Iterative Pruning-based Model Compression for Pose Estimation on Resource-constrained Devices

Sunghyun Choi
Department of Computer Science and Engineering
Sungkyunkwan University Suwon, South Korea
cshyun23@gmail.com

Wonje Choi
Department of Computer Science and Engineering
Sungkyunkwan University Suwon, South Korea
wjchoi@g.skku.edu

Youngseok Lee
Department of Electrical and Computer Engineering
Sungkyunkwan University Suwon, South Korea
yslee.gs@skku.edu

Honguk Woo†
Department of Computer Science and Engineering
Sungkyunkwan University Suwon, South Korea
hwoo@skku.edu

ABSTRACT

In this work, we propose a pruning-based model compression scheme, aiming at achieving an efficient model that has strength in both accuracy and inference time on an embedded device environment with limited resources. The proposed scheme consists of (1) pruning profiling and (2) iterative pruning via knowledge distillation. With the scheme, we develop a resource-efficient 2D pose estimation model using HRNet and evaluate the model on NVIDA JetsonNano with the Microsoft COCO keypoint dataset. Specifically, our compressed model obtains the fast pose estimation of 20.3 *FPS* on NVIDA JetsonNano, while maintaining a high accuracy of 74.1 *AP*. Compared to the conventional HRNet model without compression, the proposed compression technique achieves 33 % improvement in *FPS* with only 0.4 % degradation in *AP*.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Machine learning approaches**;

KEYWORDS

Model compression, Pruning, Knowledge Distillation, Pose estimation, Embedded System inference

ACM Reference Format:
Sunghyun Choi, Wonje Choi, Youngseok Lee, and Honguk Woo†. 2022. Iterative Pruning-based Model Compression for Pose Estimation on Resource-constrained Devices. In *2022 the 5th International Conference on Machine Vision and Applications (ICMVA) (ICMVA 2022), February 18–20, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3523111.3523128

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICMVA 2022, February 18–20, 2022, Singapore, Singapore
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9567-0/22/02…$15.00
https://doi.org/10.1145/3523111.3523128

1 INTRODUCTION

Recently, various applications and systems have utilized deep learning methodologies as solutions, and deep learning for pose estimation is coming close to our daily life by commercialization, e.g., activity recognition and motion capture. Deep learning models for pose estimation generally require a lot of computing power for high accuracy. Thus, it is difficult for the models to be deployed in resource-constrained devices such as IoT, mobile, and embedded devices. To address the issue, several studies [1, 2] have investigated lightweight models for pose estimation in resource-constrained devices. However, these studies only focused on reducing the model size while rarely focusing on optimizing the performance of pose estimation on target devices.

In this paper, we present an iTerative pRuning-based modEl comprEssion with profiliNg (TREEN) scheme for 2D single human pose estimation, aiming to optimize not only inference time but also accuracy in resource-constrained environments. For achieving a lightweight model optimized on a target resource-constrained condition, we specifically explore the procedure of pruning profiling and iterative pruning with knowledge distillation (KD) methods.

1. The pruning profiling determines the backbone model by evaluating the trade-offs between *AP* and *FPS* in a given resource-constrained condition. It also determines two parameter settings required for the iterative pruning on the backbone model. The two parameters such as the iterative threshold and depth pruning priority confines the pruning space to a low complexity range, while performing iterative pruning.
2. Given the confined pruning space on the backbone model, the iterative pruning efficiently performs width- and depth-wise pruning to establish a lightweight yet high-performance model. Each pruning step leverages KD to minimize the performance degradation.

Through various experiments on the JetsonNano embedded board, we demonstrate that the TREEN scheme can generate a resource-efficient pose estimation model based on HRNet [3], showing that the model achieves the fast pose estimation of 20.3 *FPS* and maintains a high accuracy of 74.1 *AP*. Compared to the HRNet model without the TREEN's compression, our technique achieves 33 % improvement in *FPS* with only 0.4 % degradation in *AP*.

The rest of the paper is organized as follows. Section 2 and Section 3 present related works and the overall architecture of

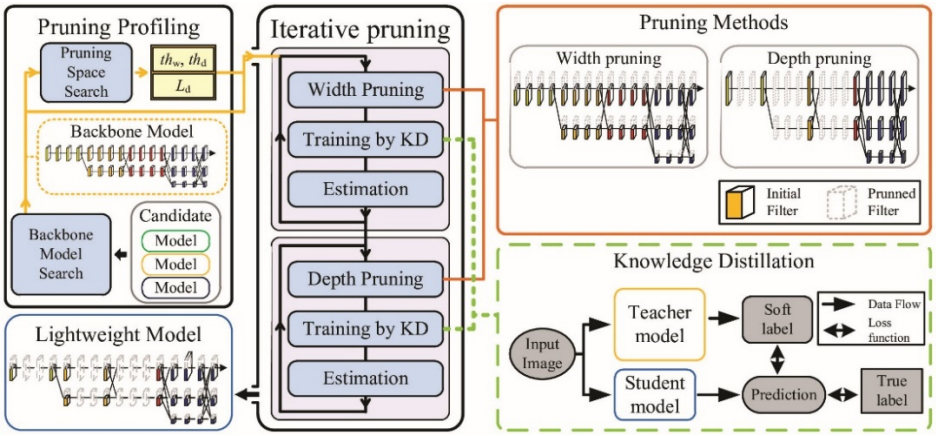


Figure 1: The model compression scheme TREEN with pruning profiling and iterative pruning

TREEN. The pruning profiling and iterative pruning modules in TREEN are detailed in Section 4 and 5, respectively. Then, Section 6 discusses the pose estimation performance by the TREEN-based model.

2 RELATED WORKS

To compress and accelerate a deep learning model, pruning is commonly used, which removes less important and redundant neurons from a neural network. Pruning can reduce the model size and shorten the inference time. As a large CNN model tends to be over-parameterized [4], pruning for a CNN-based pose estimation model is considered feasible in model compression. For pruning CNN models, structure pruning [5, 6] are used with two types such as filter pruning (width-wise) and layer-wise pruning (depth-wise). It has been also proven that iterative pruning [7, 8] is more effective than one-shot pruning for CNNs, as a pruned model can maintain accuracy through fine-tuning procedures.

KD in computer vision transforms teachers' features of images such as attention [9], activation [10, 11], or loss [12] into effective forms for students' learning. Xu et al. [13] used KD to compress existing 2D multi-person pose estimation models. Similarly, Wang et al. [14] and Hwang et al [15] used KD to generate efficient models for 3D pose estimation. Our work incorporates KD into iterative pruning on pose estimation models.

3 OVERALL ARCHITECTURE

In this section, we introduce our model compression scheme TREEN. Figure 1 shows the architecture of TREEN where the pruning profiling and iterative pruning modules are connected.

In the pruning profiling, a backbone model is first determined according to the designed backbone score evaluated among the backbone model candidates, given a target device resource condition. Then, the pruning space parameters (iterative thresholds ($th_{w,d}$) and depth pruning priority (L_d)) are determined according to the designed pruning score evaluated over pruned model candidates. Each pruning result is evaluated and scored according to the trade-off between accuracy and inference speed achieved by that pruning operation. The backbone model is then compressed through the iterative pruning procedure of width-wise (depth-wise) pruning, KD, and estimation.

1. Pruning: Two types of pruning performed sequentially with obtained pruning space parameters
2. KD: Transferring knowledge of the backbone model to the pruned model to minimize performance degradation
3. Estimation: Determining the optimal pruning iteration stage by calculating the pruning score

4 PRUNING PROFILING

In this section, we describe the pruning profiling with the backbone model and pruning space searches.

4.1 Backbone Model Search

To determine the most robust and high-performing model as our backbone model, we define and utilize the backbone score. Specifically, we consider the trade-off between accuracy and inference speed of a model on a given resource-constrained condition. The backbone score S_{bone} is defined as

$$S_{bone} = \alpha \times \left(FPS - \overline{FPS}\right) + (1-\alpha) \times \left(AP - \overline{AP}\right) \quad (1)$$

where FPS and AP denote inference time and accuracy on the condition, respectively. α denotes the control parameter for the trade-off between FPS and AP. In addition, $\overline{(\cdot)}$ denotes the average value of $(\cdot)$.

Table 1 presents the pose estimation performance of the back model candidates evaluated on a sever and a JetsonNano embedded board, where GFLOPs denotes the number of floating computations in model inference. The performance metric definitions are given in Section 6.1. We consider Hourglass [16], Simplebaselines [17], and HRNet [3] as our backbone model candidates. As expected, we observe that the large models generally obtain high accuracy and slow inference. Specifically, the HRNet-w32 model outperforms other models on JetsonNano, showing the highest AP value and fast FPS. Furthermore, regarding the difference between FPS_{base} (FPS on a Server) and FPS (FPS on JetsonNano), the HRNet models yield 66% in degradation while the Simplebaseline and Hourglass models yield 86 and 78%, respectively. The HRNet-w32 model yields the highest

Table 1: Performance of the backbone model candidates

Model	#Params	GFLOPs	FPS_{base}	FPS	AP	S_{bone}
Hourglass-1	3.6M	4.1	96.6	21.0	66.5	1.23
Hourglass-2	6.7M	6.3	66.7	15.1	71.2	0.63
Hourglass-4	13.0M	10.7	41.5	8.7	73.0	-1.67
Simplebaselines-18	15.4M	5.8	150.8	22.5	66.0	1.73
Simplebaselines-34	25.5M	7.5	118.8	18.8	69.4	1.58
Simplebaselines-50	34.0M	9.0	104.6	14.0	70.5	-0.27
Simplebaselines-101	53.0M	12.4	73.7	10.0	71.3	-1.87
Simplebaselines-152	68.6M	15.8	56.0	8.0	72.0	-2.52
HRNet-w32	28.5M	7.1	33.9	15.0	74.4	**2.18**
HRNet-w48	68.6M	14.6	33.5	8.0	75.0	-1.02

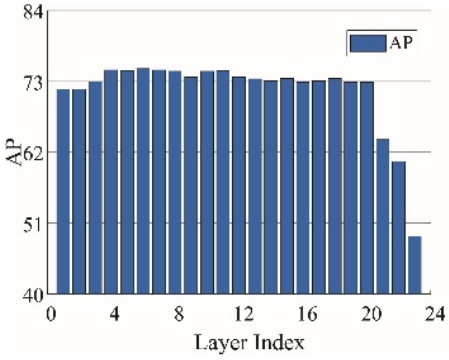


Figure 2: Accuracy by different pruned layers

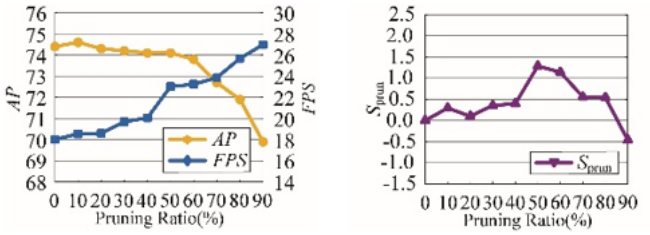
S_{bone} score (set to α =0.5). Therefore, we use the HRNet-w32 model as our backbone model.

4.2 Pruning Space Search

To determine the pruning space parameter values such as the depth pruning priority L_d and the iterative thresholds th_w, th_d, we evaluate both the impact of individual depth-wise pruning and (conventional) depth-wise and width-wise pruning. The width- and depth-wise pruning methods are explained in Section 5.1.

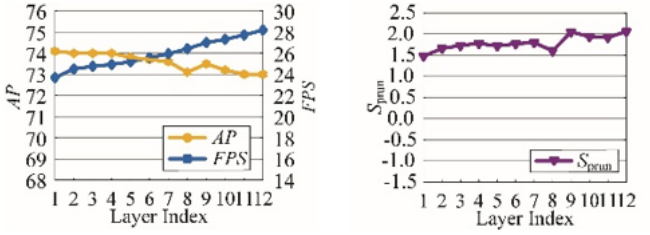
We first investigate which layers have less (or more) impact on the performance upon the depth-wise pruning. Figure 2 presents the accuracy of the backbone model, when the depth-wise pruning is adopted only for an individual layer. The x-axis denotes the location (index) of layers pruned. We observe that AP decreases significantly for the layer of larger indices, e.g., from 21. As such, we can prioritize the layers for depth-wise pruning in that removing the backward layers is more critical. This is consistent that the backbone model HRNet is structed with more parallel layers in backward than forward ones. Given this observation, we sort the layer indices in the order of being least influenced, and use them in Algorithm 1 as input, depth pruning priority L_d.

We also analyze the pruning space of the backbone model by depth and width side separately. Figure 3 presents the performance by each pruning, and the first row corresponds to the width-wise pruning space search and the other corresponds to the depth-wise



(a.1) Performance **(a.2) Pruning score**

(a) Width-wise pruning space search



(b.1) Performance **(b.2) Pruning score**

(b) Depth-wise pruning space search

Figure 3: Profiling results of pruning

pruning space search. We evaluate the performance using FPS, AP and S_{prun}. The metric S_{prun} is defined as

$$S_{prun} = \alpha \times (AP_{prun} - AP_{bone}) + (1 - \alpha) \times (FPS_{prun} - FPS_{bone}) \quad (2)$$

where $(\cdot)_{prun}$ and $(\cdot)_{bone}$ denote the performance of the pruned and backbone models, respectively. We implement Algorithm 1 that performs the width- and depth-wise pruning sequentially. In Figure 3 (a.1, b.1), we investigate the aspects of performance-complexity relation, and find that the width-wise pruning shows large fluctuation than the depth-wise pruning as the iteration continues. In Figure 3 (a.2, b.2), S_{prun} visualizes the trade-off, indicating the most appropriate setting for the target environment. In Figure 3 (a.2), S_{prun} is the highest at the ratio of 50% on the width side. In Figure

3 (b.2), the layer index 9 shows the highest. We set the iterative threshold th_w, th_d to the highest S_{prun} at each pruning side by using the result of these pruning space searches, expecting that Algorithm 1 can get higher S_{prun}at the iterative pruning process.

5 ITERATIVE PRUNING WITH KNOWLEDGE DISTILLATION

Algorithm 1 The procedure of TREEN

Models: M
Iterative thresholds th_w, th_d
List of priority layer indices L_d
/* From Profiling module */
th_w, th_d, $L_\mathrm{d} \leftarrow$ Profiling(M)
$i, j = 0$
/* Width-wise Pruning */
while $True$ do
 WidthPruning($M_{i,j}$)
Training(M , $M_{i,j}$)
$S_{prun} \leftarrow$ Estimation($M_{i,j}$)
 if $\Delta S_{prun} < 0$ and $S_\mathrm{prun} > th_w$ then
 break
 end if
 i += 1
end while
/* Depth-wise Pruning */
while $True$ do
 DepthPruning($M_{i,j}$, $L_\mathrm{d}[j]$)
Training(M , $M_{i,j}$)
$S_{prun} \leftarrow$ Estimation($M_{i,j}$)
 if $\Delta S_{prun} < 0$ and $S_\mathrm{prun} > th_d$ then
 break
 end if
 j += 1
end while
return M_(i,j)

5.1 Pruning

In the pruning stage of iterative pruning, the model weights are progressively removed in the width- or depth-wise aspects. $M_{i,j}$ represent the compressed model obtained by i-th width- and j-th depth-wise pruning iterations. The width-wise pruning removes nonessential filters, rendering an effect of achieving regular sparsity. At each iteration, the least nonessential filter is chosen according to the lowest L1 loss value. The L1 loss for the m-th filter in $M_{i,j}$ is computed by

$$\sum_{k=1}^{N_m^w} |W_{m,k}| \tag{3}$$

where $W_{m,k}$ denotes the k-th weight value in the m-th filter, and N_m^wdenotes the total number of weight values in the m-th filter. In our implementation, 10% of the whole filters are removed at each iteration.

The depth-wise pruning is a sort of layer-wise structure pruning that removes the whole layers in parallel. As the backbone model (HRNet) consists of parallel blocks with various sizes layers, it is less effective to apply general layer-wise pruning in terms of inference speed. Thus, we use the depth-wise pruning to remove a set of layers that are in parallel at j-th iteration. As the depth-wise pruning might cause unexpected accuracy degradation or structural disruption issues, we exploit the depth pruning priority L_d (obtained by the previous pruning profiling) to avoid such unpredictable results.

5.2 Fine-tuning via Knowledge Distillation

After the width- or depth-wise pruning, the weights of $M_{i,j}$ are updated by the previous iteration. According to the lottery ticket hypothesis [18], fine-tuning a pruned model converges to high accuracy with the initial weights of the original model. Thus, we initialize $M_{i,j}$ using the initial weights of the original model M. In our implementation, approximately 120 epochs are required for model convergence; that corresponds to 4 hours by our GPU server. Experimentally, we confirm the accuracy approaches close to the convergence after 30 epochs, so we normally conduct model training up to 30 epochs. We also adapt KD for fine-tuning to minimize degradation and earlier convergence for accuracy. Similar to [19], our loss function for this fine-tuning is defined as

$$\alpha \times l_{KD} + (1 - \alpha) \times l_{mse} \tag{4}$$

where l_{KD} denotes loss between our prediction and the teacher model's prediction, l_{mse} denotes loss between our prediction and the ground truth labels, and α is set to 0.5.

5.3 Estimation

We evaluate all $M_{i,j}$ after training using S_prun in Eq. 2) which takes account of *FPS* and *AP*. The iteration is continuously performed until the following termination condition is satisfied

$$\Delta S_{prun} < \ 0 \text{ and } S_{prun} > \ \text{th} \tag{5}$$

where ΔS_prun denotes the change of S_prun, and th is th_w or th_d.

6 EVALUATION

In this section, we evaluate the performance of TREEN-based models in terms of accuracy *AP* and inference time *FPS*, and analyze the effects of iterative pruning and KD.

6.1 Experiment Environment

We evaluate our models learned on the Microsoft COCO [20] keypoints 2017 dataset. We train the models on the train2017 set (57 K images), validated on the val2017 set (5 K images), and evaluated on the test-dev2017 set (20 K images). Here, we test the models on resource-sufficient (Server) and resource-constrained (JetsonNano) environments in Table 2. The computing power of the JetsonNano board is about seven times lower than that of the server in terms of the inference performance for ResNet-50.

For comparison, we use the following metrics: #Params (The number of model parameters), FPS_base (The inference speed on the server), *FPS* (The inference speed on JetsonNano), *AP* (The accuracy on JetsonNano). Note that *AP* is computed over 10 OKSs (object keypoints similarity in [21]) and AP^k represents the *AP* score for OKS larger than k. We set k = {0.50, 0.55, . . ., 0.95}.

Table 2: Test Environments

Environment	CPU	GPU	Memory
Server	Intel(R) Core(TM) i9-10940X	NVIDIA 2080Ti	32 G
JetsonNano	Quad-core ARM A57	128-core Maxwell	4 G

Table 3: Overall performance comparison (on JetsonNano with COCO test-dev2017)

		Server	JetsonNano						
		FPS_{base}	FPS	AP	AP^{50}	AP^{75}	AR	AP^{M}	AP^{L}
DSPNet	14.8M	47.0	7.6	66.8	88.4	74.1	70.1	64.1	71.2
LPN-50	2.9M	73.0	15.2	59.4	84.3	65.4	67.1	56.6	65.9
LPN-101	5.3M	42.8	13.5	69.5	88.2	77.5	75.4	66.2	76.1
LPN-152	7.4M	28.0	11.9	67.5	88.0	74.6	73.6	64.2	74.1
EfficientPose-A	1.3M	104.1	20.4	66.5	87.0	74.3	72.5	63.1	73.3
EfficientPose-B	3.3M	84.7	12.4	71.1	89.1	79.0	76.7	67.4	77.9
EfficientPose-C	5.0M	80.2	11.3	71.2	89.1	79.0	76.9	67.5	78.1
TREEN	15.0M	34.9	20.3	74.1	89.9	81.2	79.6	70.3	81.1

6.2 Models in Comparison

For comparison, we implement several models including DSPNet [22], LPN [23], and EfficientPose [24] for pose estimation, in addition to our model.

1. DSPNet: The EfficientNet [25] model is compressed by using separable transposed convolution, channel-wise attention, and attention mechanism. DSPNet is obtained by using the EfficientNet-B2 model as a backbone model.
2. LPN: The Simplebaseline [17] model is accelerated by using depth-wise convolution and attention mechanism. The models are categorized into LPN-x with respect to the backbone model ResNet-x of the Simplebaseline model, e.g., ResNet-50, ResNet-101, and ResNet-152.
3. EfficientPose: The MobileNet [26] model is compressed by the differentiable NAS (Neural Architecture Search) according to the pre-defined search space, i.e., EfficientPose-A, EfficientPose-B, and EfficientPose-C. The model parameter size increases from EfficientPose-A to EfficientPose-C.
4. TREEN: The HRNet [3] model with 32 channels is pruned by our TREEN scheme.

6.3 TREEN Performance

The performance is evaluated on JetsonNano after TensorRT [27] optimization is applied.

Table 3 represents the performance of TREEN and other models. The TREEN model on JetsonNano obtains a fastest inference of 20.3 *FPS* in most cases, and it achieves a higher accuracy of 4.1 ∼ 24.7 *AP* than others. The TREEN model also maintains the best accuracy of 70.3 AP^{M} and 81.1 AP^{L} regardless of the object size of input images. AP^{M} and AP^{L} denote *AP* over input images with the medium (1,024 9,216 pixels) and large (9,126 10^{10} pixels) objects, respectively. Furthermore, it shows the least drop of 14.3 *FPS* from server to JetsonNano while the EfficientPose-A model experiences the worst performance degradation of 83.7 *FPS* on JetsonNano. This is consistent with our expectation since our TREEN scheme utilizes the optimal backbone model and pruning space that are obtained by pruning profiling.

To evaluate the individual effect of iterative pruning and KD, we test the $\text{TREEN}_{\text{one-shot}}$ model that uses only one-shot pruning, and the TREEN_{KD} model uses KD and one-shot pruning. The TREEN_{KD} model shows a higher accuracy of 0.4 *AP* than the $\text{TREEN}_{\text{one-shot}}$ model, and the TREEN model shows a higher accuracy of 0.6 *AP* than the TREEN_{KD} model. As a result, the TREEN model has only 0.4 % accuracy degradation in comparison to the backbone HRNet model.

7 CONCLUSION

In this paper, we presented the iterative pruning-based model compression scheme and applied the scheme for pose estimation, considering resource-constrained device conditions where a real-time pose estimation application runs. Specifically, we developed the pruning profiling and iterative pruning; the pruning profiling determines the best backbone model and pruning space, and the iterative pruning effectively compresses the backbone model according to a given resource-constrained condition. We demonstrated that the compressed pose estimation model by our scheme achieved the fast inference of 20.3 *FPS* and high accuracy of 74.1 *AP* on NVIDA JetsonNano.

ACKNOWLEDGMENTS

This work was supported by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant 2021-0-00900 and the DNA+ Drone Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) under Grant 2020M3C1C2A01080819.

Table 4: The effect of iterative pruning and KD in TREEN

	In TREEN		Performance	
	Pruning	KD	*FPS*	*AP*
HRNet	-	False	15.0	74.4
TREENone-shot	One-shot	False	20.3	73.1
TREENKD	One-shot	True	20.3	73.5
TREEN	Iterative	True	20.3	74.1

REFERENCES

[1] Yamazaki Masayuki and Mori Eigo. "Rethinking Deconvolution for 2D Human Pose Estimation Light yet Accurate Model for Real-time Edge Computing". In: arXiv:2111.04226 (2021).

[2] Zhipeng Ma *et al.* "Multi-Person Pose Estimation on Embedded Device". In: Proceedings of International Conference on Intelligent Computing and Human Computer Interaction (ICHCI). Sanya, China, Dec. 2020, pp. 57–61.

[3] Ke Sun *et al.* "Deep High-Resolution Representation Learning for Human Pose Estimation". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, California, Jun. 2019, pp. 5693–5703.

[4] Denil Misha *et al.* "Predicting Parameters in Deep Learning". In: Proceedings of the 26th International Conference on Neural Information Processing Systems(NIPS). New York, New York, Dec. 2013, pp. 2148–2156.

[5] Vadim Lebedev and Victor Lempitsky. "Fast ConvNets Using Group-Wise Brain Damage". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, Nevada, Jun. 2016, pp. 2554–2564.

[6] Yang He *et al.* "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, California, Jun. 2019, pp. 1–10.

[7] Wenxiao Wang *et al.* "Accelerate CNNs from Three Dimensions: A Comprehensive Pruning Framework". In: Proceedings of the 38th International Conference on Machine Learning(PMLR). Vitual only, Jul. 2021, pp. 10717–10726.

[8] Tao Jiang *et al.* "Layer-wise Deep Neural Network Pruning via Iteratively Reweighted Optimization". In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing(ICASSP). Brighton, UK, May 2019, pp. 5606–5610.

[9] Sergey Zagoruyko and Nikos Komodakis. "Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer". In: Proceedings of the 5th International Conference on Learning Representations (ICLR). Toulon, France, Apr. 2017, pp. 1–13.

[10] Fida Mohammad Thoker and Juergen Gall. "Cross-Modal Knowledge Distillation for Action Recognition". In: Proceedings of IEEE International Conference on Image Processing (ICIP). Taipei, Taiwan, Sep. 2019, pp. 6–10.

[11] Yifan Liu *et al.* "Structured Knowledge Distillation for Semantic Segmentation". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, California, Jun. 2019, pp. 1–10.

[12] Kim Jangho, Park Seonguk, and Kwak Nojun. "Paraphrasing Complex Network: Network Compression via Factor Transfer". In: arXiv:1802.04977 (2018).

[13] Xixia Xu *et al.* "Integral Knowledge Distillation for Multi-Person Pose Estimation". In: IEEE Signal Processing Letters 27 (2020), pp. 436–440.

[14] Chaoyang Wang, Chen Kong, and Simon Lucey. "Distill Knowledge From NRSfM for Weakly Supervised 3D Pose Learning". In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). Seoul, Korea, Oct. 2019, pp. 743–752.

[15] Dong-Hyun Hwang *et al.* "Lightweight 3D Human Pose Estimation Network Training Using Teacher-Student Learning". In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). Aspen, Colorado, Mar. 2020, pp. 1–10.

[16] A. Newell, K. Yang, and J. Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: Proceedings of the European Conference on Computer Vision (ECCV). Amsterdam, Netherlands, Oct. 2016, pp. 483–499.

[17] Bin Xiao, Haiping Wu, and Yichen Wei. "Simple Baselines for Human Pose Estimation and Tracking". In: Proceedings of the European Conference on Computer Vision (ECCV). Munich, Germany, Sep. 2018, pp. 466–481.

[18] Frankle Jonathan and Carbin Michael. "The Lottery Ticket Hypothesis: Training Pruned Neural Networks". In: Proceedings of the 7th International Conference on Learning Representations (ICLR). New Orleans, Louisiana, May 2019, pp. 1–42.

[19] Feng Zhang, Xiatian Zhu, and Mao Ye. "Fast Human Pose Estimation". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, California, Jun. 2019, pp. 1–10.

[20] in Tsung-Yi *et al.* "Microsoft COCO: Common Objects in Context". In: Proceedings of European Conference on Computer Vision (ECCV). Zurich, Switzerland, Sep. 2014, pp. 740–755.

[21] Matteo Ruggero Ronchi and Pietro Perona. "Benchmarking and Error Diagnosis in Multi-Instance Pose Estimation". In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). Venice, Italy, Oct. 2017, pp. 1–10.

[22] Zhong Fujin *et al.* "DSPNet: A low computational-cost network for human pose estimation". In: Neurocomputing 423 (2021), pp. 327–335.

[23] Wenqiang Zhang, Jiemin Fang, and Gangshan Wu. "Simple and Lightweight Human Pose Estimation". In: arXiv:1911.10346 (2019).

[24] Zhang Wenqiang *et al.* "Efficient human pose estimation with neural architecture search". In: Computational Visual Media 7 (2021), pp. 335–347.

[25] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: Proceedings of International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI). Long Beach, CA, Jun. 2019, pp. 6105–6114.

[26] Mark Sandler *et al.* "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR). Salt Lake City, Utah, Jun. 2018, pp. 4510–4520.

[27] Nvidia TensorRT. [Online]. Available: https://developer.nvidia.com/tensorrt. Accessed on: Jan. 5, 2022