Edge Artificial Intelligence for Road User Detection in Different Weather Conditions with Data Augmentation.

Hung Lo, Master Student
Smart Transportation Application and Research Lab, Department of Civil and Environmental Engineering
University of Washington
honkuro@uw.edu

Ruimin Ke, PhD
Assistant Professor
Department of Civil Engineering
The University of Texas at El Paso
rke@utep.edu

Corresponding Author
Yinhai Wang, PhD, PE
Professor and Director, Smart Transportation Application and Research Lab, Department of Civil and Environmental Engineering
Pacific Northwest Transportation Consortium, USDOT University Transportation Center for Federal Region 10
University of Washington
Seattle, WA 98195-2700
yinhai@uw.edu
206-616-2696

Total number of words: 6,437 (text) + 750 (3 tables) = 7,187

Number of figures: 13

Submitted for presentation at the TRB 101st Annual Meeting

Submission Date: August 1, 2021

ABSTRACT
The technology of autonomous vehicles (AVs) is developing at a fast pace. When AVs enter the traffic systems, it brings lots of opportunities and challenges. The major challenges in AV perception tasks are the lack of extreme corner cases (e.g., extreme weather conditions), correctly identifying vulnerable road users (VRUs), e.g., cyclists and pedestrians, and the trade-off between the computation cost and the operation performance. To address these challenges, we propose a lightweight artificial intelligence model which can detect road users in real-time on edge devices. We further investigate new data augmentation methods against weather noises in the road user detection scenario. With the proposed data augmentation pipeline to provide robust detection in extreme corner weather conditions, the performance of our method can improve 5% in different weather conditions. Our system is evaluated with the KITTI datasets. Performances are evaluated for road user detection, under three different weather conditions, and with different image input filters. Our experiment indicates that the proposed method achieves a state-of-the-art mean accuracy of 66% on VRU detection in extreme weather conditions at a real-time speed of ~112 FPS on Tesla V100 and 27 FPS without GPU. The system has a great potential to tackle the practical challenges of road user detection cost-effectively for AVs.

Keywords: Autonomous vehicles, environment perception, weather augmentation, object detection, object classification, road user detection, computer vision.

INTRODUCTION

Driven by fast advances in both hardware and software, operation and research on autonomous vehicles(AVs) have significantly progressed in recent years. On the other hand, a great number of extreme challenges occurred and must be solved before the widespread adoption of AVs involved in the traffic system. Legal and supervising issues, safety and public perspective of AVs technologies, availability and reliability of edge sensing equipment, and computing resource limitations of current IoT systems and artificial intelligence algorithms of AVs are the challenges that need to be solved for the total acceptance of AVs. Environment perception is one of the important fields for AV development; In order to let automated driver assistance systems make decisions, accurate perception is needed. Components of the environment include these objects: traffic signals, road markings, lanes, and other road users (vehicles, cyclists, and pedestrians). Among these objects, the pedestrians and cyclists remain among the most challenging for AV perception systems to detect accurately, but crashes with them lead to the most severe losses. This issue shows the high proportion of vulnerable road users (VRUs) among traffic fatality records. In recent years, traffic crashes have been the leading cause of fatalities worldwide for people. In 2013, there were 1.25 million people who died on roads worldwide. Half of these fatalities were VRUs (1). A similar situation can be found during 2017 in the European Union (2). Therefore, proposing a reliable road user detection model to road users is necessary. Before AVs can ensure the minimal risk that they pose to VRUs, the public would not accept AVs entering the traffic system.

Edge computing is one of the emerging concepts that is proposed to process sensor data to where they are generated. One of the challenges for AVs to make accurate detection is low computing power on edge devices. Compared to the cloud-based service, the computing resources for the edge device is limited. In addition, the low-latency computer vision model is also a high requirement since AVs need to agilely make decisions for a diverse environment in a real-time manner. The onboard object detector needs to respond quickly to approach the complex intersecting with the environment. Therefore, developing a lightweight and high-speed artificial intelligence model has become a popular topic as the edge computing requirement increases. The other factor that will disturb the accuracy of detection is environment noises. In the real-time settings, the weather noises will diminish the performance of the model prediction. For instance, over exposure will generate the glare and the edge of the object to be blurred. Rainy and foggy weather will increase the environment noises too. To be more specific, the steam will block the object in the scene. Therefore, the method to against the weather noises is the way to improve the accuracy of road user detection.

This paper provides a literature review on the state-of-the-art in object detection, then proposes an edge artificial intelligence model to detect road users with efficiency, reliability, and fast deployment; particularly in extreme weather conditions with an image augmentation method. These three metrics are as follows: efficiency is about the real-time frame per second; Performance is the accuracy of the model to detect the road user in various weather conditions. Following the recent research on artificial intelligence and computer vision, we implemented an image pipeline to do the weather noise augmentation and object detector fine-tuned on a road user benchmark dataset on the edge devices. This paper proposed the synthetic dataset pipeline to increase the model robustness for weather noises, a YOLO-Ghostnet detector that performs high availability and consistency results for the real-time operation, and the pipeline to deploy the YOLO-Ghostnet detector on edge devices.

LITERATURE REVIEW

KITTI Dataset

Data is always the most important thing to get a high-performance deep learning model, so this section gives an overview of the KITTI dataset (3) which is most relevant for road user detection. The public dataset is one of the main factors which is enabled to improve the computer vision model. As well as removing the requirement for researchers to have access to physical computer vision device collections, many of these datasets have dedicated benchmarking websites where new models may be independently evaluated using standard processing. KITTI dataset is a public access dataset. It provides LIDAR and camera frames. The collecting vehicle has two-point gray fleas and two video cameras, equipped with the GPS and Velodyne HDL-64E Laserscanner. The highly accurate LIDAR data also serves as a ground truth label, allowing the comparison of image-based detection algorithms against 3D point cloud data. The KITTI dataset contains the city, country, and highway scenario. the vehicle, pedestrians, and cyclists in one single image is up to 15. It provides a very detailed information scenario for training. For instance, it has the truncated and occluded label in the dataset, which can provide more features for machines to learn. KITTI is therefore currently the best choice for researchers to develop and test new algorithms for road user detection.

FIGURE 1 The key hardware components of the IoT system: standard station wagon with two high-resolution color and grayscale video cameras. And utilizing Velodyne laser scanner and a GPS localization system to get the ground truth.

Data Augmentation

Instead of getting more dataset, data augmentation is also the proved and useful method to increase accuracy of the deep learning model. Data augmentation can increase input images with more variance so that the object detection model would be robust to the images captured from different scenarios and avoid the overfitting of the image. Photometric distortions and geometric distortions are two generally used data augmentation methods that have benefits for improving the object detection model. Commonly, we do photometric distortion and adjust the noise, saturation, contrast, hue, and brightness of an image. As for geometric distortion, random flipping, scaling, rotating, and cropping are considered.

These are pixel modifications that would remain original pixel information in the adjusted area. At the same time, some researchers proposed data augmentation and put their emphasis on simulating object occlusion issues. They have achieved good results in image classification and object detection. For example, Mosaic (4) and CutOut (5) can randomly select the rectangle region in an image and fill in a random or padding value of zero. The other concept is randomly or evenly selecting multiple regions in an image and closing their value into zero. The other like close some pixel cases is the dropout design. To avoid the overfitting issue, we will hide some kernel or the feature during the training, e.g., there are DropOut (6) and DropBlock (7) methods. Furthermore, using multiple images together to advance data augmentation is one useful technique, especially for small object detection. For instance, MixUp (8) uses two images to compose and scale with different coefficient ratios and then adjusts the bounding box (BBox) with these composing ratios. As for CutMix (9), it is to overlap the selected image to the target area of other images and modify the BBox with the size of the mixing area. Finally, increase the noise or generate a fake dataset that can train the model against the noise and avoid the texture bias or overfitting made by convolutional neural networks (CNN).

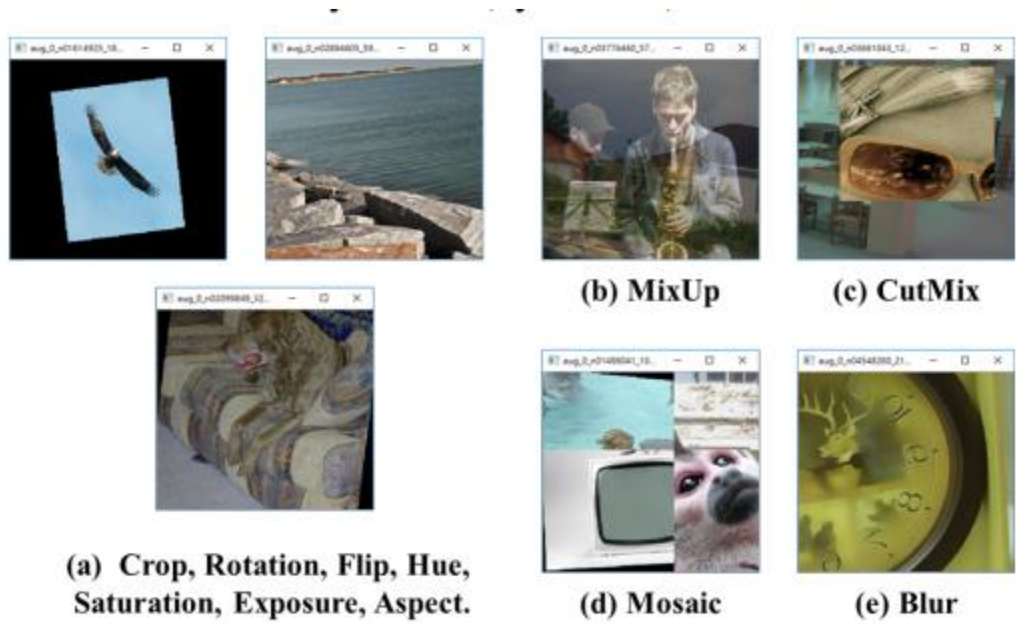

FIGURE 2 Various image augmentation approaches.

Object Detection Networks

Overview of Recent Object Detection Networks

A modern object detector is usually composed of two parts, one is the backbone and the other is the head. A backbone is a CNN that pools image pixels to form features. Typically, the backbone is pretrained on a benchmark dataset. The head is used to predict BBox and classes of objects. Here is the common backbone network which is running on the GPU platform. VGG (10) , ResNet (11) , ResNeXt (12) or DenseNet (13) . On the other hand, these models are running on CPU platforms, such as SqueezeNet (14) , MobileNet (15) , EfficentNet (16) , or ShuffleNet (17) . There are two main trends for the head part. One is a one-stage object detector and the other is a two-stage object detector. The most representative two-stage object is the R-CNN (18) series. As for one-stage objects, the most representative are YOLO (19) and SSD (20).

At CVPR2016, the YOLO algorithm was proposed by Redmon et al. During two years of development, YOLO developed from V1 to V3 (21) . There are several techniques used in the model. One technique is add-on modules and post-processing methods that only increase the calculation cost by a little bit but can significantly advance the accuracy of object detection. Generally speaking, these add-on modules are for enhancing certain attributes in a model, e.g., increase the receptive field, introducing attention mechanism, or advance feature integration capability, etc., and post-processing is a method for screening model prediction results. In general, SPP (22), ASPP (23) and RFB (24) modules are widely used to enhance the receptive field. The SPP module was created from Spatial Pyramid Matching (SPM) (25) . The idea of SPMs was to split the feature map into several $d \times d$ equal blocks, where d is {1, 2, 3, ...}, thus forming a spatial pyramid, and then it will extract bag-of-word features. Instead of extracting bag-of-word features, SPP uses a max-pooling operation which is done by CNN operation. Since the SPP module will output a one-dimensional feature vector, it is infeasible to be applied in Fully Convolutional Network (FCN). In the design of YOLOv3, Redmon and Farhadi improve the SPP module to the concatenation of max-pooling outputs with kernel size $k \times k$, where $k = \{1, 3, 5, 9, 13\}$, and stride equals to 1. Under this design, a relatively k × k max pooling validly increases the receptive field of the backbone feature.

Vanishing Gradient

Traditionally, vanishing gradient for gradient-based learning methods and backpropagation is the challenge for the deep neural network. In order to solve this issue researchers proposed different activation functions to prevent gradient vanish. For instance, ReLU (26) , LReLU (27) , Scaled Exponential Linear Unit (SELU) (28), Swish (29) , hard-Swish (30) , etc., which are also used to solve the vanishing gradient problem, have been proposed. The main purpose of LReLU is to solve the problem that the gradient of ReLU is zero when the output is less than zero. As for hard-Swish, they are designed for quantization networks. For self-normalizing a neural network, the SELU activation function is proposed to satisfy the goal.

Objective Function

The objective function of the anchor BBox Regression is the key component for the object detection network. Traditionally, it usually uses Mean Square Error(MSE) for BBox and anchor methods. As for anchor, a corresponding offset is used, i.e., offset between x,y, width, height, or offset between four corners top left, top right, bottom left, bottom right. As for the BBox between the center point coordinates and width height of BBox, for instance, $\{Center_x, Center_y, Offset_x, Offset_y\}$. Or use the four boundaries $\{X_0, X_1, Y_0, Y_1,\}$. However, the problem for the l1 or l2 loss will increase with the scale. To solve this problem, researchers proposed the IoU loss. IoU (31) is the loss function converge area between predicted BBox area and ground truth BBox area into consideration, this scale-invariant representation can avoid Homoscedasticity issues. In YOLOv5 (32) , it decides to use GIoU (33) loss which also considers the shape and orientation of the converging area.

YOLOv5

YOLOv5 did improvements to reduce the space and the time complexity between the layer and the input of the image. The following section will introduce the improvement which YOLOv5 achieves. First, as we mention the YOLO learning provides lots of anchors and BBox for object detection, and the loss function includes the anchor and BBox loss is the main target for models wanting to learn. In YOLOv5, it proposes a self-learning anchor design. The idea is to calculate the predicted BBox difference between ground truth BBox. Therefore, during the training, the network will reweight the BBox and anchor corresponding to the historical difference.

In order to make the image size consistent the training for the different sizes of the image will be padded with the empty pixel. However, this will increase the operation time for the learning of the image, also the zero value is meaningless for the learning. Therefore, the YOLOv5 provides the method, during the down sampling YOLOv5 design, the self-learning resizes algorithm to reduce the padding of the image corresponding to the coefficient between width and height. Based on this concept, YOLOv5 saves the learning time for the image padding part.

YOLOv5 proposes the focus layer to reduce the time of two-dimension convolutions. The focusing layer transforms image data from space to depth by splitting the large image. Generally, in a normal backbone, e.g., ResNet, we have a stem layer from the bottom with two-dimension convulsions to reduce the resolution and increase the number of channels. In order to reduce the cost of two-dimension convolutions, the YOLOv5 proposes to split the image and do the partial convolution then concat the split image back to the deep but narrow data format. Input will be transformed likes this [batch size, channel, height, width] into [batch size, channel*2, ${}^{height}/_{2}$, ${}^{width}/_{2}$].

Lightweight Network Architecture

Driven by hardware development, there is lots of research trying to launch the deep neural network on the edge device. Generally, there are two ways to increase the inference speed of the model, use high-performance devices or model compression. However, due to computing resource limitations for the edge devices compared to cloud-based services, researchers can only try to compress the model for the deep neural network running on the edge device. First, pruning unnecessary connections or channels between neurons can reduce the cost of operation. Model quantization represents weights or activations in neural networks with discrete values for compression and calculation acceleration. Specifically, binarization methods with only 1-bit values can accelerate the model by efficient binary operations compared to float data types. The other technique is tensor decomposition which reduces the parameters or computation by exploiting the redundancy and low-rank property in the weights. Knowledge distillation utilizes larger models to teach smaller ones, which improves the performance of smaller models.

As the growing demand for deploying neural networks on edge devices, a series of compact models have been proposed in recent years. MobileNet series are widely used lightweight deep neural networks based on depth-wise separable convolutions. MobileNetV2 proposes inverted residual blocks and MobileNetV3 further utilizes AutoML technology achieving better performance with fewer FLOPs. ShuffleNet introduces channel shuffle operation to improve the information flow exchange between channel groups. ShuffleNetV2 further considers the actual speed on target hardware for compact model design. Although these models obtain great performance with very few FLOPs.The correlation and redundancy between feature maps have never been well exploited, so Ghostnet comes out.

To reduce the computational costs of recent deep neural networks, Ghostnet (34) presents the Ghost module for building lightweight neural networks. Generally, to create several intrinsic feature maps, Ghost modules separate the original convolutional layer into two parts and apply fewer filters which can save the duplicate convolution. In addition, the design of cheap transformation operations can produce feature maps efficiently.

PROPOSED SOLUTION AND DESIGN

Label Data Pre-processing

Original KITTI dataset annotations are not the same as the YOLOv5 annotations, since the KITTI BBox is {top left, top right, down left, downright}, the YOLOv5 annotations are {center x, y, x offset, y offset}, we need to transfer it into the YOLOv5 format. In addition, there is a diverse label in the KITTI dataset, but we only retain the BBox, truncate, block, and category of the dataset to our training labels. We only care about pedestrians, cyclists, and vehicles, so we combine all vehicles into one label: car. As a result, we only remain in three categories in the dataset: pedestrians, cyclists, and cars. Finally, the common settings for the proportion of the train: validation: test is 8:1:1, we set up this distribution.

Weather Data Augmentation

In the introduction, we talk about how the weather noise decreases the accuracy of road user detection. As we discussed the data augmentation method in literature review, the data augmentation can train the model against the noise and optimally use the features from the data. On the other hand, we cannot always ensure we can get a large enough dataset for different scenarios. Inspired by the GANs, synthesizing the image with specific noises is considered the powerful solution to train the model. The next step is finding the feature that we need to add to our ground truth KITTI dataset. It is possible to get more datasets under different weather scenarios. To synthesize the different weather conditions, we need to modify the normal image into the image with different weather noises. In "Single Image Haze removal using dark channel prior" (36) provides the way to dehaze the image into a clear image. In contrast, we can use the haze step to achieve the synthesizing of foggy images and other weather. In this paper, we focus on developing the synthesized image based on rainy, foggy, and overexposed conditions. First, we need to determine the factor for the different conditions.

<u>Rainy Data Augmentation</u>

First, we need to determine the color of the drop. In our test, the gray drop is most close to the real rainy situation. Second, we define the factor which can allow the different densities of the drop, from drizzle to rain cats and dogs. Third, on rainy days, the environment light will be reduced, we call this the alpha, which controls the illumination of the image. Finally, we display this change on our image, then we get the rainy condition image. Here is the function to create the rainy condition. Alpha is the illumination factor. On a rainy day, the light will be weaker than normal. In this paper, we determine alpha=0.7 will be a suitable setting. Second, we create a random line to present the real drop. Finally, we need to determine how many drops need to be present in the image depends on area of image.

Figure 3, rainy image augmentation

Overexposed Data Augmentation

In the overexposed case, we need to consider the upper and lower bound of the pixel value. Therefore, we define the alpha and beta parameters. Alpha is the factor for the power of the current value, the beta is the shift value that makes the pixel change the illumination. Here is the function of the pixel change in overexposed conditions. Alpha is the factor controlling the overexposed strength, the beta is the shift factor, to increase the illumination of the image.

$$pixel_{channel,height,width} = pixel_{channel,height,width} * alpha + beta \qquad (1)$$

Figure 4, Overexposed image augmentation

Foggy Data Augmentation:

There are four parameters to create a foggy image. First is the center, which indicates the center of the fog, this factor will control the impact of the haze. The closer to the center, the haze will be more serious. Second, alpha is the brightness factor, which controls the brightness of the pixel. Third, beta is the density of the fog, lower beta means clearer sight. Here is the function to approach the foggy image.

$$d = -0.04 * ((x - center_x)^2 + (x - center_x)^2) + size \qquad (2)$$

d is the distance factor, the position far from the $(center_x, center_y)$ will be less foggy

$$pixel_{channel,height,width} = pixel_{channel,height,width} * e^{-beta*d} + Alpha * (1 - beta * d) \qquad (3)$$

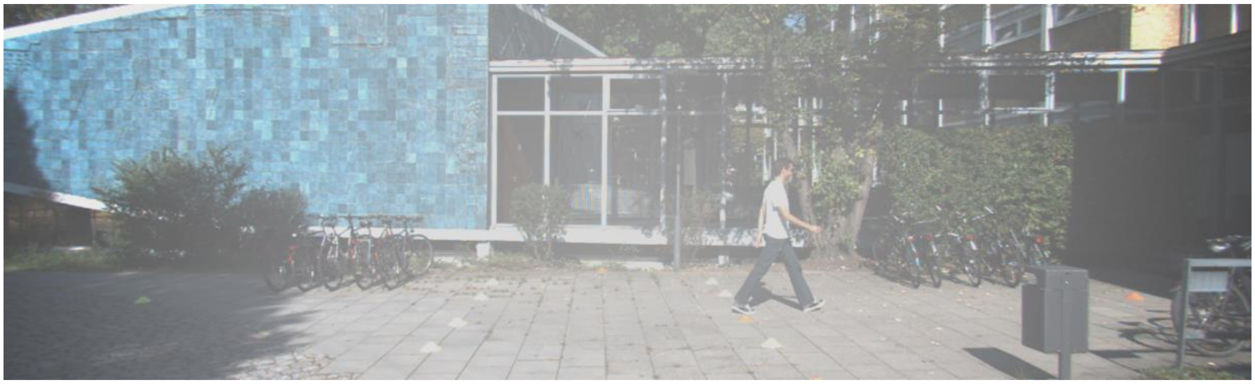

Figure 5 Foggy image augmentation

Network Structure Improvement and Architecture of YOLO-Ghostnet:

As we discuss in the literature review section, YOLOv5 has high accuracy and a short inference time for deployment and training. However, there are some issues that the YOLOv5 can be improved. In this paper, we decide to use the Ghostnet as our method's backbone, because Ghostnet has the most lightweight structure which makes the detection faster than other backbones. This is appropriate for an IoT device with limited computational resources. In addition, Ghostbottlenect performs better than CSPbottlenect used in the YOLOv5, so we decide to use the Ghostbottlenect to replace the CSPbottlenect. In addition, we decided to replace all the activation layers with LReLU which is saving lots of time than the swish function and only decreases a bit accuracy. The backbone with the lightweight and efficient property can satisfy our requirement for light computing resources.

As the pipeline shows, there are three main parts for the YOLO-Ghostnet, backbone, head, and prediction. In the backbone part, we will do the downsampling, and extract the feature map from the image. In the first step, we extract the feature by the squeeze and extract module. The first two SE modules will keep preventing vanishing gradient. This skip concat design will concat with the head part. In the head, we do the upsampling to find the object in the feature map, and we do the skip concat, to ensure the residual in the model does not vanish. In each upsampling, we will extract the BBox in the feature map to recover in the prediction part. Finally, after the upsampling and downsampling, we will get the four different BBox of the ground truth prediction, and we will return the predicted BBox, and calculate the GIOU to backpropagates the weight in the method.

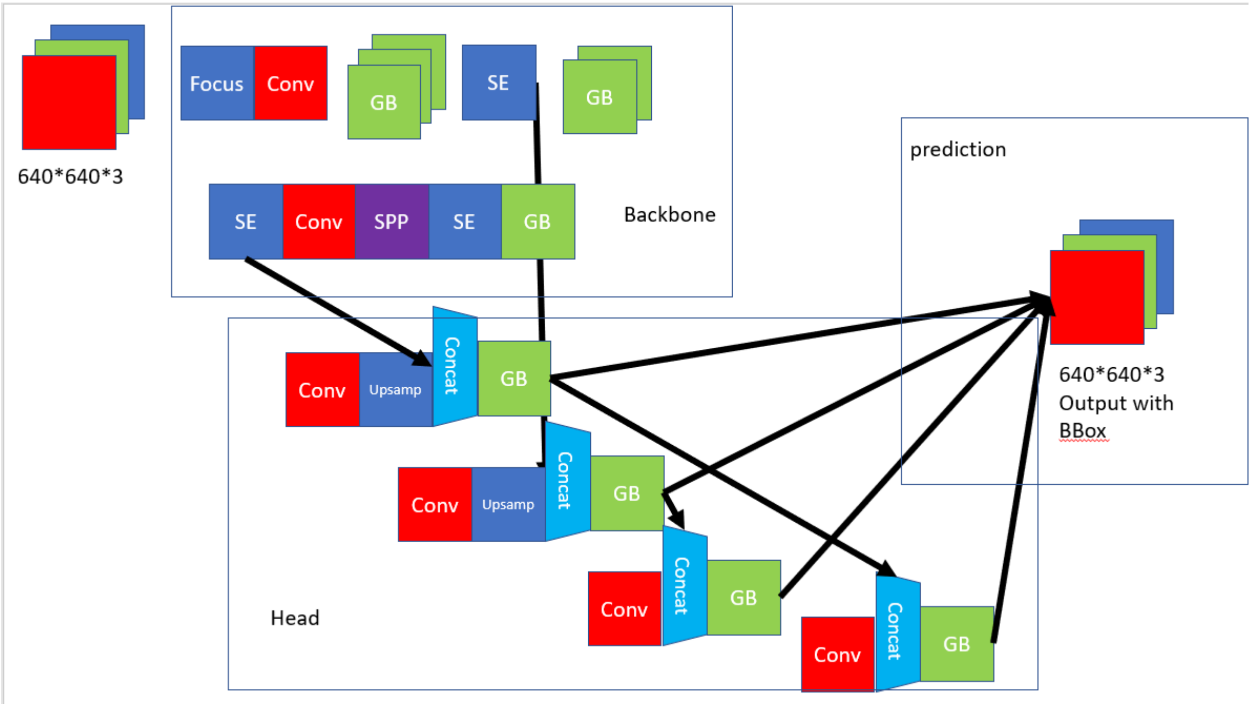

Figure 6 Detail network design

Export PyTorch Weight into the CoreML

In this paper, we use PyTorch and python to approach YOLO-Ghostnet. In recent years PyTorch well supports the deployment on IoT devices, with the pipeline to transfer the PyTorch model to ONNX, the PyTorch can easily deploy on the edge device.

The final step is to transfer the PyTorch model into the edge device. Since this project is trying to deploy models on the IOS device. We need to transform our model into CoreML. The pipeline to export the PyTorch model to CoreML needs to transfer to the ONNX file first. In this transformation, we need to encode the existing propagation function into the ONNX format. As a result, we can use this ONNX to turn it into the CoreML file and deploy the CoreML file on an edge device.

EXPERIMENTAL RESULTS AND ANALYSIS

We test the influence of different training improvement techniques on accuracy of the KITTI dataset, and then on the accuracy of the detector on KITTI weather synthesized dataset.

model information

Experimental Setup

In this paper we use the following data augmentation methods in the training process: mosaic augmentation, random flip, color jitter, random scale, random cropped. In road user classification experiments, the default hyper-parameters are as follows: the training epoch is 300; the batch size is 32, respectively; the decay learning rate scheduling strategy is adopted with an initial learning rate of 0.1; warm up epoch is 3; warm up momentum=0.8; warm up bias learning rate is 0.1; the momentum and weight decay are respectively set as 0.9 and 0.005. There are 7790 images in the training dataset. All of our weather image augmentation uses the same hyper-parameter as the default setting.

As Table 1 shows, we will use the YOLOv5 with the Ghostnet as the backbone. The FPS of the model is fast on the GPU device, and the total parameters are very small compared to the other architectures.

Table 1 the overall summary of the YOLO-Ghostnet property compared to state-of-the-art:

Method	Backbone	Image size	mAP	mAP50	Parameters	FPS
YOLOv5	Ghostnet	640	0.43	0.66	3.75M	111
YOLOv4	CSPDarknet	608	0.44	0.68	27M	62

EfficientDet-D1	Efficient-B1	640	0.40	0.59	12M	66

Recognition Performance

In order to verify the validity of the method in this paper, YOLO-Ghostnet is built based on the PyTorch library with python.
First, YOLO-Ghostnet is trained on the training set of the KITTI dataset, and the model is saved after training. Then the images of the test set are input into the best model which performs the best mAP during the training and validated by validation dataset. Then the images of the test set are input into the best model for road user detection and the classification results are output. confusion matrix between pedestrian, cyclist, vehicle. Finally, according to the classification results of 1197 test images, the recognition accuracy is calculated, and the confusion matrix of road users is down, as shown in the figure. The values on the diagonal of the confusion matrix represent the recognition accuracy of each category.

As Figure 7 and Table 2 show, the model is poor in the pedestrian and cyclist category. However, the most confusing case between these two categories is that the pedestrian will easily be confused with the cyclists. I believe this results in the shape. Generally, cyclists= bike+ human. Besides the bike's shape is quite similar compared to the vehicle, since it is smaller than the people. Therefore, the people can occlude the cycle image. The second challenge is that there are still some vehicles that will be determined as the background noise. I think this will be considered by the truncate or the occluded case that the shape of the vehicle looks like the random noises coming from the background. As a result, I think if we assert the VRUs group the precision is high enough.

Table 2 Precision, recall, and mAP50 and mAP without weather data augmentation

Class	Precision	Recall	mAP50	mAP
all	0.68	0.64	0.62	0.35
Car	0.62	0.77	0.70	0.48
Pedestrian	0.71	0.53	0.56	0.27
Cyclist	0.70	0.61	0.60	0.30

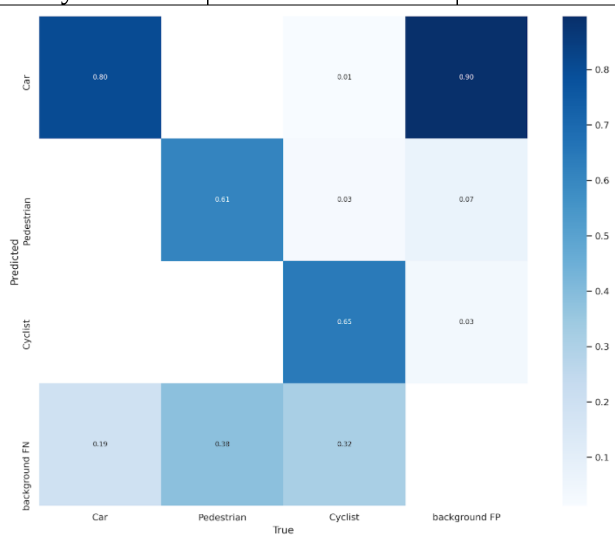

Figure 7 Confusion matrix of YOLO-Ghostnet without weather data augmentation

effects of weather image augmentation

In order to validate the effectiveness of data augmentation for the task of road user detection, in this paper, the training images in KITTI dataset with or without data augmentation are fed into the YOLO-Ghostnet for training, and the test images are used for validation. The accuracy curves of two methods on the validation set are shown in figure 7. The experimental results are shown in table 3. The mAP50 of YOLO-Ghostnet only around 62%, and the mAP is only 35% without data augmentation. On the other hand, with weather data augmentation, the mAP50 of YOLO-Ghostnet significantly advanced to 66% and the mAP came to 43%.

Table 3 Precision, recall, and mAP50 and mAP with weather data augmentation:

Class	Precision	Recall	mAP50	mAP
All	0.68	0.73	0.66	0.43
Car	0.66	0.82	0.70	0.53
Pedestrian	0.73	0.63	0.65	0.36
Cyclist	0.63	0.74	0.64	0.40

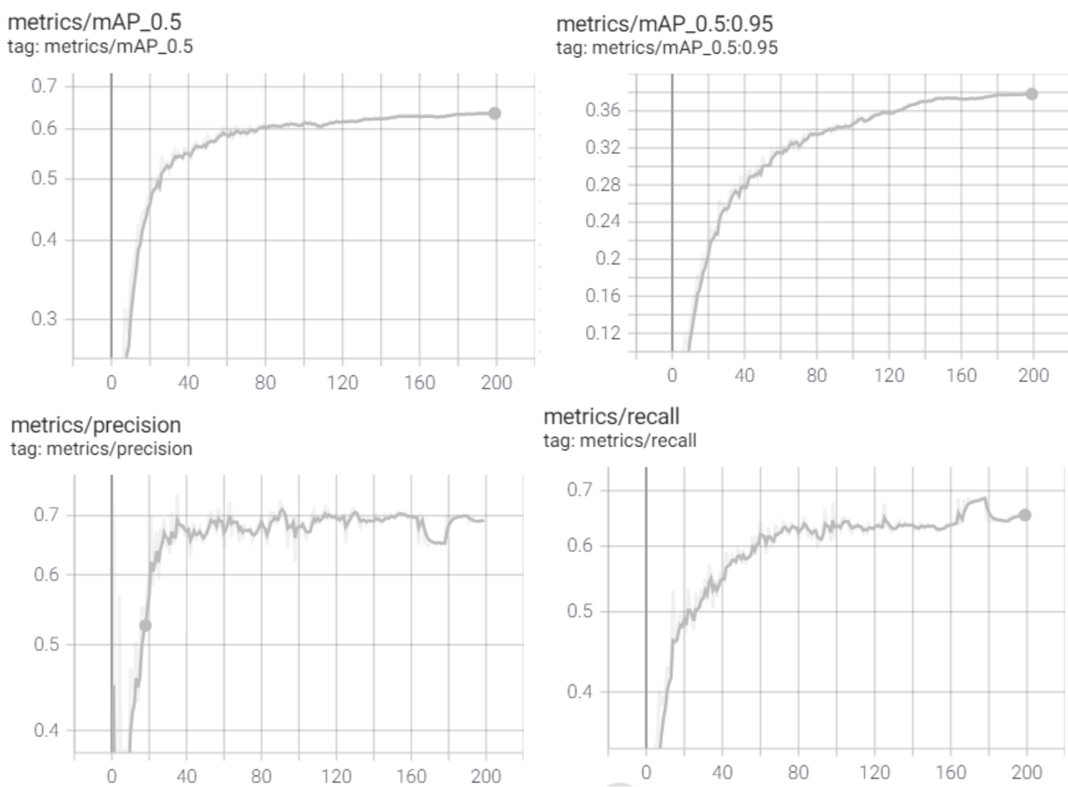

Figure 7 mAP50, mAP, precision, recall rate corresponds to epoch without augmentation

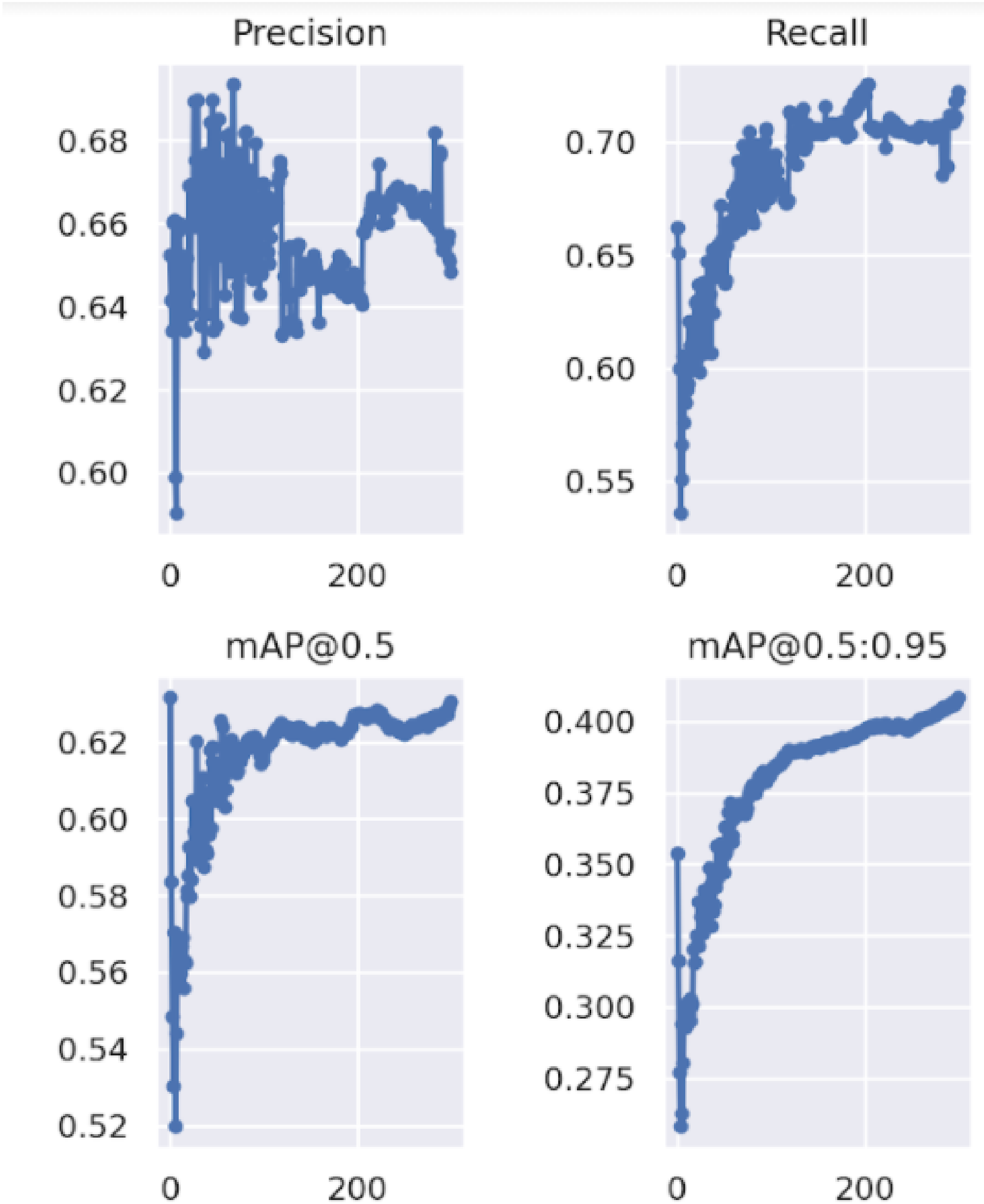

Figure 8 mAP 50, mAP, precision, recall rate corresponds to epoch with augmentation

Moreover, as figure 7 shows, it can also be seen from the curve that overfitting began to appear in YOLO-Ghostnet after about 80 training epochs, and the accuracy of the validation set no longer increased. Compared to figure 8, weather augmentation could significantly inhibit the overfitting phenomenon, greatly advance the recognition accuracy of road users' detection, and enhance the generalization and robustness of those models.

Export PyTorch Weight into the CoreML

In this paper, we use PyTorch and Python to approach YOLO-Ghostnet. First, in recent years PyTorch well supports the deployment on IoT devices, with the pipeline to transfer the PyTorch model to ONNX, the PyTorch can easily deploy on the edge device. As the figure 9 shows the road map to deploy the PyTorch model on IOS devices.

The final step is to transfer the PyTorch model into the edge device. Since this project is trying to deploy models on the IOT device. We need to transform the .pt file into CoreML. The pipeline to export the .pt model to CoreML needs to transfer to the ONNX file. In this transformation, we need to encode the existing propagation function into the ONNX format. As a result, we can use this ONNX to turn it into the CoreML file and deploy the CoreML file on an edge device.

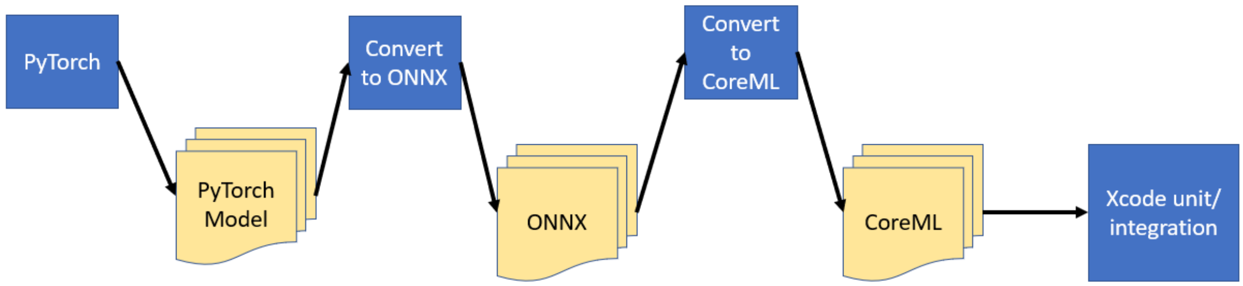

Figure 9 the roadmap to deploy the PyTorch model on IOS device.

Results and Discussion

The above section shows the accuracy and the performance of YOLO-Ghostnet on the KITTI weather dataset. In this paper, we discuss how to advance the reliability and availability of road user detection. The following is the observation and discussion about the test results.

General case:

As figure 9 shows, the common challenges still have space to improve. For instance, the small object overlaps and the occlude between in the scene still cannot be split detected by the model. Secondly, cyclists can easily confuse with the pedestrian because the difference between cyclist and pedestrian is bike or not. This pattern can be seen in the 4.2 section's confusion matrix. As we mention in the experiment setting, the total label of the pedestrian, vehicle, and cyclist is around 1:8:1, which is the serious label unbalance case. Therefore, adding more cyclists and pedestrians will be helpful to increase the accuracy.

Figure 10 detects the batch image of the test dataset.

Foggy Scenario:

In the foggy scenario, the edge of the object will be bluer than usual. This pattern shows in the detected image. As the figure 9 shows, the transportation signals are treated as the pedestrian. In order to improve the accuracy of foggy scenarios, there are two ways. First, increase more pedestrian cases to make the model more accurate. Second, increasing the deep learning neural network is also possible, but this will decrease the availability of the application.

Figure 11 Foggy weather image, we can see the miss label occur in this scenario.

Rainy scenario:

In rainy scenario, raindrops would become the noise which will make the BBox unable to be accurate. This pattern shows in the detected image. As the figure 11 shows, the transportation signals are treated as the pedestrian. In order to improve the accuracy of foggy scenarios, there are two ways. First, increase more pedestrian cases to make the model more accurate. Second, increasing the deep learning neural network is also possible, but this will decrease the availability of the application.

Figure 12 rainy weather image, as figure shows, the drop will distinguish as the pedestrians.

overexposed scenario:

In the overexposed scenario, it does not impact the accuracy of the road user detection, because it only changes the illumination of the pixel. Overexposed not dramatically change the object shape, so features are still the same in overexposed scenario.

Figure 13 overexposed image, we can see the accuracy is not impact in this scenario

CONCLUSION

This paper presents the design, development, deployment, and evaluation of a smart, efficient, and lightweight road user detection on edge device. In addition, this paper also provides the pipeline to do the weather condition augmentation which can enhance the model robustness against the weather noises. The deep learning models on the edge device are still in a primary stage. We conduct a thorough literature review on the state-of-the-art object detector, image augmentation method, and the benchmark of road user detection. The system processing pipeline and algorithms reduce the workload of the edge device. This model achieved 66% mAP with 112 FPS on the Tesla V100 environment and 27 FPS without GPU. Also, the weather image augmentation shows a significant improvement in the accuracy against weather noises.

To improve the performance of the current model, the label imbalance needs to be solved. Currently, the current label of the dataset is around 80% is the vehicle, only 20% is for pedestrians and cyclists. Therefore, increasing the VRUs dataset would be a proper way to increase the accuracy. Testing on the practice environment on the other edge device, such as surveillance camera can help validate the accuracy and performance of this system. In the future, we can design more weather data augmentation method to improve the performance of road user detection on the edge device.

AUTHOR CONTRIBUTION STATEMENT

The authors confirm contribution to the paper as follows: study conception and system design: Hung Lo, Ruimin Ke, Yinhai Wang; software and hardware programming: Hung Lo; data collection: Hung Lo; analysis and interpretation of results: Hung Lo; draft manuscript preparation: Hung Lo, Ruimin Ke, Yinhai Wang. All authors reviewed the results and approved the final version of the manuscript**.**

REFERENCES

(1) “Global status report on road safety 2015,” World Health Organization (WHO), Tech. Rep., 2015. [Online]. Available: http://www.who.int/violence injury prevention/road safety status/2015/en/

(2) N. Sajn. “General safety of vehicles and protection of vulnerable road users”. PE 593.542, Nov 2016.

(3) Andreas Geiger ,Philip Lenz ,Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. CVPR, 2012.

(4) Glenn Jocher. Mosaic data augmentation. ultralytics/yolov3. Apr 3, 2019.

(5) Terrance DeVries, Graham W Taylor. Improved regularization of convolutional neural networks with CutOut. arXiv:1708.04552, 15 Aug, 2017.

(6) Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. DropOut: A simple way to prevent neural networks from overfitting. The journal of machine learning research, 13 Nov, 2014.

(7) Golnaz Ghiasi, Tsung-Yi Lin, Quoc V Le. DropBlock: A regularization method for convolutional networks. arXiv:1810.12890, 30 Oct, 2018.

(8) Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, David Lopez-Paz. MixUp: Beyond empirical risk minimization. arXiv:1710.09412, 25 Oct, 2017.

(9) Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. arXiv:1905.04899, 13 May, 2019.

(10) Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 4 Sep, 2014.

(11) Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. arXiv:1512.03385, 10 Dec, 2015.

(12) Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, Kaiming He. Aggregated residual transformations for deep neural networks. arXiv:1611.05431, 16 Nov, 2016.

(13) Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. arXiv:1608.06993, 25 Aug, 2016.

(14) Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size. arXiv:1602.07360, 24 Feb, 2016.

(15) Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. arXiv:1704.04861. 17 Apr, 2017.

(16) Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946, 28 May, 2019.

(17) Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, Jian Sun. ShuffleNetV2: Practical guidelines for efficient cnn 15 architecture design. arXiv:1807.11164, 30 Jul, 2018.

(18) Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, Dahua Lin. Libra R-CNN: Towards balanced learning for object detection. arXiv:1904.02701, 4 Apr, 2019.

(19) Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640. 8 Jun, 2015.

(20) Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. arXiv:1512.02325. 8 Dec 2015.

(21) Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. arXiv:1804.02767. 8 Apr, 2018.

(22) Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv:1406.4729. 18 Jun, 2014.

(23) Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv:1606.00915. 2 Jun, 2016.

(24) Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. arXiv:1711.07767. 21 Nov, 2017.

(25) S. Lazebnik, C. Schmid and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, pp. 2169-2178, doi: 10.1109/CVPR.2006.68.

(26) Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of International Conference on Machine Learning (ICML), pages 807–814, 2010.

(27) Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of International Conference on Machine Learning (ICML), volume 30, page 3.

(28) Günter Klambauer, Thomas Unterthiner, Andreas Mayr, Sepp Hochreiter. Self-Normalizing Neural Networks. arXiv:1706.02515. 8 Jun, 2017

(29) Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. arXiv:1710.05941, 16 Oct, 2017.

(30) Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. arXiv:1905.02244, 6 May, 2019.

(31) Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. UnitBox: An advanced object detection network. arXiv:1608.01471. 4 Aug, 2016.

(32) G. J. et al., “ultralytics/yolov5: v3.0,” Aug. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3983579

(33) Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. arXiv:1902.09630. 25 Feb, 2019.

(34) Kai Han,Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, Chang Xu. GhostNet: More Features from Cheap Operations. arXiv:1911.11907, 13 Mar, 2020.

(35) Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection arXiv:2004.10934, 23 Apr, 2020.

(36) Kaiming He; Jian Sun; Xiaoou Tang. Single image haze removal using dark channel prior. 10.1109/CVPR.2009.5206515. 18 Aug, 2009.