

Step by Step Network

Dongchen Han¹ Tianzhu Ye¹ Zhuofan Xia¹ Kaiyi Chen¹ Yulin Wang¹ Hanting Chen² Gao Huang^{1*}
¹ Tsinghua University ² Huawei Noah’s Ark Lab

Abstract

Scaling up network depth is a fundamental pursuit in neural architecture design, as theory suggests that deeper models offer exponentially greater capability. Benefiting from the residual connections, modern neural networks can scale up to more than one hundred layers and enjoy wide success. However, as networks continue to deepen, current architectures often struggle to realize their theoretical capacity improvements, calling for more advanced designs to further unleash the potential of deeper networks. In this paper, we identify two key barriers that obstruct residual models from scaling deeper: shortcut degradation and limited width. Shortcut degradation hinders deep-layer learning, while the inherent depth-width trade-off imposes limited width. To mitigate these issues, we propose a generalized residual architecture dubbed **Step by Step Network (StepsNet)** to bridge the gap between theoretical potential and practical performance of deep models. Specifically, we separate features along the channel dimension and let the model learn progressively via stacking blocks with increasing width. The resulting method mitigates the two identified problems and serves as a versatile macro design applicable to various models. Extensive experiments show that our method consistently outperforms residual models across diverse tasks, including image classification, object detection, semantic segmentation, and language modeling. These results position StepsNet as a superior generalization of the widely adopted residual architecture.

1. Introduction

In recent years, great success in computer vision has been witnessed through scaling up network depth. Early theoretical works [8, 11, 32, 34] suggest that deeper neural networks can capture exponentially more informative representations than shallower ones. This theoretical insight propelled one of the most magnificent works, ResNet [19], whose residual architecture allows networks with more than one hundred layers to be optimized effectively. Benefited from increased depth, residual models can effectively

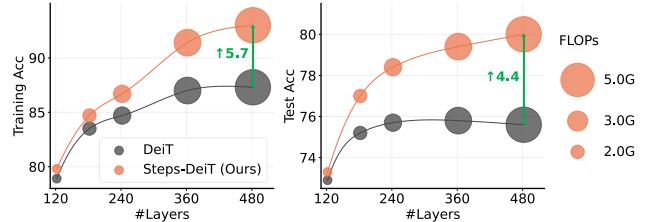


Figure 1. Training accuracy (left) and test accuracy (right) on ImageNet-1K as we gradually increase model depth while keeping the width fixed. The residual architecture DeiT does not deliver satisfactory results as depth increases. In contrast, our method enables the model to leverage increased depth more effectively, achieving much higher results on both training and test sets. Notably, the DeiT models used in this pilot study are *not* standard DeiT-T/S/B. Please refer to Sec. 5.3 for experiment details.

learn from large scale of data and enjoy substantial improvements, driving advances in the field of computer vision [10, 26, 31, 39] and various other scenarios [33, 42].

However, have we fully revealed the potential of network depth? Maybe not. Despite the theoretical power of very deep networks, modern CNNs [26, 35] and Vision Transformers [10, 25, 39] generally saturate their optimal performance at some relatively modest depths around one hundred layers. As one simply increase depth further, these models remain trainable but often struggle to deliver enough improvements. Notably, the core reason behind this phenomenon is *not* simple overfitting [40, 54]. A typical example is shown in Fig. 1, where the deeper DeiT models do not demonstrate the theoretical gain in training accuracy, and thus the test results. This disparity from theory indicates that the widely successful residual architecture may still exist unresolved challenges at increased depth, which possibly hinders the real expressive capability of deeper networks. This motivates us to rethink the design of residual connections and come up with a fundamental research question:

How should we further scale up the network depth effectively and unleash the great potential of deeper networks?

In this paper, we start by reconsidering the residual architecture in deep models, offering both theoretical and empirical analyses to shed some light on this crucial question. Specifically, we analyze the key challenges of deep residual networks from two perspectives: shortcut degradation and

*Corresponding Author.

limited width. Firstly, we reveal that the shortcuts in deep residual models can “degrade” during early training, which prevents deep layers from accessing input information and propagating back their gradients, resulting in optimization difficulties. Secondly, the inherent width-depth trade-off limits deep residual models to insufficient width. Under a practical computational budget, deeper residual models have to be restricted to thinner widths than their shallower counterparts, impairing their expressive capacity [2, 17, 30].

Based on our analysis, we propose a generalized residual architecture, **Step by Step Network (StepsNet)**, as a promising remedy for these limitations. Instead of processing the input all at once, we split it into multiple parts along the channel dimension and progressively take each part of the information into consideration. This leads to a macro architecture composed of multiple residual networks of increasing widths, following a narrow-to-wide stacking strategy (see Fig. 4). We demonstrate that our design improves the conventional residual architecture with easier optimization at large depth and mitigates its width-depth trade-off, thereby effectively overcoming the two identified limitations and further revealing the potential of deep networks. Empirical studies across diverse tasks are conducted to validate the effectiveness of StepsNet, including image classification, object detection, semantic segmentation and language modeling. The results, with an example shown in Fig. 1, confirm that StepsNet benefits from increased depth and suggest it as an simple, effective and scalable generalization of the widely adopted residual architecture.

Our main contributions and takeaways are three-folded:

- Through both theoretical and empirical analyses, we identify two key challenges of deep residual models: shortcut degradation and limited width.
- We present a simple, clean yet effective residual architecture named Step by Step Network (StepsNet), which mitigates these two challenges and supports the development of deeper models with enhanced expressive power. StepsNet is independent of the micro designs and applicable to various residual models.
- Extensive empirical evaluations across diverse tasks confirm that StepsNet enables models to benefit from increased depth, achieving improved results on conventional residual models without additional modifications.

2. Related Works

Theoretical power of network depth. Understanding the impact of depth is a fundamental research question in deep learning, with many studies offering theoretical insights. Early studies [8, 11] reveal that certain functions can be represented far more efficiently by deep networks than shallow ones. [32, 34] further demonstrate that deep networks can represent exponentially more complex functions with increasing depth but not width. Other researches [29, 36]

show that the number of linear regions grows exponentially with network depth, but only polynomially with width. [38] proves that there exist deeper networks that can not be approximated by shallower ones without an exponential increase in size. Given the dramatic power of network depth, going deeper has long been the pursuit of neural networks, with extremely deep models being considered promising to address highly complex tasks.

Network architectures are known to significantly impact the capacity and optimization of neural networks. ResNet [19] introduces residual connections, which greatly benefit network optimization and enable the development of deeper, more expressive models. Fractalnet [22] proposes the fractal architecture with paths of varying depths, matching ResNet’s performance without shortcuts. DenseNet [21] enhances information flow by using dense connections, achieving high parameter efficiency. Building on residual architectures, the Transformer model [42] is proposed in natural language processing and quickly adopted in computer vision [10, 13, 16, 45], achieving success in image classification [14, 15, 25, 44], semantic segmentation [5, 47], object detection [4, 23], and multi-modal tasks [33].

Deep models. Motivated by the potential of very deep networks, many works are conducted to develop expressive deep models. For instance, ReZero [1] and LayerScale [40] introduce learnable scale parameters in residual branches to improve convergence in deep residual models. FixUp [51] and DeepNet [43] propose initialization strategies to stabilize deep model training. DeepViT [54] addresses attention collapse in deep vision Transformers through re-attention. Normformer [37] improves transformer pretraining with extra normalization. Despite their elegant outcomes, these approaches still rely on conventional residual architectures, potentially limiting the capabilities of very deep networks. In this paper, we start with analyzing the challenges of deep residual models, and introduce a novel macro architecture named Step by Step Network to further unleash the power of depth. Our method serves as a strong alternative to residual architectures, compatible with various models, micro designs, and initialization techniques.

3. Challenges of Very Deep Residual Models

Theoretical studies indicate that the expressive power of neural networks grows exponentially with depth [32, 34]. Therefore, extremely deep networks are believed to have great potential for capturing intricate patterns and solving complex tasks. While the widely adopted residual architecture [19] allows models to benefit from over one hundred layers, further increasing depth often fails to yield expected returns. In this section, we dive into the challenges of very deep residual models, offering detailed analyses from two perspectives: shortcut degradation and limited width.

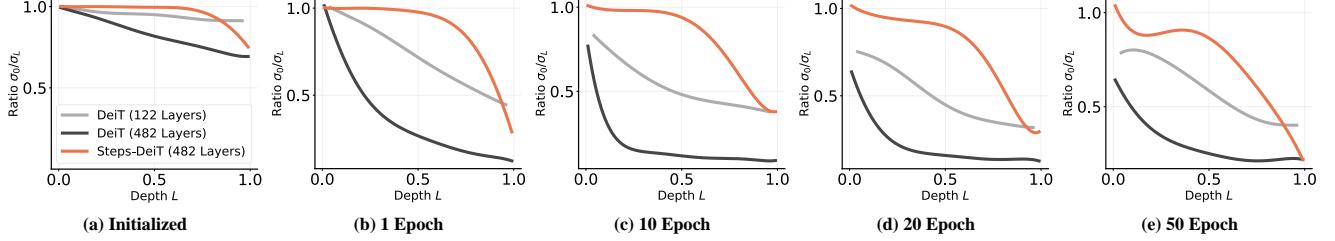


Figure 2. **The shortcut ratio** $\gamma_l = \frac{\sigma_0}{\sigma_l}$ in DeiT and Steps-DeiT, where σ_0 and σ_l are the standard deviations of the input z_0 and feature after l blocks z_l , respectively. The depth is normalized to $[0, 1]$, where 0 and 1 denote input and output. In a very deep residual model (more than 400 layers), the shortcut ratio $\frac{\sigma_0}{\sigma_l}$ approaches zero at early training stages, which prevents later residual blocks from obtaining input information and propagating its gradient back to the input, thus leading to optimization difficulties.

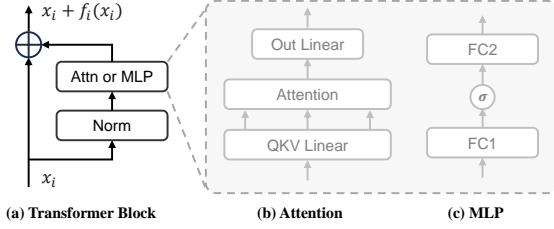


Figure 3. An illustration of a Transformer block to help understand the analyses in Sec. 3.

3.1. Shortcut Degradation

Residual architectures have been prominent in deep learning since the advent of ResNet [19]. They can be formulated as a sequence of blocks of the form:

$$z_l = z_{l-1} + \mathcal{R}_l(z_{l-1}) = z_0 + \sum_{i=1}^l \mathcal{R}_i(z_{i-1}) \triangleq z_0 + r_l, \quad (1)$$

where z_l is the representation after l blocks, \mathcal{R}_l denotes the l -th residual block, and $r_l = \sum_{i=1}^l \mathcal{R}_i(z_{i-1})$. Let’s consider how z_l is processed by the next block \mathcal{R}_{l+1} . Taking the Transformer architecture in Fig. 3 as an example, a normalization layer at the input of \mathcal{R}_{l+1} transforms z_l into:

$$\begin{aligned} \hat{z}_l &= (z_0 + r_l - \mu_l)/\sigma_l \\ &= \frac{\sigma_0}{\sigma_l} \hat{z}_0 + \frac{\sigma_r}{\sigma_l} \hat{r}_l + \frac{\mu_0 + \mu_r - \mu_l}{\sigma_l} \\ &= \frac{\sigma_0}{\sigma_l} \hat{z}_0 + \frac{\sigma_r}{\sigma_l} \hat{r}_l, \end{aligned} \quad (2)$$

where $\mu_0, \mu_l, \mu_r, \sigma_0, \sigma_l, \sigma_r$ and $\hat{z}_0, \hat{z}_l, \hat{r}_l$ are the means, standard deviations and normalized results of z_0, z_l, r_l , respectively. Note that $\mu_l = \mu_0 + \mu_r$, as $z_l = z_0 + r_l$. It is shown that the normalized \hat{z}_l is a weighted combination of \hat{z}_0 and \hat{r}_l , weighted by $\frac{\sigma_0}{\sigma_l}$ and $\frac{\sigma_r}{\sigma_l}$. In early training stages, the $r_l = \sum_{i=1}^l \mathcal{R}_i(z_{i-1})$ contains minimal useful patterns due to randomly initialized \mathcal{R}_i parameters. Consequently, \hat{z}_l can be interpreted as a mixture of the input \hat{z}_0 and noisy residuals \hat{r}_l . We define the shortcut ratio $\gamma_l = \frac{\sigma_0}{\sigma_l}$ to quantify the preservation of the input shortcut up to depth l .

Previous studies demonstrate that feature variance σ_l grows with network depth in residual architectures [7, 18, 51], thus causing γ_l to decrease as l increases. In Fig. 2, we report γ_l at various training epochs for the DeiT models presented in Fig. 1. Consistent with our analysis, the 122-layer DeiT exhibits a decreasing γ_l with depth, with an average value of around 0.5. In contrast, the 482-layer DeiT, which contains more residual terms, exhibits a rapidly diminishing shortcut ratio γ_l that quickly approaches zero. Similar to the analyses in [1, 7, 43], we find such near-zero shortcut ratio γ_l can lead to two critical issues in early training phases:

- $\hat{z}_l = \frac{\sigma_0}{\sigma_l} \hat{z}_0 + \frac{\sigma_r}{\sigma_l} \hat{r}_l \approx \frac{\sigma_r}{\sigma_l} \hat{r}_l$ becomes dominated by noisy residuals rather than the shortcut, preventing the block \mathcal{R}_{l+1} from obtaining the input information \hat{z}_0 .
- The gradient $\partial \hat{z}_l / \partial \hat{z}_0 = \sigma_0 / \sigma_l \approx 0$ vanishes, hindering effective back propagation from \mathcal{R}_{l+1} to the input.

These analyses suggest that while current residual architecture enables models with around one hundred layers to be trained effectively, the shortcuts in even deeper residual models can “degrade” during training, thus leading to optimization difficulties. Notably, this challenge is not confined to the Transformer architecture, and similar analyses work for other residual models. The key observation is that very deep residual models add too much “noise” to the shortcut, and the proportion of input information diminishes with depth, resulting in both forward and backward issues. Moreover, z_0 in our analysis may represent not only the raw input data but also features extracted by several early layers or blocks, and our analysis elucidates how z_0 changes through subsequent residual blocks.

3.2. Limited Width

Beyond optimization challenges posed by shortcut degradation, we further identify a fundamental architectural limitation arising from the depth-width trade-off under fixed computational budgets. Specifically, for a residual model M with D blocks of width C , we always have $\Omega(M) = \mathcal{O}(C^2 D)$, where $\Omega(M)$ denotes the total computation complexity of M . For example, assume each block of model M is a Transformer block (as shown in Fig. 3, comprising an

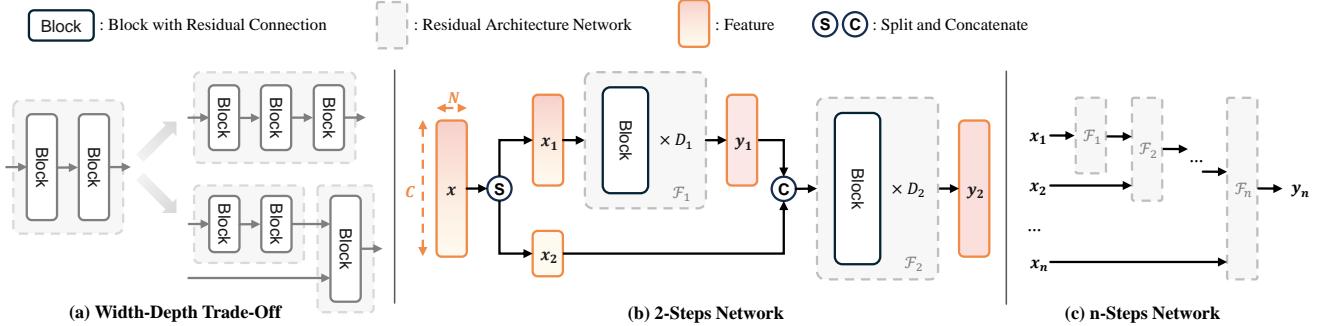


Figure 4. An illustration of the proposed Step by Step Network. For simplicity, the shortcut in each residual block is omitted. (a) **Width-depth trade-off.** When enlarging the depth of a residual model, the width has to be reduced to maintain similar computation. In contrast, StepsNet makes it possible for the model to be deeper with fixed width and computation. (b) **The 2-step network.** Given an input $x \in \mathbb{R}^{N \times C}$, the model first splits the information into two parts x_1, x_2 along the channel dimension C . Subsequently, x_1 and x_2 are processed sequentially by two networks \mathcal{F}_1 and \mathcal{F}_2 , generating the final output y_2 . (c) **The n -step network.** Repeatedly substituting the first network with a 2-step architecture creates an n -step network, which divides x into n parts and processes them progressively.

attention sub-block and an MLP sub-block), then we have:

$$\begin{aligned}\Omega(M) &= (\underbrace{3NC^2 + NC^2}_{\text{QKV, Out Linear}} + \underbrace{2N^2C}_{\text{Attention}} + \underbrace{8NC^2}_{\text{MLP}}) \times D \\ &= 12NC^2D + 2N^2CD = \mathcal{O}(C^2D),\end{aligned}\quad (3)$$

where N is the number of input tokens.

In deep learning, comparing models with similar computational budget T is common practice, as bigger models (models with higher computation) always have better capacity. Given $\Omega(M) = \mathcal{O}(C^2D)$, with fixed budget T and block design, the width C will inevitably decrease proportionally as depth D increases. For instance, Fig. 4(a) illustrates a simple two-block residual network. To design a three-block model with the same computation, we have to narrow the model width by a factor of about $\sqrt{3/2}$. Therefore, very deep residual models have a more limited width than their shallower counterparts with *comparable computational budgets*. As theoretical works [2, 17, 30] prove that networks with insufficient width can not achieve universal function approximation even with infinite depth, this limited dimensionality can be a key constraint in very deep models, hindering their capability when compared to shallower counterparts of similar computation.

4. Step by Step Network

As discussed in Sec. 3, despite their success, current residual architectures still face challenges at greater depths, potentially impeding the development of deeper networks. Specifically, we present two theoretical observations:

- *The shortcut can “degrade” in very deep residual models.* The model adds too many noisy residual terms to the shortcut, preventing deep layers from accessing input information and propagating their gradients back during training. To mitigate this issue, early features need to be transmitted to deep layers more efficiently.

- *The width-depth trade-off limits deep residual models to insufficient width.* More advanced design is needed to achieve large depth and sufficient width concurrently, without exceeding the computational budget.

Motivated by these analyses, we introduce a progressive modeling strategy to alleviate both challenges simultaneously, presenting the **Step by Step Network (StepsNet)** as an improved generalization of the residual architecture. For better understanding, we first present the StepsNet design in Sec. 4.1, and then offer detailed analyses of how it addresses the two challenges in Sec. 4.2.

4.1. Let the Network think Step by Step

Given an input sequence $x \in \mathbb{R}^{N \times C}$, the architecture of proposed step by step network can be formulated as:

$$y_1 = \mathcal{F}_1(x_1), \quad y_2 = \mathcal{F}_2([y_1, x_2]), \quad (4)$$

where $x_1 \in \mathbb{R}^{N \times d_1}, x_2 \in \mathbb{R}^{N \times d_2}$ are the two parts of x split along the width dimension C , with $d_1 + d_2 = C$ and $x = [x_1, x_2]$. \mathcal{F}_1 and \mathcal{F}_2 denote two residual architecture networks with widths $C_1 = d_1$ and $C_2 = d_1 + d_2$, respectively.

Fig. 4(b) provides an illustration of this architecture. Unlike conventional residual architecture networks that process the entire input $x \in \mathbb{R}^{N \times C}$ directly, our method starts with a *subspace of the input* $x_1 \in \mathbb{R}^{N \times d_1} \subset \mathbb{R}^{N \times C}$, modeling the information in this subspace with $y_1 = \mathcal{F}_1(x_1) \in \mathbb{R}^{N \times d_1}$. This intermediate result y_1 is then combined with x_2 to form the intermediate feature in the *entire space* $[y_1, x_2] \in \mathbb{R}^{N \times C}$, which is processed by \mathcal{F}_2 to obtain the final output $y_2 = \mathcal{F}_2([y_1, x_2]) \in \mathbb{R}^{N \times C}$.

Intuitively, the input x contains diverse information. The proposed method explicitly directs the model to focus first on the first part of information x_1 and “think one step” with network \mathcal{F}_1 , yielding some intermediate results y_1 . On this basis, the second part of information x_2 is taken into consideration and the model can “think another step” to produce

the final output based on y_1 and x_2 . This step by step process shares some similarities with human problem-solving: when tackling complex problems, we often start with part of the conditions, derive preliminary insights, and then proceed to the remaining factors. Therefore, we refer to the proposed structure as Step by Step Network (StepsNet). Additionally, we term x_1, x_2 as *slow path* and *fast path*, respectively, since the information in x_1 takes two thinking steps while x_2 is directly considered by the second step.

Furthermore, the model defined by Eq. (4) can be viewed as a 2-step network, as there are two networks with different widths, \mathcal{F}_1 and \mathcal{F}_2 . More generally, we define the n -step network as follows:

$$\begin{aligned} y_1 &= \mathcal{F}_1(x_1), \\ y_i &= \mathcal{F}_i([y_{i-1}, x_i]), \quad i = 2, \dots, n, \end{aligned} \quad (5)$$

where $x_i \in \mathbb{R}^{N \times d_i}$, $\sum_{i=1}^n d_i = C$, $x = [x_1, \dots, x_n]$, $y_i \in \mathbb{R}^{N \times C_i}$, $C_i = \sum_{j=1}^i d_j$, and y_n denotes the final output. Here, \mathcal{F}_i represents the i -th step residual model with D_i blocks of width C_i , $i = 1, \dots, n$. In practice, we set the width of each step to be $\sqrt{2}$ narrower than its subsequent step, *i.e.* $C_i = C_{i+1}/\sqrt{2}$. The width of the whole network is C , matching the dimensions of x, y_n and \mathcal{F}_n , and the total depth (in blocks) is $\sum_{i=1}^n D_i$. Interestingly, the proposed StepsNet practically forms a generalization of the residual architecture, where different blocks can have various widths, following a narrow-to-wide pattern. Furthermore, the conventional residual models can be viewed as 1-step variants of StepsNet.

4.2. Going Deeper with StepsNet

We present a detailed analysis of how StepsNet overcomes the two challenges of very deep residual models:

The shortcut degradation. As analyzed in Sec. 3, very deep residual models introduce excessive noisy residuals into the shortcut, hindering optimization. StepsNet alleviates this problem with its progressive modeling strategy. Specifically, as shown in Fig. 4(c), in the early steps of StepsNet, the residual terms are added only to a subspace of the entire input x . For example, \mathcal{F}_1 only processes the x_1 subspace, while x_2, \dots, x_n are kept uncorrupted. This effectively constrains the impact of residual terms and directly transmits the input information x_2, \dots, x_n to deep layers $\mathcal{F}_2, \dots, \mathcal{F}_n$. We verify our analysis by calculating the shortcut ratio γ_l in the deep Steps-DeiT model (482 layers). Fig. 2 shows that the value of γ_l in the early layers is close to 1.0 and remains around 0.5 in deep layers, akin to the trend in the 122-layer DeiT. Therefore, StepsNet effectively overcomes shortcut degradation, allowing deep models to be optimized as easily as shallow residual networks.

The limited width. As analyzed in Sec. 3, residual architectures exhibit an inherent trade-off between width and depth. Under fixed computational constraints, residual

models cannot be both wide and deep, resulting in limited width for extremely deep networks. In contrast, with our StepsNet design, it is possible to make a model deeper while fixing its width and computation. For example, consider the two-block residual network depicted in Fig. 4(a). We could change the first block into two narrower blocks with equivalent computation while preserving the second block unchanged. In this way, the model is turned into a 2-step StepsNet with 3 blocks while maintaining the original width and computation. Moreover, applying this process recursively yields networks with $3, 4, \dots, n$ steps. Therefore, the proposed StepsNet architecture allows for a significant model depth increase without sacrificing the entire model width or introducing computation overhead. This design effectively mitigates the width limitation of deep residual networks, enabling the construction of deep models with sufficient width within a fixed computational budget.

5. Experiments

Extremely deep networks are believed to have great expressive power but often fail to yield satisfactory results in practice. In Sec. 3 and Sec. 4, we analyzed the challenges of current very deep networks, and proposed the StepsNet architecture as a possible solution for deep models. In this section, we conduct experimental studies to fully validate the effectiveness of our method.

5.1. Setups

The proposed StepsNet is a simple and clean macro architecture applicable to various models, regardless of their micro block designs. To assess its effectiveness, we implement StepsNet on three representative CNN and Transformer models, including ResNet [19], DeiT [39], and Swin Transformer [25]. Comparisons with advanced methods are also included. We report experimental results on ImageNet-1K classification [9], COCO object detection [24], and ADE20K semantic segmentation [53]. Additionally, we also apply StepsNet to language modeling tasks and conduct experiments using WikiText-103 [28]. For a fair comparison, *all experiment settings are consistent with baseline methods*. Please refer to the Appendix for details.

5.2. Main Results and Broad Comparisons

In this section, we conduct comprehensive experiments to fully evaluate the effectiveness of StepsNet across various model sizes and tasks.

ImageNet classification. We replace the residual architecture of ResNet [19], DeiT [39], and Swin Transformer [25] with our StepsNet, without any other modifications. *Please see detailed model architectures in the Appendix.*

The results are presented in Tab. 1. StepsNet achieves consistent improvements against baseline models. For instance, our Steps-Swin-T surpasses Swin-T by 1.1% in ac-

Method	#Blocks	#Layers	#Params	FLOPs	Top-1
ResNet-18 [19]	8	18	11.7M	1.8G	70.2
ResNet-34	16	34	21.8M	3.7G	75.0
ResNet-50	16	50	25.6M	4.1G	78.8
Steps-ResNet-18	18	38	11.7M	1.8G	71.8
Steps-ResNet-34	34	70	21.8M	3.7G	76.0
Steps-ResNet-50	34	104	25.6M	4.1G	79.9
DeiT-T [39]	12	62	5.7M	1.3G	72.2
DeiT-S	12	62	22.1M	4.6G	79.9
DeiT-B	12	62	86.6M	17.6G	81.8
Steps-DeiT-T	24	122	5.7M	1.3G	73.2
Steps-DeiT-S	24	122	22.1M	4.7G	81.0
Steps-DeiT-B	24	122	86.7M	17.9G	82.7
Swin-T [25]	12	65	28.3M	4.5G	81.3
Swin-S	24	125	49.6M	8.8G	83.0
Swin-B	24	125	87.8M	15.5G	83.5
Steps-Swin-T	26	135	27.8M	4.5G	82.4
Steps-Swin-S	38	195	49.1M	8.8G	83.6
Steps-Swin-B	38	195	86.7M	15.5G	84.1

Table 1. Comparison of different models on ImageNet-1K. Consistent with ResNet [19], we count each convolutional or linear layer as one layer. Therefore, each Transformer block (see Fig. 3) is counted as $3(QKV, \text{Attention}, \text{Out}) + 2(\text{MLP}) = 5$ layers.

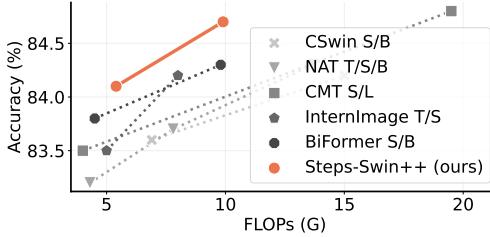


Figure 5. Comparison with advanced methods on ImageNet-1K.

curacy, and Steps-Swin-S outperforms Swin-B while using only 56% of the parameters and FLOPs. The 302-layer Steps-DeiT-B model matches the performance of DeiT-B with 50% fewer parameters and FLOPs. These results suggest that StepsNet tends to be a superior alternative to the widely adopted residual architecture. Furthermore, the results also demonstrate the incredible power of network depth, as our method introduces no additional operations but solely benefits the model with increased depth.

To further explore the capacity of StepsNet and compare with advanced methods, we integrate components commonly used in advanced models [12, 55] like LPU [12] and ConvFFN [12] into the baseline Swin model. As shown in Fig. 5, the enhanced Steps-Swin++ models achieve highly competitive results and surpass various advanced designs, highlighting the superior capacity of our method.

COCO object detection. In Tab. 2, we provide the object detection results of different model sizes and detection heads. Consistent with the trend observed in classification, StepsNet enables the model to benefit from increased depth,

(a) Mask R-CNN Object Detection on COCO								
Method	FLOPs	Sch.	AP ^b	AP ^b ₅₀	AP ^b ₇₅	AP ^m	AP ^m ₅₀	AP ^m ₇₅
Swin-T [25]	267G	1x	43.7	66.6	47.7	39.8	63.3	42.7
Steps-Swin-T	266G	1x	44.1	67.0	48.3	40.2	63.8	42.9
Swin-S	359G	1x	45.7	67.9	50.4	41.1	64.9	44.2
Steps-Swin-S	357G	1x	46.3	68.8	50.9	41.9	65.7	45.0
Swin-B	503G	1x	46.9	-	-	42.3	-	-
Steps-Swin-B	501G	1x	47.1	69.6	51.6	42.5	66.6	45.6

(b) Cascade Mask R-CNN Object Detection on COCO								
Method	FLOPs	Sch.	AP ^b	AP ^b ₅₀	AP ^b ₇₅	AP ^m	AP ^m ₅₀	AP ^m ₇₅
Swin-T [25]	745G	3x	50.4	69.2	54.7	43.7	66.6	47.3
Steps-Swin-T	745G	3x	51.2	70.1	55.8	44.3	67.4	48.3
Swin-S	837G	3x	51.9	70.7	56.3	45.0	68.2	48.8
Steps-Swin-S	836G	3x	52.3	71.1	57.0	45.2	68.5	49.1
Swin-B	981G	3x	51.9	70.5	56.4	45.0	68.1	48.9
Steps-Swin-B	979G	3x	52.5	71.4	57.2	45.4	69.1	49.0

Table 2. Results on COCO dataset. The FLOPs are computed over the backbone, FPN, and detection head with a 1280×800 input.

Semantic Segmentation on ADE20K					
Backbone	Method	FLOPs	#Params	mIoU	mAcc
Swin-T [25]	Upennet	945G	60M	44.5	55.6
Steps-Swin-T	Upennet	941G	58M	45.5	56.8

Table 3. Results of semantic segmentation. The FLOPs are computed over encoders and decoders with a 512×2048 input.

Language Modeling on WikiText-103				
Backbone	#Params	#Blocks	#Layers	PPL _↓
Transformer [41]	30M	6	31	25.28
Steps-Transformer	30M	12	61	24.39
Transformer	44M	6	31	24.45
Steps-Transformer	45M	12	61	24.00
Transformer	63M	12	61	22.05
Steps-Transformer	64M	24	121	21.64

Table 4. Results of language modeling perplexity on WikiText-103. Parameters from the word embedding layer are included in the total parameter count.

delivering improved results in all settings. This validates its effectiveness in dense prediction scenarios.

ADE20K semantic segmentation. We report the results on ADE20K dataset in Tab. 5. Similar to the object detection task, our method yields better results in semantic segmentation, further verifying the effectiveness of StepsNet architecture and the importance of depth.

Language modeling. Recent research [48] suggests that deeper language models can achieve superior performance in reasoning tasks compared to their shallower counterparts, calling for the exploration of model depth in language modeling. As shown in Tab. 4, we apply StepsNet architecture to decoder-only Transformer [41] to empirically assess its effectiveness in language modeling. The models are trained on WikiText-103. In terms of language modeling perplexity, StepsNet consistently outperform vanilla Transformer across different model sizes. This confirms the effectiveness

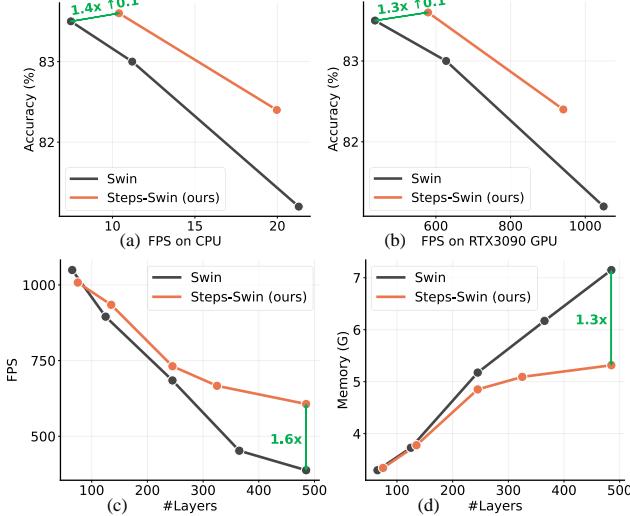


Figure 6. Speed and memory measurements. (a, b) Throughput-accuracy curves on CPU and RTX3090 GPU. (c) Throughput for models of varying depths, tested on an RTX3090 GPU. (d) Memory usage of models with different depths is measured as the change in total memory when batch size increases from 32 to 64.

of our design in language modeling task and demonstrates the benefits of increased model depth.

Throughput and memory. We further present real speed and memory usage measurements in Fig. 6. StepsNet achieves a better trade-off between throughput and accuracy, providing a 1.3x-1.4x speedup with improved performance on both CPU and RTX3090 GPU. In Fig. 6(c) and (d), we illustrate the throughput and memory usage of models in Tab. 5(b). In residual architectures, deeper models tend to exhibit lower throughput and increased memory usage. Our design mitigates this issue, enabling a 1.6x speedup and 1.3x less memory usage at large depths.

5.3. Building Deeper Models

In Sec. 5.2, we validated our method using existing residual models. In this section, we develop deeper models at the DeiT-T/S scale ($1.0 \sim 5.0\text{G FLOPs}$) to further evaluate StepsNet at greater depths with manageable computational costs. We conduct two series of experiments:

- *Deeper models with fixed width.* The standard DeiT-T model employs 192 width and 12 blocks. To build deeper model at DeiT-T scale, we first reduce the width to 136 (by a factor of $\sqrt{2}$) while doubling the depth to 24 blocks. Based on this model, we progressively increase model depth to 96 blocks while keeping width fixed, establishing the baseline results. To ensure a fair comparison, we set the width of Steps-DeiT to $136 \times 2^{\frac{i-5}{2}}$, $i = 1, \dots, 7$, maintaining a computational cost similar to the corresponding baseline model at the same depth.

- *Deeper models with fixed FLOPs.* We start from the stan-

(a) Going Deeper with Fixed Width									
DeiT [39]				Steps-DeiT					
#B	#L	#Params	FLOPs	Acc.	#B	#L	#Params	FLOPs	Acc.
24	122	5.6M	1.3G	72.9	24	122	5.9M	1.3G	73.3
36	182	8.3M	2.0G	75.2	36	182	8.8M	2.0G	77.0
48	242	11.0M	2.6G	75.7	48	242	11.3M	2.6G	78.4
72	362	16.4M	4.0G	75.8	72	362	16.6M	3.8G	79.4
96	482	21.8M	5.3G	75.6	96	482	22.0M	5.1G	80.0

(b) Going Deeper with Fixed FLOPs									
Swin [25]				Steps-Swin					
#B	#L	#Params	FLOPs	Acc.	#B	#L	#Params	FLOPs	Acc.
12	65	28.3M	4.5G	81.3	14	75	27.8M	4.5G	81.8
24	125	25.1M	4.5G	81.8	26	135	27.8M	4.5G	82.4
48	245	24.5M	4.5G	80.9	48	245	28.8M	4.6G	82.4
72	365	24.1M	4.5G	80.0	64	325	28.8M	4.6G	82.2
96	485	24.3M	4.6G	79.8	96	485	28.8M	4.6G	82.1

Table 5. Deeper model test accuracy on ImageNet. #B and #L denote number of blocks and layers, respectively.

dard Swin-T and hold FLOPs constant, incrementally increasing model depth while reducing the width. This allows a controlled examination of width-depth trade-off. The Steps-Swin-T is configured similarly.

The results are provided in Tab. 5 and Fig. 1. In *the first setup*, where width is fixed, the residual architecture DeiT exhibits modest accuracy gains beyond 200 layers. Notably, this is not mainly due to overfitting, as similar trends also appear in training accuracy (see Fig. 1). We attribute this to the optimization difficulties brought by the shortcut degradation analyzed in Sec. 3. In contrast, our design alleviates this problem, enabling Steps-DeiT to achieve increasingly better results at greater depth. In *the second setup*, we can observe a clear width-depth trade-off under fixed FLOPs in residual architectures. Specifically, Swin model achieves peak performance around 120 layers but degrades significantly as depth further increases. As discussed in Sec. 4, StepsNet mitigates this trade-off, helping the model benefit from depth more effectively. This is verified by the results that Steps-Swin model consistently outperforms across all depths and achieves fairly strong accuracy at great depth. These results support our analyses in Sec. 3 and Sec. 4, and demonstrate the effectiveness of StepsNet.

5.4. Analysis of StepsNet

In this section, we offer additional analysis and discussion of StepsNet’s behavior and effectiveness.

Slow and fast path. We offer analytical results to demystify the role of slow and fast paths (defined in Sec. 4.1) in StepsNet. Specifically, we employ the pretrained 3-step Steps-DeiT-S, setting n channels of the slow (x_1) or fast path (x_2, x_3) to zero, and directly evaluating the model’s accuracy. The results are depicted in Tab. 6. Setting channels in the slow path to zero leads to a significant performance drop, whereas masking fast path channels has a relatively minor effect. This indicates that StepsNet can learn

Mask Out Channels	None	32	64	96	128	160
Slow Path	81.0	77.3	68.3	49.0	20.4	4.4
Fast Path	81.0	80.9	80.9	80.8	80.7	80.3

Table 6. Performances on ImageNet-1K when masking out the first or last n channels of input x before the StepsNet architecture in Steps-DeiT-S. Models are tested without retraining.

Dropping Position	\mathcal{F}_1	\mathcal{F}_2	\mathcal{F}_3
Acc.	77.0	78.1	80.5

Table 7. Dropping similar computations in the three networks of Steps-DeiT-S. Models are tested without retraining.

to route essential information to the slow path to process it with more blocks and steps, while placing complementary information in fast channels to consider them later.

Contribution of each step. We drop similar computations in $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ of the pretrained Steps-DeiT-S to assess the contribution of similar computations in different steps. Specifically, we drop the last 4 blocks in \mathcal{F}_1 , 2 blocks in \mathcal{F}_2 , or 1 block in \mathcal{F}_3 (as they have similar computations), and evaluate the model’s accuracy. Tab. 7 shows that dropping similar computations in earlier steps leads to a more significant performance drop. This suggests that similar computations play a more important role when placed in narrower networks like \mathcal{F}_1 . Therefore, changing the 1-step residual model into multi-step StepsNet enables similar computations to perform better, leading to StepsNet’s effectiveness.

5.5. Ablation Studies

In this section, we conduct ablation studies of the key designs in StepsNet. We report ImageNet-1K classification results using Steps-DeiT-S and Steps-Swin-T.

The number of steps. In Tab. 8, we examine the effect of increasing the number of steps in StepsNet. Results indicate that even a 2-step network outperforms the baseline residual model (equivalent to 1-step StepsNet). Appropriately introducing more steps can further benefit the model with depth. In practice, we choose 3 steps as the default, which doubles the model depth with little computation overhead.

Computation allocation. We further evaluate the computation allocation of StepsNet in Tab. 9. Steps-DeiT-S comprises three networks, $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$, with widths $\frac{C}{2}, \frac{C}{\sqrt{2}}, C$ and depths D_1, D_2, D_3 , respectively. We gradually reduce the depth of the widest sub-model \mathcal{F}_3 and correspondingly increase the depths of $\mathcal{F}_1, \mathcal{F}_2$ to maintain similar complexity. We find that a simple $(12, 6, 6)$ allocation is effective, which is achieved by allocating half of each subsequent network’s computation to the previous one.

The reverse design. StepsNet uses a narrow-to-wide stacking strategy, where the width of each network \mathcal{F}_i increases with i , as shown in Fig. 4(c). One reverse design is stacking the blocks wide to narrow, which can be formulated as:

$$\begin{aligned} x_{i+1}, y_i &= \text{Split}(\mathcal{F}_i(x_i)), \quad i = 1, \dots, n-1, \\ y_n &= \mathcal{F}_n(x_n), \end{aligned} \quad (6)$$

#Steps	#Blocks	#Layers	#Param	FLOPS	Acc.
1	12	62	22.1M	4.6G	79.9
2	18	92	22.1M	4.7G	80.8
3	24	122	22.1M	4.7G	81.0
4	30	152	22.1M	4.8G	81.0
5	36	182	22.1M	4.8G	81.2

Table 8. Ablation on the number of steps.

(D_1, D_2, D_3)	#Blocks	#Layers	#Param	FLOPS	Acc.
$(0, 0, 12)$	12	62	22.1M	4.6G	79.9
$(2, 1, 11)$	14	72	22.1M	4.6G	80.5
$(6, 3, 9)$	18	92	22.1M	4.7G	80.8
$(12, 6, 6)$	24	122	22.1M	4.7G	81.0
$(18, 9, 3)$	30	152	22.1M	4.8G	81.0
$(22, 11, 1)$	34	172	22.1M	4.9G	81.0

Table 9. Ablation on computation allocation.

Name	#Blocks	#Layers	#Param	FLOPS	Acc.
Reverse	26	135	27.8M	4.5G	81.3
Steps-Swin-T	26	135	27.8M	4.5G	82.4

Table 10. Ablation on the key designs of StepsNet.

where $x_1 \in \mathbb{R}^{N \times C}$ denotes the input and $[y_1, \dots, y_n] \in \mathbb{R}^{N \times C}$ is the output. The results in Tab. 10 show that StepsNet significantly outperforms the reverse design despite having identical parameters and FLOPs. We attribute this disparity to different model depths. In StepsNet, although x_i passes only through $\mathcal{F}_j, j = i, \dots, n$, it can integrate with the features from all previous layers y_{i-1} through $y_i = \mathcal{F}_i([y_{i-1}, x_i])$ (see Eq. (5)). Therefore, each output channel of StepsNet contains the deepest feature from x_1 , processed by all sub-models $\mathcal{F}_i, i = 1, \dots, n$. In the reverse design, however, only y_n passes through all layers, while $y_i, i = 1, \dots, n-1$ are comparatively shallow. Therefore, the results validate the effectiveness of our narrow-to-wide stacking strategy and confirm that StepsNet benefits the model with increased depth.

6. Conclusion

Developing deeper, more expressive networks is a fundamental research topic in deep learning. Despite great advances, modern residual networks still struggle to deliver anticipated results at large depths. In this paper, we identify two key challenges of deep residual models: shortcut degradation and limited width. Based on these analyses, we introduce a generalized and improved residual architecture, Step by Step Network (StepsNet), as a possible solution for deep models. Our method effectively alleviates the identified limitations and is compatible with various models and micro designs. Empirical validations across various tasks confirm that StepsNet benefits models with increased depth, suggesting it as a simple, superior, and scalable generalization of the widely adopted residual architectures.

Acknowledgement

This work is supported in part by the National Key R&D Program of China under Grant 2024YFB4708200, the National Natural Science Foundation of China under Grants U24B20173 and 62276150, and the Scientific Research Innovation Capability Support Project for Young Faculty under Grant ZYGXQNJSKYCXNLZCXM-I20.

References

- [1] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, 2021. [2](#), [3](#)
- [2] Yongqiang Cai. Achieve the minimum width of neural networks for universal approximation. In *ICLR*, 2023. [2](#), [4](#)
- [3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018. [1](#)
- [4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. [2](#)
- [5] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021. [2](#)
- [6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, 2020. [1](#)
- [7] Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In *NeurIPS*, 2020. [3](#)
- [8] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *NeurIPS*, 2011. [1](#), [2](#)
- [9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. [5](#), [1](#)
- [10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. [1](#), [2](#)
- [11] Ronen Eldan and Ohad Shamir. The power of depth for feed-forward neural networks. In *Conference on learning theory*, 2016. [1](#), [2](#)
- [12] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. Cmt: Convolutional neural networks meet vision transformers. In *CVPR*, 2022. [6](#), [1](#)
- [13] Dongchen Han, Xuran Pan, Yizeng Han, Shiji Song, and Gao Huang. Flatten transformer: Vision transformer using focused linear attention. In *ICCV*, 2023. [2](#)
- [14] Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. In *NeurIPS*, 2024. [2](#)
- [15] Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective. In *NeurIPS*, 2024. [2](#)
- [16] Dongchen Han, Tianzhu Ye, Yizeng Han, Zhuofan Xia, Shiji Song, and Gao Huang. Agent attention: On the integration of softmax and linear attention. In *ECCV*, 2024. [2](#)
- [17] Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 2019. [2](#), [4](#)
- [18] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *NeurIPS*, 2018. [3](#)
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. [1](#), [2](#), [3](#), [5](#), [6](#)
- [20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. [1](#)
- [21] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. [2](#)
- [22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. [2](#)
- [23] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022. [2](#)
- [24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. [5](#), [1](#)
- [25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. [1](#), [2](#), [5](#), [6](#), [7](#)
- [26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. [1](#)
- [27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. [1](#)
- [28] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017. [5](#), [1](#)
- [29] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *NeurIPS*, 2014. [2](#)
- [30] Sejun Park, Chulhee Yun, Jaeho Lee, and Jinwoo Shin. Minimum width for universal approximation. In *ICLR*, 2021. [2](#), [4](#)
- [31] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. [1](#)
- [32] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *NeurIPS*, 2016. [1](#), [2](#)
- [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen

- Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. [1](#) [2](#)
- [34] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, 2017. [1](#) [2](#)
- [35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. [1](#)
- [36] Thiago Serra, Christian Tjandraatmadja, and Sri Kumar Ramalingam. Bounding and counting linear regions of deep neural networks. In *ICML*, 2018. [2](#)
- [37] Sam Shleifer, Jason Weston, and Myle Ott. Normformer: Improved transformer pretraining with extra normalization. *arXiv preprint arXiv:2110.09456*, 2021. [2](#)
- [38] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, 2016. [2](#)
- [39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. [1](#), [5](#), [6](#), [7](#), [2](#)
- [40] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021. [1](#), [2](#)
- [41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. [6](#)
- [42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. [1](#), [2](#)
- [43] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *TPAMI*, 2024. [2](#), [3](#)
- [44] Wenhui Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021. [2](#)
- [45] Yulin Wang, Yang Yue, Yang Yue, Huanqian Wang, Haojun Jiang, Yizeng Han, Zanlin Ni, Yifan Pu, Minglei Shi, Rui Lu, et al. Emulating human-like adaptive vision for efficient and flexible machine visual perception. *Nature Machine Intelligence*, 2025. [2](#)
- [46] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. [1](#)
- [47] Enze Xie, Wenhui Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. [2](#)
- [48] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. In *ICLR*, 2025. [6](#)
- [49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. [1](#)
- [50] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. [1](#)
- [51] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *ICLR*, 2019. [2](#), [3](#)
- [52] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. [1](#)
- [53] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2019. [5](#), [1](#)
- [54] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021. [1](#), [2](#)
- [55] Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson WH Lau. Biformer: Vision transformer with bi-level routing attention. In *CVPR*, 2023. [6](#), [1](#)

Step by Step Network

Supplementary Material

A. Datasets and Experiment Details

ImageNet classification. The ImageNet-1K [9] dataset consists of 1.28M training images and 50K validation images. For a fair comparison, we train all our models under the same settings as the baseline models [25, 39]. Specifically, our models are trained from scratch for 300 epochs using the AdamW [27] optimizer. We employ a cosine learning rate decay with a 20-epoch linear warm-up and a weight decay of 0.05. The base learning rate is set to 1×10^{-3} for a batch size of 1024 and scaled linearly with batch size. Augmentation strategies include RandAugment [6], Mixup [50], CutMix [49], and random erasing [52].

COCO object detection. The COCO dataset [24], with 118K training and 5K validation images, is a common benchmark for object detection and instance segmentation. We follow the standard Mask R-CNN [20] and Cascade Mask R-CNN [3] training settings to conduct experiments.

ADE20K semantic segmentation. ADE20K [53] is a well-established benchmark for semantic segmentation which encompasses 20K training images, 2K validation images and 150 semantic categories. UPerNet [46] is used as the segmentation framework and the same training setting as Swin Transformer [25] is adopted.

Language modeling. WikiText-103 [28] is a standard text dataset used for word-level language modeling. It contains 103M tokens from English Wikipedia articles. We train models on WikiText-103 with a vocabulary of size 50K, sequence length of 128, and batch size of 0.128M tokens.

B. Model Architectures

We provide the architectures of ResNet [19], DeiT [39], Swin [25] and our Steps-ResNet, Steps-DeiT, Steps-Swin in Tab. 12, Tab. 13 and Tab. 14. We simply apply the macro StepsNet architecture to these models while keeping their micro block design, patch embedding layers, and other network components unchanged. The macro design of Steps-Swin++ is the same as Steps-Swin, while its micro block employs components commonly used in advanced models [12, 55] like LPU [12] and ConvFFN [12].

C. Additional Results

StepsNet at different stages. For hierarchical models like Swin, we additionally investigate applying StepsNet at different stages. As depicted in Tab. 11, substituting StepsNet in the last two stages improves performance, while changes in the first two stages yield minimal effect. This could be attributed to the larger widths of last two stages, which sup-

port relatively wide sub-networks and enable StepsNet to better benefit the model with increased depth.

Stages w/ StepsNet				#Blocks	#Layers	#Param	FLOPS	Acc.
Stage1	Stage2	Stage3	Stage4					
✓				13	70	28.3M	4.5G	81.3
	✓			13	70	28.3M	4.5G	81.3
		✓		18	95	28.3M	4.5G	81.5
			✓	20	105	27.8M	4.5G	82.2
			✓	26	135	27.8M	4.5G	82.4
Swin-T				12	65	28.3M	4.5G	81.3

Table 11. Applying StepsNet at different stages of Swin-T model.

Model	Stage 1	Stage 2	Stage 3	Stage 4
ResNet-18 [19]	64×1	128×1	256×1	512×1
ResNet-34 [19]	64×2	128×3	256×5	512×2
ResNet-50 [19]	64×2	128×3	256×5	512×2
Steps-ResNet-18	64×1	128×1	$64 \times 2, 90 \times 1, 128 \times 1, 181 \times 1, 256 \times 0$	$64 \times 2, 90 \times 1, 128 \times 1, 181 \times 1, 256 \times 1, 362 \times 1, 512 \times 0$
Steps-ResNet-34	64×1	128×1	$64 \times 6, 90 \times 3, 128 \times 3, 181 \times 3, 256 \times 2$	$64 \times 2, 90 \times 1, 128 \times 1, 181 \times 1, 256 \times 1, 362 \times 1, 512 \times 1$
Steps-ResNet-50	64×1	128×1	$64 \times 6, 90 \times 3, 128 \times 3, 181 \times 3, 256 \times 2$	$64 \times 2, 90 \times 1, 128 \times 1, 181 \times 1, 256 \times 1, 362 \times 1, 512 \times 1$

Table 12. Architectures of ResNet [19] and Steps-ResNet models of the form $C \times D$, where C is the bottleneck dimension and D denotes number of blocks. Notably, in ResNet, the first block of each stage is a downsample block, which is not listed in this table.

Model	Architecture
DeiT-T [39]	$\begin{bmatrix} C 192 \\ H 3 \end{bmatrix} \times 12$
DeiT-S [39]	$\begin{bmatrix} C 384 \\ H 6 \end{bmatrix} \times 12$
DeiT-B [39]	$\begin{bmatrix} C 768 \\ H 12 \end{bmatrix} \times 12$
Steps-DeiT-T	$\begin{bmatrix} C 96 \\ H 2 \end{bmatrix} \times 12, \begin{bmatrix} C 136 \\ H 2 \end{bmatrix} \times 6, \begin{bmatrix} C 192 \\ H 3 \end{bmatrix} \times 6$
Steps-DeiT-S	$\begin{bmatrix} C 192 \\ H 3 \end{bmatrix} \times 12, \begin{bmatrix} C 272 \\ H 4 \end{bmatrix} \times 6, \begin{bmatrix} C 384 \\ H 6 \end{bmatrix} \times 6$
Steps-DeiT-B	$\begin{bmatrix} C 96 \\ H 2 \end{bmatrix} \times 12, \begin{bmatrix} C 136 \\ H 2 \end{bmatrix} \times 12, \begin{bmatrix} C 192 \\ H 3 \end{bmatrix} \times 12, \begin{bmatrix} C 272 \\ H 4 \end{bmatrix} \times 12, \begin{bmatrix} C 384 \\ H 6 \end{bmatrix} \times 12$

Table 13. Architectures of DeiT [39] and Steps-DeiT models. C and H denote block width and number of heads, respectively.

Model	Stage 1	Stage 2	Stage 3	Stage 4
Swin-T [25]	$\begin{bmatrix} C 96 \\ H 3 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 2$	$\begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 6$	$\begin{bmatrix} C 768 \\ H 24 \end{bmatrix} \times 2$
Swin-S [25]	$\begin{bmatrix} C 96 \\ H 3 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 2$	$\begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 18$	$\begin{bmatrix} C 768 \\ H 24 \end{bmatrix} \times 2$
Swin-B [25]	$\begin{bmatrix} C 128 \\ H 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C 256 \\ H 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C 512 \\ H 16 \end{bmatrix} \times 18$	$\begin{bmatrix} C 1024 \\ H 32 \end{bmatrix} \times 2$
Steps-Swin-T	$\begin{bmatrix} C 96 \\ H 3 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 6, \begin{bmatrix} C 272 \\ H 8 \end{bmatrix} \times 3, \begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 3, \begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 4, \begin{bmatrix} C 272 \\ H 8 \end{bmatrix} \times 2, \begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 2, \begin{bmatrix} C 544 \\ H 16 \end{bmatrix} \times 2, \begin{bmatrix} C 768 \\ H 24 \end{bmatrix} \times 0$	
Steps-Swin-S	$\begin{bmatrix} C 96 \\ H 3 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 2$	$\begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 6, \begin{bmatrix} C 272 \\ H 8 \end{bmatrix} \times 3, \begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 15, \begin{bmatrix} C 192 \\ H 6 \end{bmatrix} \times 4, \begin{bmatrix} C 272 \\ H 8 \end{bmatrix} \times 2, \begin{bmatrix} C 384 \\ H 12 \end{bmatrix} \times 2, \begin{bmatrix} C 544 \\ H 16 \end{bmatrix} \times 2, \begin{bmatrix} C 768 \\ H 24 \end{bmatrix} \times 0$	
Steps-Swin-B	$\begin{bmatrix} C 128 \\ H 4 \end{bmatrix} \times 2$	$\begin{bmatrix} C 256 \\ H 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C 256 \\ H 8 \end{bmatrix} \times 6, \begin{bmatrix} C 360 \\ H 12 \end{bmatrix} \times 3, \begin{bmatrix} C 512 \\ H 16 \end{bmatrix} \times 15, \begin{bmatrix} C 256 \\ H 8 \end{bmatrix} \times 4, \begin{bmatrix} C 360 \\ H 12 \end{bmatrix} \times 2, \begin{bmatrix} C 512 \\ H 16 \end{bmatrix} \times 2, \begin{bmatrix} C 720 \\ H 24 \end{bmatrix} \times 2, \begin{bmatrix} C 1024 \\ H 32 \end{bmatrix} \times 0$	

Table 14. Architectures of Swin [25] and Steps-Swin models. C and H denote block width and number of heads, respectively.