DATA-FREE BACKBONE FINE-TUNING FOR PRUNED NEURAL NETWORKS

Adrian Holzbock*, Achyut Hegde†, Klaus Dietmayer* and Vasileios Belagiannis‡
*Institute for Measurement-, Control-, and Microtechnology
Ulm University, Germany
adrian.holzbock@uni-ulm.de, klaus.dietmayer@uni-ulm.de
†Institute of Information Security and Dependability
Karlsruhe Institute of Technology, Germany
achyut.hegde@kit.edu
‡Chair of Multimedia Communications and Signal Processing
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
vasileios.belagiannis@fau.de

***Abstract*—Model compression techniques reduce the computational load and memory consumption of deep neural networks. After the compression operation, e.g. parameter pruning, the model is normally fine-tuned on the original training dataset to recover from the performance drop caused by compression. However, the training data is not always available due to privacy issues or other factors. In this work, we present a data-free fine-tuning approach for pruning the backbone of deep neural networks. In particular, the pruned network backbone is trained with synthetically generated images, and our proposed intermediate supervision to mimic the unpruned backbone's output feature map. Afterwards, the pruned backbone can be combined with the original network head to make predictions. We generate synthetic images by back-propagating gradients to noise images while relying on L1-pruning for the backbone pruning. In our experiments, we show that our approach is task-independent due to pruning only the backbone. By evaluating our approach on 2D human pose estimation, object detection, and image classification, we demonstrate promising performance compared to the unpruned model. Our code is available at https://github.com/holzbock/dfbf.**

***Index Terms*—Data-Free Neural Network Pruning, Data-Free Object Detection Pruning, Data-Free Pose Estimation Pruning**

I. Introduction

Computer vision tasks like object detection [1], [2] or human pose estimation [3], [4] have long been studied in the literature. Their leading performance though comes at the cost of high computational and memory requirements, making them difficult to deploy on resource-constrained devices. For deep neural networks, the compute and memory demands are often reduced with model compression, e.g. parameter pruning [5] or knowledge distillation [6]. However, most neural network compression techniques normally require the availability of the training set. Due to privacy issues or large dataset size, access to the training dataset cannot always be guaranteed.

As an alternative to the data-driven model compression [5]–[7], approaches using only a few samples of the training dataset [8] and data-free methods have been developed to overcome the limited access to the original training dataset. The data-free approaches generate synthetic images to regain the knowledge after pruning [9] or transfer the knowledge with knowledge distillation [10] to a smaller model. The synthetic images can be generated with an additional generator network trained with the original model's knowledge [9], [11]. Another method is to optimize noisy images by back-propagating directly onto the pixels [12]. In this work, we focus on neural network pruning similar to [9]. However, we rely on the back-propagation of gradients for image generation instead of employing a generator network. In contrast to existing methods, our approach is not limited to a specific task, but is applicable to different computer vision tasks. We achieve task-independence by only pruning the backbone of the model, keeping the task-dependent head unchanged.

Part of the work was supported by INTUITIVER (7547.223-3/4/), funded by the State Ministry of Baden-Württemberg for Sciences, Research and Arts and the State Ministry of Transport Baden-Württemberg.

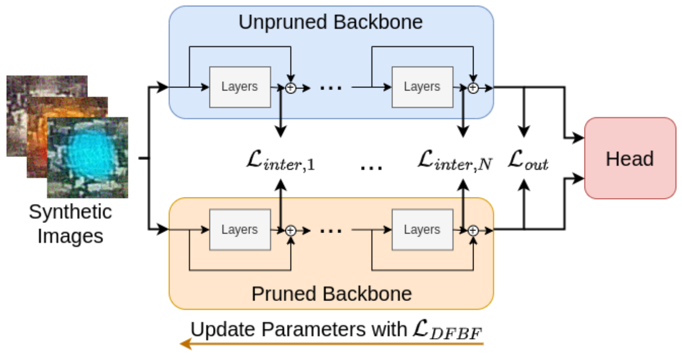

Fig. 1. Training of the pruned backbone with label-free synthetic images. Besides the backbone's output feature map, some intermediate feature maps are used to recover the backbone's knowledge.

We present a **D**ata-**F**ree **B**ackbone **F**ine-tuning approach (DFBF) for pruning the backbone of deep neural networks. In particular, we train the pruned backbone with synthetic images to mimic the output of the unpruned backbone. Later, we can use the pruned backbone with the original network head to make predictions. The synthetic images are generated by back-propagating gradients directly on noise images, while the backbone is pruned by ℓ_1-pruning, leaving the parameters of the network head unmodified. Only backbone pruning keeps the pruning approach task-independent, as no training of the task-specific head is required. For the training of the backbone, we introduce an intermediate loss function that adjusts the pruned backbone's intermediate feature maps and output to match it with the original backbone. An overview of the proposed method is given in Fig. 1. After the training, the original network's head can be attached to the pruned backbone for the task-specific output.

We examine our approach on three vision tasks, namely object detection, human pose estimation, and image classification. First, we use the Faster R-CNN object detector [1] for object detection and present the results for different pruning ratios on the COCO detection dataset [13] and Pascal VOC dataset [14]. We evaluate pose estimation with the OpenPifPaf pose estimator [3] trained on the COCO keypoint dataset [13]. Additionally, we compare the performance on CIFAR10 [15] for image classification. As backbone, we use ResNet [16] and VGG [17] models in our experiments. To the best of our knowledge, we are the first to introduce a data-free fine-tuning method for pruning that can be applied to complex computer vision tasks like object detection and human pose estimation. The proposed intermediate loss function that applies the prediction of the unpruned model as ground truth enables the task independence of DFBF.

II. Method

In the following, we present our data-free fine-tuning method for pruned neural networks, which process input images $\mathbf{x} \in \mathbb{R}^{3\times W\times H}$, where W and H are the width and height, respectively. We assume that the neural network $\mathbf{y} = f(\mathbf{x};\theta)$ can be divided into a backbone $\mathbf{z} = b(\mathbf{x};\theta_b)$ and a head $\mathbf{y} = h(\mathbf{z};\theta_h)$, where the head h is a task-specific output network. $\mathbf{y}$ is the output prediction, $\mathbf{z}$ is the output feature map from the backbone forwarded to the task-specific head, while θ, θ_b, and θ_h are the model parameters of the whole model, the backbone, and the head, respectively. Since we only prune the model backbone, our approach is independent of the shape of the model prediction $\mathbf{y}$ and thus can be applied to different computer vision tasks.

A. Preliminaries

Pruning: Utilizing pruning algorithms reduces neural networks' resource demand by removing unnecessary or redundant parameters. Pruning convolutional neural networks can be divided into different groups. The parameter pruning only removes single weights [18], leading to sparse kernels and no satisfying reduction in computation. In contrast, filter pruning reduces the computation effort by removing entire filters [19]. To decide which filters are pruned, the methods rely on the knowledge of the model, like in ℓ_1-pruning [20] and Batch Normalization pruning [21], or even use additional neural network layers [22]. Because of its simplicity and the fact that no extra neural network layers are needed, we rely on the ℓ_1-pruning [20] to get the pruned backbone $b_p(\cdot)$ with its reduced parameters θ_{bp}. However, every other filter pruning approach can be used to prune the model's backbone for DFBF. The ℓ_1-pruning calculates for each filter $\mathcal{F}_i$ of a neural network layer the sum of the filter weights w_i by $s_i = \sum |w_i|$ and removes filters with the smallest sum s_i according to the required sparsity. The reduction of the current layer's filters makes it necessary to remove the corresponding input channels of the following layer. Furthermore, the pruning sparsity can be adapted to the number of filters in the layer, i.e. in layers with many filters, percentual more filters are pruned than in layers with fewer filters.

Image Synthesis: Since the original training set is unavailable, we synthesize training images by transferring knowledge from the model to a noisy image, similar to DeepInversion [23], but without being limited to the classification task. At the beginning of the image generation, a noise image $\hat{\mathbf{x}} \in \mathcal{R}^{3\times w\times h}$ with width w and height h is fed into the backbone, while the synthetic image size can differ from the original $(W \neq w; H \neq h)$. Then, the loss $\mathcal{L}_{image}$ is propagated back onto the noise image $\hat{\mathbf{x}}$ to adapt the pixels directly without changing the model parameters. The loss $\mathcal{L}_{image}$ for optimizing the noise image independent of the underlying task is the weighted sum of the following three parts. The Batch Normalization loss is calculated between the batch statistics of the Batch Normalization layers and the statistics of the actual noise images. The total variance loss is defined by the ℓ_2-norm of the differences between horizontally and vertically adjacent image pixels and is calculated on the noise image directly. Additionally, the loss is regularized by penalizing the ℓ_2-norm of the entire image. The task independence of $\mathcal{L}_{image}$ is reached by not using the task-specific outputs of the head, whereby images of different computer vision tasks can be synthesized. To train the pruned model, we generate a synthetic dataset $\mathcal{D}$ containing M synthetic images $\hat{\mathbf{x}}$.

B. Overall Pruning Procedure

Our method is designed for pruning a neural network of any image-dependent task in a data-free manner. Our main contribution is the fine-tuning step after the pruning without relying on training data. Therefore, we focus on reducing the size of the backbone $b(\cdot)$ and keeping the head $h(\cdot)$ as is. We perform the task-independent data-free pruning in the following three steps: 1) image generation with an adapted loss function, 2) network pruning with the ℓ_1-pruning, and 3) fine-tuning only the backbone with the proposed intermediate loss function. A systematic overview of the data-free fine-tuning of the backbone is given in Fig.1.

C. Data-Free Training

During the pruning process, the model loses some of its knowledge which can be recovered in the following training procedure. In data-driven pruning methods, fine-tuning is performed with the original training dataset. In contrast, we use the synthetic dataset $\mathcal{D}$, which only contains pseudo images $\hat{\mathbf{x}}$, but no labels. However, the loss calculation in the standard training with the original data needs labels for the parameter update. To overcome the issue of having no ground truth labels for the pseudo images, we introduce a method that generates pseudo labels with the help of the unpruned model and applies them to improve the performance of the pruned model.

The final output of a neural network depends on the defined model task. Therefore, we propose not handling the output of the unpruned network's head as pseudo ground truth but instead using the output feature map $\mathbf{z}$ of the unpruned model's backbone. This design choice makes our approach independent of the task-specific model heads. More precisely, we define the loss as the ℓ_1-loss between the output feature map of the unpruned backbone $\mathbf{z}$ and the pruned backbone $\hat{\mathbf{z}}$. Our output feature map loss $\mathcal{L}_{out}$ can be formulated as

$$\mathcal{L}_{out} = \frac{1}{w_o * h_o} \sum_{i=0}^{w_o} \sum_{j=0}^{h_o} |\hat{\mathbf{z}}_{i,j} - \mathbf{z}_{i,j}|, \tag{1}$$

where w_o and h_o are the width and height of the output feature map, respectively.

Additionally, we define an intermediate feature map loss $\mathcal{L}_{inter}$. For $\mathcal{L}_{inter}$, we calculate the ℓ_1-loss between intermediate feature maps of the unpruned model $\mathbf{a}_n, n \in [1 \dots N]$ and the corresponding feature maps of the pruned model $\hat{\mathbf{a}}_n, n \in [1 \dots N]$, where N defines the number of intermediate feature maps. Empirically, we found that the intermediate feature maps after the Batch Normalization layers [24] lead to the best results. Since the feature maps behind the pruned layers differ in their dimensions from the unpruned model ones, it is impossible to calculate the loss between them. Therefore, we propose to skip single layers in the pruning and use them later for the loss calculation. Using N intermediate feature maps for the loss calculation, $\mathcal{L}_{inter}$ can be calculated as

$$\mathcal{L}_{inter} = \sum_{n=1}^{N} \left(\frac{\mu_n}{w_n * h_n} \sum_{i=0}^{w_n} \sum_{j=0}^{h_n} |\hat{\mathbf{a}}_{n,i,j} - \mathbf{a}_{n,i,j}| \right). \tag{2}$$

The influence of the different feature maps on the loss $\mathcal{L}_{inter}$ is defined by the parameter μ_n, which we define as $\frac{n}{N+1} * \gamma + 1$, where the parameter γ is a scaling factor. w_n and h_n are the width and height of the intermediate feature map n.

The overall loss $\mathcal{L}_{DFBF}$ in our data-free training combines $\mathcal{L}_{out}$ and $\mathcal{L}_{inter}$:

$$\mathcal{L}_{DFBF} = \mu_{out} \mathcal{L}_{out} + \mathcal{L}_{inter}, \tag{3}$$

where μ_{out} is the weighting factor of the output loss that we define as $\gamma + 1$. Importantly, $\mathcal{L}_{DFBF}$ is applied to optimize the parameters θ_{bp} of the pruned backbone $b_p(\cdot)$ and not to update the parameters θ_h of the head $h(\cdot)$.

During inference, we combine the optimized pruned backbone $\hat{\mathbf{z}} = b_p(\mathbf{x}, \theta_{bp})$ with the pre-trained head $\hat{\mathbf{y}} = h(\hat{\mathbf{z}}, \theta_h)$ to obtain the pruned model as:

$$\hat{\mathbf{y}} = h(b_p(\mathbf{x}, \theta_{bp}), \theta_h). \tag{4}$$

For the prediction, we feed original images $\mathbf{x}$ to the model and get a slightly different model output $\hat{\mathbf{y}}$ because of the modified backbone weights θ_p. Due to the backbone training with the synthetic images, the output of the original model $\mathbf{y}$ and the pruned model $\hat{\mathbf{y}}$ are nearly identical.

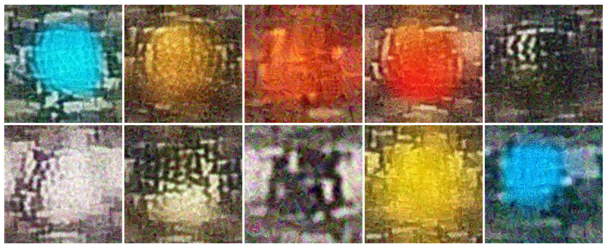
Fig. 2. Synthetic images generated with Faster R-CNN object detector [1] with a ResNet50 [16] backbone trained on the COCO detection dataset [13].

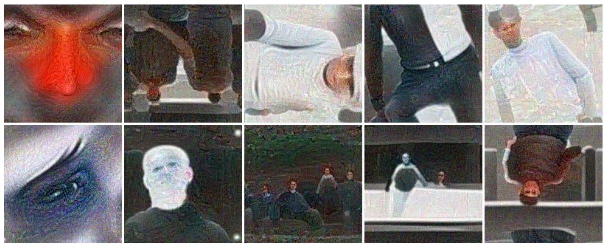
Fig. 3. Synthetic images generated with an OpenPifPaf pose estimator [3] with a ResNet50 [16] backbone trained on the COCO keypoint dataset [13].

III. Experiments

We show the effectiveness of our proposed method in three challenging computer vision tasks: object detection, human pose estimation, and image classification. In object detection, we use Faster R-CNN [1] trained on the COCO detection [13] as well as on the Pascal VOC dataset [14]. The base for the human pose estimation is an Open PifPaf pose estimator [3] trained on the COCO keypoint dataset [13]. Both tasks utilize a ResNet50 [16] backbone, and VGG16 [17] is used as an additional backbone in object detection. We prune between 10% and 40% of the backbone filters with ℓ_1-pruning during the evaluation. In image classification, we compare with Tang et al. [9] and follow their evaluation protocol using Batch Normalization pruning [21] for the VGG models [17] and ℓ_1 pruning [20] for the ResNet models [16]. The pruned backbone is trained in each task with 1600 synthetic images using the SGD optimizer with a learning rate of 0.01, a momentum of 0.9, and a weight decay of 5×10^{-4}.

To the best of our knowledge, we are the first to prune an object detection or a human pose estimation model in a data-free manner. Therefore, we cannot make a direct comparison with other methods. We compare DFBF with the unpruned baseline referred to as *Baseline*, the pruned but not fine-tuned model referred to as *w/o fine-tuning*, and the pruned model trained with 1600 random sampled original training images and our intermediate loss $\mathcal{L}_{DFBF}$ referred to as *orig. img.* Additionally, we report the number of removed filters and parameters, while the parameter count differs from the filter count because of different pruning sparsities in separate layers.

TABLE I

RESULTS OF DFBF FOR OBJECT DETECTION WHEN PRUNING FASTER R-CNN [1] WITH RESNET50 [16] AND VGG16 [17] BACKBONE TRAINED ON COCO DETECTION [13] AND PASCAL VOC [14]. OPENPIFPAF [3] WITH A RESNET50 [16] BACKBONE TRAINED ON COCO KEYPOINTS [13] FOR POSE ESTIMATION. *w/o fine-tuning*: WITHOUT FINE-TUNING AFTER PRUNING; *orig. img*: ORIGINAL IMAGES AND $\mathcal{L}_{DFBF}$ LOSS.

Method	Removed Params in %	Removed Filters in %	COCO detection mAP$^{0.5:0.95}$ in %	COCO detection mAP$^{0.5:0.95}$ in %	VOC mAP$^{0.5}$ in %	COCO keypoints AP in %
Backbone	ResNet50/ VGG16	ResNet50/ VGG16	ResNet50	VGG16	ResNet50	ResNet50
Baseline	0/0	0	37.4	23.0	78.1	68.1
w/o fine-tuning	14/16	10	34.0	19.4	74.7	57.9
orig. img	14/16	10	35.8	21.3	77.7	63.3
DFBF	14/16	10	36.2	20.9	77.6	65.2
w/o fine-tuning	28/32	20	24.6	11.6	60.3	36.1
orig. img	28/32	20	36.3	20.0	77.4	60.2
DFBF	28/32	20	33.4	18.8	75.1	62.2
w/o fine-tuning	40/46	30	13.8	5.3	36.9	20.4
orig. img	40/46	30	35.3	18.1	76.0	55.5
DFBF	40/46	30	28.0	15.2	71.5	55.5
w/o fine-tuning	52/59	40	4.6	2.2	20.7	4.6
orig. img	52/59	40	33.1	14.1	74.0	48.8
DFBF	52/59	40	19.1	10.7	63.2	49.1

A. *Object Detection*

We set the synthetic image resolution for the object detection task to 250×250 pixels and γ to 1. The results for pruning a Faster R-CNN object detection model [1] in a data-free manner are shown in Tab. I. For both datasets and backbones, the performance evolves similarly during pruning and training. The performance of the object detector after pruning depends on the pruning sparsity and decreases with higher rates. Also, we can see that with DFBF, the performance of the pruned model can recover during training with the synthetic images near the initial results. The model's performance trained with synthetic images is behind the model trained with the original images. Furthermore, the influence of synthetic images at higher pruning rates (30% and 40%) can be seen. Here, the difference between the model trained with the synthetic and the original images increases compared to lower pruning rates (10% and 20%). We show some synthetic images generated from an object detector trained on the COCO detection dataset [13] in Fig. 2. Compared to the original images, no objects can be recognized in the synthetic images. The abstract look of the synthetic images can cause the performance gap between the synthetic and original training image's performance using high pruning sparsities.

B. *Human Pose Estimation*

Besides object detection, we evaluate DFBF for human pose estimation which is often the base for gesture recognition [25] or human motion prediction [26]. We use synthetic images with a resolution of 160×160 and 6 as value for γ. The results for the human pose estimation are shown in Tab. I. As for object detection, the performance decreases with a higher pruning rate in human pose estimation. For all pruning rates, DFBF can recover the performance after pruning. In contrast to object detection, we perform better on human pose estimation with the synthetic images than with the original training images. This could be due to the difference between the synthetic images obtained from both tasks. Synthetic images generated with OpenPifPaf trained on the COCO keypoint dataset can be seen in Fig. 3. Compared to the images generated in the object detection task in Fig. 2, the pose images show pose information and pose-related details such as faces. Therefore, the generated images' variation can be broader compared to the original images. Moreover, the synthetic images show only patterns important for human pose estimation, and the pruned model can concentrate on learning the essential shapes.

C. *Image Classification*

Unlike the other tasks, we compare image classification with the state-of-the-art approach from Tang et al. [9]. During pruning the image classification model, the resolution of the synthetic training images is 32×32, and we set the factor γ to 0. Tab. II presents the CIFAR10 [15] results for different ResNet and VGG models fine-tuned with DFBF and the approach of Tang et al. [9]. In the pruning, we remove 30% and 50% of the filters from the baseline models. The results show that our approach is on par with Tang et al. [9], while we are not limited to the classification task due to the proposed task-agnostic loss function.

IV. ABLATION STUDIES

An essential part of DFBF is the proposed intermediate loss function $\mathcal{L}_{DFBF}$ for which we do further investigations. We use the OpenPifPaf settings from Sec. III as standard settings.

TABLE II
COMPARISON OF OUR METHOD WITH TANG ET AL. [9] ON THE CIFAR10 [15] DATASET WITH VGG16/19 [17] AND RESNET18/34 [16]. MODEL ACCURACY AND REMOVED (RM) FILTERS ARE GIVEN IN %.

Method	RM Filters	VGG 16	VGG 19	ResNet 18	ResNet 34
Baseline	0	94.00	93.95	93.07	93.33
DFBF	30	92.85	92.85	92.66	92.92
Tang et al.	30	92.78	92.82	-	-
DFBF	50	92.07	92.14	92.48	92.60
Tang et al.	50	92.43	92.78	92.68	93.25

TABLE III
RESULTS OF DIFFERENT SCALING FACTORS γ IN $\mathcal{L}_{DFBF}$.

γ	2	3	4	5	6	7	8
AP	60.0	60.8	61.4	61.8	62.2	62.4	62.6

In $\mathcal{L}_{DFBF}$, we take the feature map after each residual connection and the output feature map to calculate the loss and get an overall performance of 62.2%. The performance drops by 1.0% AP to 61.2% AP when skipping every second intermediate feature map in the loss calculation. Neglecting all intermediate feature maps and using only the output feature map in the loss calculation decreases the performance to 44.7% AP. These experiments demonstrate the importance of the intermediate feature maps on $\mathcal{L}_{DFBF}$. Furthermore, we present the effect of the scaling factor γ in Tab. III. We vary the impact of the different feature maps on $\mathcal{L}_{DFBF}$ by setting $\gamma \in [2, \ldots, 8]$. With increasing γ, the overall performance improves.

V. CONCLUSION

We presented a data-free backbone fine-tuning approach for pruning the backbone of deep neural networks. Our approach relies on synthetically generated images to fine-tune the backbone of the pruned neural network, where pruning is based on the ℓ_1 norm. Notably, we proposed an intermediate loss function to match the pruned backbone's output feature map such that the pruned backbone can later be combined with the original network head to perform predictions. Our evaluations showed that our approach is task-independent by evaluating the tasks of object detection, human pose estimation, and image classification.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[2] Y. Chen, F. Liu, and K. Pei, "Monocular vehicle 3d bounding box estimation using homograhy and geometry in traffic scene," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 1995–1999.

[3] S. Kreiss, L. Bertoni, and A. Alahi, "Openpifpaf: Composite fields for semantic keypoint detection and spatio-temporal association," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[4] V. Belagiannis, C. Amann, N. Navab, and S. Ilic, "Holistic human pose estimation with regression forests," in *AMDO: 8th International Conference, AMDO 2014, Palma de Mallorca, Spain, July 16-18, 2014. Proceedings 8*, 2014, pp. 20–30.

[5] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *Proceedings of machine learning and systems*, vol. 2, pp. 129–146, 2020.

[6] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[7] L. B. Letaifa and J.-L. Rouas, "Transformer model compression for end-to-end speech recognition on mobile devices," in *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE, 2022, pp. 439–443.

[8] Z. Zhou, Y. Zhou, Z. Jiang, A. Men, and H. Wang, "An efficient method for model pruning using knowledge distillation with few samples," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 2515–2519.

[9] J. Tang, M. Liu, N. Jiang, H. Cai, W. Yu, and J. Zhou, "Data-free network pruning for model compression," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[10] Y. Choi, J. Choi, M. El-Khamy, and J. Lee, "Data-free network quantization with adversarial knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 710–711.

[11] H. Chen *et al.*, "Data-free learning of student networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3514–3522.

[12] Y. Li *et al.*, "Mixmix: All you need for data-free compression are feature and data mixing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4410–4419.

[13] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014, pp. 740–755.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[15] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, USA, May 7-9, 2015.*

[18] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15, 2015, p. 1135–1143.

[19] H. Zhang, L. Liu, H. Zhou, W. Hou, H. Sun, and N. Zheng, "Akecp: Adaptive knowledge extraction from feature maps for fast and efficient channel pruning," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 648–657.

[20] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[21] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[22] H. Xu *et al.*, "Data agnostic filter gating for efficient deep networks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3503–3507.

[23] H. Yin *et al.*, "Dreaming to distill: Data-free knowledge transfer via deepinversion," in *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[25] A. Holzbock, A. Tsaregorodtsev, Y. Dawoud, K. Dietmayer, and V. Belagiannis, "A spatio-temporal multilayer perceptron for gesture recognition," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1099–1106.

[26] A. Bouazizi, A. Holzbock, U. Kressel, K. Dietmayer, and V. Belagiannis, "Motionmixer: Mlp-based 3d human body pose forecasting," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022, pp. 791–798.