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
{llyang166, 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 runtime (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:

$$\hat{\mathcal{L}} = \mathcal{L}(f(x; \{W_l\}_{l=1}^L), t) + \lambda \sum_{l=1}^L \sum_{i=1}^{G_l} \mathcal{P}(W_{l,i})$$

(1)

where $f(x; \{W_l\}_{l=1}^L)$ computes the outputs of DNN parameterized by $\{W_l\}_{l=1}^L$ w.r.t the input $x$. $\mathcal{L}(\cdot, \cdot)$ is the objective function of DNN (e.g., cross-entropy loss). $\mathcal{P}(W_{l,i}) = ||W_{l,i}||_2$ calculates the Euclidean norm of the indexed weight group $W_{l,i}$. The second term in the R.H.S of Eq. (1) is the $L_1$-norm of $\{\mathcal{P}(W_{l,i})\}$ (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 $\lambda$ 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:

$$\min_{\{W_l\}_{l=1}^L} \sum_{i=1}^N \mathcal{L}_i \left( f(x; (W_l \cdot M_{l,i})_{l=1}^L), t \right)$$

(2)

subject to $\{M_{l,i}\}_{l=1}^L = \arg \min \hat{\mathcal{L}}(f(x; (W_l \cdot M_{l,i})), t, \lambda_i, a_i)$

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 $W_l$ is in shape of $\mathbb{R}^{h \times k_h \times k_w \times w}$. $M_{l,i} \in \{0, 1\}^{p \times q \times 1 \times 1}$ 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., $M_{l,i} \cdot M_{l,j} \neq 0, \forall i, j \in \{1, ..., N\}$). $\lambda_i$ 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 $\hat{\mathcal{L}}$ 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
where shown in Fig. 3. Then after intra-group iteration, the updated weights are used to the loss function as go backward. Considering two cases:

- When \(||W_{l,i}||_2 \geq \delta_l\), it indicates that weights in \(W_{l,i}\) 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 \(||W_{l,i}||_2\) in \(L\) with \(\delta_l\). Hereby, we have to highlight that \(\delta_l\) is treated as a constant, where its calculation is removed from the backward computation graph.
- When \(||W_{l,i}||_2 < \delta_l\), we keep the weight penalty of \(||W_{l,i}||_2\) in its original value, thus the Group Lasso term can continuously affect \(W_{l,i}\) 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:

\[
M_{l,i} = \mathbf{1}_{W_{l,i} \neq 0}
\]

Through choosing the different \(\lambda_i\), 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.

Algorithm 1 Fused subnets training method

Require: Given a target DNN, its subnets are sampled by proposed Clipped-Lasso.
1: for $i \leftarrow 1, N_{nets}$ do
2: for subnet $i$ in dynamic model do
3: Compute loss: $L_i(f(x; \{W_i, \beta_i\}_1^t), t)$
4: Compute and accumulate gradients
5: end for
6: compute and accumulate gradients
7: end for

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 augmentation is identical to the configuration adopted in [24]. Except the full net size, we sample three different subnets, where the $\lambda$ 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 $\lambda$ 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 $1 \times 1$ convolution layer. Fig. 5 shows an example that the $1 \times 1$ convolution layer is helpful to set the different number of input and output channels.

![Figure 4: The overview of fused subnets training.](image)

![Figure 5: Basic block modification for non-uniform sampling.](image)

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 $(10^5)$ and number of parameters $(10^7)$ 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.

<table>
<thead>
<tr>
<th>Subnets</th>
<th>subnet1</th>
<th>subnet2</th>
<th>subnet3</th>
<th>subnet4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lasso</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>1.86</td>
<td>2.71</td>
</tr>
<tr>
<td>Individual</td>
<td>76.5</td>
<td>85.3</td>
<td>89.1</td>
<td>91.6</td>
</tr>
<tr>
<td>S-NN</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>1.69</td>
<td>2.62</td>
</tr>
<tr>
<td>Individual</td>
<td>80.1</td>
<td>86.3</td>
<td>89.5</td>
<td>91.3</td>
</tr>
<tr>
<td>Ours</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>1.22</td>
<td>2.21</td>
</tr>
<tr>
<td>Individual</td>
<td>81.3</td>
<td>87.3</td>
<td>89.3</td>
<td>91.4</td>
</tr>
</tbody>
</table>

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 $(10^5)$ of each subnet, our method achieves almost same or better accuracy in both individual and dynamic networks.

<table>
<thead>
<tr>
<th>Subnets</th>
<th>subnet1</th>
<th>subnet2</th>
<th>subnet3</th>
<th>subnet4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>2.71</td>
<td>5.74</td>
</tr>
<tr>
<td>Individual</td>
<td>76.5</td>
<td>85.3</td>
<td>89.1</td>
<td>91.6</td>
</tr>
<tr>
<td>S-NN</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>2.62</td>
<td>10.26</td>
</tr>
<tr>
<td>Individual</td>
<td>80.1</td>
<td>86.3</td>
<td>89.5</td>
<td>91.3</td>
</tr>
<tr>
<td>Ours</td>
<td>Parameters</td>
<td>FLOPS</td>
<td>1.22</td>
<td>2.21</td>
</tr>
<tr>
<td>Individual</td>
<td>81.3</td>
<td>87.3</td>
<td>89.3</td>
<td>91.4</td>
</tr>
</tbody>
</table>

Table II: Inference accuracy (%) comparison of ResNet20 on ImageNet. '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.

C. Pruned result visualization

Many works have been proposed to conduct pruning by using different pruning criterion as mentioned in Section II-A.
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 $3 \times 3$ convolutional layers. The y-axis is the corresponding weight channels in each layer.

(a) S-NN
(b) Group Lasso
(c) Ours

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

<table>
<thead>
<tr>
<th>Subnets</th>
<th>subnet1</th>
<th>subnet2</th>
<th>subnet3</th>
<th>subnet4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-NN</td>
<td>Parameters</td>
<td>0.83</td>
<td>3.05</td>
<td>6.68</td>
</tr>
<tr>
<td></td>
<td>FLOPS</td>
<td>1.35</td>
<td>4.83</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>Individual</td>
<td>49.9</td>
<td>61.1</td>
<td>66.7</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>48.7</td>
<td>60.9</td>
<td>66.6</td>
</tr>
<tr>
<td>Ours</td>
<td>Parameters</td>
<td>0.66</td>
<td>2.73</td>
<td>5.14</td>
</tr>
<tr>
<td></td>
<td>FLOPS</td>
<td>0.89</td>
<td>3.97</td>
<td>7.17</td>
</tr>
<tr>
<td></td>
<td>Individual</td>
<td>50.1</td>
<td>62.6</td>
<td>66.9</td>
</tr>
<tr>
<td></td>
<td>Dynamic</td>
<td>48.4</td>
<td>61.8</td>
<td>66.8</td>
</tr>
</tbody>
</table>

Figure 7: The number of channels of each subnets for ResNet18 on ImageNet dataset that sampled by three different methods. All the layers are $3 \times 3$ convolutional 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
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


