Pedestrian Detection

Investigative Studio 2
Artificial Intelligence (AI) CS205

Authors: *M. Gedye and I. Getman*

Tutor: *Aisha Ajmal*
Course: BSE Year 2
Date: November 27, 2022

Contents

1 Introduction

Increasing safety levels for pedestrians is an important issue when it comes to reducing road traffic injuries. The World Health Organisation (WHO) elaborates that these incidents cause much suffering and grief as well as economic hardship (and) should not be accepted as inevitable because they are predictable and preventable.

In New Zealand, pedestrian deaths have declined gradually over the last 30 years although the numbers recorded for injuries are still high. In 1990, there were 105 pedestrian deaths whereas in 2020, the death toll was reported at 32. While this number may seem low, 224 pedestrians were seriously injured and 551 pedestrians suffered minor injuries in that same year.

NZ organisation AT(Auckland Transport)'s vision makes a strong point that no death or serious injury is acceptable. After reviewing the evolution of pedestrian detection technologies and research in the last 20 years (our paper), it is fair to say there is real potential to one day achieve AT's vision.

The purpose of this report is to share the learnings we have taken onboard in the following order:

- reintroduce our chosen topic, give an outline of the report
- recap of the literature we reviewed from the research paper
- implement a selection of PD models
- evaluate and compare the results of these models
- discuss limitations we faced and raise ideas for future work

2 Literature Review Recap

This section contains excerpts from our research paper that reviewed the progress in this field from over the last 20 years.

Pedestrian detection is a task in the Computer Vision field that refers to automatic detection of walking people based on different sensors, such as Light Detection and Ranging (LiDar) sensors, Millimeter-Wave Radar (MWR) sensors and the most common sensor these days - cameras.

The progress of object detection has been generalised into two historical periods: **traditional object detection period** (before 2014) and **deep learning based detection period** (after 2014) (Zou et al).

2.1 Traditional Methods (pre 2010)

- Viola and Jones pioneered in 2001 with real-time face detection [1] by using sliding window approach as the most sufficient method in low to medium resolution images

whereby segmentation or point based methods often fail.

- Histograms of Oriented Gradients (HOG) were created in 2005 by N.Dalal and B.Triggs [2]. To detect objects of different sizes, the HOG detector rescales the input image for multiple times while keeping the size of a detection window unchanged.

- Deformable Part-based Model (DPM) was originally proposed by Felzenszwalb et al.in 2008 [3]. DPM heavily relies on descriminative training with partially labeled data combined a margin-sensitive approach for data-mining hard negative examples with latent SVM.

2.2 Deep Learning Methods (post 2010)

As the performance of hand-crafted features became exhausted, object detection has reached a plateau after 2010. And two years later, the world saw the rebirth of Convolutional Neural Networks (CNN or ConvNet). Since then, CNN became the dominant paradigm in object detection area and pedestrian detection in particular.

In the deep learning era, object detection has been typically grouped into two categories: **two-stage detection**, **one-stage detection**. A two-stage detector uses separate networks: one to extract region of interest(RoI) and the second model is used to classify and refine. On the other hand, a one-stage detector only requires a single pass through a neural network to make a prediction of all bounding boxes on a grid.

In the last few years though, there have been two recent approaches that have made waves in the object detection communities; **multi-scale detection** models which are aimed at solving issues such as occlusion and night-time (common challenges in PD) and **visual transformers** which are making breakthroughs in terms of model accuracy and speed.

2.2.1 Two-stage Detectors

R-CNN (Regions with CNN features)
R. Girshick et al. in 2014 proposed the R-CNN for object detection [4]. R-CNN achieved significant results by combining two key insights:

1. Object proposal being extracted, rescaled and fed into CNN model to localise and segment features.

2. Linear SVM classifier predicts the object within each region and recognises the category. On the 200-class ILSVRC2013 detection dataset, R-CNN's mAP (mean Average Precision) was 31.4% and outperformed another CNN-based method, OverFeat by a large margin.

SPPNet (Spatial Pyramid Pooling Networks)
In 2014, He et al. [5] proposed a new pooling strategy that eliminated required fixed-size image for CNN. Thus avoiding extensive repetitions passing through convolutional network, SSPNet was 20 times faster than R-CNN and achieved better or comparable accuracy on Pascal VOC

2007.

Fast R-CNN
In 2015 R. Girshick proposed a new training algorithm [6] that fixes the disadvantages of R-CNN and SPPnet and called it Fast R-CNN. The Fast R-CNN method has several advantages:

- Higher detection quality (mAP) than R-CNN, SPPnet
- Training is a single-stage process
- Training can update all network layers
- No disk storage is required

Faster R-CNN
In 2017 some questions about object proposals using CNN model were answered by Ren et al [7]. Faster R-CNN is the first end-to-end, and the first near-real-time deep learning detector (VOC 2007 mAP=73.2%).

Region Proposal Network Region Proposal Network (RPN) was introduced to generate region proposals directly in the network instead of using external algorithms. RPN uses anchor boxes for object detection. This method is almost cost-free for region proposals.

Feature Pyramid Networks
Introduced by Lin at al. in 2017 [8]. The construction of the pyramid involves a bottom-up pathway and top-down pathway. Without bells and whistles, this detector reached the state-of-the-art single-model result on the COCO detection benchmark. It is simply based on a basic faster CNN detector. This improvement was achieved without increasing testing time over a single-scale baseline. FPN has now become a basic building block of many of the latest detectors [9].

2.2.2 One-stage Detectors

You Only Look Once (YOLO)
In 2016 Redmon, Divvala, Girshick and Farhadi [10] presented YOLO, a regression based method that uses a single neural network to predict bounding boxes and class probabilities from images in one evaluation. This unified approach made significant progress in terms of real-time speed and performance; the base YOLO model is able to process images at 45 frames per second and the Fast YOLO model (a smaller version of the network) can process 155 frames per second.

Single Shot Multibox Detector (SSD)
Another breakthrough model in 2016 made by Liu et al. [11] was also detecting objects in images using a single deep neural network. This network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match

the object shape.

Retina-Net
Despite one-stage detectors surpassing two-stage detectors in terms of speed and simplicity, there was not a substantial difference in accuracy between the two at the time [12]. A group of Facebook AI Researchers in 2017 explored this problem and found that the main cause of this issue was linked to a foreground-background class imbalance during the training process of these one-stage 'dense' detectors. To solve this issue, they used a focal loss approach which reshapes the standard cross entropy loss so that it down-weights the loss assigned to well-classified examples. The group created a simple dense detector called RetinaNet and trained this model with the focal loss and found that its speed matched previous one-stage detectors while surpassing accuracy of all existing state-of-the-art two stage detectors.

2.2.3 Multi-scale Detectors

In recent years, multi-scale detection models have been proven to deal with some of the most common environmental challenges such as occlusion, small-scale object detection and detection at night.

The underlying idea of detection models that have the multi-scale feature is that they are typically constructed of several CNN models, that operate across multiple scales concurrently.

In 2020, Peng, Chen, Fu and Yi [13] proposed a system for intelligent vehicles to detect pedestrians at night, using two traditional methods mentioned earlier, SVM and HOG alongside a number of image preprocessing methods to optimise the night images and detect driving lanes. A camera is mounted on a vehicle which captures image data of the road ahead. The RBG image is then converted into a hue saturation lightness (HSL) image which is heavily influenced by the lighting of the vehicle's headlights. The highest accuracy for detection was tested under good light conditions, reaching 91.8% compared to conditions with no light being the lowest accuracy rate at 81.2%.

Last year, Ref. [14] took on the small-scale object detection challenge and proposed a multi-scale pedestrian detector based on self-attention mechanism and adaptive spatial feature fusion. Using CenterNet as the baseline, they embedded the asymmetric pyramid non-local block (APNB) module to capture global information at certain levels, bringing richer spatial information. An adaptively spatial feature fusion (ASFF) was used to filter the relevant features for the combination of feature maps in different layers. Using the Caltech and CityPersons datasets, the results were most effective on pedestrians at a reasonable distance with a miss rate of 8.9% compared to pedestrians at a medium distance 31.48% and far distance 69.39% [**?**].

This year (2022), Ref. [15] proposed a convolutional block attention module (CBAM), a DeepSort pedestrian tracking method and the Kalman filter to estimate the pedestrian motion state. The detection model consisted of the Darknet-53 backbone network module to extract pedestrian features, following the spatial attention and channel attention modules to perform multi-scale pedestrian prediction. They used the Hungarian algorithm and Kalman filter estimation

for its good tracking accuracy. Motion measure is the standard used in data association for this algorithm which does not perform well in tracking targets that become occluded. To deal with these situations, the DeepSort algorithm was used which added appearance measure information during the data association process. This improvement provided an ability to deal with this specific type of occlusion problem.

2.2.4 Visual Transformers (ViT)

In recent papers [16], [17] we found mentioning of a rise of Transformer Architecture as a new object detection method. Transformer architecture has been the De-Facto standard for natural language processing (NLP) tasks yet computer vision field heavily based on ConvNet detectors and applications using Visual Transformers are few. When CNN uses pixel arrays, ViT splits the images into visual tokens; correctly embeds each of them, and includes positional embedding as an input to the transformer encoder. Moreover, ViT models outperform CNNs by almost four times when it comes to computational efficiency and accuracy. Most significant models of Visual Transformer are Vision Transformer-Faster RCNN (ViT-FRCNN) (Beal et al. [18]), Detection Transformer (DETR) (Carion et al. [19]), Pointformer (Pan et al. [20]), Deformable DETR (Zhu et al. [21]) and Swin Transformer (Liu et al. [22]).

3 Implementation

Changes to the initial proposal
In the original proposal, the plan was to implement a single pedestrian detector using Visual Transformers (or ViT) with the reasons being that it was being new as well as making breakthroughs in performance, especially in terms of training speed. Due to lack of resources, it was not possible for our team to understand and practice well enough in time for the assessment deadline. Instead, the decision was made to implement multiple models that we had previously researched in more depth, thus, enabling us to complete the required tasks.

This section has been divided into two experiments; the first is a pedestrian detector model using the Caltech Pedestrian dataset that differentiates between two classes: ‘person’ and ‘person-like’. The second explores two pre-trained models; one is a pedestrian detector that solely classifies pedestrians to be true or not and the second model is an object detector that contains 80 classes.

3.1 Experiment One

3.1.1 Dataset

For our implementation phase we have chosen a subset of Caltech dataset[ref]. It is significantly smaller (170 MB) than original Caltech dataset of 2GB. This dataset was specifically prepared to address false positive values that often occur when model only trained on images with person objects. Dataset has two classes: “person” and “person-like”. “Person-like” objects have very similar features to those of a person: scarecrows, mannequins, robots, statues. Dataset is already split in three folders: Train, Test and Validation, each of them contain two

```
train_info.head(20)
```

	xmin	ymin	xmax	ymax	name	label
0	158	44	289	167	image (1)	person
1	185	56	287	241	image (10)	person
2	2	86	344	374	image (100)	person
3	220	95	500	290	image (100)	person
4	13	110	213	375	image (101)	person

Figure 1: data sample

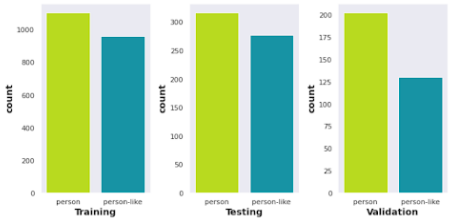


Figure 2: Number of person and person-like objects in each dataset

other folders, one with images, one with annotations. 2066 entries in train data, 595 in test data and 333 in validation data. Annotations are in xml format.

3.1.2 Data loading, reading and preprocessing

After downloading and unzipping our data, we created a first function to assign .xml annotations to images in each data subsets and this function also will buid bounding boxes around objects with exact location in image of each side of the boxes. This function will return Panda DataFrame. Next step is to create three new datasets for each folder containing information of labels, bounding boxes sizes and their locations in each image. Function croppingFromImage will extract all the objects from images and return cropped images and annotations and store them separately for each data split part. As a result of this process we received images of different shapes that would require re-scaling in order to generalize the input size for the future steps. According to Saponara and Elhanashi [**?**] , larger input images might increase training time up to four times, so to get faster training time, we re-scaled our images to the size of (32,32,3).

3.1.3 Basic CNN Model

Our first model is a simple convent consisting of three convolutional layers for feature extractions, than Flatten and Dense layers for classifications. We used ‘adam’ optimizer, Sparse Categorical Crossentropy as a loss function and ‘accuracy’ as metrics.

3.1.4 Tuned CNN Model

Parameters we changed in the basic model:

- In the first Convolutional layer we will change kernel size and add padding ’same’.
- In the third Conv2D layer we are going to double number of filters which is a common practice.
- In the classifier head we will use ’sigmoid’ activation function in the final Dense layer
- Add number of the epochs (40 compare to 7 in the initial model)

3.1.5 CNN Model with pretrained base

This time for our feature extracting step we will use a pretrained base. The most commonly used dataset for pretraining is ImageNet, Keras has a number of applications modules pre-trained on ImageNet. The pretrained module we will use is VGG16. VGG model consists of 13 convolutional layers, 5 max-poolings and 3 dense layers. Number of layers with tunable parameters is 16 (13 convolutional layers plus 3 Dense) and that is why the model was named VGG16. VGG stands for Visual Geometry Group. VGG16 can deal with large-scale images, by default its input size=(244,244,3). However, we will downscale the input size to (140,140,3) to get faster training time. First time, when we tried to train VGG16 model with default parameters it took us 6+ hours. Therefore our scaling parameter will be 140 to reshape all extracted pictures to (140,140,3) size. We concatenated Xtrain and Xtest to obtain bigger dataset for training. Training size is 2661 images and 333 images for testing (original Validation set). Shape of features extracted by the VGG16 model to the training set is now (2661, 4, 4, 512). Next step would be to flatten a 4D array into 1D array to be fed as an input for future classification tasks. Shape of training set after flattening is (2661,8192). Number of epochs is 30, batch size=100.

3.2 Experiment Two

In this second experiment, the focus is evaluating the performance of two pre-trained models, which means the stage of manually preprocessing, transforming and labelling data is not needed in this situation. Pre-trained models are typically easier to use because the architecture is already constructed and they are usually trained on very heavy datasets, saving time.

The first approach is considered a traditional method, using OpenCV (an open-source library used for computer vision tasks) to access a built-in method known as the HOG (Histogram of Oriented Gradients) + L-SVM (Linear Support Vector Machine) model. This is a well known model used for pedestrian detection and has the ability to classify pedestrians from images and video streams.

The second approach is considered a deep learning method called YOLOv5. This is a very popular object detection algorithm that has been implemented through the pytorch framework.

3.2.1 Approach 1: HOG + Linear SVM

The architecture or pipeline for the first approach relies on two main parts; the first is the histogram of oriented gradients (HOG) which is an algorithm which creates descriptions for the features it detects. The HOG algorithm scans the surrounding pixels of every single pixel and grades the current pixel's tone of darkness in comparison to the surrounding pixels. It then generates arrows called gradients to show the direction of where the image pixels are becoming gradually darker.

The second part is the Linear Support Vector Machine (L-SVM) which is an algorithm used to solve classification and regression problems. Once the HOG processes image pixels that have been organised into gradient features, L-SVM can then plot data points from these features and then using a hyperplane, it is able to split this data into classes, in this case there are two

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
max_pooling2d (MaxPooling2D  (None, 15, 15, 32)        0
)
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)          0
2D)
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
flatten (Flatten)            (None, 1024)              0
dense (Dense)                (None, 64)                65600
dense_1 (Dense)              (None, 1)                 65
flatten_1 (Flatten)          (None, 1)                 0
dense_2 (Dense)              (None, 64)                128
dense_3 (Dense)              (None, 1)                 65
flatten_2 (Flatten)          (None, 1)                 0
dense_4 (Dense)              (None, 64)                128
dense_5 (Dense)              (None, 10)                650
flatten_3 (Flatten)          (None, 10)                0
dense_6 (Dense)              (None, 64)                704
dense_7 (Dense)              (None, 2)                 130
=================================================================
Total params: 123,790
Trainable params: 123,790
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: Basic CNN model architecture

```
VGG_Model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 140, 140, 3)]     0
block1_conv1 (Conv2D)        (None, 140, 140, 64)      1792
block1_conv2 (Conv2D)        (None, 140, 140, 64)      36928
block1_pool (MaxPooling2D)   (None, 70, 70, 64)        0
block2_conv1 (Conv2D)        (None, 70, 70, 128)       73856
block2_conv2 (Conv2D)        (None, 70, 70, 128)       147584
block2_pool (MaxPooling2D)   (None, 35, 35, 128)       0
block3_conv1 (Conv2D)        (None, 35, 35, 256)       295168
block3_conv2 (Conv2D)        (None, 35, 35, 256)       590080
block3_conv3 (Conv2D)        (None, 35, 35, 256)       590080
block3_pool (MaxPooling2D)   (None, 17, 17, 256)       0
block4_conv1 (Conv2D)        (None, 17, 17, 512)       1180160
block4_conv2 (Conv2D)        (None, 17, 17, 512)       2359808
block4_conv3 (Conv2D)        (None, 17, 17, 512)       2359808
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 4: VGG16 model architecture

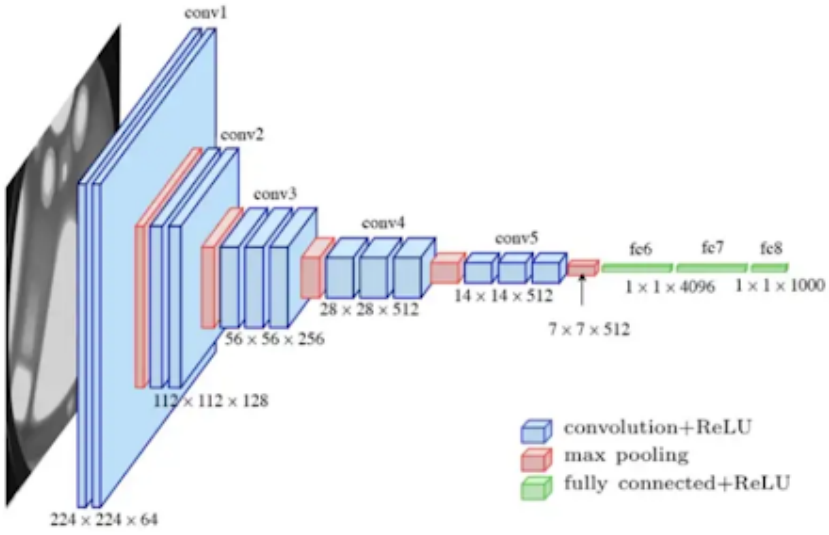
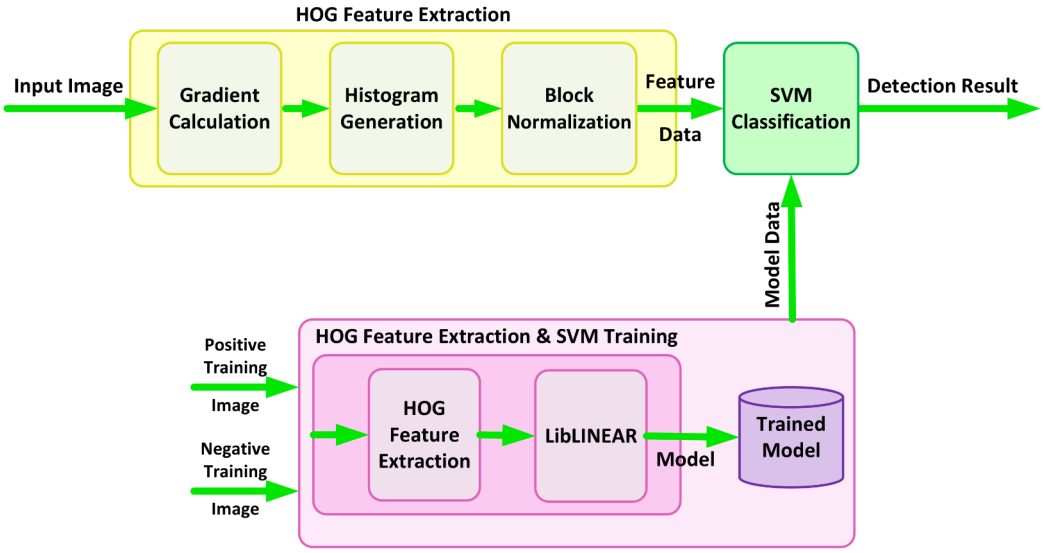


Figure 5: VGG16 model

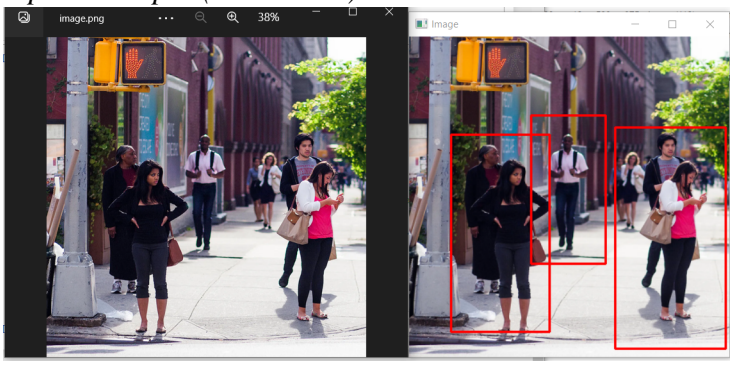
classes; 1(pedestrian, positive) and 0(not pedestrian, negative).



Once L-SVM has classified the data, it is now possible to create drawing functions to display a bounding box around the pixels classed as positive (containing a pedestrian).

This model performs very quickly and can easily identify pedestrians from a medium position/distance although it was seen that detection was almost non-existent for pedestrians that are positioned far away. Another downfall was that pedestrians identified walking close together and crowds were contained in a single bounding box.

Input vs output (screenshot)

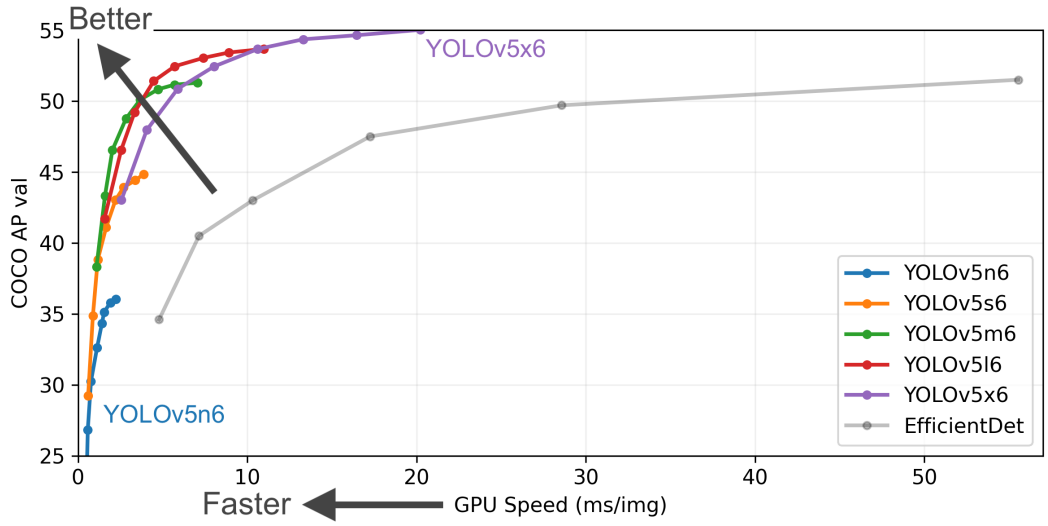


3.2.2 Approach 2: YOLOv5

The YOLO model was the first end-to-end object detector that combined the process of creating bounding boxes with assigning class labels. It is pretrained on the COCO dataset and contains 80 object classes.

YOLOv5 also had a fast detection output although due to hardware limitations, specifically a slow GPU; the use of this quick prototype was not as successful as other examples found online. Due to this issue, evaluation was performed with the two smaller variants of the YOLOv5

family model: YOLOv5n (nano) and YOLOv5s (small), which also meant lower overall accuracy rate.



Video output (screenshot)

4 Results

4.1 Experiment One

- Accuracy of Basic CNN model is 0.7462
- Accuracy of Tuned CNN model is 0.7916
- Accuracy of VGG16 model is 0.9853

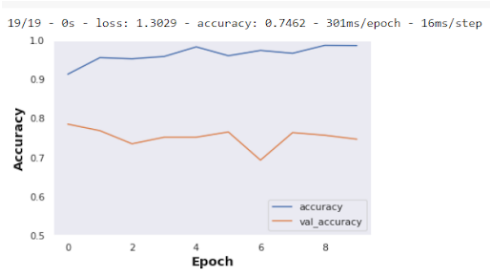


Figure 6: Basic CNN model result

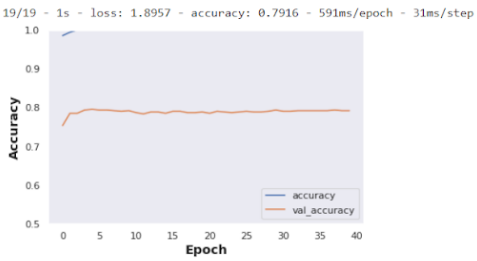


Figure 7: Tuned CNN model result

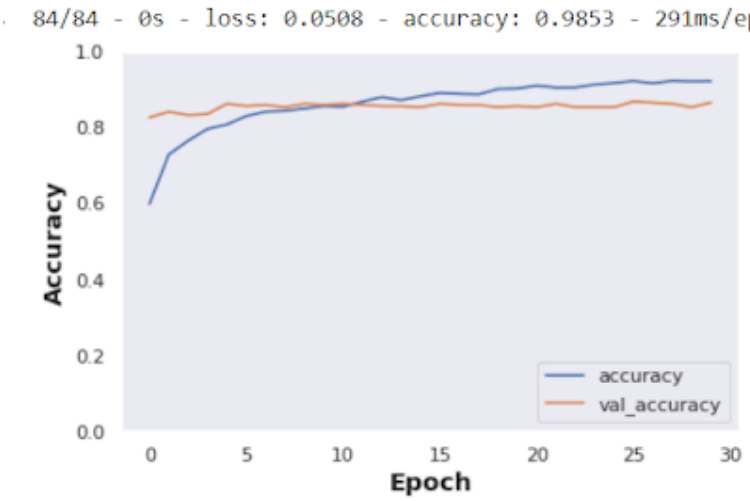


Figure 8: VGG16 model result

4.2 Experiment Two

YOLOv5n YOLOv5s

Model	size (pixels)	mAP^{val} 50-95	mAP^{val} 50	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5

Sourced from https://github.com/ultralytics/yolov5pretrained-checkpoints

5 Conclusion

FROM THE BRIEF:
discuss some of the results and comparison with other existing methods' limitations. Discuss possible future work.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," vol. 1, pp. I–I, 2001.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," vol. 1, pp. 886–893, 2005.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," pp. 580–587, 2014.

[5] I. uF reD F h ngD, "F en und tf unf p ti l yr mid ooling in heep gone volution l xetworks for isu le ognition f snx ieee transactions on pattern analysis and machine intelligence qufw@ phisad f iwhr," *IWITF doiX*, vol. 10, p. 1109.

[6] R. Girshick, "Fast r-cnn," pp. 1440–1448, 2015.

[7] S. Ren, K. He, and R. Girshick, "Jjitopa sun 2017 intelligence m," *Faster r-cnn: Towards real-time object detection with region proposal networks*, vol. 39, pp. 1137–49.

[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," pp. 2117–2125, 2017.

[9] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[10] Y. O. L. Once, "Unified, real-time object detection/redmon j., divvala s., girshick r., farhadi a," 2016.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," pp. 21–37, 2016.

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," pp. 2980–2988, 2017.

[13] B. Peng, Z.-B. Chen, E. Fu, and Z.-C. Yi, "The algorithm of nighttime pedestrian detection in intelligent surveillance for renewable energy power stations," *Energy Exploration & Exploitation*, vol. 38, no. 5, pp. 2019–2036, 2020.

[14] M. Wang, H. Chen, Y. Li, Y. You, and J. Zhu, "Multi-scale pedestrian detection based on self-attention and adaptively spatial feature fusion," *IET Intelligent Transport Systems*, vol. 15, no. 6, pp. 837–849, 2021.

[15] X. Chen, Y. Jia, X. Tong, and Z. Li, "Research on pedestrian detection and deepsort tracking in front of intelligent vehicle based on deep learning," *Sustainability*, vol. 14, no. 15, p. 9281, 2022.

[16] I. Hasan, S. Liao, J. Li, S. U. Akram, and L. Shao, "Pedestrian detection: Domain generalization, cnns, transformers and beyond," *arXiv preprint arXiv:2201.03176*, 2022.

[17] P. Ma, C. Li, M. M. Rahaman, Y. Yao, J. Zhang, S. Zou, X. Zhao, and M. Grzegorzek, "A state-of-the-art survey of object detection techniques in microorganism image analysis: from classical methods to deep learning approaches," *Artificial Intelligence Review*, pp. 1–72, 2022.

[18] J. Beal, E. Kim, E. Tzeng, D. H. Park, A. Zhai, and D. Kislyuk, "Toward transformer-based object detection," *arXiv preprint arXiv:2012.09958*, 2020.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," pp. 213–229, 2020.

[20] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3d object detection with pointformer," pp. 7463–7472, 2021.

[21] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[22] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," pp. 10 012–10 022, 2021.