CS 464 – Introduction to Machine Learning

Final Report

Project Name: Sign Language Controlled Password

Group 2

Haider Raheem – 21801244

Muhammad Ali Khan – 21801210

Fahad Waseem Butt – 21801356

Abdullah Hannan – 21801076

Ahsan Mehmood – 21903575

1.0 INTRODUCTION

In this project, we have trained a Machine Learning model which can read and identify different American Sign Language hand gestures by use of pictures. We make use of methods such as Convolutional Neural Networks (CNN), Residual Neural Networks (ResNet50), and VGG16. Making use of this implementation of Machine Learning we worked to make sign language controlled password methodology, with a minimum 4 character length.

With such a goal in mind, our chosen dataset naturally has classes that number to the number of alphabets in the English alphabet. The chosen dataset is greyscale images taken of different hand sign representations of American Sign Language gestures. RGB is not necessary for our model, so our dataset has hence been standardized such that the pixel values lie in the range of 0 to 255, and this results in a better performance of our model. Each hand sign is its own class and there are 40500 total samples and 1500 images per class, including the unknown class.

2.0 PROBLEM DESCRIPTION

In everyday life, there are numerous examples of people who are either born without the capability of speech or hearing, or lose these abilities to age or some unfortunate accident. Sign language is a necessary part of many of these peoples' lives, as they still have a need for communication. While communicating, people with functional speech and hearing find it difficult to understand sign language as it is not a universal language, hence people who rely upon sign language have a need for a translator or translation software. The basis of this project has a foundation in this purpose and we hope to take a step forward by integrating, into the lives of people reliant upon sign language, a crucial part of the average person in their everyday lives which has been normalized in the modern day. Namely, this is access to a password.

This project is based upon the use of biometrics as an authentication method. However, as a facial dataset and fingerprint dataset is unavailable for privacy measures, sign language was used to make a controlled authentication method. This project can help improve further by adding biometric recognition to create a better biometric recognized password.

The fundamental question this project is trying to address is, "Can sign language be used for passwords?"

3.0 METHODS

3.1 Dataset Description

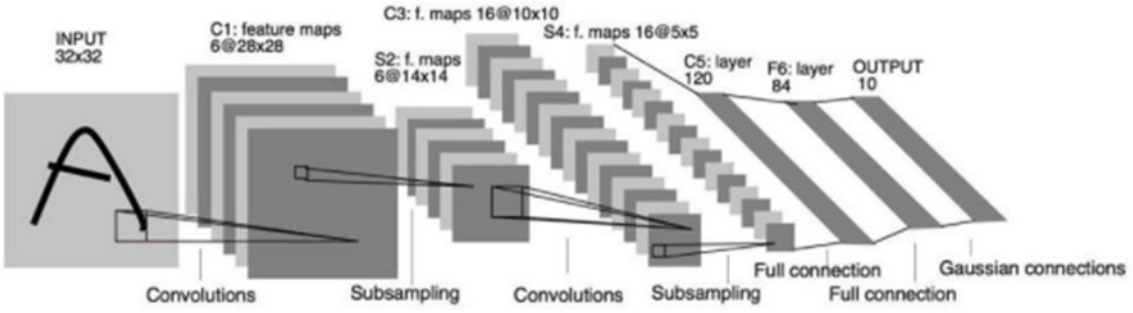
The dataset taken is from Kaggle [1]. It includes 27 classes, namely the sign language representation of the alphabets from A to Z and one additional unknown class. The training set is 40500 images and each class has 1500 images. Each pixel data column has a pixel value between 0 – 255 of a grayscale image. Moreover, our selected models work better if we have normalized data which ranges between 0 - 1 rather than 0 – 255. The normalization was done by dividing the data by 255. The labels were also binarized to make them better understandable for a machine.

Moreover, to expand the dataset, some alterations were done like horizontal flips, vertical flips, random crops, translations and rotations. These alterations were done to avoid overfitting and increase the data, hence helping us increase the accuracy overall. We only performed random rotations between 0 to 100 degrees, randomly zoomed the image and randomly shifted the images horizontally and vertically.

3.2 CNN

Among Deep Learning algorithms, Convolutional Neural Networks are a model that take an image, assign importance, for instance as biases and learnable weights, to objects or aspects of an image, and make use of this for the purpose of differentiating between one from another [2].

The main task of this project is to define the images taken and label them. Hence, for this purpose, we used a CNN architecture to train our model. It is used mainly for image classification and processing. CNN is a type of neural network model which allows us to extract higher representations for the image content [3]. CNN uses raw pixel data to extract the features and then train our model.



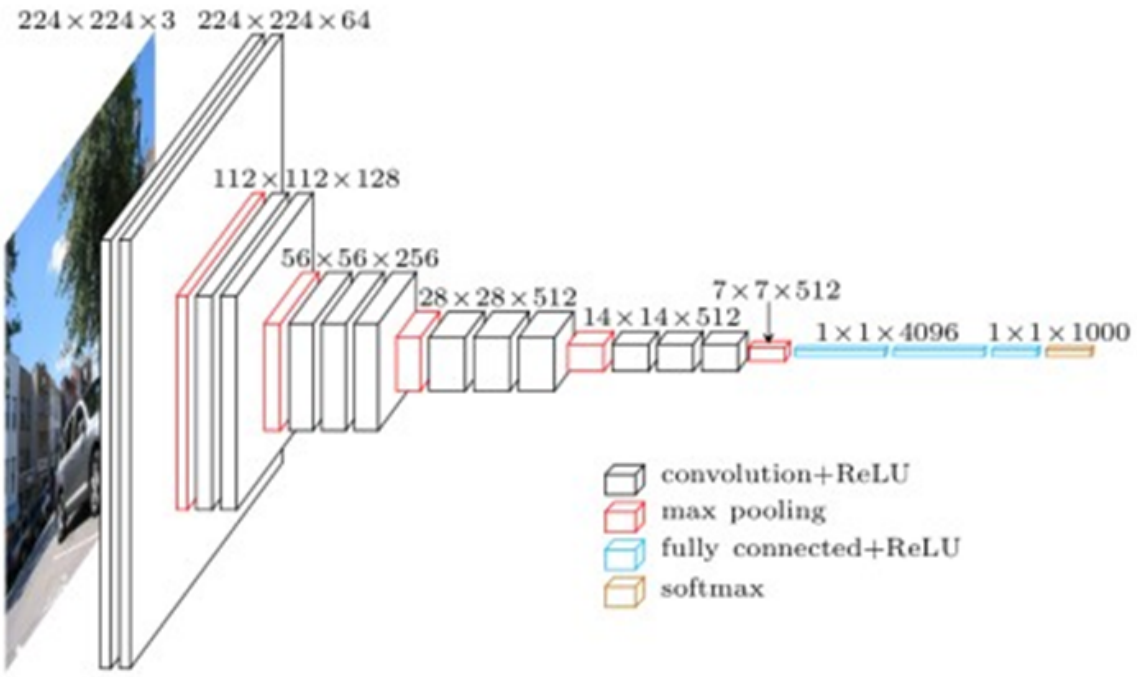
CNN Architecture

As we can see in the figure above, a CNN architecture extracts the features from an image by a combination of Convolution and Subsampling. CNN produces feature maps that are the convolved features. Depending on the filter values, these convolved features will always change [3]. More the filter values, more features will be extracted and hence a better estimation of the image label.

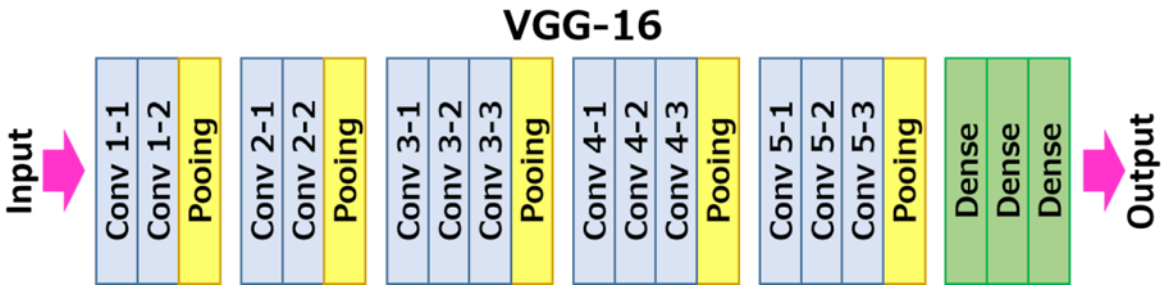
3.3 VGG 16

3.3.1 Background Research

VGG 16 is known for being one of the best image recognition model architectures around. It won the LSVR ImageNet competition in 2014. As the name suggests, VGG 16 is a 16 layer (weighted)architecture. The main idea of VGG 16 is that instead of using many hyper-parameters, use convolution layers of 3x3 with a single stride and a maxpool layer of 2x2 with double stride. The architecture also includes fully connected layers and softmax for the output [4]. The VGG 16 architecture is demonstrated in the image below:



A more organized representation is shown in the image below:



It can be noted from these that there are repeated multiple convolution layers followed by pooling and finally followed by 3 dense layers. Fixed sizes are associated with each layer and the sizes decrease with the progression of layers. The function of the pooling layer is to decrease the spatial dimensions of the representation so that the computation complexity of the architecture decreases with progression. The input RGB image dimensions must be fixed to 224x224.

3.3.2 Implementation of VGG 16

VGG 16 architecture was implemented using transfer learning, otherwise it would have taken a lot of time for training the data. In transfer learning, we basically use the weights of an already pretrained model. The pre-trained architecture's top layer is not used, rather the remaining architecture is appended with custom fully connected layers (dense layers). The rest of the VGG 16 architecture is frozen.

Also, since the dimension of the original dataset was 28x28, the data had to be resized for it to be fit for VGG 16. The data was split into training data and validation data with a ratio of 80:20. 4 layers were appended to the VGG 16 architecture: Flattening layer, 2 dense layers, and one classification layer.

The model that was implemented used Adam optimizer because it gave better results compared to other optimizers like RMSprop, Adagrad, SGD. The loss was calculated using categorical cross entropy loss function which is defined as:

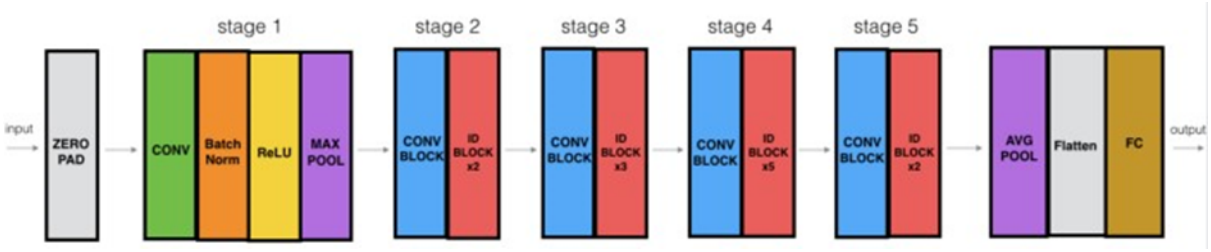
$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

3.4 RESNET 50

3.4.1 Background Research

Resnet basically means Residual Networks. The reason for its importance is that it helps to train very deep neural networks without any vanishing gradient problem as it is back propagated and multiplied repeatedly. The problem is that as more layers are added, more activation functions are added which causes the loss function to approach zero [5]. It solves this problem by using skip connections to connect an output from one of the earlier layers to a layer which is yet to come as shown below:

These connections skip some of the layers that are not able to identify any new information in deeper layers. Since our neural network has about 75 layers, Resnet50 will help if our performance saturates. The main architecture structure is shown in the diagram below:



There are convolution blocks and identity blocks, and each of these has three convolutional layers.

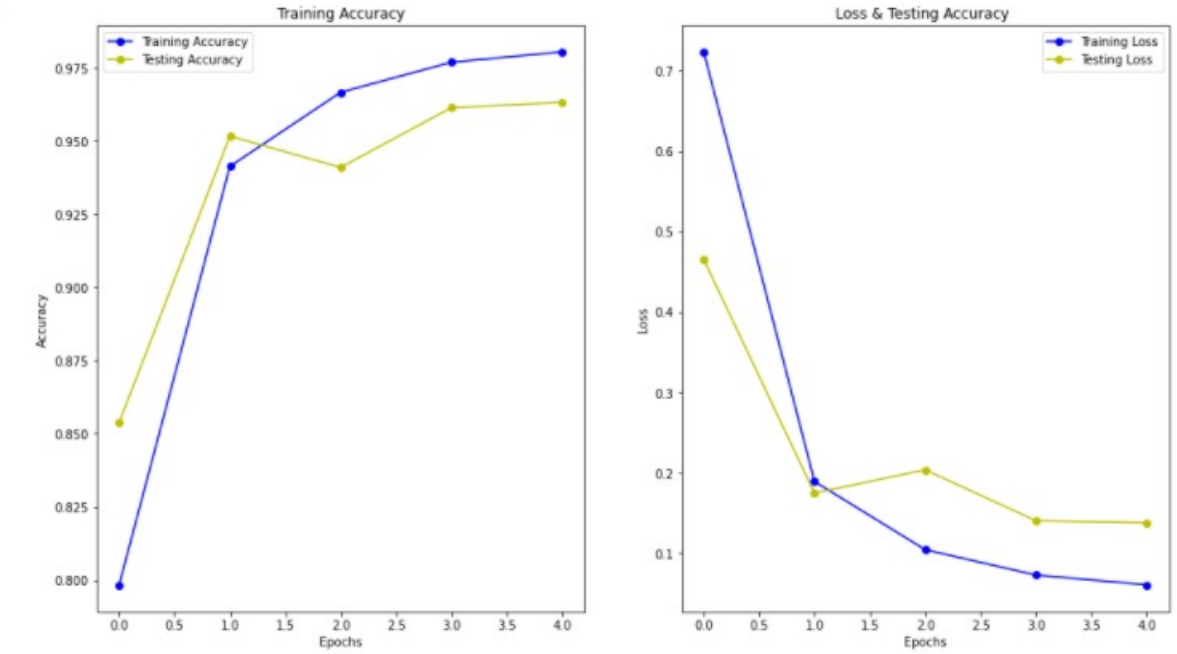
3.4.2 Implementation of ResNet 50

This model was also trained using transfer learning, which as stated in 3.3.2, uses the weights of an already trained model and the dense layers are skipped. The motivation for using this method for learning was to save the training time of the model. The implementation of this model is similar to the implementation of VGG 16.

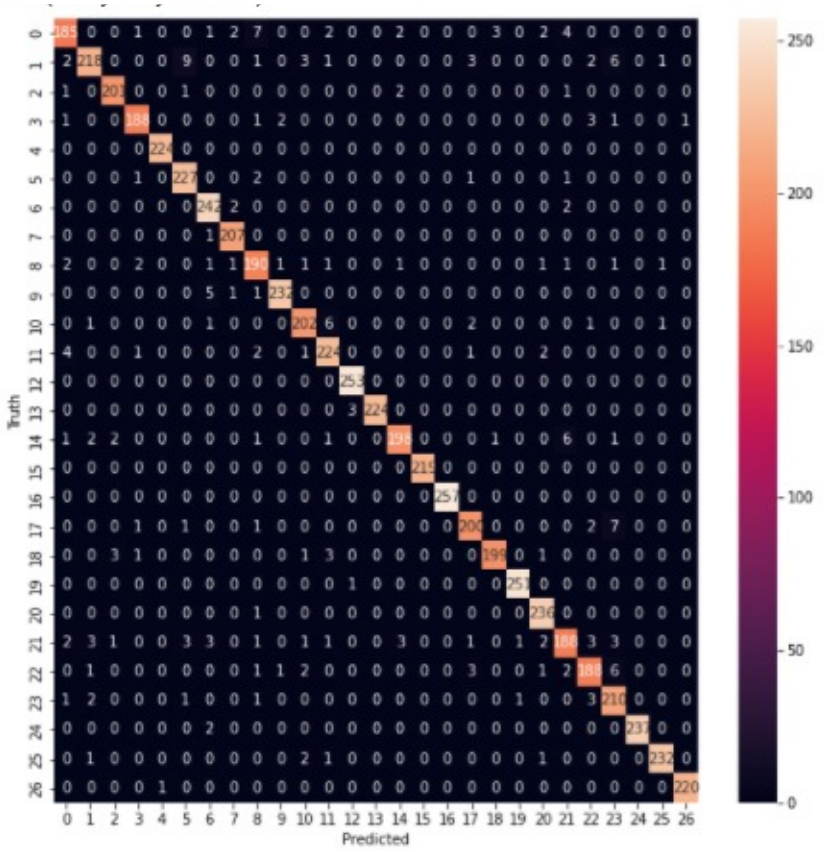
4.0 RESULTS

4.1 CNN Results

The accuracy we received for CNN is 96.32922%. Further, we have made an accuracy and loss graph vs epoch for training and testing. This is shown below:



As we can see, as the number of epochs are increasing, the accuracy increases and the loss decreases. The accuracy and the loss graph stabilize around 4 epochs and we therefore had 4 epochs for the final model. Below are the confusion matrix and performance report for our CNN model.

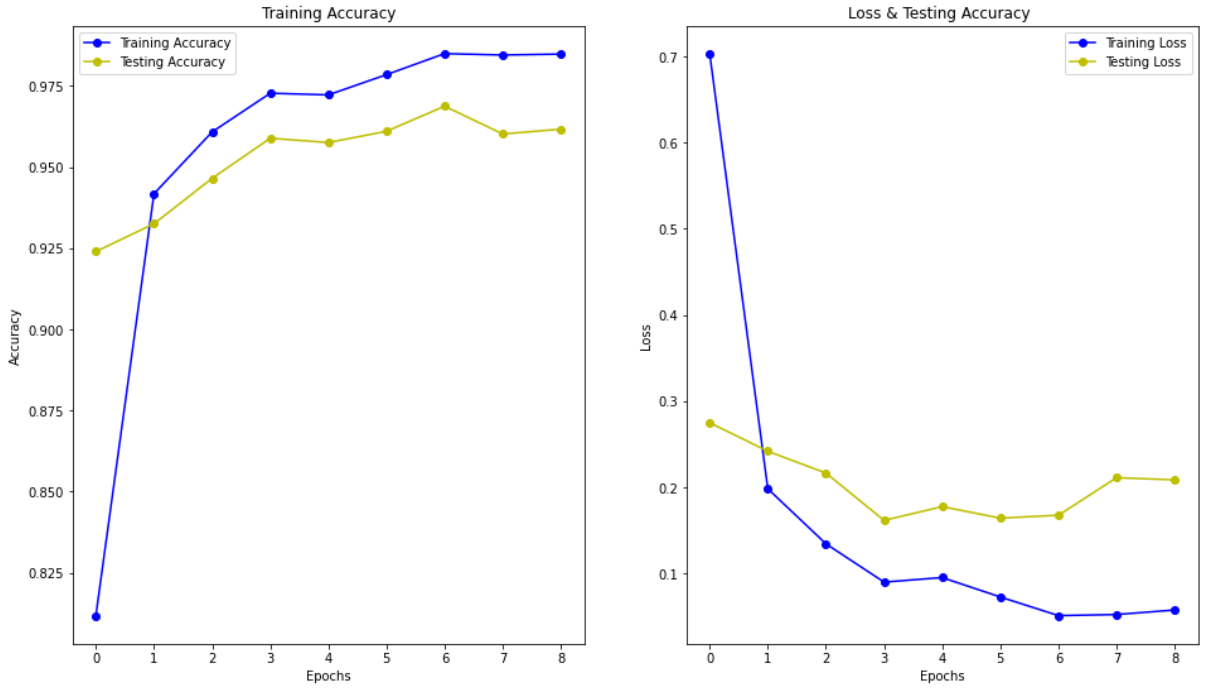


	precision	recall	f1-score	support
a	0.93	0.89	0.91	209
b	0.96	0.89	0.92	246
c	0.97	0.98	0.97	206
d	0.96	0.95	0.96	197
e	1.00	1.00	1.00	224
f	0.94	0.98	0.96	232
g	0.95	0.98	0.96	246
h	0.97	1.00	0.98	208
i	0.90	0.93	0.92	204
j	0.98	0.97	0.98	239
k	0.95	0.94	0.95	214
l	0.93	0.95	0.94	235
m	0.98	1.00	0.99	253
n	1.00	0.99	0.99	227
o	0.96	0.93	0.95	213
p	1.00	1.00	1.00	219
q	1.00	1.00	1.00	257
r	0.95	0.94	0.95	212
s	0.98	0.96	0.97	208
t	0.99	1.00	0.99	252
u	0.96	1.00	0.98	237
unknown	0.92	0.87	0.89	216
v	0.93	0.92	0.92	205
w	0.89	0.96	0.93	219
x	1.00	0.99	1.00	239
y	0.99	0.98	0.98	237
z	1.00	1.00	1.00	221
accuracy			0.96	6075
macro avg	0.96	0.96	0.96	6075
weighted avg	0.96	0.96	0.96	6075

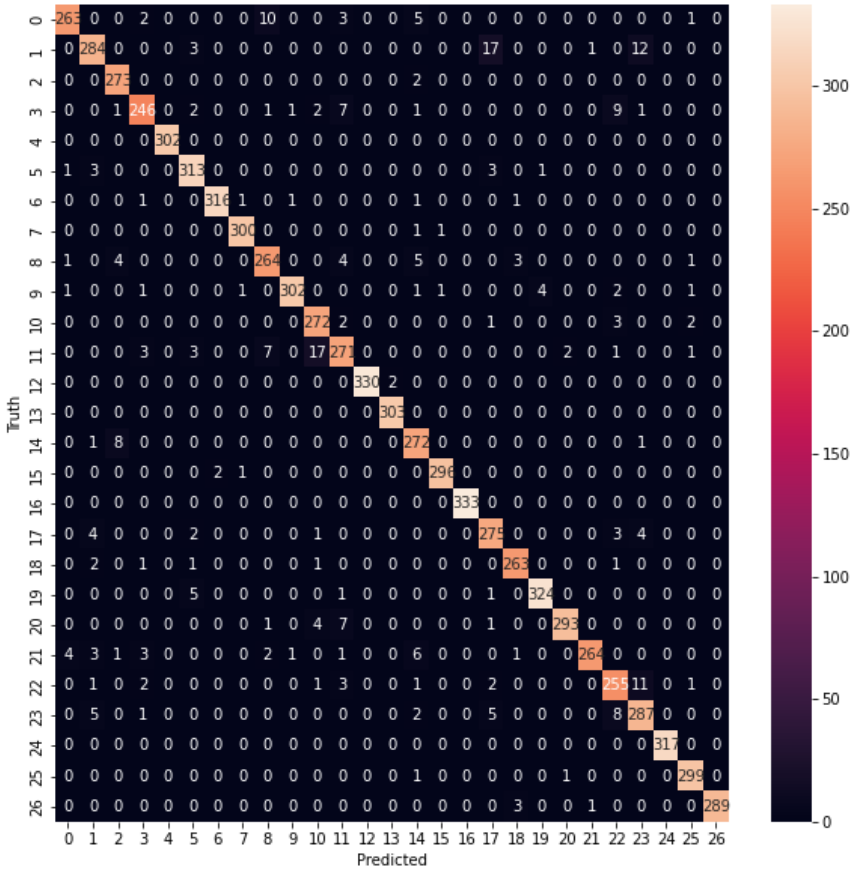
From the performance report, the precision, recall, f1 score, and support can be noted for each individual class as well as the macro and weighted averages along with the total accuracy of this CNN model.

4.2 VGG 16 Results

The accuracy we received for VGG 16 is 96.370 %. Further, we have made an accuracy and loss graph vs epoch for training and testing. This is shown below:



As we can see, as the number of epochs are increasing, the accuracy increases and the loss decreases. The accuracy and the loss graph stabilize around 8 epochs and we therefore had 8 epochs for the final model. Below are the confusion matrix and performance report for our VGG 16 model.

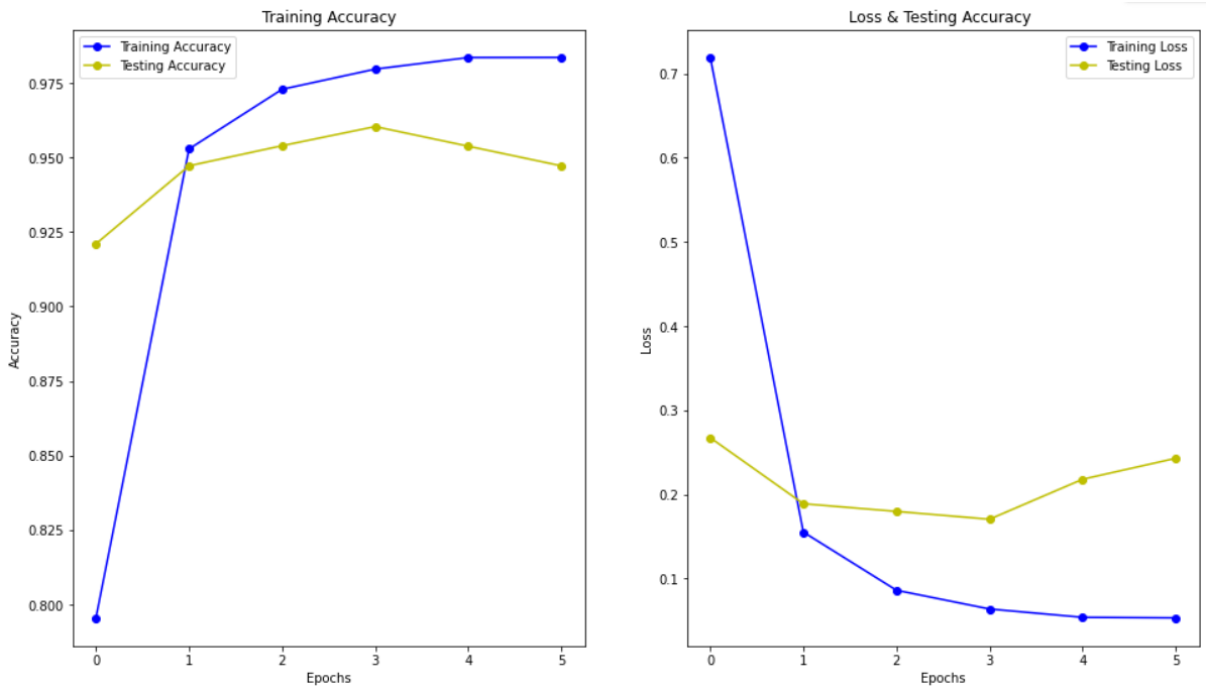


	precision	recall	f1-score	support
a	0.90	0.93	0.92	209
b	0.97	0.88	0.92	246
c	0.96	0.96	0.96	206
d	0.90	0.97	0.93	197
e	0.99	1.00	1.00	224
f	0.95	0.94	0.95	232
g	0.99	0.98	0.99	246
h	0.95	1.00	0.97	208
i	0.89	0.92	0.90	204
j	0.99	0.95	0.97	239
k	0.94	0.93	0.93	214
l	0.90	0.93	0.91	235
m	1.00	1.00	1.00	253
n	1.00	1.00	1.00	227
o	0.95	0.96	0.96	213
p	1.00	0.98	0.99	219
q	1.00	0.99	0.99	257
r	0.94	0.92	0.93	212
s	0.93	0.98	0.96	208
t	1.00	0.99	1.00	252
u	0.98	0.97	0.98	237
unknown	0.97	0.93	0.95	216
v	0.92	0.88	0.90	205
w	0.91	0.93	0.92	219
x	0.99	1.00	1.00	239
y	0.98	0.98	0.98	237
z	1.00	0.99	0.99	221
accuracy			0.96	6075
macro avg	0.96	0.96	0.96	6075
weighted avg	0.96	0.96	0.96	6075

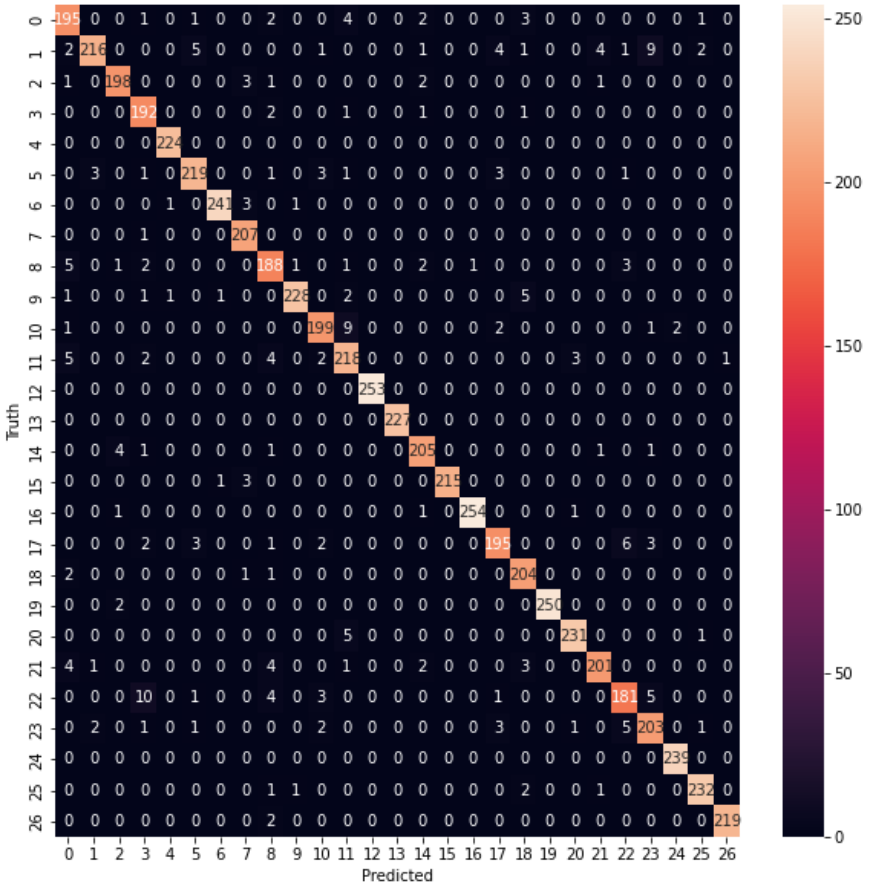
From the performance report, the precision, recall, f1 score, and support can be noted for each individual class as well as the macro and weighted averages along with the total accuracy of this VGG 16 model.

4.3 ResNet50 Results

The accuracy we received for ResNet50 is 96.03292181% Further, we have made an accuracy and loss graph vs epoch for training and testing. This is shown below:



As we can see, as the number of epochs are increasing, the accuracy increases and the loss decreases. The accuracy and the loss graph stabilize around 5 epochs and we therefore had 5 epochs for the final model. Below are the confusion matrix and performance report for our ResNet50 model.



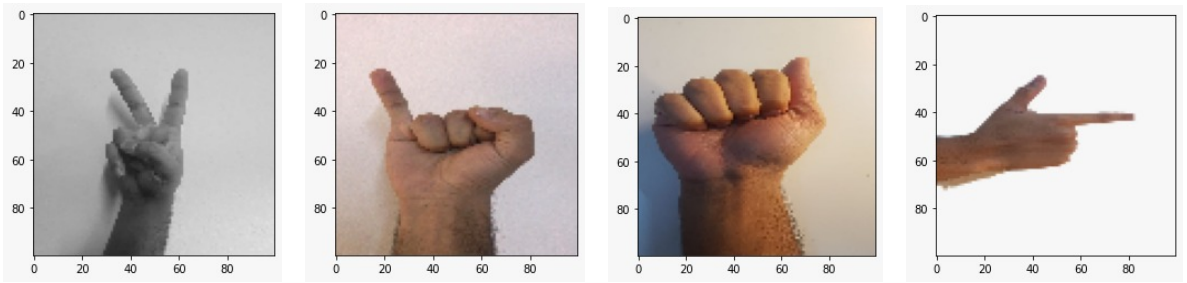
	precision	recall	f1-score	support
a	0.90	0.93	0.92	209
b	0.97	0.88	0.92	246
c	0.96	0.96	0.96	206
d	0.90	0.97	0.93	197
e	0.99	1.00	1.00	224
f	0.95	0.94	0.95	232
g	0.99	0.98	0.99	246
h	0.95	1.00	0.97	208
i	0.89	0.92	0.90	204
j	0.99	0.95	0.97	239
k	0.94	0.93	0.93	214
l	0.90	0.93	0.91	235
m	1.00	1.00	1.00	253
n	1.00	1.00	1.00	227
o	0.95	0.96	0.96	213
p	1.00	0.98	0.99	219
q	1.00	0.99	0.99	257
r	0.94	0.92	0.93	212
s	0.93	0.98	0.96	208
t	1.00	0.99	1.00	252
u	0.98	0.97	0.98	237
unknown	0.97	0.93	0.95	216
v	0.92	0.88	0.90	205
w	0.91	0.93	0.92	219
x	0.99	1.00	1.00	239
y	0.98	0.98	0.98	237
z	1.00	0.99	0.99	221
accuracy			0.96	6075
macro avg	0.96	0.96	0.96	6075
weighted avg	0.96	0.96	0.96	6075

From the performance report, the precision, recall, f1 score, and support can be noted for each individual class as well as the macro and weighted averages along with the total accuracy of this ResNet 50 model.

4.4 Password Implementation

Since the purpose of this project was to implement a password with the use of sign language, we gave the models sample passwords and for each of the passwords we got the desired results. An example of this is shown using the images below.

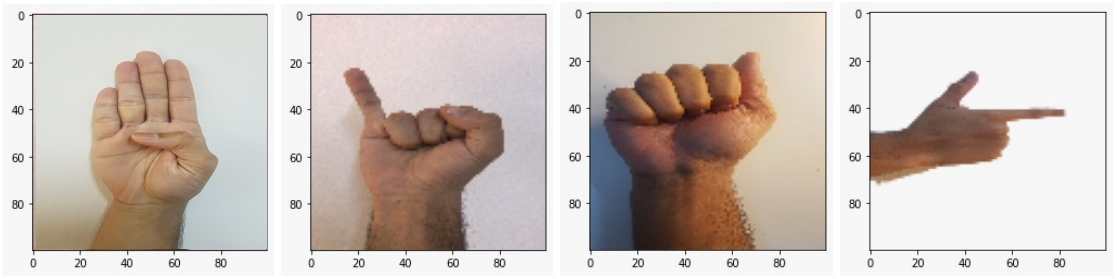
The correct answer for this example is chosen as V I A Z.



V I A Z

When given the above input, we got the correct results for each model, which is that the password was correct.

Again, The correct answer for this example is chosen as V I A Z.



B I A Z

When given the above input, we got the correct results for each model, which is that the password was incorrect.

5.0 DISCUSSION

5.1 CNN Discussion

The results of the CNN implementation show that the number of epochs in general did not have too much of a bearing on accuracy, and the reason for this is because we used the ADAM optimizer and so it converged at a relatively quick rate, which was at 4 epochs. Currently the best model we have trained, as can be seen in the results, is the state-of-the-art VGG 16. However, it is possible that if we used a different optimizer and were able to train the model for more epochs, we may have higher accuracy. Of course, since the accuracy is at an astounding 96.32922%, it was a satisfactory result, as more epochs would also mean that the CNN model would take more time to be trained. So while the accuracy of the CNN is good, and may have some room to improve, we determined that the VGG 16 model was still the best model we could implement.

5.2 VGG 16 Discussion

We have covered the background and implementation of the VGG 16 model in the prior sections and noted that it works using transfer learning. This was done in the interest of saving time spent in training the model, and was done by using the weights of a pretrained model. Our results indicate clearly that this method of implementation took much less time. However it is worth mentioning that the VGG 16 model still took more time to train than the ResNet50 model, so it is not the best in terms of speed. However, the state-of-the-art VGG 16 model is the best performing model and boasts the highest accuracy out of the three models we trained.

5.3 ResNet50 Discussion

As was discussed prior, the ResNet50 model is trained by the use of transfer learning, where we use the weights of a pre-trained model, with the dense layers being skipped. As such, the ResNet50 model is chosen for the purpose of saving time, and as can be noted from the results, the training time of the ResNet50 model is much less when compared with VGG 16, so this goal was successfully accomplished. It is already trained on the ImageNet dataset which means that this is also an appropriate model to be used on our image dataset, and this can further be confirmed from the results. It is noted that ResNet50 has the minimum error probability on the ImageNet.

6.0 CONCLUSIONS

The question asked in this project was, “Can sign language be used for passwords?” Based on the results we have obtained, it is safe to conclude that we have successfully answered this question - yes, sign language can be implemented for usage in passwords. Our models correctly analyzed and identified the sign language gestures, hence strings of alphabets in sign language can be used to form customized passwords. Of course, in the end we made sure to keep only the best performing model, which was VGG 16.

In this project, we researched and learnt about three different machine learning methods - Convolutional Neural Networks (CNN), Residual Neural Networks (ResNet50), and VGG16. For the sake of implementation, we needed to understand these models. The detailed workings of each of these three models has been covered in prior sections but it was found that for us, a model such as VGG 16, which makes use of transfer learning, worked best.

When thinking of future directions this project can move towards, since the basis is a password system, which is a form of security and authentication, we can do so by using biometrics and facial recognition. This would allow users to either be able to choose between multiple security options, or layer them together for tighter security.

7.0 APPENDIX

Ahsan Mehmood re-implemented and optimised the already existing CNN model in order to improve its working capability even further beyond, as well as provided feedback and support in the implementation of the VGG 16 training model.

Fahad Waseem Butt and Muhammad Ali Khan worked on the ResNet50 training model in order to find possibly better results than the other models being implemented. These two also assisted in the process of using the VGG 16 model.

Haider Raheem and Abdullah Hannan executed the VGG 16 training model. They also had a similar aim of improving the accuracy of the other models. Haider Raheem also assisted in the CNN model, and Abdullah Hannan also assisted in the ResNet50 model

All of us worked collectively to implement the algorithm for the password after determining the best model we had trained. We divided sections of the report and collectively finished the sections assigned to us, and did the same when handling the presentation.

8.0 REFERENCES

[1] "Sign Language for Alphabets: Hand Gesture Recognition Datasets for Alphabets," Kaggle. 14 November, 2019. [Online]. Available: https://www.kaggle.com/muhammadkhalid/sign-language-for-alphabets. (Accessed 26 October, 2021).

[2] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way," Towards data science. 17 December, 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (Accessed 13 November, 2021).

[3] V. Tatan, "Understanding CNN (Convolutional Neural Network)," Towards data science. 23 December 2019. [Online]. Available: https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4. (Accessed 20 November, 2021)

[4] "VGG16 – Convolutional Network for Classification and Detection," Neurohive. 20 November, 2018. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. (Accessed 25 November, 2021)

[5] P. Dwivedi. "Understanding and Coding a ResNet in Keras," Towards data science. 4 January, 2019. [Online]. Available: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33. (Accessed 26 November, 2021)