Dynamic Neural Network to Enable Run-Time Trade-off between Accuracy and Latency

Li Yang
Arizona State University
Tempe, Arizona
lyang166@asu.edu

Deliang Fan
Arizona State University
Tempe, Arizona
dfan@asu.edu

ABSTRACT

To deploy powerful deep neural network (DNN) into smart, but resource limited IoT devices, many prior works have been proposed to compress DNN to reduce the network size and computation complexity with negligible accuracy degradation, such as weight quantization, network pruning, convolution decomposition, etc. However, by utilizing conventional DNN compression methods, a smaller, but fixed, network is generated from a relative large background model to achieve resource limited hardware acceleration. However, such optimization lacks the ability to adjust its structure in real-time to adapt for a dynamic computing hardware resource allocation and workloads. In this paper, we mainly review our two prior works [13, 15] to tackle this challenge, discussing how to construct a dynamic DNN by means of either uniform or non-uniform sub-nets generation methods. Moreover, to generate multiple non-uniform sub-nets, [15] needs to fully retrain the background model for each sub-net individually, named as *multi-path* method. To reduce the training cost, in this work, we further propose a *single-path* sub-nets generation method that can sample multiple sub-nets in different epochs within one training round. The constructed dynamic DNN, consisting of multiple sub-nets, provides the ability to run-time trade-off the inference accuracy and latency according to hardware resources and environment requirements. In the end, we study the the dynamic DNNs with different sub-nets generation methods on both CIFAR-10 and ImageNet dataset. We also present the run-time tuning of accuracy and latency on both GPU and CPU.

CCS CONCEPTS

• **Computer systems organization** → Real time system; • **Networks** → Dynamic neural network.

KEYWORDS

dynamic neural networks

ACM Reference Format:
Li Yang and Deliang Fan. 2021. Dynamic Neural Network to Enable Run-Time Trade-off between Accuracy and Latency. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3394885.3431628

1 INTRODUCTION

Recently, Deep Neural Network (DNN) evolves into deeper layers, wider channels, and denser connections to further improve performance, while rapidly growing network parameters and computation complexity. Such development trend rises great challenge to deploy into power and resource limited hardware platform, such as IOT devices, mobile phone, etc. To tackle this issue, various DNN compression techniques to reduce the network size and computation cost have been proposed, such as weight and activation quantization [4, 5, 8], network pruning [2, 14], convolution decomposition [7, 11], etc.

However, such model compression techniques mainly have two limitations: 1) given a DNN model, it needs to be optimized specially for each target hardware platform owing to their unique computing resources; 2) for one specific hardware platform, although a compact network can be generated by utilizing these techniques, it is fixed after deployment. Oppositely, in the real-world applications, both the hardware platform and workload environment are dynamic. For example, the mobile phone could be in high performance model or battery saving mode. Another example could be, owing to the dynamic sensing efforts or communication channels, the workloads of real-time machine learning computing are also dynamic with streaming input data. Thus, it urges the need to construct a new dynamic DNN model with the ability to adjust its structure on-the-fly to fast adapt for a dynamic computing hardware resource allocation and workloads.

Based on our two previous research works published in ASP-DAC'20 and DAC'20 [13, 15] aiming to tackle this challenge, in this work, we mainly review the general dynamic neural network framework, which consists of multiple sub-nets that can run-time multiplex between them, tuning the inference accuracy according to hardware resources and environment requirement. Generally, the overflow of the framework includes two successive phases: *Sub-nets generation* and *Fused sub-nets training*. In the first phase, multiple sub-nets are sampled from a background model. Two sub-nets generation methods (i.e. *uniform* and *non-uniform*) are discussed in [13, 15] respectively. Specifically, [13] presents a *uniform* sub-net sampling method, which manually sets the channel width of all layers to be a constant ratio (e.g., 0.25, 0.5), along with utilizing knowledge distillation [6] to improve performance. Alternatively, [15] generates *non-uniform* sub-nets, inspired by the model pruning [10, 12**?**] that different layers in a DNN model may have different sensitivities, resulting in non-uniform layer-wise sparsity. In the

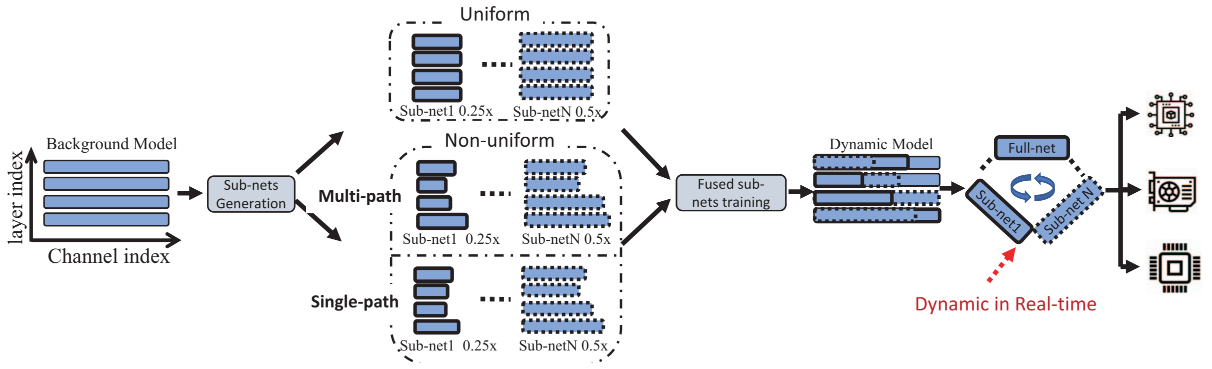


Figure 1: Overview of two-phase dynamic neural network framework for both uniform and non-uniform sub-nets sampling. In the first phase, #N sub-nets are generated from the background model concerning difference sizes. Note that, the overlapped weights of sub-nets are partially shared w.r.t the same weight channels location. Then in the followed second phase, fused sub-nets training is leveraged to construct the a single full-size *dynamic model*. In the end, such dynamic model with #N sub-nets can run-time trade-off the inference accuracy via selecting different sub-nets according to hardware and workload dynamics.[13, 15].

second phase, to construct a dynamic network, the sub-nets sampled by different methods above are trained by a general method that are fused into a cross entropy loss function for multi-subnets optimizations.

Moreover, to sample multiple non-uniform sub-nets, [15] needs to fully retrain the background model for each sub-net individually, named as *multi-path* sampling method. To reduce the training cost, in this work, we further propose a *single-path* sub-nets sampling method that can generate multiple sub-nets in different epochs within only one training round.

2 DYNAMIC NEURAL NETWORK

In this section, we first describe the overflow of the dynamic neural network framework and then discuss the various uniform and non-uniform sampling methods [13, 15] in detail. As shown in fig.1, the overflow of the dynamic neural network framework consists of two phases:

- **Sub-nets generation:** In this phase, multiple sub-nets are generated from the background model concerning different model sizes. Our previous research papers [13, 15] generate the sub-nets in uniform and non-uniform formats respectively. In terms of non-uniform sub-nets generation, the method in [15] samples each sub-net by fully re-training the background model once, named multi-path non-uniform sub-nets generation. To reduce the training cost, in this work, we propose a single-path sub-nets generation method, which generates multiple sub-nets from different training epochs within only one training round.
- **Fused sub-nets training:** To construct a dynamic model where the sub-nets generated from phase 1 can be executed independently, two fusion techniques are utilized: 1) **weights fusion**: the overlapped weights of sub-nets are partially shared within a dynamic model. 2) **optimization fusion**: these sampled sub-nets are fused into a loss function with multi-objective optimization [16]. By doing so, the dynamic model could switch between different sub-nets for run-time tuning between speed, power consumption and accuracy.

2.1 Sub-nets generation

To better understand the method, we clarify the notation used in this work. Given a deep neural network $f(\boldsymbol{x}; \mathbf{W})$ with L layers, where $\boldsymbol{x}$ is the data and $\mathbf{W}_l$ means the weight parameters. We denote the ith sub-net as a set of layer-wise weight binary mask $\{\mathbf{M}_{l,i}\}_{l=1}^{L} \in \{0, 1\}$.

2.1.1 Uniform sub-nets generation method. As illustrated in fig.2(a), [13, 16] sample each sub-net by manually setting a channel-width ratio, which is a constant factor (e.g., 0.25×, 0.5×) shared by all layers. Such constrain can be formatted as

$$R(\mathbf{M}_{0,i}) = R(\mathbf{M}_{1,i}) = \dots = R(\mathbf{M}_{L,i}) \tag{1}$$

Where R represents the channel-width ratio which is a fixed constant among all layers for the ith sub-net. For example, the 0.25× sub-net sequentially selects quarter number of weight output channels for all layers except the last one. Also the 0.5× sub-net selects half number of weight output channels sequentially in same order as 0.25× sub-net, which means 0.25× sub-net is a complete subset of the 0.25× sub-net.

2.1.2 Non-uniform sub-nets generation method. Different from the uniform sub-nets generation which simply selects sub-nets by hand, the non-uniform dynamic neural network utilizes optimized group Lasso-based regularization method, named clipped Lasso [14, 15], to learn the sub-nets structures.

Generation via clipped Lasso regularization. Group Lasso is a general regularization-based method for structured pruning [12], which acts as a additional term in the loss function to penalize all the weights. Different from that, the clipped group Lasso only acts on the weights with larger magnitude, which are considered as "important" weights. To achieve this, an adaptive Weight Penalty

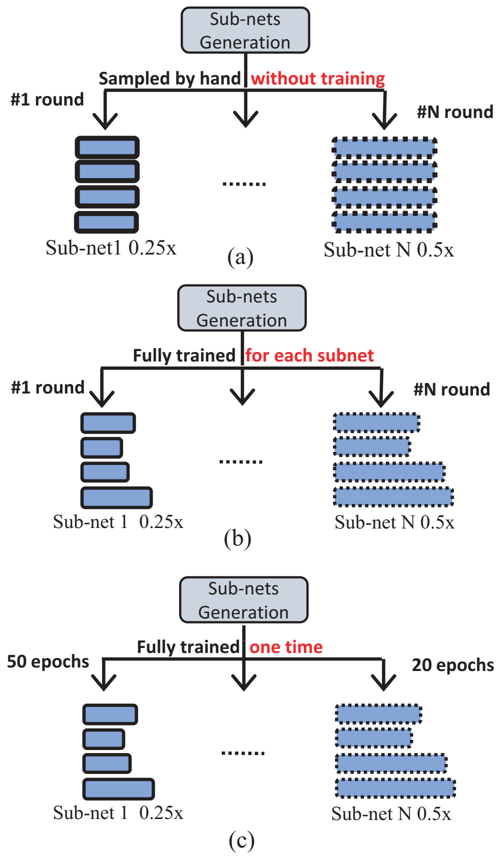


Figure 2: Overview of uniform (a), multi-path (b) and single-path (c) non-uniform sub-nets generation.

Clipping (WPC) in [15] is adopted, which is given by:

$$\mathcal{L}^* = \mathcal{L}(f(\boldsymbol{x}; \{\mathbf{W}_l\}_{l=1}^L)) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{G_l} \underbrace{\min(||\mathbf{W}_{l,i}||_2; \tau_l)}_{\text{WPC}} \tag{2}$$

$$\text{s.t.} \quad \tau_l = \beta \cdot \frac{1}{G_l} \sum_{i=1}^{G_l} ||\mathbf{W}_{l,i}||_2$$

Where $\mathcal{L}$ is a general loss function(i.e. cross-entropy), and the second term in the loss function is a weight penalty to perform sampling by pruning. λ is a hyper-parameter to adjust the weight regularization capacity. Moreover, the adaptive threshold τ_l is utilized to clip the penalty on weights with larger L_2-norm $||W_{l,i}||_2$. The clipping threshold τ_l is equal to the mean of $||W_{l,i}||_2$ with a scaling factor $\beta \in (0, 1)$. By doing so, the weight penalty will be only acted on the "important" weights, which has larger L_2-norm $||W_{l,i}||_2$ in comparison to the threshold τ_l.

Multi-path non-uniform sub-nets generation. As illustrated in fig.2(b), to generate each sub-net, the background model needs to be retrained for each subnet with clipped group Lasso regularization as depicted in eq.2. It can be formatted as:

$$\{\mathbf{M}_{l,i}\}_{l=1}^L = \arg\min \mathcal{L}^*(f(\boldsymbol{x}; \{\mathbf{W}_l \cdot \mathbf{M}_{l,i}\}), \lambda_i, \beta_i) \tag{3}$$

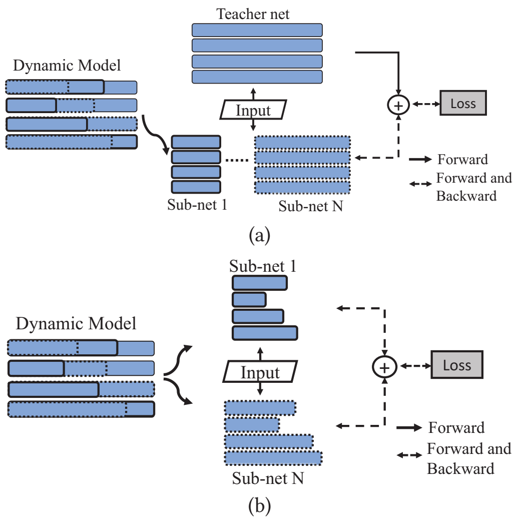


Figure 3: Fused sub-nets training pipeline for (a) uniform[13] and (b) non-uniform sub-nets[15].

By choosing different hyper-parameter λ_i and β_i for ith sub-net, multiple non-uniform sub-nets with different model sizes can be generated [14].

Single-path non-uniform sub-nets generation. To reduce the training cost of the multi-path non-uniform sub-nets generation, in this work, we propose to sample multiple sub-nets within only one training round. Such method is inspired by an important observation: *given a fixed hyper-parameter λ_i, the non-sparse background model is pruned progressively during training with clipped lasso regularization, which means multiple sub-nets with different model sizes can be generated in difference epochs.* It can be formatted as:

$$\{\mathbf{M}_{l,i}\}_{l=1}^L = \arg\min \mathcal{L}^*(f(\boldsymbol{x}; \{\mathbf{W}_l \cdot \mathbf{M}_{l,i}\}), \lambda, \beta, e_i) \tag{4}$$

where e_i represents the epoch to select the corresponding ith sub-net during one-round training (e.g., 20 epoch, 50 epoch).

2.2 Fused sub-nets training

After multiple sub-nets are sampled in phase one, the fused sub-net training is utilized to fuse those sampled sub-nets into a general cross entropy loss function with multiple objective optimization. Such fused sub-nets training has two important properties: 1) **weights fusion**: the overlapped weights of different sub-nets are partially shared within a dynamic model. 2) **optimization fusion**: these sampled sub-nets are fused into a loss function with multi-objective optimization [16], which can be formatted as:

$$\min_{\{\mathbf{W}_l\}_{l=1}^L} \sum_{i=1}^{N} \mathcal{L}_i\Big(f(\boldsymbol{x}; \{\mathbf{W}_l \cdot \mathbf{M}_{l,i}\}_{l=1}^L)\Big) \tag{5}$$

The fused sub-net training pipeline is illustrated in fig.3. Same as multiple networks optimized individually, all sub-nets pass thought the forward and backward to compute their gradients. But differently, these gradients are collected to update the weights once. Although the computation complexity is increased comparing with a single network training procedure, the dynamic model greatly reduces the developer-cost of training these sub-nets individually. Furthermore, it achieves the run-time trade-off among speed, power and accuracy. So, utilizing dynamic neural network, users can adjust the execution model in real-time into a suitable sub-net w.r.t the current hardware resources and environment requirements.

As illustrated in fig. 3(a), knowledge distillation [6, 13] could be leveraged to improve accuracy, but sacrificing the memory and computation cost. The main idea behind it is to train sub-nets by mimicking a pre-trained larger model (teacher net as shown in Fig. 3(a)). Specifically, the sub-nets are not only optimized by minimizing cross entropy loss with data labels, but also by distilling information from the teacher model whose output is considered as soft label. Alternatively, as illustrated in fig. 3(b), it is also fine to directly compute a general cross entropy loss for all sampled sub-nets.

3 EXPERIMENTS

In this section, we first elaborate on the experiments configurations, then study the accuracy and latency of different methods. In the end, we analyze and visualize the sub-nets structures.

3.1 Experiment Setup

3.1.1 Dataset and training configurations. We evaluate the experiments for uniform[13] and multi-path [15] non-uniform methods on both CIFAR-10 [9] and ImageNet [1] dataset using ResNet [3] architecture. The detailed experiments setting can refer to our prior works [13, 15]. Moreover, we also test the single-path non-uniform method on CIFAR-10 using ResNet-20. Specifically, during sub-nets generation phase, we adopt SGD optimizer with the 0.01 learning rate trained by 60 epochs. We mainly select four sub-nets with different model sizes. For the fused training phase, single-path non-uniform method follows the same configuration with multi-path method [15].

3.1.2 Baseline methods. There are mainly three baseline methods for comparison purpose. First, Slimmable Neural Network (S-NN) [16], which is also a uniform dynamic neural network method without using knowledge distillation. Second, the sampled sub-nets that train from scratch individually. Third, the sub-nets that are sampled by conventional group Lasso regularization method[12].

3.2 Uniform dynamic neural network

Table 1 illustrates the accuracy results on CIFAR-10. Note that, the teacher model is also ResNet-20. We denote the uniform sub-nets generation method as 'Ours-U'. It shows that our full precision (FP) models achieve better accuracy on all selected sub-nets than S-NN. Moreover, it has negligible accuracy loss on 16-bits and 8-bits quantization on both weights and activations.

We further study the accuracy on ImageNet dataset as shown in table 2 . [15] also compares with the selected sub-nets that are trained independently under the same sizes (denoted as 'I-NN' in table 2). The results show that 'Ours-U' achieves the best accuracy for each sub-net with the same size of parameters.

Network	Width	S-NN [16]	Ours-U[13]		
		FP	FP	16-bit	8-bit
ResNet20	1.0×	89.8	91.1	91.0	91.1
	0.75×	88.4	90.2	89.8	89.8
	0.5×	85.6	87.5	87.0	86.9
	0.25×	79.5	81.0	80.7	80.6

Table 1: Inference accuracy (%) of ResNet20 on Cifar10 for uniform dynamic neural network[13].

Network	Width	I-NN	S-NN [16]	Ours-U[13]	Params(MB)
ResNet50	1.0×	76.1	76.0	76.6	25.5
	0.75×	74.7	74.9	75.4	14.7
	0.5×	72.0	72.1	72.4	6.9
	0.25×	63.8	65.0	65.2	2.0

Table 2: Inference accuracy (%) of ResNet50 on ImageNet for uniform dynamic neural network[15].

3.3 Non-uniform dynamic neural network

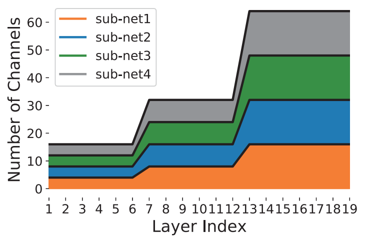
The evaluation results are illustrated in table. 3 and table. 4. Note that, we denote the multi-path non-uniform method as 'Ours-M' and single-path method as 'Ours-S'. Ours-M method achieves almost the same or better accuracy for both individual and dynamic networks, with even smaller number of parameters (10^6) and FLOPS (10^8) of each sub-net. It's worthy to note that 'Ours-S' has worse accuracy than 'Ours-M', but reducing $\sim 2.75\times$ training time.

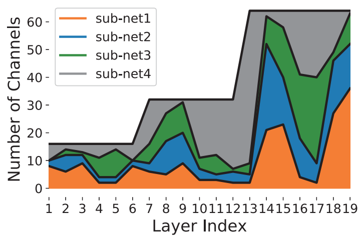
Sub-nets		subnet1	subnet2	subnet3	subnet4
Group Lasso	Parameters	1.86	5.74	15.02	28.33
	FLOPS	2.71	7.91	22.27	43.49
	Individual	76.5	85.5	89.1	91.4
S-NN[16]	Parameters	1.69	6.74	15.14	26.83
	FLOPS	2.62	10.26	22.91	40.58
	Individual	80.1	86.5	89.5	91.3
	Dynamic	79.0	85.4	88.6	89.7
Ours-M[15]	Parameters	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
Ours-S	Parameters	1.41	6.26	14.78	28.23
	FLOPS	2.48	9.11	20.07	43.49
	Individual	80.3	86.8	88.9	91.4
	Dynamic	79.5	85.9	88.2	89.9

Table 3: Accuracy (%) results on CIFAR-10 using ResNet-20. 'Individual' means that the sub-nets are trained from scratch independently. To the contrary, 'Dynamic' indicates the dynmaic model is trained via sub-nets fusion method[15]

3.4 Hardware performance

We further study the real hardware latency of dynamic neural network on two platforms: INTEL Xeon CPU and NVIDIA Titan-Xp GPU. Fig.6 illustrates the trade-off between accuracy and latency


(a) S-NN[16]


(b) Group Lasso[12]

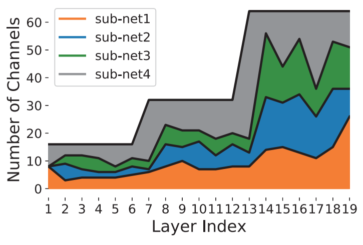

(c) Ours-M[15]

Figure 4: The sub-nets structures visualization on CIFAR-10 dataset using ResNet20[15]

Sub-nets		subnet1	subnet2	subnet3	subnet4
S-NN [16]	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-M [15]	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 4: Accuracy (%) results on ImageNet using ResNet18 for the dynamic neural network via multi-path non-uniform sub-nets generation method[15].

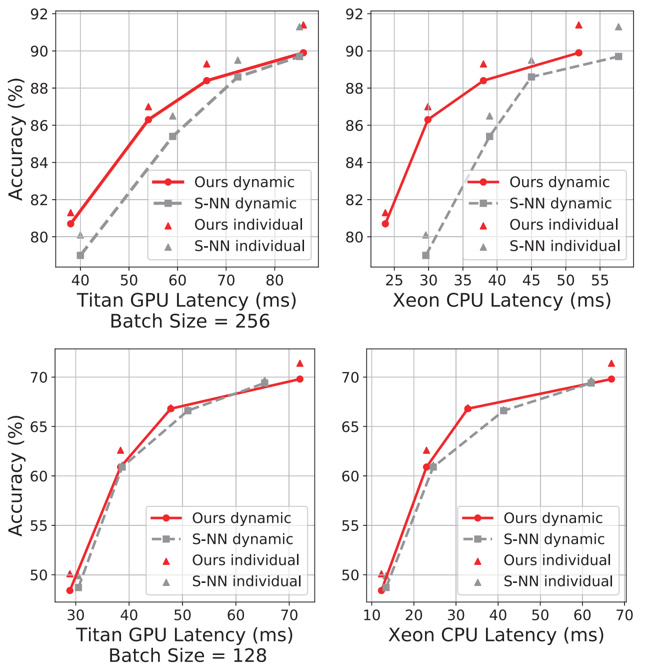
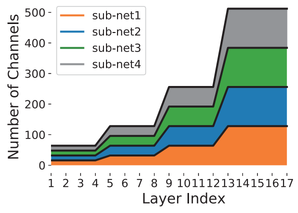


Figure 6: The trade off between latency and accuracy on Titan GPU and Xeon CPU, for (top) ResNet-20 on CIFAR-10 and (bottom) ResNet-18 on ImageNet.


S-NN[16]

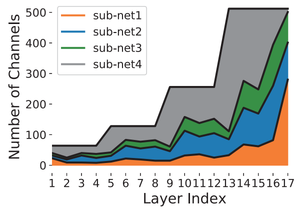

Ours-M[15]

Figure 5: The sub-nets structures on ImageNet dataset using ResNet18 [15].

on both CPU and GPU. The experiment results show that multi-path non-uniform method significantly improve the performance of dynamic trade-off.

3.5 Sub-nets structure visualization

Although many works have proposed various pruning methods resulting in different pruned model structure, there is no golden solution that is considered as the best one. As shown in fig.4 and fig.5, we study the sampled sub-nets structures via non-uniform generation method[15] to further explore network pruning and architecture search, which is described as follows:

- Except the full-size network, the first layers of all sampled sub-nets almost keep the same channel numbers, while the last layers are gradually having more number of channels with larger sub-net size. This observation implies that large enough first and last layers are needed to keep representation ability towards high accuracy.
- Significant higher channel numbers are occupied for the layer after a down sampling operation (e.g., conv8 and conv14 in fig.4). This may because that down sampling reduces the feature map dimension. Thus, more channels are needed to have the same amount of feature information.

- In comparison to conventional group Lasso method, sub-nets that sampled by clipped Lasso [15] is more balanced in each layer.

4 DISCUSSION

4.1 Teacher Model Selection

We study that how different teach models used by knowledge distillation will effect the performance of dynamic neural network[13]. As illustrated in table 5, three different teacher models are used as teacher: ResNet20/32/44. The experiment results show that, in general, the trained dynamic model with larger teacher model can achieve better performance.

Student	Width	Teacher		
		ResNet20	ResNet32	ResNet44
ResNet20	1.0x	91.1	91.4	91.9
	0.75x	90.2	90.7	91.2
	0.5x	87.5	88.6	88.7
	0.25x	81.0	82.9	82.5

Table 5: Teacher model selection for uniform sub-nets generation method[13]

4.2 Dynamic Network with Larger Number of Sub-nets

In the main experiment results above, we only show the model with four sampled sub-nets. Our methods also support much larger number of sub-nets. To show its performance, we further study a dynamic neural network with larger number of non-uniform sub-nets via multi-path method [15]. Inspired by [17], the low bound (e.g., 0.25×) and high bound (full model size) for the sub-nets are configured. As illustrated in fig.7, nine different sub-nets are evaluated on CIFAR-10 dataset using ResNet-20. The results show that better accuracy can be achieved at most sub-nets compared with S-NN.

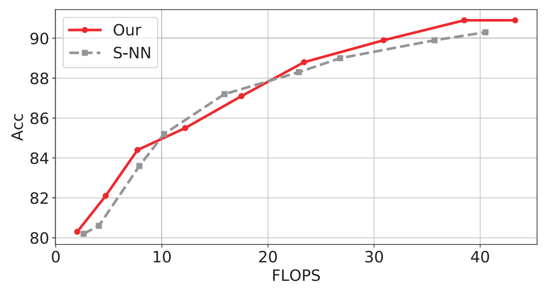


Figure 7: The trade-off between accuracy and FLOPS of a dynamic network with nine different sub-nets on ResNet20[15]

5 CONCLUSION

In this work, we explicitly review different methods to construct a dynamic neural network, which mainly includes two phases: sub-nets generation and fused sub-nets training. Two different sub-nets generation methods are discussed, which are used for the uniform and non-uniform dynamic neural networks respectively. Furthermore, to reduce the training cost of multi-path non-uniform sub-nets generation method, in this work, we also propose a single-path method. Experiments on CIFAR-10 and ImageNet both validate the effectiveness of our methods. We also demonstrate the run-time trade-off between inference accuracy and latency for dynamic neural network on Titan GPU and Xeon CPU.

6 ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation under Grant No.2005209, No.1931871, No. 2019548

REFERENCES

[1] Deng, J., et al. Imagenet: A large-scale hierarchical image database. In *CVPR* (2009), Ieee, pp. 248–255.
[2] Han, S., et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR'16*.
[3] He, K., et al. Deep residual learning for image recognition. In *Proceedings of the IEEE CVPR* (2016), pp. 770–778.
[4] He, Z., and Fan, D. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 11438–11446.
[5] He, Z., Gong, B., and Fan, D. Optimize deep convolutional neural network with ternarized weights and high accuracy. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 913–921.
[6] Hinton, G., et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[7] Howard, A. G., et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint:1704.04861* (2017).
[8] Hubara, I., et al. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research 18*, 1 (2017), 6869–6898.
[9] Krizhevsky, A., and Hinton, G. Learning multiple layers of features from tiny images. Tech. rep., Citeseer, 2009.
[10] Molchanov, D., et al. Variational dropout sparsifies deep neural networks. In *ICML* (2017), JMLR. org, pp. 2498–2507.
[11] Sandler, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR* (2018), pp. 4510–4520.
[12] Wen, W., et al. Learning structured sparsity in deep neural networks. In *NIPS* (2016), pp. 2074–2082.
[13] Yang, L., et al. A flexible processing-in-memory accelerator for dynamic channel-adaptive deep neural networks. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)* (2020), IEEE.
[14] Yang, L., et al. Harmonious coexistence of structured weight pruning and ternarization for deep neural networks. In *AAAI* (2020), pp. 6623–6630.
[15] Yang, L., He, Z., and Fan, D. Non-uniform dnn structured subnets sampling for dynamic inference. *57th Design Automation Conference (DAC)* (2020).
[16] Yu, J., et al. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).
[17] Yu, J., and Huang, T. Universally slimmable networks and improved training techniques. *arXiv preprint arXiv:1903.05134* (2019).