

# Object Detection in Automobiles

1<sup>st</sup> Rami Alkafahje

2<sup>st</sup> Kai Perich

San Diego State University

San Diego, United States

alkafahje.ra@gmail.com,

kaiperich@gmail.com

## Abstract

Object detection has been widely utilized in the automobiles industry. In this paper, we collected our own automobile dataset from Kaggle and trained YOLOv3, YOLOv4, YOLOv5 models as well as R-CNN models with VGG16, ResNet50, DenseNet201, and EfficientNetB7 as the baseline on our own dataset. Next, We compared their convergence speed and accuracy on the automobile dataset. Furthermore, we analysed the results of the experiments qualitatively and quantitatively. In the end, we applied our best model onto the arbitrary dataset that contain automobiles to observe the performance.

## 1 INTRODUCTION

In recent years, there has been a major surge in research interest in the development of the autonomous car, which is an automobile platform capable of perceiving and reacting to its immediate environment in order to navigate roadways without the assistance of humans. Perception refers to the task of detecting the environment and includes a number of subtasks such as classification, 3D position estimation, simultaneous localization and mapping (SLAM) [1], and object detection. Specifically, Object detection is one of the most significant criteria for autonomous navigation in many autonomous driving systems, since it allows the car controller to account for barriers when calculating possible future trajectories.

Object detection refers to the capability of computer and software systems to locate and identify items in an image or scene. It creates a bounding box around the object and classifies the object's class. The rapid advances of deep learning techniques have greatly accelerated the momentum of object detection. With deep learning networks and the computing power of GPU's, the performance of object detectors and trackers has been greatly improved, achieving significant breakthroughs in object detection.

As the object detection became extremely popular, lots of models were created

to conduct the task, such as Region Based Convolutional Neural Networks (R-CNN) [2] and You Only Look Once (YOLO) [3]. The goal of R-CNN is to take an input image and produce a set of bounding boxes as output, where each bounding box contains an object and the category (e.g. car, walking people) of the object. However, it takes long time to extract regions for each image. Comparing to R-CNN, YOLO uses CNN to forecast multiple bounding boxes and class probabilities at the same time. The approach only takes a single forward propagation through a neural network, as the name suggests.

In this paper, we aim to compare the performance of applying YOLO models and RCNN models on automobile images. Precisely, we trained YOLOv3, YOLOv4, and YOLOv5 models, which were pretrained on the COCO dataset, on the same automobile dataset. Moreover, we trained R-CNN with different baseline models such as VGG16, ResNet50, DenseNet201, and EfficientNetB7, which were pretrained on the ImageNet dataset, on the same automobile dataset. Afterwards, we analysed the results of experiments qualitatively and quantitatively. It turned out that YOLOv5 outperformed the other models not only because of its faster convergence speed, but also because of its greater accuracy.

## 2 RELATED WORK

### 2.1 Data Augmentation

The goal of data augmentation is to increase the variability of the input images so that the designed object detection model can handle images from a variety of environments. Furthermore, some data augmentation researchers focused their efforts on simulating object occlusion issues. In terms of image classification and object detection, they've performed well. For example, random erase [28] and CutOut [29], can select a rectangle region in an image at random and fill it with a random or complementary value of zero.

Furthermore, some researchers have proposed methods for performing data augmentation by combining multiple images. For example, MixUp [30] multiplies and superimposes two images with different coefficient ratios, then adjusts the label with these superimposed ratios. CutMix [31], covers the cropped image to the rectangle region of other images and adjusts the label in accordance with the mix area's size. In addition to the methods mentioned above, style transfer GAN [32] is also used for data augmentation, and this method can effectively reduce CNN's texture bias.

### 2.2 Detector Structure

A modern detector usually has two parts: a backbone that has been pretrained and a head that predicts object classes and bounding boxes. Object detectors that were developed in recent years frequently insert layers between the backbone and the head, which are then used to collect feature maps from various stages. It's referred to as an object detector's neck. A neck is usually made

up of a number of bottom-up and top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN) [7], Path Aggregation Network (PAN) [14], BiFPN [13], and NAS-FPN [23]. In addition to the above models, some researchers have focused on developing a new object detection backbone (DetNet [24], DetNAS [25]) or a new whole model (SpineNet [26], HitDetector [27]).

### 3 DATASET

In this research, YOLO models were pretrained on COCO dataset, while R-CNN models were pretrained on ImageNet dataset. We used pretrained weights as the initialization of our models and trained our models on our own collected automobile dataset.



Figure 1: Automobile Dataset

#### 3.1 COCO Dataset

The MS COCO dataset is a large-scale object detection, segmentation, and captioning dataset published by Microsoft. Machine Learning and Computer Vision engineers popularly use the COCO dataset for various computer vision projects. Understanding visual scenes is a primary goal of computer vision; it involves recognizing what objects are present, localizing the objects in 2D and 3D, determining the object's attributes, and characterizing the relationship between objects. Therefore, algorithms for object detection and object classification can be trained using the dataset.

### 3.2 ImageNet Dataset

The ImageNet project is a vast visual database created to aid in the development of visual object recognition software. The project has hand-annotated over 14 million photos to specify what objects are depicted, with bounding boxes provided in at least one million of the photographs. There are almost 20,000 categories in ImageNet, with each category containing several hundred photographs. Though the actual images are not held by ImageNet, the database of annotations of third-party image URLs is freely available straight from ImageNet. Since dealing with a new dataset, even though we all know that in general meaning, YOLO is better than R-CNN.

2010, the ImageNet project has hosted the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual software competition in which software applications compete to accurately identify and recognize objects and scenes. A "trimmed" list of 1,000 non-overlapping classes is used in the challenge.

### 3.3 Automobile Dataset

We collected the automobile dataset from Kaggle. The total number of images were 1176. We split the data into training set (85%) and test set (15%), respectively. It should be noticed that in the training set, only 559 incidents contain the automobiles and their bounding box information. We recorded the bounding box coordinates referring to image index into a single csv file. Fig.1 shows several visualization examples of our own automobile dataset.

As can be seen in Fig.1, the images were taken on the street from a certain angle throughout all days. The qualities of images are relatively low, and the number of images as training set is small. Therefore, the automobile dataset brings great challenge to the models.

## 4 YOLO MODEL

YOLO is a Deep Neural Network (DNN) based object detection and localization technique that runs extremely quickly and can be employed in real-time applications. The author has released the newest version of YOLO, i.e, YOLOv5, which is a development of previous versions.

### 4.1 YOLOv3

DarkNet-53, the backbone feature extraction network employed by YOLOv3 [5], has two key features. One is the residual network [6], while the DarknetConv2D structure is the other from figure 2. Firstly, the images are compressed by the convolutional layers when entering DarkNet-53. Next, the residual convolution is utilized to generate the residual structure by combining the output of the residual convolutional layer with the original features. We further deepen the network by continually superimposing the residual edges, which forms the residual block.

Moreover, DarkNetConv2D structure applies L2 regularization on each convolution and utilizes LeakyReLU activation function. The application of L2 regularization can significantly reduce the risk of being overfitting and the Leaky ReLU assigns a non-zero slope to all negative values, allowing for a small and positive gradient.

The process of obtaining prediction results from features can be divided into two parts, i.e., building Feature Pyramid Networks (FPN) [7] for enhanced feature extraction and using YoloHead to obtain the prediction results.

YOLOv3 forms three feature layers, which are located at different positions from the backbone network, i.e., in the middle layer, lower middle layer, and bottom layer, respectively. After obtaining three feature layers, the FPN layer will be constructed based on them. Afterwards, three feature layers will be passed into the YoloHead to obtain the prediction results.

YoloHead is essentially a  $3 \times 3$  convolution combined with a  $1 \times 1$  convolution. The  $3 \times 3$  convolution is used for feature integration, while the  $1 \times 1$  convolution is utilized for adjusting the number of channels.

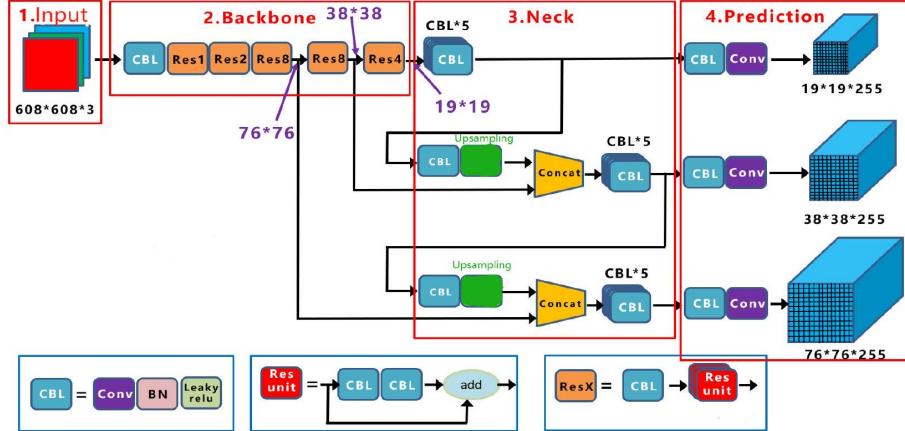


Figure 2: YOLOv3 architecture

## 4.2 YOLOv4

Based on YOLOv3, YOLOv4 [8] is created incorporating several tricks. In general, YOLOv4 manages to combine the speed and accuracy despite the lack of a revolutionary change in object detection. Specifically, YOLOv4 uses three feature layers for either classification or regression prediction, which is similar to YOLOv3. However, when comparing to YOLOv3, the author designed more complicated structure for YOLOv4's, which enables YOLOv4 to compute

a better performance. Improvements to the input side during training, including Mosaic data augmentation, Cross mini-Batch Normalization (CmBN) [9], and Self-Adversarial Training (SAT), can prominently increase the accuracy of YOLOv4.

Mosaic data augmentation is designed to avoid imbalance issues. Precisely, the input images will be concatenated together with ones through random scaling, cropping, flipping, and rotating. This method greatly enhances the dataset and increase the generalization ability of the model. Except the input side, various new approaches including CSPDarknet53 and Mish activation function are also added to the YOLOv4.

CSPDarknet53 is developed based on Darknet53, which contains 5 Cross Stage Partial (CSP) [10] modules. The convolutional kernel before each CSP module is with the size of  $3 \times 3$  with the stride of 2, which enables the downsampling. Additionally, CSP modules are used for splitting and merging the layer's feature mapping, which ensures accuracy while reducing the computational effort. Therefore, CSPDarknet53 structure can not only enhance the learning capability of CNN, but also reduce the computational bottlenecks and memory cost. Being unbounded above is a desirable property for any activation function, since it avoids saturation which generally causes training to drastically slow down due to near-zero gradients. Comparing to ReLu, Mish activation [11] outperforms ReLu in either training or validation. However, Mish is more expensive.

Object detection networks often insert layers between the Backbone and the final output layer, such as Spatial Pyramid Pooling (SPP) [12] module, Feature Pyramid Network (FPN) [7], and Path Aggregation Network (PAN) [14] structure. Regardless of the size of the input, SPP can produce the output with fixed size. SPP improves scale-invariance and reduces overfitting by handling different aspect ratios and sizes of the input image. Multiple pooling windows are also used and the experiments showed that having a variety of training image sizes makes network convergence easier than having only one size. Additionally, SPP has no effect on the structure of the network, as long as it is placed behind the last convolutional layer. The only change is the replacement of the original pooling layer.

To obtain the feature map for prediction, FPN upsamples the high-level features. YOLOv4 adds a bottom-up feature pyramid behind the FPN layer, as opposed to YOLOv3's FPN layer. The FPN layer conveys strong semantic features from the top down, while the feature pyramid conveys strong localization features from the bottom up, aggregating parameters from various backbone layers for various detection layers in this combined operation.

The Anchor box mechanism of the output layer is the same as YOLOv3. The main improvements are the replacement of Complete-IOU-Loss (CIoU-loss) function for training and the change from using the Non-Maximum Suppression (NMS) to utilizing the Distance-IOU-NMS (DIoU-NMS) [15] for prediction.

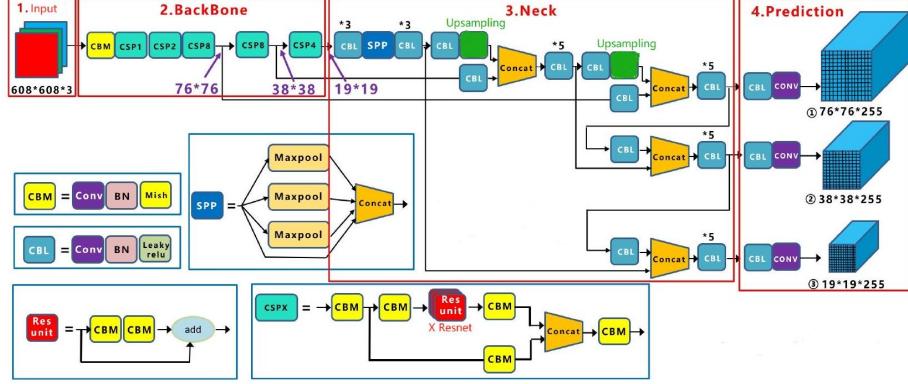


Figure 3: YOLOv4 architecture

### 4.3 YOLOV5

The structure of YOLOv5 is very similar to YOLOv4. The Fig.4 shows the structure of YOLOv5. It can be seen that YOLOv5 is also divided into four parts, i.e., Input, Backbone, Neck, and Prediction. First of all, auto learning bounding box anchors is applied in YOLOv5. For different data sets, there are anchor boxes with initial set of length and width. During training, the network outputs the prediction box based on the initial anchor box and compares it with the ground truth box to calculate the difference. Next, network parameters will be iterated through Back Propagation. In YOLOv3 and YOLOv4, when training different datasets, the calculation of the initial anchor box value is run by a separate program. However, YOLOv5 embeds this function into the code to calculate the best anchor box value for different training sets each time.

With the help of focus module, the computational power is enhanced without information loss. As the number of channels increases and the image shrinks, the amount of computation required decreases dramatically. In addition to focus module, other new methods were introduced. For instance, CSP structure, Activation function, FPN+PAN structure and so on.

The only difference between YOLOv5 and YOLOv4 is that YOLOv4 uses the CSP structure for the backbone network. In YOLOv5, two CSP structures are designed, and Leaky ReLU and Sigmoid activation function are used. Considering YOLOv5 network: the CSP1\_X structure is utilized in the Backbone network, while the CSP2\_X structure is utilized in the Neck network.

All standard convolution operations are used from the YOLOv4 Neck structure. Yolov5's Neck structure, employs the CSP2 structure from the CSPnet design to improve network feature fusion capability. Moreover, YOLOv5 adopts DIoUNMS based on Generalized Intersection Over Union (GIoUloss).

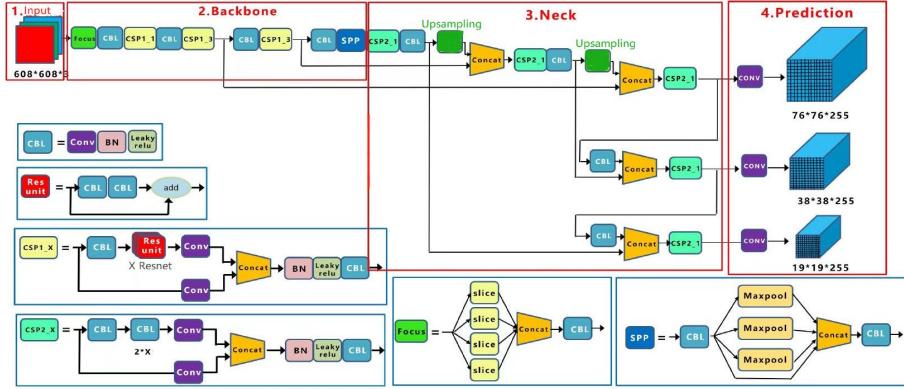


Figure 4: YOLOv5 architecture

## 5 R-CNN MODEL

The RCNN algorithm is divided into 4 steps, i.e., input, region proposals generation, feature extraction, Class determination, and position refinement.

After the images went into the model, the model will generate approximately 2,000 region proposals. Next, fixedlength feature vector will be extracted using a Convolutional Neural Network (CNN). Afterwards, A classifier for each class will be used to determine whether a feature vector belongs to a certain class. Lastly, the model will utilize the regressor to refine the position of region proposals. Specifically, the basic concept behind selective search [17] is that each input image is divided into approximately 2000 small regions. The two adjacent regions with the highest probability (most similar) merged using the merging rule. Repeating until there is only one complete picture left. All previously existing regions are outputs, so-called region proposals. Object detection requires not only locating the bounding box of the object, but also identifying the object within the bounding box. R-CNN recognizes multiple rectangular boxes in an image that could be objects, and then assigns a classification probability to each rectangular box to determine which rectangular boxes are useless. Non-maximum suppression (NMS) [18] is achieved by sorting rectangular boxes from smallest to largest based on the classifier's probability.

R-CNN starts with the maximum probability rectangular box to determine whether the IOU of each other rectangular box is greater than the set threshold. The model will remove any boxes with an IOU greater than the threshold and leave only the maximum probability rectangle. Then, it will select the box with the highest probability from the remaining rectangular boxes. If its IOU with other boxes is greater than a certain threshold, the model will throw it away and mark the second rectangular box with the highest probability. R-CNN simply keeps conducting this until all of the rectangular boxes are found in order to suppress the non-maximal elements and look for local maximum values. Espe-

cially, the CNN baseline of R-CNN can be replaced as any existing CNN model, such as VGG16, ResNet50, DenseNet201, and EfficientNetB7.

Our paper will not discuss the RCNN Model in depth, but our results of the RCNN Model will still be presented.

## 6 EXPERIMENT AND DISCUSSION

In this section, we conducted several experiments to compare the performance of different models on our own collected automobile dataset.

### 6.1 YOLO Performance Comparison

The YOLO models were pretrained on the COCO dataset. We used the pretrained weights as the initialization of our models. In order to control variables, we set the same hyperparameters for YOLOv3, YOLOv4, and YOLOv5, respectively. The training epoch was designed as 30.

As shown in table 1, the model we used is small (S) due to the limited computational power. Other optional models are medium (M) and large (L), which can be found in the author’s GitHub. Additionally, the python script for training requires one specific file (.yaml) to point out the path of training set, validation set, and test (optional).

Fig.5 demonstrates the performance of different YOLO models. The precision of YOLOv3, YOLOv4, and YOLOv5 reached 90.20%, 96%, and 98.10%, respectively. Based on the theoretical analysis, we’ve known that YOLOv4 is an improved version of YOLOv3, and YOLOv5 is an improved version of YOLOv4. The validation accuracy was increased after stacking several tricks such as CSP-DarkNet53, FPN, and PAN. Additionally, as mentioned before, the quality of dataset is with concerns, and the number of dataset is small. However, the performance of YOLOv5 is excellent under such a circumstance.

| Setting       | Type/Value |
|---------------|------------|
| Model Size    | S          |
| Optimizer     | SGD        |
| Learning Rate | 0.0100     |
| Momentum      | 0.9370     |
| Weight Decay  | 0.0005     |
| Batch Size    | 4          |

Table 1: Hyperparameters for YOLO

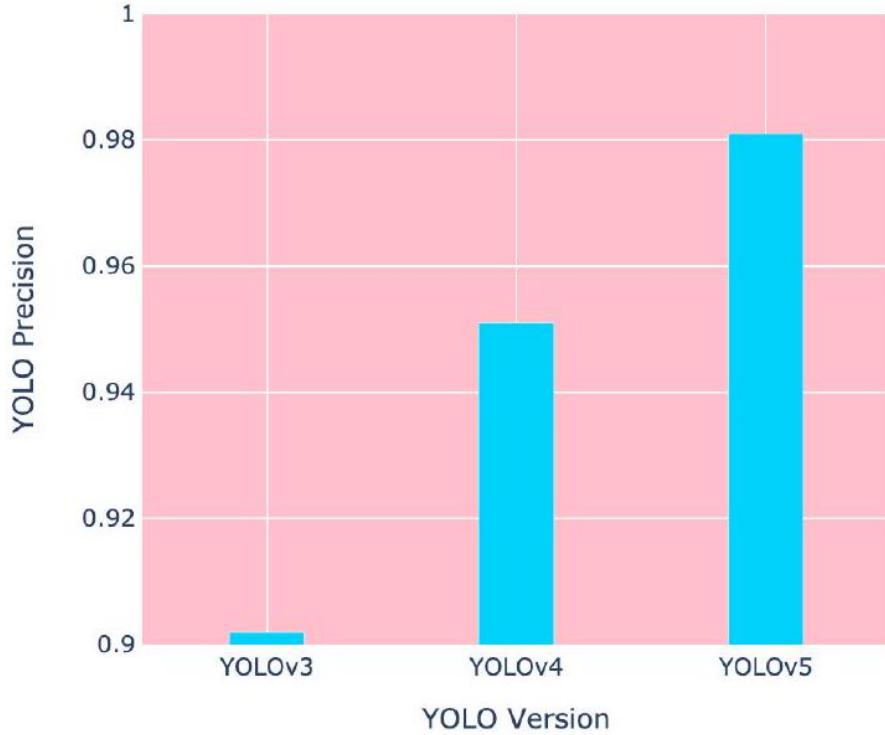


Figure 5: YOLO Comparison

## 6.2 R-CNN Performance Comparison

The R-CNN models were pretrained on the ImageNet dataset, we used the pretrained weights as the initialization of our models. In order to observe the performance of R-CNNs with different CNN baselines, we replaced the CNN baseline as VGG16, ResNet50, DenseNet201, and EfficientNetB7, respectively. The training epoch was designed as 50.

As shown in the table II, the optimizer we used for RCNN models changed to Adam. Comparing to SGD, the recommended learning rate of Adam decreases to 0.001. Specifically, the batch size for all R-CNN models is set as 1. As we split the training set into training set (67%) and validation set (33%), the robustness for convergence can be ensured. Fig.6 demonstrates the performance of different R-CNN models. The oldest CNN baseline VGG16 computed the best accuacy (97.77%), while the accuracy of the R-CNN model with the state-of-the-art CNN baseline EfficientNetB7 only reached 97.34%. ResNet50, as expected, could usually compute a satisfactory accuracy of 97.59%, while DenseNet201 seems not to be quite adaptable to our automobile dataset. Its accuracy only reached 96.1 %. The NFL theorem might be a good explanation for the result.

Namely, there is no generally best model that is suitable for all datasets. For a new dataset, the oldest model might even outperform the state-of-the-art.

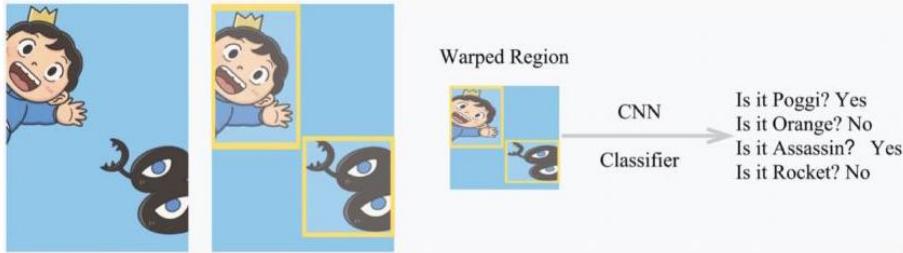


Figure 6: RCNN Comparison

| Setting       | Type/Value |
|---------------|------------|
| Optimizer     | Adam       |
| Learning Rate | 0.0010     |
| Beta 1        | 0.9000     |
| Beta 2        | 0.9990     |
| Epsilon       | 1e $-7$    |
| Batch Size    | 1          |

Table 2: Hyperparameters for CNN

### 6.3 Best Model on Arbitrary Dataset

Since the training set is small and the quality is bad, we aim to observe the performance of the best model i.e., YOLOv5 on several unseen datasets. We conducted three experiments in this section, i.e., testing YOLOv5 on the Taxi Car Video, Police Car Video, Sports Car Images, respectively.

As can be seen from Fig. 1, automobiles from the training set are mostly black, white, and red. Due to the fixed angle of the camera, we only have the side-body images from cars. However, the automobiles from the Taxi Car Video are mostly yellow, and some of them even showed the front body in the video. Even if it is quite challenging for a model to detect the object with such a limited dataset, YOLOv5 is still able to accomplish the task, as shown from the first row in Fig.7. The reason might be that the images from the training set were taken at different time points, such that the sunlight varies, which is similar to data augmentation and indirectly increased the generalization ability of the model. It should also be noticed that, when detecting the cars with yellow color

or front direction, the confidence will be significantly reduced (10%-30%). The second row showed the detection results on the Police Car Video. The cars from the video are with the mixed color of black and white. The size varies significantly from extremely small (far away from the camera) to a large size (close to the camera). Additionally, the cars in several frames are quite blurry, as shown in the second row (middle), due to the fast move of police cars. However, YOLOv5 proved its capability again. It can perfectly detect all the cars with high confidence, even if the cars are blurry or with a small size. One mistake that YOLOv5 made is that it falsely detected the bicycle as a car, which can be found in the second row (right). As mentioned, the weights we used as the initialization are pretrained from the COCO dataset. In the original yaml file, the author defined 80 classes in total including a "bicycle" class. Therefore, it might influence our weights, since our dataset is too small and might be not enough for training to differentiate the cars and bicycles perfectly. The third row demonstrated the detection results on the Maserati, Ferrari, and Lamborghini image, respectively. Since the original dataset does not include any sports car images, we were curious about if YOLOv5 is still able to detect them. As shown from the Fig.7, the Maserati cars can be perfectly detected with high confidence (88%, 92%, and 89%, respectively). For Ferrari, even though YOLOv5 can still detect the cars, the confidence went down significantly due to some unseen and unique colors and shapes of cars (green Ferrari with a confidence of only 46% ), as can be found from the third row (middle). Moreover, it becomes extremely tricky for YOLOv5 to detect Lamborghini cars, as the shapes are much flatter, and the colors are more rarely to see. Previously, we set the threshold of confidence as 0.3, such that the bounding box will not appear under the threshold. As the image from the third row (right) showed, for the green Lamborghini, the bounding box didn't even show up due to the low confidence.

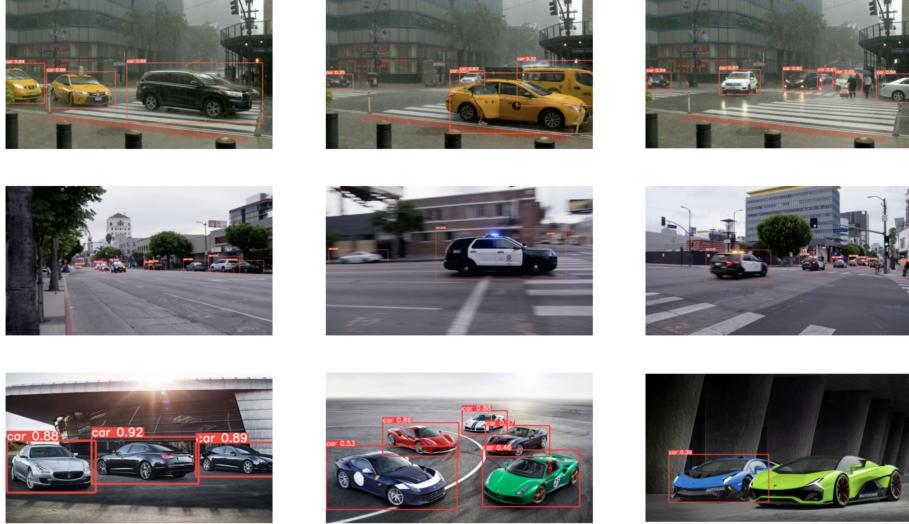


Figure 7: Arbitrary Dataset

## 7 Potential Improvement

There are several possibilities to improve the performance of models, such as increase of dataset, data augmentation, and replacement of optimizer. As already discussed, the main issue for detecting taxi and sports cars is the lack of certain type of cars in the training set. Increasing the dataset and adding more types (more different colors, shapes, and sizes) of cars into the training set will certainly help machine to detect automobiles.

With the help of data augmentation, the images can be rotated, blurred, mirrored, recolored, etc. This can simulate the automobiles under different scenes and will increase the generalization ability of models. For instance, changing the color of cars in the training set to yellow or green might help machine to detect the Taxi or Ferrari cars with a higher confidence.

The optimizer can indeed influence the convergence of models. For YOLO and R-CNN, we utilized SGD and Adam, respectively. There are other great optimizers such as AdamW, RMSprop, which might lead the model to a better minimum and thus give us a greater accuracy.

## 8 CONCLUSION

In this paper, we researched on different YOLO models, i.e., YOLOv3, YOLOv4, and YOLOv5, as well as RCNN models with different baselines, i.e., VGG16, ResNet50, DenseNet201, and EfficientNetB7. We introduced their structures and advantages. Based on the theoretical analysis, we conducted several experiments. Firstly, we collected our own automobile dataset from Kaggle. Next, we implemented seven models, utilized their pretrained weights, and trained them successfully on the dataset. Then, we compared their performance and analysed the results qualitatively and quantitatively. Next, we selected the best model i.e., YOLOv5, to test its performance on the unseen videos and sports car images. Lastly, we discussed reasons for failures and provided potential improvements for the future work.

## 9 REFERENCES

- [1] M.Montemerlo,S.Thrun,D.Koller,B.Wegbreit,etal.Fastslam: A factored solution to the simultaneous localization and mapping problem. 2002.
- [2] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.
- [3] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
- [4] Yao, Z., Cao, Y., Zheng, S., Huang, G. and Lin, S., 2021. Cross-iteration batch normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12331-12340).
- [5] Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
- [6] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [7] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).
- [8] Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv: 2004.10934
- [9] Yao, Z., Cao, Y., Zheng, S., Huang, G. and Lin, S., 2021. Cross-iteration batch normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12331-12340).
- [10] Wang, C.Y., Liao, H.Y.M., Wu, Y.H., Chen, P.Y., Hsieh, J.W. and Yeh, I.H., 2020. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 390-391).
- [11] Misra, D., 2019. Mish: A self regularized non-monotonic neural activation function. arXiv preprint arXiv: 1908.08681, 4, p.2.
- [11] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R. and Ren, D., 2020, April. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 12993-13000 ).
- [12] He, K., Zhang, X., Ren, S. and Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), pp.1904-1916.
- [13] Tan, M., Pang, R. and Le, Q.V., 2020. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10781-10790).
- [14] Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J., 2018. Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8759-8768).

- [15] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R. and Ren, D., 2020, April. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 12993-13000 ).
- [16] DeVries, T. and Taylor, G.W., 2017. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552
- [17] Uijlings, J.R., Van De Sande, K.E., Gevers, T. and Smeulders, A.W., 2013. Selective search for object recognition. International journal of computer vision, 104(2), pp.154-171.
- [18] Neubeck, A. and Van Gool, L., 2006, August. Efficient non-maximum suppression. In 18th International Conference on Pattern Recognition (ICPR'06) (Vol. 3, pp. 850-855). IEEE.
- [19] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
- [20] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
- [21] Tan, M. and Le, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.
- [22] Elsken, T., Metzen, J.H. and Hutter, F., 2019. Neural architecture search: A survey. The Journal of Machine Learning Research, 20(1), pp.19972017.
- [23] Ghiasi, G., Lin, T.Y. and Le, Q.V., 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7036-7045).
- [24] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y. and Sun, J., 2018. Detnet: Design backbone for object detection. In Proceedings of the European conference on computer vision (ECCV) (pp. 334-350).
- [25] Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X. and Sun, J., 2019. Detnas: Backbone search for object detection. Advances in Neural Information Processing Systems, 32, pp.6642-6652.
- [26] Du, X., Lin, T.Y., Jin, P., Ghiasi, G., Tan, M., Cui, Y., Le, Q.V. and Song, X., 2020. SpineNet: Learning scale-permuted backbone for recognition and localization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11592-11601).
- [27] Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X. and Xu, C., 2020. Hit-detector: Hierarchical trinity architecture search for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11405-11414).
- [28] Zhong, Z., Zheng, L., Kang, G., Li, S. and Yang, Y., 2020, April. Random erasing data augmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 07, pp. 13001-13008).
- [29] DeVries, T. and Taylor, G.W., 2017. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

- [30] Zhang, H., Cisse, M., Dauphin, Y.N. and Lopez-Paz, D., 2017. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.
- [31] Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J. and Yoo, Y., 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 6023-6032).
- [32] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A. and Brendel, W., 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv: 1811.12231.