COMP4471 Final Report:
Real-time PPE Detection in Construction Sites

Paco Yip
HKUST
ptpyip@connect.ust.hk

Simon Cheng
HKUST
hkchengag@connect.ust.hk

Tsui Hok Wai
HKUST
hwtsui@connect.ust.hk

Abstract

In this paper, we will discuss the performance of 3 different deep neural net- work Faster-R-CNN, SSD, and YOLO architectures to perform real-time PPE detection. Under our experiments and findings, YOLLOv5 shows a lead among other deep learning object detection algorithms.

1. Introduction

In the construction industry, safety has the utmost important priority. Unfortunately, in Hong Kong, the number of industrial accidents stay high, which there are around 12.4 per 1000 workers get hurt in 2020 [1]. Workers should wear proper equipment to protect them from getting injured. According to Occupational Safety and Health Council, Personal protective equipment (PPE) refers to "all equipment which is intended to be worn or otherwise used by a person at work and which protects the person against one or more hazards to his/her safety or health" [2]

Safety Supervisors are hence introduced to ensure safety in construction sites, like checking whether workers have been put on PPE properly. Yet, supervisors' manual checking might be time-consuming and prone to negligence. Therefore, we are interested in finding out new autonomous methods via computer vision to make PPE detection more efficient to meet this end.

To do so, we are going to investigate and conduct experiments to find a Deep-Learning based object detection algorithm, which is capable of real-time PPE detection.

[1] Written reply by the Secretary for Labour and Welfare in the Legislative Council, 2022-01-19, https://www.info.gov.hk/gia/general/202201/19/P2022011900273.htm

[2] Personal Protective Equipment, Occupational Safety and Health Council, https://www.oshc.org.hk/eng/main/hot/ppe/

2. Problem Statement

Our project will examine 3 different deep neural network architectures to perform real-time PPE detection. The models should detect and recognize whether a person is wearing PPE properly and output predicted bounding boxes. Our goal is that the detector can check if a worker have worn **safety helmet, reflective vest, and shoes** and tell the result (PASS or FAIL) instantly.

Since the PPE detection will be done in the construction site in real-time with edge devices, the final model has to be fast in inference time and be light weight. Besides, the camera resolution and computational resources available should not significantly affect the model's performance. In addition, the model should also gives decent results under adverse weather situation or any other environmental factor.

3. Related Work

Object detection and classification has been a heated yet challenging area in computer vision for a long time. Earlier works has proven that this problem can be solved. With the advancement of deep neural networks (DNN), data-centric approaches using transfer learning on pre-trained neural networks, have shown their robustness in this area.

In the following, we will introduce some related works to be reviewed in our final report.

a) **Image Descriptors and Classifiers as Detector**
Before the era of deep learning, one of the common ways to perform object detection is to apply image descriptors, which learn to extract the feature of an object, and classifiers, which learn to perform detection according to the feature.

Histogram of Oriented Gradients (HOG) is one of them. As the name implies, HOG use histograms of oriented gradients as the feature to be extracted.

We found a paper that used HOG and MLP to perform a similar task, helmet detection [7]. Despite of the results of their work has achieved a high accuracy of 91.37%, such method is slow and computational expensive.

Yet, studies have shown that CNN-based detectors outperform the image-descriptor-based detectors [2]. Feature extraction algorithms have lower robustness for complex scene images and weaker generalization ability when compare to deep leaning algorithms.

b) **Deep Learning Based Detector**
Here are some paper and the findings on using deep learning to perform PPE detection.

- SSD Detector: lightweight detection models are built with 83.89% mAP [8].
- YOLO-based Detector: 86.55% mAP with 52FPS on GPU [9].
- modified YOLO-based Detector: the YOLOv5 algorithm with the squeeze-and-excitation block brings 94.5% accuracy [10].

We also found projects on GitHub that inspire a lot of our work [3] [4].

We found that most of the existing works related to PPE detection focuses only on investigating one of the 3 models that we are interested in, and seldom experiment all of them on the same dataset. This motivates us to do the comparison among Faster-RCNN, SSD and YOLO in performing PPE detection.

4. Methods

To perform object detection in real-time along with requirements discussed above, Deep Learning based algorithms are used to built our detector.

Compared to traditional Computer Vision methods, these algorithms are Faster-R-CNN, SSD, and YOLO.

Previous works have already shown promising results using similar methods to perform PPE detection. Moreover, we want to explore further on 2 aspects, including model comparisons and multi-class detection. Hence, we trained the following 3 models to perform experiments to find out the best approach.

4.1. Faster R-CNN

Faster R-CNN [**?**] is a two-stage object detection algorithm. A two-stage object detection consists of the following things:

1. Generation of region proposals. Faster R-CNN uses a Region Proposal Network to generate Regions of Interest. It returns a set of object proposals.

2. Object classification for each region proposal. Use a fully connected layer(softmax and linear regression) to perform bounding-box regression. It classifies the detected objects in bounding boxes if any.

In Faster R-CNN, it first passes the image to a convolution layer and generates feature maps for that image. Then it applies the Region Proposal Network(RPN) on those feature maps and gets object proposals. The object proposals and the feature maps will then pass to the RoI pooling layer to ensure that all outputs are in same size. Finally, it passes the proposals to a fully-connected layer and does the classification.

Since Faster R-CNN is a two-stage object detection, it has high accuracy but the inference time will be longer. As our model will be run on mobile devices, it is hard to detect objects in real time. The original Faster R-CNN uses VGG-16 as the feature extractor, we have chosen another version that replaces it with MobileNet, a lightweight model for object detection.

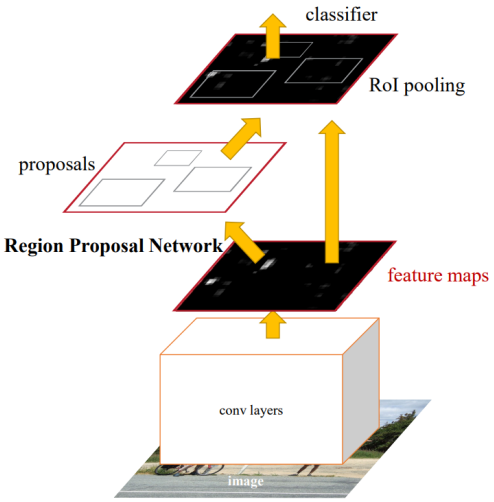


Figure 1. Faster R-CNN architecture

4.2. SSD

SSD, Single Shot Detector [4] utilizes VGG-16 architecture as the backbone network, removing the fully connected layers and adding 6 more convolution layers.

Different from Faster R-CNN, it is a one-stage object detection algorithm. It only takes one single shot to detect multiple objects within the image, thus it is faster compared to the two-stage approach.

[3] Nested PPE Detection, https://github.com/mohammadakz/Nested_PPE_detection_FasterR-CNN

[4] Construction Safety Using Deep Learning, https://github.com/D-Thatcher/SafetyNet

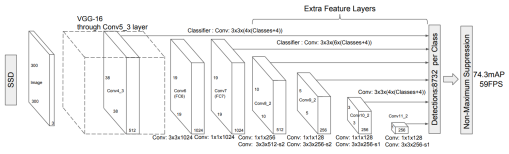


Figure 2. SSD architecture

4.3. YOLO

YOLO, You Only Look Once is another proposal-free detection algorithm. While SSD requires multiple CNNs for detection, it only has one CNN layer to "look' at the image. In the original version [6], it will first divide the input image into grids. Using those grids, YOLO will try to draw bounding boxes for desired objects for each gird, for which the center of the object is located inside a box. Then, it will also predict the probability of which class an object belongs to. Eventually, the algorithm will combine everything to locate and classify objects that existed in the image.

Among all versions of YOLO algorithm, YOLOv5 is the most commonly adopted one. Compared to former versions, YOLOv5 is not just an algorithm on paper but also a python library with pre-trained models and easy-to-use APIs. In the following, we will introduce some important concepts of YOLOv5.

Despite how complicated the structure in Figure 3 [5] may seem, the general idea is still the same as above. Different middle structures are used for various functions plus as part of optimization:

1. **Input**
   In the implementation of YOLOv5, it automatically utilized lots of data augmentation on the Input. This simplifies the training process as developers can apply different kinds and different degrees of augmentation by simply changing the configuration file.

2. **Backbone**
   Backbone basically replaced the HOG descriptor we mentioned above as a feature extractor. It also performs an early classification so that Head can draw bounding boxes. Under the hood, it uses Focus block to extract features, and CSP [6] block to reduce memories and parameters needed to use.

3. **Neck**
   Neck further extracts features by combining features from different dimensions, preparing for detection.

4. **Detection Head**
   Finally, the detection head is some CNN layers to locate the object are coding to the feature extracted.

5. **Loss**
   The loss function is designed to help the model learn how to correctly draw bounding boxes. Hence the loss involves class loss and IOU loss.

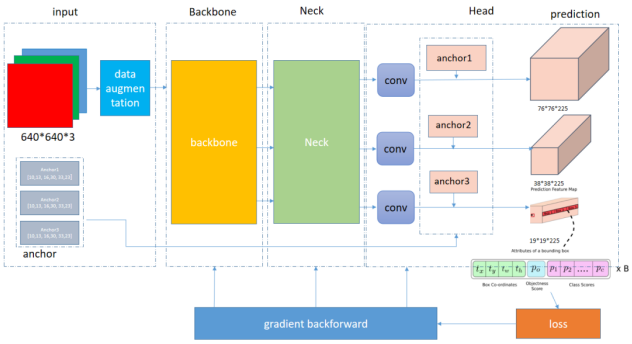


Figure 3. Simplified visualization of YOLOv5

5. Datasets

In addition to the above models, it is of vital importance to use a dataset with a good image and label quality for our detector.

We use the Safety Helmet Wearing Dataset [1] from Github. Annotated in Pascal VOC format, it contains 7.6k images labeled in 2 classes, no helmet, and with helmet, with 1 : 12 helmet and no-helmet ratio. We found this dataset useful because it contains photos of workers on construction sits and photos of normal people, which expand the knowledge of the model. With bounding boxes focusing on people's heads, we think this would help the models to learn where to "look at" and learn the difference between a head with and without a helmet. A typical image is shown below.

captionRed box indicates people who were safety helmet while blue box indicates who didn't.

[5] source: https://blog.csdn.net/weixin_51237675/article/details/112342494

[6] Cross Stage Partial Network(CSPNet) has also been shown the ability to improve speed and computational costs

For data augmentation, we applied the following to the training images: normalization, ± 10 degree of rotation, ± 0.9 of scale, ± 10 degree of vertical/horizontal shear, and horizontal flip with 0.5 probability. Notice above will be applied with a value randomly chosen in the given range.

6. Metrics

We used an open-source detection evaluation metric implementation[7] to compare trained models. Both the locality and the classification label of an object are essential to object detectors. Hence, a detection output is often represented by 3 attributes: the object class, the corresponding bounding box, and the confidence score. The common detection evaluation metrics are then defined differently.

In the following, we will first explain how to define metrics in the Confusion Matrix for object detection tasks. Then, we will describe how to calculate and get insight from the 2 metrics we chose, Average Precision and Recall. [8]

a) **Confusion Matrix for Object Detection**
Instead of using classification scores, the Confusion Matrix for object detection measures how close the detected bounding boxes are to the groundtruth, using IoU, intersection over union. The IoU is a ratio between the overlapping or intersection area and the union area of the detected region and the groundtruth., which is illustrated down below. A perfect detection region means the IOU = 1. Hence, the high the IOU value is, the better the detected bounding box is.

$$\text{IOU} = \frac{\text{area of overlap}}{\text{area of union}} =$$

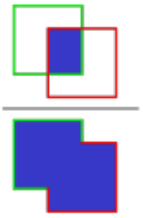


Figure 4. Illustration of IoU. [5]

In practice, a prediction of the detection is regarded as correct if **i)** the class label matches the target label and **ii)** the IoU between the detected bounding box and the groundtruth is greater than a threshold. With this in mind, the following metrics in the Confusion Matrix are defined:

- True Positive (**TP**) : A correct detection.
- False Positive (**FP**) : A correct classification with incorrect region; or an incorrect classification with correct region.
- False Negative (**FN**) : An object is not detected
- True Negative (**TN**) : An non-targeted object or background detected. (Not very interested for object detection.)

b) **Precision and Recall**
Precision shows the ability of a model to correctly detect only relevant objects in a given class. It is defined as:

$$Precision = \frac{CorrectDetections}{TotalDetections} = \frac{TP}{TP + FP}$$

Recall, aka sensitivity, shows the ability of a model to detect all relevant objects in a given class. It is defined as:

$$Recall = \frac{CorrectDetections}{TotalGoundTruth} = \frac{TP}{TP + FN}$$

For multiclass detection, the classification confidence level is also commonly taken into account in the calculations. Only detections whose confidence is larger than a confidence threshold τ will be considered positive. Notice that now TP, FP, and FN become functions of τ, and hence precision and recall as well.

A Precision-Recall Curve (PR Curve) is also commonly plotted to find the best confidence threshold. It a plot of precision against recall for different confidence threshold values.

c) **Average Precision**
Average Precision (AP) is the most commonly used metric for performance comparison of object detection models. It tries to eliminate the dependency of selecting a confidence threshold, which can be subjective. Mathematically, the average precision is the area under the PR Curve. Approaches to find such area can be found in this paper [5]. Besides, mean Average Precision (mAP), which is the average AP over all classes, is also used. It is defined as:

$$mAP = \frac{1}{C} \sum_{k=1}^{C} AP_k$$

In the calculation of AP, the IOU threshold need to be set. We will use 2 AP, including setting AP@0.5 and AP@[0.5:0.95], which the latter one is computed by taking the average of 10 different IOU thresholds in the range of threshold values.

[7] Source code can be found in https://github.com/rafaelpadilla/review_object_detection_metrics

[8] This part is heavily referenced and inspired by this paper [5].

d) **Average Recall**
Another important metric is Average Recall (AR), which summarizes the distribution of recall across a range of IoU thresholds. It can help to evaluate the effectiveness of detection proposals and the accuracy of a bounding box. Yet, we will not use it in our evaluations since the bounding box's location is not our primary concern.

7. Experiments and Results

To save time and as a fair comparison, we choose to perform transfer learning on the pre-trained models. After fine-tuning the models with the above dataset, we perform validation and inference to investigate which one suits the best for our detector.

While YOLOv5 is somewhat easy to train, we implemented our training pipeline on other model using Pytorch [9]. Using Pytorch can ensure fairness since the yolov5 library is using Pytorch under its API. Here we will first discuss the settings while training Faster-R-CNN and SSD. Then, we will discuss how we train different models.

7.1. Pytorch Training Pipeline

We tried various optimization methods and hyperparameters fine-tuning on top of the pre-rained model we use.

First, we tried different batch sizes, which using the most common values leads to exceeding Colab's GPU memory. Hence, we our training pipelines using only 2 to 3 batch sizes, which 2 values have minimal effects while training.

Second, according to the information taught on lectures, we choose SGD with Momentum as our optimizer. We hope this can avoid gradient being not able to decent while training. Besides, we also include a learning rate scheduler hoping for better results.

7.2. Transfer Learning

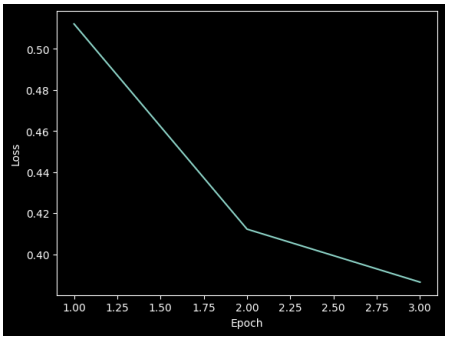
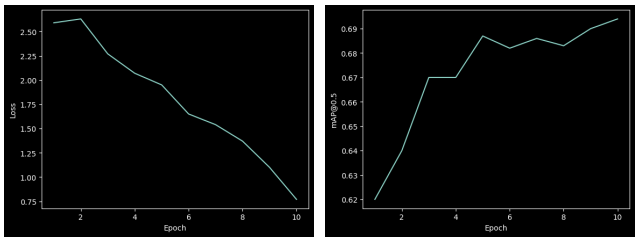


Figure 5. Loss graph of Faster R-CNN

[9] which is based on the TorchVision Object Detection Finetuning Tutorial, https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

1. **Training Faster-R-CNN**
We used the pre-trained version 2 of the Faster R-CNN model with a ResNet-50-FPN backbone from Pytorch, which the model is based on in this paper [3]. During training, we first had a hard time reducing loss. We tried learning rates from .01 to 1e-7, yet nothing seems to work. Another setback is the slow training speed of Faster R-CNN since it consists of 2 large networks to be trained.
Yet, after we wait for a few epochs, it turns out that the performance is already very good, with a validation mAP@0.5 0.89. One possible reason for this is that the current pre-trained network is trained so well or even the dataset contains similar classes.

   Unfortunately, limited by Colab, we can only train 3 epochs at last. Here is the loss graph showing the loss decreases from a small loss.



(a) Loss graph of SSD (b) mAP@0.5 graph of SSD

Figure 6. Comparisons on far away objects

2. **Training SSD**
We used the pre-trained SSD300 with VGG16 Backbone, the implementation of the SSD algorithm [4]algorithm with 3000x300 input resolution. During training, we use 0.001 as the learning rate and 0.9 momentum. As shown on Figure 6, the loss gradually goes down while the mAP goes up while training.

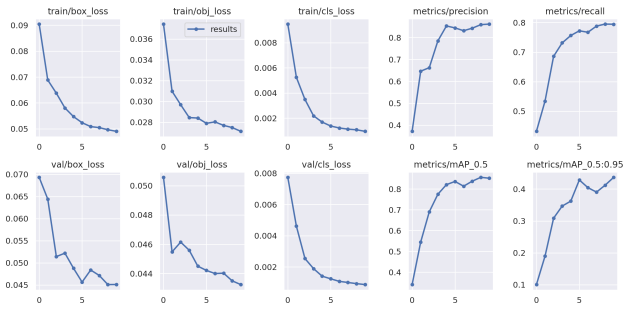


Figure 7. Results of YOLOv5

3. **Training YOLO**
We use the yolov5 Python library from ultralytic [10],

[10] https://github.com/ultralytics/yolov5

with 0.01 as the initial learning rate, 0.9 momentum while other hyperparameters remain as default. During training, we keep track of the loss and mAP. All the losses decrease and mAP increases as expected. Eventually, we train YOLOv5 for 10 epoch.

8. Evaluation

8.1. Quantitative Evaluation

1. **Performance**

   We evaluate our trained models with Mean Average Precision (mAP). Our trained YOLOv5s perform the best, achieving mAP@0.5 of 0.92 and mAP@0.5:0.95 of 0.58. Faster R-CNN ResNet-50, YOLOv5s and YOLOv5n perform nearly the same, with mAP@0.5 around 0.8. SSD300 VGG-16 and Faster R-CNN MobileNet perform slightly worse, with mAP@0.5 of 0.69 and 0.47 respectively.

2. **Speed**

   Using the test dataset, we also calculated the mean inference time on some of the models. We observe that YOLOv5n has the shortest inference time.

3. **Model Size**

   The model size varies with the number of parameters or operations in a neural network. A smaller size model can reduce the storage needed and requires less computational power, thus more suitable for mobile devices. Among the models that we trained, Faster R-CNN with ResNet-50 has the largest size of 167MB, while the YOLOv5n has the smallest size of 3.7 MB only. To reduce the model size, we can also apply some lightweight models such as MobileNet as the model's feature extractor. We have trained a Faster R-CNN with MobileNet, resulting in a model size of 74 MB. Compared with the ResNet one, the model has been reduced by half.

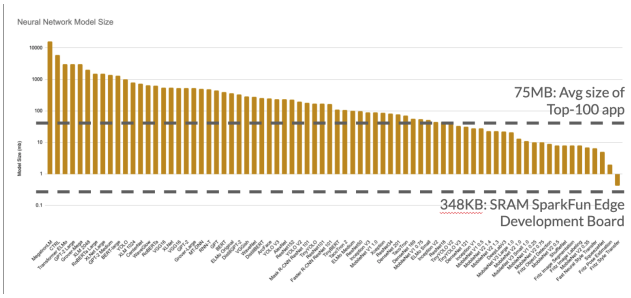


Figure 8. Model size of Different Neural Network

(a) YOLOv5 Test image 1.

(b) Faster R-CNN Test image 1.

Figure 9. Comparisons on far away objects

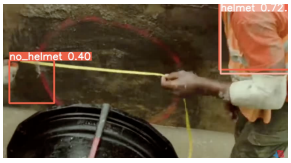


Figure 10. Captures from the output video

8.2. Qualitative Evaluation

1. Small objects

   By comparing test images among these models here is what we found. While both Faster R-CNN and YOLO struggle to detect small and far objects, Faster R-CNN still be able to differentiate the background and the crown who wearing helmets. In contrast, YOLO cannot identify there are people on the back. Yet, we were surprised that the detectors did not fooled by those helmets on the floor. This shows they know the difference between a helmet being put on or not.

2. Inference on Video

   So far, YOLO seems to be a better model for our detector. Yet, for future development and deployment, we also conducted object detection on video to test whether the models can be use as a real-time detector. Using a video found from YouTube[11], we have trained a video using 0.4 IOU threshold and have the following findings: **i)** On the good side, YOLOv5 works find even the face is covered; ii) unfortunately, miss-detection still occured.

9. Conclusion

In conclusion, we have examined the potential and some power of novel Deep-Learning based object detection models. For our use case, construction sites, GPU is not available. Hence we are interested in the CPU Conference speed and performance. By this, we think YOLOv5 get advantages since it has a high mAP, fast speed and small model size.

[11] video can be found here: https://youtu.be/s1n6nBhVzdo

We hope future works can expand our results to more PPE rather than simply on helmet. With a larger dataset and better computational resources better work must be able to given high quality results and give efforts to safeguard lives in construction sites.

References

[1] Github. Safety helmet wearing dataset. 2019. 3

[2] Phutphalla Kong, Matei Mancas, Nimol Thuon, Seng Kheang, and Bernard Gosselin. Do deep-learning saliency models really model saliency? pages 2331–2335, 10 2018. 2

[3] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers, 2021. 5

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016. 2, 5

[5] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021. 4

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. 3

[7] Romuere Silva, Kelson Aires, and Rodrigo Veras. Helmet detection on motorcyclists using image descriptors and classifiers. pages 141–148, 08 2014. 2

[8] Zijian Wang, Yimin Wu, Lichao Yang, Arjun Thirunavukarasu, Colin Evison, and Yifan Zhao. Fast personal protective equipment detection for real construction sites using deep learning approaches. *Sensors*, 21(10), 2021. 2

[9] Jixiu Wu, Nian Cai, Wenjie Chen, Huiheng Wang, and Guotian Wang. Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset. *Automation in Construction*, 106:102894, 2019. 2

[10] Z P Xu, Y Zhang, J Cheng, and G Ge. Safety helmet wearing detection based on yolov5 of attention mechanism. *Journal of Physics: Conference Series*, 2213(1):012038, mar 2022. 2