Implementation of Security Access Control using American Sign Language Recognition via Deep Learning Approach

Jimmuel P. Tolete
Batangas State University - Alangilan Campus
Batangas City, Philippines
jimmuel.tolete@g.batstate-u.edu.ph

I. Introduction

Sign language is a special type of communication which contains a collection of different gestures and postures for communicating with the deaf and mute people. These gestures can be made with hands which signify meaningful messages. Thus, these hand gestures become a natural mean of communication. Signals made from hand gestures can also be used in several applications like system control, Augmented Reality (AR), gaming, robotics, vision based applications etc. From those mentioned applications, sign language interpretation through computer vision hasn't been implemented as a security access control which can be a potential higher level of authentication and better administrative measures.

A modern method for recognizing ASL alphabet has been proposed in [1] which used a low-cost depth camera called Microsoft's Kinect. This method achieved about 90% accuracy to recognize 24 ASL static alphabet signs. In [2] a system has been developed to detect hand sign based digits and speaking out the results in Bangla language automatically. The system core has been built with CNN based deep learning model with a result about 92% accuracy. A deep convolutional network based classifier has been presented in [3] that classifies both the images of letters and digits in ASL with a result of 82% accuracy on the alphabet gestures and with a 97% validation set accuracy on digits. A detailed survey done on subsisting techniques has been presented in [4] that has been used for hand written digit recognition. The paper focused on the application of Neural Network (NN) and the modified algorithms of it. It gives an overview of the present research study of NN algorithms in the field of hand written digit recognition. Contour-SVM based method and CNN method under various conditions has been compared in [5] which has been implemented for recognition of ASL 0-9 digits. The contour-SVM based method achieved 69% and the CNN based approach achieved 98.31% accuracy. In [6] a CNN based model has been proposed to recognize Indian sing language gestures. Their proposed model achieved 92.88% accuracy.

In order to create such security access control, the method of interpreting the input should be reliable. From those mentioned related studies, the results may not be enough to achieve the standards of creating a logical smart access control, which can be degrading to the social impact of this kind of method.

This study's goal is to deploy Deep Convolutional Neural Network that can interpret the American Sign Language as input and create a functional control access system. Specifically, the YOLOv3 method is proposed in this work, which has the advantage of higher accuracy yet requiring minimal technical effort. The said method has its algorithm to classify the hand sign language, model training and testing via the custom dataset. The validation scoring utilized for object detection is mAP (mean average precision) and the higher score means higher confidence to implement it as the security control access system.

This proposed study will be beneficial to such firms needing security controls like military, industries, guarded facilities, and even personal restricted areas. The created acess control system uses hand gesture which signifies its respective letter. Although, technically this study uses letter permutation as the password, it is not restricted to the letters it but it can be a unique gesture that the legitimate user only knows.

In this paper we worked with the American Sign Language which is based on Phonetic alphabet currently used in USA, Canada, as well in English speaking countries, and here in Philippines commonly for educational use and communication. In ASL there are originally 26 letters but the letter J and Y has gesture movement. YOLOv3 can only train static images therefore a part of the movement is replaced for the latter.. Online free GPU is provided by GoogleGolab and not all picture

II. Methodology

The workflow of the system is depicted in Fig. 1. The system's development or design went through several steps or stages, including dataset preparation, data splitting, model training and evaluation, and finally, model inferencing together with the access control system.

A. Dataset Preparation

The image dataset is downloaded from an open-source contributor from Kaggle. In this study only 80 pictures out of 3000 per letter is collected. The images were already cropped at a size of 200x200 pixel. A method is made such that pictures are randomly selected to ensure that such image has great significant difference compared to other image within the same classification. 80% weent to train dataset and 20% went to validation. Fig.2 are the sample selected images.

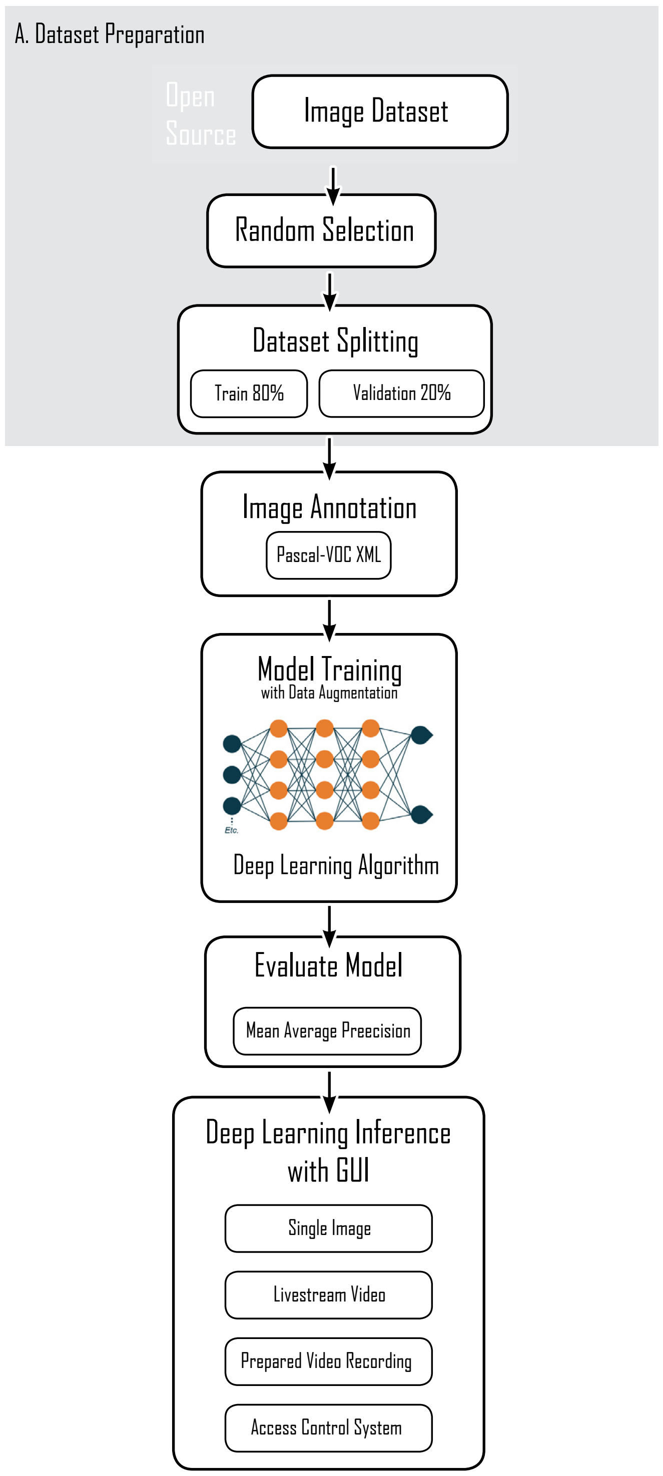


Fig. 1. Diagram of the system workflow

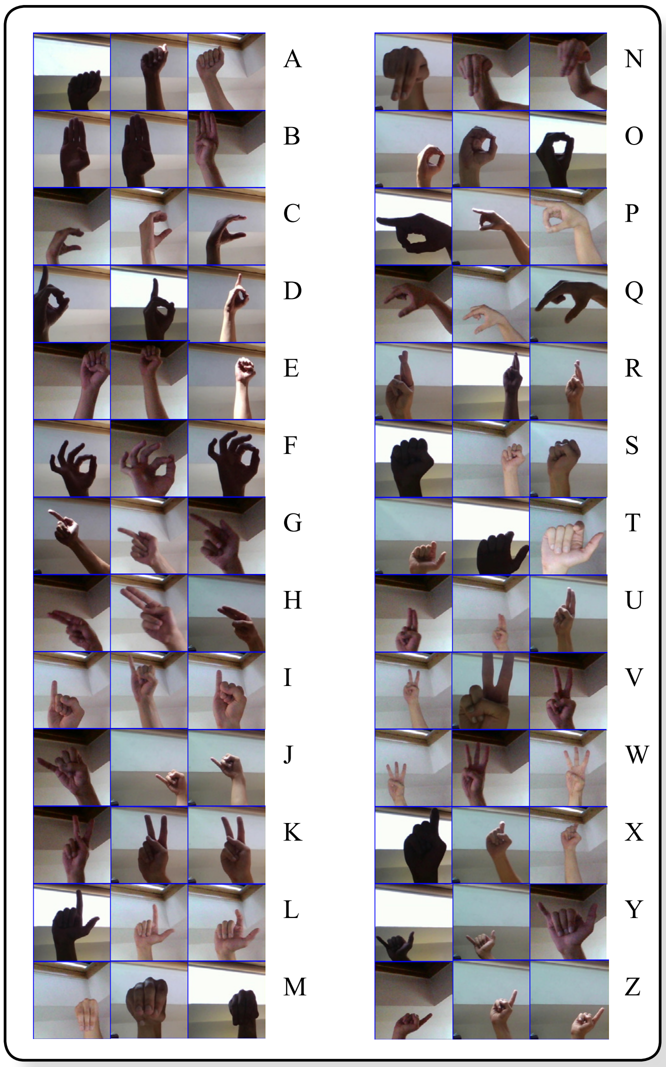


Fig. 2. Sample Datasets per Letter

B. Dataset Annotation

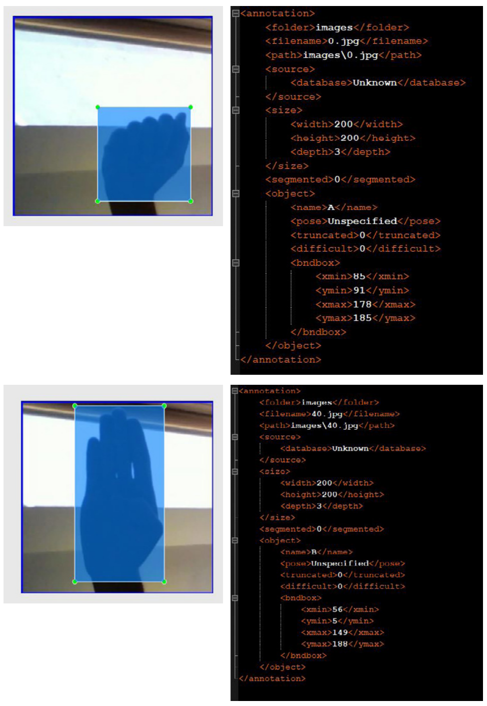
To annotate images, an open source program written in Python called LabelImg.py was utilized. This annotator program provides the PASCAL Visual Object Classes format, which is the format used for this study. It has a convenient method wherein it automatically writes the coordinates of the corners where the target object is. It produces one XML file per image. A sample image with its corresponding XML content is shown at Fig.3.

C. Deep Learning Algorithm

The YOLOv3 is recognized for being efficient, accurate, and fast, in terms of mean average precision (mAP) and intersection over union (IOU) scores. Studies also shows it executes significantly faster other than detection meethods.

YOLO's architecture resembles with Convolutional Neural Network (CNN) for demonstrating objct dtection in real-time. CNN is a refereence to classifier-based system because its process took input images as structured arrays of data and distinguish patterns among them. YOLO's advantage over other architecture is its shorter time of detecting but accuracy is maintained. Such model created by YOLO is able to look at the whole image at test time, therefore its prediction are informed by the global context in the image. YOLO and other CNN algorithms "score" regions based on resemblance to predefined classes. This is the case why YOLOv3 will be mainly used for this study for ease of use.

During model training, data augmentation was implemented; a technique used to increase the amount of data by adding slightly modified copies of already existing data. This helps to avoid overfitting when applying model training. The augmentation utilzed four filtering methods that help the model perform better: scaling, cropping, HSV distortion, and



```
<annotation>
    <folder>images</folder>
    <filename>0.jpg</filename>
    <path>images\0.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>200</width>
        <height>200</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>A</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>85</xmin>
            <ymin>91</ymin>
            <xmax>178</xmax>
            <ymax>185</ymax>
        </bndbox>
    </object>
</annotation>
```

```
<annotation>
    <folder>images</folder>
    <filename>40.jpg</filename>
    <path>images\40.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <sizo>
        <width>200</width>
        <hoight>200</hoight>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>B</name>
        <pose>Unspeci fied</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>56</xmin>
            <ymin>5</ymin>
            <xmax>149</xmax>
            <ymax>188</ymax>
        </bndbox>
    </object>
</annotation>
```

Fig. 3. Image Annotation with XML file

image flipping.

D. Model Evaluation

As every epoch made, it produce models that is shown to have lower losses in every iteration. Yet, the loss value does not pertain that it gives a better confidence to predict the hand gestures. To have a better definition of model's efficiency especially when inferencing, the mAP was utilized to compare the trained models, also for documentation purposes whether there could be a relationship between the model's loss to mAP scorees.

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above per class. mAP (mean average precision) is the average of AP.

By computing a precision and recall at every position in the ranked sequence of documents, one can plot a precision-recall curve, plotting precision *p(r)* as a function of recall {\ displaystyle *r*. Average precision computes the average value of *p(r)* over the interval from *r = 0 to r = 1*.

$$\text{AveP} = \int_0^1 p(r)dr$$

That is the area under the precision-recall curve. This integral is in practice replaced with a finite sum over every position in the ranked sequence of documents:

$$\text{AveP} = \sum_{k=1}^{n} P(k)\Delta r(k)$$

E. Model Inferencing and Testing

Through GUI, the model can be tested in this convenient platform which composed of four platforms; image detection, video detection, live stream detection and access control mode. The GUI is built by using various program

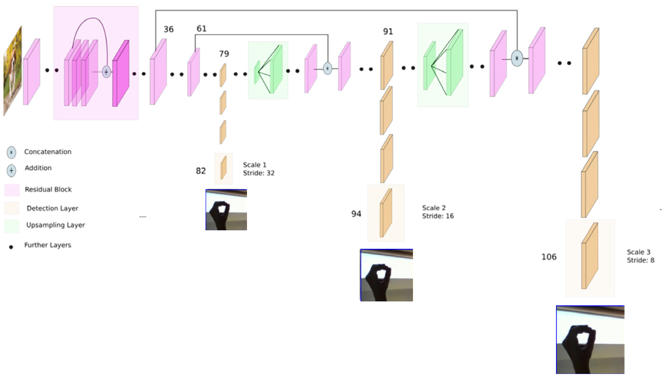


Fig. 4. YOLOv3 Architecture

developing platform which is the Anaconda IDE, and Python programming packages like PyQt5, ImageAI, OpenCV2, and ImageAI detection library.

As for the testing procedure, set of images were made such that those images does not belong to the train and validation dataset images during the training phase. This is to avoid bias.

III. RESULTS AND DISCUSSION

A. Training and Validation Process

Fig.5 shows the dataset training result and validation outcomes. There were 14 epochs produced. It started with a model layer loss of 13.5 and concluded with 2.8.

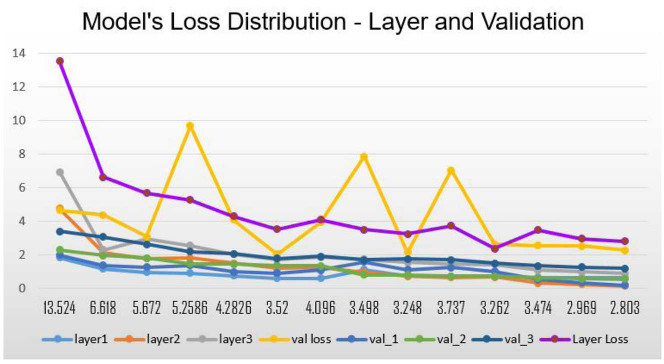


Fig. 5. Model's Loss Distribution

From the graph, the curve indicates as epochs were made the loss of training and validation decrease as it should be; it learns from the dataset provided and learns more as epochs were created. Except for the validation loss (yellow line) it has sudden increasing peak loss but the peak value is lower compared to the previous lone.

B. Evaluation of Model

The models accuracy is described by mAP wherein a 100% or 1 is the maximum possible it can reach. Fig. 7 is a plotting of the model's loss towards it mAP.

The target mAP is 95% above and five models are above this boundary. The highest mAP achieved was 98.69% by the model loss of 2.803. It can be inferred that the lower the model's loss, the higher mAP it can give, an indrect proportion.

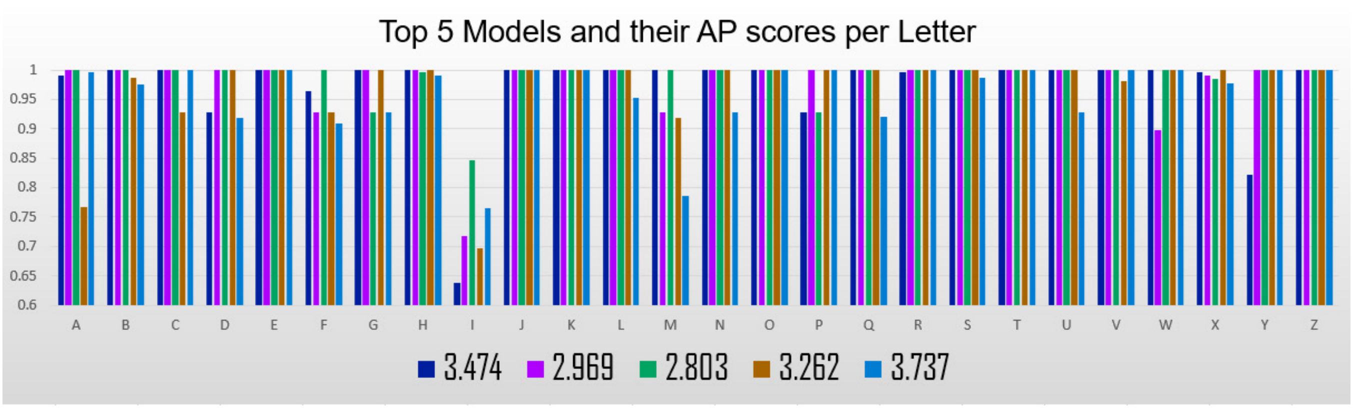


Fig. 6. mAP Scores

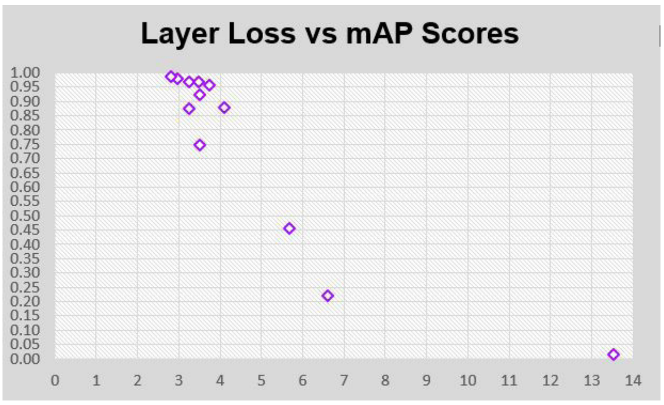
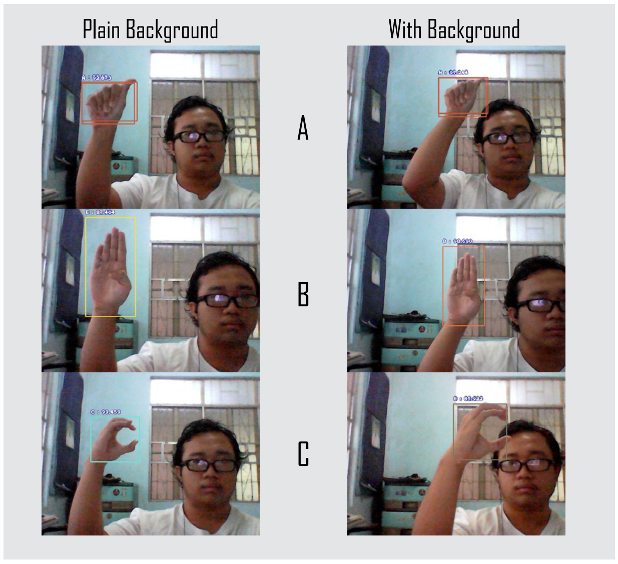


Fig. 7. mAP Scores

Fig. 6 is another visualization of the mAP performance of top 5 models that reached 95% mark. This graph shows the average precision per classification from A to Z. Frequent scores of 1 can be found all throughout the classification except for letter I. The letter I, judging the chart visually, have the lowest average precision score.

C. Inference Training Model

A new dataset is made which is introduced for inferencing the trained model. Sample dataset is shown at Fig. 8 with the detection boxes and its classification score.



The new dataset consists of 2 sets of images from A to Z which is the hand performing gesture placed on the area with plain background and with background (w/BG). The top three models were used to test the models against the new dataset.

The detection library is able to generate the detection boxes where it guesses where the hand is in the image and what letter it performs. It also shows how much that classified hand is similar to what it knows in percentage. Settings are configured wherein the percent score will only accept 80% above or else it is dropped to zero. Fig. 8 is the graph that pertains to this matter,

Fig.8 performance is entirely different compared to performance from Fig.7 that uses the dataset that belongs to the training-validation process. According from Fig.8, some hand signs can not be detected or perhaps maybe the score was lower than 80%. A great example is using model loss of 2.969 that can not detected the letter ‘G’ the hand sign.

To summarize the Fig.8, Table 1 shows the numerical average of the models performance.

Model Loss	Plain (%)	w/BG (%)
2.803	97.95	96.97
2.969	94.48	94.40
3.474	97.63	85.76

Table 1 Average Score of Models

The model with loss of 2.803 shows the higherst average scores of 97.95% and 96.96% for plain and w/BG respectively. The model loss of 3.474 has the poorest among all. Therefore it is safe to say to use model 2.803 to be utilized with the built GUI and so for the security access control system

Fig. 9 shows what the GUI looks like with the four choices icons.. It can do import image and video from the local storage, make live feed, and perform its function as an access control system. The minimum percentage is configured to 92%. The score results from hand sign detection will always be written in a CSV file.

Fig. 10 shows a pseudocode on making the access control system. It requires another model which is detecting the face who knows the password.

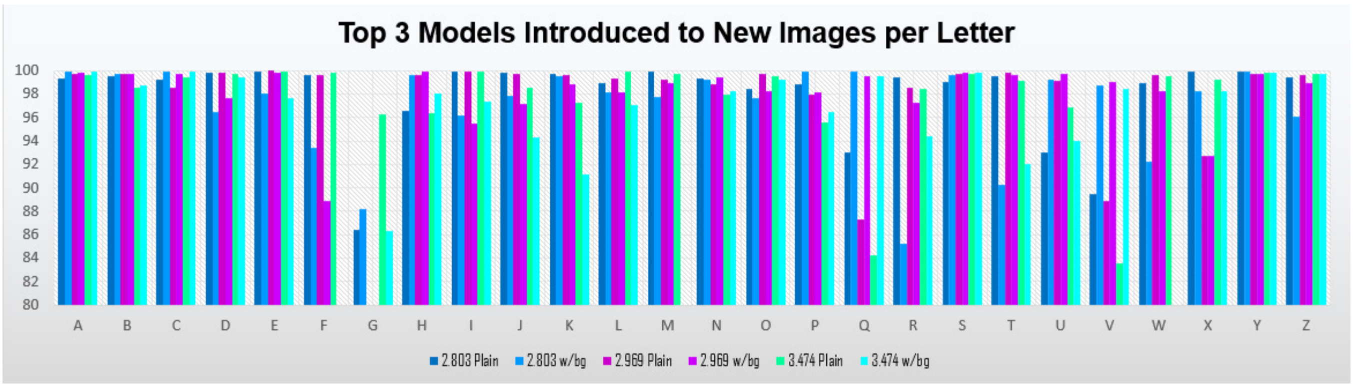


Fig. 8. Models' Performance Against New Dataset

```
function AccCtrl():
    Loop 4 times:
        Display ("Prepare Hand"), delay 3s
        Command cv2 to take single frame
    Load model for person detection
    Loop 4 times:
        Get next image
        Detect person
        Write score in CSV file
    Do 4 CSVs file contains the same person:
        False: Display "You may not enter"
               Break function
        True:
           Load model for sign language detection
           Create empty list
           Loop 4 times:
               Detect sign language from same image
               Write score in CSV file
               Append class having the highest score.
           Does list matches person's password?
               True: Display "You may enter!"
               False: Display "Try again"
```

Fig. 9. Pseudocode for Access Control System

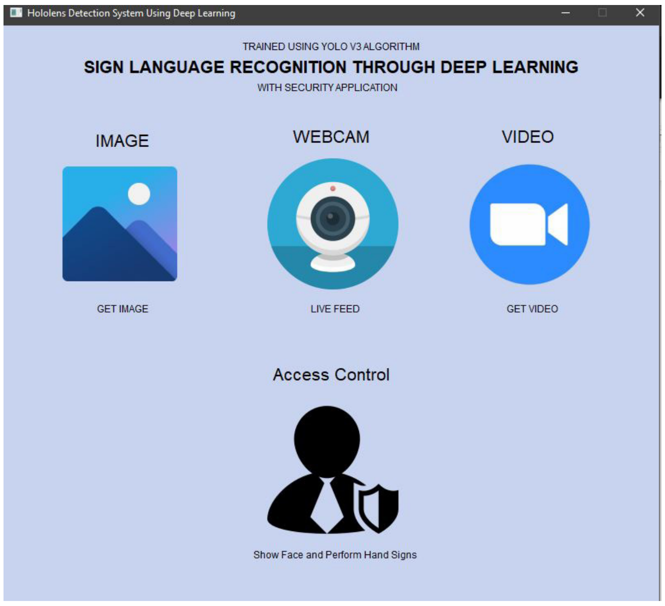

Hololens Detection System Using Deep Learning
TRAINED USING YOLO V3 ALGORITHM
SIGN LANGUAGE RECOGNITION THROUGH DEEP LEARNING
WITH SECURITY APPLICATION
IMAGE
WEBCAM
VIDEO
GET IMAGE
LIVE FEED
GET VIDEO
Access Control
Show Face and Perform Hand Signs

Fig. 9. GUI for Model Inferencing