Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4013647, 10 pages
https://doi.org/10.1155/2020/4013647

Research Article

Object Detection from the Video Taken by Drone via Convolutional Neural Networks

Chenfan Sun, Wei Zhan, Jinhiu She, and Yangyang Zhang

School of Computer Science, Yangtze University, Jingzhou, Hubei, China

Correspondence should be addressed to Wei Zhan; zhanwei814@yangtzeu.edu.cn

Received 23 July 2020; Revised 12 September 2020; Accepted 25 September 2020; Published 13 October 2020

Academic Editor: Liangtian Wan

The aim of this research is to show the implementation of object detection on drone videos using TensorFlow object detection API. The function of the research is the recognition effect and performance of the popular target detection algorithm and feature extractor for recognizing people, trees, cars, and buildings from real-world video frames taken by drones. The study found that using different target detection algorithms on the "normal" image (an ordinary camera) has different performance effects on the number of instances, detection accuracy, and performance consumption of the target and the application of the algorithm to the image data acquired by the drone is different. Object detection is a key part of the realization of any robot's complete autonomy, while unmanned aerial vehicles (UAVs) are a very active area of this field. In order to explore the performance of the most advanced target detection algorithm in the image data captured by UAV, we have done a lot of experiments to solve our functional problems and compared two different types of representative of the most advanced convolution target detection systems, such as SSD and Faster R-CNN, with MobileNet, GoogleNet/Inception, and ResNet50 base feature extractors.

1. Introduction

An object recognition system uses a priori known object model to find real-world pairs from images of the world [1, 2]. Human beings can perform object detection very easily and effortlessly, but this problem is amazingly difficult for machines.

The need for object detection systems is increasing due to the ever-growing number of digital images in both public and private collections. Object recognition systems are important for reaching higher-level autonomy for robots [3]. Applying computer vision (CV) and machine learning (ML), it is a hot area of research in robotics. Drones are being used more and more as robotic platforms. The research in this article is to determine how to use existing object detection systems and models can be used on image data from a drone. One of the advantages of using a drone to detect objects in a scene may be that the drone can move close to objects compared to other robots [4], for example, a wheeled robot. However, there are difficulties with UAVs because of top-down view angels [5] and the issue to combine with a deep learning system for compute-intensive operations [6]. When a drone navigates a scene in search for objects, it is of interest for the drone to be able to view as much of its surroundings as possible [7, 8]. However, images taken by UAVs or drones are quite different from images taken by using a normal camera. For that reason, it cannot be assumed that object detection algorithms normally used on "normal" images perform well on taken by drone images. Previous works on this stress that the images captured by a drone often are different from those available for training, which are often taken by a hand-held camera. Difficulties in detecting objects in data from a drone may arise due to the positioning [9, 10] of the camera compared to images taken by a human, depending on what type of images the network is trained on.

In the previous research, the aim of work was to show whether a network trained on normal camera images could be used on images taken by a drone with satisfactory results. They have used a fish-eye camera and conducted several experiments on three kinds of datasets such as images from a normal camera, images from a fish-eye camera, and rectified images from a fish-eye camera. The result shows that using

drone's fish-eye camera, we can detect many objects–some of the extra number of detected objects compared to the normal camera and the outside field of the view can cause different image quality. [11, 12]. They have done some experiments to prove that more objects can be detected from a closer viewpoint, and the result proved the hypothesis. The result of research work shows that the number of detected object instance and accuracy of detection very depend on the angle of the camera. Their observation tells that it is important to use the same camera angle as in the training dataset. However, different types of objects in the training data are generally photographed from different angles. For example, many images of cups are taken from the side while many images of keyboards are taken from above [9]. Therefore, each object type has an optimal viewing angle depending on the training data. However, as the camera on our drone is assumed to be in a fixed position and the drone itself cannot tilt, one viewing angle has to be chosen. That is why, in our project, we retrained the model using images taken by the drone, and it helped to improve the accuracy of detection.

Other recent studies have used real-time [13, 14] identification of pedestrians, trees, and different types of vehicles and ships from real-world videos of the Caffe framework [15], which are captured with UAVs. In that work, they have observed several experiments to derive the optimal batch size, iteration count, and learning rate for the model to converge [16].

In our experiment, we used TensorFlow object detection API to realize object detection of UAV videos. In our observation, we compared SSD and Faster R-CNN object detection systems depending on speed of detection for frame per second (FPS) and accuracy. In our project, we use Single Shot Detector (SSD) and Faster R-CNN topology as our detection components. We have used them to assess the frame rate and accuracy of several videos we have taken with drones [11].

Section 2 represents TensorFlow object detection API and transfer learning. Section 3 describes design of solution, methods, and network architecture. Section 4 presents the analysis and experiments and shows the results. Section 5 closes the study with a conclusion.

2. Why Choose TensorFlow Object Detection API

If you want to train a complete CNN from scratch, it will take much time and requires very large image datasets [17, 18]. There is a solution to this problem: the solution is to use the advantages of migration learning [12, 19] and TensorFlow's object detection API, which we can use to train, build, and deploy object detection models. Fortunately, API has some pretrained models. Some concepts of transfer learning are described below.

Transfer learning [19] is mainly performed to extract the best solution from another task, and after that, it is applied for the different but related tasks. In deep learning, there are three main ways to use it.

First method, we can use convolutional neural networks as a fixed feature extractor [20], but we change the last fully connected layer of it, and the front part is used as the fixed feature extractor for our new image dataset.

The second way is fine-tuning the CNNs, which is almost the same as the first method, but there is a difference. This method uses continuous back propagation to fine-tune the weights of the pretrained network.

The last is pretrained models. In order to have a new CNN structure, it may take much time if we train it from scratch. Fortunately, TensorFlow model zoo has several pretrained models. Researchers usually open source the checkpoint files of the CNNs [21] they finally trained, so we can use the network for fine-tuning.

In our experiments, we applied transfer learning method on a pretrained GoogleNet/Inception V3 model (trained on Microsoft COCO dataset). The last fully connected layer of the network will be initialized with random weights (or zeroes), so when we input new data to train our model, the final layer's weights will be readjusted. Some of the initial features such as edges and curves have been learned in the topology basic layers. We can apply these initial features [22] for our model to new problems, and this is the main concept of transfer learning. In conclusion, the weight of the fully connected layers will be changed according to the specific tags trained in the dataset of the problem we want to solve.

3. Meta-architectures

Convolutional neural networks (ConvNets) are the most advanced artificial neural networks [23] used for high-accuracy object detection in this decade. Most recently published papers in the areas of ConvNets and computer vision come after this default boxes approach and then reduce a regression loss and associated classification [17, 23] that is explained below. According to default box d, we catch paired ground truth box g (if it exists). If there are pairs, we tag d as "positive" and attach a class label $y_{\mathrm{d}} \in \{1, \ldots, K\}$ and box's vector encoding g with respect to default box d (called the box encoding $\varphi(g_{\mathrm{d}}; d)$). In the case of no pairs, we mark d as "negative" and set the class label to zero. If we suppose box encoding for default box d $f_{\mathrm{loc}}(I; d, \theta)$ and equivalent class $f_{\mathrm{cls}}(I; d, \theta)$, where I represents the image and θ is the model parameters, then the loss of d and the loss of classification are as follows:

$$L(d, I; \theta) = \alpha l_{\mathrm{loc}}\left(\varphi\left(g_{\mathrm{d}}; d\right) - f_{\mathrm{loc}}(I; d, \theta)\right) + \beta l_{\mathrm{cls}}\left(y_{\mathrm{d}}, f_{\mathrm{cls}}(I; d, \theta)\right), \quad (1)$$

where α and β weights are adjust localization and classification losses [24]. Equation (1) is averaged in default boxes and minimized with respect to parameters θ for training our model.

3.1. Single Shot Detector (SSD). The SSD approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location [8]. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape [13]. Additionally, the network combines predictions from multiple feature maps with different resolutions [25–27] to naturally handle objects of

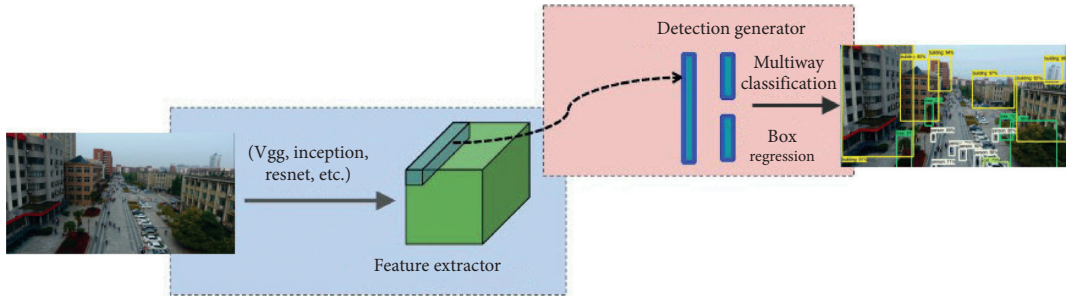


Figure 1: High-level diagram of SSD object detection system.

various sizes. SSD is simple and relative to methods that require object proposals [28] because it eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network (Figure 1). This makes SSD easy to train and straightforward to integrate into systems that require a detection component.

The SSD training method is obtained from the MultiBox method but can accommodate multiple object classes. Suppose $x_{ij}^{p} = \{1, 0\}$, an indicator for pairing the i-th default box to the j-th ground truth box of class p. From the pairing method above, we can write $\sum_i x_{i,j}^{p} \geq 1$. The long-term loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}\left(L_{\text{conf}}(x, c) + \alpha L_{\text{loc}}(x, l, g)\right), \tag{2}$$

where N is the number of paired default boxes. If $N = 0$, set the loss to 0.

$$\begin{aligned} L_{\text{loc}}(x, l, g) &= \sum_{i \in \text{Pos}}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{i,j}^{p}\, \text{smooth}_{L1}\left(l_i^m - \widehat{g}_j^m\right), \\ \widehat{g}_j^{cx} &= \frac{g_j^{cx} - d_i^{cx}}{d_i^w}, \\ \widehat{g}_j^{cx} &= \frac{g_j^{cx} - d_i^{cx}}{d_i^w}, \\ \widehat{g}_j^w &= \log\left(\frac{g_j^w}{d_i^w}\right), \\ \widehat{g}_j^h &= \log\left(\frac{g_j^h}{d_i^h}\right). \end{aligned} \tag{3}$$

The loss of confidence is the softmax on multiclasses confidence (c):

$$L_{\text{conf}}(x, c) = -\sum_{i \in \text{Pos}}^{N} x_{i,j}^{p} \log\left(\widehat{c}_i^{p}\right) - \sum_{i \in \text{Neg}} \log\left(\widehat{c}_i^{0}\right), \tag{4}$$

where

$$\widehat{c}_i^{p} = \frac{\exp\left(c_i^{p}\right)}{\sum_p \exp\left(c_i^{p}\right)}, \tag{5}$$

and the weight term α is set to one through crossvalidation.

3.2. Faster Region-Based Convolutional Neural Networks (Faster R-CNN). In the Faster R-CNN [13] model, the detection process is divided into two stages (Figure 2). The first stage is called as region proposal network (RPN), images are processed by a feature extractor (in our case, GoogleNet/Inception V3), and features at some selected intermediate level (e.g., "conv5") are used to predict class-agnostic box proposals. The last stage is to crop features from these box proposals, and then it is fed to the remainder of the feature extractor (e.g., "fc6" followed by "fc7"). We will get the predict from each proposal (a class and class-specific box refinement) [28]. What we need to know is that the running time depends on the number of regions proposed by the RPN network [13] because a part of the computation must be run in each region, but it does not crop proposals directly from the image and rerun crops through the feature extractor [11], and this is repeated.

3.3. GoogleNet/Inception Module. GoogleNet devised a module called inception module [13, 29] that approximates a sparse CNN with a normal dense construction (shown in the Figure 3), which uses convolutions of varied sizes to capture details at different scales, and the width/number of convolutional filters which have special kernel size is small just because only a little neurons are efficient.

One of the salient points about the module is that it has a so-called bottleneck layer (1×1 convolutions in Figure 3). Bottleneck layers help in massive decrease of the computation requirement as explained here.

We know the first inception module of GoogleNet, which the input has 192 channels. This module has 128 3×3 kernel size filters and 32 5×5 size filters, and the calculation order of 5×5 filter is $25 \times 32 \times 192$. When the width of the network and the number of 5×5 filters increase further, it will explode as we go deep into the network [23]. In order to

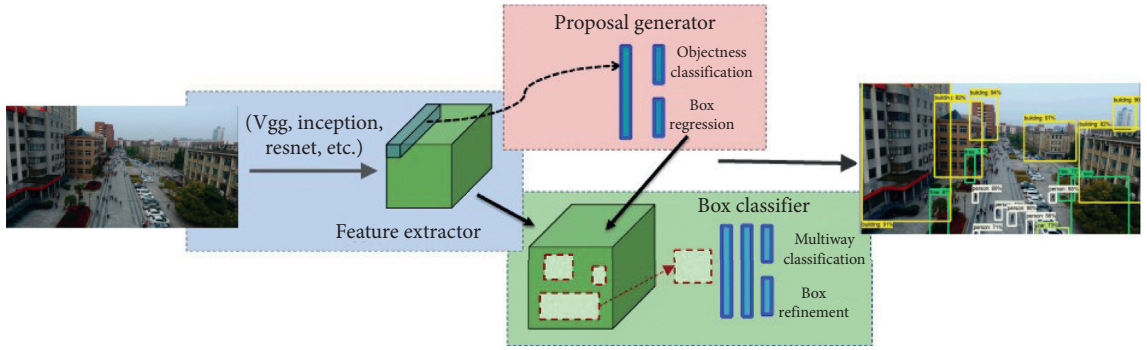


Figure 2: High-level diagram of Faster R-CNN object detection system.

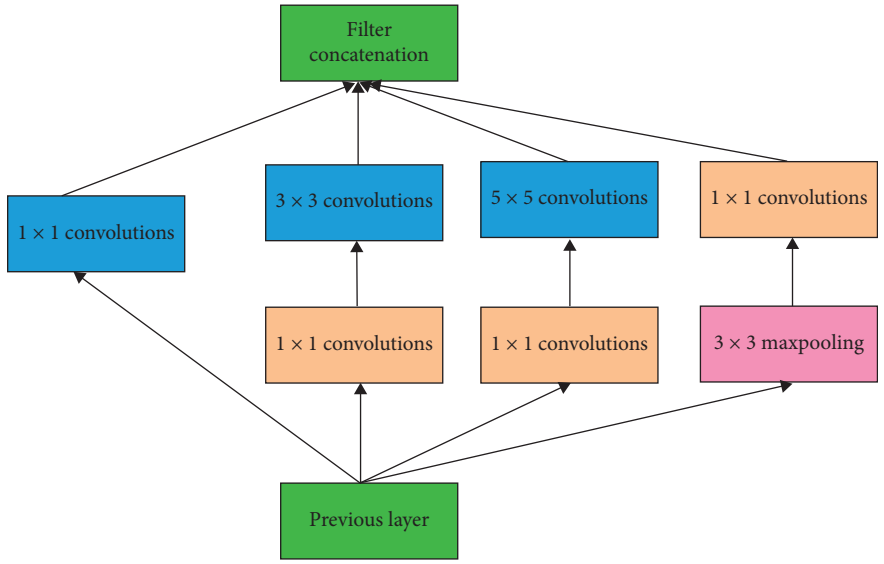


Figure 3: Inception modules.

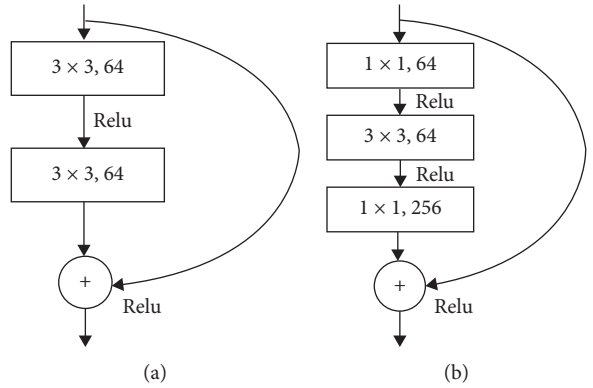


Figure 4: Residual blocks. (a) 64-d, (b) 256-d.

solve this problem, we need to use 1×1 convolution to reduce the channel dimension of the input image before we apply the larger size kernels and then feed them into these larger convolution kernels. Therefore, the input of the inception module is first fed to a filter with only 16 1×1 convolutions and then fed to a 5×5 convolution. As a result, these changes allow the network to have a greater width and depth.

3.4. ResNet. ResNet, or residual networks [30, 31], is one of the deepest networks to this day, and this network won the 2015 ILSVRC.

For any network model we design, we need to calculate the error gradient and then use the back propagation technology to improve the network model. When we use gradient descent (back propagation) to calculate the gradient error, we will use the chain rule of multiplication to calculate the gradient error. Whence, multiplication of many things less than one in the long chain will lead the result to be very small. It will affect the earlier layers in a deep architecture because it cannot update parameters. This will cause the training to become very slow, and the effect is not good; if the gradient becomes zero, the early training parameters cannot be updated.

What will happen if we use back propagation via the identity function? In that case, the gradient would be multiplied by one, and the gradient will not be missed. ResNet stacks residual blocks together [32, 33], which uses the identity function to maintain the gradient [19] (Figure 4).

Residual blocks are very simple to explain. We concatenate the result of applying some functions to the original input with the original input to ensure that the gradient is not less than 1, and no gradient disappears.

Mathematically, we can represent the residual block as follows:

$$H(x) = F(x) + x. \tag{6}$$

We can see from equation (1) that the gradient is unlikely to become zero and we can propagate all gradients backwards. These residual connections are just like a "gradient highway" because the gradient distributes evenly at sums in a computation graph [34].

These residual blocks are very powerful, which can make our network structure deeper. The deepest variant of ResNet was ResNet151 [31]. In our experiments, we have used ResNet50 as a feature extractor.

After revolution of ResNets, now researchers can use skip connections to create a deeper network architecture [32]. Example of ResNet architecture is shown in Figure 5.

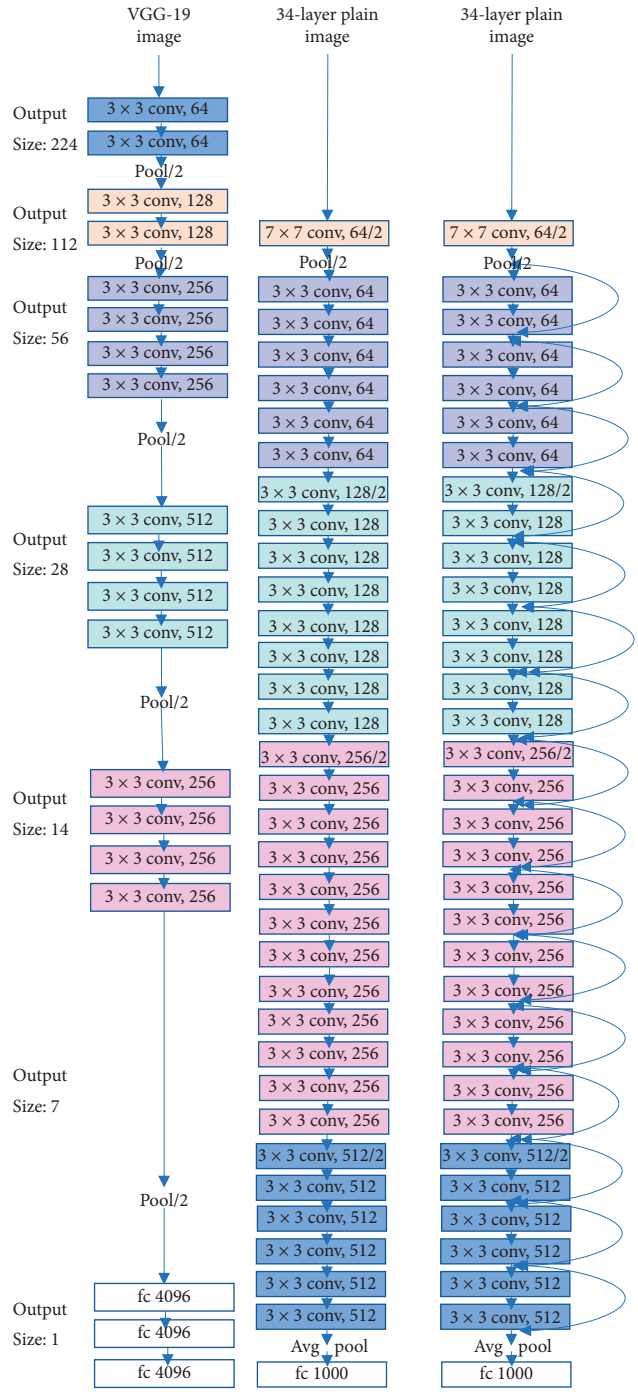


Figure 5: Example ResNet architecture.

4. Experiment Results

In this section, we analyse the results of detections and compare Single Shot Multibox Detector and Faster Region-[31, 32] based convolutional neural network object detection systems for accuracy, speed of detection on GPU and CPU, and memory usage. We organized three sets of experiments to explore object detection on videos captured by the drone. The first and second sets of experiments are focused on testing accuracy of object detection via Faster R-CNNs, and specially, reality of our idea is to apply object recognition systems for drone [35, 36] videos. The third experiment is done using the SSD object detection system.

4.1. Analyses

4.1.1. Accuracy and Time. In the scatter plot (Figure 6), the average mAP of each meta–architecture is visualized. Each image average running time is shown in the figure; it goes from a few hundreds milliseconds to 200 milliseconds. In general, our experiment shows that SSD meta-architectures are faster but with lower accuracy, and Faster R-CNN meta-architectures are more accurate; however, it requires at least 130 milliseconds for one image.

4.1.2. The Effect of the Feature Extractor. To explore the effect of feature extractors, in Table 1, we show some well-known feature extractors that have been used in other works.

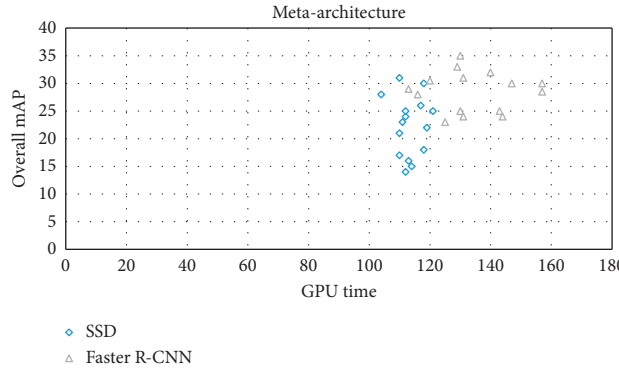


Figure 6: Accuracy and time. Overall mAP and average GPU time.

Table 1: Convolutional detection models.

Paper	Meta-architecture	Feature extractor	Matching	Box encoding φ (;)	Location loss functions
Szegedy et al.	SSD	Inception V3|	Bipartite	$[x_0, y_0, x_1, y_1]$	L_2
Redmon et al.	SSD	R-CNN	Box centre	$[x_c, y_c, \sqrt{w}, \sqrt{h}]$	L_2
Ren et al.	Faster R-CNN	VGG	Argmax	$[(x_c/w_d), (y_c/h_d), \log w, \log h]$	Smooth L_1
He et al.	Faster R-CNN	ResNet101	Argmax	$(x_c/w_d), (y_c/h_d), \log w, \log h$	Smooth L_1
Liu et al.	SSD	Inception V3	Argmax	$[x_0, y_0, x_1, y_1]$	L_2
Liu et al. v1, v2	SSD	VGG	Argmax	$[(x_c/w_d), (y_c/h_d), \log w, \log h]$	Smooth L_1

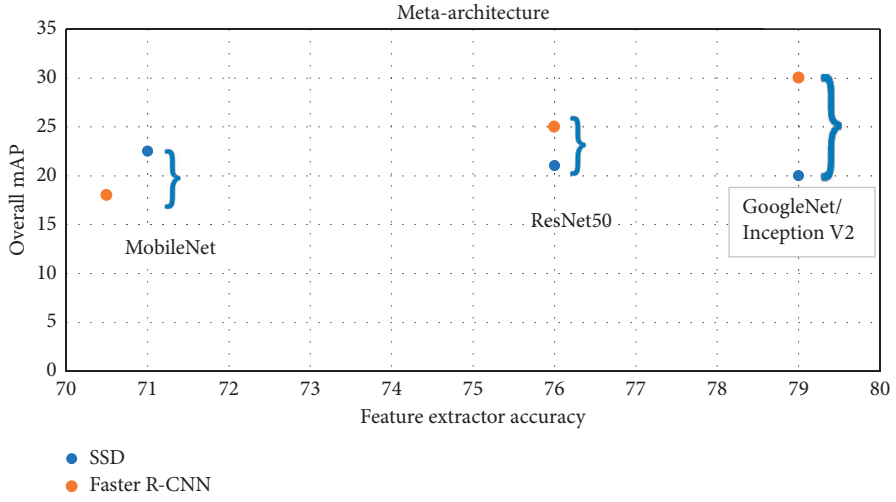


Figure 7: The effect of feature extractors for the overall mAP.

Stronger performance on detection is positively connected with stronger performance on classification, and Figure 7 shows such an overall ratio relationship between classification and detection performance really exists.

4.1.3. The Effect of Image Size. Image resolution can hugely affect object detection accuracy, and it has been shown in many other state-of-the-art researches. Our experiments show that decreasing image dimensions coherently decreases accuracy (by 18% on average) and reduces average running time by a relative factor of 23%. So high-resolution images can solve the problem of small objects, just as Figure 8 shows.

4.1.4. Memory Analysis. In our experiments, the image resolution is 300. We have measured total usage of alternate peak usage for the memory benchmarking. Figure 9 shows the memory usage of different meta-architecture and feature

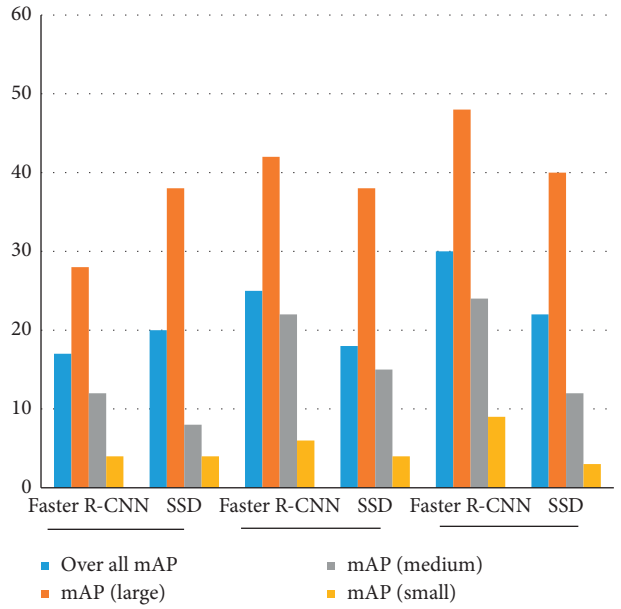


Figure 8: The influence of different object sizes, meta-architectures and feature extractors on accuracy.

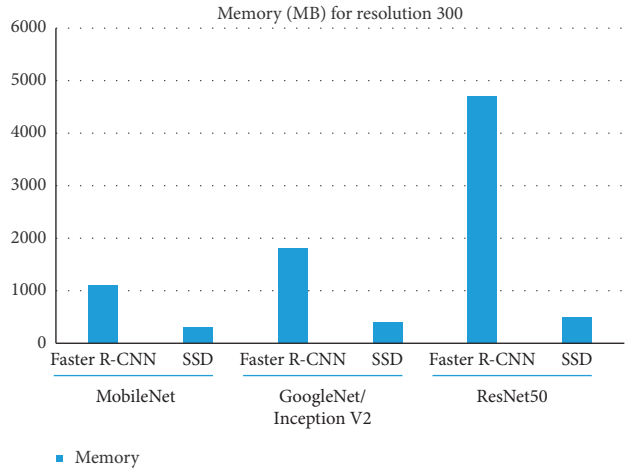
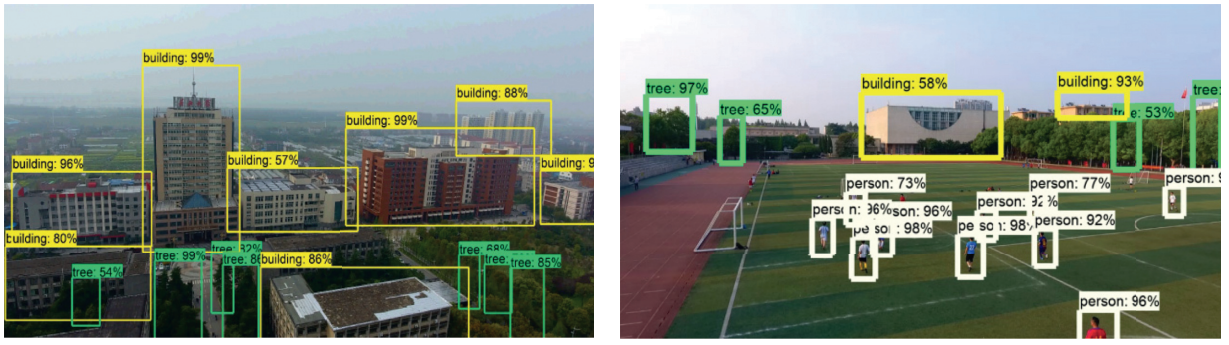
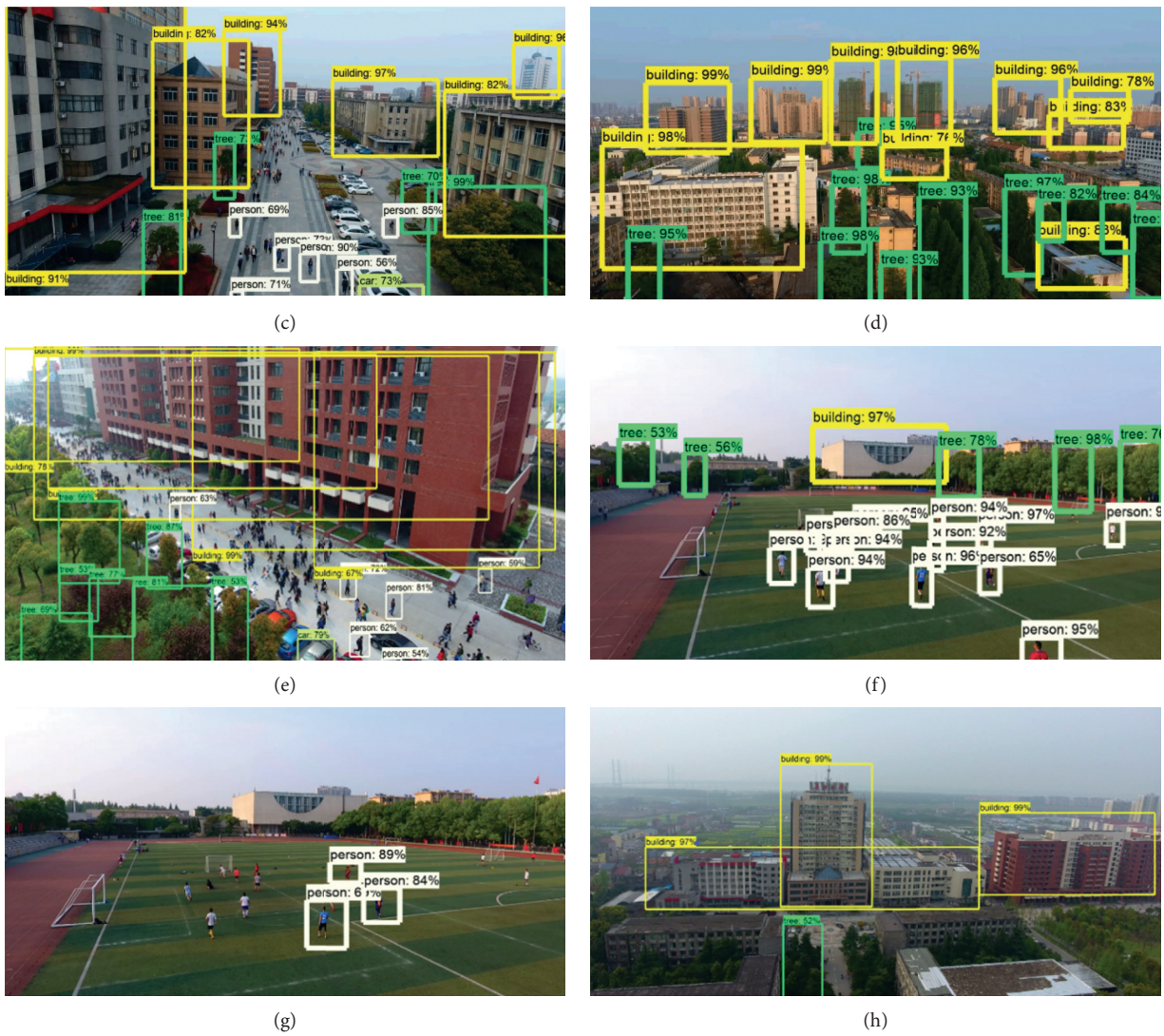


Figure 9: Memory usage for each model.



(a) (b)

Figure 10: Continued.



(c) (d)

(e) (f)

(g) (h)

Figure 10: Example detections from different models and feature extractors. (a) Result of detection with Faster R-CNN base on GoogleNet/Inception V3. (b) Result of detection with Faster R-CNN base on GoogleNet/Inception V3. (c) Result of detection with Faster R-CNN base on GoogleNet/Inception V3. (d) Result of detection with Faster R-CNN base on ResNet50. (e) Result of detection with Faster R-CNN base on ResNet50. (f) Result of detection with Faster R-CNN base on ResNet50. (g) Result of detection with SSD base on GoogleNet/Inception V3. (h) Result of detection with SSD base on GoogleNet/Inception V3.

extractors. Generally, experiments show that larger and powerful feature extractors take much more memory.

We can see that Mobilenet requires the least memory, even less than 1 Gb (almost) in almost all settings.

5. Conclusion

This research is based on object detection from the video taken by the drone via convolutional neural network [37]. In this research, we put forward for consideration to use CNNs to allow drones to recognize some of object types such as building, car, tree, and person [38]. Convolutional neural networks are computationally expensive [39]; even so, we use the method transfer learning to train our neural networks with smaller image datasets. In our project, we use TensorFlow's powerful object detection API, and it helped us to easily construct a new model and deploy for detection.

From the results, we observed that the detection accuracy of the two models for buildings, trees, cars, and people is very high, with an average of more than 85% and a maximum of 99%. We have done some experimental comparison of two modern object detectors for memory usage, speed, and accuracy. SSD models pay more attention to scale, aspect ratio and predictions sampling location than Faster R-CNN, the average time of each frame is 115 ms, but the target detection rate is low. However, Faster R-CNN is more accurate and finds more objects from scene, almost 95% of all objects in the image can be recognized, but the average time of each frame is at least 140 ms. In general, our experiments show that the SSD model is faster on average. On the contrary, Faster R-CNN is relatively slower but more

accurate. But one way to speed up the Faster R-CNN model is to limit the number of proposed regions. In our experiment, we tested the memory and runtime requirements of MobileNet, ResNet50, and GoogleNet/Inception V2. MobileNet requires the least memory, which is less than 1 GB, and ResNet50 has nearly 5 GB of memory on Faster R-CNN, while the memory required by GoogleNet/Inception V2 is in the middle, less than 2 GB. In general, we obtained high correlation with running time and memory between the model and the feature extractors.

The detection results shown in Figure 10 were obtained after implementation on a video captured by a drone.

Hopefully, these experimental results help other researchers to choose a suitable algorithm when selecting object detection for deployment in the real word. The experiment shows that viewing small objects from close is important in order to detect as many objects as possible. This proposes the usage of a drone can be assumed for detecting as many objects as possible as a drone is designed flexible.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form part of an ongoing study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper was supported by the 2018 Jingzhou Science and Technology Development Plan Project "Research and Development of New Methods of UAV Visual Data Deep Learning Identification Technology in Urban Management" and the 2019 Yangtze University 12th batch of college students innovation and entrepreneurship project "Implementation of Real-Time Body Gesture Tracking Algorithm on Android Mobile Terminal under the Framework of TensorFlow Lite."

References

[1] Z. Zhang, W. Zhan, Z. He, and Y. Zou, "Application of spatio-temporal context and convolution neural network (CNN) in grooming behavior of *Bactrocera minax* (Diptera: trypetidae) detection and statistics," *Insects*, vol. 11, no. 9, p. 565, 2020.

[2] W. Zhan, I. Ramatov, W. Cui et al., "Survey of deep learning target detection algorithms based on candidate regions," *Journal of Yangtze University*, vol. 16, no. 5, pp. 108–115, 2019.

[3] X. Hui, B. Jiang, Y. Yu, X. Zhao, and M. Tan, "A novel autonomous navigation approach for UAV power line inspection," in *Proceedings of the 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Macau, China, December 2017.

[4] R. Jain, R. Kasturi, and B. G. Schunck, *Object Recognition*, Machine Vision, McGraw-Hill, NY, USA, 1995.

[5] F. Wen, Z. Zhang, K. Wang, G. Sheng, and G. Zhang, "Angle estimation and mutual coupling self-calibration for ULA-based bistatic MIMO radar," *Signal Processing*, vol. 144, pp. 61–67, 2018.

[6] F. Wen and J. Shi, "Fast direction finding for bistatic EMVS-MIMO radar without pairing," *Signal Processing*, vol. 173, Article ID 107512, 2020.

[7] J. C. van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich, "Nature conservation drones for automatic localization and counting of animals," *Computer VisionECCV 2014 Workshops*, vol. 8925, pp. 255–270, 2015.

[8] P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery," in *Proceedings of the 2008 IEEE Aerospace Conference*, Big Sky, MT, USA, March 2008.

[9] L. Grip, "Vision based indoor object detection for a drone," M.Sc. thesis dissertation, KTH, Stockholm, Sweden, 2017.

[10] J. Zheng, T. Yang, H. Liu, T. Su, and L. Wan, "Accurate detection and localization of UAV swarms-enabled MEC system," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[11] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe, "Scalable, high-quality object detection," 2014, https://arxiv.org/abs/1412.1441.

[12] C. N. Truong, N. Q. H. Ton, H. P. Do, and S. P. Nguyen, "Digit detection from digital devices in multiple environment conditions," in *Proceedings of the 2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*, Ho Chi Minh City, Vietnam, October 2020.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Washington State Convention Center, Seattle, WA, USA, pp. 1316–1322, May 2015.

[15] R. A. (Intel), Krishnaprasad T. (Intel), Object detection on drone videos using caffe framework, 2018, http://icml.cc/2016/?page_id=97.

[16] A. G. Howard, M. Zhu, B. Chen et al., "MobileNets: efficient convolutional neural networks for mobile vision applications," 2017, https://arxiv.org/abs/1704.04861.

[17] K. Bregar and M. Mohorcic, "Improving indoor localization using convolutional neural networks on computationally restricted devices," *IEEE Access*, vol. 6, pp. 17429–17441, 2018.

[18] V. Viitaniemi and J. Laaksonen, "Techniques for image classification, object detection and object segmentation," *Visual Information Systems. Web-Based Visual Information Search and Management, Lecture Notes in Computer Science*, pp. 231–234, 2008.

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, vol. 27, Montreal, Canada, December 2014.

[20] L. Wan, X. Kong, and F. Xia, "Joint range-Doppler-angle estimation for intelligent tracking of moving aerial targets," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1625–1636, 2018.

[21] X. Wang, L. Wang, X. Li, and G. Bi, "Nuclear norm minimization framework for DOA estimation in MIMO radar," *Signal Processing*, vol. 135, pp. 147–152, 2017.

[22] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for

recognition," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, Columbus, OH, USA, June 2014.

[23] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, Boston, MA, USA, June 2015.

[24] A. Krizhevsky, I. Sulskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[25] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," 2017, https://www.researchgate.net/publication/335844785_DATA_AUGMENTATION_APPROACHES_FOR_SATELLITE_IMAGE_SUPER-RESOLUTION.

[26] J. Wu, Y. Yu, C. Huang, and K. Yu, "Deep multiple instance learning for image classification and auto-annotation," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3460–3469, Boston, MA, USA, June 2015.

[27] V. Mnih and G. Hinton, "Learning to detect roads in high-resolution aerial images," in *Proceedings of the 11th European conference on Trends and Topics in Computer Vision*, pp. 1–14, Heraklion, Greece, September 2010.

[28] I. Mansour, R. Hoffman, O. Rawashdeh, and B. Seifert, "A proposed heterogeneous convolutional neural network for embedded automotive vision algorithms," in *Proceedings of the 2018 IEEE International Conference on Electro/Information Technology (EIT)*, Rochester, MI, USA, May 2018.

[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015, https://arxiv.org/abs/1512.00567.

[30] H. Kaiming, "Deep residual networks: deep learning gets way deeper," in *Proceedings of the ICML 2015 32nd International Conference on Machine Learning*, Lille, France, July 2015.

[31] F. Wang, M. Jiang, C. Qian et al., "Residual attention network for image classification," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 3156–3164, Honolulu, HI, USA, July 2017.

[32] A. B. Makar, K. E. McMartin, M. Palese, and T. R. Tephly, "Formate assay in body fluids: application in methanol poisoning," *Biochemical Medicine*, vol. 13, no. 2, pp. 117–126, 1975.

[33] S. Wu, S. Zhong, and Y. Liu, "Deep residual learning for image steganalysis," *Multimedia Tools and Applications*, pp. 1–17, 2017.

[34] J. Huang, V. Rathod, C. Sun et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2017, pp. 3296–3305, Las Vegas, NV, USA, July 2017.

[35] F. J. C. Condessa, "Robust image classification with context and rejection," *IEEE GRSS Workshop Hyperspectral Image Signal*, 2015.

[36] P. Chen, Y. Dang, R. Liang, W. Zhu, and X. He, "Real-time object tracking on a drone with multi-inertial sensing data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 131–139, 2018.

[37] W. Cui, W. Zhan, J. Yu, C. Sun, and Y. Zhang, "Face recognition via convolutional neural networks and siamese neural networks," in *Proceedings of the International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pp. 746–750, Chongqing, China, December 2019.

[38] V. Lempitsky and A. Zisserman, "Learning to count objects in images," *Advances in Neural Information Processing Systems*, pp. 1324–1332, MIT Press, Cambridge, MA, USA, 2010.

[39] T. Liu, F. Wen, J. Shi, Z. Gong, and H. Xu, "A computationally economic location algorithm for bistatic EVMS-MIMO radar," *IEEE Access*, vol. 7, pp. 120533–120540, 2019.