**Table of Content**

[1.0 AI Part and System Integration 3](#_Toc115217419)

[1.1 Methodology 3](#_Toc115217420)

[1.2 Data Generation and Preparation 5](#_Toc115217421)

[1.3 Model Development and AI training 7](#_Toc115217422)

[1.4 AI testing and prediction results 10](#_Toc115217423)

[1.5 Model problems and solutions 14](#_Toc115217424)

[Conclusion 16](#_Toc115217425)

**Table of Figures**

[Figure 1 YOLO Methodology (Bandyopadhyay, 2022) 3](#_Toc115217446)

[Figure 2 YOLO network architecture (Bandyopadhyay, 2022) 4](#_Toc115217447)

[Figure 3 labelimg program 5](#_Toc115217448)

[Figure 4 YOLO annotation data format 6](#_Toc115217449)

[Figure 5 Connecting to Google Drive 7](#_Toc115217450)

[Figure 6 Cloning Darknet 7](#_Toc115217451)

[Figure 7 Configuring training file 8](#_Toc115217452)

[Figure 8 Creating names and data files 8](#_Toc115217453)

[Figure 9 Save training and objects in Google Drive 8](#_Toc115217454)

[Figure 10 Insert dataset 8](#_Toc115217455)

[Figure 11 Create train.txt file 9](#_Toc115217456)

[Figure 12 Download weights 9](#_Toc115217457)

[Figure 13 Begin model training 9](#_Toc115217458)

[Figure 14 Imports and Read video data 10](#_Toc115217459)

[Figure 15 Loop video and Detect objects 11](#_Toc115217460)

[Figure 16 Label detected objects 11](#_Toc115217461)

[Figure 17 AI Glove Type detection results 13](#_Toc115217462)

[Figure 18 Incorrect Glove Identification 14](#_Toc115217463)

[Figure 19 Output result of an inaccurate model 15](#_Toc115217464)

# 1.0 AI Part and System Integration

## 1.1 Methodology

For the development of the AI part for our system, we have used the YOLO V3 algorithm. The YOLO V3 algorithm is an example of a Convolutional Neural Network (CNN) under the YOLO algorithm family. The YOLO algorithm uses one-stage object detection which allows it to perform object detection with only one fully connected layer, allowing real-time detection at high performance. Two-stage object detectors like the RCNN family of algorithms perform object detection in two stages. The first stage detects the possible regions of interest using a region proposal network, the second stage then performs the recognition on the regions separately. One Stage detection makes the YOLO algorithm faster than the other object detectors in the market. (Bandyopadhyay, 2022) YOLO is also the most popular algorithm and is discussed in many literatures and journals.

A picture containing text, indoor, display, screenshot

Description automatically generated

Figure 1 YOLO Methodology (Bandyopadhyay, 2022)

The YOLO methodology begins with dividing the input image into several grids. The algorithm then proceeds to predict the bounding boxes of the object in each grid, including the probability of the object being present in the grid. Using this operation will result in many duplicated and overlapping bounding boxes on the image and the YOLO algorithm uses Non-Maximal Suppression to address this problem. Non-Maximal Suppression removes the bounding boxes that have lower probability scores. The algorithm we’re using for our AI system, YOLO V3, is an improved version of previous YOLO versions. YOLO V3 has 106 layers together with residual blocks and upsampling networks. YOLO V3’s architecture allows detection at 3 different scales, which improves the detection of smaller objects. (Bandyopadhyay, 2022) The figure below describes the architecture of the original YOLO neural network, which consists of several convolutional layers and 2 fully connected layers.

Diagram, engineering drawing

Description automatically generatedFigure 2 YOLO network architecture (Bandyopadhyay, 2022)

## 1.2 Data Generation and Preparation

For the training data, we have captured videos of 3 different types of gloves, latex gloves, fur gloves, and ski gloves. We then use a python program to capture each frame by a set amount of intervals between each frame. We then save each frame captured in .jpg format and proceed for annotation. The YOLO algorithm requires training image data to have the objects annotated with rectangular boxes. We use a program called LabelImg, which is a graphical image annotation tool. We used the program to draw boxes on the object to be identified and save in YOLO dataset format. The YOLO dataset consists of the training images, annotation data, and a class file. The class file contains all the classes we chose to label the objects, and the annotation data consists of the class index, coordinates, and the height and width of the bounding box. We have prepared 47 images for ski gloves, 35 for latex gloves, and 59 for fur gloves. The figures below show the labelImg program as we are drawing the label box for a latex glove and the YOLO dataset output .txt file.

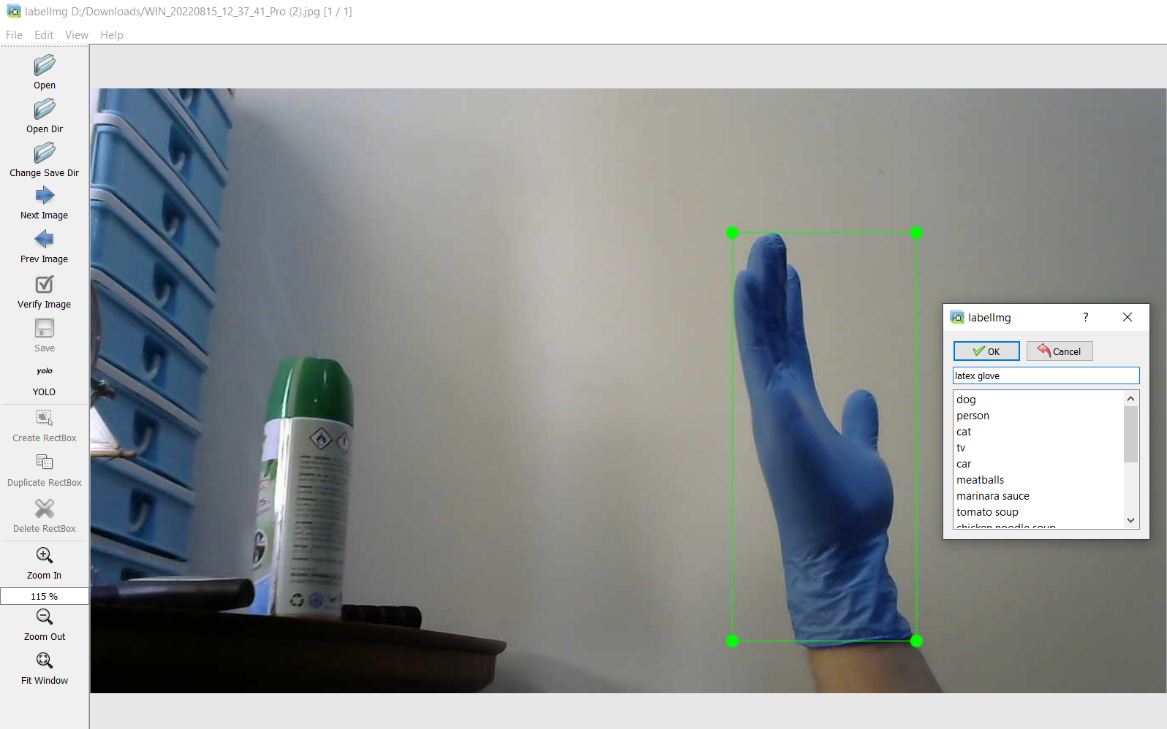


Figure 3 labelimg program

Text

Description automatically generated

Figure 4 YOLO annotation data format

## 1.3 Model Development and AI training

We utilized the Google Colab as the Integrated Development Environment (IDE) for training machine learning model with different glove types. The Google Colab was integrated with the Google Drive where the dataset was uploaded. The uploaded dataset contained of 282 files (141 jpg files & 141 txt files). All the raw image data comes with text files that store the labelled image information.

Text

Description automatically generated

Figure 5 Connecting to Google Drive

First of all, the program checks whether the NVIDIA GPU rented from Google Colab is utilized or not. The Graphics Processing Unit (GPU) is able to boost the model training speed as it enables multiple computations process simultaneously, resulting in high efficiency. After that, the program connects to the Google Drive to retrieve the data in order to train the machine learning model.

Text

Description automatically generated

Figure 6 Cloning Darknet

We cloned the Darknet from GitHub, configure it for the OpenCV, GPU, and CUDA Deep Neural Network (cuDNN), and compile them for the model training process. Darknet is a neural network framework providing real-time object detection technique with high accuracy and fast performance.

Text

Description automatically generated

Figure 7 Configuring training file

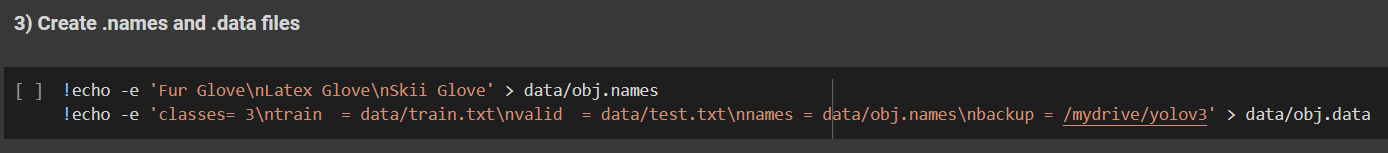


Figure 8 Creating names and data files

Text

Description automatically generated

Figure 9 Save training and objects in Google Drive

After that, we configure the yolov3.cfg to make it specify to the following model training process by altering the batch, subdivisions, maximum batches, classes, and filters. There are three classes defined for classifying the data which are Fur Glove, Latex Glove, and Skii Glove. For the third and fourth steps, we defined the three classes’ name and saved the configuration and classes in separate files for testing the model performance in local device. More details will be explained in the section 6.4 while testing the model performance.

Graphical user interface, website

Description automatically generated

Figure 10 Insert dataset

Then, we create a folder to store the unzipped data from the uploaded dataset. This folder will be used to train the machine learning model in the Google Colab.

Text

Description automatically generated

Figure 11 Create train.txt file

We create a train.txt file to store the image’s path, such as ‘data/obj/test\_32.jpg’. This path will be helpful for the following model training process.

Graphical user interface, website

Description automatically generated

Figure 12 Download weights

After that, we download the convolutional weights which are pre-trained on Imagenet layers file. These pre-trained weights can boost the model training process as it provides the basic weights for a general machine learning model.

Text

Description automatically generated

Figure 13 Begin model training

Finally, we start the model training by executing the darknet program. We have two options to start the training process, which are starting with or without the previous trained weight file. By training with the weight file, the file must be available in the Google Drive which is connected to this program.

## 1.4 AI testing and prediction results

The following section will be explaining on the AI model testing process using the trained weight files with the YOLO V3 configuration file.

Text

Description automatically generated

Figure 14 Imports and Read video data

First of all, we imported the necessary libraries and loaded the DNN model downloaded from the Google Drive where stores the training results from the Google Colab. ‘Classes.txt’ stored the three classes (Fur Glove, Latex Glove, Skii Glove) used to label the data. Then, the program read the video data from the local device via the provided video path.

Text

Description automatically generated

Figure 15 Loop video and Detect objects

After that, we start looping the video and label the object within the inputted video data. cv2.dnn.blobFromImage will pre-processes the input data by performing mean subtraction, scaling, and channel swapping optimally and create a 4-dimensional blob from the input data. net.setInput will define the blob as the input for testing the model performance, while net.getUnconnectedOutLayersNames will perform the prediction and store the output inside the ‘layerOutputs’ list. Then, we extract the bounding boxes, confidence levels, and object classes.

Text

Description automatically generated

Figure 16 Label detected objects

To avoid the exceeding number of boxes displayed on the output, we utilized cv2.dnn.NMSBoxes to select the boxes with high confidences. Finally, we draw a bounding box for the detected and identified object in the output and assign the bounding box with the corresponding glove type and its confidence level.

A picture containing text, bird of prey

Description automatically generated  Graphical user interface

Description automatically generated

Figure 17 AI Glove Type detection results

The diagram above shows the examples of output for the machine learning model testing. Overall, the trained machine learning model is able to achieve a high accuracy with high confidence level. The ski glove and latex glove have a high confidence level as their data inputted for training the model is enough, while the fur glove has a low confidence level as its data inputted for training the model is different from the model testing. By having more training data, the model will perform better.

## 

## 1.5 Model problems and solutions

During the machine learning model development, we met several problems, such as small dataset, time-consuming model training process, and low model performance. Initially, we tried to input a small amount of data to train the model. However, when testing the model performance, the output result shows a low confidence level and sometimes leads to incorrect data label (shown as Figure 54). Hence, we collect more data by taking more photos of different types of gloves. Besides, we perform data augment on the collected image dataset by implementing image cropping, zooming, shifting, and rotating in order to increase the amount of data. As a result, the model performance is better, and the confidence level is improved.

A picture containing text, handwear, clothing

Description automatically generated

Figure 18 Incorrect Glove Identification

Apart from that, the model training speed is slow, and the early produced weight has a low performance. Initially, we trained the machine learning model in our own local device with the CPU instead of GPU. It requires a long training time with 9-11 hours, and the trained model weight is not able to produce an accurate detection result. As shown in figure 55, there is no glove inside the video data. However, the model detects and identifies some incorrect areas with ‘Glove’ label.

Hence, after doing some research, we decided to use Google Colab with its provided free GPU to train the machine learning model. As a result, the model training process is faster than using local device with CPU. After three hours model training, the trained model weight can detect the glove types with a high confidence level.

A picture containing text, indoor

Description automatically generated

Figure 19 Output result of an inaccurate model

# Conclusion

Throughout the conducting of this project, we gained significant amounts of knowledge and insight into the advances and recent developments of ML and AI algorithms and their various implementations. Additionally, literary reviews conducted have contributed to more easily grasping of the methods and techniques used for object detection, and have implemented our specific technique, which uses YOLO v3 for glove type identification, easier and smoother. We have also learned about the methodology and processes of the different models and algorithms. Using the multitude of techniques and inspirations acquired from within the course and review of existing algorithms, we developed a Python application using cv2 and its various functions to facilitate object detection, measurement, color, and defect detection. The use of YOLO v3 with steps from data gathering and preparation, model implementation, and training to the tools and techniques used have provided us with valuable experience in the realm of integration regarding image recognition and AI. By completing this project, we have strengthened our skills in the research and development of object detection algorithms. The knowledge and skills we gained from this module, Image Processing, Computer Vision, and Pattern Recognition, will provide us with a significant advantage in our career, FYP, and future projects.