Interpretable Customized Neural Network Pruning by Identifying Critical Data Routing Path

Dongyue Li
Shanghai Jiao Tong University
Student ID: 516030910502
Email: lidongyue@sjtu.edu.cn

Mengqi Cao
Shanghai Jiao Tong University
Student ID: 516030910497
Email: rogercmq@163.com

***Abstract*—Convolutional neural network has achieved great success but requires tremendous computation resources. Due to the constrained computation resources, neural network compression and pruning are widely demanded currently due to the resource constraints on most deployment targets. Prior network compression and pruning methods either suffer from accuracy degradation or achieve poor acceleration. Recent work on network interpretability that critical path formed by a few class-specific neurons contribute most information for specific classes provides us new ideas to prune network.**

In this paper, we propose a novel pruning method based on identifying critical data routing path in neural network and create customized models based on that. We associate control gates values with channels, train it with the idea of network distilling, and prune the network based on gate values. Experiment results show that we can achieve 98.99% average accuracy for single class tasks and far above original model accuracy for multi-class tasks, and we can achieve 99% pruning ratio without hurting the model performance.

I. Introduction

Convolutional neural networks (CNN) have obtained tremendous success in recent years, achieving near-human performances on tasks such as visual recognition [9], object detection [4], and semantic segmentation [13]. With Large-scale datasets, high-end modern GPUs and new network architectures, CNN models like AlexNet [9], VGGNet [19], GoogleNet [20], and ResNets [6] have grown unprecedented large from several layers to over 100 layers. However, larger CNNs require tremendous computation resources [6]. This is unlikely to be affordable on resource-constrained platforms such as embedded systems, mobile devices, wearable or Internet of Things (IoT) devices. Thus, limited computation power of these devices make it difficult to deploy in real world application.

Real world applications often requires customized classification models which only predict a few classes but need to be fast with constrained computation resources. Thus, the deployment of CNNs in real world applications are mostly constrained by model size, run-time memory and number of computing operations [11]. researchers have sought ways to make the model more lightweight by network pruning methods [5], [7], [18] or more sparse by regularization [2], [22]. However, one of the main problems of these methods is that they suffer from great accuracy degradation if the sparsity or the pruned rate of the network is overly high. Other methods including binarization [3], [17] or weight pruning [5] require specially designed software/hardware accelerators for execution speed-up. Moreover, all the pruning methods ignore the customization demand of resource-constrained platforms and waste efforts in pruning model for all classes.

To obtain compact customized model, we seek for interpretable pruning methods. Most works are based only on quantitative characteristics of the convolutional filters, and highly overlook the qualitative interpretation of individual filters specific functionality. Recent works in network interpretability [15] and critical path [21], [24] give us inspiration. Researchers have found that the neurons in higher layers of neural networks are usually specialized to encode information relevant to a subset of classification targets [14], [25]. By ablating certain neurons in convolutional layers, significant accuracy degradation of specific classes can be observed [23]. This reveals the neuron differentiation process across multiple layers, but also reveal that the class relevant information is significantly concentrated in a few neurons [24]. In other words, the per-class information flow is directed by a **critical path** formed by a few class-specific neurons with significant contributions. Specifically, we denote the **critical nodes** on the routing paths as the important channels of intermediate layer's output that if they were suppressed to zeros, the final test performance would deteriorate severely, which is the same definition by ablation methods. Formally, we define the **Critical Data Routing Paths (CDRP)** as the union of all the critical node routing paths for a specific target class.

In this paper, we propose a novel network pruning scheme based on identifying critical data routing path. To find critical data routing path, we implement the distillation guided routing method proposed in [21]. Specifically, we associate a scalar control gate to each layers output channel to learn the optimal routing paths for each individual sample. We optimize the control gate values by minimizing the difference between the network output with control gates and the original model, which is inspired by the idea of knowledge distillation [8]. We concatenate control gates values in all layers to obtain one critical path representation for one sample. Large gate value implies great significance of a convolution channel, otherwise the convolution channel is less important. Since every pic has its own routing path, we merge all routing paths of pictures of one same class to obtain a critical path representation for

one class.

By identifying critical data routing paths, we find that the critical paths for different classes are co-existing but do not overlap with each other. This property of exclusiveness provides the chance to distil the critical paths and further decouple the CNN for customized prediction tasks. Therefore, we derive own interpretable customized network pruning methods. We select out the critical neurons for given classes by the value of control gates and prune out the rest unimportant neurons, which results in models for customized tasks. After pruning, the resulting network can achieve specified sparsity compared to the initial wide network.

Experiments show that we can obtain very sparse pruned network with the accuracy far more better than original model. We test our pruning algorithm on VGG-16 network model. First, similar to the work in [24], we keep the critical path of neurons for single classes and perform one-vs-all tasks. We achieve average test accuracy 98.99% which is better than the results in [24] and we can set the pruning ratio even higher with the accuracy not degrading. Second, we create customized model for more classes, such as $5, 10$, and 25 classes and achieve average test accuracy 94.8%, 90.5%, and 75% respectively. We also find out that with pruning ratio increased, results of out method have not degraded, thus we can achieve much higher accuracy with much higher accuracy than prior works.

II. Background and Related work

In [10], the authors present an acceleration method for CNNs, where they prune filters from CNNs that are identified as having a small effect on the output accuracy based on l_1-norm ranking. By removing whole filters in the network together with their connecting feature maps, the computation costs are reduced significantly.

In [12], the authors propose the network slimming technique to learn more compact CNNs, which directly imposes sparsity-induced regularization on the scaling factors in batch normalization layers, and unimportant channels can thus be automatically identified during training and then pruned.

In [24], the authors demonstrate that the filters in higher layers learn extremely task-specific features, which are exclusive for only a small subset of the overall tasks, or even a single class. Based on such findings, they propose a critical path distillation method, which adopts gradient ascent algorithm to maximize a neurons activation with input X. Based on previous research, in [16], they also show that the functional repetitive filters could be effectively pruned from neural network to provide computation redundancy and propose an interpretable filter pruning method by first clustering filters with same functions together and then reducing the repetitive filters with smallest significance and contribution.

In [21], the authors develop a Distillation Guided Routing method, which is a flexible framework to interpret a deep neural network by identifying critical data routing paths and analyzing the functional processing behavior of the corresponding layers. In terms of critical data routing path representation, they propose to discover the critical nodes on the data information flow during network inferring prediction for individual input samples by learning associated control gates for each layers output channel. Our pruning method is inspired by this work and we reconsider this frame in respect to the neural network partition problem.

III. Methodology

In this section, we introduce our methods of finding critical data routing path and pruning models which is inspired by [1], [7], [12], [21]. We first describe the advantage of channel level sparsity. Second, we describe our methods of finding critical data routing path by training scalar control gates associating with each layers output channel. Third, we propose our prune method base on these control gate values.

A. Advantages of Channel-level Sparsity

Network sparsity can be realized at different levels [12], e.g., weight-level, kernel-level, channel-level or layer-level. Fine-grained level (e.g., weight-level) sparsity gives the highest flexibility and generality leads to higher compression rate, but it usually requires special software or hardware accelerators to do fast inference on the sparsified model. On the contrary, the coarsest layer-level sparsity does not require special packages to harvest the inference speedup, while it is less flexible as some whole layers need to be pruned. In fact, removing layers is only effective when the depth is sufficiently large, e.g., more than 50 layers. In comparison, channel-level sparsity provides a nice trade-off between flexibility and ease of implementation. It can be applied to any typical CNNs or fully-connected networks (treat each neuron as a channel), and the resulting network is essentially a thinned version of the unpruned network, which can be efficiently inferenced on conventional CNN platforms.

B. Channel-Wise Control Gate

We introduce a channel-wise scalar λ to identify the critical data routing paths of each given input $\mathbf{x}$. Such channel-wise scalars are expected to maximize a neuron's activation. During inference forward pass, a group of control gates λ_k will be multiplied to the k_{th} layer's channel output, resulting in actual routing nodes. Each layer's routing nodes are connected to form the routing paths. The problem of identifying the critical data routing paths reduces to optimize $\mathbf{\Lambda} = \{\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, ..., \boldsymbol{\lambda_K}\}$, where K corresponds to the depth of our pretrained network.

For valid and reasonable critical data routing paths, we consider each λ should satisfy these two conditions [21]:

- λ is non-negative
- λ should be as sparse as possible.

Denote $\mathbf{\Lambda}^* = \{\boldsymbol{\lambda_1^*}, \boldsymbol{\lambda_2^*}, ..., \boldsymbol{\lambda_K^*}\}$ as the optimized control gates, and the corresponding identified CDRPs can be represented by

$$\boldsymbol{v} = \textit{concatenate}([\boldsymbol{\lambda_1^*}, \boldsymbol{\lambda_2^*}, ..., \boldsymbol{\lambda_K^*}]) \tag{1}$$

C. Critical Data Routing Path

We denote the critical nodes on the routing paths as the important channels of intermediate layers output that if they were suppressed to zeros, the final test performance would deteriorate severely. Specifically, we associate the scalar control gate to each layers output channel to learn the optimal routing paths for each individual sample. The Critical Data Routing Paths(CDRPs) can, therefore, be represented based on the responses of concatenated control gates from all the layers, which reflect the networks semantic selectivity regarding to the input patterns and more detailed functional process across different layer levels. Interestingly, such critical data routing paths not only reflect the functional process of intermediate layers in the network, but also reflect the input data layout patterns [21].

For image-classification tasks, critical data routing paths for single image class can be derived via merging all CDRPs of each given same-labelled input. Our present merging method is to sum up control gate values of the same-labelled input samples in the corresponding channels. As for multi-class CDPRs, we simply joint all desired single-class CDPRs.

D. Control Gate Training

Algorithm 1 Channel-Wise Gate Training

Require: Input image $\mathbf{x}$, pretrained network $f_\theta(\cdot)$, control gates $\mathbf{\Lambda}$ initialized with 1, balanced parameter γ. Max iterations T, SGD optimizer
Ensure: Identified CDRPs representation v

original prediction class $i \leftarrow argmax\ f_\theta(\mathbf{x})$
for $t \leftarrow 1\ to\ T$ **do**
  Compute loss cur_loss by Equation (2);
  Compute control gates gradients $\mathbf{\Lambda}$ by Equation (3);
  Update $\mathbf{\Lambda}$ by SGD optimizer and clip $\mathbf{\Lambda}$ to be non-negative;
  new prediction class $j \leftarrow argmax\ f_\theta(x; \Lambda)$;
  if $i = j$ **then**
    if cur_loss is minimum **then**
      $v \leftarrow concatenate(\mathbf{\Lambda})$;
    end if
  end if
end for
return v;

Inspired by [21], we design and formulate our gate training algorithm, which is shown in Algorithm 1. The optimization objective for all the control gates $\mathbf{\Lambda}$ of each given input is

$$\min_{\mathbf{\Lambda}} crossEntropy(f_\theta(\boldsymbol{x}), f_\theta(\boldsymbol{x}; \mathbf{\Lambda})) + \gamma \sum_k |\boldsymbol{\lambda}_k| \quad (2)$$

where the original full model prediction probability $f_\theta(\boldsymbol{x}) = [p_1, p_2, ..., p_m]$ and the new prediction probability $f_\theta(\boldsymbol{x}; \mathbf{\Lambda}) = [q_1, q_2, ..., q_m]$ where m is the category number, and γ is the balanced parameter. After calculating the original full model prediction probability for the given input data, the gradients for control gates are computed by

$$\frac{\partial Loss}{\partial \mathbf{\Lambda}} = \frac{\partial L}{\partial \mathbf{\Lambda}} + \gamma * \text{sign}(\mathbf{\Lambda}) \quad (3)$$

where $L = crossEntropy(f_\theta(\boldsymbol{x}), f_\theta(\boldsymbol{x}; \mathbf{\Lambda}))$, which are used for performing stochastic gradient descent on control gates.

E. Interpretable Customized Pruning Algorithm

Our pruning algorithm is shown in Algorithm 2. After the input-level critical paths are derived, we merge all CDRPs of each given same-labelled input and therefore obtain class-level critical paths. For multi-class conditions, we joint all desired critical paths for each desired image class. For each channel-wise scalar in corresponding convolution layers, we sort those channel output values and proportionately remove channels with smaller λ from the original network. We prune channels with a global threshold across all layers, which is defined as a certain percentile of all the scaling factor values.

Algorithm 2 Interpretable Customized Network Pruning Algorithm

Require: Input image of targeted classes $TargetImages$, pretrained network $f_\theta(\cdot)$, $prune\_ratio$
Ensure: Final pruned neural network model $f_{\theta'}(\cdot)$

$Paths = \{\}$;
for each given training image $\mathbf{x}$ in $TargetImages$ **do**
  $Path_{\boldsymbol{x}} = GateTraining(\boldsymbol{x}, f_\theta(\cdot))$;
  $Paths$.add($Path_{\mathbf{x}}$);
end for
Compute class-level critical data routing path Sum_Path via merging $Paths$;
for Sum_Path's $\boldsymbol{\lambda}_k$ in the k_{th} convolution layers **do**
  sort all λs decreasingly and remove certain channels with smaller gate values via $prune\_ratio$;
end for
Fine-tune the pruned network and obtain $f_{\theta'}(\cdot)$;

IV. Experiment

A. Implementation Details

In this section, we implement our pruning algorithm based on the TensorFlow 1.4.1 with CUDA 8.0. We use CIFAR-100 dataset and VGG-16 network for all the experiments. Codes are available at Github Link.

CIFAR The two CIFAR datasets consist of natural images with resolution 32×32. CIFAR-10 is drawn from 10 classes and CIFAR-100 from 100 classes. The train and test sets contain 50,000 and 10,000 images respectively. We report the final test accuracy after gate training or fine-tuning on all training images. The input data is normalized using channel means and standard deviations.

Pretrained Network Model On CIFAR-100 dataset, we evaluate our pruning method on the VGG-16 [19]. The VGGNet is originally designed for ImageNet classification. For our experiment a variation of the original VGGNet for CIFAR

dataset is taken from VGGNet-CIFAR. In order to proceed the training process smoothly, we add Batch Normalization and Xavier Initialization in the origin VGG-16.

In our adopted pretrained VGGNet, top-1 accuracy is 71.56% and top-5 accuracy is 91.52%, which achieves the best performance among available VGGNet pretrained models.

Control Gate Training Before optimizing the objective in Equation (1), all control gates in $\mathbf{\Lambda}$ are initialized with 1, which activates all the nodes. After calculating the original full models prediction probability for the given input data $\mathbf{x}$, we perform SGD on the same input $\boldsymbol{x}$ for $T = 100$ iterations, with learning rate of 0.1 and L1-norm penalty of 0.03. After finishing the iterations, the optimized CDRP representation $\boldsymbol{v}$ is formed by concatenating $\mathbf{\Lambda}$, which can result in the lowest loss value while retaining the exact same top-1 prediction with the original model. We allow λ larger than 1 to compensate the distribution variation of output channels after multiplied by control gates.

To obtain Critical Data Routing Paths for some class k, we collect images labelled as class k in CIFAR-100 training dataset and merge their gate encodings via summing up corresponding λ in each channel. Through similar method, we can obtain multi-class CDRPs.

Pruning When we prune the channels of models trained with sparsity, a pruning threshold on the scaling factors needs to be determined. In short, we use a global pruning threshold for simplicity. The pruning threshold is determined by a percentile among all scaling factors , e.g., 60% or 80% channels are pruned. The pruning process is implemented by copying the corresponding weights from the model trained with sparsity and masking zeros to pruned channels so that such neurons would be inactivated during feedforwarding.

We also implement our pruning method via set the pruning threshold numerically, e.g., channels with the control gate value smaller than ten are pruned, but the overall pruned ratio is unpredictable and the performance strongly depends on the domain. It means that different sub-combinations of our CIFAR100 classes lead to either expected or catastrophic performance, which is unexplainable to our knowledge.

Fine-tuning After the pruning process, we obtain a narrower and more compact model, which is then fine-tuned. Since we focus on the network partition problem, we slightly change the original VGGNet architecture via changing the output dimension of the last Fully-Connected Layer into the number of our target classes. It would not have an fierce influence on the performance after fine-tuning. On CIFAR-100 datasets, the fine-tuning uses the same optimization setting as in training.

B. Experiment Results

General Results We perform experiments on VGG-16 network for different customized network. For customization, we set the target classes of customized model with different number. First, similar to the work in [24], we prune the original model for customization models of single class and perform one-vs-all tasks. The experiment results show that our pruned customized model of single classes achieve 98.99% average test accuracy, which is better than the results of 98.6% average True Positive rate and 11.0% False Positive (FP) rate in [24]. Even more, we can increase the pruning ratio from 80% to 99% without hurting the test accuracy in contrast to the fixed 90% pruning ratio in [24].

Furthermore, we also perform our pruning methods on customized model of $5, 10$, and 25 classes and test the accuracy for pruning ratio of $80\%, 90\%, 95\%$, and 99%. The results on CIFAR-100 are shown in Table I and Table II, where pr refers to the hyperparameter pruned percentile, e.g. $pr = 0.8$ indicates 80% of channels are inactivated. In Table I, we demonstrate the average accuracy of pruned customized model of $5, 10$, and 25 classes with pruning ratios of 80% which have not been fine tuned. This results show that the accuracy degrades very much after pruning. In reality, performances are strongly dependent on our targeted classes. For certain targeted classes, the result can achieve above 90% test accuracy. However, for some of targeted classes, the result can degrade to 0.1% test accuracy, causing the average accuracy very low.

TABLE I: Average Accuracy of Customized Models: Pruned but not Fine-tuned

	pr=0.8
number of targeted classes=5	0.1431
number of targeted classes=10	0.5982
number of targeted classes=25	0.4675

In Table II, we present the results of average accuracy of pruned and fine-tuned customized model of $5, 10$, and 25 classes with pruning ratios of $80\%, 90\%, 95\%$, and 99%. Notice that, after fine tuning, the accuracy has been improved. For customized model of 5 classes with pruning ratio 80%, the average accuracy is 94.8%. For customized model of 10 classes with pruning ratio 80%, the average accuracy is 90.5%. For customized model of 25 classes with pruning ratio 80%, the average accuracy is 75%, which is still better than original model accuracy. In details, for the customized model of 25 classes with pruning ratio 80%, the best accuracy we notice is above 80% and the worst accuracy is 68%, making the average accuracy is 75%. Interestingly, we increase the pruning ratio later on and find out that the average accuracy haven not been hurt and remain the almost the same accuracy of pruning ratio 80%. This results show that pruning methods based on critical data routing path can achieve better network sparsity because the critical data routing path is relatively sparse compared to whole network model. This not only prove the existence of critical path but also offer us higher pruning ratio we can achieve, thus making the network so small that they can be deployed to resource constrained devices.

Case Analysis In this part, we randomly select 5 cases of 25 classes from the CIFAR-100 dataset and test each accuracy, which is shown in Table III.

First, there is no exceptional case where the pruned and fine-tuned model fails to predict, e.g. zero-accuracy cases. Second, our customized network for smaller amount of classes perform better than the baseline pretrained network. Such findings

TABLE II: Average Accuracy of Customized Models: Pruned and Fine-tuned

	pr=0.8	pr=0.9	pr=0.95	pr=0.99
number of targeted classes=5	0.948	0.950	0.949	0.948
number of targeted classes=10	0.905	0.897	0.901	0.902
number of targeted classes=25	0.750	0.747	0.755	0.747

further validate that our critical data routing path method makes sense. Our pruned sub-network is more resource-saving but more effective.

Second, currently, sub-network pruning is task-specific. The above result shows that the average accuracy of our pruned model for 25 classes is 0.747 (if $pr = 0.99$). However, Case 3 is far more better than average while Case 4 is the other way around. It indicates that the relationship between different combinations of classes and the network performance is far from totally proven. In the future we will work on that.

Case Number	Selected Classes ID	Accuracy
Case 1	11, 12, 17, 18, 20, 23, 27, 34, 38, 43, 45, 56, 59, 60, 63, 64, 72, 77, 81, 83, 88, 93, 95, 96, 98	0.775
Case 2	2, 4, 8, 9, 11, 15, 19, 20, 31, 33, 47, 48, 63, 64, 68, 70, 80, 81, 85, 93, 94, 95, 96, 97, 99	0.745
Case 3	0, 1, 5, 7, 15, 22, 24, 25, 27, 30, 31, 38, 42, 43, 46, 58, 65, 71, 76, 77, 80, 82, 85, 88, 91	0.8
Case 4	2, 3, 8, 12, 14, 23, 25, 43, 45, 52, 59, 66, 69, 72, 73, 77, 81, 82, 83, 84, 85, 93, 95, 96, 99	0.685

TABLE III: Case Analysis for $pr = 0.99$

V. Conclusion

Neural network compression and acceleration are widely demanded currently due to the resource constraints on most deployment targets. In this work, we propose a pruning algorithm based on identifying critical data routing paths for deploying customized models to resource constraint platforms. Through experiments, the results show that our method create 98.99% accuracy models for single task and achieve average test accuracy 94.8%, 90.5%, and 75% for multi tasks of $5, 10$ and 25 classes respectively. Furthermore, our method can achieve 99% pruning sparsity without hurting model performance.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):32, 2017.

[2] Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

[3] M Courbariaux and Y Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or- 1. arxiv: 1602.02830, 2016, 2017.

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

[5] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[6] 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*, pages 770–778, 2016.

[7] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[11] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017.

[12] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[14] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

[15] Zhuwei Qin, Fuxun Yu, Chenchen Liu, Liang Zhao, and Xiang Chen. Interpretable convolutional filter pruning. *arXiv preprint arXiv:1810.07322*, 2018.

[16] Zhuwei Qin, Fuxun Yu, Chenchen Liu, Liang Zhao, and Xiang Chen. Interpretable convolutional filter pruning. *CoRR*, abs/1810.07322, 2018.

[17] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[18] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by identifying critical data routing paths. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[22] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *International Conference on Machine Learning*, pages 3958–3966, 2017.

[23] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[24] Fuxun Yu, Zhuwei Qin, and Xiang Chen. Distilling critical paths in convolutional neural networks. *arXiv preprint arXiv:1811.02643*, 2018.

[25] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.