2021 The 21st International Conference on Control, Automation and Systems (ICCAS 2021)
Ramada Plaza Hotel, Jeju, Korea, Oct. 12∼15, 2021

Model Diet: A Simple yet Effective Model Compression for Vision Tasks

Jongmin Lee[1*], Armagan Elibol[2], Nak Young Chong[3*]

School of Information Science, Japan Advanced Institute of Science and Technology
Nomi, Ishikawa 923-1292, Japan

([1]myeddie@jaist.ac.jp, [2]aelibol@jaist.ac.jp, [3]nakyoung@jaist.ac.jp) * Corresponding authors

Abstract: Computer vision coupled with machine learning algorithms has greatly helped mobile robotic platforms become more intelligent and capable of performing in the real world. Specifically, Convolutional Neural Networks (CNNs) have achieved a high accuracy on a range of visual perception tasks (*e.g.*, object detection, classification, segmentation, and similar others). One of the bottlenecks in CNNs is their high computational requirement. This makes most of them not easily deployable on robotic platforms, since their on-board computational power is limited. Recently, *Involution* successfully reduced the number of parameters of CNNs by replacing all the 3×3 convolution kernels with involution kernels, which use 1×1 convolution for the kernel generation. Filter pruning methods have also successively reduced the number of parameters in CNNs. Notably, however, Involution has reshaping layers and the kernel size is unknown when loading the pre-trained model. In this paper, we propose a pruning method named *Model Diet* that can be applied to Involution and other CNNs. We present experimental results showing that it has better results compared with randomly initialized weights.

Keywords: computer vision, deep neural networks, involution, filter pruning, model compression

1. INTRODUCTION

Modern neural networks[1–4] especially in the form of Convolutional Neural Networks (CNNs) have been actively employed in numerous tasks within robotic platforms thanks to their generalization capability and high accuracy (*e.g.*, achieved nearly 90% accuracy on the ImageNet dataset [5]). However, their main drawback is the total number of parameters used since most of the time it is tremendously large. The most widely used CNN models for mainly image recognition tasks are VGG16[6], InceptionV3[7], and ResNet18[8], which have 138M, 24M, 11M parameters respectively. This brings the requirement of high computational power and time leading to limitations on their usage on mobile platforms. On the other hand, the number of parameters in deep learning models are important as they mean the power of expression for a given dataset. Therefore reducing the parameter also means losing the power for generalizing the data.

Recent CNN models reduced the redundancies by replacing big kernels with smaller ones like Inception[7], or even generate the kernels for convolution with 1×1 convolution like Involution [9]. Although the aforementioned models have great performance on image classification tasks, they still require high computational cost, which makes the usage of deep neural networks on mobile devices still challenging.

Attempts for deploying deep neural nets on mobile devices have been studied. An object detection model using depthwise convolution were proposed in [10]. Depthwise convolution is an algorithm, which uses the same convolution kernel for every channel. Models proposed in [11] and [12] are image classification models that use channel shuffle for group convolution to reduce the computational complexity. Their proposal ghost module uses a low computational costly operation to reduce the number of parameters and computational complexity. However depthwise convolution has less power of expression, and channel shuffle regard the channel groups mutually independent therefore the information within channel groups is not exchanged.

Several approaches have been presented to reduce the number of parameters while trying not to lose the performance. Knowledge distillation[13] is a method that uses a dense network as a teacher model, and a sparse network as a student model. The term distillation means that the student model learns the soft label of the teacher model. By learning both hard label, which is the loss function of the prediction and the ground truth, and the soft label i.e. the loss between the prediction of the student and the teacher model, so that the student model can mimic the output of the teacher model. If properly trained, the student model will act similarly with the teacher model with fewer parameters. Filter pruning[14] is a method for reducing the number of filters in CNN. When pruning the convolution filters, it is necessary to sort by the sum of weights for each kernel for every layer. Since kernels with weights close to 0 will not affect much of the performance of the model, therefore when given a proper threshold it is possible to reduce the model and obtain a sparse model with similar performance. Duo Li et al.[9] defined that convolution has both "channel-specific and spatial-agnostic" features. The term spatial-agnostic means that the we limit the receptive field of convolution kernels to extract the features in various positions. However, if we have same features with different scale or features that are far apart, convolution lacks the ability of capturing these kind of features. This yielded to make filters that capture different features of the same object, causing the inter channel redundancy. In summary, the main problems when pruning Involution kernels are

twofold: (1) Involution kernels can have arbitrary size but the saved weight file does not contain information about the kernel size. (2) Even if the kernel size is known, there are still extra manipulations needed e.g., summation and sorting.

This research aims to reduce the redundancies of CNN using involution[9] and prune the network while keeping the implementation as simple as possible. Involution kernels are different from convolution kernels as they include reshaping layers. Therefore if the filters are sorted similarly in conventional methods[14], it causes the involution kernel to lose the spatial information. In this paper, we propose a method, which is easy to implement and reduces 50% of the model's convolution filters. Moreover, we show that this method could be applied in other CNN models without losing much accuracy.

Our main contributions are summarized as follows:

- Remove a certain portion of weights regardless of the kernel size.
- Model diet does not require extra manipulation operation.
- It can be applied to other CNN models.

Since many robotic vision applications such as depth estimation, object detection, image segmentation highly rely on visual features and Image classification is a task to test the ability of a model how well it extracts the visual features from an image, our computational experiments have been carried out on image classification models.

2. BACKGROUND AND RELATED WORK

2.1 Involution

Involution is proposed aiming to reduce the inter-channel redundancy of CNNs. It reversed the inherence of convolutions and has spatial-specific and channel-agnostic features. This means that involution kernels refer to the channels for each pixel when generating the kernel values. Involution makes use of linear transforms for each pixel and reshapes the output for a given kernel size. Therefore every pixel has different kernel values when they have different pixel values. Unlike randomly initialized weights of the kernel as in CNNs, the kernel of involution is conditioned by the values for each pixel.

Let $X \in \mathbb{R}^{H \times W \times C_i}$ be a feature map and $X_{i,j} \in \mathbb{R}^{C_i}$ be a pixel in the feature map where C_i denotes the number of channels of the feature map. We also define the involution filters $\mathcal{H} \in \mathbb{R}^{H \times W \times K \times K \times C_i}$ and the involution filter $\mathcal{H}_{i,j} \in \mathbb{R}^{K \times K \times C_i}$. We then define a linear projection $W : \mathbb{R}^{C_i} \mapsto \mathbb{R}^{K^2 \times C_i}$ where K denotes the involution kernel size. For each pixel $X_{i,j}$, linear projection W will be operated and the output will be reshaped into C_i number of $K \times K$ shaped kernels. Since the involution filter is a concatenated tensor of involution kernels, the entire kernel generation can be written as below.

$$\mathcal{H}_{i,j} = Reshape(W(X_{i,j})) \tag{1}$$

The kernel generation function is defined as $\phi : \mathbb{R}^{C_i} \mapsto \mathbb{R}^{K \times K \times C_i}$ such that

$$\mathcal{H}_{i,j} = \phi(X_{i,j}). \tag{2}$$

Once the involution kernel is generated, we perform a multiply-add operation through the channel dimension. The difference between convolution and Involution is that for convolution, every pixel shares the same kernel. On the other hand, for Involution, the values of the kernel vary depending on the pixels channel therefore all pixels have different kernels.

2.2 Model compression

Hinton et al.[13] proposed knowledge distillation to train a sparse network i.e., the student model by learning the soft labels of the dense network i.e., the teacher model. The student model uses both hard labels from the ground truth and the soft labels from the teacher model. The term "soft" comes from the temperature constant applied in the softmax function. We can penalize the softmax function using the temperature constant T. Then the penalized softmax can be written as below.

$$Softmax_p(x_i) = \frac{exp(x_i/T)}{\sum_j exp(x_j/T)} \tag{3}$$

As T gets bigger, the difference of each element of x gets smaller and as T gets smaller, the difference gets bigger. If the elements have big difference we can say that the output of the softmax is close to the ground truth i.e. the hard label, and the small difference result in all elements having similar values i.e. the soft label. We define the soft label loss i.e. kd loss $\mathcal{L}_{\mathcal{KD}}$ as

$$\mathcal{L}_{KD} = Softmax_p(MSE(S(x, \theta_S), T(x, \theta_T)), \tau) \tag{4}$$

where MSE is the mean square error, τ is the temperature constant and $S(x, \theta_S), T(x, \theta_T))$ stands for the output of the student and teacher model respectively. The hard label loss is the common cross entropy loss for multi label classification.

$$\mathcal{L}_{CE} = CE(S(x, \theta_S), T(x, \theta_T)) \tag{5}$$

The overall loss function is defined as

$$\mathcal{L} = \sum_{\mathbb{B}} \mathcal{L}_{KD} + \mathcal{L}_{CE} \tag{6}$$

which means the sum of all losses along the batch dimension. The penalized softmax function makes the student model to mimic the characteristics of the teacher model, where the hard label from the ground truth guarantees the accuracy of the prediction. If successfully learned, we can train a model which acts similar with fewer parameters.

2.3 Pruning

Pruning was widely used throughout deep learning models. Dropout[15] is a technique that removes some weights in order to avoid overfitting. Pruning can also

be used in model compression. Hao Li et.al. [14] proposed a filter pruning algorithm to reduce the number of weights in CNN. [16] proposed a filter attenuation algorithm which prunes the filters by using several different criteria. For each filter in the network, the sum of all weights in the filter are calculated and then sorted the filters by the sum of their weights. If the sum is smaller than a certain threshold, we discard the filter. Filters with weights that are close to 0 means that they are most likely to have less effect on the performance. Removing the filters that less affect the performance can result in reducing the parameters of the network.

3. PROPOSED MODEL DIET

In order to use pre-trained models, the weights of the model need to be loaded and the weights of the saved model differ from their shape. For example, convolutional layers are saved as (output channels, input channels, K, K) where K stands for the kernel size, and linear projection layers are saved as (output features, input features). Since Involution kernels are generated from linear projection and a reshaping layer, the linear projection of the involution kernel is saved as (output features, input features). The saved model does not contain the reshaping layer since reshaping is an operation rather than a neural network layer.

The reshaping permutes the output of the linear projection into a convolution kernel, therefore the kernel size is unknown when the weights of the pre-trained model are loaded. Since the kernel size could be an arbitrary number, the entire model structure is needed to be known and read from the file in order to use the conventional filter pruning methods. To overcome the unknown kernel size problem we intended to keep the algorithm as simple as possible. The term *diet* comes from reducing the weights in half while maintaining the model depth. Let N be the number of the weights. The weights are first split into g groups, where each group G_i takes the index from $(N/g) \times i$ to $(N/g) \times (i+1)$ where i is a natural number smaller than g. Then the weights of each group are summed and the group that has the biggest sum is used for the diet model. Since the bias and batch normalization with the convolution results must be matched, the sum of groups is needed instead of the sum of kernels. Our aim is to change the shape of the weight (output channel, input channel, K, K) into (output channel/g, input channel/g, K, K) therefore the weights from index $(N/g) \times i$ to index $(N/g) \times (i+1)$ where i indicates the index of the group which has the max sum of elements are kept. Any whole number g where g is a divisor of N/K^2 can be chosen. For example, if $g = 2$, half of the weights will be used. Fig. 1 depicts the graphical explanations of conventional pruning methods and the proposed model. The left vector corresponds to a weight vector of a linear projection. The right side indicates the involution kernel after the reshaping layer. Note that the darker color means the bigger value of the weight and the red cross means that the weight is pruned. In order to prune the correct filter, we need to add K^2 number of weights before sorting where K is the kernel size. However saved model files does not provide any information about the kernel size therefore deep technical details about the model are needed in order to apply pruning methods.

Algorithm 1 Model diet pseudo code

```
Load Full model state dict
for each key in Full model do
    for i, g in enumerate(groups) do
        start=(N/g) * i
        end = (N/g) * (i + 1)
        if full[key].shape is not diet[key].shape then
            if len(full[key].shape) is 1 then
                g[key]=full[key][start:end]
                groupsum[g]+=sum(g[key])
            end if
            if len(full[key].shape) is 2 then
                g[key]=full[key][:, start:end]
                groupsum[g]+=sum(g[key])
            end if
            if len(full[key].shape) is 4 then
                g[key]=full[key][start:end,start:end,:,:]
                groupsum[g]+=sum(g[key])
            end if
        end if
    end for
end for
diet = group[argmax(groupsum)]
Save diet as state dict
```

Algorithm 1 shows the pseudo code of the diet operation. State dict is the dictionary where the model weights are saved. When the full model's state dict is loaded, the algorithm first checks the type of weight. Length 1 corresponds to the bias of a layer, length 2 corresponds to the weights to a fully connected layer and length 4 corresponds to the weights of a convolutional layer. The term half means the half of the shape with the corresponding dimension. As can be seen in the algorithm, diet operation can be done regardless of the kernel size.

4. EXPERIMENTAL RESULTS

We follow the work in [9] using RedNet model to test our method. RedNet is a successor model of ResNet [8] in image recognition but the convolution layers are replaced with Involution layers. The full model refers to the model without the diet while the diet model is the model after applying the diet algorithm. A randomly initialized model is a model that has the same architecture as the diet model but has random weights. First, we test the models using RedNet to show that diets are effective to Involution. Then we use different CNN models [6, 8] to show that diet also works with conventional CNNs. Conventional CNNs follow the implementation of PyTorch. The test accuracy was measured with both diet only and knowledge distilled model.

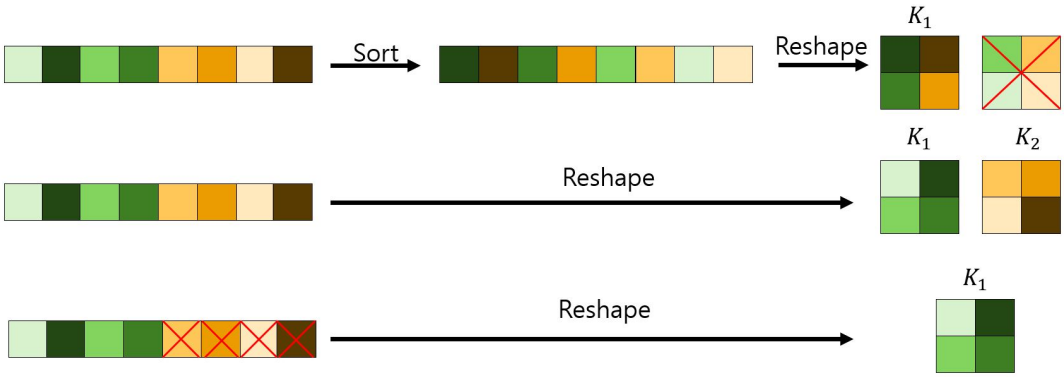

Fig. 1. Conventional filter pruning (top), the actual involution kernel (middle), and the diet operation (bottom). The green and brown indicates different kernels and K_i stands for the i-th kernel. If the results are different from the middle, the spatial information is lost. Conventional pruning sorts the weights before reshaping therefore the result differs from the actual kernel. Diet in contrast prunes the weights before reshaping therefore the results are equal to the actual kernel.

4.1 Dataset

The Imagenette [1] dataset is a small subset of the ImageNet dataset [5] that is composed of 10 different class labels for image classification. It contains about 9000 images for training and 3000 images for testing. Imagenette is a useful subset as it allows for faster processing. Since benchmark datasets usually have millions of images, it might require more computational power and time to process.

We first fine tune the ImageNet pre-trained full model with Imagenette. Then we diet the full model reducing the number of weights and build a new diet model. We compare the change of top-1 classification accuracy during the training epochs between the diet models i.e. the model that inherits the pre-trained weights and the model that was randomly initialized.

4.2 Parameter setup

Images in the dataset are 3-channel RGB. The values of the channels are biased, an image of a red flower will have high pixel values for red channels, while an image of a blue sky might have large values for blue channels. This bias will affect the ability to capture the visual patterns therefore we apply normalization each channel to remove the channel bias. All the input images are normalized with the mean $(0.4914, 0.4822, 0.4465)$ and the standard deviation $(0.247, 0.243, 0.261)$ indicating for each channel correspondingly. We crop each image with random axes and flip the image with 0.5 probability to avoid overfitting. We also used constant learning rate of $1e-5$ with batch size 32 using Adam optimization. The training epochs were same during training both the diet model and the randomly initialized model.

4.3 Computational Results and Analysis

Table 2 and Table 3 shows the number of parameters and the test accuracies before and after diet. We could reduce the number of parameters more than half by simply halving the number of filters. The diet could reduce up to approximately 75% of the parameters. Table 1 and Fig. 2 shows the size in MB of each model and the test accuracy respectively. Although the accuracy was lowered about 7% compared to the full model, the results still remain in a reasonable area. The accuracy is lower than the previous work but its implementation is much easier as well as the computational cost needed. Model diet can be operated by simply using if-then rules depending on the weight shape without requiring extra data manipulation operations such as adding or sorting.

Table 4 shows the number of floating-point operations in a billion scale. It can be seen that the floating-point operations have reduced about 50-60%. This difference will make a drastic difference in the inference time when it is needed to operate in real time.

Model diet has its limitations on the program design since it becomes complicated to implement when the model architecture does not consist of basic blocks. Pytorch implements ResNet by using basic block layers and RedNet implementation follows the work of Pytorch. Although the diet algorithm can still be applied to those models, its implementation may not be as simple as done by if-then rules. Also, more sophisticated methods can be developed on how to choose weights that would be kept in the reduced model.

5. CONCLUSIONS AND FUTURE WORK

In this research, we proposed a novel method for compressing the neural network models by reducing the weights used. Our experimental results provided that it can reduce up to 75% of the parameters without requiring much computational efforts. Also Involution kernels have reshaping layers so the kernel size remains unknown from weight files. Even if the kernel size is known, we still need extra manipulation such as summation and sorting. The proposed Model diet overcomes these limitations since it does neither need the kernel size nor require extra manipulation. Diet models can learn faster com-

[1] https://github.com/fastai/imagenette

Table 1. Size comparison for each model (in MB)

Model Size	RedNet26			ResNet18			VGG16		
	Full	Diet	Reduction Rate	Full	Diet	Reduction Rate	Full	Diet	Reduction Rate
Inference	492.54	287.04	41.72%	62.79	52.06	17.09%	322.13	161.07	50.00%
Paramaters	27.48	7.00	74.53%	42.65	11.21	73.72%	512.35	128.15	74.99%
Total	520.60	297.62	42.83%	106.01	63.84	39.78%	835.06	289.79	65.30%

Table 2. Top 1 accuracy(%) of the diet model.

Model	Full	Diet	with KD	random initialized
RedNet26	93.9%	86.7%	89.5%	0.60%
ResNet18	97.4%	90.2%	90.8%	0.75%
VGG16	98.2%	89.7%	90.8%	0.79%

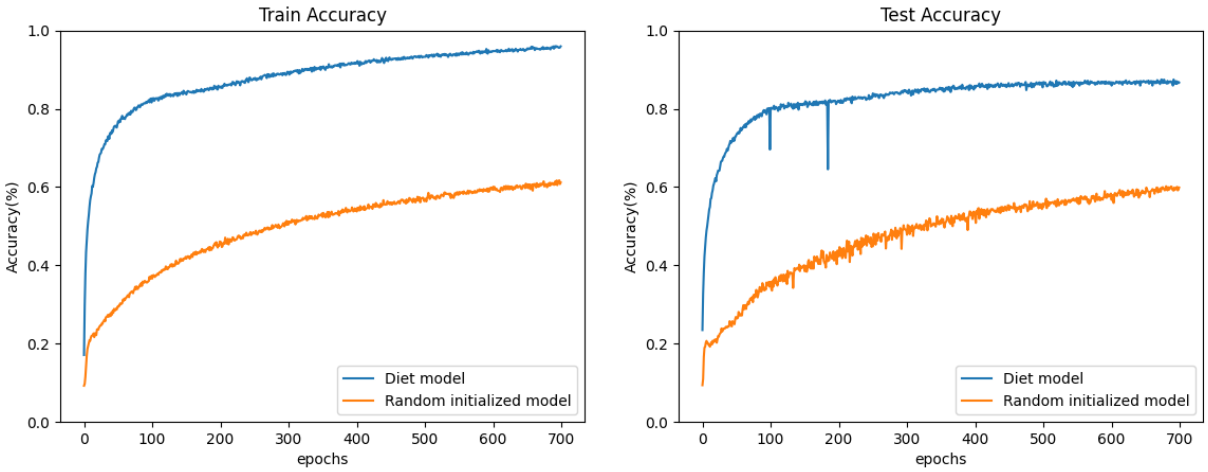

Fig. 2. The train, test accuracy of random initialized and the diet model(Tested with Involution).

Table 3. Number of parameters ($\times 10^6$) before and after diet.

Model	Full	Diet	Reduction Rate
RedNet26	7.20	1.84	74.44%
ResNet18	11.18	2.94	73.70%
VGG16	134.31	33.59	74.99%

Table 4. Number of GFLOPs before and after diet.

Model	Full	Diet	Reduction Rate
RedNet26	1.75	0.67	61.71%
ResNet18	1.83	0.92	49.72%
VGG16	15.54	3.93	74.71%

pared to randomly initialized models and can be applied on conventional CNN models as well. The accuracy is decreased by approximately 7% but still remains on a reasonable area. Since diet RedNet26 only have 1.84M parameters, model diet can be applied on devices that require less parameters such as mobile devices or mobile robots.

For future work, some experimental and performance analysis will be targeted in other models for object detection, depth estimation and similar other tasks. Also, more sophisticated ways to select a group of weights apart from the summation of weights will be studied.

REFERENCES

[1] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V. Le. “Meta Pseudo Labels”. In: *CoRR* abs/2003.10580

(2020). arXiv: 2003.10580. URL: https://arxiv.org/abs/2003.10580.

[2] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. "High-Performance Large-Scale Image Recognition Without Normalization". In: *CoRR* abs/2102.06171 (2021). arXiv: 2102.06171. URL: https://arxiv.org/abs/2102.06171.

[3] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. *Scaling Vision with Sparse Mixture of Experts*. 2021. arXiv: 2106.05974 [cs.CV].

[4] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. *Scaling Vision Transformers*. 2021. arXiv: 2106.04560 [cs.CV].

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, pp. 248–255.

[6] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[9] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. *Involution: Inverting the Inherence of Convolution for Visual Recognition*. 2021. arXiv: 2103.06255 [cs.CV].

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474.

[11] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[12] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. "GhostNet: More Features From Cheap Operations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML].

[14] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. *Pruning Filters for Efficient ConvNets*. 2017. arXiv: 1608.08710 [cs.CV].

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[16] Morteza Mousa-Pasandi, Mohsen Hajabdollahi, Nader Karimi, Shadrokh Samavi, and Shahram Shirani. "Convolutional Neural Network Pruning Using Filter Attenuation". In: *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2020, pp. 2905–2909.