Non-uniform DNN Structured Subnets Sampling for Dynamic Inference

Li Yang, Zhezhi He, Yu Cao and Deliang Fan
School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ
{lyang166, zhezhihe, ycao, dfan}@asu.edu

***Abstract*—With the success of Deep Neural Networks (DNN), many recent works have been focusing on developing hardware accelerator for power and resource-limited system via model compression techniques, such as quantization, pruning, low-rank approximation and etc. However, almost all existing compressed DNNs are fixed after deployment, which lacks run-time adaptive structure to adapt to its dynamic hardware resource allocation, power budget, throughput requirement, as well as dynamic workload. As the countermeasure, to construct a novel run-time dynamic DNN structure, we propose a novel DNN sub-network sampling method via non-uniform channel selection for subnets generation. Thus, user can trade off between power, speed, computing load and accuracy on-the-fly after the deployment, depending on the dynamic requirements or specifications of the given system. We verify the proposed model on both CIFAR-10 and ImageNet dataset using ResNets, which outperforms the same sub-nets trained individually and other related works. It shows that, our method can achieve latency trade-off among 13.4, 24.6, 41.3, 62.1(ms) and 30.5, 38.7, 51, 65.4(ms) for GPU with 128 batch-size and CPU respectively on ImageNet using ResNet18.**

I. Introduction

In the last couple of years, the climate of Artificial Intelligence, especially Deep Neural Networks (DNN), has swept various domains owing to its prominent performance over traditional methods [1]. However, DNNs grow into more complex structures consisting of deeper layers, larger model size, and denser connections. Such ”bulky” models rise challenge to hardware deployment, especially edge devices (e.g., smartphone). To solve this problem, researchers either design compact models specialized for mobile [2], [3], or accelerate the existing models by compression [4], including network quantization [5], low-rank approximation [6], weight non-structured/structured pruning [7], [8] and etc [9]. However, the available resource are non-identical for different hardware platforms, which requires different degrees of compression under similar latency requirement. Even for one specific hardware platform, it expects the dynamic switching ability in real-world scenarios. For example, smartphone may become too hot or is running out of battery, which resulting in different allocated computing resources to DNN computation and thus different throughputs, latency, etc. In these cases, DNNs need to be retrained and reloaded to meet various/dynamic requirements, which is highly cost, even not realistic. A new challenge is then raised: *How to develop an adaptive DNN model that could dynamically adjust its computing complexity, model size and accuracy to meet with dynamic application requirement and workload on-the-fly, without reloading new models?*

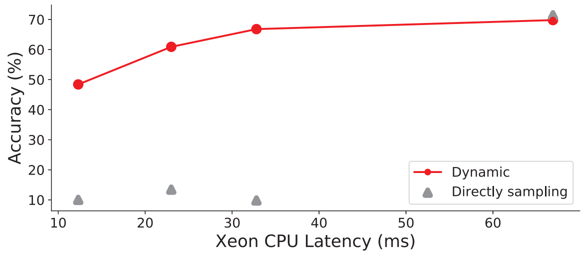


Figure 1: Directly sampling without retraining VS our proposed dynamic sampling for four subnets

To address this challenge, in this work, we target to construct a dynamic DNN structure, which consists of many subnets, through a novel sub-network sampling method via non-uniform channel selection. It is inspired by the fact that parametric layers (e.g., convolution or fully-connected layers) of DNN owns non-identical sensitivity to capacity reduction, which has been heuristically demonstrated by model pruning [8], [10], [11] and NAS works [12]–[14]. The proposed method can be divided into two successive steps: *Subnets generation* and *fused subnets training*. In the first step, multiple subnets are sampled from a complete model, in terms of different model capacities utilizing a proposed *clipped-Lasso* based channel sampling method. Then, the identified subnets are fused as a single ensemble loss function for multiple objective optimization to construct a dynamic inference network.

Thus, the new dynamic DNN could adjust the involved convolution channel (i.e. model size, computing load) at run-time (i.e. at inference stage without retraining) to dynamically trade off between computing complexity (thus power, speed) and accuracy as depicted in Fig. 1. Unlike prior works with uniform structure, our proposed dynamic DNN, i.e. supernet model, includes multiple subnets, each owns non-uniform structure to achieve optimal efficiency. We summarize our main contributions as follows:

- We propose a new dynamic neural network structure (a supernet consists of a group of subnets), which can adjust the model size at run-time to dynamically trade off computing complexity and accuracy.
- To extract the non-uniform and hardware-aware subnet structures, we also propose *clipped-Lasso* structured pruning method as the guide for subnets generation.
- We test our method on CIFAR-10 and ImageNet dataset which shows that our method achieves either similar or

better accuracy comparing with individual pruned models and other recent works with the same model size. In addition, we also prove that our method achieves better accuracy and latency trade off on both CPU and GPU.

II. Related Work

A. Neural Network Structured Pruning

DNN pruning is a popular compression technique, which enforces partial of weights be zero for both model size reduction and computation simplification [7]. According to the shape of sparsity pattern, it can be divided into two categories: non-structured [7] and structured pruning [8], [10], [15]. Structured pruning leads to sparsity patterns with highly regular shapes, which is much more hardware-friendly. Various sparsity pattern (e.g., channel/kernel/customized-group) are explored in different works [8], [10], [15]. In [10], the unimportant filters are directly pruned based on its L_1-norm. Liu *et al.* [15] introduce L_1 regularization on the scaling coefficient of batch normalization layers as a penalty term, where the channels with small scaling coefficient are removed. In contrast to the aforementioned two works, the structured pruning methods in [8] use the identical technique - Group Lasso.

Group Lasso is initially introduced in [16], then Wen *et al.* [8] apply it as an additional term in the loss function when training DNN with back-propagation for learning the structured sparse weight pattern, which can be formalized as:

$$\hat{\mathcal{L}} = \mathcal{L}(f(\boldsymbol{x};\{\boldsymbol{W}_l\}_{l=1}^{L}),\boldsymbol{t}) + \lambda \underbrace{\sum_{l=1}^{L}\sum_{i=1}^{G_l} \overbrace{\mathcal{P}(\boldsymbol{W}_{l,i})}^{\text{Intra-group } L_2\text{-norm}}}_{\text{Inter-group } L_1\text{-norm}} \tag{1}$$

where $f(\boldsymbol{x};\{\boldsymbol{W}_l\}_{l=1}^{L})$ computes the outputs of DNN parameterized by $\{\boldsymbol{W}_l\}_{l=1}^{L}$ w.r.t the input $\boldsymbol{x}$. $\mathcal{L}(\cdot,\cdot)$ is the objective function of DNN (e.g., cross-entropy loss). $\mathcal{P}(\boldsymbol{W}_{l,i}) = ||\boldsymbol{W}_{l,i}||_2$ calculates the Euclidean norm of the indexed weight group $\boldsymbol{W}_{l,i}$. The second term in the R.H.S of Eq. (1) is the L_1-norm of $\{\mathcal{P}(\boldsymbol{W}_{l,i})\}$ (aka. Group Lasso [16]), which acts as the group-wise weight penalty for improving the group-wise sparsity during the optimization. G_l is the number of groups in l-th layer, and λ is the hyper-parameter to be tuned based on the dataset. In this work, we focus on channel-wise pruning where the G_l represents the number of output channels.

B. Dynamic neural networks

Dynamic neural network is a certain model that can generate various subnets with different structures. [17] presents a feed-forward DNN that allows selective execution with controller modules. [18] uses a gating network to selectively skip convolutional blocks based on the activations of the previous layer. [19] incorporates multi classifiers as early-exits into a single DNN and inter-connects them with dense connectivity. [20] proposes Slimmable Neural Network (S-NN) which can train a single DNN to support multiple channel width in each layer. Further, the authors extend the S-NN to Universally Slimmable Networks (US-NN) which can execute arbitrary channel width in [21]. However, the pruning ratio of channel width is fixed for all layers in S-NN and US-NN, which means they only support subnets with uniform structures.

III. Dynamic Deep Neural Network

In this work, we aim to propose a framework to create a run-time dynamic DNN, whose subnets of varying model sizes can operate independently to meet various hardware specifications. As the overview of the proposed framework illustrated in Fig. 2, the entire optimization flow can be generally divided into two successive steps: **1) Subnets generation,** where multiple subnets are sampled from a complete model, in terms of different model capacities utilizing our clipped-LASSO based channel sampling method; **2) Fused subnets training**, where the subnets identified in the first step are fused as a single ensemble loss function for multiple objective optimization. Thus, each subnet can operate independently with the capability of balancing inference latency w.r.t the prediction accuracy on-the-fly.

To formally define the design objective, the entire workflow depicted in Fig. 2 can be mathematically expressed as:

$$\begin{gathered} \overbrace{\min_{\{\mathbf{W}_l\}_{l=1}^{L}} \sum_{i=1}^{N} \mathcal{L}_i\Big(f(\boldsymbol{x};\{\mathbf{W}_l\cdot\mathbf{M}_{l,i}\}_{l=1}^{L}),\boldsymbol{t}\Big)}^{\text{training for dynamic inference}} \\ \underbrace{s.t.\,\{\mathbf{M}_{l,i}\}_{l=1}^{L} = \arg\min \tilde{\mathcal{L}}(f(\boldsymbol{x};\{\mathbf{W}_l\cdot\mathbf{M}_{l,i}\}),\boldsymbol{t},\lambda_i,a_i)}_{\text{subnets generation/sampling}} \end{gathered} \tag{2}$$

where N is the total number of subnets, and $i \in \{1,...,N\}$ is the index of subnets. Assume the l-th layer is a convolution layer whose weight tensor $\mathbf{W}_l$ is in shape of $\mathbb{R}^{p\times q\times kh\times kw}$, $\mathbf{M}_{l,i} \in \{0,1\}^{p\times q\times 1\times 1}$ denotes the weight sampling mask of subnet-i in l-th layer. It is noteworthy that all the subnets are partially sharing the weight w.r.t the full model (i.e., $\mathbf{M}_{l,i}\cdot\mathbf{M}_{l,j} \neq \mathbf{0}, \forall i,j \in \{1,...,N\}$). λ_i and a_i are the hyper-parameter used to control the model size while performing subnet sampling. $\mathcal{L}_i$ is just the normal cross-entropy loss for subnet-i, while $\tilde{\mathcal{L}}$ is subnet sampling loss will be discussed in Eq. (3). Each step will be elaborated in the following subsections.

A. Subnets generation

As the initial step, subnets generation is critical to ensure the obtained dynamic DNN can perform well with its each subnet. We discussed in Section II-A that Group Lasso technique is widely used to perform channel-wise DNN structured pruning in [8]. In this work, we propose to utilize an optimized Group Lasso to sample the subnet in a channel-wise fashion, targeting to achieve hardware-friendly structured pruned subnet group, instead of uniform structures.

Our optimization is mainly to address the issue that a model aggressively pruned by Group Lasso normally counters obvious accuracy degradation, which is a big issue in our subnet generation since the generated subnet group needs to contain different model sizes with various degrees of pruning. As described in Eq. (1), Group Lasso is a weight penalty

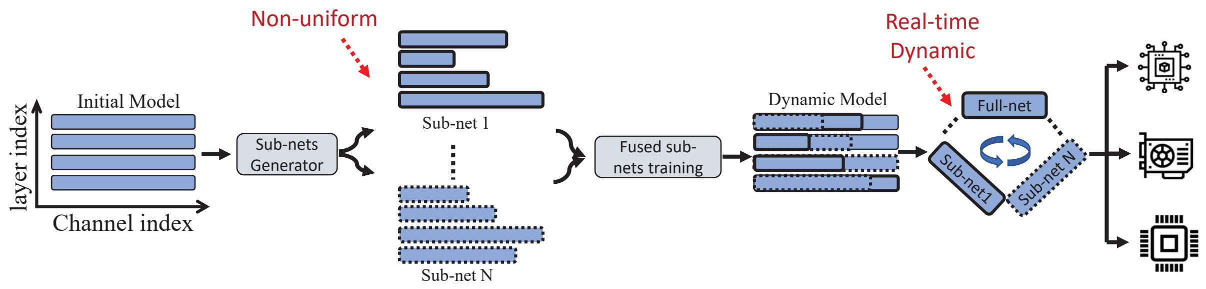


Figure 2: Overview of proposed framework. The subnets generator samples many subnets (1 to N) from the initial model with different model capacities. Note that, those sampled subnets are **partially sharing** the weights as indicated by the overlapped channel index. Then, through the followed specially designed training/optimization step, the initial model can act as a *dynamic model*, while each subnet can perform inference independently at different power, speed, accuracy.

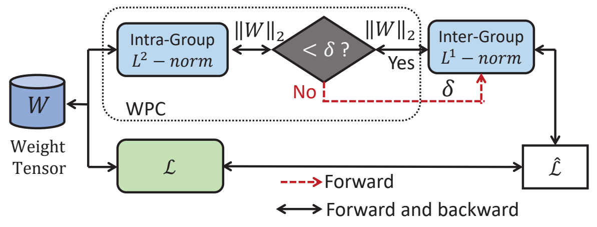


Figure 3: The overview of weight penalty clipping with self-adapting threshold.

term which consistently act on all the weights. Based on our preliminary experiments, the L-1 norm of weights of compact model (i.e., with less number of channels) are prone to be greater than redundant counterpart (i.e., with more number of channels). Our hypothesis is that the accuracy degradation caused by Group Lasso is partially attributed by the consistent weight penalty on those non-zero/non-sparse weight which are prone to be in larger magnitudes. Therefore, we propose a clipped Group Lasso with adaptive Weight Penalty Clipping (WPC) as the countermeasure for the conditional non-ideal weight penalty.

To incorporate the weight penalty clipping term, the loss function with conventional Group Lasso term in Eq. (1) can be reformatted as:

$$\tilde{\mathcal{L}} = \mathcal{L}(f(\boldsymbol{x}; \{\boldsymbol{W}_l\}_{l=1}^{L}), \boldsymbol{t}) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{C_l} \underbrace{\min\big(||\boldsymbol{W}_{l,i}||_2; \delta_l\big)}_{\text{Weight Penalty Clipping}} \tag{3}$$
$$s.t. \quad \delta_l = a \cdot \frac{1}{C_l} \sum_{i=1}^{C_l} ||\boldsymbol{W}_{l,i}||_2$$

where δ_l denotes the layer-wise clipping threshold, which is utilized to mitigate the intra-group L_2-norm penalty on large weights, and a is scaling coefficient. Note that, once the intra-group L_2-norm penalty of $\boldsymbol{W}_{l,i}$ is clipped, the inter-group L_1-norm penalty is clipped as well. In each training iteration, the updated weights are used to the loss function as shown in Fig. 3. Then after intra-group L_2-norm calculation, by comparing with a threshold δ_l, WPC will decide whether the corresponding $||\boldsymbol{W}_{l,i}||_2$ will be used on loss function and go backward. Considering two cases:

- When $||\boldsymbol{W}_{l,i}||_2 \geq \delta_l$, it indicates that weights in $\boldsymbol{W}_{l,i}$ are relatively large (i.e., important) which are not supposed to be pruned by the Group Lasso term in. Then, the weight penalty clipping is performed which replaces the weight penalty of $||\boldsymbol{W}_{l,i}||_2$ in $\hat{\mathcal{L}}$ with δ_l. Hereby, we have to highlight that δ_l is treated as a constant, where its calculation is removed from the backward computation graph.
- When $||\boldsymbol{W}_{l,i}||_2 < \delta_l$, we keep the weight penalty of $||\boldsymbol{W}_{l,i}||_2$ in its original value, thus the Group Lasso term can continuously affect $\boldsymbol{W}_{l,i}$ and prune the weights in group-wise fashion.

As illustrated above, the main difference with conventional Group Lasso is that our method only add weight penalty to those "unimportant weights" (smaller magnitude) and skip the important weight by utilizing WPC.

In virtue of training target DNN with the loss function proposed in Eq. (3), we are able to get highly compact/sparse network with minimal accuracy degradation in comparison to full-model baseline. Note that, since our main interest is only the architecture of subnet instead of the exact weight value, the subnet sampling is performed via directly counting the number of non-sparse output channels of each layer ($\{m_{l,i}\}_{l=1}^{L}$, $m_{l,i} \in \{1, 2, ..., q\}$, where q is the number of output channels of l-th layer in given full-model), following the data-flow within the forward propagation path. Then, the subnet sampling can be equivalently viewed as the weight mask generation:

$$\mathbf{M}_{l,i} = \mathbf{1}_{\text{w}\neq 0} \tag{4}$$

Through choosing the different λ_i, we are able to obtain a set subnets in various model size with non-uniform selected channels (Fig. 6 and Fig. 7).

B. *Fused subnets training*

For achieving the goal that each sampled subnet can independently perform the inference task, we propose to leverage the fused subnet training which is a multi-objective training method (Eq. (2)). As the training procedure depicted in Fig. 4, in each iteration, every subnet will go forward and backward one time to calculate loss and gradients. Then the gradients will be accumulated to update the weight by using a given optimizer (e.g., SGD). Although the training cost is increased compared with the conventional single model training scenario, it still less engineer-cost than those subnets that are trained independently. Moreover, user can trade off between power, speed, computing load and accuracy on-the-fly after deployment through executing a proper subnet, depending on the requirements or specifications of the given system by using such dynamic model. In addition, inspired by [20], we utilize additional batch-norm layers for each subnet to mitigate accuracy loss. The detailed algorithm is listed in Algorithm 1.

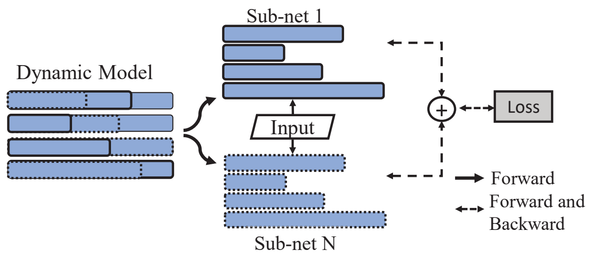


Figure 4: The overview of fused subnets training.

Algorithm 1 Fused subnets training method

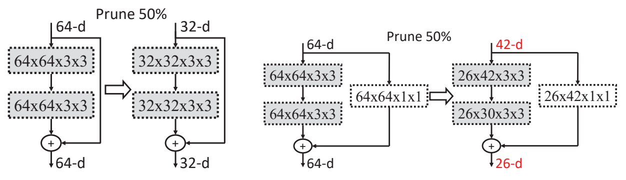
```
Require: Given a target DNN, its subnets are sampled by
    proposed Clipped-Lasso.
1: for $i \leftarrow 1, n_{\text{iters}}$ do
2:     for subnet $i$ in dynamic model do
3:         Compute loss: $\mathcal{L}_i\Big(f(\boldsymbol{x}; \{\mathbf{W}_l \cdot \mathbf{M}_{l,i}\}_{l=1}^{L}), \boldsymbol{t}\Big)$
4:         Compute and accumulate gradients
5:     end for
6:     update weights
7: end for
```

IV. Experiments

A. Experiment Setup

1) Dataset and training configurations: In this work, we take the classic image classification task as an example to examine the performance of our proposed technique. Two datasets are used in this work, which are CIFAR-10 [22] and ImageNet [23]. For CIFAR-10, we adopt the ResNet-20 [24]. We train the network using momentum SGD optimizer, where the initial learning rate is 0.1, which is scaled by 0.1 at epoch 80, 120, 160 respectively. The data argumentation is identical to the configuration adopted in [24]. Except the full net size, we sample three different subnets, where the λ is 0.01, and the scaling coefficient a are 0.3, 0.75, 1.5 respectively. For ImageNet experiments, we use the ResNet-18 [24] architecture and identical data augmentation used in [24]. We train the network using SGD optimizer, where the initial learning rate is 0.1, which is scaled by 0.1 at epoch 30, 60, 90 respectively. To sample three subnets, the values of λ are all 0.01 and the scaling coefficient a are 0.25, 0.5, 1.5 respectively.

2) Structure modification for non-uniform subnet sampling: Different from other single-pass network structures, e.g., AlexNet or VGG, ResNet utilizes short-cut connections to jump over some layers. In this case, the number of input and output channels of each block needs to be consistent. Thus it is hard to sample the convolutional layers non-uniformly. To solve this problem, we replace the identity function in the short-cut connection with a 1×1 convolution layer. Fig. 5 shows an example that the 1×1 convolution layer is helpful to set the different number of input and output channels.



(a) Conventional basic block (b) Our basic block

Figure 5: Basic block modification for non-uniform sampling.

B. Results

1) CIFAR-10: The experiments results is listed in Table I. we sample four subnets with different model sizes, named as subnet(1-4) respectively. To conduct fair comparison, the FLOPS (10^6) and number of parameters (10^4) of each subnet for three different methods are shown. As discussed in Section III, our proposed framework includes two steps: **Subnets generation** and **Fused subnets training**. First, to show the efficiency of the weight penalty clipping method in **Subnets generation**, we report the results of those four subnets that are trained individually from scratch. Thus it represents a one-time sampled network without dynamic inference. Our proposed clipped-Lasso method is obviously helpful to the conventional Group Lasso method. Second, we also get better results in comparison to S-NN [20] (i.e., a prior work of dynamic DNN) individual results, which samples its subnets in a naive uniform fashion. The result shows that the non-uniform structure provides better accuracy than the uniform one with the similar model size. Third, better accuracy is achieved of our method compared with S-NN for the dynamic results.

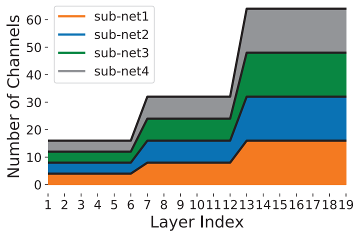
Subnets		subnet1	subnet2	subnet3	subnet4
Group Lasso	Parameters	1.86	5.74	15.02	26.83
	FLOPS	2.71	7.91	22.27	43.49
	Individual	76.5	85.5	89.1	91.4
S-NN	Parameters	1.69	6.74	15.14	26.83
	FLOPS	2.62	10.26	22.91	40.58
	Individual	80.1	86.5	89.5	91.3
	Dynamic	79	85.4	88.6	89.7
Ours	Parameters	1.22	6.67	14.39	28.19
	FLOPS	2.21	9.56	19.75	43.49
	Individual	81.3	87	89.3	91.4
	Dynamic	80.7	86.3	88.4	89.9

Table I: Inference accuracy (%) comparison of ResNet20 on CIFAR-10. 'Individual' indicates that the subnets are trained independently from scratch. To the contrary, 'Dynamic' means network to be trained by our multi subnets training method as illustrate in Section III-B.

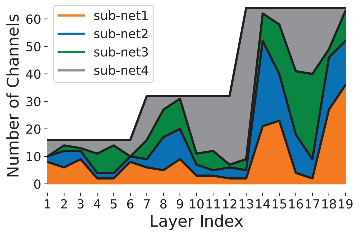
2) ImageNet: Similar to CIFAR-10 experiment, four non-uniform subnets are sampled. As shown in Table II, with the smaller number of parameters(10^6) and FLOPS (10^8) of each subnet, our method achieves almost same or better accuracy in both individual and dynamic networks.

C. Pruned result visualization

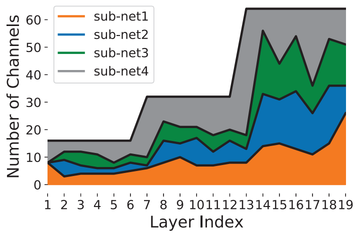
Many works have been proposed to conduct pruning by using different pruning criterion as mentioned in Section II-A.



(a) S-NN



(b) Group Lasso



(c) Ours

Figure 6: The number of channels of each subnet for ResNet20 on CIFAR-10 dataset that sampled by three different methods. All the layers are 3×3 convolutional layers. The y-axis is the corresponding weight channels in each layer.

Subnets		subnet1	subnet2	subnet3	subnet4
S-NN	Parameters	0.83	3.05	6.68	11.68
	FLOPS	1.35	4.83	10.4	18.14
	Individual	49.9	61.1	66.7	69.7
	Dynamic	48.7	60.9	66.6	69.4
Ours	Parameters	0.66	2.73	5.14	12.2
	FLOPS	0.89	3.97	7.17	19.8
	Individual	50.1	62.6	66.9	71.4
	Dynamic	48.4	61.8	66.8	69.8

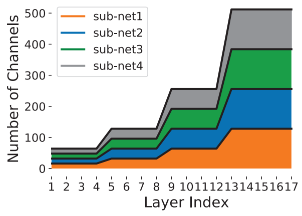
Table II: Inference accuracy (%) of ResNet18 on ImageNet

However, there is no standard learning scheme that is considered as a clear winner. We illustrate the pruned structures learned by our method as shown in Fig. 6 and Fig. 7 to provide some heuristic for pruning and network architecture search (NAS) exploration, which is summarized as follows:

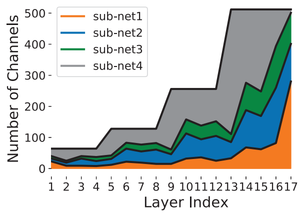
First, comparing with S-NN uniform subnet structure, we observe that all the three pruned subnets almost have the same number of channels in the first layer and gradually increasing number of channels in the last layers, which implies that a large enough fist layer is needed , as well as last several layers, to extract more information towards high accuracy. Second, significant peeks are generated after a down sampling operation. Down sampling reduces the feature map size which needs more channels to carry the same amount of information. Fig. 6 shows that conv8 and conv14 are the layers where larger number of channels are sampled after feature map degradation. A similar phenomenon is also found in the ResNet18 for ImageNet dataset as shown in Fig. 7. Third, comparing with Group Lasso method, our method is more balanced in each layer stage. To the contrary, Group Lasso method samples some extreme narrow layers. It can be considered that too narrow in some certain layers will hurt the information transformation.

D. Hardware performance

Fig. 8 depicts the results on two hardware platforms: INTEL Xeon CPU and NVIDIA Titan-Xp GPU. Our method improves the dynamic trade-off between accuracy and latency by a significant margin on both CPU and GPU comparing with S-NN. It reveals that the uniform subnets sampling has limited impact on efficiency improvement. Our method, involving non-uniform subnet structures, can efficiently improve the performance, especially in the relatively small model size. It



S-NN



Ours

Figure 7: The number of channels of each subnets for ResNet18 on ImageNet dataset that sampled by three different methods. All the layers are 3×3 convolution layers. The y-axis is the corresponding weight channels in each layer.

is noteworthy that the performance of subnet with full model size is a little bit worse than S-NN, since our model has larger parameters as illustrated in Fig. 5.

V. Discussion

A. Arbitrary subnets training

We extend multi-subnets to arbitrary subnets training. Inspired by [21], we set the low bound and high bound (full model size) for the subnets structures setting to improve the performance. Different from their uniform structures for all subnets, we use our clipped Lasso based structured weight pruning method to create a non-uniform low bound. For the middle arbitrary subnets, we increase the number of channels in each layer using the same ratio. We test our method using ResNet20 on CIFAR10 dataset. The low bound is subnet1 as shown in Table I. Fig. 9 shows the results with nine different subnets. We achieve better accuracy at most subnets, especially in the tight subnets part.

B. Why our clipped-Lasso method is better than Group Lasso?

[25] mentioned there are two requirements should be met for pruning: (1) the norm deviation of the filters should be large; (2) the minimum norm of the filters should be small. As shown in Fig. 10, comparing with conventional Group Lasso, norm distribution of our method has larger norm deviation, which means that norm distribution becomes suitable for gradually pruning during training. Obviously, important and

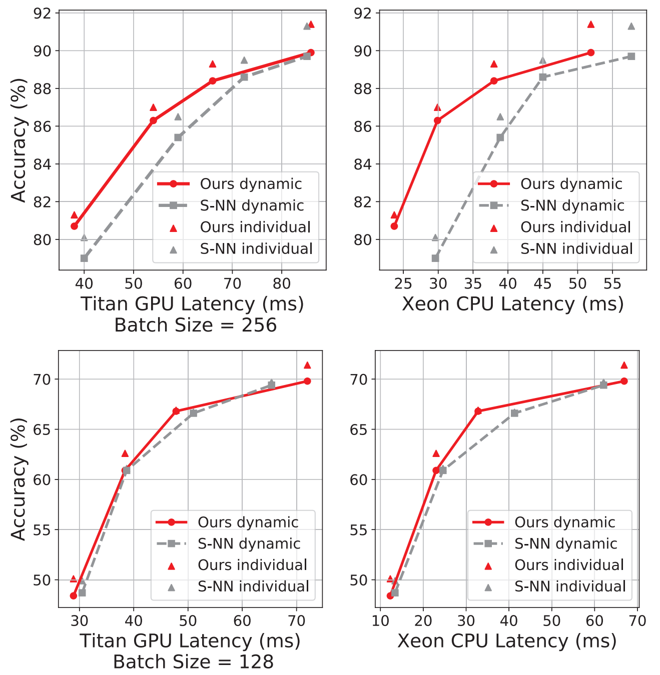


Figure 8: The trade off between latency and accuracy on Titan GPU and Xeon CPU, for (top) ResNet-20 on CIFAR-10 and (bottom) ResNet-18 on ImageNet.

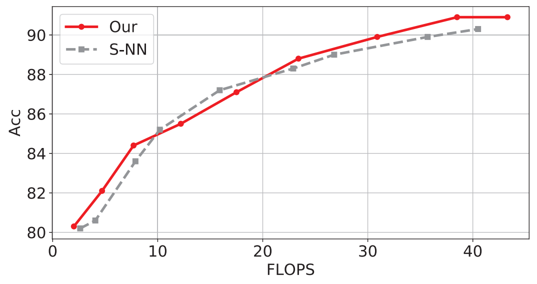


Figure 9: Nine dynamic subnets for ResNet20 trade-off between accuracy and FLOPS

unimportant weight channels can be clearly distinguished according to the absolute value of the norm for our method.

VI. Conclusion

In this work, we target to construct a dynamic DNN structure through a novel sub-network sampling method via non-uniform channel selection. Experiments on CIFAR-10 and ImageNet both validate the effectiveness of the method. Beyond that, we test the inference latency for each subnet on Titan GPU and Xeon CPU to show the trade-off between accuracy and latency.

VII. Acknowledgement

This work is supported in part by the National Science Foundation under Grant No.2005209, No. 1931871 and Semiconductor Research Corporation nCORE

References

[1] Y. LeCun *et al.*, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

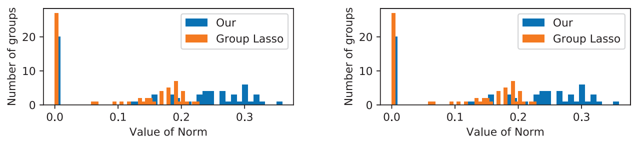


Figure 10: Norm based criterion on ResNet20 for CIFAR-10. (Left) is the conv8 layer and (Right) is the conv15 layer. X-axis represents the norm value of each channel in a layer. Y-axis is the number of channels.

[2] A. G. Howard *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
[3] M. Sandler *et al.*, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
[4] S. Han *et al.*, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” *ICLR*, 2015.
[5] I. Hubara *et al.*, “Quantized neural networks: Training neural networks with low precision weights and activations,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.
[6] E. L. Denton *et al.*, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *NIPS*, 2014, pp. 1269–1277.
[7] S. Han *et al.*, “Learning both weights and connections for efficient neural network,” in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
[8] W. Wen *et al.*, “Learning structured sparsity in deep neural networks,” in *Advances in neural information processing systems*, 2016, pp. 2074–2082.
[9] G. Hinton *et al.*, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
[10] H. Li *et al.*, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.
[11] D. Molchanov *et al.*, “Variational dropout sparsifies deep neural networks,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2498–2507.
[12] B. Zoph *et al.*, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.
[13] C. Liu *et al.*, “Progressive neural architecture search,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.
[14] H. Liu *et al.*, “Darts: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018.
[15] Z. Liu *et al.*, “Learning efficient convolutional networks through network slimming,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.
[16] M. Yuan *et al.*, “Model selection and estimation in regression with grouped variables,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.
[17] L. Liu *et al.*, “Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[18] X. Wang *et al.*, “Skipnet: Learning dynamic routing in convolutional networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.
[19] G. Huang *et al.*, “Multi-scale dense networks for resource efficient image classification,” *arXiv preprint arXiv:1703.09844*, 2017.
[20] J. Yu and othe, “Slimmable neural networks,” *arXiv preprint arXiv:1812.08928*, 2018.
[21] J. Yu *et al.*, “Universally slimmable networks and improved training techniques,” *arXiv preprint arXiv:1903.05134*, 2019.
[22] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.
[23] J. Deng *et al.*, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
[24] K. He *et al.*, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[25] Y. He *et al.*, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349.