National College of Ireland

Enhancing Safety in Construction: A Computer Vision Approach for Personal Protective Equipment Detection

MSc Research Project
Data Analytics

Dhrumil Nanji Sidapara
Student ID: X21241210

School of Computing
National College of Ireland

Supervisor: Sasirekha Palaniswamy,

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Dhrumil Nanji Sidapara
Student ID:	X21241210
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Sasirekha Palaniswamy,
Submission Due Date:	14/12/2023
Project Title:	Enhancing Safety in Construction: A Computer Vision Approach for Personal Protective Equipment Detection
Word Count:	900
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Dhrumil Nanji Sidapara
Date:	9th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Safety in Construction: A Computer Vision Approach for Personal Protective Equipment Detection

Dhrumil Nanji Sidapara
X21241210

1 Introduction

The configuration manual offers concise details regarding the hardware and software prerequisites for this research. The program will additionally offer a detailed roadmap for completing the analysis project successfully. The manual is divided into various sections for informational clarity and guidance.

2 System Requirement

2.1 Hardware Requirements

1- Model Name: Acer Aspire E15

2- Operating System: Windows 10 64-bi

3- Processor: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz

4- Memory: 8:00 GB Ram Storage

2.2 Software Requirements

Python was the computer programming language used. The basis for this project was Google Colab PRO, a cloud resource from Google. The software was executed in the background during execution using a Tesla T4 GPU, NVIDIA-SMI 525.105.17, Driver Version: 525.105.17, and CUDA Version: 12.0. An overview of all the software needs used in this study is given in Figure 1.

3 Importing Libraries

The 'pip' statement must be used for installing in cases when particular libraries are not already installed. To set up Roboflow, for example, in a Google Colab, use "%pip install Roboflow." The commands that can be used for importing libraries are shown in Figure 2. All of these libraries combined offer the features required to handle a Deep Learning Image Dataset effectively.

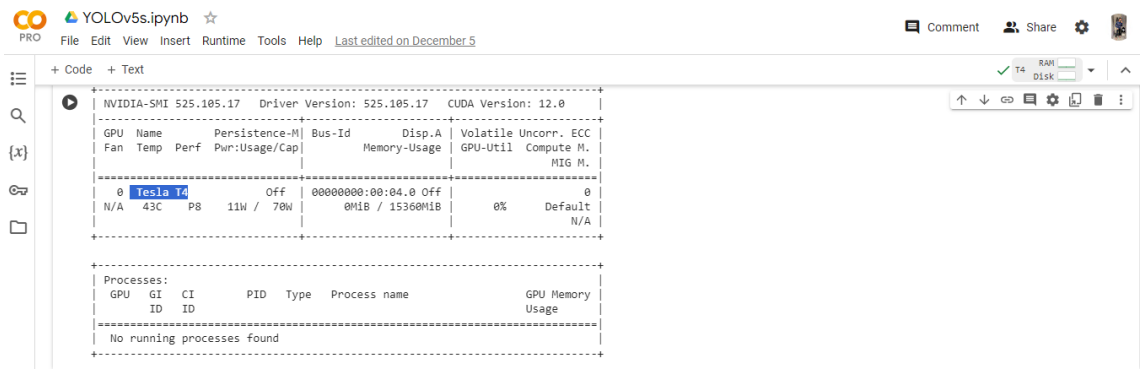

Figure 1: Google Colab

```
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
%pip install -qr requirements.txt  # install
%pip install -q roboflow


import torch
import os
from IPython.display import Image, clear_output

from roboflow import Roboflow
from ultralytics import YOLO
import glob
```

Figure 2: Importing Libraries

4 Dataset Exploration

The set of pictures was downloaded from the Roboflow website and contains 11,978 photos divided into 6,473 training images, 3,570 validation images, and 1,935 test images. Figure 3 shows the data file loading procedure and the API key. The collection of data was fetched into the Google Colab notebook using the Roboflow API.

Figure 3: Importing Dataset

5 Evaluation

The YOLOv5 repository was extracted along with necessary prerequisites and library imports before model analysis was started. This included installing the required dependencies, the Ultralytics YOLOv5 model, the required Roboflow prerequisites, and the indispensable packages. To display pictures in the final result, the 'IPython.display' package was also loaded. The extensive configuration was designed to provide a smooth and fully-equipped setting for the evaluation of the model and examination that followed are shown in figure 4

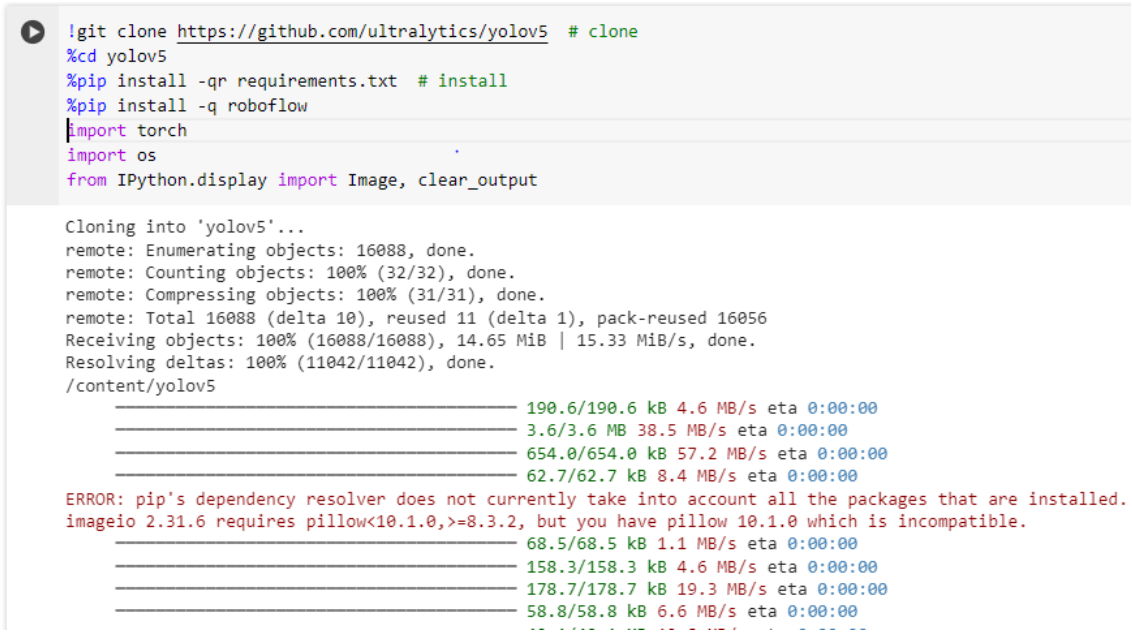

Figure 4: YOLOv5 repository

5.1 Ultralytics YOLOv5 Hyperparameter

Within Ultralytics YOLO, important parameters include things like learning rate and structural information like neuron count and activation function kinds. As an example, 'Learning Rate lr0' establishes the step size in the direction of the loss function minimum. 'Number of Epochs epochs' denotes a complete forward and backward processing pass over all training examples, set here for 5 epochs. 'Batch Size batch' refers to the number of pictures processed concurrently in a forward pass. The Tuner class was used in the'model.tune()' function to parametric tune YOLOv5n on PPE-8 over five epochs. This included implementing an AdamW optimizer and limiting confirmation, checkpointing, and visualization to the last epoch in order to facilitate rapid tweaking. Using momentum and stochastic gradient descent (SGD), the optimizer and weight decay coefficients set at 0.9209 and 0.0005, respectively. The YAML file encapsulates the optimally performing hyperparameters discovered during the tuning process, as illustrated in Figure 6.

```
[ ]  from ultralytics import YOLO

     # Initialize the YOLO model
     model = YOLO('yolov5s.pt')

     # Tune hyperparameters on PPE-8 for 5 epochs
     model.tune(data='/content/datasets/PPEs-8/data.yaml', epochs=5, iterations=5, optimizer='AdamW', plots=False, save=False, val=False)
```

Figure 5: Hyperparameter

```
lr0: 0.01001
lrf: 0.00919
momentum: 0.9209
weight_decay: 0.0005
warmup_epochs: 2.72434
warmup_momentum: 0.79397
box: 7.37398
cls: 0.50774
dfl: 1.51006
hsv_h: 0.01526
hsv_s: 0.69304
hsv_v: 0.38952
degrees: 0.0
translate: 0.09797
scale: 0.48178
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.50675
mosaic: 0.9757
mixup: 0.0
copy_paste: 0.0
```

Figure 6: Results of Hyperparameter

5.2 Training YOLOv5 Detection

Multiple variables are given in this context: 'img' indicates the dimensions of the source picture, 'batch' the batch size, 'epochs' the number of simulated epochs, and 'data' the

path to the YAML file. 'Cache' speeds up training by caching pictures, whereas 'nosave' only save the final checkpoint. The PPE-8 the file from Roboflow was used to simulate YOLOv5s in order to evaluate the model; this took about 5.282 hours for 50 epochs. The analysis criteria, which utilizes an Intersection over Union (IoU) criterion with a score threshold of 0.5, is calculating the percentage of overlap between the predicted and real bounding boxes. Figure 7 illustrates this process.

```
!python train.py --img 416 --batch 16 --epochs 50 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache

2023-12-04 19:16:55.573168: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory f
2023-12-04 19:16:55.573219: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory fo
2023-12-04 19:16:55.573256: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factory
train: weights=yolov5s.pt, cfg=, data=/content/datasets/PPEs-8/data.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=50, batch_size=16, imgsz=416, rect=False,
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v7.0-247-g3f02fde Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 280MB/s]
```

Figure 7: Train yolov5s on custom data for 50 epochs

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      49/49      1.83G    0.02991    0.01172  0.0006721         36        416: 100% 1214/1214 [06:03<00:00,  3.34it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 112/112 [00:27<00:00,  4.11it/s]
                   all       3570       7710      0.677       0.43      0.304      0.155

50 epochs completed in 5.282 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7042489 parameters, 0 gradients, 15.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 112/112 [00:31<00:00,  3.57it/s]
                   all       3570       7710      0.665      0.458      0.311      0.158
                 glove       3570        984      0.712      0.937       0.91      0.475
               goggles       3570       1192       0.81      0.601      0.643      0.384
                helmet       3570        283          1          0          0          0
                  mask       3570        253          1          0          0          0
               no-suit       3570         15      0.016        0.8     0.0306     0.0261
              no_glove       3570       1548      0.553      0.731        0.6      0.277
            no_goggles       3570       1384      0.702      0.699      0.702      0.338
             no_helmet       3570        229          1          0          0          0
               no_mask       3570        640          1          0          0          0
              no_shoes       3570        525      0.157      0.169      0.104     0.0168
                 shoes       3570        639      0.992      0.556      0.599      0.231
                  suit       3570         18     0.0404          1      0.149      0.144
Results saved to runs/train/exp
```

Figure 8: Model Summary after running 50 epochs

5.3 Dectecting all the valid images

Using the best.pt weights file from the YOLOv5 model, 3570 valid photos (with a picture size of 416) were analyzed, and all 12 classes from 1 to 12 were correctly detected and numbered. Figure 9 displays the findings, together with the detection status, time frame in milliseconds, and class numbering.

5.4 Printing the final results

A collection of legitimate photographs found in the PPE dataset are printed using the code in image 10. It looks for folders that match a given pattern by using the glob function. Furthermore, the pictures are displayed by the code using "IPython.display import Image". The final result of valid labels is shown in Figure 11, while the end result of valid predicted pictures is shown in Figure 12.

```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source {dataset.location}/valid/images
image 3077/3570 /content/datasets/PPEs-8/valid/images/p65213195_jpg.rf.af45f2527310832369dddd3fb509dcbc.jpg: 416x416 (no detections), 6.9ms
image 3078/3570 /content/datasets/PPEs-8/valid/images/p65213198_jpg.rf.c4443d19353b694eb7723d5aecf48d5d.jpg: 416x416 (no detections), 5.6ms
image 3079/3570 /content/datasets/PPEs-8/valid/images/p65213199_jpg.rf.6c212f055bfdafb515bbb7d1a864a0fd.jpg: 416x416 (no detections), 7.1ms
image 3080/3570 /content/datasets/PPEs-8/valid/images/p656213103_jpg.rf.0cdf2829594e7ba5973dc3842429c508.jpg: 416x416 2 no-suits, 1 no_glove, 7.6ms
image 3081/3570 /content/datasets/PPEs-8/valid/images/p656213109_jpg.rf.cc19c97002cd6bad41944b3f073eb4d6.jpg: 416x416 1 no-suit, 1 no_glove, 5.8ms
image 3082/3570 /content/datasets/PPEs-8/valid/images/p656213111_jpg.rf.f0b19b905fb778f105f21cc2d9702421.jpg: 416x416 2 no-suits, 1 no_glove, 5.5ms
image 3083/3570 /content/datasets/PPEs-8/valid/images/p656213113_jpg.rf.888b64a22213f8f9327356caa99944a6.jpg: 416x416 1 no-suit, 1 no_glove, 5.4ms
image 3084/3570 /content/datasets/PPEs-8/valid/images/p656213117_jpg.rf.bf50d99d49458f6cb95ad52eb90a158d.jpg: 416x416 1 no-suit, 5.4ms
image 3085/3570 /content/datasets/PPEs-8/valid/images/p656213121_jpg.rf.91a2d2549a951609c4bfc450ffe34315.jpg: 416x416 1 no-suit, 1 no_glove, 5.8ms
image 3086/3570 /content/datasets/PPEs-8/valid/images/p656213128_jpg.rf.bd4923bc5764374bd087cc86d1bb1bcb.jpg: 416x416 1 no_glove, 7.4ms
image 3087/3570 /content/datasets/PPEs-8/valid/images/p656213131_jpg.rf.9725d7fa180766750a84ecfb89c4ab28.jpg: 416x416 1 no_glove, 5.7ms
image 3088/3570 /content/datasets/PPEs-8/valid/images/p656213155_jpg.rf.007851220549c70d44fa710ad74fea66.jpg: 416x416 1 no_glove, 9.3ms
image 3089/3570 /content/datasets/PPEs-8/valid/images/p656213163_jpg.rf.4c4d86760c18dac9c0fad7ba9ab59e5a.jpg: 416x416 1 no-suit, 9.3ms
image 3090/3570 /content/datasets/PPEs-8/valid/images/p656213165_jpg.rf.669f8ac0fd14e8c27362b13733eb6575.jpg: 416x416 2 no-suits, 1 no_glove, 9.1ms
image 3091/3570 /content/datasets/PPEs-8/valid/images/p656213172_jpg.rf.1676e58d797559af3d3685741d03946f.jpg: 416x416 1 no_glove, 9.9ms
image 3092/3570 /content/datasets/PPEs-8/valid/images/p656213173_jpg.rf.072bebbc6949791b0db8f3be93191ae3.jpg: 416x416 1 no_glove, 7.5ms
image 3093/3570 /content/datasets/PPEs-8/valid/images/p65621336_jpg.rf.a10f5645c445c04df92716df78ef5514.jpg: 416x416 1 no_glove, 7.3ms
image 3094/3570 /content/datasets/PPEs-8/valid/images/p65621340_jpg.rf.094bf071ea7c4386d65c0c7be3fcb8fe.jpg: 416x416 1 no-suit, 1 no_glove, 8.9ms
image 3095/3570 /content/datasets/PPEs-8/valid/images/p65621348_jpg.rf.0e9eadcc8c26b733703fe14988228e95.jpg: 416x416 1 no-suit, 2 no_gloves, 9.1ms
image 3096/3570 /content/datasets/PPEs-8/valid/images/p65621356_jpg.rf.f38475f6144bc44376202c025478c256.jpg: 416x416 2 no_gloves, 7.1ms
image 3097/3570 /content/datasets/PPEs-8/valid/images/p6562135_jpg.rf.d81451c14edc10005eed503e1b74bc3a.jpg: 416x416 1 no-suit, 1 no_glove, 7.1ms
```

Figure 9: Valid set

```
import glob
from IPython.display import Image, display

i = 0
for imageName in glob.glob('/content/yolov5/runs/train/exp/*.jpg'):
    i += 1

    if i < 8:
        display(Image(filename=imageName))
        print("\n")
```

Figure 10: Printing final results

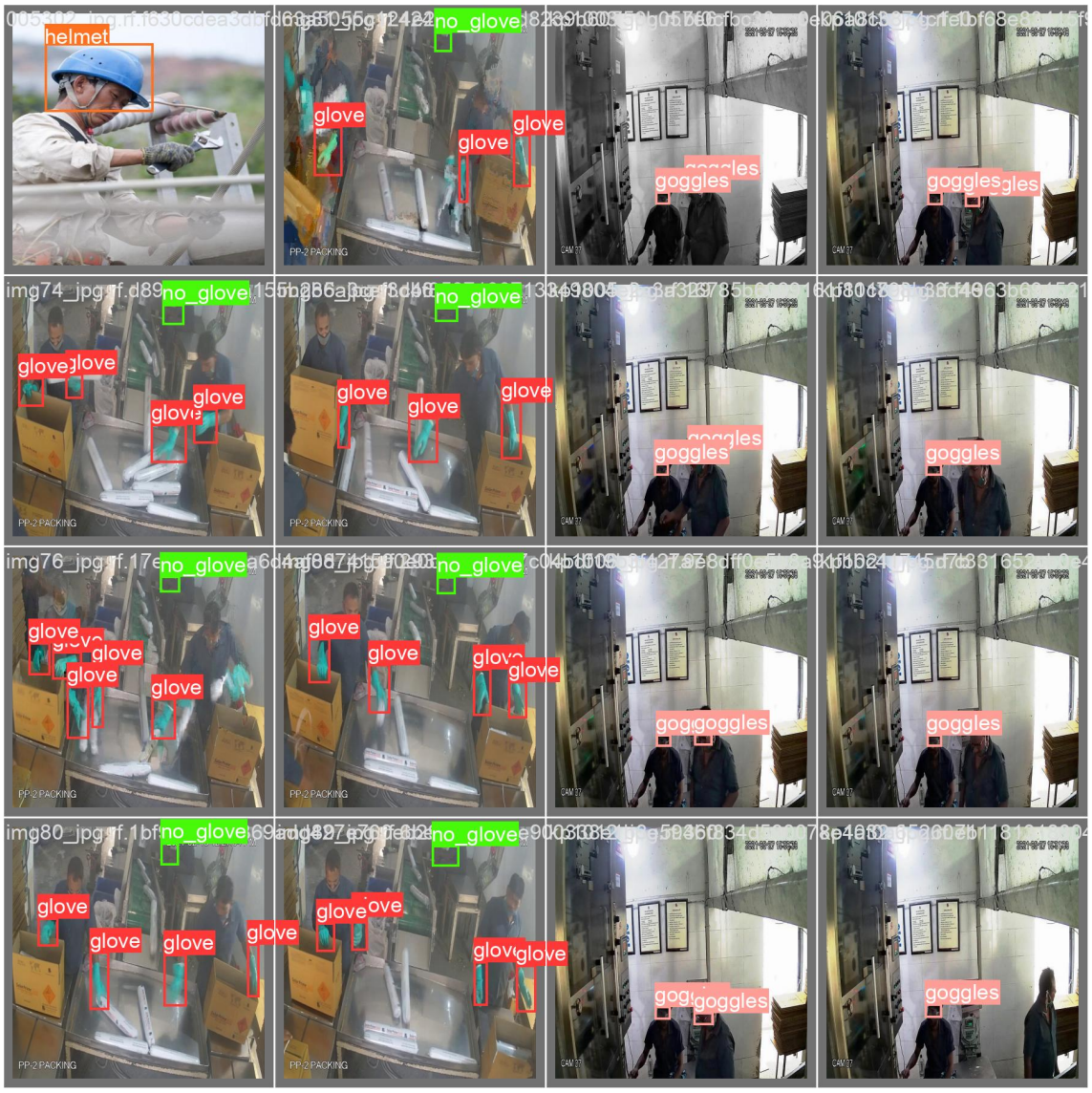

Figure 11: valid labels Results

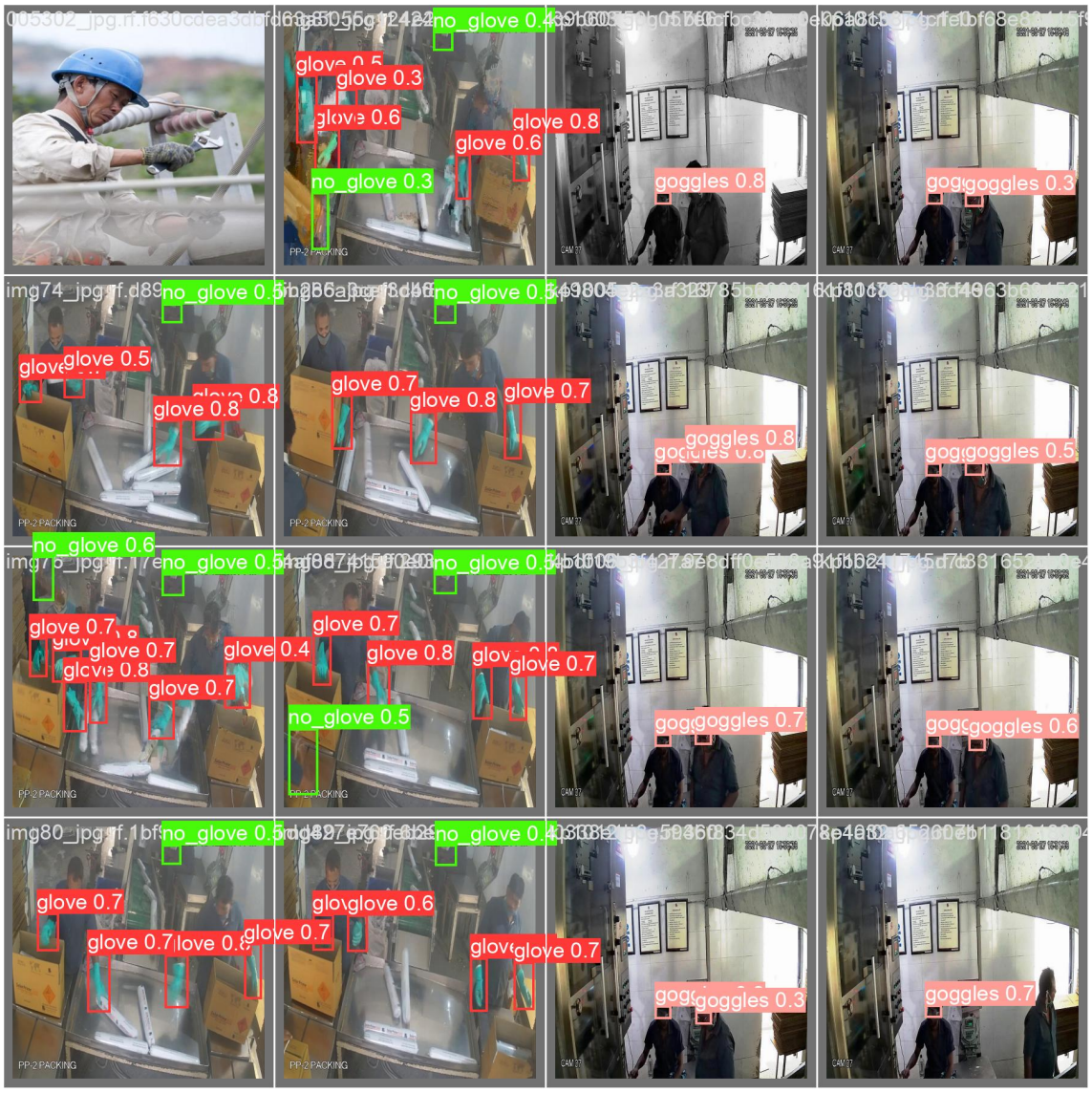

Figure 12: valid predicted results