Optimizing YOLOv5 for Green AI: A Study on Model Pruning and Lightweight Networks

Bangguo Xu
Contributed to sections 4 and 5.Responsible for model pruning, fine-tuning and code integration .

Simei Yan
Contributed to sections 3 and 6, and responsible for backbone replacement and literature collection

Liang Liu
Contributed to sections 1 and 2, and responsible for sparse training and literature collection

ABSTRACT

To achieve state-of-the-art performance, deep learning models are becoming increasingly complex, leading to a significant increase in demand for high-performance computing resources and, in turn, concerns about environmental impact. In this context, the concept of Green AI is proposed, which advocates optimizing performance by improving model efficiency, rather than relying solely on an increase in computing resources, thereby reducing the impact on the environment. Object detection is a research hotspot in the field of computer vision. This paper focuses on the commonly used YOLOv5 network in object detection, optimizing the YOLOv5 model through pruning and the use of lightweight networks. Implemented on a custom THWS campus image dataset, a balance between reducing computational load and maintaining accuracy was achieved. The experimental results confirm that strategic model pruning and thoughtful network architecture selection can produce environmentally responsible and computationally efficient deep learning models without significantly reducing performance, aligning with the goals of Green AI. All related code of the project is available at: `https://github.com/xbgthws/Green-AI-project.git`

KEYWORDS

Transfer Learning, MobileNet, ShuffleNet, GhostNet, LabelImg

1 INTRODUCTION

In recent years, the complexity and performance of Artificial Intelligence (AI) and machine learning models have continually increased, leading to a significant demand for high-performance computing resources. This demand encompasses not only a reliance on high-performance computers but also involves substantial electricity consumption and corresponding cooling requirements, thereby having a considerable environmental impact. This issue becomes particularly pronounced in training large-scale deep learning models, where energy consumption and carbon emissions have become focal points of concern. The concept of "Green AI[24]," first introduced by Roy Schwartz et al. in 2019, emphasizes the importance of minimizing the environmental impact while advancing and innovating in AI technology. Green AI advocates for optimizing model performance through innovation and efficiency improvements rather than solely relying on increasing computational resources.

In this context, YOLOv5 (You Only Look Once version 5), was proposed as a popular object detection algorithm. This algorithm is widely recognized for its exceptional speed and accuracy, making it particularly suitable for real-time object detection tasks. We aim to explore how to reduce energy consumption and enhance the efficiency of YOLOv5 through algorithmic and network structure optimizations, without significantly sacrificing performance, which aligns with the objectives of Green AI.

2 RELATED WORK

2.1 LabelImg

LabelImg, an open-source tool developed in Python, provides an intuitive interface for image annotation, essential for computer vision tasks. In our dataset, we annotated all classes within each image, saving these annotations in the YOLO format, which records the bounding box coordinates and corresponding class labels. LabelImg has been widely adopted by recent research literature to perform object detection tasks related to YOLO networks, e.g. [4, 11, 26].

2.2 YOLO

In the domain of deep learning for object detection, there are generally two approaches: two-stage and one-stage methods. The two-stage methods, such as R-CNN[3], Fast R-CNN[2] and Faster R-CNN[21], first generate region proposals and then classify them, generally achieving high accuracy but at a slower pace. On the other hand, one-stage methods like YOLO and SSD (Single Shot MultiBox Detector)[14] detect objects directly in a single step, suitable for application with high real-time requirements.

The YOLO series, initiated by Joseph Redmon et al. in 2016 with YOLOv1[18], innovatively solves object detection as a regression problem, obtaining the position, class, and confidence probability of objects in an image through a single inference. YOLOv2, or YOLO9000[19], introduced in 2017, improved in accuracy, speed, and detectable object categories. YOLOv3[20], released in 2018, enhanced the network architecture with multi-scale detection. YOLOv4 [1], developed in 2020 by Alexey Bochkovskiy et al., integrated various previous research techniques for a balance of speed and accuracy. YOLOv5, by Glenn Jocher and others, is an open-source project from Ultralytics with versions ranging from n to x, scaling up in parameter size and performance.

2.3 Pretrained Models and Fine-tuning

Transfer learning is a method that allows knowledge trained on one task to be applied to another related task[29]. Pretrained models are a commonly used tool in transfer learning, which refer to models that have been trained on large datasets. By using a pre-trained model as a starting point and further training (i.e., fine-tuning) on a task-specific dataset, you can speed up the learning process and improve the model's performance on new tasks.

Recent studies have shown that initializing with transferred features can improve the generalization performance of a model, even after substantial fine-tuning on a new task[27]. This has been

identified as a generally useful technique for enhancing the performance of deep neural networks. Additionally, convolutional neural networks are capable of learning rich mid-level image features that are transferrable to a variety of visual recognition tasks[17]. In the application of YOLOv5, this means that the model does not need to learn all features from scratch but can quickly adapt to new tasks by fine-tuning a pretrained network, thus identifying specific objects in new datasets, reducing training time, and improving learning efficiency. In scenarios where labeled data is scarce, transfer learning offers a significant advantage to researchers or projects lacking the resources for extensive data annotation, thereby saving computational resources and lowering barriers to entry[22]. In this way, YOLOv5 can effectively utilize transfer learning to achieve higher performance and efficiency in new domains or tasks.

3 PRELIMINARY

This paper aims to explore the possibility of realizing Green AI by optimizing the YOLOv5 network, especially to improve model efficiency while minimizing the impact on accuracy. In the previous section, we have discussed the relevant tools used in this experiment. Next, we will discuss in depth the theoretical foundations and practical applications of model pruning and lightweight neural networks. These strategies are crucial for optimizing the YOLOv5 network to adapt to resource-constrained environments.

3.1 Pruning

In our research, we focus on adjusting the importance coefficients of specific layers in neural networks. These coefficients determine the contribution of each feature to the network's output. By identifying and gradually reducing the importance of less contributive features[15], we can optimize the network's structure. By integrating L1 regularization into the loss function as an additional penalty, we encourage the model to recognize and reduce the dependency on unimportant features[6]. This process, guiding the sparsity of the gamma parameters in the batch-normalization layers, further refines the network's architecture. As a result, unimportant features are removed from the final network structure, simplifying the model. Applying this technique to the YOLOv5 model for object detection allows us to maintain high accuracy while reducing model complexity and computational demands. Practically, this involves applying regularization pressure to specific network parameters, driving them towards zero throughout the training iterations. This strategy provides us with a more streamlined and efficient network structure[8, 13]. Our experimental process is divided into three stages: first, we conduct standard training on the YOLOv5 model to establish a baseline; next, we apply sparsity training to prepare for model pruning; and finally, we perform the pruning process, removing features deemed non-essential, and meticulously adjust the model.

3.2 Lightweight Network

With the rapid advancement of deep learning in computer vision, starting from the advent of AlexNet[12] in 2012, followed by VGG[25] in 2014, and ResNet[7] in 2015, convolutional neural networks have made significant strides in image processing. However, the increasing size and complexity of these networks have escalated hardware requirements, often limiting them to high-performance servers. In order to address the hardware constraints of mobile and embedded devices, lightweight neural networks offer efficient solutions for resource-limited settings, maintaining accuracy while significantly reducing model size and increasing computational speed. In our experiments, substituting YOLOv5's backbone network with these lightweight networks achieved a considerable reduction in model size while preserving accuracy.

MobileNet MobileNetV1[10], introduced by Andrew G. Howard et al. in 2017, marked a significant shift in convolutional neural network design. Its key innovation is the use of depthwise separable convolutions, which break down standard convolutions into depthwise and pointwise convolutions, greatly reducing parameters and computational load. Unlike traditional convolutions, depthwise convolutions operate separately on each channel of the input feature map, while pointwise convolutions combine these features, akin to the role of fully connected layers. MobileNetV2[23] further refined the architecture by introducing linear bottlenecks and inverted residual structures, leading to deeper yet more compact and faster networks. MobileNetV3[9] advanced performance further with hardware-aware Neural Architecture Search (NAS), achieving better performance with less computation.

ShuffleNet[28], proposed by Xiangyu Zhang et al. in 2017, significantly reduces parameters and computational costs through the use of pointwise grouped convolutions. To counter potential information loss from grouped convolutions, it introduces a channel shuffle operation, which rearranges channels after grouped convolutions to exchange information between different groups, enriching feature representation. This unique structural design effectively enhances the network's learning ability, ensuring ShuffleNet maintains high computational efficiency and accuracy while being lightweight, making it particularly suitable for resource-constrained application environments.

GhostNet[5] creates feature maps using fewer parameters, reducing computational demands. The key to this method is its Ghost module. This module uses a small number of convolution kernels to produce basic feature maps, and then simple linear operations are applied to create additional "ghost" feature maps. This design concept is crucial for creating lightweight and efficient networks. To integrate the GhostNet architecture into YOLOv5, we modified the original convolution layers, replacing standard convolution layers with Ghost modules[16]. These modules first use fewer filters to generate the main feature maps, then apply simple operations to produce the so-called "ghost" feature maps. This expands the number of feature maps without needing extensive computation.

4 EXPERIMENT

To evaluate the impact of model pruning and backbone network substitution, the experiments analyzed accuracy, parameters, GFLOPs, model size and inference time before and after modifications. Training utilized Google Colab and NVIDIA Tesla T4 GPU, while inference speed was measured on CPUs to simulate a resource-constrained environment.

Table 1: Comparative Performance Metrics of Pruned Models on the THWS Dataset

Model Name	Sparcity	Prune ratio	Precision	mAP50[1]	Parameters	Model Size	GFLOPs	Inference Speed(CPU)
Normal Training (Baseline)	-	-	0.8	0.722	7.05M	14.5MB	15.9	295.7ms
Sparse Model 1	0.0001	-	0.837	0.741	7.05M	14.5MB	15.9	302.8ms
Sparse Model 2	0.0005	-	0.825	0.771	7.05M	14.5MB	15.9	296.9ms
Sparse Model 3	0.01	-	0.718	0.658	7.05M	14.5MB	15.9	295ms
Prune Model 1	0.0001	30%	0.455	0.304	4.12M	8.111MB	12.2	246.7ms
Prune Model 2	0.0001	40%	0.128	0.071	3.36M	6.67MB	10.8	220.7ms
Prune Model 3	0.0005	30%	0.835	0.736	4.42M	8.687MB	13.2	257.8ms
Prune Model 4	0.0005	50%	0.578	0.44	2.72M	5.429MB	11.2	219.8ms
Prune Model 5	0.01	25%	0.717	0.658	5.11M	10.014MB	13.2	254.1ms
Fine-tune Model 1	0.0001	30%	0.834	0.788	4.12M	8.302MB	12.2	240.4ms
Fine-tune Model 2	0.0001	40%	0.808	0.789	3.36M	6.862MB	**10.8**	**217.6ms**
Fine-tune Model 3	0.0005	30%	0.771	0.79	4.42M	8.876MB	13.2	253ms
Fine-tune Model 4	0.0005	50%	**0.86**	0.786	**2.72M**	**5.621MB**	11.2	223.8ms
Fine-tune Model 5	0.01	25%	0.687	0.679	5.11M	10.208MB	13.2	258.1ms

Table 2: Performance Comparison of Models with Different Backbone Networks on the THWS Dataset

Model Name	Precision	mAP50	Parameters	Model Size	GFLOPs	Inference Speed(CPU)
Normal Training (Baseline)	0.8	0.722	7.05M	14.5MB	15.9	295.7ms
MobileNetv2	0.776	0.661	2.94M	6.4MB	7.1	243.3ms
MobileNetv3 Small	0.748	0.663	5.05M	10.6MB	11.4	227.8ms
MobileNetv3 Large	0.747	0.674	5.57M	11.6MB	10.8	265.8ms
ShuffleNet	0.784	0.66	3.82M	8.1MB	8.1	212.5ms
GhostNet	0.759	0.63	3.72M	7.9MB	8.3	206ms

4.1 Dataset

The dataset utilized in the experiment was provided by Mr. Niko Wörtmann, consisting of the THWS Campus Image Dataset. This dataset encompasses a variety of natural campus landscapes, buildings, facilities, and various campus events and celebrations. Our team selected 565 images from this dataset, spanning 14 categories for training purposes. These categories include chair, person, tree, robot, door, fire extinguisher, screen, clock, podium, projector, backpack, laptop, bottle, and window. All images were annotated using the LabelImg tool and were divided into training, validation, and test sets at a ratio of 8:1:1. Figure 1 shows the recognition result of some sample images from the dataset. Given the limited size of the dataset, we employed a transfer learning approach using the YOLOv5s as a pre-trained model to expedite the learning process.

4.2 Training, Pruning and Fine-tuning

Normal Training. We began by employing YOLOv5s.pt as the pre-training model for transfer learning on our THWS image dataset, setting as a baseline. We experimented with various optimizers, including Adam, AdamW, and SGD, each using a uniform input image size of 640 and training for 100 epochs. To reduce data transfer time

Figure 1: Model Predictions on Some Sample Dataset Images

[1]mAP50 refers to the Mean Average Precision at an Intersection over Union (IoU) threshold of 50%. It is a common metric used for evaluating the performance of an object detection model at predicting bounding boxes around objects.

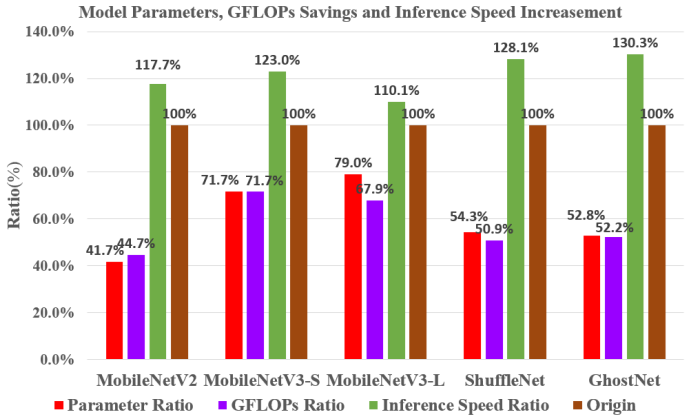

Figure 2: Comparison of models using lightweight network as backbone with the original model on THWS dataset.

and reading latency and to enhance GPU utilization, we preloaded and stored image size and label information of the training dataset in memory. This approach resulted in faster model training. For simplicity, we chose AdamW as the optimizer for all subsequent related trainings.

Training with Sparsity. On top of the normal training, we selected different values of the sparse hyper parameter λ from 10^{-5} to 10^{-2} for the channel sparse regularization. To avoid redundant training, all sparse training processes are limited to 50 epochs.

Pruning. After sparse training, we pruned models at different sparsity levels. Our pruning process involved collecting and sorting BN layer weights, setting a threshold to preserve essential features, calculating and displaying the gamma threshold and corresponding pruning ratio, and then adjusting the actual pruning ratio based on the maximum calculated value.

Fine-tuning. We fine-tuned the pruned models, which had a significant reduction in the number of parameters and model size. We made slight alterations to the hyperparameter settings used for fine-tuning compared to normal training. We lowered the initial learning rate from 0.01 to 0.001 and increased the learning rate decay to facilitate faster model convergence. We also made minor adjustments to other hyperparameters related to data augmentation.

4.3 Backbone Replacement.

In the experiment involving backbone network replacement, we substituted YOLOv5's original backbone network with various alternatives, including MobileNetV2, MobileNetV3-Small, MobileNetV3-Large, ShuffleNet, and GhostNet, for individual training sessions. Unlike normal training, due to the alteration of the backbone network architecture, we initiated training from scratch instead of using the YOLOv5s model as a pre-training model. Also, we maintained the same hyperparameters as those used in normal training.

5 DISCUSSION

Analysis of the data presented in Table 1 indicates a general decline in mAP50 as model sparsity increases. The pruning experiments reveal that most models undergo substantial accuracy degradation after more than 30% of parameters are eliminated. However, post-pruning fine-tuning markedly recovers performance. In contrast with the models trained normally, the fine-tuned models exhibit a mere 1.77% drop in average accuracy, yet benefit from a 43% reduction in size, a 20% decrease in GFLOPs, and an 18% average acceleration in CPU inference speed. This underscores fine-tuning as an effective countermeasure to pruning's adverse effects.

The results of replacing the backbone network are similar to those of model pruning. As shown in Table 2 and Figure 2 for example, after using MobileNetV2 as the backbone network, the model size is reduced by 44.1% compared with the standard model, the inference speed is increased by 17.7%, the GFLOPs are only 44.6% of the baseline, while the mAP50 remains at 0.661. These results demonstrate that lightweight networks can still achieve satisfactory performance in object detection tasks despite the reduced parameters and computational complexity of the model.

In summary, both pruning and backbone network replacement with lightweight networks can effectively reduce the model size, decrease the model complexity as well as improve the inference speed, but may sacrifice the detection accuracy of the model. In view of this, future research should explore how to choose the appropriate pruning ratio and perform related fine-tuning to find a better balance between performance and efficiency. It is also important to investigate how to choose the right backbone network to optimise the trade-off between speed and accuracy in different application scenarios.

6 CONCLUSIONS

In this study, we systematically studied the effect of model pruning and replacement backbone network on optimizing the YOLOv5 model. Our experimental results show that by proposing the designed pruning strategy and upgrading the model while significantly reducing the number of parameters, the model can still maintain a high accuracy. Although we achieved certain results, the slight decrease in accuracy also points to the direction of future research. Future research should focus on how to further optimize pruning and fine-tuning strategies without sacrificing detection accuracy to achieve the best balance between performance and efficiency. In addition, it is also crucial to select the most appropriate backbone network for different application scenarios to finely adjust the balance between speed and accuracy.

ACKNOWLEDGMENTS

We thank Prof. Dr. Frank-Michael Schleif for his guidance and support throughout the duration of this project. His expertise and insightful suggestions were crucial in shaping both the direction and success of our research.

REFERENCES

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

[2] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[4] Marielet Guillermo, Rogelio Ruzcko Tobias, Luigi Carlo De Jesus, Robert Kerwin Billones, Edwin Sybingco, Elmer P Dadios, and Alexis Fillone. 2020. Detection and classification of public security threats in the philippines using neural networks. In *2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*. IEEE, 320–324.

[5] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. 2020. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1580–1589.

[6] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1510.00149

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 770–778. https://doi.org/10.1109/CVPR.2016.90

[8] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research* 22, 1 (2021), 10882–11005.

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[11] Pavuluri Jithendra, Tummala Vinay Sai, Raj Kumar Mannam, Ramini Manideep, and Shahana Bano. 2020. Cognitive Model for Object Detection based on Speech-to-Text Conversion. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*. IEEE, 843–847.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 1106–1114. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning Filters for Efficient ConvNets. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJqFGTslg

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 21–37.

[15] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*. 2736–2744.

[16] Tanzilal Mustaqim, Chastine Fatichah, and Nanik Suciati. 2022. Modification of YOLO with GhostNet to Reduce Parameters and Computing Resources for Detecting Acute Lymphoblastic Leukemia. In *2022 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 167–172.

[17] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1717–1724.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[19] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

[20] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015).

[22] Sajal Saha, Anwar Haque, and Greg Sidebottom. 2023. Transfer learning based efficient traffic prediction with limited training data. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 477–480.

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[24] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020. Green ai. *Commun. ACM* 63, 12 (2020), 54–63.

[25] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.1556

[26] Vibhanshu Singh Sindhu. 2021. Vehicle identification from traffic video surveillance using YOLOv4. In *2021 5th international conference on intelligent computing and control systems (ICICCS)*. IEEE, 1768–1775.

[27] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems* 27 (2014).

[28] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6848–6856.

[29] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. 2021. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* 109, 1 (2021), 43–76. https://doi.org/10.1109/JPROC.2020.3004555