Towards a Lightweight Object Detection through Model Pruning Approaches

Hyerim Yu[1], SangEun Lee[2], Byeongsang Yeo[3], Jinyoung Han[4], Eunil Park[1,4,*], Sangheon Pack[5]
[1]Department of Interaction Science, Sungkyunkwan University, Seoul, Korea
[2]Content Intelligence Research Section, Electronics and Telecommunications Research Institute, Daejeon, Korea
[3]AI Team, Pai Media Lab, Seoul, Korea
[4]Department of Applied Artificial Intelligence, Sungkyunkwan University, Seoul, Korea
[5]School of Electrical Engineering, Korea University, Seoul, Korea

***Abstract*—Object detection tasks represent one of the most prevalent areas of study in computer vision, leading to the introduction of numerous techniques. Among these, the You Only Look Once (YOLO) series of object detection models continued to evolve and progress. The latest iterations within the YOLO family exhibit enhanced performance and quicker inference times. However, the increased capacities and memory demands of these models present real-world challenges in terms of practical deployment. This underscores the importance of developing lightweight versions of the updated YOLO models to ensure their applicability in real-life scenarios. In this context, this study introduces YOLOv7 lightweight, building upon a prior channel pruning technique employed for YOLOv5. By adopting the foundational method to align with the YOLOv7 architecture, we effectively managed to reduce the model's complexity. Furthermore, this research delves into identifying the appropriate pruning levels and model configurations tailored specifically for human detection tasks. In the course of our investigation, we evaluated the trade-off between performance degradation and reductions in parameters and computational complexity. This analysis led us to select a pruning protection ratio of 50% as the most optimal value. Moreover, this article presents the optimization of the lightweight YOLOv7 model for efficient human detection. In essence, our research not only suggests enhancements to existing methodologies for updated models but also emphasizes the practical application of such methods through a comprehensive grasp of the unique characteristics of updated models.**

***Index Terms*—Deep Learning, YOLOv7, Object Detection, Pruning, Human Detection**

I. Introduction

Object detection is a crucial task that involves both classifying and localizing objects within an image [1, 2]. Its diverse applications, such as face recognition [3, 4], video surveillance [5, 6], and action recognition [7, 8], make designing a portable and efficient detection network a significant focus in current research.

This work was supported by an internal fund/grant of the Electronics and Telecommunications Research Institute(ETRI). [23ZT1100, ICT convergence technology support and development based on local industry in the metropolitan area]. This research was also supported by the MSIT, Korea, under the ICAN support program(IITP-2023-RS-2023-00259497) supervised by the IITP. Moreover, this research was supported by Seoul R&BD Program (CY220090) through the Seoul Business Agency (SBA) funded by the Seoul Metropolitan Government.

*Corresponding author: Eunil Park (eunilpark@skku.edu)

One of the major challenges in object detection is striking the right balance between accuracy and computational complexity. The emergence of one-stage object detectors, which achieve faster prediction speeds by simultaneously handling classification and localization, has drawn considerable attention to YOLO (You Only Look Once) in the object detection field [9]. Actually, YOLO models have demonstrated successful application in real-world scenarios such as detecting workers and small objects within smart factories [10], as well as performing human detection tasks [11]. However, YOLO still faces difficulties in meeting the real-time object detection requirements on portable devices due to the significant computational resources it demands [12]. Furthermore, the ongoing concern surrounding up-to-date YOLO models is their requirement for a high-end computing environment [13].

On the contrary, there has been a growing interest in developing lightweight deep-learning frameworks to ease the deployment of models on portable devices [14]. Among these techniques, pruning methods have proven effective in reducing the computational burden by removing redundant parameters from neural networks [15, 16, 17]. For instance, Yin et al. [18] demonstrated the efficacy of pruning methods in Chinese character recognition, achieving high performance while reducing the network size. Moreover, Agarwal et al. [19] showed that using pruning algorithms for lung segmentation in COVID-19 diagnosis results in low storage requirements and fast model inference.

We introduce a lightweight object detection model tailored specifically for human detection tasks within the context of a smart factory environment. This model can incorporate the pruning technique applied to YOLOv7, one of the most recent versions in the YOLO series. Building on the success of previous pruned YOLO models, we integrate the channel pruning strategy into YOLOv7. Channel pruning involves eliminating a set of filters at the channel level in convolutional layers based on the computational importance of each channel.

Experimental results showcase the effectiveness of our lightweight model in the human detection task, automatically and accurately capturing regions of human heads or bodies. This approach holds great promise for efficient object detection on resource-constrained devices.

Thus, the main contributions are summarized as follows.

- We present a lightweight method for YOLOv7, utilizing the channel pruning strategy. The goal of this method is to reduce the computational burden when deploying the model on portable devices. By removing unimportant channels in the convolutional layers, we can achieve efficient model deployment without compromising the high performance of YOLOv7.
- Empirical results validate the effectiveness of our lightweight method in human detection tasks.

The structure of the paper is organized as follows: Section II discusses and analyzes recent studies on object detection and lightweight deep learning. Then, Section III proposes our lightweight method for YOLOv7. Section IV covers the experiment settings, dataset, and presents the results obtained from our method. Finally, in Section V concludes the research findings and provide an in-depth analysis of the significance and implications of this study.

II. Related Work

A. Object Detection Using Deep Neural Networks

In recent times, real-time object detection tasks have garnered increased attention, leading to the introduction and examination of various innovative frameworks. One of the most widely used detection networks is YOLO [9], known for its fast inference speed. YOLO employs a single unified neural network that directly predicts object-bounding boxes in a one-stage manner. Among the YOLO family of models, YOLOv7 [20] stands out as one of the latest advancements, achieving improved prediction accuracy while maintaining similar inference speed to previous models. YOLOv7 incorporates two key improvements: model architecture and optimization strategies.

In terms of the model architecture, YOLOv7 introduces Extended-ELAN (E-ELAN), which gradually enhances the learning ability of the network as the number of computational blocks increases. The compound model scaling technique is also employed to address the challenge of changing the in-degree of a translation layer in concatenation-based models. Furthermore, YOLOv7 implements additional improvements to optimize the network effectively. It adopts RepConvN in place of RepConv, removing identity connections to reparameterize residual and concatenation connections. Additionally, the model is enhanced with an auxiliary head structure to be trained under deep supervision [21]. These enhancements collectively contribute to the improved performance of YOLOv7 in real-time object detection tasks.

B. Lightweight Deep Learning

Despite the superior performance of deep neural networks, their heavy computing requirements pose challenges for real-world model deployment. To address this issue, numerous studies have focused on developing lightweight networks that offer efficiency while maintaining high performance [22, 23, 24]. Lightweight methods can generally be classified into four main strategies [25]; pruning ("*a lightweight deep learning method that eliminates unnecessary parameters in the neural network, reducing its size without compromising performance*") [26, 27], quantization ("*an effective technique involves storing weights, gradients, and activations with fewer bits, making the model more efficient during both training and inference stages*") [28], knowledge distillation ("*it focuses on transferring the learned knowledge from a large and accurate model (teacher model) to a smaller and more efficient model (student model), effectively compressing the knowledge*") [29], and neural architecture search ("*automatically explores optimized compact architectures to achieve model compression.*") [30].

C. Model Pruning

In the pursuit of optimizing neural networks, researchers have recognized that only a subset of parameters actively contributes to performing specific tasks. This realization has led to the development of various pruning methods aimed at reducing the number of parameters in neural networks [15, 16, 17]. Two prominent approaches in the pruning domain are filter pruning and weight pruning [25].Weight pruning, also known as unstructured pruning, involves eliminating individual parameters in a weight matrix [31, 32]. While weight pruning effectively reduces the model size, it often leads to sparse weight matrices, which can be inefficient in accelerating computation on hardware.

On the other hand, filter pruning, or structured pruning, targets the removal of entire filters from convolutional layers, based on their importance [33, 34]. This method determines the relevance of each filter and eliminates them as units from the network. Compared to weight pruning, filter pruning offers advantages in real-world scenarios, as it is more compatible with general frameworks and hardware, making it a preferred choice in many applications.

III. Methodology

In this study, we applied the channel pruning method to achieve the lightweight version of YOLOv7. We adopted the method [12] and made modifications to tailor it to the specific structure of YOLOv7.

A. Channel Pruning

The first main implication of the base method lies in its importance calculation for channel reduction, considering all components in the convolution module. Unlike general channel pruning methods that typically follow two steps (1) calculating the importance of each channel and (2) dropping channels with low importance based on a threshold, the base method distinguishes accurately whether channels will be dropped or retained by leveraging both the convolution kernel and the BN (Batch Normalization) scale factor. The calculation of importance (θ) is presented, based on Equation 1.

$$\theta_i = |\gamma_i| * \sum_{t \in \alpha_i} |t|, i \in [1, 2, ..., C_{out}] \quad (1)$$

- $\mathbf{C_{out}}$: Channel number of output feature map
- $\gamma_{\mathbf{i}}$: Scale factor of *i*th filter
- $\alpha_{\mathbf{i}}$: *i*th filter of convolution layer

The preservation or removal of each channel is determined through a comparison with the threshold. The importance threshold of each channel is set by multiplying the number of all channels with the pruning protect ratio for each layer. The pruning protect ratio represents the minimum proportion of remaining channels and can be set arbitrarily. For example, if a protection ratio of 0.3 is considered, it means that at least 30% of channels per layer should be retained.

The second implication involves rapid pruning achieved by combining a fine-tuning step with a sparsity training step. During training, the L1 regularization coefficient undergoes changes through cosine decay. A higher coefficient in the initial training stages can swiftly induce sparsity in the model. Any drop in accuracy during this stage is subsequently restored in the later training phases using a smaller coefficient, which also serves as a fine-tuning mechanism. Additionally, the sparsity process can be expedited using a soft mask strategy that assesses channel-level sparsity and intentionally reduces the scale factor of channels that are nearing sparsity during the training process. This approach effectively encourages channel sparsity and leads to faster convergence towards a compact and efficient model.

In the context of applying YOLOv7, we made two key modifications. Firstly, we revised the channel reduction target. While the previous approach for YOLOv5 only involved Conv modules consisting of convolution and BN layers, in YOLOv7, both Conv and RepConv layers were considered for channel pruning. We computed the channel importance within the Conv and RepConv modules, sorted them, and employed this information for pruning. For other modules, the original weights were retained and stored in a new model.

Secondly, we adjusted the iteration process. The original method traversed all layers in the model for various tasks, including importance calculation and storing original weights. However, this approach wasn't suitable for YOLOv7 due to structural and depth differences between YOLOv5 and YOLOv7. Consequently, we eliminated the process of recording previous weights for the SPPF and C3 modules, unique to YOLOv5. Instead, we incorporated a process akin to the SPPCSPC module, tailored for YOLOv7. Ultimately, we restructured the iteration order and inter-layer connectivity in each backbone and head component to align with these changes.

In this study, we employed a channel pruning method on an object detection model and proceeded to train a lightweight model using a dataset that was meticulously curated for human detection tasks. The primary aim was to ascertain whether this approach could effectively reduce the model's complexity while preserving object detection performance to a satisfactory degree. To validate our approach, we trained both the original model and the pruned model using the same dataset.

IV. Experiment

The experiments were carried out on an RTX A6000 D6 48GB GPU and the implementation was done in Python 3.7. Our experimentation involved varying the pruning protection ratio from 10% to 90% and subsequently deriving lightweight models through weight-based pruning. This was accomplished using a training dataset specifically designed for human detection tasks. To ensure a more precise comparison, factors such as image size, epoch count, and batch size were maintained identically across the experiments. We focused on comparing the number of parameters and computational complexity associated with each pruning protection ratio.

A. Dataset

The experiment was carried out using a lightweight model on the CrowdHuman dataset [35], which is designed for human detection. In this dataset, each image contains annotations for human objects, including both head bounding boxes and body bounding boxes. This dataset proves useful in scenarios where body parts might be obscured, as it enables the detection of heads or visible body regions. An illustration of the dataset is presented in Figure 1. The complete dataset was divided into training, validation, and testing sets, with 15,000 images for training, 4,370 images for validation, and 5,000 images for testing purposes.

Fig. 1: Examples of dataset

B. Result

Table I presents the results of pruning for each pruning protection ratio. The comparison of pruned models is based on three factors: the number of parameters, computational complexity (Flops), and the detection performance (mAP) [36].

Table I illustrates that as the protection ratio decreased, there was a corresponding reduction in the count of retained parameters and Flops. Nevertheless, when the protection ratio dropped below a certain threshold, the detection performance also experienced a rapid decline. Thus, it becomes crucial to determine a suitable pruning ratio that ensures both lightweight characteristics and the preservation of performance.

Pruning Protection Ratio	Layers	Parameters	Flops	mAP
100%(Original)	415	37,201,950	105.1	0.79
90%	415	32,307,876	94.6	0.767
80%	415	27,413,514	75.0	0.763
70%	415	23,142,702	61.9	0.76
60%	415	19,319,628	48.4	0.752
50%	415	15,474,254	35.8	0.74
40%	415	13,058,542	30.2	0.73
30%	415	10,904,020	22.2	0.724
20%	415	9,313,344	17.1	0.712
10%	415	8,238,590	14.4	0.674

TABLE I: Summary of the results

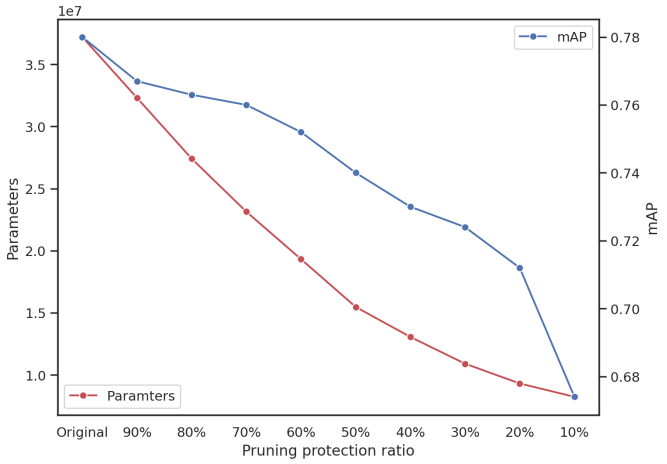

Fig. 2: Changes in parameters and mAP based on the pruning protection ratio.

Figure 2 illustrates the changes in the number of parameters and mAP with respect to the pruning protection ratio. Similarly, Figure 3 shows the alterations in Flops and mAP. Notably, when employing a protection ratio of 50%, we noted a relatively moderate decline in performance despite considerable reductions in parameters and computational workload. This observation suggests that such a pruning level is well-suited for tasks with lower complexity, like human detection.

V. Conclusion and Discussion

This study aimed to reduce the computational complexity of YOLOv7 through channel pruning, building upon a previous method [12]. We adapted the base method to suit the YOLOv7 architecture and assessed the viability of this approach for human detection tasks. As a result, we achieved a remarkable 66% reduction in GFlops and a 64% reduction in parameters using a 50% pruning protection ratio.

The primary contributions of this research lie in the creation of a lightweight YOLOv7 model based on established pruning techniques and the exploration of an object detection model optimized for human detection. Given its intricate structure, there has been relatively limited investigation into lightweight iterations of YOLOv7, one of the latest models in the YOLO series. Developing a lightweight version for YOLOv7 presents challenges, but this study offers an enhanced methodology that simplifies the process. By employing existing methodologies

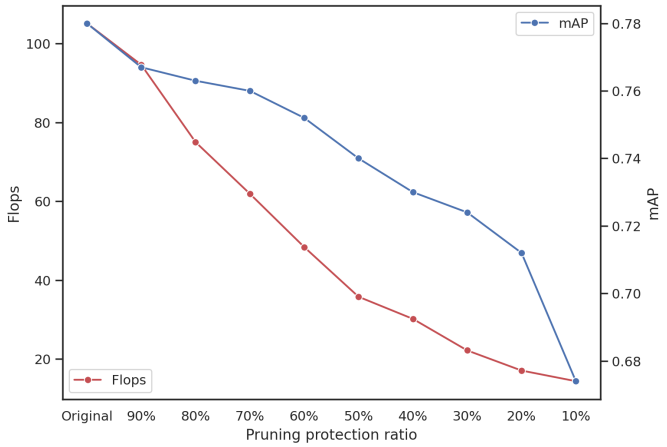

Fig. 3: Changes in Flops and mAP pruning protection ratio.

in a straightforward manner, we propose a practical solution to make YOLOv7 more lightweight. Therefore, this research not only showcases the academic potential of refining existing methods through an understanding of the latest models, but also validates the effectiveness of the pruning method for human detection tasks.

Models in the YOLO series are tailored for object detection and are commonly trained using the COCO dataset [37]. These object detection techniques find application in various scenarios, from identifying abnormal objects or situations to detecting specific humans or signs in real-life scenarios. Depending on the context, the complexity of object detection tasks can vary, and this study specifically concentrated on simpler tasks like human detection. Our research aimed to pinpoint the optimal lightweight level of YOLOv7 for practical use in real-world human detection scenarios. Consequently, we successfully determined an appropriate pruning ratio and achieved promising results for YOLOv7 using a human detection dataset.

In future work, we will evaluate the efficacy of the proposed lightweight YOLOv7 for the task of detecting workers within the context of a smart factory environment. On an academic front, we envision conducting subsequent investigations that involve leveraging real-world factory data for training purposes, aiming to enhance object detection performance, particularly concerning the higher Interaction Over Union (IOU) threshold.

References

[1] S. Lee, J. Kim, and E. Park, "Can book covers help predict bestsellers using machine learning approaches?" *Telematics and Informatics*, vol. 78, p. 101948, 2023.

[2] M. Lee, H. Ji, and E. Park, "Deepaup: A deep neural network framework for abnormal underground heat transport pipelines," *IEEE Transactions on Automation Science and Engineering*, 2023.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. of CVPR '01*, vol. 1. IEEE, 2001, pp. 511–518.

[4] D. Qi, W. Tan, Q. Yao, and J. Liu, "Yolo5face: Why reinventing a face detector," in *Proc. of ECCV Workshop '23*. Springer, 2023, pp. 228–244.

[5] A. Raghunandan, P. Raghav, H. R. Aradhya *et al.*, "Object detection algorithms for video surveillance applications," in *Proc. of ICCSP '18*. IEEE, 2018, pp. 0563–0568.

[6] X. Cheng, H. Xiong, D.-P. Fan, Y. Zhong, M. Harandi, T. Drummond, and Z. Ge, "Implicit motion handling for video camouflaged object detection," in *Proc. of CVPR '22*, 2022, pp. 13 864–13 873.

[7] J. Materzynska, T. Xiao, R. Herzig, H. Xu, X. Wang, and T. Darrell, "Something-else: Compositional action recognition with spatial-temporal interaction networks," in *Proc. of CVPR '20*, 2020, pp. 1049–1059.

[8] X. Wang, Y. Wu, L. Zhu, and Y. Yang, "Symbiotic attention with privileged information for egocentric action recognition," in *Proc. of AAAI '20*, vol. 34, no. 07, 2020, pp. 12 249–12 256.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of CVPR '16*, 2016, pp. 779–788.

[10] B. Ku, K. Kim, and J. Jeong, "Real-time isr-yolov4 based small object detection for safe shop floor in smart factories," *Electronics*, vol. 11, no. 15, p. 2348, 2022.

[11] L. Heda and P. Sahare, "Performance evaluation of yolov3, yolov4 and yolov5 for real-time human detection," in *2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS)*. IEEE, 2023, pp. 1–6.

[12] J. Zhang, P. Wang, Z. Zhao, and F. Su, "Pruned-yolo: Learning efficient object detector using model pruning," in *Proc. of ICANN '21*. Springer, 2021, pp. 34–45.

[13] J. Lee and K.-i. Hwang, "Yolo with adaptive frame control for real-time object detection applications," *Multimedia Tools and Applications*, vol. 81, no. 25, pp. 36 375–36 396, 2022.

[14] D. Jeong, S. Oh, and E. Park, "Demohash: Hashtag recommendation based on user demographic information," *Expert Systems with Applications*, vol. 210, p. 118375, 2022.

[15] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, and R. Schettini, "Automated pruning for deep neural network compression," in *Proc. of ICPR '18*. IEEE, 2018, pp. 657–664.

[16] L. Li, J. Zhu, and M.-T. Sun, "Deep learning based method for pruning deep neural networks," in *Proc. of ICMEW '19*. IEEE, 2019, pp. 312–317.

[17] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. of CVPR '19*, 2019, pp. 11 264–11 272.

[18] Y. Yin, W. Zhang, S. Hong, J. Yang, J. Xiong, and G. Gui, "Deep learning-aided ocr techniques for chinese uppercase characters in the application of internet of things," *IEEE Access*, vol. 7, pp. 47 043–47 049, 2019.

[19] M. Agarwal, S. Agarwal, L. Saba, G. L. Chabert, S. Gupta, A. Carriero, A. Pasche, P. Danna, A. Mehmedovic, G. Faa *et al.*, "Eight pruning deep learning models for low storage and high-speed covid-19 computed tomography lung segmentation and heatmap-based lesion localization: A multicenter study using covlias 2.0," *Computers in biology and medicine*, vol. 146, p. 105571, 2022.

[20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. of AISTATS '15*. PMLR, 2015, pp. 562–570.

[22] S. Kanimozhi, G. Gayathri, and T. Mala, "Multiple real-time object identification using single shot multi-box detection," in *Proc. of ICCIDS '19*, 2019, pp. 1–5.

[23] H. Zhao, Y. Zhou, L. Zhang, Y. Peng, X. Hu, H. Peng, and X. Cai, "Mixed yolov3-lite: A lightweight real-time object detection method," *Sensors*, vol. 20, p. 1861, 2020.

[24] X. Xu, W. Liang, J. Zhao, and H. Gao, "Tiny fcos: A lightweight anchor-free object detection algorithm for mobile scenarios," *Mobile Networks and Applications*, vol. 26, p. 2219–2229, 2021.

[25] C.-H. Wang, K.-Y. Huang, Y. Yao, J.-C. Chen, H.-H. Shuai, and W.-H. Cheng, "Lightweight deep learning: an overview," *IEEE Consumer Electronics Magazine*, 2022.

[26] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. of CVPR '17*, 2017, pp. 5687–5695.

[27] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[28] E. Fiesler, A. Choudry, and H. J. Caulfield, "Weight discretization paradigm for optical neural networks," in *Proc. of Optical interconnections and networks*, vol. 1281. SPIE, 1990, pp. 164–173.

[29] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proc. of CVPR '22*, 2022, pp. 11 953–11 962.

[30] W. Li, S. Wen, K. Shi, Y. Yang, and T. Huang, "Neural architecture search with a lightweight transformer for text-to-image synthesis," *IEEE Transactions on Network Science and Engineering*, vol. 9, pp. 1567–1576, 2022.

[31] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proc. of ECCV '18*, 2018, pp. 184–199.

[32] X. Jiang, N. Wang, J. Xin, X. Xia, X. Yang, and X. Gao, "Learning lightweight super-resolution networks with

weight pruning," *Neural Networks*, vol. 144, pp. 21–32, 2021.

[33] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. of ICCV '17*, 2017, pp. 5058–5066.

[34] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *Proc. of CVPR '20*, 2020, pp. 2009–2018.

[35] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, "Crowdhuman: A benchmark for detecting human in a crowd," *arXiv preprint arXiv:1805.00123*, 2018.

[36] H. Ji, C. An, M. Lee, J. Yang, and E. Park, "Fused deep neural networks for sustainable and computational management of heat-transfer pipeline diagnosis," *Developments in the Built Environment*, vol. 14, p. 100144, 2023.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. of ECCV '14*. Springer, 2014, pp. 740–755.