Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9310558, 11 pages
https://doi.org/10.1155/2021/9310558

Research Article

Joint Channel Pruning and Quantization-Based CNN Network Learning with Mobile Computing-Based Image Recognition

Huanyu Liu,[1] Qing Luo,[2,3] Mingmei Shao,[1] Jeng-Shyang Pan,[4] and Junbao Li[1]

[1]*School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin 150001, China*
[2]*College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China*
[3]*Shenyang Aircraft Design & Research Institute, Shenyang 110035, China*
[4]*College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China*

Correspondence should be addressed to Junbao Li; lijunbao@hit.edu.cn

Received 11 August 2021; Accepted 4 September 2021; Published 8 October 2021

Academic Editor: Chi-Hua Chen

The development of the Internet and communication technology has ushered in a new era of the Internet of Things (IoT). Moreover, with the rapid development of artificial intelligence, objects are endowed with intelligence, such as home automation and smart healthcare, which are typical applications of artificial intelligence technology in IoT. With the rise of convolutional neural network (CNN) in the field of computer vision, more and more practical applications need to deploy CNN on mobile devices. However, due to the large amount of CNN computing operations and the large number of parameters, it is difficult to deploy on ordinary edge devices. The neural network model compression method has become a popular technology to reduce the computational cost and has attracted more and more attention. We specifically design a small target detection network for hardware platforms with limited computing resources, use pruning and quantization methods to compress, and demonstrate in VOC dataset and RSOD dataset on the actual hardware platform. Experiments show that the proposed method can maintain a fairly accurate rate while greatly speeding up the inference speed.

1. Introduction

In recent years, with the rapid development of the mobile infrastructure of IoT and the increasing popularity of the application of IoT, the complexity and operability of various mobile applications have been continuously increasing, and the requirements for the intelligence of mobile applications are getting higher and higher. At the same time, artificial intelligence has been gradually applied to all aspects of IoT, such as home automation [1], smart healthcare [2], smart security [3], autopilot [4], and other fields.

In recent years, convolutional neural network (CNN) is regarded as one of the best techniques for understanding image content and has shown great performance in image classification [5], segmentation [6], and detection [7] tasks. The features of CNN such as local connection, weight sharing, and pooling operations can effectively reduce the complexity of the network and reduce the number of training parameters. Since the advent of AlexNet [8], most frontrunners in image processing competitions have adopted CNN-based methods.

However, current state-of-the-art CNN cannot adapt well to today's smart mobile devices. Especially, in tasks with higher complexity like target detection, computing power and on-device memory has become the two major bottlenecks for the CNN model deployment on the mobile side. Therefore, many recent research is aimed at reducing the model computing operations and model parameters with minimal accuracy losses. However, most model compression methods are performed on classification models. Common classification networks usually have small input resolution and simple model architecture, while detection network usually has a larger input resolution and more complicated architecture. Detection network may also include multiple inputs, as well as some postprocessing, which put forward higher requirements for the design of the network compression algorithm. Common compression methods include pruning [9], quantization [10], and distillation. Model pruning methods can usually be divided into two types: structured and unstructured. Unstructured pruning methods usually only work on specific hardware due to the sparsity on neuron level [11]. Existing target detection datasets have prominent target features and clear details. However, in practical applications, due to the high shooting height, the target size is too small compared to the image, and the target features are incomplete, and the target occurs a certain degree of deformation affected by the shooting angle; the relative motion between the target and the drone causes the target background to change significantly, etc., making the task of drone image target detection challenging. In order to meet the above needs and solve the technical difficulties of UAV target detection, in recent years, researchers have carried out a series of related research. Traditional UAV image target detection methods include frame difference method, background subtraction method, sliding window-based feature extraction algorithm [12], mean-shift algorithm, and edge detection algorithm, and recently, deep learning methods are proposed, for examples, fast deep neural networks with knowledge guided training and predicted regions of interests [13], small unmanned aerial vehicle [14], object-based hierarchical change detection [15], application of unmanned aerial vehicles [16], and real-time implementation using GPUs [17]. Traditional sliding window-based features are usually artificially designed Histogram of Oriented gradient features (HoG) [18], Scale-invariant feature transform features (SIFT) [19], Haar-like wavelet features [20], etc. This method for implementing feature has high computational complexity and cannot be detected in real time. In 2012, S. Janand others combined the multiscale mean-shift algorithm with the edge information of the target to solve the saliency object detection of images taken by the drone. In addition, there are various real-time moving object detections in aerial surveillance algorithms such as local null space pursuit [21]. These methods are slow in calculation and weak in robustness and cannot meet the actual application requirements of real-time detection. In 2016, researchers used neural networks to detect rice field weeds from aerial images of drones flying at a height of 50 meters [22]. Zhang and others searched regions of interest (ROIs) based on the characteristics of adjacent parallel lines [23] and determined the final airport area through transfer learning on the AlexNet network. Xiao and others used the new GoogleNet-LF model to extract multiscale deep integrated feature combination SVM for detection and recognition [24]. In 2018, Wang and others used CNN target detectors with RetinaNet [25] as the backbone network to perform pedestrian detection on the Stanford drone data set [26] and verified the targets of the CNN-based target detector on the drone image advantages in detection. In the highly complex tasks such as target detection, computing power and memory space become the two bottlenecks in the deployment of the model on the mobile terminal. Therefore, many recent research results are aimed at reducing the amount of model calculation and model parameters and affecting the model accuracy as low as possible. However, most of the model compression methods are based on the classification model. Common classification models usually have small input resolution and simple model structure, while target detection models usually have larger input resolution and more complex network structure. The target detection model may also include multiple inputs and some postprocessing processes, which put forward higher requirements for the design of model compression algorithm. Xiao et al. [27] introduced comparative learning into model distillation. The main idea of contrastive learning is to learn a representation. In some metric spaces, the positive sample pairs are close together, and the negative sample pairs are separated as much as possible. The author models the contrast loss between each layer of teacher network and student network, so that the teacher network and student network are not only consistent in the prediction probability of output but also similar in internal representation. The introduction of comparative learning further improves the effect of model distillation. Tian et al. [28] proposed the variational student method. It combines the compressibility of knowledge distillation framework and the sparsity and guiding ability of variational reasoning technology. The author creates a sparse student network. The sparsity of the network is introduced by optimizing the variational parameters of the loss function based on variational reasoning. This is achieved by using the knowledge learned from the teacher network. The author considers Bayesian neural network in a general knowledge distillation framework, in which students adopt a variational penalty least squares objective function. Model compression is a very important part of neural network model training and deployment. Most neural network deployment tools integrate the model compression technology of quantification after training. However, a widely applicable model compression tool is still very rare. Distiller [29] is a toolkit that supports multiple model compression methods. It supports structural and unstructured pruning, quantification after and during training, and knowledge distillation methods. However, these functions can only be used in the classification model. For the more complex target detection model, it only supports unstructured pruning. And it does not provide the interface related to embedded deployment, which hinders the deployment of the final model. The situation of paddleslim is similar to that of distiller, and it also lacks support for target detection network. Recently, TensorFlow [30] supports unstructured model pruning and quantization algorithms during training. Relying on TensorFlow Lite, TensorFlow's model can be easily deployed in the embedded segment. The latest version of PyTorch [31] also supports some basic quantitative tools but less support for the deployment of embedded platforms.

As the previous works, the popular object detection algorithms are usually based on convolutional neural networks, which are difficult to be deployed on platforms with limited computational resources such as embedded platforms due to the limitation of computational amount and large number of parameters. However, with the increasing demand for object detection task in the industry, the detection speed of the common object detection model is not up to the standard. Although the computing power of embedded platform is developing rapidly in recent years, the large amount of computation and the large number of parameters of object detection algorithm are still the main factors that restrict the practical application of object detection algorithm. Therefore, it is of great significance to study the model compression algorithm of object detection algorithm for the application of object detection algorithm in industry. In this paper, an object detection network suitable for embedded platform deployment is optimized and designed through in-depth study of object detection algorithm. On the basis of optimized network, the object detection network compression algorithm based on pruning and quantization is proposed, which can greatly compress the model size and improve the reasoning speed of the model while maintaining the accuracy of the model. The structured pruning method usually prunes on the channel level, which can be accelerated on common hardware. Quantization methods can generally be divided into posttraining quantization and quantization during training. Quantization converts floating-point numbers to integers symmetrically or asymmetrically according to the quantization bit depth, zero point, and scale parameters. Posttraining quantization usually calculates the quantization parameters of each layer on the calibration set.

The channel pruning algorithm can be divided into three steps: sparse training, channel parameter pruning, and constant channel elimination. The purpose of sparsity training is to make the parameters of the model produce structured sparsity, which is convenient for subsequent screening of channels that have little impact on the accuracy of the network. Structured sparsity means that sparsity occurs not only on a single parameter but also on the whole channel. The second step is to cut out the channels that have little impact on the accuracy. Generally, the closer the parameter to 0, the less the impact on the accuracy of the network. When cutting off the channel, the influence of residual structure and other structures should be considered to prevent the structure of the network from being damaged. Finally, constant channel elimination can eliminate the influence of constant channel and restore the accuracy of the model. In the process of model quantization, we first need to insert a pseudo quantization module into the model. The purpose of this step is to reduce the quantization loss caused by the model quantization process, and the quantization parameters can also be introduced into the calculation diagram. Then, quantitative training, in which the weight and quantitative parameters of the learning model, will be optimized at the same time. Finally, the trained model is transformed into a quantized model. In this step, the learned quantization parameters are applied to the model weight, and the model weight is transformed into a low-precision representation. A pseudo quantization operation corresponds to a tensor, which corresponds to the quantization parameters of the tensor. The pseudo quantization operation has two functions: one is to simulate the quantization loss, and the other is to introduce the quantization parameters into the calculation diagram, so that the back propagation can optimize the quantization parameters. Therefore, we need to add pseudo quantization operation after all tensors to let them learn the quantization parameters of the tensor. These tensors include each operation weight tensor, input tensor, and output tensor. In convolutional networks, most operations can be reduced to three categories: product, addition, and connection. These three operations will change the value domain distribution of the tensor and affect the quantization parameters of the tensor. In addition, the nonlinear activation function will also affect the value range distribution of the tensor. However, the activation functions used in this paper are ReLU activation functions, which can be regarded as piecewise linear. One difficulty of pseudo quantization is how to reverse broadcast the quantized tensor. There are many undefined points in the function curve of pseudo quantization operation. In this paper, we choose to estimate the derivative directly.

In this paper, we apply quantization during training, aka quantization aware training (QAT), that learns and observes weights and quantization parameters at the same time, which can achieve accuracy closer to the floating-point model. And then, there are different computation graphs during training and inference. In training graph, we simulate quantized convolution operation through the fake-quantization module and fuse the BN layer parameters into the convolution weights to better fit the actual value. The distillation method achieves model compression by transferring the knowledge of the trained large model to a smaller model. So, the higher performance is achieved for this method. Common compression methods usually only demonstrate on large models. However, in order to be able to deploy the CNN on platforms with limited computing resources, choose a small network designed specifically for mobile devices, and then, use the compression technique to achieve better results. We use channel pruning and quantization methods to compress the small detection network and test inference time on actual hardware.

2. Proposed Method

In this section, we discuss the details of the methods and principles used to compress the object detection network. First, we describe the architecture of the proposed detection network and then discuss the algorithm of pruning and quantization methods, respectively.

2.1. Network Architecture. We optimized the recognition network as follows. The target recognition network based on deep learning with good generalization is used to complete the target recognition of airports, bridges, and ports under low resolution. The following introduces the identification of the backbone structure of the network, candidate frame generation in the network, calculation of the network loss function, and training strategies.

First, the basic structure of the remote sensing image target recognition network under low resolution is introduced. The basic structure of the remote sensing target recognition network used in this subject is shown in Figure 1; the basic network structure of the VGG16 is continued on the network backbone structure. The first five layers still use the five convolutional layers of the VGG16 network, discarding the fully connected layers of the sixth and seventh layers of the VGG16 network, while using the astrous

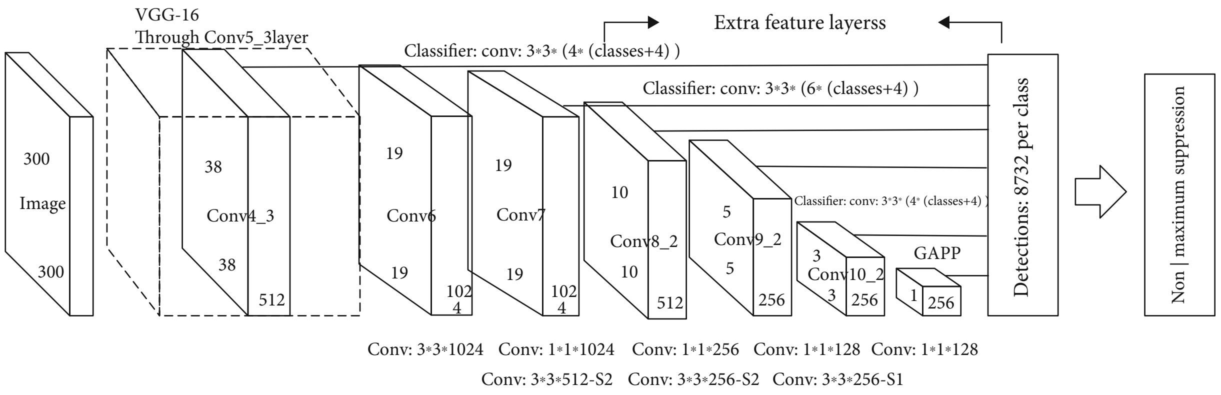
VGG-16
Through Conv5_3layer
Extra feature layerss
Classifier: conv: 3*3* (4* (classes+4))
Classifier: conv: 3*3* (6* (classes+4))
Classifier: conv: 3*3* (4* (classes+4))
300
Image
300
38
Conv4_3
38
512
19
Conv6
19
1024
19
Conv7
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
GAPP
1
256
Detections: 8732 per class
Non | maximum suppression
Conv: 3*3*1024 Conv: 1*1*1024 Conv: 1*1*256 Conv: 1*1*128 Conv: 1*1*128
Conv: 3*3*512-S2 Conv: 3*3*256-S2 Conv: 3*3*256-S1

Figure 1: Basic structure of low-resolution remote sensing image recognition network.

algorithm, using the convolutional method to construct two new convolution floor.

The conventional pooling layer in a deep neural network causes a decrease in resolution while increasing the receptive field, and the decrease in resolution loses some feature information. The advantage of this hole convolution is to avoid the decrease in resolution caused by pooling. The comparison between hole convolution and ordinary convolution is shown in Figure 2. It can be seen from the figure that under the same calculation parameters, a larger receptive field can be obtained by using hole convolution instead of ordinary convolution.

After the newly added sixth and seventh convolutional layers, three more convolutional layers (conv8, conv9, and conv10) are added, and layer is added to the network at the end to convert the output feature MAP of the previous layer into a one-dimensional vector. For the remote sensing targets studied in this subject, there is a large intraclass gap for the same type of target, and there is still a problem of scale gap for the same type of target. Therefore, multiscale recognition is particularly important. Considering the scale change of the target object, the network outputs feature MAPs of different scales at different layers and sends them to the detector to predict the degree of confidence and position coordinate offset of each category. As shown in Figure 3, the front-most feature MAP is output after the Conv4_3 layer. The feature MAPs of the first few layers in the network describe the shallower features in the input image, and their receptive fields are relatively small. In contrast, the deeper feature MAPs are responsible for describing the more advanced composite features. Their lower-level feature MAPs of receptive fields are larger, and it also has stronger advanced semantic information. At the end of the network, in order to avoid the result that the same target is detected by the multilayer feature detector at the same time, a nonmaximum suppression process is added, as shown in Figure 3. From this, the final test result is obtained. The network backbone structure does not use a fully connected layer. On the one hand, the output of each layer can only feel the characteristics of the area near the target, not the global information. On the other hand, it also reduces the number of computing parameters in the network.

The architecture of proposed object detection network is shown in Figure 4. We choose MobileNetv2 as our feature extractor. MobileNetv2 is a small network designed specifically for mobile devices, which maintains high accuracy while having a small amount of operations. MobileNetv2 greatly reduces computing operations through the depthwise convolutions and inverse residual structure and is more friendly to CPU. It achieved 72.0% Top-1 accuracy on the ImageNet dataset.

The postprocessing of the network is mainly composed of decoding and nonmaximum suppression (NMS). Decoding converts the outputs of the convolutional layer into predictive value of the bounding boxes. The shape of the tensor is $B \times H \times W \times A \times (C+5)$, where B is the batch size, H and W are the height and width of the feature MAP, A is the number of anchors, and C is the number of categories. The last 5 values of the last dimension are the offsets from the center of the grid point and one foreground probability, denoted as$(\hat{x}_{\min}, \hat{y}_{\min}, \hat{x}_{\max}, \hat{y}_{\max}, p_F)$. Let the grid points of the feature MAP be$P_{i,j} = (x_i, y_j) = (i, j) \quad 0 \le i \le W, 0 \le j \le H$. The relationship between the decoded coordinates and the outputs of the convolution layer will be:

$$\begin{aligned}(x_{\min}, y_{\min}) &= P + \exp\left((\hat{x}_{\min}, \hat{y}_{\min})\right) + 0.5,\\ (x_{\max}, y_{\max}) &= P + \exp\left((\hat{x}_{\max}, \hat{y}_{\max})\right) + 0.5.\end{aligned} \tag{1}$$

$x_{\min}, y_{\min}$ are the elements of the images, and P is the grid points of the feature MAP. We apply NMS on the decoded outputs of detection heads to get the finally bounding boxes. In detection head, we use BCE loss and GIOU loss [22] for classification and box regression.

2.2. Pruning. The channel pruning algorithm can be divided into three steps: sparse training, pruning, and fine-tuning, which is shown in Figure 5. The pruning process can be carried out once or multiple times.

For sparse training, in the training process, L1 norm penalty is applied to γ of the BN layer after the convolutional layer that needs to be sparse. The specific steps of sparse training are as follows. Schematic diagram of channel pruning is shown in Figure 6.

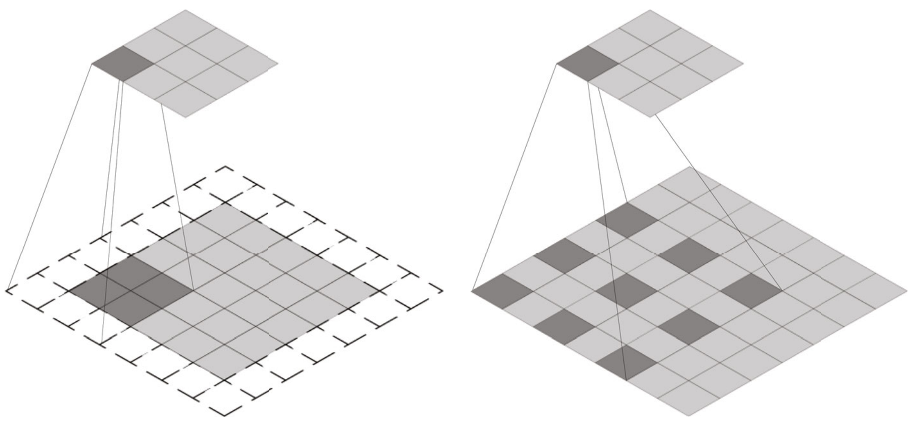

Figure 2: Comparison of ordinary convolution (a) and hole convolution (b).

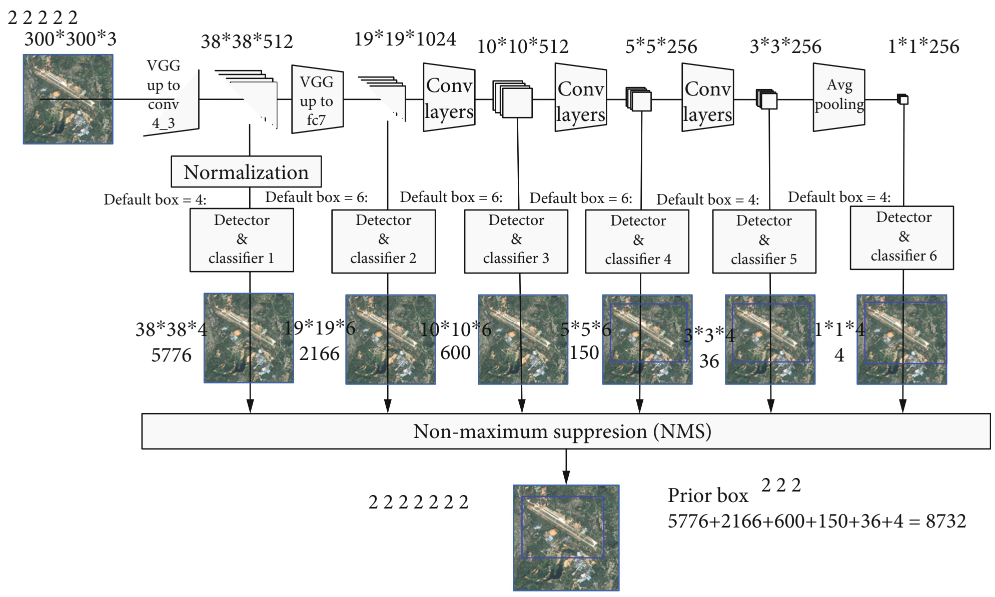

Figure 3: Multiscale detection in the network.

Let the shape of the current layer convolution kernel be $(N_{\text{in}}, N_{\text{out}}, K_h, K_w)$. Then, the convolution kernel after pruning is shown in Equation (2).

$$F'_x = F_x[M_{\text{in}}, M_x, :, :], \quad (2)$$

where $N_{\text{in}}, N_{\text{out}}, K_h, K_w$ represent the number of input channels, the number of output channels, and the convolution kernel height and width, respectively. $A[M_1, M_2]$ means filtering out and restructuring A by M_1, M_2dimension-wise. Colon indicates a mask of all ones.

Finally, set current layer's pruning mask M_x as the next layer's input pruning mask M_{in}.

For depth-wise convolution, the depth-wise convolution can be seen as a convolution with the same number of groups as the number of input channels and output channels. For this type of convolution, it is only necessary to remove the corresponding convolution kernel according to the input pruning mask M_{in}, then set the number of output channels and the number of groups to the new number of input channels.

2.3. Quantization. We have different computation graph during training and inference. The training graph and the inference graph are shown in Figure 7.

In training graph, we simulate quantized convolution operation through the fake-quantization module and fuse the BN layer parameters into the convolution weights to better fit the actual value. We obverse the range of values after each convolution and activation to calculate the quantization parameters. We use 8-bit signed integers to quantize the 32-bit floating-point weights and 8-bit unsigned integers for the activations. Both network input and convolution weights are quantized to 8-bit integers. However, since the biases of convolution are sensitive to errors, 32-bit unsigned integers are used for quantization.

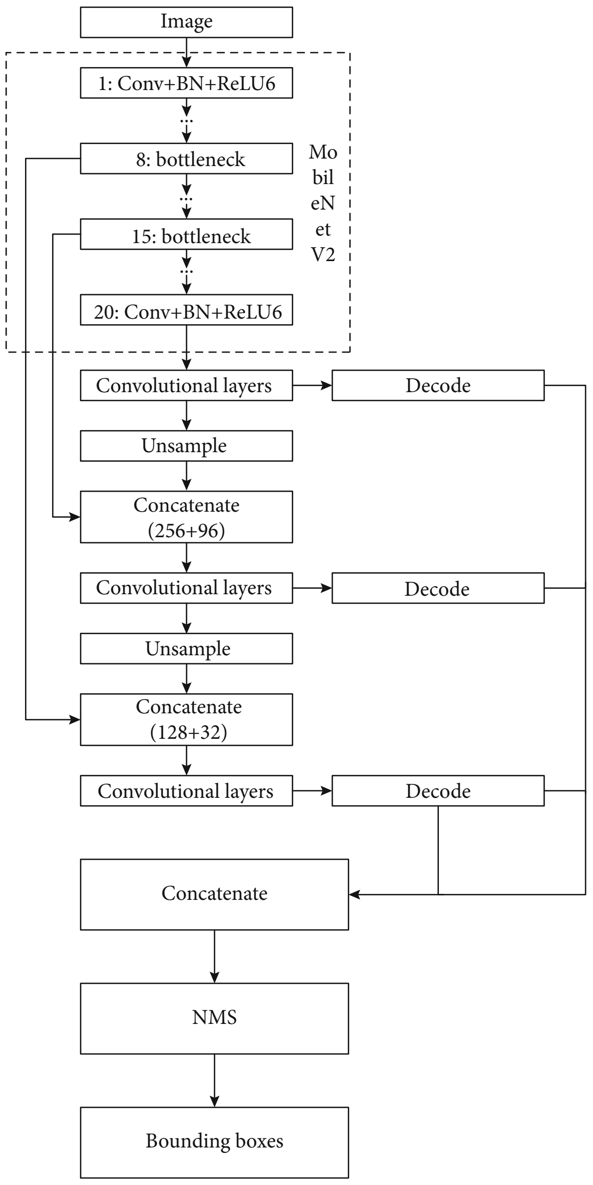

FIGURE 4: Architecture of the proposed object detection network.

The inference graph only contains quantized convolution operations, since batch-norm layers have been fused into pervious convolution weights.

3. Experiments

This section gives a comparison between proposed network and other state-of-art lightweight detection networks. Then, we show the acceleration of pruning and quantization.

We train our model on the VOC dataset. We set training epoch to 80 for all cases, using standard data augment methods, including random cropping and horizontal flipping, and additionally using mixup. Adopting Adam optimization algorithm, cosine annealing learning rate strategy with warming up, the initial learning rate is 4*e*-3, and the batch size is 16.

The target detection model compression algorithm is evaluated, so the main technical indicators of the algorithm also focus on the performance of the model before and after compression. The technical index shall reflect the compression effect of the compression algorithm on the model performance in terms of computation and memory use. On the server side, the original target detection model is obtained through the target detection training algorithm. Then, the original model is compressed by pruning and quantization through the target detection model compression algorithm. Then, with the help of model deployment tools. The original model and the compressed model are compared in multiple indexes, and then, a number of technical index results are obtained.

According to the needs of the subject, four technical indexes are proposed, such as compression ratio, model calculation, reasoning delay, and accuracy loss, as follows. (1) Compression ratio refers to the ratio of the original model storage space (in bytes) to the compressed model storage space. Compression ratio greater than 6; (2) model calculation quantity refers to the quantity of all multiply accumulate (MAC) operations during forward reasoning of target detection model. The calculation amount of the compressed model is less than 15 MACs; (3) reasoning delay refers to the time consumed from reading the input image to returning the result of forward reasoning in the process of predicting the image by the target detection model. The reasoning delay of the compressed model is less than 50 ms; (4) accuracy loss refers to the difference between the original model and the compressed model in evaluating the mean of average precision (MAP) index on the validation set data set and between the original model and the compressed model MAP. Accuracy loss is less than 5%.

As a result, proposed model got 78.46% of the test set MAP under the input image size of 512×512. FLOPs of our model is 4.25G MACs, and the number of parameters is 6.775M. See Table 1 for comparison with other network in terms of accuracy, MACs, and number of parameters.

The materials and methods section should contain sufficient detail so that all procedures can be repeated. It may be divided into headed subsections if several methods are described.

We have evaluated the versions of Yolo algorithms, and the most advanced Yolo algorithms are YOLOv3 and YOLOv4. YOLOv3 divides the input image into multiple grids on a two-dimensional plane. If the IOU between the detection frame and the predefined anchor frame of a detection target is greater than the threshold, the feature map output vector is responsible for predicting the target. YOLOv3 clusters the detection frames on the data set and divides them into 9 groups according to IOU distance. Using the predefined anchor box, YOLOv3 can assign different targets to different detection heads according to scale and aspect ratio. YOLOv3 uses darknet-53 as a feature extractor. Darknet-53 has the same amount of computation as resnet-50, but its speed and performance exceed resnet-50. YOLOv3 also introduces the idea of feature pyramid network (FPN). After darknet-53, the feature maps of different scales are fused and divided into three detection heads to predict large-, medium-, and small-scale targets, respectively.

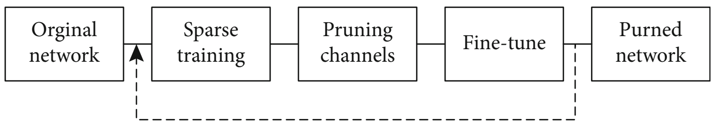

Figure 5: Flow chart of channel pruning.

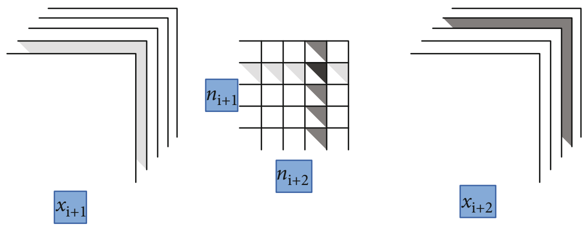

Figure 6: Pruning strategy in general situation.

All model in Table 1 was evaluated on the VOC 2007 test set at the input image size of 512×512. By comparison, the proposed model is close to the YOLOv3 in accuracy, but both MACs and the number of parameters are reduced a lot. Compared with Tiny YOLOv3 and YOLO Nano, we still have great advantages in accuracy and MACs.

The hyperparameter settings of sparse training are the same as previous. In sparse training, we set the sparse rate to 0.01 and reached 75.65% MAP on VOC 2007 test set. After pruning 40% of channels and fine-tuning for 20 epochs, it finally reached 75.44% test set MAP, which decreased by 3.0% compared with the unpruned model. MACs were reduced to 2.606G, and the number of parameters was reduced to 2.31M. Compared with the unpruned model, it was reduced by 38.6% and 65.9%, respectively. The number of channels of each layer before and after pruning is shown in Figure 8. See Table 2 for detailed comparison.

We then apply quantization on 40% pruned model. Training settings are the same as above. We choose 8-bit quantization, freezing BN layer parameters after 10 epochs, and freezing quantization parameters after 15 epochs. The quantized model achieved 76.74% MAP on test set, a decrease of 1.7% from the original model. We benchmark all models on E5-2630 v4 CPU, and the network input is 512×512. The benchmark is performed on the first 100 test set images of VOC dataset. See Table 3 for details.

After pruning and quantization on original model, we achieved 4.76 times acceleration on CPU, and there is only a small loss of accuracy.

We also experiment on RSOD dataset [32]. The RSOD dataset is an open dataset for object detection in remote sensing images. The dataset includes aircraft, oil tank, playground, and overpass. The image sizes range from 500×500 to 2000×1000. We randomly select 80% of the original images as the training set and 20% as the test set. As a result, the training and test sets include 752 images and 187 images, respectively.

During training, we randomly crop a 512×512 patch on original images and then scale it to the size of 320~608. We apply similar augmentation as before but add horizontal flipping. We pad the test images to nearest multiple of 32 and evaluate on original size. The model is trained by 20k

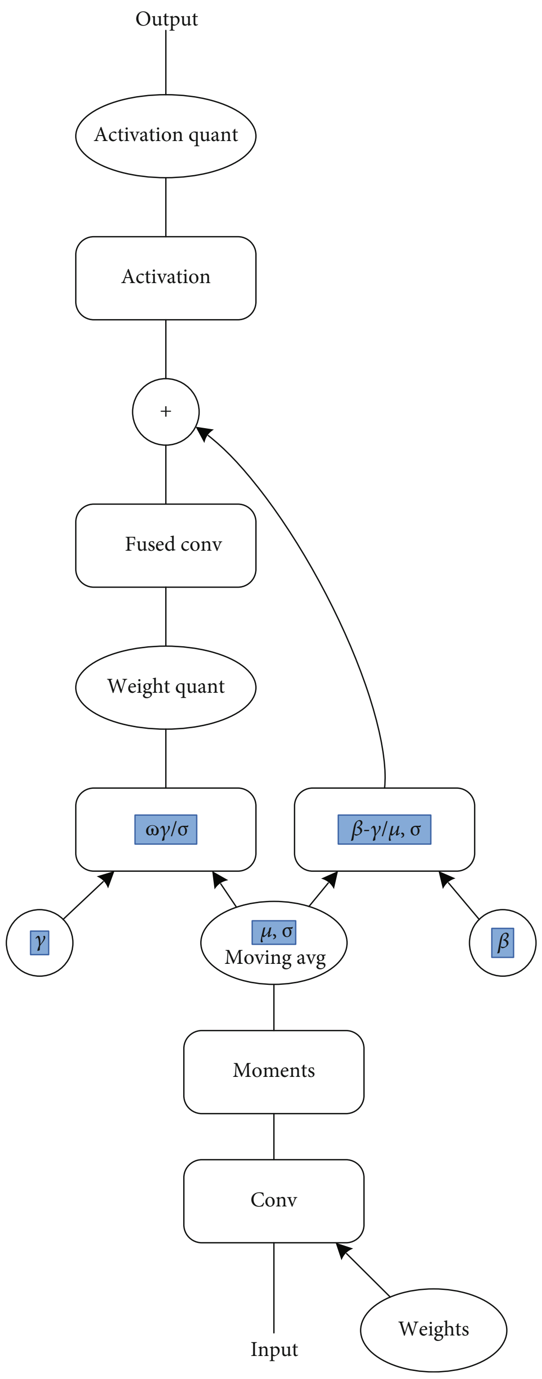

Figure 7: Computation graph of quantization-aware training.

iterations in total, and other training hyperparameter settings are as same as before.

In sparse training, we set sparse ratio to 0.1 and round remaining number of channels to multiple of 8. To avoid

Table 1: Comparison of the accuracy, MACs, and number of parameters.

Name	MAP	MACs	Params
YOLOv3 [20]	79.2%	49.8G	59.25M
Tiny YOLOv3 [20]	61.3%	8.36G	8.5M
YOLO Nano [26]	71.7%	6.92G	1.1M
Ours	78.46%	4.25G	6.775M

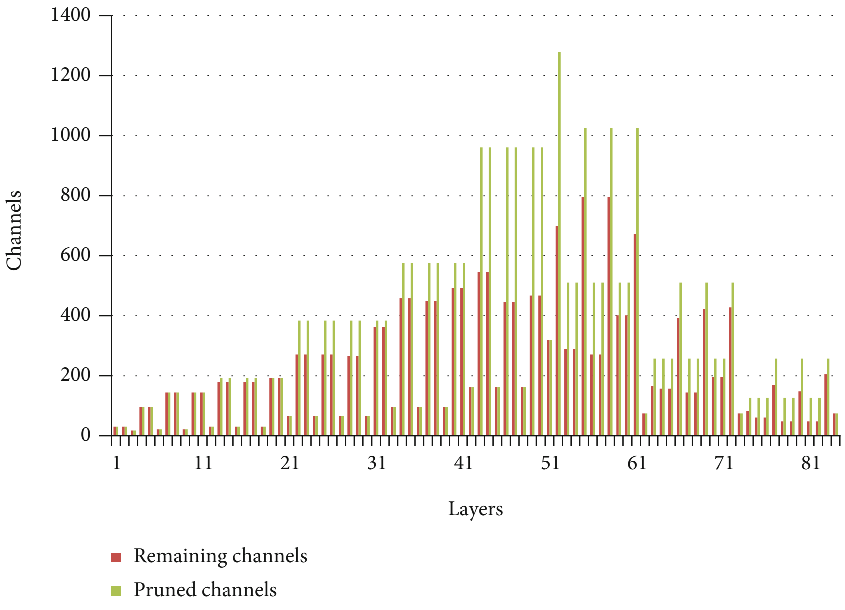

Figure 8: Comparison of the number of channels per layer before and after pruning on VOC dataset.

Table 2: Comparison between original and pruned model on VOC dataset.

Name	MAP	MACs	Params
Original	78.46%	4.25G	6.775M
30% pruned	76.30%	3.195G	3.463M
40% pruned	75.44%	2.606G	2.308M

Table 3: Benchmark for original, pruned, and quantized model on VOC dataset.

Name	MAP	MACs	Params	Inference time	Speedup
Original	78.46%	4.25G	6.775M	126.6 ms	1
40% pruned	75.44%	2.606G	2.308M	102.7 ms	1.23
40% pruned + quantized	76.74%	—	—	26.6 ms	4.76

pruning out all channels, we set the minimum number of channels to 16.

As showed in Table 4, our model reached 92.59% MAP on test set. We found that 85% of pruning on model greatly reduce FLOPs to 1.335G MACs, and the number of parameters is 180.5K, nearly 37x smaller than unpruned model with trivial accuracy drop.

We found that if pruning ratio higher than 85%, MAP drops sharply. The pruning limit is around 0.876 in this case, perhaps that is why.

For comparison, we also draw a figure shown the number of channels with and without pruning (see Figure 9). Note that the layers as main stem of residual blocks make up most of FLOPs.

Finally, we quantize the 85% pruned model with QAT. We benchmark 100 images on CPU with 512×512 input. Table 5 shows comparison between original, pruned, and quantized model. Our final model is 6.31 times faster than original one. The size of final model is only 253 KB.

Table 4: Comparison between original and pruned model on RSOD dataset.

Name	MAP	MACs	Params
Original	92.59%	4.248G	6.689M
30% pruned	91.76%	3.274G	3.506M
60% pruned	91.38%	2.334G	1.085G
80% pruned	90.78%	1.650G	326.3K
85% pruned	90.81%	1.335G	180K
90% pruned	63.53%	1.113G	84.9K

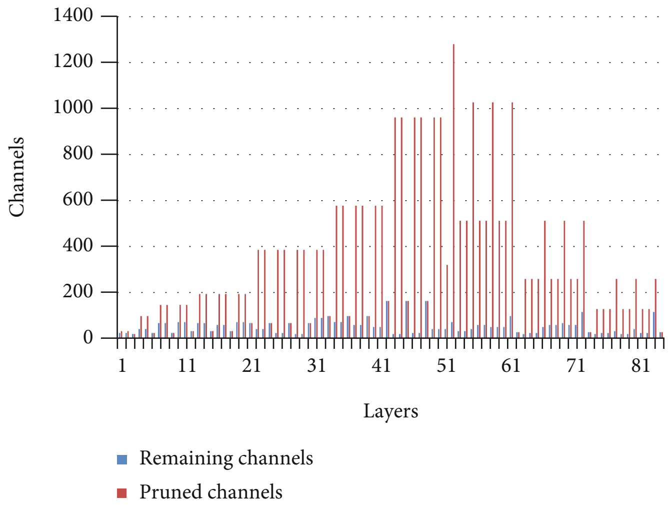

Figure 9: Comparison of the number of channels per layer before and after pruning on RSOD dataset.

Table 5: Benchmark for original, pruned, and quantized model on RSOD dataset.

Name	MAP	MACs	Params	Inference time	Speedup
Original	92.59%	4.25G	6.689M	131.3 ms	1
85% pruned	90.81%	1.335G	180.5M	180.5 ms	2.25
85% pruned + quantized	91.15%	—	—	20.8 ms	6.31

In the future practical application, the compressed algorithm needs to be deployed to the embedded platform for the convolutional neural network target detection system on the embedded platform. Its core technology is the target detection algorithm and network model compression algorithm. It is required to realize target detection based on convolutional neural network at the embedded end. Control the calculation amount of the model below 25 FLOPs. Because the task requires that the target detection algorithm needs to be deployed on the embedded end, it is necessary to select a target detection algorithm that can balance speed and accuracy and is easy to deploy at the same time. Therefore, in future practical applications, attention should be paid to the power consumption of the platform required in practical applications.

4. Conclusions

We propose a lightweight object detection model for hardware platforms with limited computing resources. We achieved this by designing a small network, channel pruning, and quantization. After pruning and quantization, we benchmark proposed model on VOC dataset and RSOD dataset, which achieved 4.76 and 6.31 times acceleration on common CPU with minor accuracy loss. In the future work, the prospects of work and research are listed as follows. In terms of channel pruning, pruning methods can be designed for special network structures such as residual structure, packet convolution, deep separable convolution, and Se module, including sparsity method and pruning channel selection strategy, so as to support target detection networks with more updated structures and achieve higher compression ratio. In the combination of compression methods, more model compression methods can be added, such as distillation and layer pruning. Combining more compression methods can further improve the compression ratio of the compression algorithm.

Data Availability

We have not used specific data from other sources for the simulations of the results. The two popular datasets in this paper, VOC dataset and RSOD dataset, are free download with the website: https://pjreddie.com/projects/pascal-voc-dataset-mirror/ and https://github.com/RSIA-LIESMARS-WHU/RSOD-Dataset-.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by stable support project funding Science and Technology on Near-Surface Detection Laboratory with grand no. TCGZ2020C005, the Science and Technology Foundation of National Defense Key Laboratory of Science and Technology on Parallel and Distributed Processing Laboratory (PDL) under Grant No. 6142110180406, the Aeronautical Science Foundation of China under Grant Nos. 2019ZC077006 and 2019-JCJQ-ZD-080-02-00, and the Foundation of Science and Technology on Electro-Optical Information Security Control Laboratory China under Grant No. 6142107200209.

References

[1] G. Evans, "Solving home automation problems using artificial intelligence techniques," *IEEE Transactions on Consumer Electronics*, vol. 37, no. 3, pp. 395–400, 1991.

[2] R. F. Mansour, A. El Amraoui, I. Nouaouri, V. G. Diaz, D. Gupta, and S. Kumar, "Artificial intelligence and Internet of Things enabled disease diagnosis model for smart healthcare systems," *IEEE Access*, vol. 9, pp. 45137–45146, 2021.

[3] Z. Lv, L. Qiao, A. K. Singh, and Q. Wang, "AI-empowered IoT security for smart cities," *ACM Transactions on Internet Technology*, vol. 21, no. 4, pp. 1–21, 2021.

[4] A. K. Yadav and P. Gaur, "AI-based adaptive control and design of autopilot system for nonlinear UAV," *Sadhana*, vol. 39, no. 4, pp. 765–783, 2014.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, 2017.

[9] Z. Liu, J. Li, Z. Shen, H. Gao, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2736–2744, 2017.

[10] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: a whitepaper," 2018, https://arxiv.org/abs/1806.08342.

[11] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *In NIPS*, pp. 1135–1143, 2015.

[12] Z. Chen, Y. Liu, and S. Zhai, *Performance Evaluation of Visual Object Detection for Moving Vehicle*, International CCF Conference on Artificial Intelligence, Springer Singapore, 2019.

[13] W. Cao, J. Yuan, Z. He, Z. Zhang, and Z. He, "Fast deep neural networks with knowledge guided training and predicted regions of interests for real-time video object detection," *IEEE Access*, vol. 6, pp. 8990–8999, 2018.

[14] M. F. B. Ramli, A. Legowo, and S. S. Shamsudin, "Object detection technique for small unmanned aerial vehicle," in *IOP Conference Series: Materials Science and Engineering. IOP Publishing*, 2017.

[15] R. Qin, "An object-based hierarchical method for change detection using unmanned aerial vehicle images," *Remote Sensing*, vol. 6, no. 9, pp. 7911–7932, 2014.

[16] S. Kamate and N. Yilmazer, "Application of object detection and tracking techniques for unmanned aerial vehicles," *Procedia Computer Science*, vol. 61, pp. 436–441, 2015.

[17] D. Jaiswal and P. Kumar, "Real-time implementation of moving object detection in UAV videos using GPUs," *Journal of Real-Time Image Processing*, vol. 17, no. 5, pp. 1301–1317, 2020.

[18] F. Han, Y. Shan, and R. Cekander, "A two-stage approach to people and vehicle detection with HOG-based SVM," *Performance Metrics for Intelligent Systems 2006 Workshop*, pp. 133–140, 2006.

[19] J. Y. Choi, K. S. Sung, and Y. K. Yang, "Multiple vehicles detection and tracking based on scale-invariant feature transform," in *2007 IEEE Intelligent Transportation Systems Conference*, pp. 528–533, IEEE, 2007.

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001.

[21] J. Sokalski, T. P. Breckon, and I. Cowling, "Automatic salient object detection in UAV imagery," in *Proc. 25th International Unmanned Air Vehicle Systems*, 2010.

[22] A. ElTantawy and M. S. Shehata, *Local Null Space Pursuit for Real-Time Moving Object Detection in Aerial Surveillance*, Signal, Image and Video Processing, 2019.

[23] G. V. Konoplich, E. O. Putin, and A. A. Filchenkov, "Application of deep learning to the problem of vehicle detection in UAV images," in *2016 XIX IEEE International Conference on Soft Computing and Measurements (SCM)*, pp. 4–6, IEEE, 2016.

[24] O. Barrero, D. Rojas, and C. Gonzalez, "Weed detection in rice fields using aerial images and neural networks," in *2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA)*, pp. 1–4, IEEE, 2016.

[25] X. Wang, P. Cheng, and X. Liu, "Fast and accurate, convolutional neural network based approach for object detection from UAV," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 3171–3175, IEEE, 2018.

[26] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, Venice, Italy, October 2017.

[27] Z. Xiao, Q. Liu, G. Tang, and X. Zhai, "Elliptic Fourier transformation-based histograms of oriented gradients for

rotationally invariant object detection in remote-sensing images," *International Journal of Remote Sensing*, vol. 36, no. 2, pp. 618–644, 2015.

[28] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," 2019, https://arxiv.org/abs/1910.10699.

[29] N. Zmora, G. Jacob, and L. Zlotnik, "Neural network distiller: a python package for DNN compression research," 2019, https://arxiv.org/abs/1910.12232.

[30] M. Abadi, A. Agarwal, and P. Barham, "TensorFlow: large-scale machine learning on heterogeneous distributed systems," 2016, https://arxiv.org/abs/1603.04467.

[31] A. Paszke, S. Gross, and F. Massa, "Pytorch: an imperative style, high-performance deep learning library," 2019, https://arxiv.org/abs/1912.01703.

[32] A. Robicquet, A. Sadeghian, and A. Alahi, *Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes*, European conference on computer vision, Springer Cham, 2016.