Neural Substitution for Branch-level Network Re-parameterization

Seungmin Oh[1][0009−0000−1431−2716] and Jongbin Ryu*[1,2][0000−0001−5574−5358]

[1] Department of Artificial Intelligence, Ajou University, Suwon, South Korea
[2] Department of Computer Engineering, Ajou University, Suwon, South Korea
seungmin.oh97@gmail.com; jongbinryu@ajou.ac.kr

Abstract. We propose the neural substitution method for network re-parameterization at the branch-level connectivity. This method learns different network topologies to maximize the benefit of the ensemble effect, as re-parameterization allows for the integration of multiple layers during inference following their individual training. Additionally, we introduce a guiding method to incorporate non-linear activation functions into a linear transformation during re-parameterization. Because branch-level connectivity necessitates multiple non-linear activation functions, they must be infused into a single activation with our guided activation method during re-parameterization. Incorporating the non-linear activation function is significant because it overcomes the limitation of the current re-parameterization method, which only works at block-level connectivity. Restricting re-parameterization to block-level connectivity limits the use of network topology, making it challenging to learn a variety of feature representations. On the other hand, the proposed approach learns a considerably richer representation than existing methods due to the unlimited topology, with branch-level connectivity, providing a generalized framework to be applied with other methods. We provide comprehensive experimental evidence for the proposed re-parameterization approach. Our code is available at https://github.com/SoongE/neural_substitution.

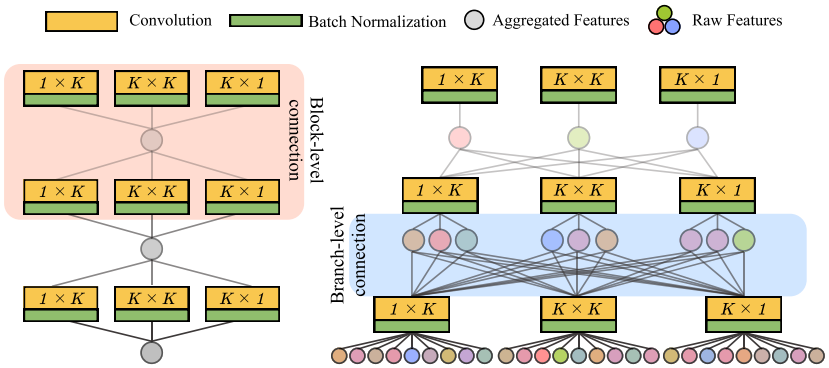
Keywords: Neural substitution · Re-parameterization · Branch-level connectivity

1 Introduction

Heavy architectural design choices employing multi-branch blocks have been thought to be a viable strategy for improving the performance of deep neural networks; however, these strategies have the downside of increasing computation time. As a result, re-parameterization [4, 5, 9, 10, 11, 13, 16, 20, 45] approaches have been investigated to solve this computational complexity while retaining the benefits of the heavy multi-branch design by reducing them to a single branch.

ACNet [9] was the first attempt to introduce re-parameterization, which separately updates the weights of each branch during training and merges the

* Corresponding author



(a) Block-level connectivity (b) Branch-level connectivity

Fig. 1: Progression of block- and branch-level connectivity in re-parameterization network. (a) Block-level aggregates all branch outputs into a single feature, but (b) branch-level generates amplified output features according to the number of branches.

weight of multiple convolution filter branches into a single one during inference. Many subsequent studies [10, 13, 20, 28, 45] exploited more diversified kernels and inception-style block designs, DBB [11], to improve the re-parameterization. These studies are block-level re-parameterization strategies that use multiple convolution branches within a single block. Due to the block-level connectivity that forwards the aggregated multiple outputs to the next block, as shown in Fig. 1a, it is prohibited to connect at the branch-level, as illustrated in Fig. 1b.

We point out as a shortcoming that existing approaches do not benefit from generating richer information like branch-level connectivity. We argue that the branch-level connectivity allows each layer to learn a robust information flow. The reason for this is that branch-level connectivity enables the creation of a nearly unlimited number of sub-network topologies within a network, enhancing the potential advantages of ensembles. Furthermore, learning different sub-network topologies serves as a kind of regularization method, facilitating a network to profit from more performance improvement as the number of branches in each block increases. Based on these considerations, a network should be forwarded with branch-level connectivity; nonetheless, two essential difficulties must be addressed. The first is that as the network's depth increases, so does the number of topologies that must be computed. The last layer of a network with multi-branches will have a near-infinite topology. This leads to an overly complex network design that cannot be employed in practice. The second difficulty is infusing the non-linear activation functions of each branch in the re-parameterization procedure. The linear convolution and batch normalization layer can be reduced by a linear transformation in existing re-parameterization methods; however, non-linear activation functions are not achievable. This means even though the network expands, it doesn't increase non-linearity. We address these two difficulties in

branch-level re-parameterizable networks by proposing 1) neural substitution and 2) guided activation method. The proposed method substitutes multiple input features with reduced numbers to substantially lower the computational cost. Further, using our guided activation method, the non-linear activation function can also be infused by the linear transformation in the re-parameterization procedure. Therefore, the proposed substitution with a guided activation method retains the benefits of the branch-level re-parameterizable networks without heavy architectural design and performs favorably compared to the existing re-parameterization methods. We designate this substitution-based approach for re-parameterization as the Neural Substitution (NS). The contribution of our paper is summarized as follows:

- We introduce neural substitution for network re-parameterization at branch-level connectivity. By substituting local paths in the network training process, we can preserve branch-level connectivity without imposing excessive complexity on the network.
- We present a strategy for reducing the non-linear activation function in linear transformation during the re-parameterization process. We allow the activated neuron of each branch to be determined by a guiding method so that the non-linear activation function can be infused.
- We propose a generalized framework that can be applied to most network designs with branch-level connectivity and re-parameterizable non-linear activation functions. We demonstrate that the proposed approach with the branch-level connectivity performs well compared to existing re-parameterization methods and that they can be adapted to our framework.

2 Related Works

2.1 Network Topology Enhancement

Enhancing the network topology is a commonly employed approach for learning rich feature representation from input images. Inception architecture [36] has been aimed at creating a network topology at a local level and then stacks multiple layers on top of each other. Res2Net [15] extended the local network topology by incorporating multiple channels-wise paths into the conventional bottleneck block of ResNet [18]. Similarly, [24, 31, 35] have integrated multiple paths in their architecture to capture distinct features effectively. It is worth noting that previous research has focused on extending the local network topology. However, in our work, we expand the network topology globally rather than locally.

2.2 Re-parameterization

The re-parameterization is classified into two approaches. First, parameter re-parameterization [4, 5, 16, 28] is an approach for transferring a meta learner to parameters of a new model, such as neural architecture search or knowledge

distillation. The second, which is the subject of this paper, is structural re-parameterization [9, 11, 13, 20, 45] approach replacing the parameters of multiple convolution and batch normalization [21] to new one via a linear transformation.

ACNet [9] produces 1×3, 3×3, and 3×1 convolution branches, which are then merged into a unified 3×3 convolution, preserving the benefit of the multiple branches. Inspired by the success of ACNet and RepVGG [13] presented the re-parameterization method that leverages the multiple 3×3 convolution branches. Subsequently, the Inception-style [36] branches, large kernel convolution, and neural architecture search [47] are applied to DBB [11], MobileOne [40], RepLKNet [12], and RepNas [45] to improve the convolutional neural networks performance. Recently, several studies [6, 26, 37, 39] have adopted the re-parameterization method in their architectural design or training algorithms to improve the recognition performance. YOLO series [25, 42] also leverages the re-parameterization approach for training accurate object detection networks.

3 Overview of Re-parameterization Method

To demonstrate this structural re-parameterization, we must first describe the operations of the convolution and batch-normalization layers as follows:

$$\mathcal{F}(x;\theta) = \frac{\gamma}{\sigma}(x * k + \mu) + \beta, \tag{1}$$

where x denotes the input feature and $\theta = (k, \gamma, \sigma, \mu, \beta)$ stands for the set of parameters of convolution and batch normalization layers. Because the function $\mathcal{F}$ is made up entirely of linear functions, parameters of multiple convolution and batch normalization layers are reducible by following linear transformation $T(\cdot)$:

$$\begin{aligned} T\left(\sum_{i=1}^{N} \theta_i\right) &= T(\theta_1) + T(\theta_2) + ... + T(\theta_N) \\ \text{s.t.} \quad \mathcal{F}(x;\tilde{\theta}) &= \mathcal{F}(x;\theta_1) + \mathcal{F}(x;\theta_2) \ldots + \mathcal{F}(x;\theta_N). \end{aligned} \tag{2}$$

This reduction to a single $\tilde{\theta}$ parameter set serves as the basis for the (structural) re-parameterization method. However, this raises a critical issue that the non-linear activation function can not be used in the re-parameterization method as:

$$\sigma(\mathcal{F}(x;\theta_1)) + \sigma(\mathcal{F}(x;\theta_2)) \neq \mathcal{F}(x;\tilde{\theta}). \tag{3}$$

If the network fails to become non-linear, ultimately, increasing the number of learnable parameters does not hold significant meaning. We will discuss in more detail in Sec. 5.2 whether there is a way to add non-linearity in re-parameterizable convolution blocks.

4 View Point of Connectivity.

In this section, we compare block- and branch-level connectivity regarding learning the feature diversity from the viewpoint of connectivity methods.

4.1 Block-level Connectivity

Many models [9, 11, 13, 15, 36] have employed the concept of using multiple branches in parallel within a single block design. These models add or concatenate feature outputs from each branch and feed them into the following block as follows:

$$y = \sum_{i=1}^{B} \mathcal{F}(x,\ \theta_i); \quad \Theta = \{\theta_i \mid 1 \leq i \leq B\}, \tag{4}$$

where B is the number of blocks and $x \in \mathbb{R}^{C \times H \times W}$ represents the kernel's input with channel C, height H, and width W. In this process, they profit from the ensemble within the block and improve the network performance. Even so, as shown in Eq. 4, the multiple outputs of the convolution layers are merged into one, which becomes a single input feature for the next block. Owing to this matter, they fail to exploit the ensemble effect fully; thus, the sub-network topology only exists in a block illustrated in Fig. 1a. Subsequently, the ensemble effect does not extend to the entire network. The number of sub-network topology generated by the block-level connectivity is counted as:

$$\#\text{ topology with block-level connectivity} : NBD, \tag{5}$$

where N and D represent the number of branches and depth of architecture.

4.2 Branch-level Connectivity

The branch-level connectivity addressed in this paper is to connect the multi-branch outputs in a block to individual branches of the following block without adding or concatenating them, as shown in Fig. 1b. Therefore, unlike Eq. 4, the inputs to each convolution layer are independent as follows:

$$\mathbf{Y} = \{\mathcal{F}(x,\ \theta) \mid x \in \mathbf{X},\ \theta \in \Theta\}; \quad \mathbf{X} = \{x_i \mid 1 \leq i \leq N,\ x_i \in \mathbb{R}^{C \times H \times W}\}, \tag{6}$$

where $|\mathbf{Y}| = NB$ and $\mathbf{X} \in \mathbb{R}^{N \times C \times H \times W}$. This ensures that the sub-network topologies are present throughout the entire network, and hence, the number of topology cases is denoted as:

$$\#\text{ topology with branch-level connectivity} : NB^D. \tag{7}$$

This allows neural networks to exploit the effectiveness of the ensemble with a much larger topology than block-level connectivity, but it has the critical issue of exponentially growing computational budget. We explain this matter in more detail in Sec. 5.1 regarding the computational cost.

5 Method

5.1 Neural Substitution

Due to the ensemble effect, branch-level connectivity diversifies the topology of the sub-network, enhancing performance even when the network is reduced

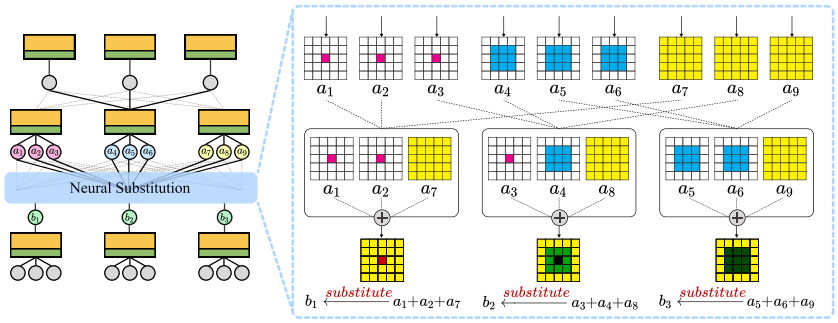


Fig. 2: The conceptual visualization of the stochastic neural substitution. Raw features $a_1, ..., a_9$ in a block is substituted by b_1, b_2, b_3 to reduce the computational complexity of the following blocks. The colors on the right side of the figure represent highlighted areas in each feature map. It is assumed that features from the same convolution block have similar feature maps. Therefore, the re-parameterized weights b_1, b_2, and b_3 show the different sums of highlighted areas following the law of subtractive color mixing.

through re-parameterization. However, as discussed in Sec. 4, there is a clear downside: the memory and computational complexity required becomes prohibitive, making practical use unfeasible. To address this downside, we present the neural substitution method as shown in Alg. 1 and Fig. 2. We set the number of branches to N for the naive branch-level connectivity so that N input feature maps and N multi-branch blocks result in N^2 outputs, as indicated in Line 7 of Alg. 1. In contrast, using the neural substitution method produces only N output features, as shown in Line 11 of Alg. 1.

There is a significant difference between the naive approach and our neural substitution method in terms of the number of input and output features. The former naive approach exhibits exponential growth with the number of blocks, while the latter ensures a consistent number of features equivalent to the inputs. Consequently, the neural substitution method guarantees a fixed number of N outputs in every network, regardless of the number of blocks involved. Additionally, we incorporate stochastic neural substitution to generate a new topology at each iteration, as seen in Lines 12-14 of Alg. 1. This stochastic method enables the training of a network to accommodate unlimited N^B topologies as shown in Eq. 7 with manageable computational cost using only N input and output features.

5.2 Guided Activation Function

Despite successfully reducing the computational budget through our neural substitution method, we still face the challenge that the non-linear activation function cannot be incorporated through re-parameterization, as demonstrated in Eq. 3. Existing re-parameterization approaches have been unable to use branch-level connectivity for the same reason. State-of-the-art (SOTA) methods [9,10,11,

Algorithm 1 Pseudo code for the proposed neural substitution method. We compare the proposed branch-level neural substitution with the stochastic method with other connectivities.

Input $X = \{x_1, x_2, ..., x_N\}$: multi-branch input features
Input $\mathcal{F} = \{f_1, f_2, ..., f_N\}$: multi-branch convolution layers in a block

procedure MULTI-BRANCH BLOCK($X, \mathcal{F}$)
  switch Connectivity **do**
    case Block-level
      $y \leftarrow f_1(x_1) + f_2(x_1) + ... + f_N(x_1) + ... + f_N(x_N)$
      return y ▷ Single output
    case Branch-level (Naive approach)
      return $f_1(x_1), f_2(x_1), ..., f_N(x_1), ..., f_N(x_N)$ ▷ $\#N^2$ outputs
    case Branch-level (Neural substitution **w/o** stoch)
      for $i \leftarrow 0$ to N **do**
        $y_i \leftarrow f_1(x_i) + f_2(x_i) + ... + f_N(x_i)$ ▷ Substitution
      return $y_1, y_2, ..., y_N$ ▷ $\#N$ outputs
    case Branch-level (Neural substitution /**w** stoch)
      $\mu \leftarrow \begin{bmatrix} 1 & ... & 1 \\ \vdots & \ddots & \vdots \\ N & ... & N \end{bmatrix} \in \mathcal{R}^{N \times N}, \quad \nu \leftarrow \begin{bmatrix} 1 & ... & 1 \\ \vdots & \ddots & \vdots \\ N & ... & N \end{bmatrix} \in \mathcal{R}^{N \times N}$
      $\mu \leftarrow \text{shuffle}(\mu), \quad \nu \leftarrow \text{shuffle}(\nu)$ ▷ Shuffle elements of index mapping matrix
      for $i \leftarrow 0$ to N **do**
        $y_i \leftarrow f_{\mu(i,1)}(x_{\nu(i,1)}) + f_{\mu(i,2)}(x_{\nu(i,2)}) + ... + f_{\mu(i,N)}(x_{\nu(i,N)})$ ▷ Substitution
      return $y_1, y_2, ..., y_N$ ▷ $\#N$ outputs

13, 20, 28, 45] employ a reduction strategy by performing re-parameterization on a block-by-block basis, excluding non-linear activation functions. This limitation arises from the fact that non-linear functions cannot be transformed using linear transformations. To address this issue, we propose using a guided activation function, as shown in Alg. 2 and Fig. 3. If a non-linear activation function is incorporated into the re-parameterization method, it cannot reduce the output features of multiple non-linear layers as:

$$\sigma\left(\sum_{i=1}^{N} x_n\right) \neq \sum_{i=1}^{N} \sigma(x_n). \tag{8}$$

Therefore, to reduce multiple non-linear layers using the proposed guided function, we devise a straightforward method to ensure that the activated features remain consistent. We adopt a similar approach to ReLU, which transforms all negative feature values to zero. We sum all the multiple inputs to create a binary guided activation mask, Gm, which retains only positive values. This mask is then

Algorithm 2 Pseudo code for the proposed guided activation. This method guides the input features to be activated similarly when a network is re-parameterized.

Input X: input features

1: **procedure** GUIDEDACTIVATION($X = \{x_1, x_2, ..., x_N\}$)
2: $X_s \leftarrow \sum_{i \in \{0,...,N\}} x_i$ ▷ Input feature when a network was re-parameterized
3: $\sigma \leftarrow \mathbf{1}_{X_s > 0}$ ▷ Guided activation mask
4: $\tilde{X} \leftarrow X \odot \sigma$ ▷ Guiding all branch features' activation according to the mask σ
5: **return** $\tilde{X}$

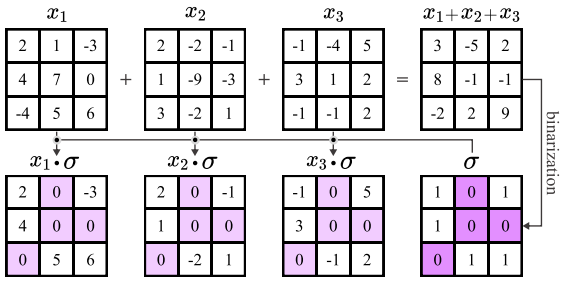


Fig. 3: Workflow of the guided activation function. x_i means feature map of each branch and σ is the guided activation map. The σ guides the activation of x_i.

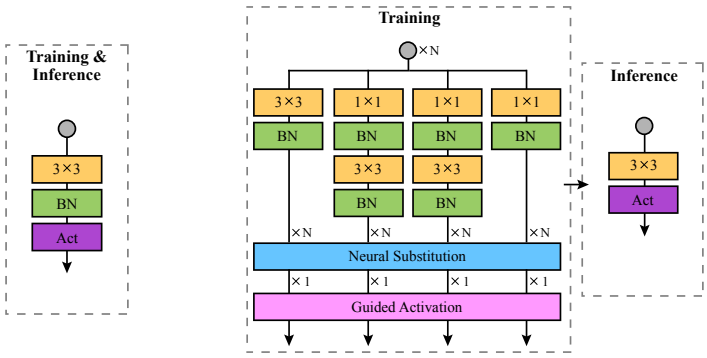
applied to the preceding multiple inputs, regardless of their values as:

$$\begin{aligned} \boldsymbol{GA}(x_i) = Gm \odot x_i; \quad Gm_{j,k} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_{i,j,k} > 0 \\ 0 & \text{otherwise} \end{cases}, \\ \boldsymbol{GA}\left(\sum_{i=1}^{N} x_n\right) = \sum_{i}^{N} \boldsymbol{GA}\left(x_n\right), \end{aligned} \tag{9}$$

where x and $\boldsymbol{GA}$ are each feature and guided activation function. The proposed guided activation function computes the aggregated features of all branches to acquire a guided activation mask, which is applied to activate features of each branch as described in Line 2-4 of Alg. 2. Therefore, by employing this guided function, it is feasible to apply a non-linear activation function to multiple branch features. For clarity, PyTorch code is included in the supplementary material.

5.3 Implementation

The proposed neural substitution and guided activation function ameliorate existing re-parameterization methods [9, 11, 13] by applying our methods to the existing block-level connectivity as shown in Fig. 4b. To accomplish this, we do not aggregate the outputs of each branch in the branch-level re-parameterization methods; instead, we leave the branch outputs connected to the following block



(a) Basic convolutional block

(b) Our convolutional block with the neural substitution

Fig. 4: (a) A basic convolutional block consists of one convolution, batch normalization, and activation layer. (b) Our convolution block comprises multiple convolution and batch normalization layers, a neural substitution, and guided activation during training. When inferring, all convolution and batch normalization blocks are re-parameterized into a single convolution block.

with the guidance of the proposed activation function. Further, the proposed method can also be used to re-parameterize multiple branches of 3×3 convolutions, where the performance boost grows as the number of branches increases. This is due to the proposed method's ability to learn a far broader range of sub-network topologies than existing block-level connectivity. The following section presents experimental results on the effectiveness of this proposed approach.

6 Experiment

In this section, we describe the dataset and details of our experiments, then compare the performance of our branch-level connectivity approach to the existing re-parameterization methods in Sec. 6.2, and conduct an ablation study in Sec. 6.3 to demonstrate whether each component of our proposed method contributes to performance improvement. Finally, we introduce the analysis of the diversified representation of learned features and weight parameters due to our approach in Sec. 6.4. All architectures used in the experiments, after re-parameterization (*i.e.*, at inference time), maintain the same resources such as parameters, FLOPs, and throughput as the original backbone. Therefore, our re-parameterization approach incurs extra resource overhead during network training, yet it has no overhead during inference. It is worth emphasizing that the main advantage of the re-parameterization method is that the original network can be used without any overhead for inference.

Table 1: Experimental results of the re-parameterization network architectures on the CIFAR100 dataset. We report Top-1 accuracy (%) for all architectures.

Architecture	Reset18	ResNet34	ResNet50	ResNext50	MobileNet
Baseline	71.90	73.86	75.52	75.01	66.42
ACNet [9]	72.82	74.68	76.02	76.13	66.65
DBB [11]	73.70	74.62	75.56	76.41	66.69
Ours NS	**75.36**	**75.72**	**77.72**	**77.64**	**67.33**

Table 2: Experimental results of block- and branch-level re-parameterization network architectures on the ImageNet.

Architecture	Reset18	ResNet50	MobileNet
Baseline	69.11	77.31	70.45
ACNet [9]	70.02	77.30	71.12
DBB [11]	70.13	77.09	70.78
Ours NS	**70.62**	**77.66**	**72.06**

6.1 Dataset and Setup

Our experiments employ two standard visual classification datasets: ImageNet [8] and CIFAR100 [23]. In the ImageNet dataset, networks are trained with a batch size of 2048 during 100 epochs. In the CIFAR100 dataset, all networks adopt a batch size of 1024 with 100 epochs. In addition, we conduct the experiment on transfer learning by using the vision encoder of CLIP [33]. It demonstrates that the proposed neural substitution works well on multi-layer perceptron (MLP). We use a total of 9 datasets, such as CIFAR100 [23], Country211 [38], Describable Textures Dataset (DTD) [7], FGVC-Aircraft [30], Food101 [1], Oxford-IIIT Pet [32], PatchCamelyon [3], StanfordCars [22], and ImageNet [8]. We utilize the MS-COCO [27] for object detection and instance segmentation and the PASCAL VOC 2012 [14] for the semantic segmentation task. The details of the training setups used in this paper are provided in the supplementary material.

6.2 Experimental Results

We utilize ResNet [18], ResNext [44], and MobileNetV1 [19] as the backbone networks and replace every convolution block to multi-branched one as shown in Fig. 4. We compare ours with three SOTA re-parameterization architectures.

Comparing to SOTA We compared the performance improvement using different block-level connectivity re-parameterization architectures across various backbones. As shown in Tab. 1, our Neural Substitution (NS) architecture consistently outperforms other architectures. It is notable that after re-parameterization,

Table 3: Implementation on existing branch-level connectivity architectures. MobileOne uses its own backbone, with the rest being based on the ResNet18.

Architecture	CIFAR100	ImageNet
MobileOne [40]	67.26	72.30
MobileOne + NS (w/ stoch)	**67.48**	**72.76**
ACNet [9]	72.82	70.02
ACNet + NS (w/ stoch)	**73.59**	**70.41**
DBB [11]	73.70	70.13
DBB + NS (w/ stoch)	**74.83**	**70.75**

Table 4: Transfer learning accuracy of NS MLP on 9 datasets. To leverage NS's characteristics, which require the creation of multiple topologies, we choose to use multiple layers of MLP as visual adapters instead of a single classifier.

Architecture	C100	Co211	DTD	FGVC	Food	Pet	Pcam	Cars	INet	Avg.
FC	77.12	**26.09**	70.48	42.00	88.20	88.77	78.37	80.36	75.08	69.61
MLP/FC	78.29	23.56	**71.65**	48.55	88.27	**90.92**	67.17	**82.47**	76.55	69.71
NS MLP/FC	**79.77**	24.74	70.59	**50.86**	**88.86**	90.57	**83.37**	82.34	**76.81**	**71.99**

the number of parameters and other computing power remains the same across all architectures, which is the main benefit of the re-parameterization method. Nonetheless, NS increased accuracy by 2.2%point over the baseline and 1.4%point over the SOTA. On ImageNet, we see the NS always performs better than block-level connectivity in Tab. 2, with an average performance gain of about 1.0%point over the baseline. All experiments show higher performance gains at the branch-level, especially on lightweight models.

Restructuring by branch-level connectivity design Furthermore, we enhance the performance of the commonly used re-parameterization method by incorporating branch-level connectivity rather than block-level connectivity. In Tab. 3, the improvements in performance on the ImageNet and CIFAR100 are demonstrated when the re-parameterization architecture at the block level is replaced with a branch-level design. Notably, our branch-level connectivity consistently outperforms block-level connectivity across all architectures, leading to significant performance improvements compared to existing designs. It is clear that our approach consistently enhances performance, regardless of the specific re-parameterization network design and backbones.

Neural substitution on MLP To show the effectiveness of the proposed neural substitution in the Vision Transformer architecture, we perform further experiments using the ViT-B/32 of CLIP [33] by applying a linear classifier and a 4x-scale MLP visual adapter. As a result, we demonstrate that our neural substitution improves the performance of the original model, as shown in Tab. 4.

Table 5: Experimental results of object detection and instance segmentation.

Head	Architecture	BoxAP	MaskAP
Faster [34]	Baseline	32.3	-
	Ours NS	32.5	
Mask [17]	Baseline	33.6	31.6
	Ours NS	34.2	32.1
Cascade [2]	Baseline	37.1	33.3
	Ours NS	37.6	33.7

Table 6: Experimental results of semantic segmentation

Head	Architecture	mIoU	mACC
FCN [29]	Baseline	36.6	47.7
	Ours NS	41.6	54.7
PSP [46]	Baseline	45.0	58.6
	Ours NS	47.5	62.7
Uper [43]	Baseline	46.4	61.0
	Ours NS	48.4	64.3

Table 7: Experimental results between block- and branch-level connectivity regarding the branches.

Branch	Block-level	Branch-level NS
Conv×2	73.16	73.30 (+0.14)
Conv×3	73.53	74.10 (+0.57)
Conv×4	74.15	74.58 (+0.43)
Conv×5	74.20	74.79 (+0.59)

Table 8: Experimental results for the ablation study of the proposed neural substitution with the stochastic.

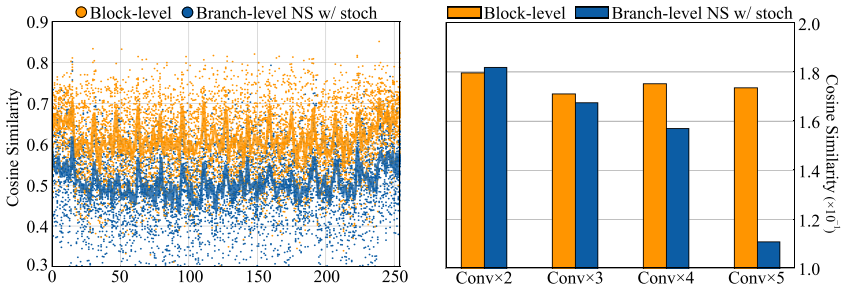
Connectivity	DBB [11]	ACNet [9]
Block-level	70.13	70.02
NS w/o stoch	70.56	70.21
NS w/ stoch	70.75	70.41

Neural substitution on dense prediction We perform experiments on the dense prediction tasks. For all experiments, ResNet-50 is used for the backbone of dense prediction heads. For the object detection and instance segmentation tasks, we train each network with 12 epochs for all heads [2, 17, 34]. Tab. 5 shows that our NS model performs consistently better than the baseline in all cases. We adopt a training setup of 20k interaction for the semantic segmentation task [29, 43, 46]. As shown in Tab. 6, our NS model also outperforms the baseline.

6.3 Ablation Study

Branch-level connectivity Tab. 7 shows the observed performance improvement according to the number of branches. Multiple 3×3 convolutions are employed for all multi-branch designs. Once the number of 3×3 convolutions surpasses four, the block-level connection experiences performance saturation. However, our branch-level NS architecture improves performance over the block-level architectures in all cases, including the five branches.

Stochastic method of neural substitution Tab. 8 presents the experimental results of the stochastic method, which randomizes the substitution at every iteration. According to these results, the proposed stochastic method is beneficial in all cases. We observe that employing the proposed NS with stochastic improves the network's performance by 0.2 to 0.6%point compared to block-level connectivity.



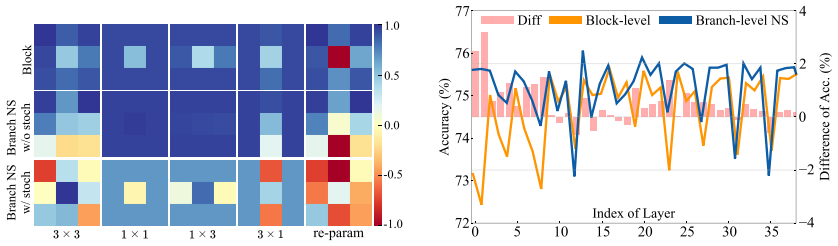
(a) Similarity Block (b) Similarity Branch

Fig. 5: Feature similarity comparison. (a) The cosine similarity is measured on the features between blocks at different depths of a network. (b) The cosine similarity between features of different branches of a block is computed.

6.4 Analysis

Block and Branch Similarity To establish that our branch-level connectivity uses more topologies than the block-level approach, we examine the feature and weight similarities of branches within a block. We conduct this similarity analysis to assess the feature diversity in the re-parameterization method. The more topologies a network has, the more diverse features with low similarity it will learn between branches within a block. This enhanced feature diversity results in lower feature similarity and leads to better performance for re-parameterized networks. In this regard, we estimate the feature similarities. In Fig. 5a, the dot points represent cosine similarities between pixels at corresponding locations in feature maps from layers at varying depths. While the similarities in block-level connectivity are scattered above 0.6, our branch-level connectivity exhibits far lower similarity values. Fig. 5b shows the cosine similarity between features from different branches in a block. As the number of branches increases, the similarity in our branch-level connectivity decreases. However, block-level connectivity suffers from feature redundancy, resulting in constant similarity values even as the number of branches increases. These two similarity analyses demonstrate that our branch-level approach offers richer learned features through the numerous topologies present in the network.

Kernel visualization We visualize the kernel weight of different branches in a block. As shown in Fig. 6a, block-level connectivity in the top row exhibits a consistent trend of all branches of 3×3, 1×1, 1×3, and 3×1 kernels with over smoothed weight. Therefore, it shows the monotonic patterns of re-parameterized weight. In contrast, the weight of the proposed branch-level connectivity NS (middle and bottom rows) offers a variety of patterns. This result further supports that our re-parameterization promotes the generation of richer learned features.



(a) Kernel Map (b) Drop path

Fig. 6: (a) Kernel weight visualization for branches in a block. (b) Performance degradation when a single block is dropped in a network. The red bar is calculated by $Branch_{Acc} - Block_{Acc}$.

Ensemble effect As discussed in a previous study [41], the absence of a single block in a network may not significantly impair performance due to the ensemble effect [18]. In this regard, Fig. 6b illustrates that our NS enhances the benefits of the ensemble effect when a single block is dropped. Therefore, this illustration substantiates our claim that NS improves ensemble benefits by generating unlimited topologies during the re-parameterization process.

7 Conclusion

In this paper, we aim to re-parameterize the networks with the branch-level connectivity, which enriches the sub-network topology in the training step. Unavoidably, the branch-level connectivity suffers from exponentially expanding the computation of the entire network during training and failing to infuse the non-linear activation functions. To overcome this challenge, we propose the stochastic neural substitution and guided activation function. To reduce the computational budget, the proposed method stochastically decides the substitution in each iteration and uses the guided function to transform the non-linear activation function linearly. Our experiments reveal that the proposed method outperforms existing block-level connectivity, demonstrating the usefulness of our strategy. We believe the proposed method will be useful in future research.

Acknowledgments. This paper was supported in part by the Electronics and Telecommunications Research Institute (ETRI) Grant funded by Korean Government (Fundamental Technology Research for Human-Centric Autonomous Intelligent Systems) under Grant 24ZB1200, Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (Artificial Intelligence Innovation Hub) under Grant RS-2021-II212068, under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2024-RS-2023-00255968), and the National Research Foundation of Korea (NRF) from the Korea Government (MSIT) under Grant RS-2024-00356486.

References

1. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: European Conference on Computer Vision (2014)
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2018)
3. Cao, A.Q., Puy, G., Boulch, A., Marlet, R.: Pcam: Product of cross-attention matrices for rigid registration of point clouds. In: IEEE International Conference on Computer Vision (2021)
4. Cao, J., Li, Y., Sun, M., Chen, Y., Lischinski, D., Cohen-Or, D., Chen, B., Tu, C.: Do-conv: Depthwise over-parameterized convolutional layer. IEEE Transactions on Image Processing (2022)
5. Chen, S., Chen, Y., Yan, S., Feng, J.: Efficient differentiable neural architecture search with meta kernels. Arxiv (2019)
6. Chu, X., Li, L., Zhang, B.: Make repvgg greater again: A quantization-aware approach. In: Proceedings of the AAAI Conference on Artificial Intelligence (2024)
7. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
9. Ding, X., Guo, Y., Ding, G., Han, J.: Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In: IEEE International Conference on Computer Vision (2019)
10. Ding, X., Xia, C., Zhang, X., Chu, X., Han, J., Ding, G.: Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. Arxiv (2021)
11. Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit. In: IEEE Conference on Computer Vision and Pattern Recognition (2021)
12. Ding, X., Zhang, X., Han, J., Ding, G.: Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
13. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: IEEE Conference on Computer Vision and Pattern Recognition (2021)
14. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision (2010)
15. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: A new multi-scale backbone architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
16. Guo, S., Alvarez, J.M., Salzmann, M.: Expandnets: Linear over-parameterization to train compact convolutional networks. Neural Information Processing Systems (2020)
17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: IEEE International Conference on Computer Vision (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)

19. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. Arxiv (2017)
20. Huang, T., You, S., Zhang, B., Du, Y., Wang, F., Qian, C., Xu, C.: Dyrep: Bootstrapping training with dynamic re-parameterization. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (2015)
22. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: IEEE International Conference on Computer Vision Workshops (2013)
23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
24. Lee, Y., Kim, J., Willette, J., Hwang, S.J.: Mpvit: Multi-path vision transformer for dense prediction. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
25. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al.: Yolov6: A single-stage object detection framework for industrial applications. Arxiv (2022)
26. Li, Z., Xiao, J., Yang, L., Gu, Q.: Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In: IEEE International Conference on Computer Vision (2023)
27. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision (2014)
28. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. Arxiv (2018)
29. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
30. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. Arxiv (2013)
31. Ouyang, D., He, S., Zhang, G., Luo, M., Guo, H., Zhan, J., Huang, Z.: Efficient multi-scale attention module with cross-spatial learning. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2023)
32. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
33. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning (2021)
34. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Neural Information Processing Systems (2015)
35. Ryu, J., Han, D., Lim, J.: Gramian attention heads are strong yet efficient vision learners. In: IEEE International Conference on Computer Vision (2023)
36. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)

37. Tang, K., Zhao, W., Peng, W., Fang, X., Cui, X., Zhu, P., Tian, Z.: Reparameterization head for efficient multi-input networks. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2024)
38. Thomee, B., Shamma, D.A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., Li, L.J.: Yfcc100m: The new data in multimedia research. Communications of the ACM (2016)
39. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Fastvit: A fast hybrid vision transformer using structural reparameterization. In: IEEE International Conference on Computer Vision (2023)
40. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: Mobileone: An improved one millisecond mobile backbone. In: IEEE Conference on Computer Vision and Pattern Recognition (2023)
41. Veit, A., Wilber, M.J., Belongie, S.: Residual networks behave like ensembles of relatively shallow networks. Neural Information Processing Systems (2016)
42. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: IEEE Conference on Computer Vision and Pattern Recognition (2023)
43. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: European Conference on Computer Vision (2018)
44. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
45. Zhang, M., Yu, X., Rong, J., Ou, L.: Repnas: Searching for efficient re-parameterizing blocks. Arxiv (2021)
46. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
47. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. Arxiv (2016)