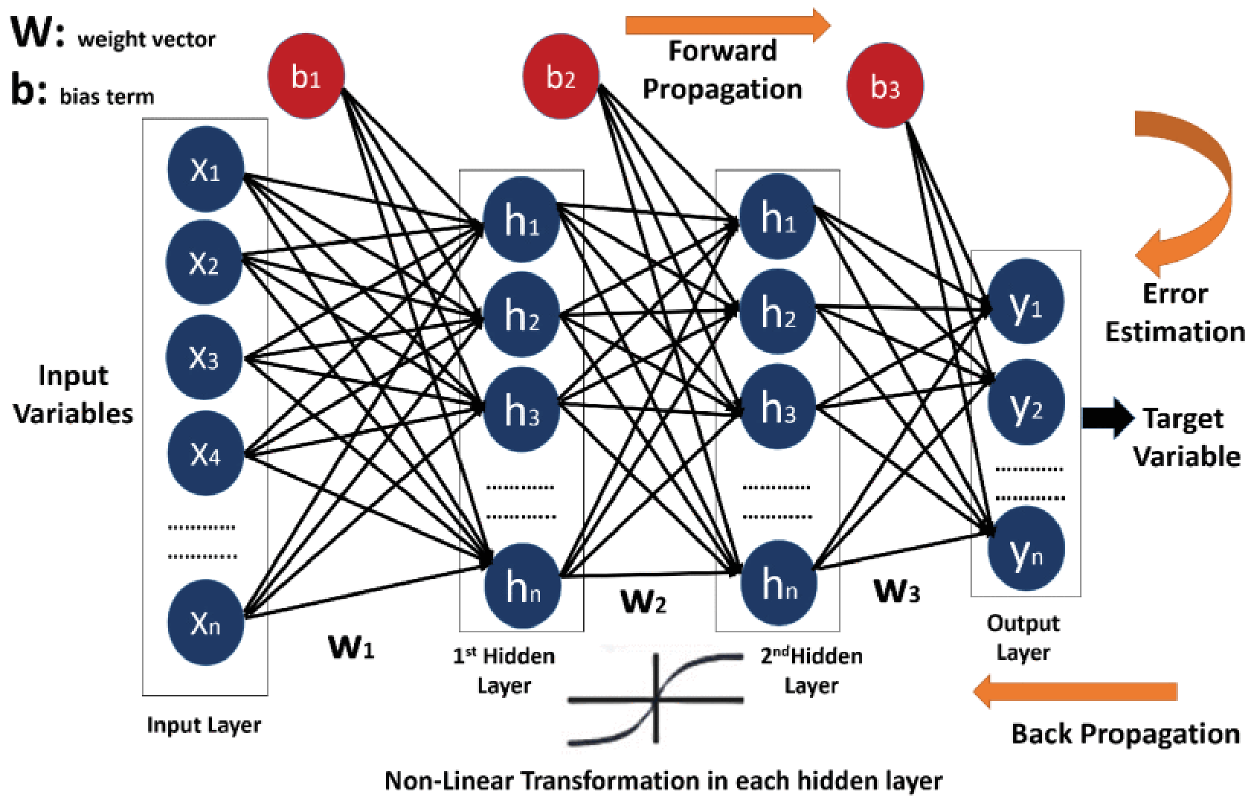


Figure 7: Neural network example

4.1.1: One-layer neural networks:

One of the simple neural networks is the single-layer neural network which as the name implies only consists of one layer, that layer consists of another two layers which are the input layer and the output layer. The single-layer neural network is considered the very first approach that is used to solve different foundations of the systems that are available on the market and being developed. The term “perceptron” was almost always used to describe these first single-layer neural networks. The perceptron would hold a value from the input type and it would perform a logical operation and a mathematical operation on it based on the input. It then would carry on to send a signal to the next neuron if the condition of its firing has been met. If that condition has not been met the neuron will not fire any signal to the next neuron. Hence, the feedforward neural networks are very efficient and carry information from the input layer to the output layer. This makes it very different from many of the other approaches. But a problem lies with it, the single-layer neural network can’t solve most problem because most of them need more layers to perform more logical operations. (Figure 8)

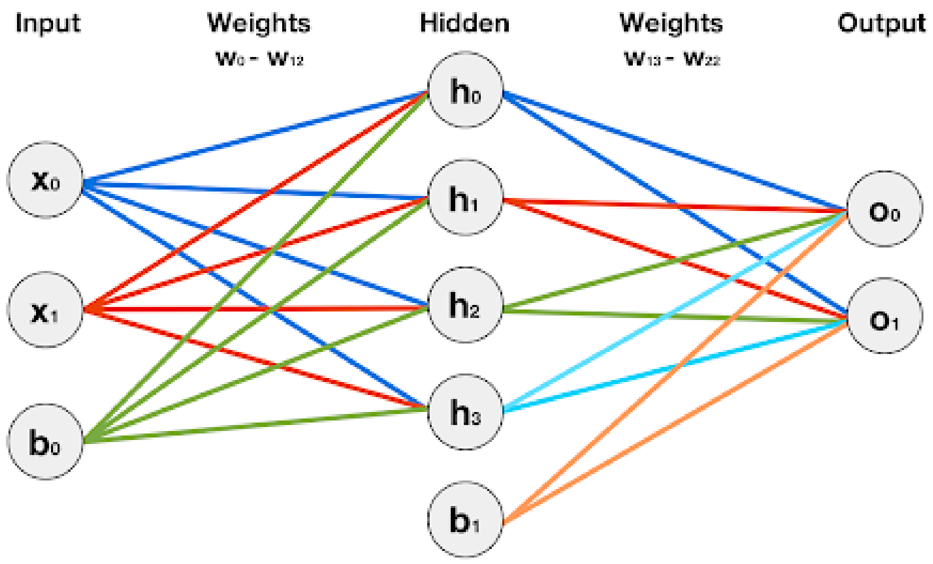


Figure 8: One-layer neural networks

4.1.2: Multi-layer neural networks:

The ability of a Neural Network (NN) to learn specific patterns is determined by the number of layers in the network. The NN becomes more sophisticated as more layers are added, requiring additional resources. Only linearly separable problems can be learned by a NN with a single active layer. An NN may construct convex areas in the data space with two active layers, which gives us the opportunity to perform patterns with many characteristics and produce a very fined output, it can split the data in any shape, which the multilayer perceptron should be able to solve any problem coming their way from recognizing dogs and cats to recognizing characters with very complex characteristics. (Figure 9) Deep neural networks are NNs that have more than three layers and need more computer resources to train. In comparison to a single fully connected layer, using an MLP with four layers can increase the complicated shapes to increase the size of filters that filter data allowing the network to work more efficiently That’s why we are choosing multi layered neural network for our project, specifically a five-layer neural network.

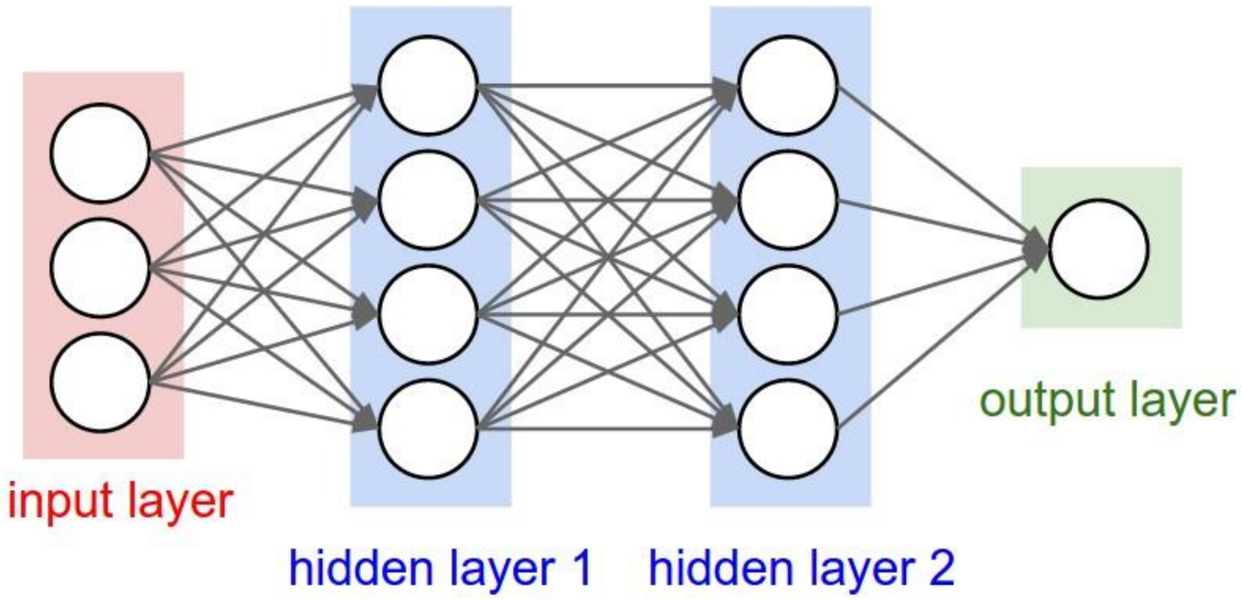


Figure 9: Multi-layer neural networks

4.2: Dataset:

The main objective of this software is to try and classify the gestures inserted into it and label those gestures using one of the labels already defined.

We then save those gestures as dataset and create a model using them. We then test that model using a prediction software to test if the software can recognize which gesture is being made.

The model is using transfer learning from an already made Keras model. Then the system must be trained on an already established dataset before making any predictions.

I've created a user interface which allows the user to make a new dataset using a single button. To make this dataset the user is asked to do a character gesture and hold that gesture using different angels and distance from the camera repeating that for each character the pictures are then saved with a resolution of 200 by 200 and saved in a folder which make up the dataset.

(Figure 10)

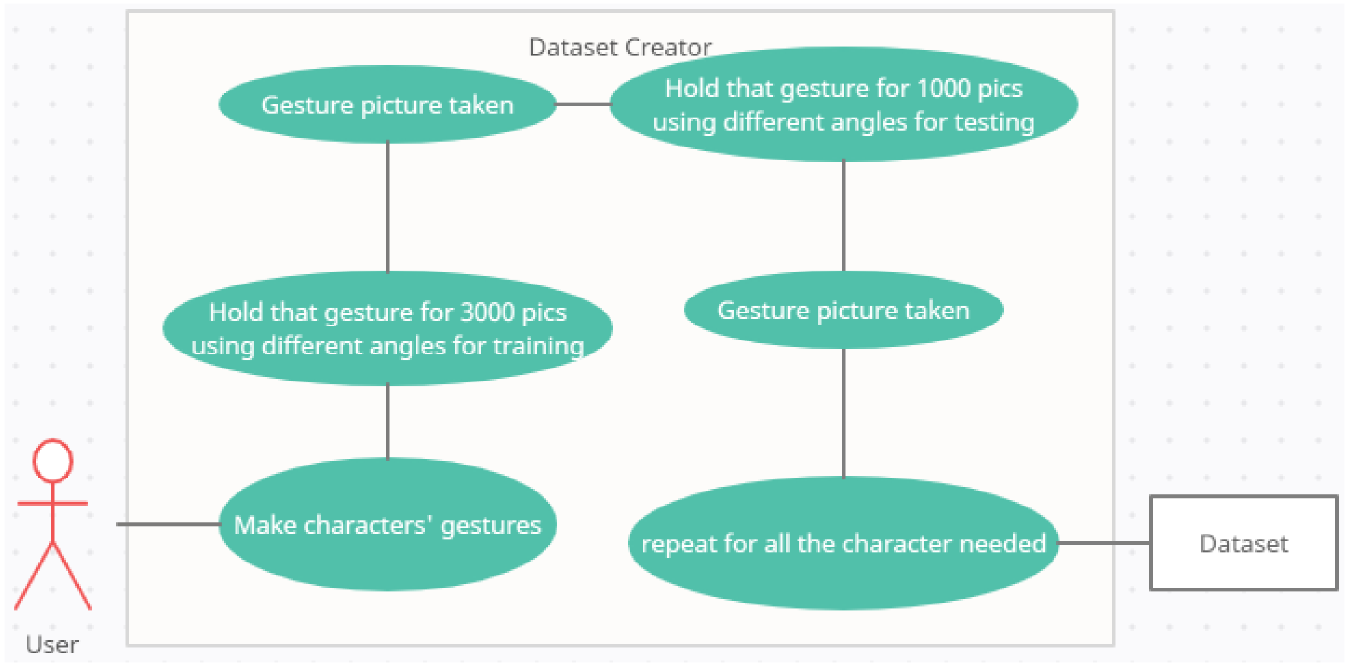

Dataset Creator
Gesture picture taken
Hold that gesture for 1000 pics using different angles for testing
Hold that gesture for 3000 pics using different angles for training
Gesture picture taken
Make characters' gestures
repeat for all the character needed
Dataset
User

Figure 10: Dataset creation

4.3: Labeling:

After establishing our dataset using that method, we might start training the model using that dataset. There's a total of 24 gestures for which the model is going to be trained to do, which is the alphabet minus the "J" and "Z" characters because they require a motion gesture. Each gesture has 4000 pictures taken of them, 3000 of which are for training and 1000 for validation testing. (Figure 10)

Then every image will be labeled with its corresponding label. So, if an "H" Character it will be in an "H" folder and will have "H" label and so on.

Then every character will be hot vectored and then shuffled for subsampling, this part is the pre-procession part which is one of the most important in any machine learning project. After shuffling the character with the hot vectored labels, we will find that they are not equal in numbers anymore because we lost some characters more than others. (Figure 11)

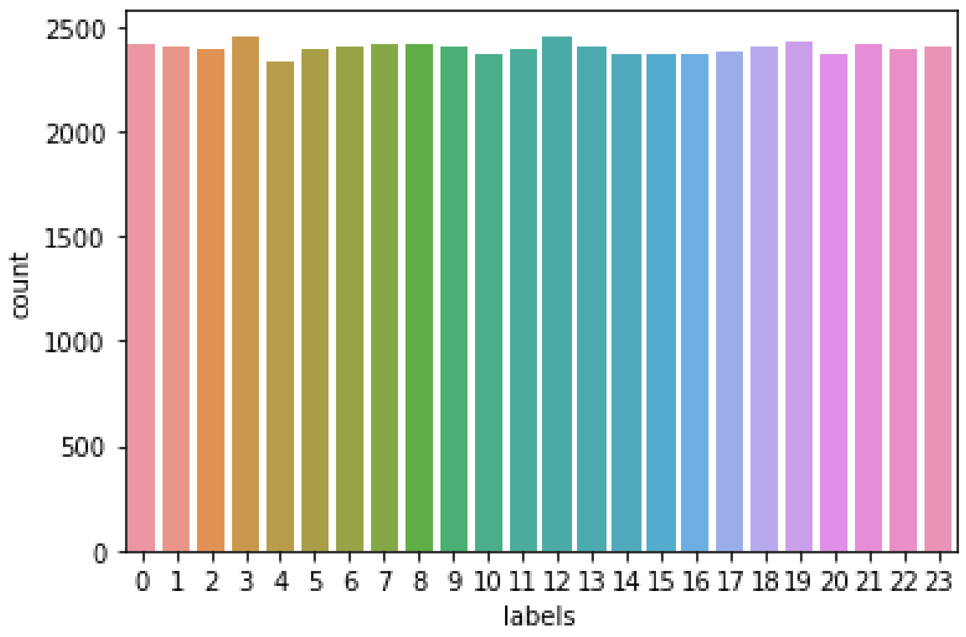
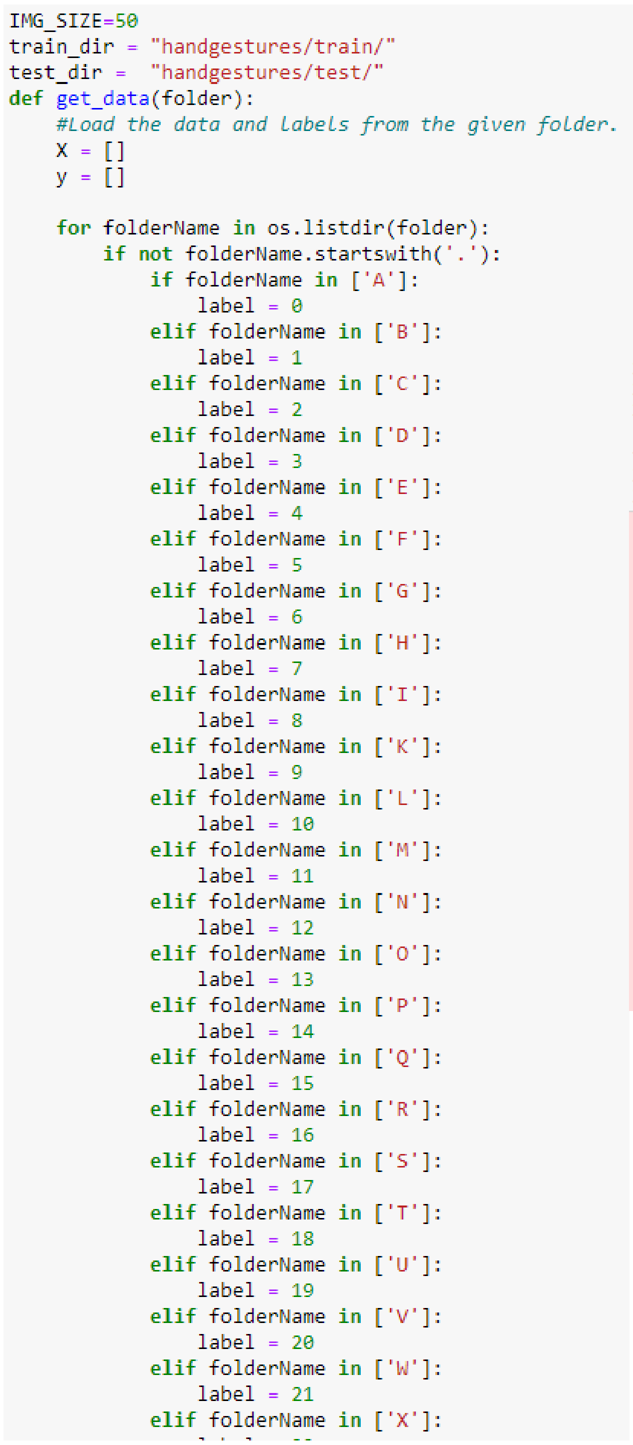
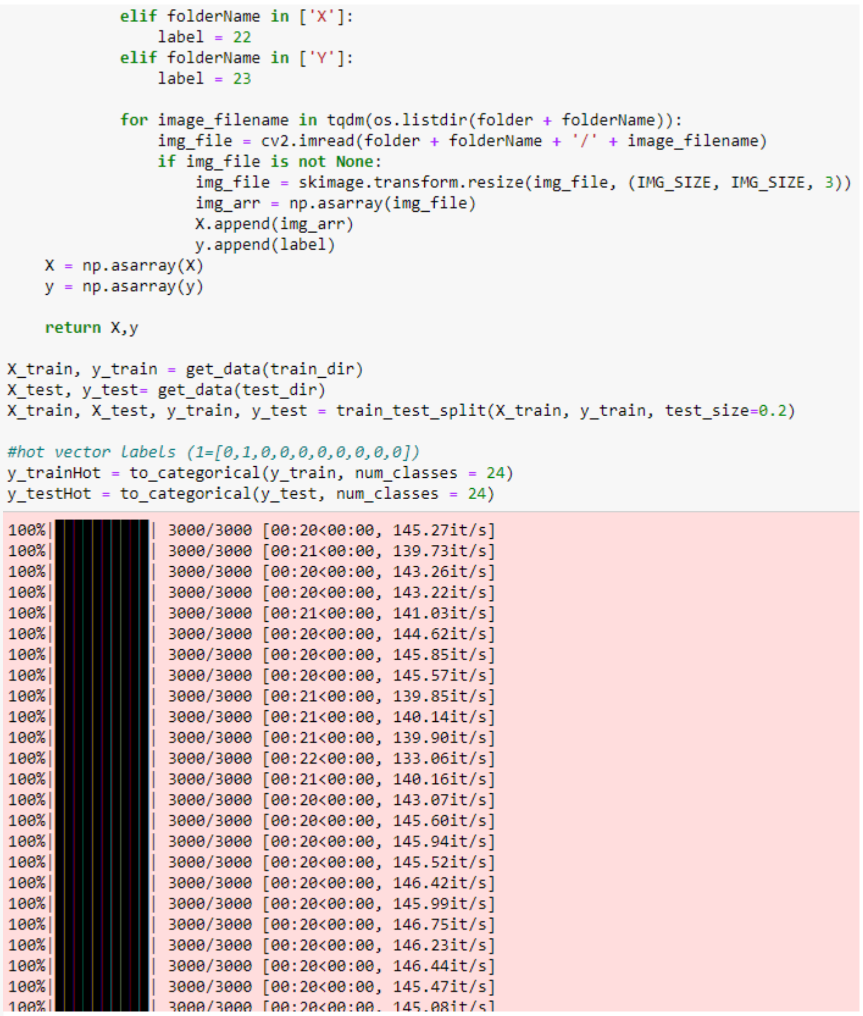


Figure 11: Labels

Labelling code screenshots:



```
IMG_SIZE=50
train_dir = "handgestures/train/"
test_dir =  "handgestures/test/"
def get_data(folder):
    #Load the data and labels from the given folder.
    X = []
    y = []

    for folderName in os.listdir(folder):
        if not folderName.startswith('.'):
            if folderName in ['A']:
                label = 0
            elif folderName in ['B']:
                label = 1
            elif folderName in ['C']:
                label = 2
            elif folderName in ['D']:
                label = 3
            elif folderName in ['E']:
                label = 4
            elif folderName in ['F']:
                label = 5
            elif folderName in ['G']:
                label = 6
            elif folderName in ['H']:
                label = 7
            elif folderName in ['I']:
                label = 8
            elif folderName in ['K']:
                label = 9
            elif folderName in ['L']:
                label = 10
            elif folderName in ['M']:
                label = 11
            elif folderName in ['N']:
                label = 12
            elif folderName in ['O']:
                label = 13
            elif folderName in ['P']:
                label = 14
            elif folderName in ['Q']:
                label = 15
            elif folderName in ['R']:
                label = 16
            elif folderName in ['S']:
                label = 17
            elif folderName in ['T']:
                label = 18
            elif folderName in ['U']:
                label = 19
            elif folderName in ['V']:
                label = 20
            elif folderName in ['W']:
                label = 21
            elif folderName in ['X']:
```



```
            elif folderName in ['X']:
                label = 22
            elif folderName in ['Y']:
                label = 23

            for image_filename in tqdm(os.listdir(folder + folderName)):
                img_file = cv2.imread(folder + folderName + '/' + image_filename)
                if img_file is not None:
                    img_file = skimage.transform.resize(img_file, (IMG_SIZE, IMG_SIZE, 3))
                    img_arr = np.asarray(img_file)
                    X.append(img_arr)
                    y.append(label)
    X = np.asarray(X)
    y = np.asarray(y)

    return X,y

X_train, y_train = get_data(train_dir)
X_test, y_test= get_data(test_dir)
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2)

#hot vector labels (1=[0,1,0,0,0,0,0,0,0,0])
y_trainHot = to_categorical(y_train, num_classes = 24)
y_testHot = to_categorical(y_test, num_classes = 24)
```

```
100%|██████████| 3000/3000 [00:20<00:00, 145.27it/s]
100%|██████████| 3000/3000 [00:21<00:00, 139.73it/s]
100%|██████████| 3000/3000 [00:20<00:00, 143.26it/s]
100%|██████████| 3000/3000 [00:20<00:00, 143.22it/s]
100%|██████████| 3000/3000 [00:21<00:00, 141.03it/s]
100%|██████████| 3000/3000 [00:20<00:00, 144.62it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.85it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.57it/s]
100%|██████████| 3000/3000 [00:21<00:00, 139.85it/s]
100%|██████████| 3000/3000 [00:21<00:00, 140.14it/s]
100%|██████████| 3000/3000 [00:21<00:00, 139.90it/s]
100%|██████████| 3000/3000 [00:22<00:00, 133.06it/s]
100%|██████████| 3000/3000 [00:21<00:00, 140.16it/s]
100%|██████████| 3000/3000 [00:20<00:00, 143.07it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.60it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.94it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.52it/s]
100%|██████████| 3000/3000 [00:20<00:00, 146.42it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.99it/s]
100%|██████████| 3000/3000 [00:20<00:00, 146.75it/s]
100%|██████████| 3000/3000 [00:20<00:00, 146.23it/s]
100%|██████████| 3000/3000 [00:20<00:00, 146.44it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.47it/s]
100%|██████████| 3000/3000 [00:20<00:00, 145.08it/s]
```

4.4: Convolution:

Convolutional neural networks consist of multiple layers of artificial neurons. Those artificial neurons are mathematical functions that calculate the weighted sum of inputs and outputs with an activation value. When an image is an input, each layer will generate activation functions that are passed to the next layer. Then the first layer extracts basic information like diagonal and horizontal edges. Then the output is relayed to the second layer which detects the more complex information like corners and combinational edges. If there's more layers in the neural network we can identify more complex information like objects, faces and gestures.

Then the classification layer outputs a score (Value between 0 and 1) that specifies how likely the models thinks the image belongs to a specific class.

Then the images go to the model layers and we start with the image in an input layer it then convolution should be done on it with the feature detector to form the feature map. This is one of the most important parts in feature detection.

The image is then convoluted with a number of features so that there's a number of features present. The more features present the better it is for the classification of the image. The main aim of convolution is to detect the features of the image. (Figure 12)

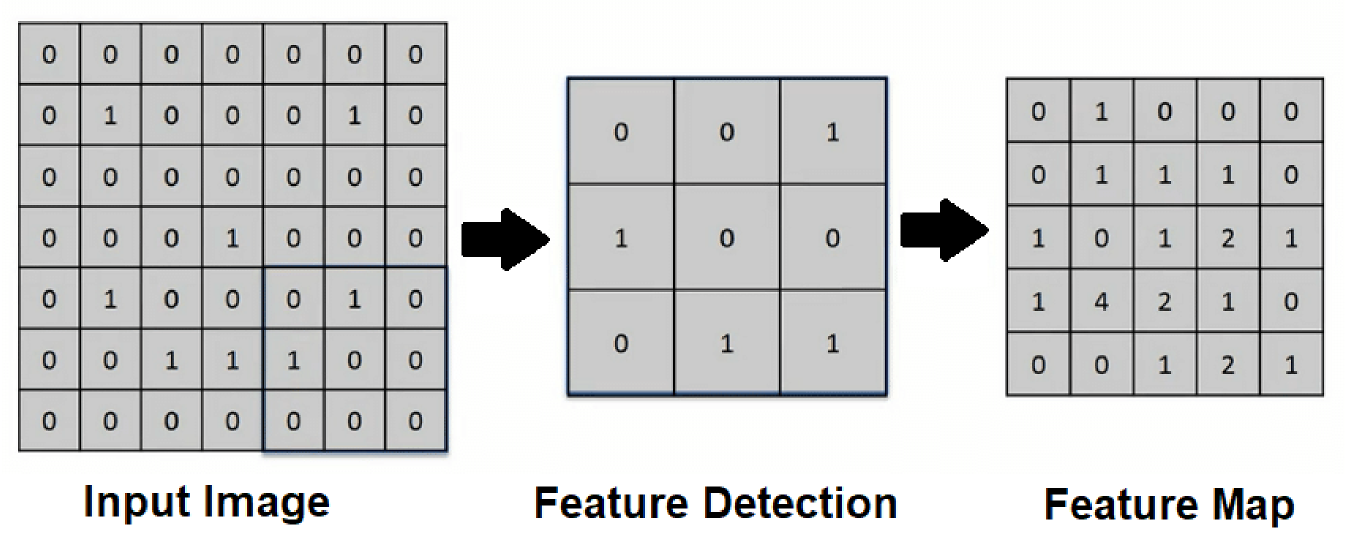


Figure 12: Convolution

4.5: Max Pool:

Similar to the convolution layer, the pooling layer is the layer responsible for reducing the size of the convolved feature. This is done so we can use less computational power in processing the data by deducing it's dimensions. In max pooling we find the maximum value of a pixel from the image and then perform a noise suppressant on it.

By doing that we discard the noisy activations and de-noise along with dimensionality reduction.

After the convoluted result is contained in the feature map, there would be a big number of feature maps. So, there will be some issues like there would be a large amount of data present to be dealt with.

Another issue that there would be differences between some of the feature maps like difference in size, orientation and such in different images.

We can address these issues using Max pooling. Max pooling is used by selecting a small grid and maximizing the preservation of the value while reducing the size of data.

(Figure 13)

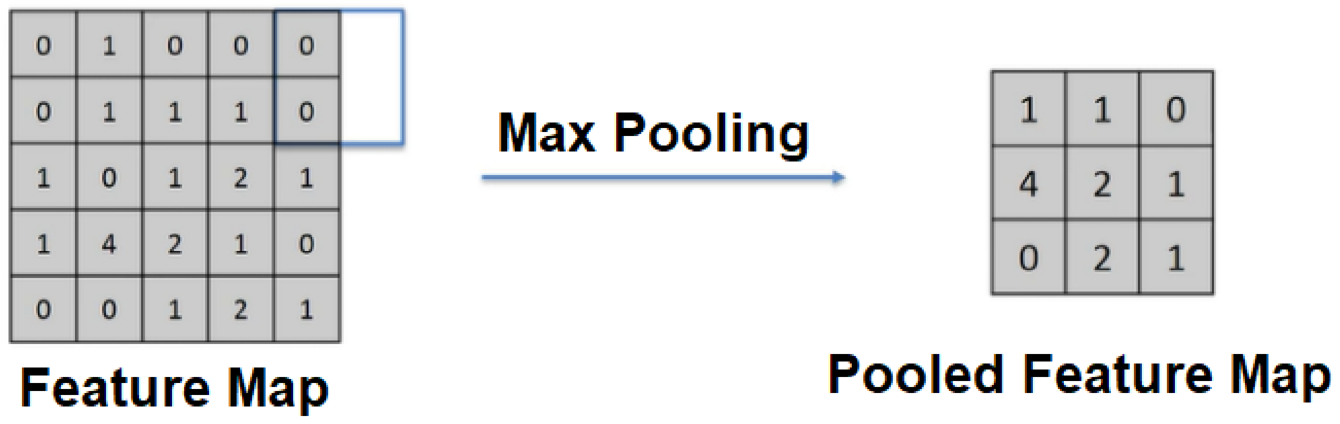


Figure 13: Max pooling

4.6: Flatten:

After The pooled feature map is created it needs to be flattened. And to do that it has to go through a flattening layer so it can be used as input in the next Neural Network.

The flattening layer is considered the input layer of the next neural network. And it's done by converting the max pooling output into a column.

As you can see in (Figure 14), we have multiple pooled feature maps from the previous layer which is max pooling.

What happens to these pooled feature maps is that it's converted to a long vector of input data that gets passed to the next layer as mentioned above so we can process it further and get the result we are hoping for.

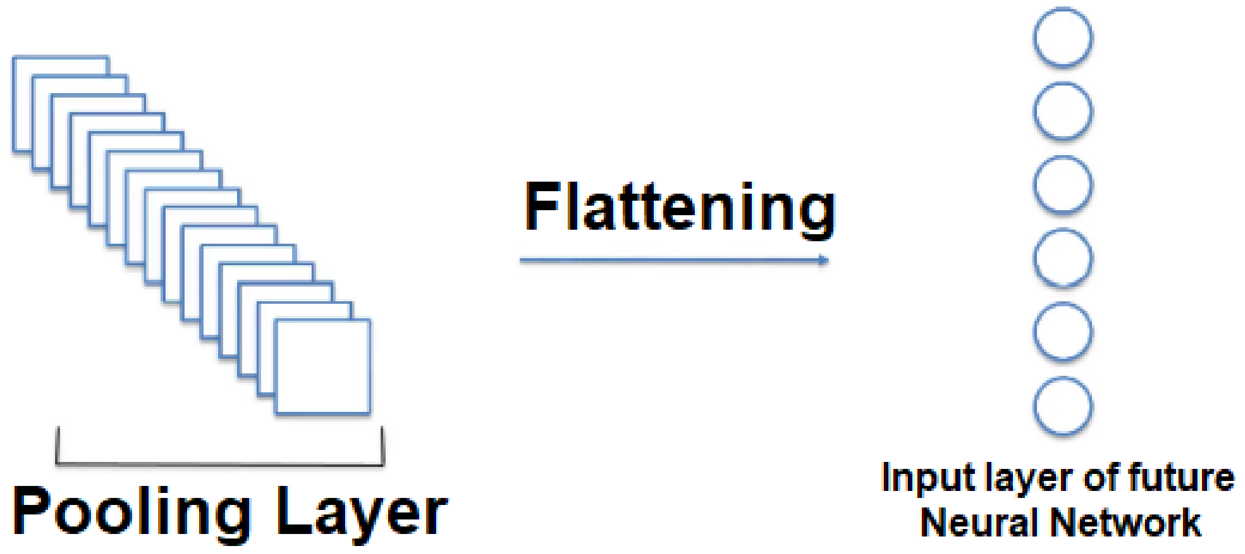


Figure 14: Flattening

4.7: Model structure:

When we combine all the previous points that are Convolution, Max Pool and Flatten, we get a single layer in our model.

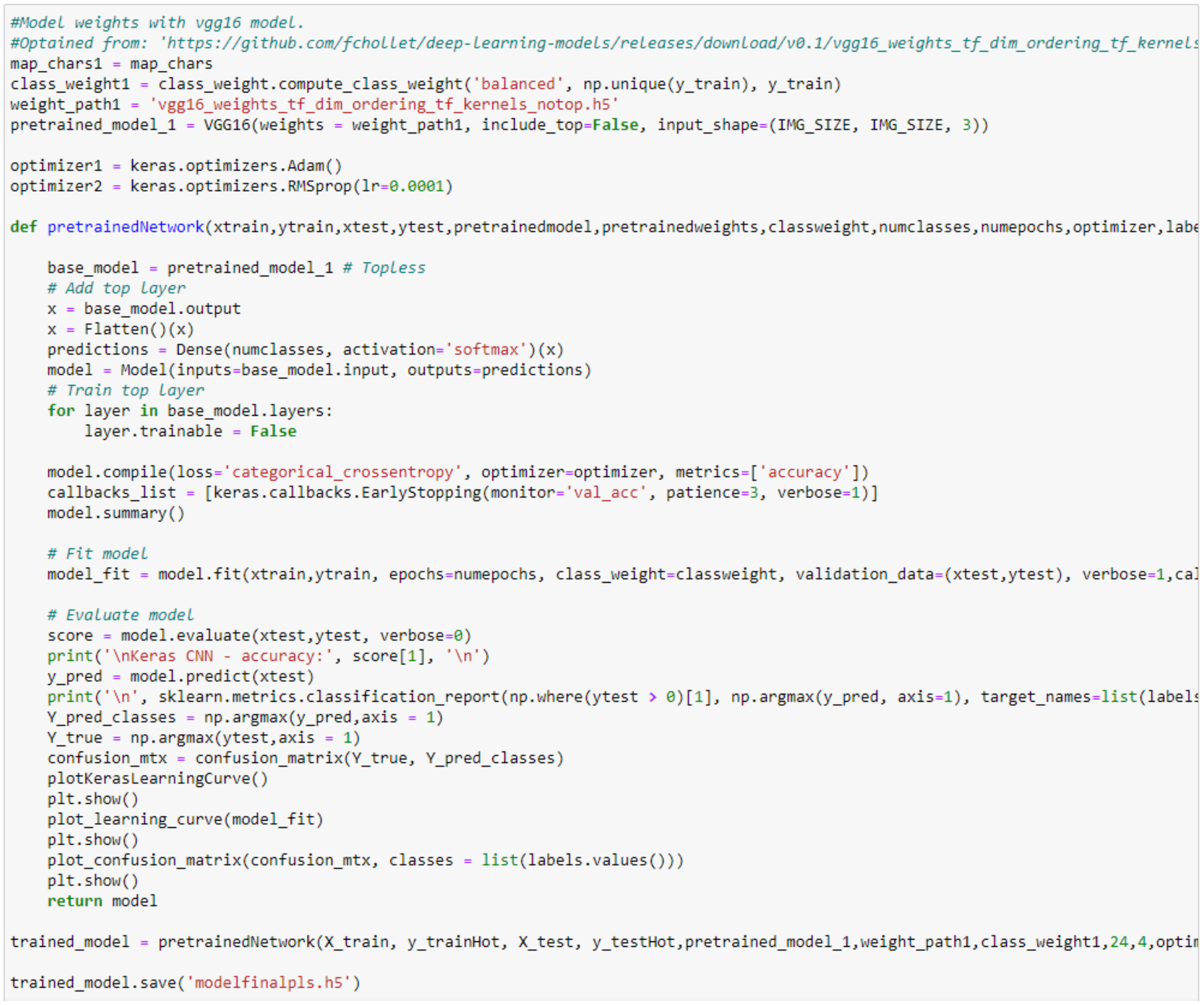
I've chosen to have 5 layers in my model because it proved the best results. When trying 3 and 4 layers it led to very bad accuracies and bad results over all. So, our model is contained of 5 of these layers as seen on this figure. (Figure 15)

Refer to (Figure 5) for model classes.

	Layer (Type)	Output Shape
	input image (InputLayer)	(None, 50, 50, 3)
Layer 1	block1_conv (Conv2D)	(None, 50, 50, 64)
	block1_pool (MaxPooling2D)	(None, 25, 25, 64)
Layer 2	block2_conv (Conv2D)	(None, 25, 25, 128)
	block2_pool (MaxPooling2D)	(None, 12, 12, 128)
Layer 3	block3_conv (Conv2D)	(None, 12, 12, 256)
	block3_pool (MaxPooling2D)	(None, 6, 6, 256)
Layer 4	block4_conv (Conv2D)	(None, 6, 6, 512)
	block4_pool (MaxPooling2D)	(None, 3, 3, 512)
Layer 5	block5_conv (Conv2D)	(None, 3, 3, 512)
	block5_pool (MaxPooling2D)	(None, 1, 1, 512)
	flatten (Flatten)	(None, 512)

Figure 15:Block Layout

Model code screenshot:



```
#Model weights with vgg16 model.
#Optained from: 'https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels
map_chars1 = map_chars
class_weight1 = class_weight.compute_class_weight('balanced', np.unique(y_train), y_train)
weight_path1 = 'vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5'
pretrained_model_1 = VGG16(weights = weight_path1, include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

optimizer1 = keras.optimizers.Adam()
optimizer2 = keras.optimizers.RMSprop(lr=0.0001)

def pretrainedNetwork(xtrain,ytrain,xtest,ytest,pretrainedmodel,pretrainedweights,classweight,numclasses,numepochs,optimizer,labe

    base_model = pretrained_model_1 # Topless
    # Add top layer
    x = base_model.output
    x = Flatten()(x)
    predictions = Dense(numclasses, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    # Train top layer
    for layer in base_model.layers:
        layer.trainable = False

    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    callbacks_list = [keras.callbacks.EarlyStopping(monitor='val_acc', patience=3, verbose=1)]
    model.summary()

    # Fit model
    model_fit = model.fit(xtrain,ytrain, epochs=numepochs, class_weight=classweight, validation_data=(xtest,ytest), verbose=1,cal

    # Evaluate model
    score = model.evaluate(xtest,ytest, verbose=0)
    print('\nKeras CNN - accuracy:', score[1], '\n')
    y_pred = model.predict(xtest)
    print('\n', sklearn.metrics.classification_report(np.where(ytest > 0)[1], np.argmax(y_pred, axis=1), target_names=list(labels
    Y_pred_classes = np.argmax(y_pred,axis = 1)
    Y_true = np.argmax(ytest,axis = 1)
    confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
    plotKerasLearningCurve()
    plt.show()
    plot_learning_curve(model_fit)
    plt.show()
    plot_confusion_matrix(confusion_mtx, classes = list(labels.values()))
    plt.show()
    return model

trained_model = pretrainedNetwork(X_train, y_trainHot, X_test, y_testHot,pretrained_model_1,weight_path1,class_weight1,24,4,optim

trained_model.save('modelfinalpls.h5')
```

4.8: Dataset structure:

The dataset is created by using the camera feed and capturing images from its frames and saving those images in a specific folder. The images saved have a resolution of 200 by 200.

The images are captured in a specific Region of Interest (R. O. I.) and then resized to the resolution of 200 by 200 after being captured.

The R. O. I. is defined by a blue rectangle and the user is expect to keep the gestures inside that rectangle so the program would recognize it in the prediction phase.

The program is programmed to capture 24 total gestures which are the English alphabet except the letter “J” and the letter “Z” because those 2 letters are done in a motion gesture.

The program asks the user to press the key “Enter” on their keyboard when they are ready to do the first gesture which is the gesture for the letter “A” and then the program proceeds to capture 3000 images of that gesture for the training phase.

(Figure 16) (Figure 17)

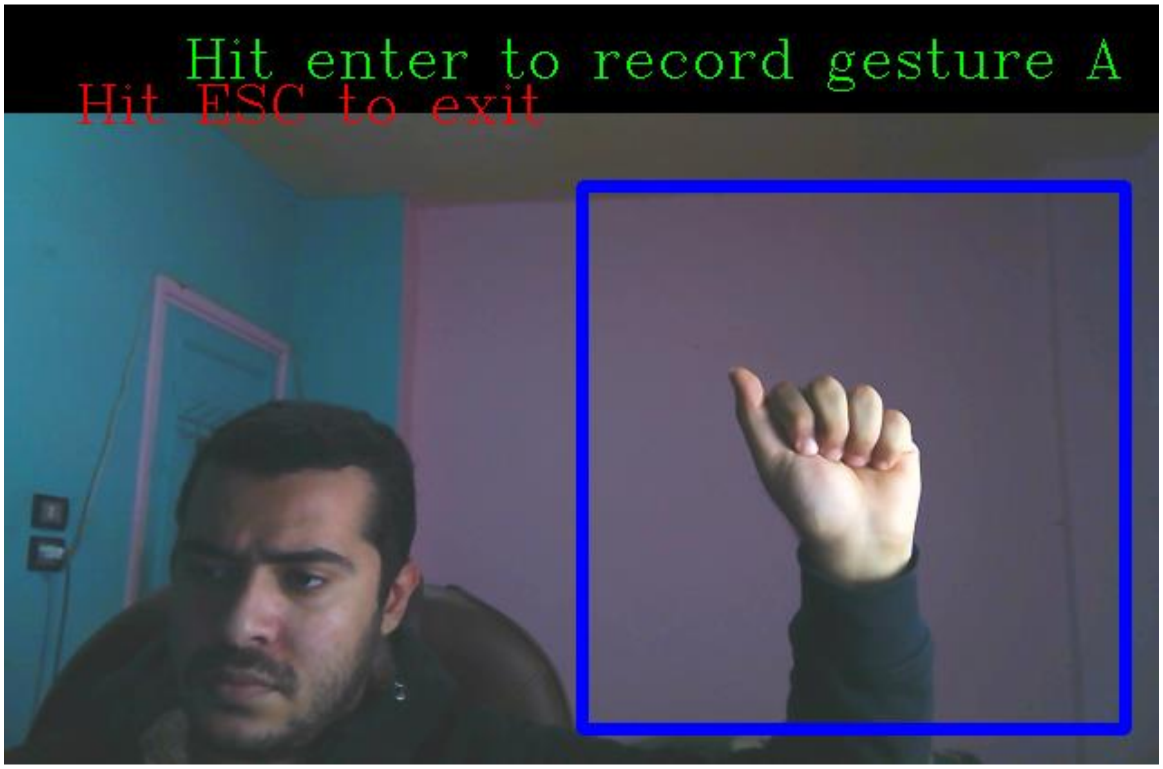


Figure 16: Capture gesture A

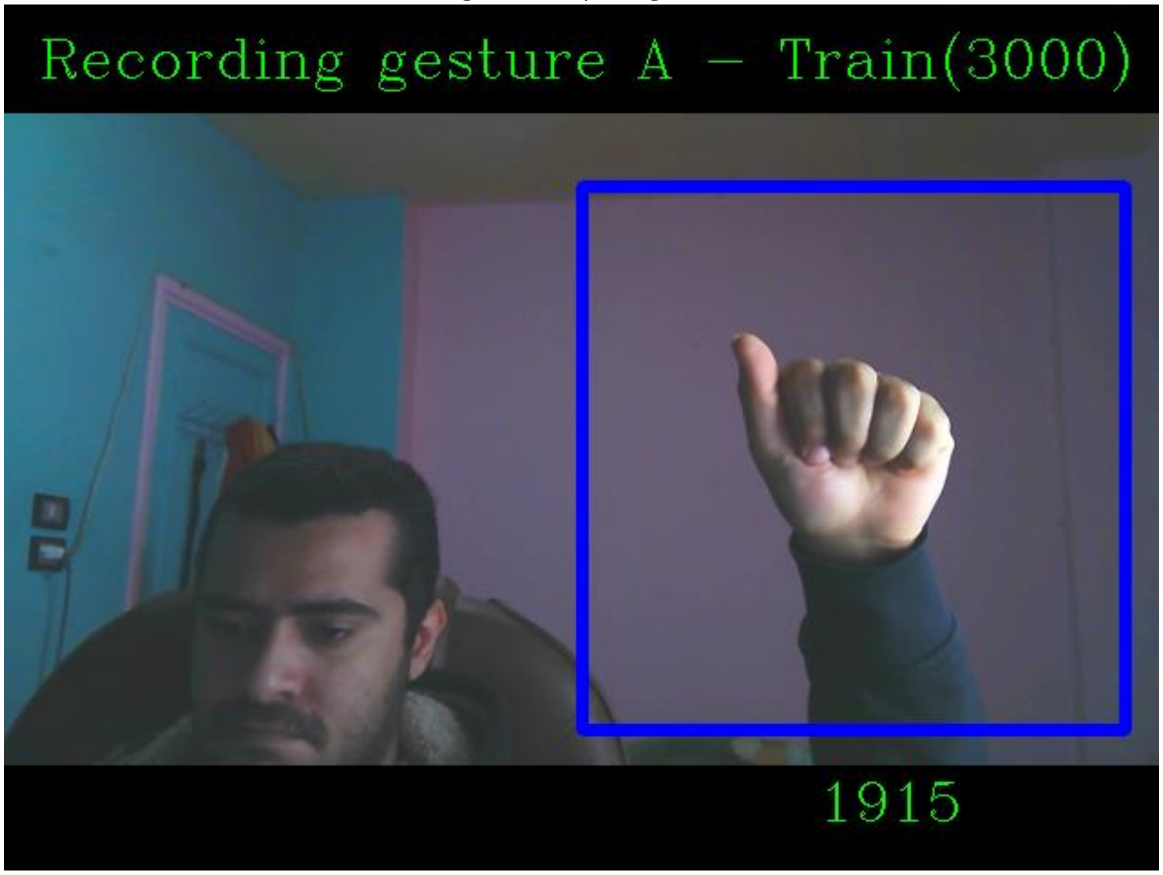


Figure 17: Capture gesture A (Training)

If the user wishes to capture less than 3000 images the user may press the key “Enter” on the keyboard to only save what’s already been captured.

Then the program proceeds to ask they user to capture 1000 more images of the same gesture for testing purposes and of course if the user wishes to end this process earlier, they can also press “Enter” key on their keyboard to end this process early. (Figure 18)

Then the program may ask the user to do the same steps for all the remaining 23 gestures. If the user doesn’t wish to continue with the gestures the user may press the “Escape” key on their keyboard to terminate the dataset process in its entirety early.

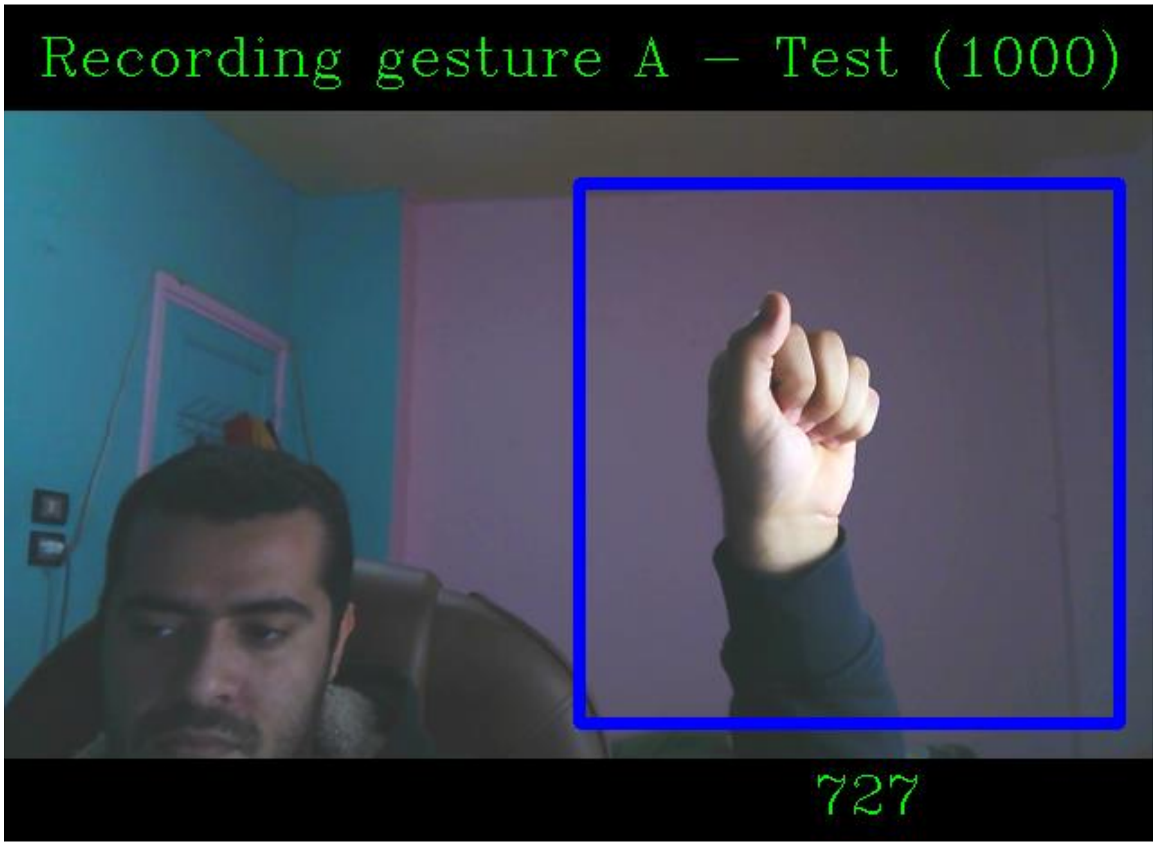
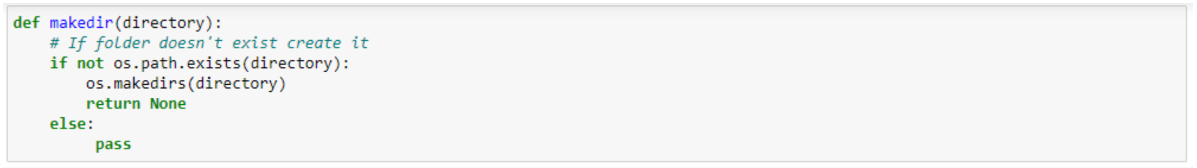
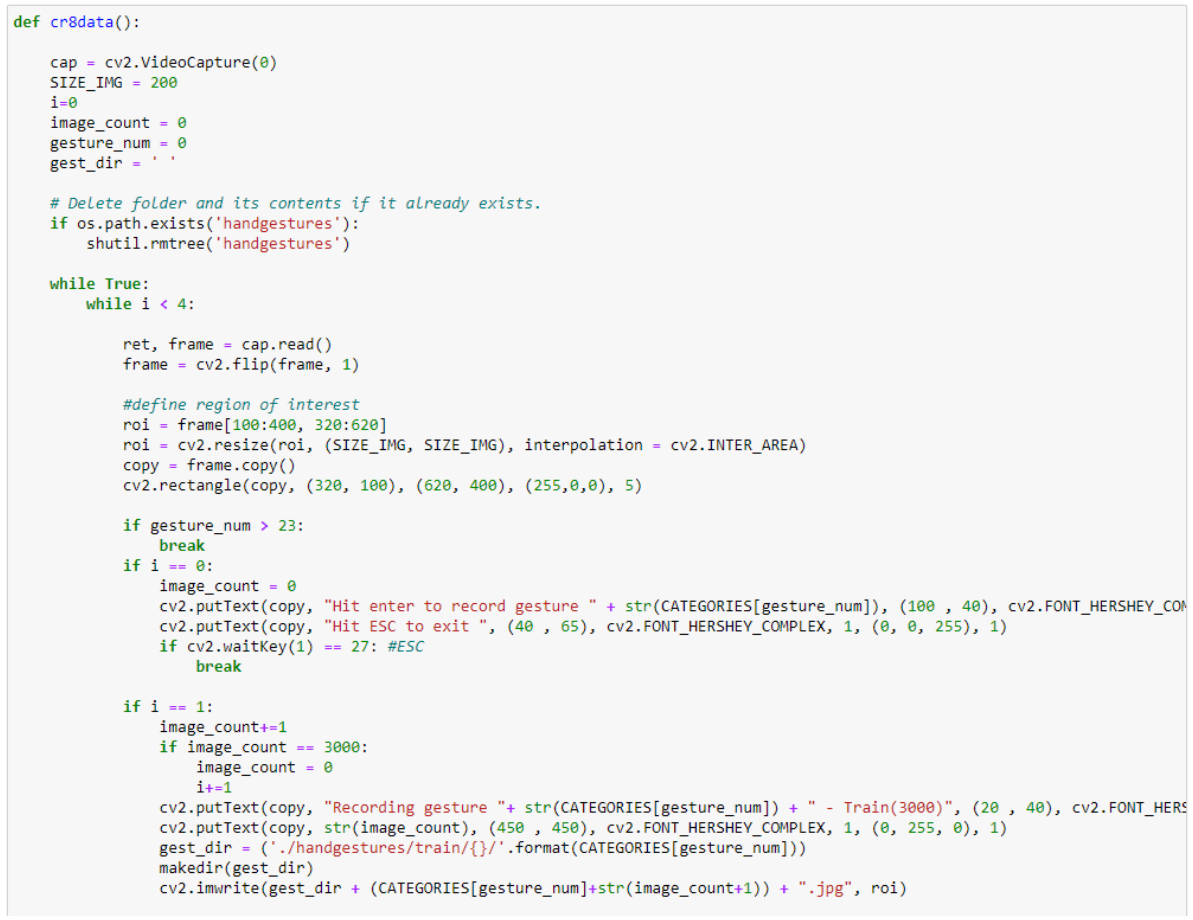


Figure 18: Capture gesture A (Testing)

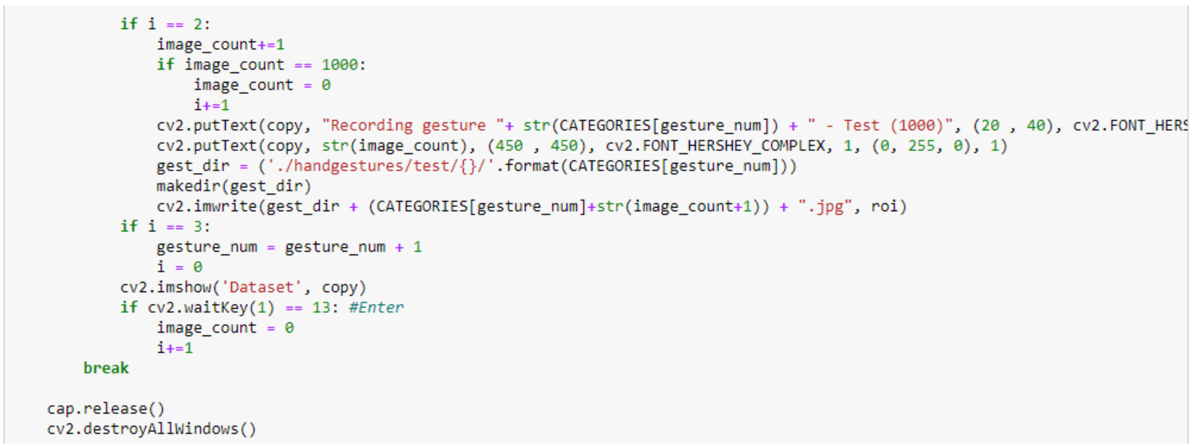
Dataset code screenshots:



```python
def makedir(directory):
    # If folder doesn't exist create it
    if not os.path.exists(directory):
        os.makedirs(directory)
        return None
    else:
         pass
```



```python
def cr8data():

    cap = cv2.VideoCapture(0)
    SIZE_IMG = 200
    i=0
    image_count = 0
    gesture_num = 0
    gest_dir = ' '

    # Delete folder and its contents if it already exists.
    if os.path.exists('handgestures'):
        shutil.rmtree('handgestures')

    while True:
        while i < 4:

            ret, frame = cap.read()
            frame = cv2.flip(frame, 1)

            #define region of interest
            roi = frame[100:400, 320:620]
            roi = cv2.resize(roi, (SIZE_IMG, SIZE_IMG), interpolation = cv2.INTER_AREA)
            copy = frame.copy()
            cv2.rectangle(copy, (320, 100), (620, 400), (255,0,0), 5)

            if gesture_num > 23:
                break
            if i == 0:
                image_count = 0
                cv2.putText(copy, "Hit enter to record gesture " + str(CATEGORIES[gesture_num]), (100 , 40), cv2.FONT_HERSHEY_COM
                cv2.putText(copy, "Hit ESC to exit ", (40 , 65), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 1)
                if cv2.waitKey(1) == 27: #ESC
                    break

            if i == 1:
                image_count+=1
                if image_count == 3000:
                    image_count = 0
                    i+=1
                cv2.putText(copy, "Recording gesture "+ str(CATEGORIES[gesture_num]) + " - Train(3000)", (20 , 40), cv2.FONT_HERS
                cv2.putText(copy, str(image_count), (450 , 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 1)
                gest_dir = ('./handgestures/train/{}/'.format(CATEGORIES[gesture_num]))
                makedir(gest_dir)
                cv2.imwrite(gest_dir + (CATEGORIES[gesture_num]+str(image_count+1)) + ".jpg", roi)
```



```python
            if i == 2:
                image_count+=1
                if image_count == 1000:
                    image_count = 0
                    i+=1
                cv2.putText(copy, "Recording gesture "+ str(CATEGORIES[gesture_num]) + " - Test (1000)", (20 , 40), cv2.FONT_HERS
                cv2.putText(copy, str(image_count), (450 , 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 1)
                gest_dir = ('./handgestures/test/{}/'.format(CATEGORIES[gesture_num]))
                makedir(gest_dir)
                cv2.imwrite(gest_dir + (CATEGORIES[gesture_num]+str(image_count+1)) + ".jpg", roi)
            if i == 3:
                gesture_num = gesture_num + 1
                i = 0
            cv2.imshow('Dataset', copy)
            if cv2.waitKey(1) == 13: #Enter
                image_count = 0
                i+=1
        break

    cap.release()
    cv2.destroyAllWindows()
```

4.9: Prediction structure:

The prediction is done by using the camera feed and capturing images from its frames and comparing those images which have a resolution of 200 by 200 to the trained model which the model then predicts which gesture is being made.

The images are captured in a specific Region of Interest (R. O. I.) defined by a blue rectangle and the user is expect to keep the gestures inside that rectangle so the program would predict which gesture is being made.

When a gesture is made the program displays which letter the model thinks the user is gesturing and if the same gesture is being made for 3 continuous seconds the program will save it so the user can form word from letters. (Figure 19)

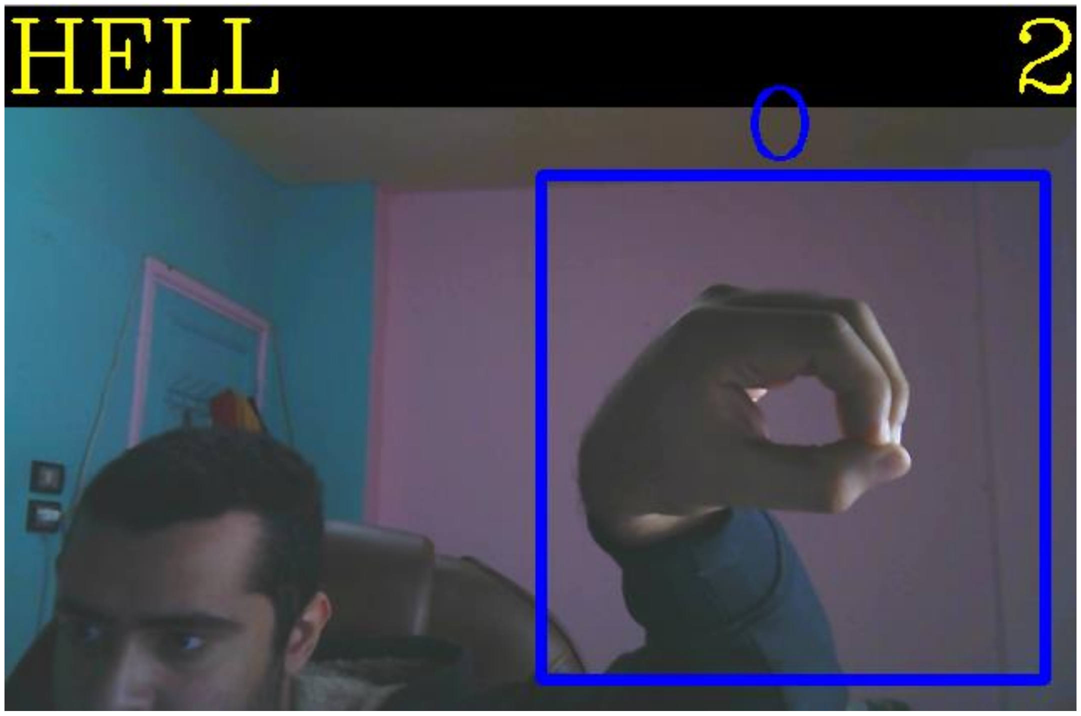
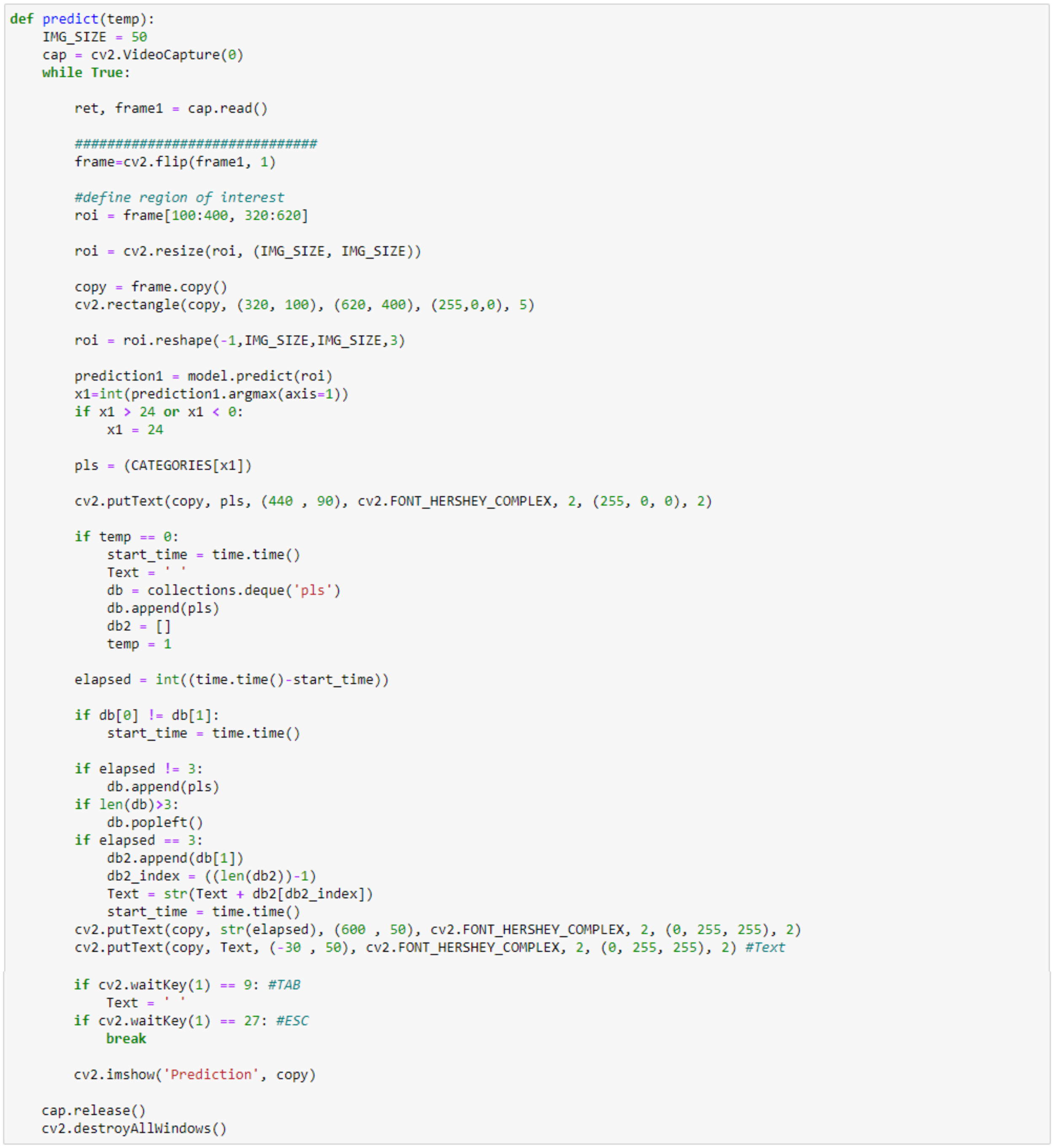


Figure 19: Gesture prediction

Prediction code screenshot:



```python
def predict(temp):
    IMG_SIZE = 50
    cap = cv2.VideoCapture(0)
    while True:

        ret, frame1 = cap.read()

        ##############################
        frame=cv2.flip(frame1, 1)

        #define region of interest
        roi = frame[100:400, 320:620]

        roi = cv2.resize(roi, (IMG_SIZE, IMG_SIZE))

        copy = frame.copy()
        cv2.rectangle(copy, (320, 100), (620, 400), (255,0,0), 5)

        roi = roi.reshape(-1,IMG_SIZE,IMG_SIZE,3)

        prediction1 = model.predict(roi)
        x1=int(prediction1.argmax(axis=1))
        if x1 > 24 or x1 < 0:
            x1 = 24

        pls = (CATEGORIES[x1])

        cv2.putText(copy, pls, (440 , 90), cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 0), 2)

        if temp == 0:
            start_time = time.time()
            Text = ' '
            db = collections.deque('pls')
            db.append(pls)
            db2 = []
            temp = 1

        elapsed = int((time.time()-start_time))

        if db[0] != db[1]:
            start_time = time.time()

        if elapsed != 3:
            db.append(pls)
        if len(db)>3:
            db.popleft()
        if elapsed == 3:
            db2.append(db[1])
            db2_index = ((len(db2))-1)
            Text = str(Text + db2[db2_index])
            start_time = time.time()
        cv2.putText(copy, str(elapsed), (600 , 50), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 255, 255), 2)
        cv2.putText(copy, Text, (-30 , 50), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 255, 255), 2) #Text

        if cv2.waitKey(1) == 9: #TAB
            Text = ' '
        if cv2.waitKey(1) == 27: #ESC
            break

        cv2.imshow('Prediction', copy)

    cap.release()
    cv2.destroyAllWindows()
```

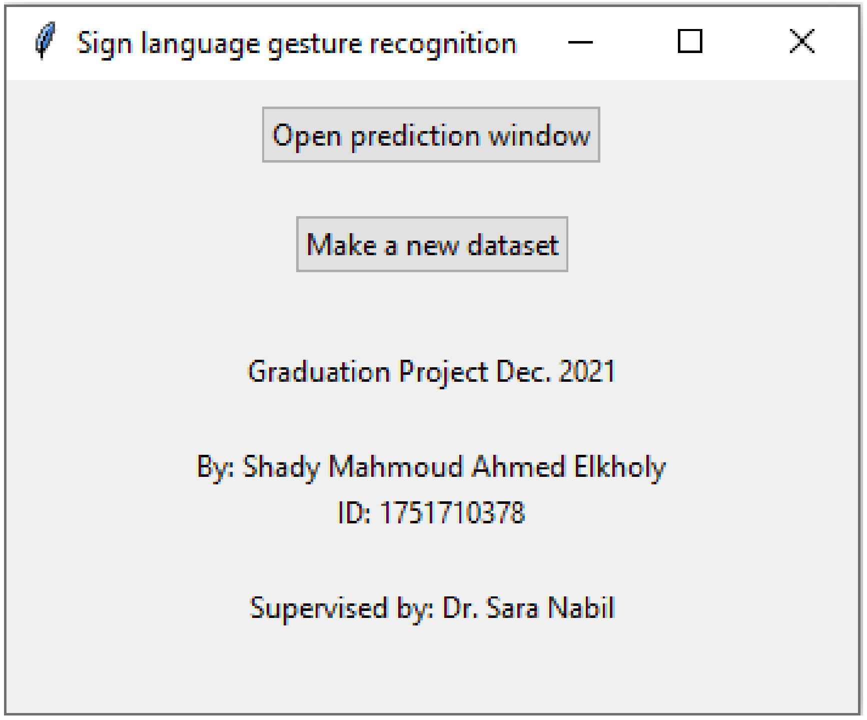
4.10: User Interface:

User interface is important to support the functionality of the program and to help meet user expectations. Any program should be expected to have a user interface that helps navigate the program easily and cleanly.

A user-friendly Interface was created using Tkinter library. It includes buttons for Making the dataset which will open a window that takes pictures of gestures for the dataset from the camera feed which will be explained in (section 4.10.1).

The user interface also has a button for opening up the prediction window which will open a camera feed and the user is then expected to gesture the character they want the program to predict in the specified frame which will be explained in (section 4.10.2).

The user interface also has some text saying my name, ID, and who supervised this project. (Figure 20)



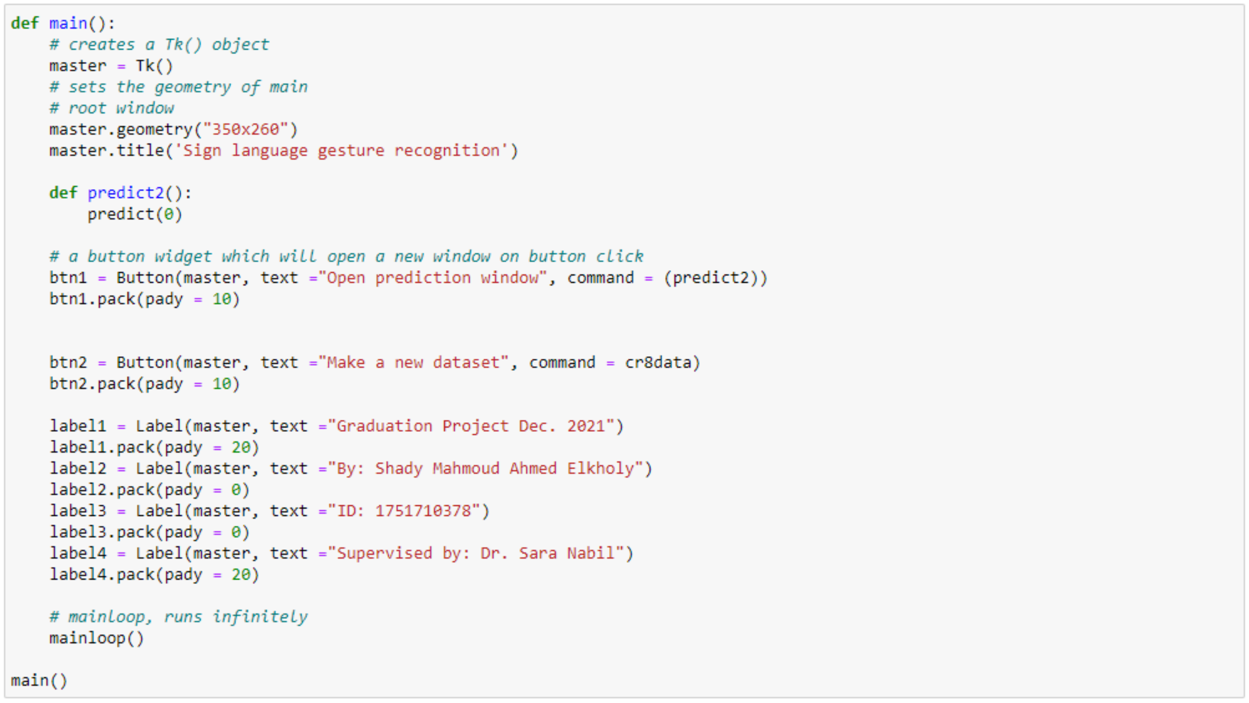
Graduation Project Dec. 2021

By: Shady Mahmoud Ahmed Elkholy
ID: 1751710378

Supervised by: Dr. Sara Nabil

Figure 20: Main user interface

User interface code screenshot:



```
def main():
    # creates a Tk() object
    master = Tk()
    # sets the geometry of main
    # root window
    master.geometry("350x260")
    master.title('Sign language gesture recognition')

    def predict2():
        predict(0)

    # a button widget which will open a new window on button click
    btn1 = Button(master, text ="Open prediction window", command = (predict2))
    btn1.pack(pady = 10)


    btn2 = Button(master, text ="Make a new dataset", command = cr8data)
    btn2.pack(pady = 10)

    label1 = Label(master, text ="Graduation Project Dec. 2021")
    label1.pack(pady = 20)
    label2 = Label(master, text ="By: Shady Mahmoud Ahmed Elkholy")
    label2.pack(pady = 0)
    label3 = Label(master, text ="ID: 1751710378")
    label3.pack(pady = 0)
    label4 = Label(master, text ="Supervised by: Dr. Sara Nabil")
    label4.pack(pady = 20)

    # mainloop, runs infinitely
    mainloop()

main()
```

4.10.1: Dataset window:

When The dataset button is pressed it will open a camera feed and ask the user to gesture the next letter and after pressing the key “Enter” on their keyboard the program will begin to take pictures of the R. O. I. to capture the gesture. (Figure 21)

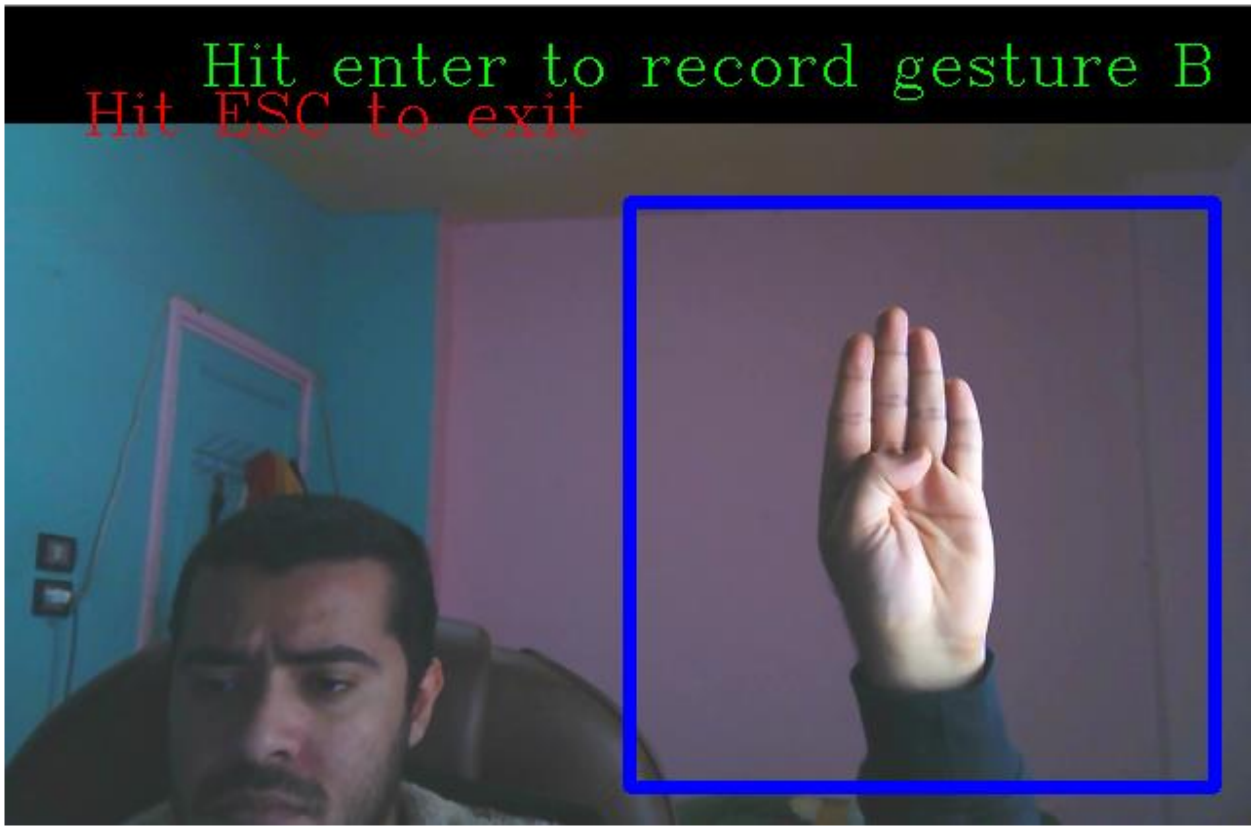


Figure 21: Dataset gesture B

It will then proceed to take up to 3000 pictures of that gesture preferably using different angles and lighting conditions.

After the program has captured 3000 pictures it will automatically go to the test pictures, if the user would like to take less than 3000 pictures they can press “Enter” on their keyboard to end the process earlier. (Figure 22)

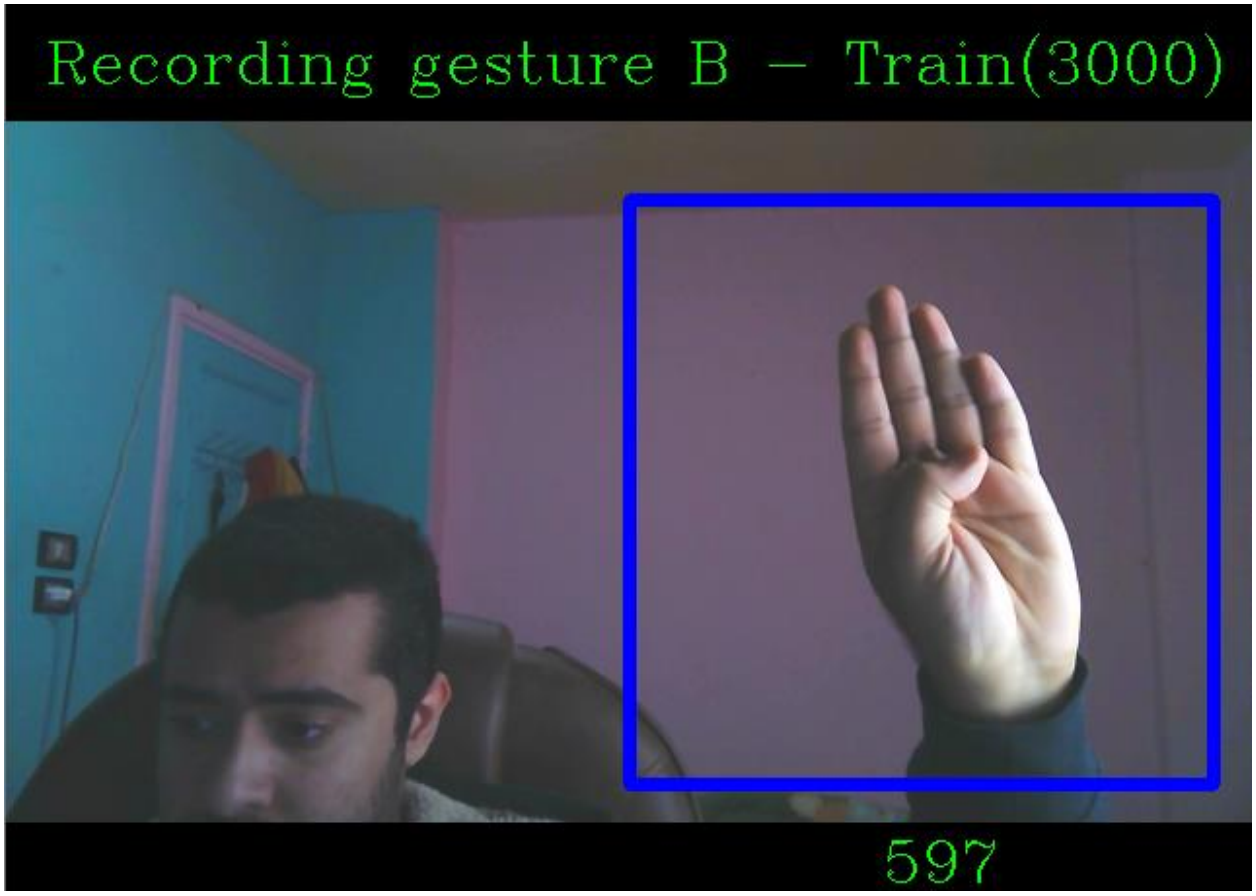


Figure 22: Dataset gesture B (Train)

The next part is 1000 pictures of the same gesture for validation testing purposes. This part can also be ended earlier by pressing the “Enter” key on the keyboard.

Then it will repeat all the previous steps for all the alphabet. If the user wants to terminate the whole process early instead of finishing all the alphabet, they can press the “Escape” key on their keyboard and all the pictures they have already taken will be saved.

(Figure 23)

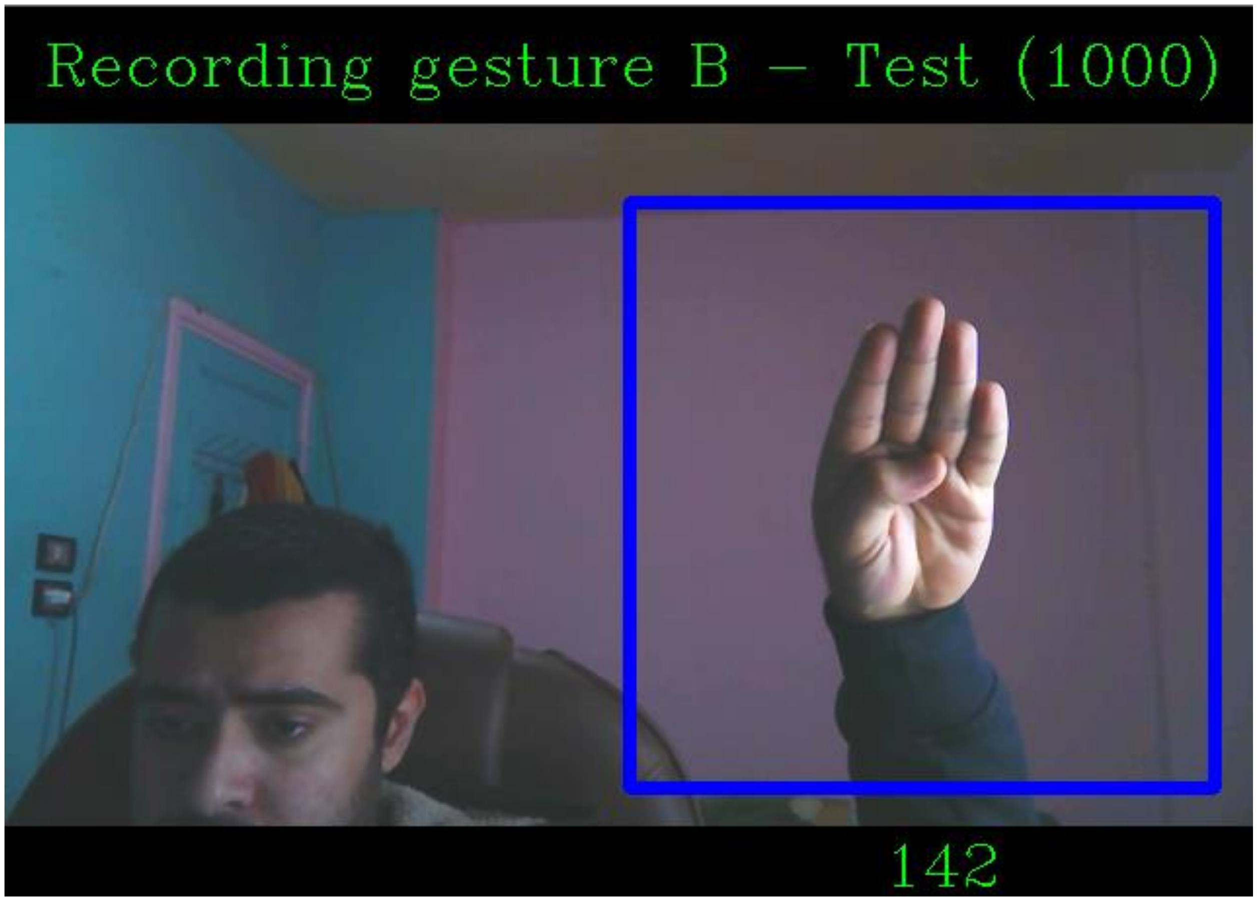


Figure 23: Dataset gesture B (Test)

4.10.2: Prediction window:

When "Open prediction window" button is pressed it will open a camera feed and the user is expected to gesture inside the R. O. I. which is the blue rectangle.

When the user gestures in the blue rectangle (R. O. I.) the program predicts which gesture is being made and outputs it on the screen above the (R. O. I.). (Figure 24)

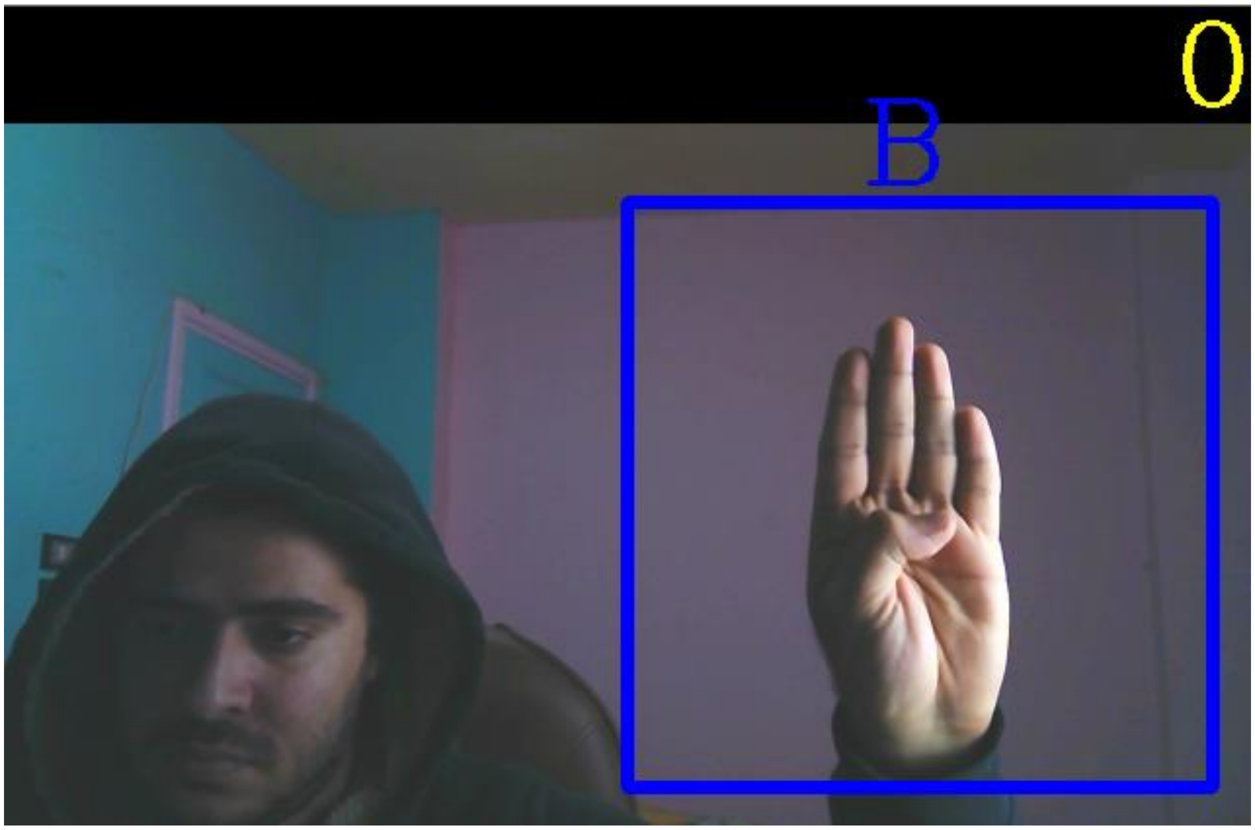


Figure 24: Prediction window

If the user is doing the same gesture for 3 seconds the program saves the letter that correspond to the gesture into a word so the user can make out words using gestures.

The prediction is done by using the camera feed and capturing images from its frames and comparing those images which have a resolution of 200 by 200 to the trained model which the model then predicts which gesture is being made.

To form a word the user is expected to spell out that word using the gestures corresponding to each letter of that word. If the user wishes to start over because of a mistake or to make another different word they press the key “Tab” on their keyboard to delete the letters they already did.

(Figure 25)

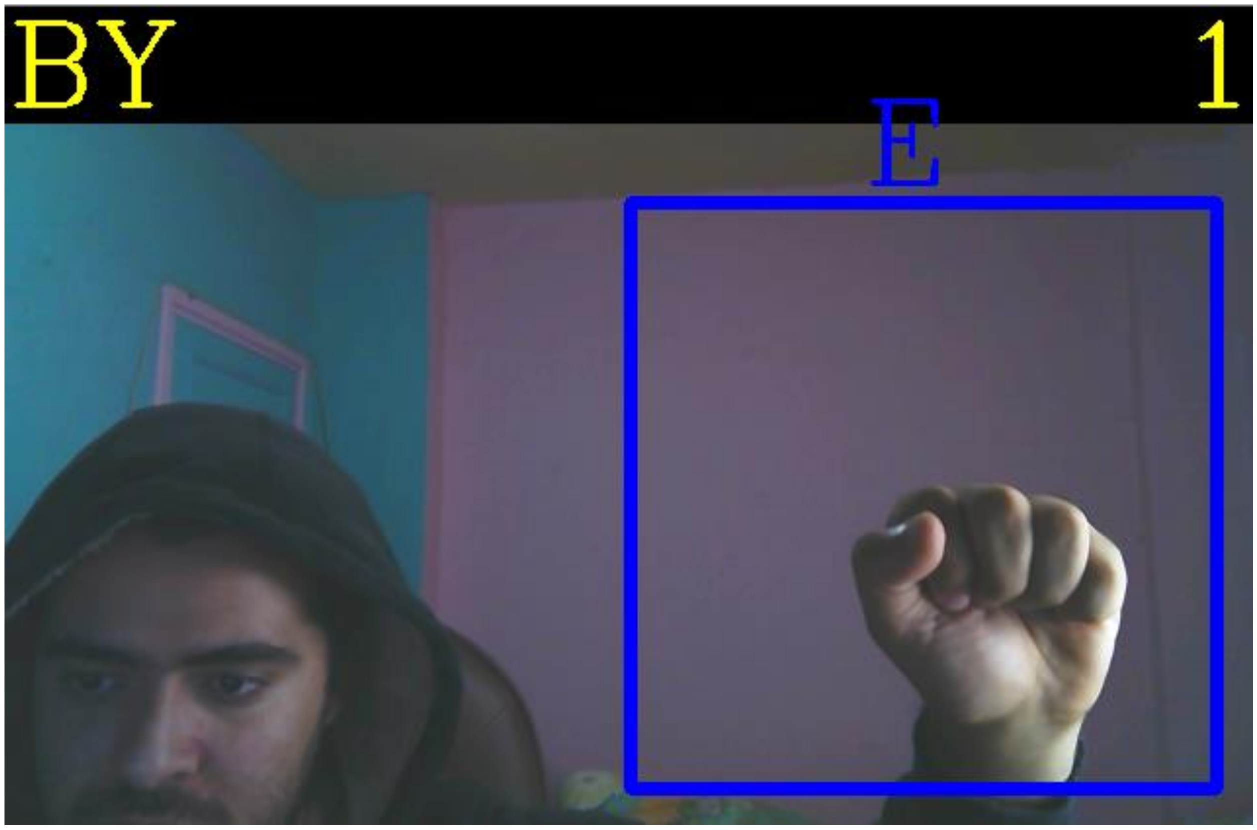


Figure 25: Prediction window (Word)

Chapter 5: Results and discussion:

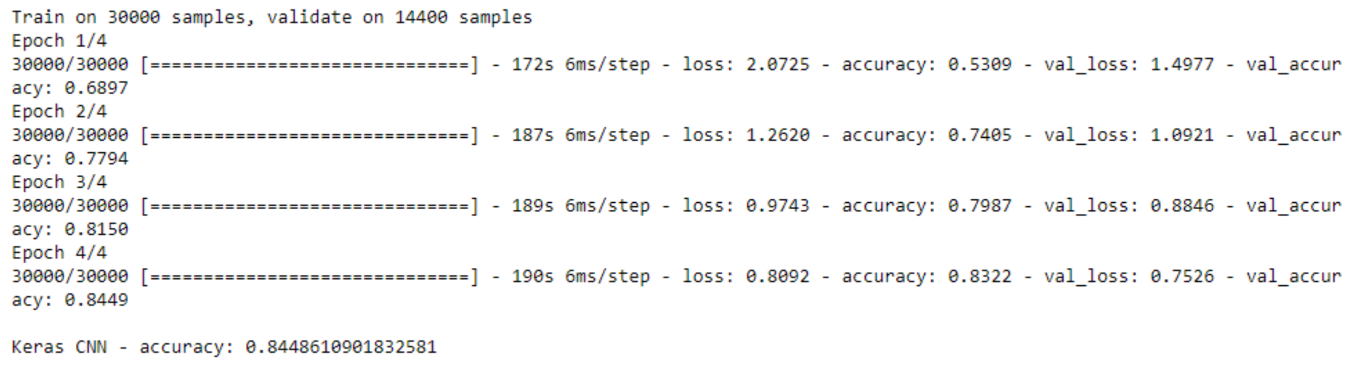
5.1: Findings:

I have achieved an accuracy of 12.1% using 1 layer of the model. And when I changed the model into 3 layers the accuracy achieved was 18.8%.

When I switch to a 5 layer an accuracy of 53.09% was achieved using a single epoch. Epochs are a full cycle of the model training and if the number of epochs rises so should the accuracy of that model epochs is the number of times the model is trained over the same data-set.

When I ran the model for 10 epochs it got an accuracy of 94% but was later found out that it overfitted the model and the prediction accuracy wasn't that great. So, I've tried the model with 3 epochs with an accuracy of 79% and found out the prediction's accuracy was okay.

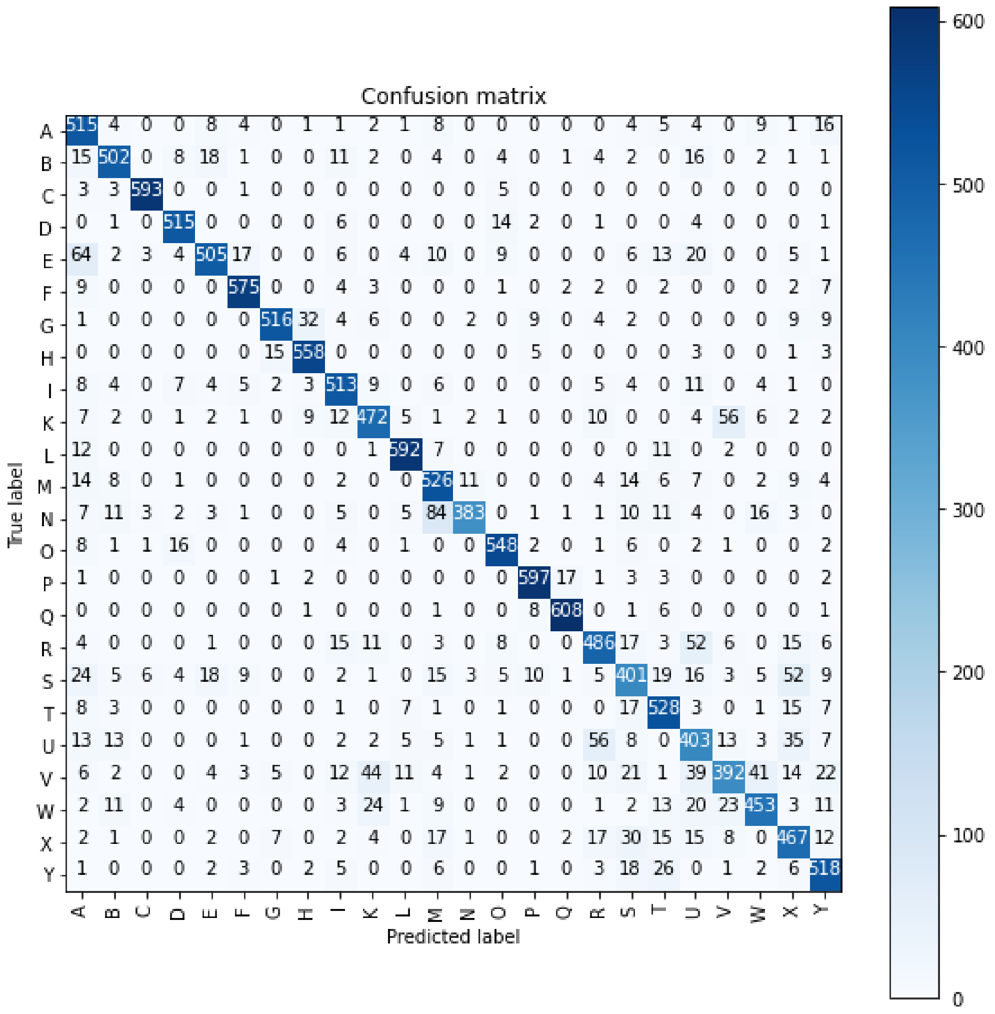
So, I ran it for 4 epochs and got an accuracy of 84.49% with some good prediction accuracy as shown in (Figure 26). So, I've decided to stick with 4 epochs for the time being due to time constraints that didn't make me able to optimize it further.



```
Train on 30000 samples, validate on 14400 samples
Epoch 1/4
30000/30000 [==============================] - 172s 6ms/step - loss: 2.0725 - accuracy: 0.5309 - val_loss: 1.4977 - val_accur
acy: 0.6897
Epoch 2/4
30000/30000 [==============================] - 187s 6ms/step - loss: 1.2620 - accuracy: 0.7405 - val_loss: 1.0921 - val_accur
acy: 0.7794
Epoch 3/4
30000/30000 [==============================] - 189s 6ms/step - loss: 0.9743 - accuracy: 0.7987 - val_loss: 0.8846 - val_accur
acy: 0.8150
Epoch 4/4
30000/30000 [==============================] - 190s 6ms/step - loss: 0.8092 - accuracy: 0.8322 - val_loss: 0.7526 - val_accur
acy: 0.8449

Keras CNN - accuracy: 0.8448610901832581
```

Figure 26: Epochs

In the previous figure (Figure 26) the model compares what it thinks the gesture is with an image that hasn't been seen by the model that we know it's corresponding letter beforehand. And then the model sees if it got that gesture right or not outputting it as "Val accuracy" which is validation accuracy. As we can see in (Figure 27) we have a confusion matrix which shows us exactly what each and every letter is guessed and how many times it was guessed. A confusion matrix is performance measurement for machine learning classifications. If we look at letter A in the top left corner for example, we can see it was guessed it correctly 515 times and the model guessed it wrongly as other letter and it shows how many times it guessed it for each other letter as well.



True label \ Predicted label	A	B	C	D	E	F	G	H	I	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
A	515	4	0	0	8	4	0	1	1	2	1	8	0	0	0	0	0	4	5	4	0	9	1	16
B	15	502	0	8	18	1	0	0	11	2	0	4	0	4	0	1	4	2	0	16	0	2	1	1
C	3	3	593	0	0	1	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0
D	0	1	0	515	0	0	0	0	6	0	0	0	0	14	2	0	1	0	0	4	0	0	0	1
E	64	2	3	4	505	17	0	0	6	0	4	10	0	9	0	0	0	6	13	20	0	0	5	1
F	9	0	0	0	0	575	0	0	4	3	0	0	0	1	0	2	2	0	2	0	0	0	2	7
G	1	0	0	0	0	0	516	32	4	6	0	0	2	0	9	0	4	2	0	0	0	0	9	9
H	0	0	0	0	0	0	15	558	0	0	0	0	0	0	5	0	0	0	0	3	0	0	1	3
I	8	4	0	7	4	5	2	3	513	9	0	6	0	0	0	0	5	4	0	11	0	4	1	0
K	7	2	0	1	2	1	0	9	12	472	5	1	2	1	0	0	10	0	0	4	56	6	2	2
L	12	0	0	0	0	0	0	0	0	1	592	7	0	0	0	0	0	0	11	0	2	0	0	0
M	14	8	0	1	0	0	0	0	2	0	0	526	11	0	0	0	4	14	6	7	0	2	9	4
N	7	11	3	2	3	1	0	0	5	0	5	84	383	0	1	1	1	10	11	4	0	16	3	0
O	8	1	1	16	0	0	0	0	4	0	1	0	0	548	2	0	1	6	0	2	1	0	0	2
P	1	0	0	0	0	0	1	2	0	0	0	0	0	0	597	17	1	3	3	0	0	0	0	2
Q	0	0	0	0	0	0	0	1	0	0	0	1	0	0	8	608	0	1	6	0	0	0	0	1
R	4	0	0	0	1	0	0	0	15	11	0	3	0	8	0	0	486	17	3	52	6	0	15	6
S	24	5	6	4	18	9	0	0	2	1	0	15	3	5	10	1	5	401	19	16	3	5	52	9
T	8	3	0	0	0	0	0	0	1	0	7	1	0	1	0	0	0	17	528	3	0	1	15	7
U	13	13	0	0	0	1	0	0	2	2	5	5	1	1	0	0	56	8	0	403	13	3	35	7
V	6	2	0	0	4	3	5	0	12	44	11	4	1	2	0	0	10	21	1	39	392	41	14	22
W	2	11	0	4	0	0	0	0	3	24	1	9	0	0	0	0	1	2	13	20	23	453	3	11
X	2	1	0	0	2	0	7	0	2	4	0	17	1	0	0	2	17	30	15	15	8	0	467	12
Y	1	0	0	0	2	3	0	2	5	0	0	6	0	0	1	0	3	18	26	0	1	2	6	518

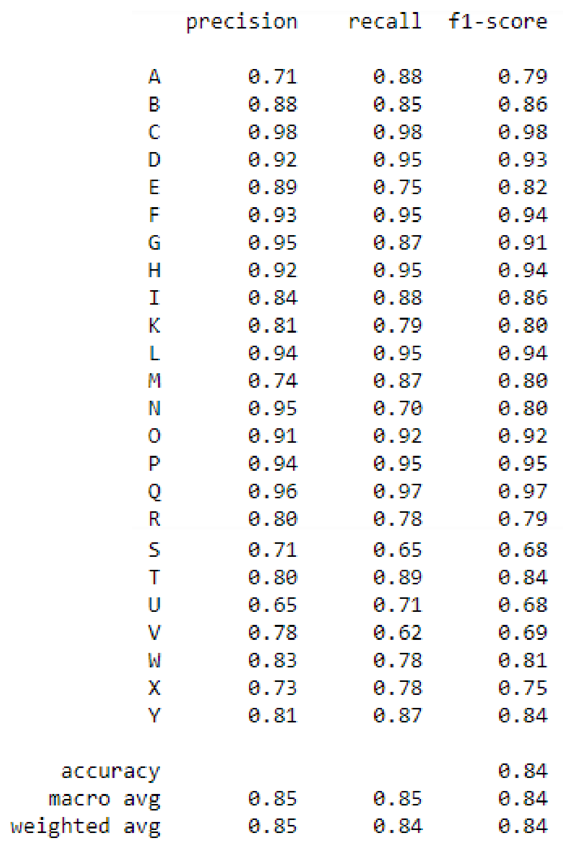
Figure 27: Confusion Matrix

And in (Figure 28) we can see the alphabet in rows and we can also see three main columns which are:

1- Precision: is a performance metric which represents the percentage of the results of the model which are relevant to the model.

2- **Recall:** is a performance metric which represents the percentage total of total pertinent results classified correctly by the machine learning algorithm.

3- **F1-score:** is a performance metric which is the combination of both precision and recall into a single parameter. F1-score is a simple way to compare classifiers. F1-score is also a harmonic metric unit. Which means it gives much more weight to low values, that means that f1-score will get a high number only if both recall and precision are high.



```
              precision    recall  f1-score

           A       0.71      0.88      0.79
           B       0.88      0.85      0.86
           C       0.98      0.98      0.98
           D       0.92      0.95      0.93
           E       0.89      0.75      0.82
           F       0.93      0.95      0.94
           G       0.95      0.87      0.91
           H       0.92      0.95      0.94
           I       0.84      0.88      0.86
           K       0.81      0.79      0.80
           L       0.94      0.95      0.94
           M       0.74      0.87      0.80
           N       0.95      0.70      0.80
           O       0.91      0.92      0.92
           P       0.94      0.95      0.95
           Q       0.96      0.97      0.97
           R       0.80      0.78      0.79
           S       0.71      0.65      0.68
           T       0.80      0.89      0.84
           U       0.65      0.71      0.68
           V       0.78      0.62      0.69
           W       0.83      0.78      0.81
           X       0.73      0.78      0.75
           Y       0.81      0.87      0.84

    accuracy                           0.84
   macro avg       0.85      0.85      0.84
weighted avg       0.85      0.84      0.84
```

Figure 28: Scores

F1-score can be calculated using the next equation:

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

And in (Figure 29) we can see model accuracy and model loss in the train and test phases and how much the differ in the 4 epochs we are using in the model. A model accuracy is used to measure the algorithm’s performance. It measures how accurate the model’s prediction is compared to the true data. A model loss is calculated on training and validation and its interpretation is based on how well the model is doing in them. It is the sum of error made for each example in each set. Loss implies how poorly or well a model behaves after each iteration.

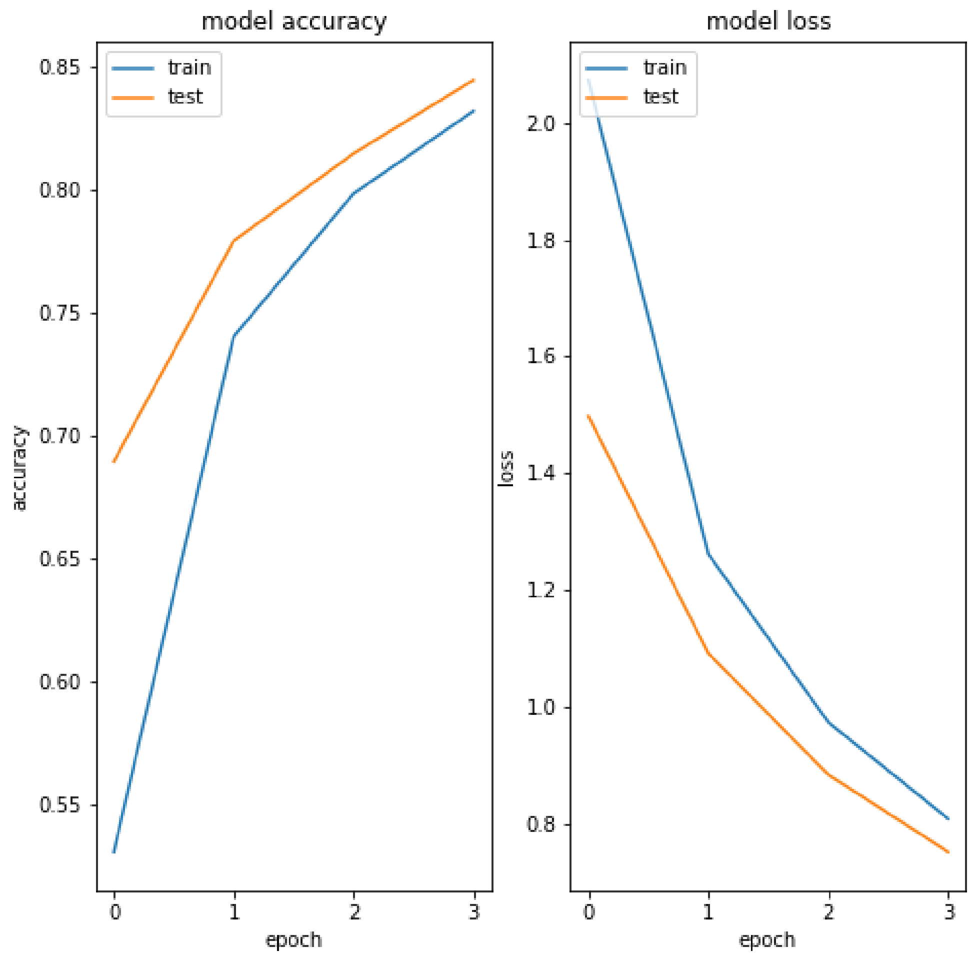


Figure 29: Model accuracy and loss

And in (Figure 30) we can see that the validation accuracy and accuracy are increasing as the number of epochs is steadily increasing which is a sign of a healthy model.

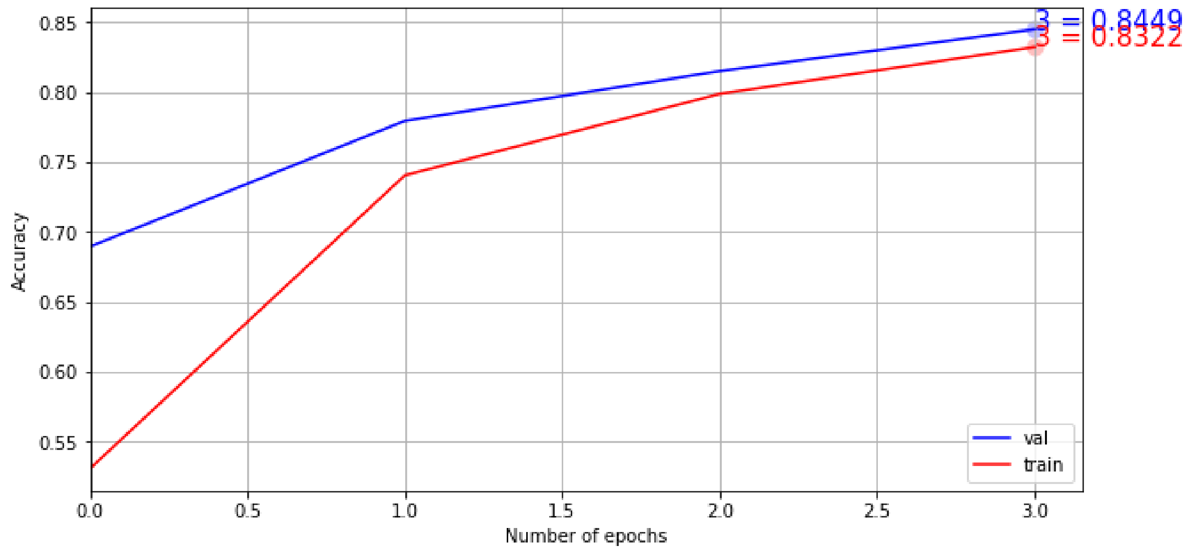


Figure 30: Model validation accuracy and accuracy compared to number of epochs

And in (Figure 31) We can see the prediction of the letter B is done correctly with a pretty high accuracy with little fluctuations.

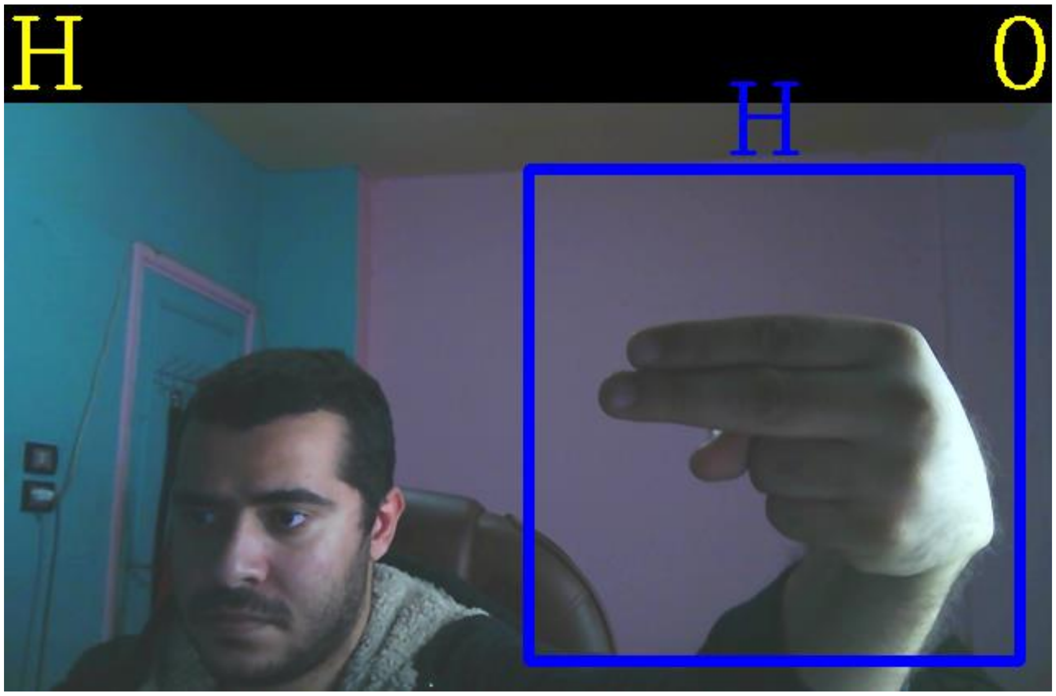


Figure 31: Prediction H

And in (Figure 32) We can see the prediction of the letter O is done correctly with a pretty high accuracy with little fluctuations and it’s the last letter to be done in a word which is “Hello” and by that we can form words by spelling them out. The user is expected to gesture a letter for 3 continuous seconds before the program adds it to a word.

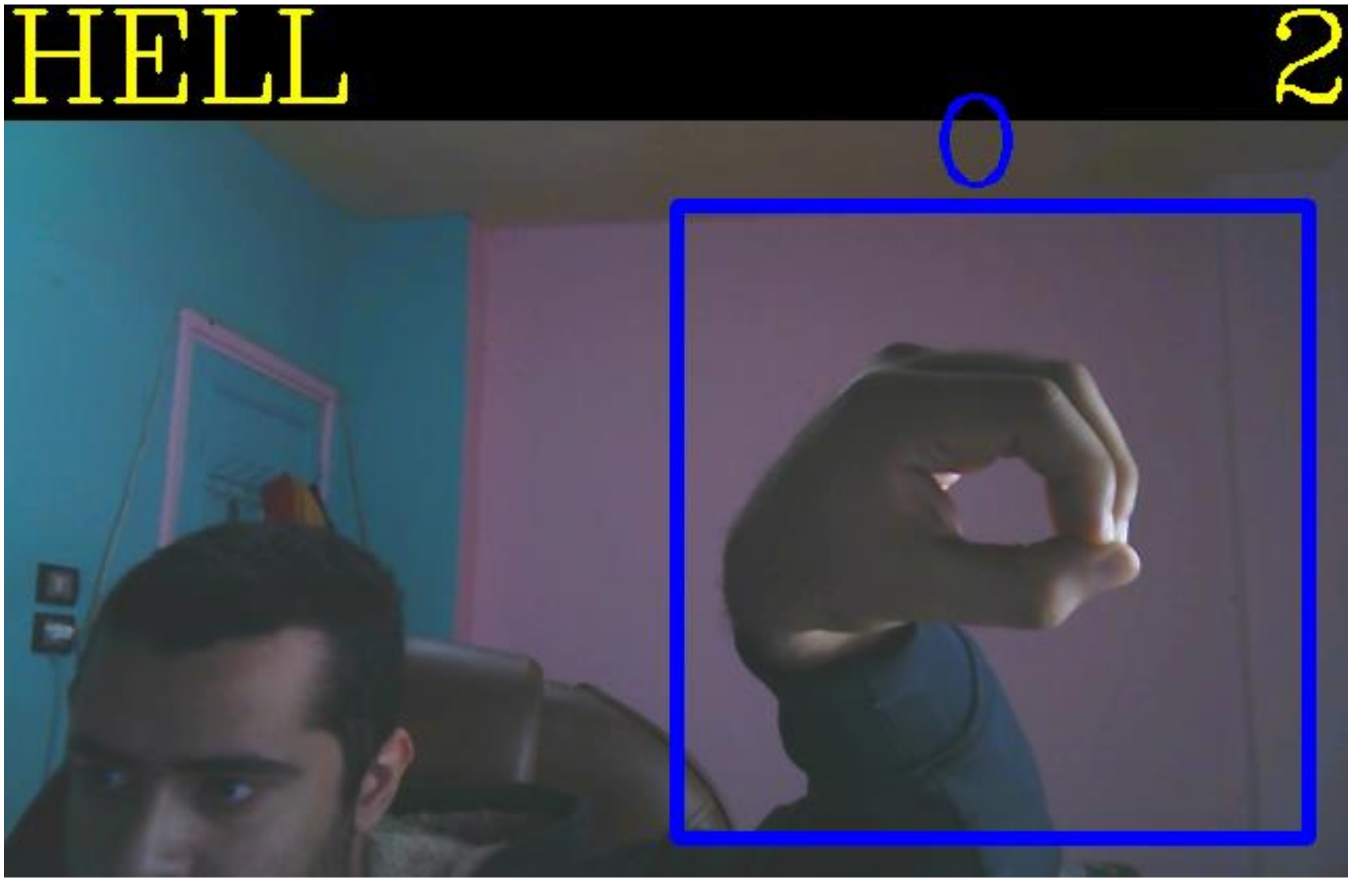


Figure 32: Hello

5.2: Goals achieved:

I have made a sign language gesture recognition system that translates 24 gestures to text and uses those 24 gestures to spell out words. The accuracy of the system is good enough to be used regularly albeit you can only spell words instead of gesturing a whole word. Deaf and dumb people will be able to convey their thoughts with hand gestures without the need for an interpreter.

I have achieved a satisfying number of details that I set out to do in the beginning of the project although I have failed to make a text to speech module using Google's text to speech API due to multiple reasons, one of them that the API has to use the internet and can't be used offline. The other reason is due to time constraints and not being able to optimize it to my satisfaction.

I've indeed put a simple module in place but whenever I pressed the button responsible for saying the text out loud it froze the system for about 2 seconds before saying it making it unusable in those 2 seconds. Time was needed to fix that issue and optimize it but sadly I was running low on time before the date the project was due on.

Overall, I am satisfied with the work done in this project and 1 missing element won't affect all the other existing elements that made this project great in the first place. I have achieved more than 95% of what I considered goals of this project and especially proud that it was done in about two weeks almost from scratch.

5.3: Further work:

This project can use a lot more improvements to the performance and the accuracy. This is done by making a better dataset and by optimizing the layers of the model or the number it’s epochs.

In the future I would like to do the 2 missing letters of the alphabet that are “J” and “Z” which weren’t done here because they require a motion gesture not a still image gesture.

I would also like to add the numbers in the dataset to let the model train on them. 10 more classes should work perfectly fine but the challenge lies in that the numbers’ gestures can be the same as some alphabet gestures which would be hard for the model to differentiate so I have to find a way to integrate it in the same model by maybe making the model learn when to use the number gesture and when to use the alphabet gesture depending on the previous and next sentence.

I would also like to add whole words in the dataset and let the model train on them too. The model should handle any number of extra classes with ease but there’s thousands of whole words out there so I have to choose which words a user would need the most and train it on them because I can’t possibly train the model on every word in existence.

I would also like to add the text to speech function I mentioned earlier since it wasn’t implemented due to time constraints. I might try to make my own text to speech so it doesn’t need the internet to function and optimize it so it wouldn’t impact the software while running.

5.4: Ethical, legal and social issues:

This project faces no ethical, legal or social issues whatsoever.

The project makes it easier for the minority of deaf and dumb to communicate with the hearing person easier and would make the hearing person be able to communicate back with the deaf and dumb without using pen and paper which would be messy and unclean.

I don’t see why this would face any issues mentioned above since it did not steal any work, or break any laws.

Chapter 6: Conclusions:

Sign language is the only middle ground of communication between people with hearing disabilities and the hearing people. While the dumb and deaf people can try to present their thoughts using actions it may not come across to the other end of the interaction efficiently. Which may result in a misunderstanding that may lead to a bigger problem.

In this project we have aimed to convert sign language into text using computer vision which we have been successful in doing.

This software can also use a personal sign language by training the model by these gestures and telling it which gesture is which letter or word. We just need to use the desired sign and it will automatically be converted into text.

This software can currently translate up to 24 gestures in its current state. And can use these gestures to spell out words. In the future we can add more gestures and even whole words to the dataset and train the model on recognizing them. Also, numbers can be added the same way.

Videos are difficult to classify because they can contain both spatial and temporal features. We can use different neural networks to classify each of them and combine it at the end together.

I wish to advance my project further towards recognizing continuous sign language gesture and more complex hand gestures like the 2 alphabet letters “J” and “Z” and whole complex words that use continuous sign language.

This project achieved good accuracies and good results all around and I am very proud to be the one that have done it.

The model consists of 5 layers of convolution and max pooling connected to each other and in the end the model does 4 iterations called epochs to further increase the accuracy.

The model produces a functioning real time vision based American sign language recognition for the deaf and dumb people.

Some of the limitations that this project can face are low light conditions and crowded backgrounds. Also backgrounds with a changing scenery is proving to be difficult to predict. If the scene is dark and there’s a darker skinned user it won’t be able to differentiate between the background and gesture resulting in a bad user experience.

The system will also fail to function properly in accelerated motions and complicated gestures involving motions.

The system also doesn’t work with two hands at the same time and using two hands might confuse the software and it will result in bad accuracy.

References

Aniruddha Bhandari, 2020. *Everything you Should Know about Confusion Matrix for Machine Learning.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/
[Accessed 09 December 2021].

Baeldung, 2021. *Epoch in Neural Networks.* [Online]
Available at: https://www.baeldung.com/cs/epoch-neural-networks
[Accessed 10 December 2021].

Bari, Anasse; Chaouchi , Mohamed; Jung, Tommy;, 2016. *How Predictive Analysis Neural Networks Work.* [Online]
Available at: https://www.dummies.com/article/technology/information-technology/ai/machine-learning/how-predictive-analysis-neural-networks-work-154308
[Accessed 11 December 2021].

geeksforgeeks, 2020. *Python GUI – tkinter.* [Online]
Available at: https://www.geeksforgeeks.org/python-gui-tkinter/
[Accessed 12 December 2021].

Huang, J. et al., 2018. *Video-based Sign Language Recognition without Temporal Segmentation,* Hefei: University of Science and Technology of China.

IBM, 2020. *What are neural networks?.* [Online]
Available at: https://www.ibm.com/eg-en/cloud/learn/neural-networks
[Accessed 02 December 2021].

Iryna Sydorenko, 2021. *What Is a Dataset in Machine Learning: Sources, Features, Analysis.* [Online]
Available at: https://labelyourdata.com/articles/what-is-dataset-in-machine-learning
[Accessed 05 December 2021].

Jason Brownlee, 2019. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks.* [Online]
Available at: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/
[Accessed 10 December 2021].

Jiwon Jeong, 2019. *The Most Intuitive and Easiest Guide for Convolutional Neural Network.* [Online]
Available at: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480
[Accessed 10 December 2021].

Joos Korstanje, 2021. *F1 score.* [Online]
Available at: https://towardsdatascience.com/the-f1-score-bec2bbc38aa6
[Accessed 09 December 2021].

Martin Riva, 2021. *Interpretation of Loss and Accuracy for a Machine Learning Model.* [Online]
Available at: https://www.baeldung.com/cs/ml-loss-accuracy
[Accessed 10 December 2021].

Nahua Kang, 2017. *Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies.* [Online]
Available at: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
[Accessed 03 December 2021].

National insitute on deafness and other communication disorders, 2021. *American Sign Language.* [Online]
Available at: https://www.nidcd.nih.gov/health/american-sign-language
[Accessed 01 December 2021].

NewzHook, 2018. *GnoSys app translates sign language into speech in real time using the power of AI.* [Online]
Available at: https://newzhook.com/story/20387/
[Accessed 17 May 2021].

O'Connor, T. F. et al., 2017. *The Language of Glove: Wireless gesture decoder with low-power and stretchable hybrid electronics,* San Diego: University of California.

Sumit Saha, 2018. *A Comprehensive Guide to Convolutional Neural Networks.* [Online]
Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
[Accessed 05 December 2021].

Techopedia, 2018. *Single-Layer Neural Network.* [Online]
Available at: https://www.techopedia.com/definition/33267/single-layer-neural-network
[Accessed 02 December 2021].

Upasana, 2019. *Difference between Loss, Accuracy, Validation loss, Validation accuracy in Keras.* [Online]
Available at: https://www.javacodemonk.com/difference-between-loss-accuracy-validation-loss-validation-accuracy-in-keras-ff358faa
[Accessed 09 December 2021].