SPECIALIZATION OF SUB-PATHS FOR ADAPTIVE DEPTH NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a novel approach to anytime networks that can control network depths instantly at runtime to provide various accuracy-efficiency trade-offs. While controlling the depth of a network is an effective way to obtain actual inference speed-up, previous adaptive depth networks require either additional intermediate classifiers or decision networks, that are challenging to train properly. Unlike previous approaches, our adaptive depth networks require virtually no architectural changes from baseline networks. Instead, we introduce a novel training method that enforces some sub-paths of the baseline networks to have a special property, with which the sub-paths do not change the semantic level of input features, but only refine them to reduce prediction errors. Those specialized sub-paths can be skipped at test time, if needed, to save computation at marginal loss of prediction accuracy. We first formally present the rationale behind the sub-paths specialization, and based on that, we propose a simple and practical training method to specialize sub-paths for adaptive depth networks. While minimal architectural changes and training efforts are required, we demonstrate that our approach significantly outperforms non-adaptive baselines in various tasks, including ImageNet classification, COCO object detection and instance segmentation. Further, we show that the smallest sub-networks of our adaptive depth networks achieve competitive model compression effect compared to recent state-of-the-art techniques.

1 INTRODUCTION

Modern deep neural networks provide state-of-the-art performance at high computational costs, and, hence, lots of efforts have been made to leverage those inference capabilities in resource-constrained systems, such as autonomous vehicles. Those efforts include compact architectures (Howard et al., 2017; Zhang et al., 2018; Han et al., 2020), network pruning (Han et al., 2016; Liu et al., 2019), weight/activation quantization (Jacob et al., 2018), knowledge distillation (Hinton et al., 2015), to name a few. However, those approaches provide static accuracy-efficiency trade-offs that are often tailored for worst-case scenarios, and, hence, the lost accuracy cannot be recovered even if more resources become available.

Adaptive networks such as anytime networks (Huang et al., 2018; Yu et al., 2018; Wan et al., 2020) attempt to provide runtime adaptability to deep neural networks by exploiting the redundancy in either depths or widths, as shown in Figure 1, or resolutions (Yang et al., 2020). Dynamic networks (Wu et al., 2018; Li et al., 2021; 2020; Zhu et al., 2021) add additional control logic to the backbone network for input-dependent adaptation. However, these adatpive networks usually require auxiliary networks, such as intermediate classifiers or decision networks, which are challenging to train properly. Further, since adaptive networks have multiple sub-networks, embedded in a single neural network, training them incurs potentially conflicting training objectives for the sub-networks, resulting in worse performance than non-adaptive networks (Li et al., 2019).

In this work, we introduce a novel approach to anytime networks that is executable in multiple depths to provide instant runtime accuracy-efficiency trade-offs. Unlike previous adaptive depth networks, our approach does not require additional add-on networks or classifiers, and, hence, it can be applied to modern residual networks easily. While maintaining the structure of original networks, we train several sub-paths, or a sequence of residual blocks, of the network to have a special property, that preserves the semantic level of input features, and only refines them to reduce prediction errors. At

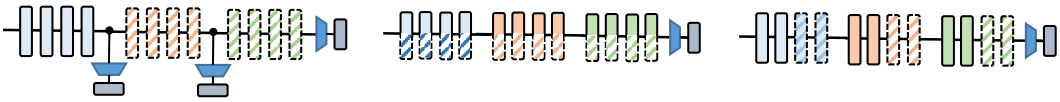


Figure 1: Anytime networks with (**left**) early-exit branches, (**middle**) adaptive widths, or channels, and (**right**) specialized sub-paths (ours). Dashed layers (or blocks) and channels can be skipped for instant accuracy-efficiency trade-offs at runtime.

test time, these specialized sub-paths can be skipped, if needed, for efficiency at marginal loss of accuracy as shown in Figure-1 (right).

The proposed sub-paths specialization is achieved by enforcing sub-networks with different depths to produce semantically equivalent features for every spatial dimension. In Section 3, we formally discuss the rationale behind the sub-paths specialization and introduce a simple and practical training method for sub-paths specialization. In most previous anytime networks, training multiple sub-networks requires training time that is linearly proportional to the number of supported sub-networks, and resolving potential conflicts between sub-networks is an important problem. In contrast, our approach does not try to resolve potential conflicts while jointly training many sub-networks. Instead, our training method exploits only two sub-networks for sub-paths specialization, and, at test time, those specialized sub-paths are exploited selectively to build many sub-networks of various depths. Therefore, the total training time is no greater than training two individual networks.

Finally, in Section 4, we demonstrate that our adaptive depth networks with sub-paths specialization outperform counterpart static networks and achieve actual inference acceleration in multiple tasks including ImageNet classification, COCO object detection and instance segmentation. Further, the smallest sub-networks of our adaptive depth networks achieve significantly better performance than recent state-of-the-art model compression methods.

2 Related Work

Adaptive Networks: Anytime networks (Larsson et al., 2017; Huang et al., 2018; Hu et al., 2019; Wan et al., 2020) and dynamic networks (Wu et al., 2018; Li et al., 2021; Guo et al., 2019; Li et al., 2020; Yang et al., 2020) have attracted lots of attention for their runtime adaptability. Most anytime networks have multiple classifiers that are connected to intermediate layers (Huang et al., 2018; Li et al., 2019; Fei et al., 2022). Training multiple classifiers is a challenging task and many anytime networks (Li et al., 2019; Zhang et al., 2019; Wan et al., 2020; Huang et al., 2018; Hu et al., 2019) exploit knowledge distillation to supervise intermediate classifiers using the last, or the best, classifier. Slimmable neural networks can adjust channel widths for adaptability and they exploit switchable batch normalization to handle multiple sub-networks with a single shared classifier (Yu et al., 2018; Yu & Huang, 2019b). While some dynamic networks (Li et al., 2021) extend anytime networks simply by adding input-conditioned decision gates at branching paths, a few dynamic networks (Wu et al., 2018; Veit & Belongie, 2018; Wang et al., 2018a) extend residual networks by applying block-level decision gates that determine if the block can be skipped. The latter approach is based on the thought that some blocks can be skipped on easy inputs. However, in these dynamic networks with adaptive depths, no formal explanation has been given why some blocks can be skipped for a given input. Therefore, users have no control over the depth of the sub-networks.

Residual Blocks with Shortcuts: Since the introduction of ResNets (He et al., 2016), residual blocks with shortcuts have received extensive attention because of their ability to train very deep networks, and have been chosen by many recent deep neural networks (Sandler et al., 2018; Tan & Le, 2019; Vaswani et al., 2017). Veit et al. (2016) argue that identity shortcuts make exponential paths and results in an ensemble of shallower networks. This thought is supported by the fact that removing individual residual blocks at test time does not significantly affect performance (Huang et al., 2016; Xie et al., 2020). Other works argue that identity shortcuts enable residual blocks to perform iterative feature refinement, where each block improves slightly but keeps the semantic of the representation of the previous layer (Greff et al., 2016; Jastrzebski et al., 2018). Our work build upon those views on residual blocks with shortcuts. We further extend them for adaptive depth networks by introducing a novel training method that exploits the properties of residual blocks more explicitly for sub-paths specialization.

3 SUB-PATHS SPECIALIZATION FOR ADAPTIVE DEPTH NETWORKS

In this section, we first formally discuss the rationale behind the sub-paths specialization. Then, the details of training of sub-paths specialization is discussed.

3.1 MOTIVATION AND OVERVIEW

In typical residual networks such as ResNets, the s-th residual stage is consisted of L identical residual blocks, which transform input features $\mathbf{h}^{s-1}$ additively and produce the output features $\mathbf{h}^{s}$, as shown in Equation 1. A block with a residual function F is supposed to perform two primary functions (1) learning new higher level features and (2) refining already learned features at the same semantic level (Jastrzebski et al., 2018). We call the first and the second functions, respectively, as *feature learning* and *feature refinement*. If a residual block mostly perform feature refinement while not changing the semantic level of features, the performance of the residual network is not significantly affected by dropping the block at test time (Huang et al., 2016; Xie et al., 2020). However, in typical residual networks, most residual blocks are trained to perform both functions, and, hence, random dropping of residual blocks at test time degrades the performance significantly. If some residual blocks can be trained to focus on one function while the remaining blocks are encouraged to focus on the other function, then we can safely skip blocks specialized to feature refinement to save computation at marginal loss of prediction accuracy, if needed for efficiency.

To this end, in our approach, residual functions of a residual stage are partitioned into two sub-paths, $\mathbf{F}_{base}$ and $\mathbf{F}_{skippable}$, as in Equation 1, and, during training, they are encouraged to focus more on respective functions, i.e., learning new level features and refining the learned features.

$$\underbrace{\mathbf{h}^{s-1} + \underbrace{F_1(\mathbf{h}^{s-1}) + F_2(\mathbf{h}^{s}_1) + ... + F_K(\mathbf{h}^{s}_{K-1})}_{\mathbf{F}_{base}}}_{=\mathbf{h}^{s}_{(base)}} + \underbrace{F_{K+1}(\mathbf{h}^{s}_{K}) + ... + F_L(\mathbf{h}^{s}_{L-1})}_{\mathbf{F}_{skippable}} = \mathbf{h}^{s}_{(super)} \quad (1)$$

Since the last half residual functions, or $\mathbf{F}_{skippable}$, is supposed to preserve the semantic level of input features, $\mathbf{F}_{skippable}$ can be skipped for efficiency and the intermediate features, or $\mathbf{h}^{s}_{base}$, can be used as the input to the next stage. When the higher inference accuracy is required, $\mathbf{F}_{skippable}$ can be applied to $\mathbf{h}^{s}_{base}$ to produce more refined features $\mathbf{h}^{s}_{super}$. Since the feature representations $\mathbf{h}^{s}_{base}$ and $\mathbf{h}^{s}_{super}$ are at the same semantic level, either $\mathbf{h}^{s}_{base}$ or $\mathbf{h}^{s}_{super}$ can be provided as an input $\mathbf{h}^{s}$ to the next $(s+1)$-th residual stage with little change in the feature distribution.

Figure 2 shows an residual stage, in which the last half blocks are specialized to preserve the semantic level of input features. Since each such residual stage provides two alternative internal paths, if there are N_s residual stages with specialized sub-paths, 2^{N_s} sub-networks with different accuracy-efficiency trade-offs become available at test time (Section 4.3).

3.2 SUB-PATHS SPECIALIZATION

In order to make the skippable sub-paths to preserve the semantic level of input features, we enforce both feature representations $\mathbf{h}^{s}_{base}$ and $\mathbf{h}^{s}_{super}$ to converge to some common target representation using the same data-generating distribution $\mathbf{X}$ during training:

$$\mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[\mathbf{h}^{s}_{super} - \mathbf{h}^{s}_{base}] = 0 \quad (2)$$

$$\mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[(\mathbf{h}^{s}_{base} + F_{K+1} + ... + F_L) - \mathbf{h}^{s}_{base}] = 0 \quad (3)$$

$$\mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[F_{K+1} + ... + F_L] = 0 \quad (4)$$

$$\mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[F_{K+1}] = \mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[F_{K+2}] = ... = \mathop{\mathbb{E}}_{\mathbf{x}\in\mathbf{X}}[F_L] = 0 \quad (5)$$

Equation 5 does not mean that residual functions in $\mathbf{F}_{skippable}$ do nothing to the input features $\mathbf{h}^{s}_{base}$. Instead, it implies that residual functions in $\mathbf{F}_{skippable}$ are unbiased estimators of target residual functions that preserve input features' semantic level (Greff et al., 2016).

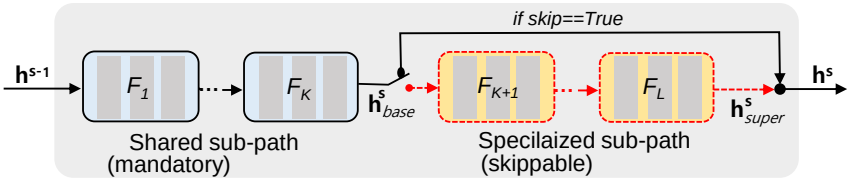


Figure 2: Illustration of a residual stage with a specialized sub-path. A residual stage is split into two sub-paths. While the first (blue) sub-path is mandatory for all sub-networks, the last (yellow) sub-path can be skipped. During training, the last sub-path is enforced to focus on refining the intermediate features $\mathbf{h}^s_{base}$ while not changing its semantic level. At test time, the specialized sub-path contributes to reducing prediction errors, but can be skipped, if needed for efficiency. When the specialized sub-path is skipped, a different set of batch normalization operators, called *skip-aware BN*s, are used in the shared sub-path. The shortcuts within residual blocks are not shown for brevity.

Then, what do the residual functions in $\mathbf{F}_{skippable}$ do? The functionality of them can be further investigated through Taylor expansion. Any loss function $\mathcal{L}$ used for training can be approximated with Taylor expansion as follows:

$$\mathcal{L}(\mathbf{h}^s_{super}) = \mathcal{L}(\mathbf{h}^s_{base} + F_{K+1} + ... + F_{L-1} + F_L) \tag{6}$$

$$\approx \mathcal{L}(\mathbf{h}^s_{base} + F_{K+1} + ... + F_{L-1}) + F_L \cdot \frac{\partial \mathcal{L}(\mathbf{h}^s_{L-1})}{\partial \mathbf{h}^s_{L-1}} + \mathcal{O}(\cdot) \tag{7}$$

$$...$$

$$\approx \mathcal{L}(\mathbf{h}^s_{base}) + \sum_{j=K+1}^{L} F_j \cdot \frac{\partial \mathcal{L}(\mathbf{h}^s_{j-1})}{\partial \mathbf{h}^s_{j-1}} \tag{8}$$

In Equations 6 -8, the loss function is iteratively expanded around $\mathbf{h}^s_{j-1}$ $(j = K+1, ..., L)$, which is the input features to F_j residual function in the skippable sub-path. Note that high order terms $\mathcal{O}(\cdot)$ can be ignored since Fs in $\mathbf{F}_{skippable}$ have small magnitude, as shown in Equation 5. Thus, in Equation 8, the overall training with any loss function $\mathcal{L}$ optimizes F_j $(j = K+1, ..., L)$ in the half space of $-\frac{\partial \mathcal{L}(\mathbf{h}^s_{j-1})}{\partial \mathbf{h}^s_{j-1}}$ to minimize the dot product between F_j and $\frac{\partial \mathcal{L}(\mathbf{h}^s_{j-1})}{\partial \mathbf{h}^s_{j-1}}$. This means that F_j reduces overall loss $\mathcal{L}(\mathbf{h}^s_{super})$ by moving iteratively in the negative direction of $\frac{\partial \mathcal{L}(\mathbf{h}^s_{j-1})}{\partial \mathbf{h}^s_{j-1}}$.

Considering these results in Equations 5 and 8 together, we can conjecture that residual functions in $\mathbf{F}_{skippable}$ (1) preserve the level of input features $\mathbf{h}^s_{base}$, but (2) refine $\mathbf{h}^s_{base}$ iteratively to minimize the loss and contribute to better inference accuracy. In other words, *the skippable sub-path is specialized for refining input features for better inference accuracy while preserving its semantic level*. Therefore, $\mathbf{F}_{skippable}$, or the specialized residual blocks, can be skipped without causing much changes of the feature distribution to the next residual stages (Sections 4.3 and 4.4).

One theory in neuroscience is the existence of regions of the brain focused on particular tasks (Caramazza & Coltheart, 2006). In parallels to neuroscience, we cautiously conjecture that if some parts of a neural network is specialized for one function by suppressing the other functions, the remaining parts of the network are encouraged to focus more on the suppressed functions. Accordingly, we expect that the first half residual functions F_j $(j = 1, ..., K)$, that are not skippable, are conversely encouraged to learn new level features more actively. Our empirical result in Section 4.1 shows that the sub-network, that consists of non-skippable blocks only, outperforms state-of-the-art model compression methods, demonstrating effective feature learning in non-skippable sub-paths.

3.3 Training Adaptive Depth Networks

To enforce $\mathbb{E}_{\mathbf{x}\in\mathbf{X}}[\mathbf{h}^s_{super} - \mathbf{h}^s_{base}] = 0$, given in Equation 2, we propose a training method shown in Algorithm 1, in which two sub-networks, or the *super-net* and the *base-net*, are exploited in a single training loop for sub-paths specialization. The *super-net* and the *base-net* are the largest and the smallest sub-networks, respectively, of the proposed adaptive depth network M. For example, the base-net skips every specialized sub-paths in M. In contrast, the super-net executes all residual blocks of M without skipping.

Algorithm 1 Training with sub-paths specialization. N_s is the number of residual stages.

1: Initialize an adaptive depth network $\boldsymbol{M}$
2: **for** i=1,...,n_{iters} **do**
3: Get next mini-batch of data $\boldsymbol{x}$ and label $\boldsymbol{y}$
4: $optimizer.zero_grad()$
5: $[\hat{\boldsymbol{y}}_{super}, \mathbf{h}_{super}] = \boldsymbol{M}(\boldsymbol{x}$, *skip=[False, ... , False]*$)$ ▷ Forward pass for the super-net of $\boldsymbol{M}$
6: $[\hat{\boldsymbol{y}}_{base}, \mathbf{h}_{base}] = \boldsymbol{M}(\boldsymbol{x}$, *skip=[True, ... , True]*$)$ ▷ Forward pass for the base-net of $\boldsymbol{M}$
7: $subpath_reg = \sum_{s=1}^{Ns} D_{KL}(gap(\mathbf{h}^s_{super}) \| gap(\mathbf{h}^s_{base})) + D_{KL}(\hat{\boldsymbol{y}}_{super} \| \hat{\boldsymbol{y}}_{base})$
8: $loss = (\alpha) criterion(\boldsymbol{y}, \hat{\boldsymbol{y}}_{super}) + (1-\alpha) subpath_reg$
9: $loss.backward()$
10: $optimizer.step()$
11: **end for**

In steps 5 and 6, the forward passes of the super-net and the base-net are executed for the same input $\boldsymbol{x}$. The second argument, *skip*, of the model M indicates in which residual stage its specialized sub-path is skipped. In ResNets with 4 residual stages, $16(= 2^4)$ sub-networks become available by varying the *skip* argument. For example, we choose the base-net of a ResNet by setting the second argument to *skip=[True, True, True, True]*. During the training, a wrapper function is used to obtain intermediate features from every residual stage as well as the outputs of the sub-networks. For instance, during the forward pass of the base-net, intermediate features from N_s residual stages, or $\mathbf{h}_{base} = [\mathbf{h}^1_{base}, ..., \mathbf{h}^{N_s}_{base}]$, and output $\hat{\boldsymbol{y}}_{base}$ are obtained.

In steps 7 and 8, we modify the training criterion by adding Kullback Leibler (KL) divergence between $\mathbf{h}_{base}$ and $\mathbf{h}_{super}$ as an regularization term. This term enforces $\mathbb{E}_{\mathbf{x} \in \mathbf{X}}[\mathbf{h}_{super} - \mathbf{h}_{base}] = 0$ for sub-paths specialization. Global average pooling, denoted by $gap(\cdot)$, is applied to $\mathbf{h}_{base}$ and $\mathbf{h}_{super}$ before computing their KL divergence since we want $\mathbf{h}_{super}$ and $\mathbf{h}_{base}$ to have semantically equivalent feature distributions, not exactly same values. In the regularization term, the KL divergence between logits, $\hat{\boldsymbol{y}}_{base}$ and $\hat{\boldsymbol{y}}_{super}$, is also included since they are last features from the sub-networks. Hyperparameter α determines the strength of the regularization term.

Unlike previous anytime networks, our approach does not train all sub-networks jointly or sequentially. Therefore, we do not need to resolve potential conflicts among sub-networks during training. Instead, in Algorithm 1, only two sub-networks are exploited during training for sub-paths specialization, and, hence, the total training time is no greater than training two sub-networks individually. However, at test time, the specialized sub-paths can be exploited selectively to construct 2^{N_s} sub-networks of various depths. We demonstrate this result in Section 4.3.

3.4 Skip-Aware Batch Normalization for Shared Sub-paths

In our adaptive depth networks, each residual stage expects the distribution of input features is not affected by the choices of internal paths of its previous residual stages. However, the shared sub-paths inside each residual stage still need adaptability to handle different sub-networks. Originally, batch normalization (BN) (Ioffe & Szegedy, 2015) was proposed to handle internal covariate shift during training non-adaptive networks by normalizing features. In our adaptive depth networks, however, internal covariate shifts occur in shared sub-paths when different sub-networks are used. To handle this internal covariate shifts, switchable BN operators, called *skip-aware BNs*, are used in shared sub-paths. For example, at each residual stage, different BN operators are used by the layers in the shared sub-path when specialized sub-path is skipped. The effectiveness of switchable BNs has been demonstrated in neural networks with adaptive widths (Yu & Huang, 2019a) and adaptive resolutions (Zhu et al., 2021), and we apply them for our adaptive depth networks. The amount of parameters for skip-aware BNs is negligible. In ResNet50, skip-aware BNs increase the parameters by 0.07%.

4 Experiments

To demonstrate the effectiveness of our approach, we conduct experiments on ImageNet classification and COCO object detection and instance segmentation benchmarks. We use two modern residual networks, ResNets (He et al., 2016) and MobileNetV2 (Sandler et al., 2018) as base models. The

models with the suffix *'-ADN'* denotes the proposed adaptive depth networks extending those base networks. For adaptive depth networks, we add skip-aware BNs for shared sub-paths and every residual stage is split equally into two sub-paths. During training, every skippable sub-path is specialized according to Algorithm 1 from scratch. One exception is the 3rd stage of ResNet101 that has exceptional long 23 blocks. In the 3rd stage of ResNet101-ADN, the first 5 blocks are used for shared sub-path and the last 18 blocks are specialized and become skippable [1].

4.1 IMAGENET CLASSIFICATION

We evaluate our method on ILSVRC2012 dataset (Russakovsky et al., 2015) that has 1000 classes. The dataset consists of 1.28M training and 50K validation images. We follow most training settings in the original papers (He et al., 2016; Sandler et al., 2018), except that ResNet models are trained for 150 epochs. The hyperparameter α is set to 0.5 and 0.7 for ResNets and MobileNetV2, respectively.

Table 1: Results on ImageNet. Latency is measured on an Nvidia RTX 3090 with batch size of 64.

Baseline	Model	Params	FLOPs	Acc@1	Acc@5	Latency
ResNet50	ResNet50 (individual network)	25.56M	4.11G	76.7%	93.2%	43.2ms
	ResNet29 (individual network)	17.11M	2.58G	75.0%	92.3%	27.6ms
	ResNet50-ADN (super-net)	25.58M	4.11G	77.6%	93.7%	43.2ms
	ResNet50-ADN (base-net)		2.58G	76.1%	93.2%	27.6ms
	Early-exiting branches	28.82M	4.11G	75.2%	-	43.2ms
	(Zhang et al., 2019; 2022)		3.52G	72.8%	-	39.0ms
			2.05G	58.5%	-	27.4ms
	(Li et al., 2019)		1.02G	43.9%	-	16.6ms
	Slimmable widhts	25.6M	4.1G	76.0%	-	43.2ms
	1.0x, 0.75x, 0.5x, 0.25x widths		2.0G	75.6%	-	33.8ms
			1.0G	74.0%	-	21.2ms
	(Yu et al., 2018; Yu & Huang, 2019a)		570M	72.2%	-	11.0ms
ResNet101	ResNet101 (individual network)	44.55M	7.83G	77.4%	93.5%	73.2ms
	ResNet35 (individual network)	29.41M	3.01G	75.7%	92.5%	31.2ms
	ResNet101-ADN (super-net)	44.58M	7.83G	78.6%	94.2%	72.5ms
	ResNet101-ADN (base-net)		3.01G	77.0%	93.6%	31.2ms
MobileNetV2	MV2 (individual network)	3.50M	0.32G	72.1%	90.3%	15.5ms
	MV2-Base (individual network)	2.99M	0.25G	70.7%	89.6%	13.2ms
	MV2-ADN (super-net)	3.72M	0.32G	72.7%	90.8%	15.5ms
	MV2-ADN (base-net)		0.25G	70.7%	89.8%	13.2ms

The results in Table 1 show that our adaptive depth networks, both the largest super-net and the smallest base-net, outperform individual counterpart networks consistently. In Table 1, ResNet50-ADN is also compared with two representative anytime networks (Zhang et al., 2019; Yu & Huang, 2019a) that adapt the depths and the widths, respectively. The result shows that our adaptive depth networks outperform these competitors significantly in every aspect. In particular, unlike competitors, the reduction of FLOPs of ResNet-ADN is closely translated into actual acceleration. For instance, when the FLOPs of ResNet50-ADN (base-net) is reduced by 37.3%, its inference latency is also similarly reduced by 36.2%. In contrast, although S-ResNet50 (0.75x) requires 22% less FLOPs than ResNet50-ADN (base-net), it shows 22.4% longer inference latency in practice. Since S-ResNet50's network depth is not changed by reducing widths, it still needs to perform similar number of GPU kernel invocations in practice. In early-exiting anytime networks, effective depths of network can be reduced. But as shown in Zhang et al. (2019)'s work, sub-networks of early-exiting networks manifest significantly low inference accuracy since they only exploit low-level features. In contrast, our sub-networks exploit features of all levels while significantly reducing effective network depths.

In Table 2, ResNet50-ADN (base-net), which is the smallest and the most efficient sub-network of ResNet50-ADN, is compared with several state-of-the-art model compression methods. Our

[1]Our evalaution shows that increasing the shared sub-path longer than 5 blocks does not improve the performance of ResNet101-AND. The effect of varying length of sub-paths is evaluated in Appendix A.1.

Table 2: Comparison of state-of-the-art methods for compressing ResNet50 on ImageNet.

Model	Params	FLOPs	↓FLOPs	Acc@1	Acc@5
ResNet50-ADN (base-net)	17.1M	2.6G	37%	76.1%	93.2%
PFP-A-ResNet50 (Liebenwein et al., 2020)	20.9M	3.7G	10%	75.9%	92.8%
Versatile-ResNet50 (Wang et al., 2018b)	11.0M	3.0G	27%	74.5%	91.8%
GReg-1 (Wang et al., 2021)	25.6M	2.8G	33%	76.1%	-
C-SGD70-ResNet-50 (Ding et al., 2019)	-	2.6G	37%	75.3%	92.5%
Meta-Pruning 0.75 (Liu et al., 2019)	-	2.3G	44%	75.4%	-
DR-ResNet50 (α=2.0) (Zhu et al., 2021)	30.5M	2.3G	44%	75.3%	92.2%
Ghost-ResNet50-2 (Han et al., 2020)	13.0M	2.2G	46%	75.0%	92.3%

ResNet50-ADN (base-net) achieves very competitive results compared to many state-of-the-art methods. Some of these state-of-the-art methods require unusual training techniques such as pretraining or retraining of pretrained networks to provide static accuracy-efficiency trade-offs. In contrast, our ResNet50-ADN (base-net) is one of many sub-network of ResNet50-ADN that is trained from scratch according to Algorithm 1, and, hence, it can increase its depth instantly at runtime if the higher inference accuracy is required. This result demonstrates that non-skippable sub-paths of our adaptive depth networks learn compact feature representations effectively while skippable sub-paths are enforced to preserve the level of input features during training.

4.2 Ablation Study

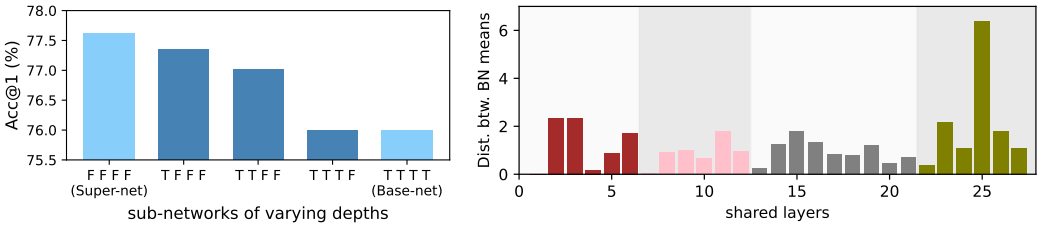
Table 3: Ablation analysis with ResNet50-ADN. Applied components are checked with $\surd$ mark.

Sub-paths specialization	Skip-aware BNs	Acc@1	
		Super-net	Base-net
		75.2% (↓ 1.5%)	72.2% (↓ 2.8%)
$\surd$		76.1% (↓ 0.6%)	74.9 % (↓ 0.1%)
	$\surd$	76.6% (↓ 0.1%)	75.1% (↑ 0.1%)
$\surd$	$\surd$	77.6% (↑ 0.9%)	76.1% (↑ 1.1%)

We conduct ablation study on ImageNet classification to investigate the influence of two key components of the proposed adaptive depth networks: (1) sub-paths specialization and (2) skip-aware BNs. The results are shown in Table 3. When neither of them is applied, the inference accuracy of the super-net and the base-net is significantly lower than that of individual networks by 1.5% and 2.8%, respectively. This result shows the difficulty of training multiple sub-networks in anytime networks. During training, uncoordinated training signals from the super-net and the base-net decrease the effect of training, resulting in worse performance than individual networks. When one of the two components is applied individually, the performance is still slightly worse than individual networks'. Finally, when both sub-paths specialization and skip-aware BNs are applied together, ResNet50-ADN achieves significantly better performance than individual networks, both in the super-net and the base-net. This result demonstrates that both sub-paths specialization and skip-aware BNs are essential for the proposed adaptive depth networks.

4.3 Varying Network Depths at Test Time

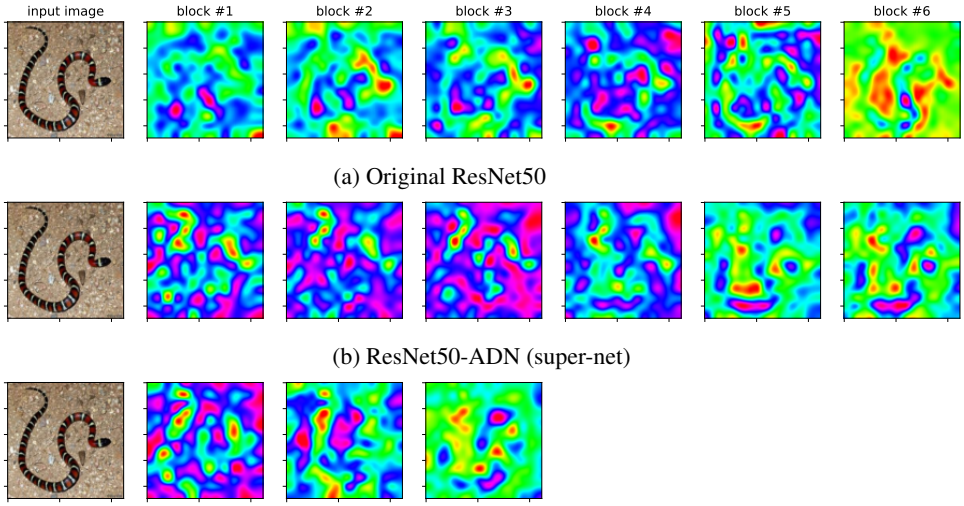
One important advantage of sub-paths specialization is that each residual stage's choice of internal paths does not affect the following residual stages. Therefore, each residual stage can independently make a decision about skipping its specialized sub-path. This implies that our adaptive depth networks can support 2^{N_s} sub-networks at test time, when N_s residual stages are given. Figure 3-(a) shows the performance of ResNet50-ADN when its depth is varied at test time. Among them, only the super-net and the base-net are used for training in Algorithm 1. The other sub-networks are built by applying different combinations of skipping decisions at test time. Although these sub-networks are not trained explicitly, their performance does not drop significantly. Instead, they show gradual degradation of performance as the depth of sub-networks becomes gradually shallower.



(a) Acc@1 of various sub-networks

(b) Distances btw. skip-aware BNs in shared sub-paths

Figure 3: **(a)** Performance while varying the depth of ResNet50-ADN. The labels under each bar are boolean values indicating the residual stages that skip their specialized sub-path. 'T' means that the specialized sub-path is skipped. **(b)** The distances between the means of skip-aware BNs in the shared sub-paths of ResNet50-ADN. Bars with different colors belong to different residual stages.



(a) Original ResNet50

(b) ResNet50-ADN (super-net)

(c) ResNet50-ADN (base-net)

Figure 4: Class Activation Map images of the 3rd stages of ResNet50s. **(a)** Original ResNet50's activation regions change gradually across all blocks, indicating that every block is involved in learning new level features and refining the learned features.**(b)** In ResNet50-ADN (super), two sub-paths have very different progressions. In the non-skippable first 3 blocks, the activation regions change significantly, implying active learning of new level features. In the skippable last 3 blocks, the activation regions are gradually refined around the target. **(c)** The base-net's final activation map is very similar to the super-net's, but has less hot areas around the target object.

4.4 Analysis of Sub-paths Specialization

The premise of sub-paths specialization is that each residual stage produce semantically equivalent features regardless of whether its specialized sub-path is skipped or not; i.e, $\mathbb{E}_{\mathbf{x}\in\mathbf{X}}[\mathbf{h}^s_{super} - \mathbf{h}^s_{base}] = 0$ $(s = 0..N_s)$. We can verify this by inspecting the input distribution of the next residual stages. Figure 3-(a) shows the distances between the means of switchable BNs in the shared sub-paths of ResNet50-ADN. In Figure 3-(a), the distance at every residual stage's first layer is close to zero. This demonstrates that their previous residual stages produce feature distributions of the same means regardless of whether the specialized sub-path is skipped or not. Figure 3-(a) also shows that the shared layers except the first layers of each residual stage have very different input distributions, demonstrating the necessity of skip-aware BNs in the shared sub-paths.

To further verify that the non-skippable sub-paths and the skippable sub-paths, respectively, focus more on learning new level features and refining learned features, we visualize the activation of 3rd residual stage of ResNet50-ADN using Grad-CAM (Selvaraju et al., 2017) in Figure 4. The 3rd

residual stage of ResNet50-ADN has six residual blocks and the last three blocks are skippable. In Figure 4-(a), the activation regions of original ResNet50 changes gradually across all consecutive blocks. This implies that all blocks are involved in gradual learning of new level features as well as the refinement of the learned features. In contrast, in Figure 4-(b), ResNet50-ADN (super-net) manifests very different progressions of activation regions in two sub-paths. In the first three residual blocks, we can observe significant changes of hot regions in wide areas, suggesting active learning of new level features. However, in the skippable last three blocks, the activation regions have little movements and gradually become more concentrated around the target object, demonstrating the refinement of learned features. Further, in Figure 4-(c), we can observe that the final activation map of the ResNet50 (base-net) is very similar to the super-net's final activation map in (b), implying that they are at the same feature level as suggested in Equation 2. However, the final activation map of the base-net has less hot activation regions around the target object than super-net's, potentially resulting in lower inference accuracy than super-net's more refined features. Finally, it should be noted that even though the three blocks of the base-net share weights with the first three blocks of the super-net, they manifest very different activation maps. This demonstrates that the skip-aware BNs are effective in processing multi-modal inputs.

4.5 Object Detection and Instance Segmentation

Table 4: Object detection and instance segmentation results on MS COCO dataset.

Detector	Backbone (super-net/base-net)	FLOPs	Individual Networks		**ResNet-ADN (ours)**	
			Box AP	Mask AP	**Box AP**	**Mask AP**
Faster-RCNN (Ren et al., 2017)	ResNet50	207.07G	36.4		37.8	
	ResNet29	175.66G	32.4		34.0	
	ResNet101	283.14G	38.7		40.0	
	ResNet35	184.61G	34.8		35.4	
Mask-RCNN (He et al., 2017)	ResNet50	260.14G	37.2	34.1	38.3	34.1
	ResNet29	228.73G	32.7	29.9	34.1	31.2
	ResNet101	336.21G	39.4	35.9	40.7	35.9
	ResNet35	237.68G	33.9	30.9	35.9	32.2

In order to investigate the generalization ability of our approach, we use MS COCO 2017 datasets on object detection and instance segmentation tasks using representative detectors. We compare individual ResNets and our adaptive depth ResNets-ADN as backbone networks of the detectors. For training of detectors, we use Algorithm 1 with slight adaptation. For object detection, the intermediate features $\mathbf{h}^s_{base}$ and $\mathbf{h}^s_{super} (s = 1..N_s)$ can be obtained directly from backbone network's feature pyramid networks (FPN) (Lin et al., 2017), and, hence, a wrapper function is not required to extract intermediate features. All networks are trained on `train2017` for 12 epochs from ImageNet pretrained weights, following the training settings suggested in (Lin et al., 2017; Han et al., 2020).

Table 4 shows the results on `val2017` containing 5000 images. Our adaptive depth backbone networks significantly outperform individual static backbone networks in terms of COCO's standard metric AP. This result demonstrates that our adaptive depth networks with the sub-paths specialization learn better feature representations in every feature level.

5 Conclusions

We presented a novel approach to adaptive depth networks that provide instant accuracy-efficiency trade-offs at runtime. Unlike previous approaches, our adaptive depth networks do not require auxiliary add-on networks or classifiers to control network depths. Instead, our approach train sub-paths of a single network to have a special property, with which the sub-paths can be skipped for efficiency at marginal loss of prediction accuracy. We discussed the theoretic rationale behind this sub-paths specialization, and presented a simple and practical training method. While our training method uses only two sub-networks, it makes various sub-networks with different depths available at test time. Through extensive experiments, we demonstrated that the proposed adaptive depth networks outperform individual static networks and state-of-the-art model compression methods while achieving actual inference speed-up.

6 REPRODUCIBILITY

For reproducibility, we provide source codes and the links to the pretrained weights as the supplementary material. Code and models will be available in authors' public repository. All programs for training and evaluation run on PCs equipped with two or four Nvidia RTX 3090 GPUs and an Intel i9-10900X CPU @3.7GHz.

REFERENCES

Alfonso Caramazza and Max Coltheart. Cognitive neuropsychology twenty years on. *Cognitive Neuropsychology*, 23(1):3–12, 2006.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Zhengcong Fei, Xu Yan, Shuhui Wang, and Qi Tian. Deecap: Dynamic early exiting for efficient image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12216–12226, June 2022.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Q. Guo, Z. Yu, Y. Wu, D. Liang, H. Qin, and J. Yan. Dynamic recursive neural network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5142–5151, 2019.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations (ICLR)*, 2016.

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*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

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

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell. Learning anytime predictions in neural networks via adaptive loss balancing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3812–3821, 2019.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *International Conference on Learning Representations*, 2018.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals, 2017.

Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8607–8617, 2021.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1891–1900, 2019.

Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BJxkOlSYDH`.

Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3296–3305, 2019.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jun 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–18, 2018.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in neural information processing systems*, 29:550–558, 2016.

Chengcheng Wan, Henry Hoffmann, Shan Lu, and Michael Maire. Orthogonalized sgd and nested architectures for anytime neural networks. In *International Conference on Machine Learning*, pp. 9807–9817. PMLR, 2020.

Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=o966_Is_nPA`.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 409–424, 2018a.

Yunhe Wang, Chang Xu, XU Chunjing, Chao Xu, and Dacheng Tao. Learning versatile filters for efficient convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1608–1618, 2018b.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826, 2018.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers, 2019a.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1803–1811, 2019b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3713–3722, 2019.

Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2022. doi: 10.1109/TPAMI.2021.3067100.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

Mingjian Zhu, Kai Han, Enhua Wu, Qiulin Zhang, Ying Nie, Zhenzhong Lan, and Yunhe Wang. Dynamic resolution network. In *Advances in Neural Information Processing Systems*, volume 35, pp. 10985–10998, 2021.

A APPENDIX

A.1 VARYING THE LENGTHS OF SUB-PATHS

In our adaptive depth networks, most residual stages have 4-6 residual blocks, and each stage is split into two sub-paths; the first sub-path is shared by all sub-networks while the second sub-path can be skipped, if needed for efficiency. One exception is the 3rd stage of ResNet101 that has exceptional long 23 blocks. We investigate how the lengths of the sub-paths affect the performance by varying the number of shared/skippable blocks in the 3rd stage of ResNet101-ADN during training.

The result in Table 5 shows that if the number of shared blocks is too small (e.g., 3 shared blocks), it results in lower performance, both for the super-net and the base-net. This result shows that the performance of the super-net is limited by the base-net's low performance since our adaptive depth networks enforce both the super-net and the base-net to learn same feature levels at every residual stage. As the more shared blocks are used by the base-net, the performance of both the super-net and the base-net improve. As the performance of the base-net is improved gradually from 75.4% to 77.9%, the super-net's performance improves from 77.8% to 78.6%. However, when the number of shared blocks become greater than 5, the performance of the super-net is bounded at about 78.6%. This result implies that some number of residual blocks is essential for proper learning of new level features, but their effectiveness diminishes after some optimal points.

Table 5: ImageNet validation accuracy (%) while varying the number of shared and skippable blocks in the 3rd residual stage of ResNet101-ADN.

# of blocks in a stage	Acc@1		FLOPs		Parameters
(Shared : Specialized)	super-net	base-net	super-net	base-net	
3 : 20	77.8	75.4	7.83G	2.58G	44.55M
5 : 18	78.6	77.0	7.83G	3.01G	44.55M
9 : 14	78.5	77.5	7.83G	3.89G	44.57M
13 : 10	78.6	77.9	7.83G	4.55G	44.59M