Efficient Attention Network: Accelerate Attention by Searching Where to Plug

Zhongzhan Huang[1,*], Senwei Liang[2,*], Mingfu Liang[3,†], Wei He[4,†], Haizhao Yang [2,‡]
[1]Tsinghua University, [2]Purdue University,
[3]Northwestern University, [4]Nanyang Technological University,
hzz_dedekinds@foxmail.com, liang339@purdue.edu,
mingfuliang2020@u.northwestern.edu, wei005@e.ntu.edu.sg
haizhao@purdue.edu

Abstract

Recently, many plug-and-play self-attention modules are proposed to enhance the model generalization by exploiting the internal information of deep convolutional neural networks (CNNs). Previous works lay an emphasis on the design of attention module for specific functionality, e.g., light-weighted or task-oriented attention. However, they ignore the importance of where to plug in the attention module since they connect the modules individually with each block of the entire CNN backbone for granted, leading to incremental computational cost and number of parameters with the growth of network depth. Thus, we propose a framework called Efficient Attention Network (EAN) to improve the efficiency for the existing attention modules. In EAN, we leverage the sharing mechanism [11] to share the attention module within the backbone and search where to connect the shared attention module via reinforcement learning. Finally, we obtain the attention network with sparse connections between the backbone and modules, while (1) maintaining accuracy (2) reducing extra parameter increment and (3) accelerating inference. Extensive experiments on widely-used benchmarks and popular attention networks show the effectiveness of EAN. Furthermore, we empirically illustrate that our EAN has the capacity of transferring to other tasks and capturing the informative features. The code is available at https://github.com/gbup-group/EAN-efficient-attention-network.

1. Introduction

Recently, many plug-and-play and straightforward self-attention modules that utilize the interior information of a network to enhance instance specifity [18] are proposed to boost the generalization of deep convolutional neural networks (CNNs) [9, 27, 15, 11, 4, 26]. The self-attention module is usually plugged into every block of a residual network (ResNet) [7] (see Fig. 2 (a) for the structure of a ResNet and Fig. 2 (b) for a network with attention modules). In general, the implementation of the attention module can be divided into three steps [11]: **(1) Extraction**: the plug-in module extracts internal features of a network by computing their statistics, like mean, variance or higher-order moments [9, 14]; **(2) Processing**: the module leverages the extracted features to adaptively generate a mask that measures the importance of the feature maps via a fully connected layer [9], convolution layer [27], or feature-wise linear transformation [18] etc.; **(3) Recalibration**: the mask is used to calibrate the feature maps of the network by element-wise multiplication or addition [9, 4]. The compu-

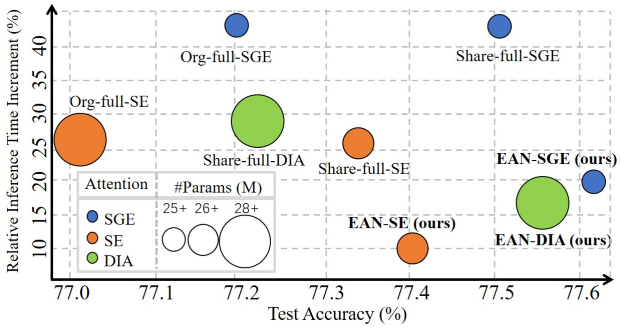

Figure 1: Comparison of relative inference time increment (see Eqn. 9), number of parameters, and test accuracy between various attention models on ImageNet 2012. We use different colors to distinguish the type of attention models, and the larger circle size means the larger number of parameters. Our models (EAN) achieve the smaller relative inference time increment, parameters, and higher test accuracy than the attention models of same type.

[*] Equal contribution
[†] Equal contribution
[‡] Corresponding author

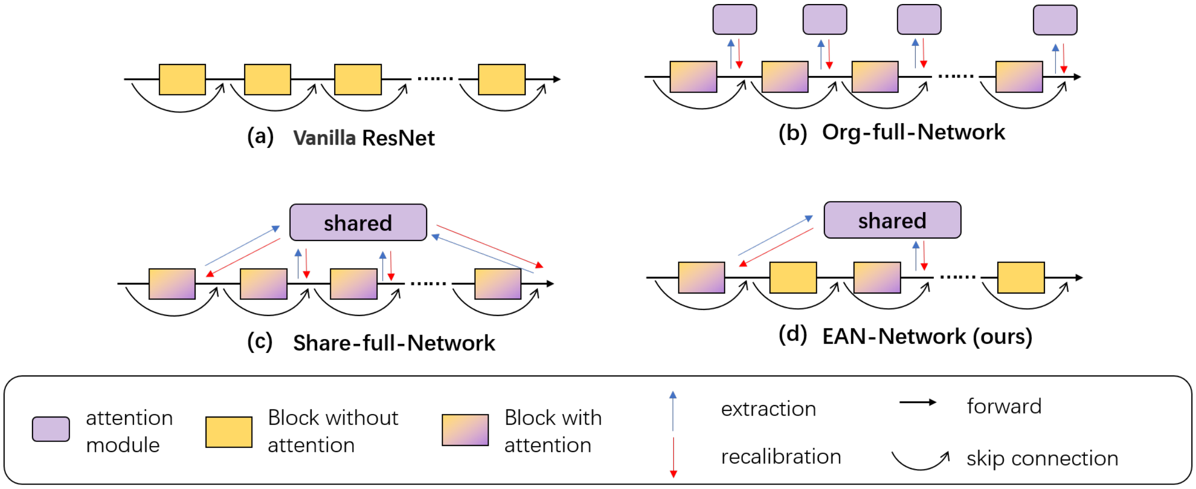

Figure 2: Comparison of network structures between (a) ResNet, (b) Org-full attention network, (c) Share-full attention network, and (d) our EAN network. The detailed introduction of different networks is shown in Section 3.

tation operations and trainable module in the implementation of self-attention inevitably require extra computational complexity and introduce many additional parameters. This limits self-attention modules usage on more practical applications that require fast inference speed and small network size. Therefore, previous works made efforts on reducing the cost of individual self-attention module in terms of its parameters or computational complexity:

Sharing Mechanism. Huang et al. [11] proposed to share an attention module throughout different network blocks in the same stage and to encourage the integration of layer-wise information. As shown in Fig. 2 (c), the sharing mechanism enables different blocks in the same stages to use an attention module with the same set of parameters. Since the parameter increment depends on the number of stages, it significantly reduces the increment of trainable parameters. However, the computational complexity remains the same.

Lightweight Module. Designing the lightweight attention module, *e.g.*, feature-wise linear transformation [15, 18], to process the features can reduce the cost. However, the lightweight module does not reduce the computational cost of “extraction” and “recalibration” steps. For example, in Fig. 1, a lightweight attention design like SGE [15] has significantly smaller parameter increment than SE [9] but it causes a longer inference time. Further, the attention modules are individually plugged into every block throughout a CNN, and hence the cost still increases with the growing number of blocks.

As discussed above, the extra computational cost remains large despite adopting the sharing mechanism or lightweight module. To improve the efficiency of the attention modules in CNNs, in this paper, a simple idea is proposed to reduce the number of interactions between blocks and attention modules instead of plugging the attention modules into each block. Meanwhile, we adopt the sharing mechanism [11], as shown in Fig. 2 (d). Comparing to Fig. 2 (b) and (c), our advantages are two-folded, ① smaller parameter increment ② smaller computational cost increment because of less connections between backbone and attention module. However, to achieve satisfactory performance, the dense connections [10] and the adequate number of trainable parameters [25] in networks are two critical factors generally. Thus, to balance efficiency and satisfactory performance, we propose to use reinforcement learning to search for the optimal connection scheme. Our goal is to obtain the attention network with sparse connections between the backbone and modules while (1) maintaining accuracy (2) reducing extra parameter increment and (3) accelerating inference. Our framework is called the Efficient Attention Network (EAN), which leverages the sharing mechanism to implement the attention module within the backbone and searches where to connect the shared attention module via reinforcement learning.

Our Contribution. We summarize our contribution as follows:

1. We propose an effective connection searching framework to improve the efficiency of the given attention network while maintaining the original accuracy, reducing the extra parameters increment, and accelerating the inference.

2. Through our empirical experiments, we illustrate that the attention network searched by our framework has the capacity of transferring to other tasks and capturing the informative features.

2. Related Works

Neural Architecture Search (NAS). Designing a satisfactory neural architecture automatically without oracles, also known as neural architecture search, is of significant interest for academics and industrial AI research. Such a problem may always be formulated as searching for the optimal combination of different network granularities.

The early NAS works require expensive computational cost for scratch-training a massive number of architecture candidates [30, 31]. To alleviate the searching cost, the recent advances of one-shot approaches for NAS bring up the concept of supernet based on the weight-sharing heuristic. Supernet serves as the search space embodiment of the candidate architectures, and it is trained by optimizing different sub-networks from the sampling paths, *e.g.*, SPOS [6], GreedyNAS [28] and FairNAS [5].

The most conceptually related work [16] aims to propose a lightweight non-local (LightNL) block [26] and searches for the optimal configuration to incorporate the non-local block into mobile neural networks. Although the inserted location of the LightNL is also considered in their NAS objective, the construction of the LightNL blocks is also jointly optimized in their objective. As both the inserted location and the construction of LightNL are integrated completely after the searching, it is hard to differentiate the net contribution of their proposed inserted location of LightNL blocks. However, in our work, we tailor to identify the importance of *where to plug* in the attention module, and we do not concentrate on only one design of the existing attention modules, compared to Li et al. [16] that only specializes on non-local block [26]. To sum up, the difference in the research target and more general consideration differentiate our work with Li et al. [16].

Self-Attention Mechanism. The self-attention mechanism is widely used in CNNs for computer vision [9, 26, 11, 4, 15, 18]. The self-attention module is modularized as a network component and inserted into different layers of the network to emphasize informative features and their importance according to the internal information. Many works focus on the design of the attention module for specific functionality. Squeeze-and-Excitation (SE) [9] module leverages global average pooling to extract the channel-wise statistics and learns the non-mutually-exclusive relationship between channels. Spatial Group-wise Enhance (SGE) [15] module learns to recalibrate features by saliency factors learned from different groups of the feature maps. Dense-Implicit-Attention (DIA) [11] module captures the layer-wise feature interrelation with a recurrent neural network (RNN).

3. Preliminaries

In this section, we briefly review ResNet [7]. Then we formulate two types of attention networks, Org-full network (Fig. 2 (b)), and Share-full network (Fig. 2 (c)). The structure of ResNet is shown in Fig. 2 (a). In general, the ResNet architecture has several stages, and each stage, whose feature maps have the same size, is a collection of consecutive blocks. Suppose a ResNet has m blocks. Let x_ℓ be the input of the ℓ^{th} block and $f_\ell(\cdot)$ be the residual mapping, and then the output $x_{\ell+1}$ of the ℓ^{th} block is defined as

$$x_{\ell+1} = x_\ell + f_\ell(x_\ell). \tag{1}$$

3.1. Org-full Attention Network

We describe an attention network as an Org-full network (Fig. 2 (b)) if the attention module is individually defined for each block. Note that the term “full” refers to a scenario that all blocks in a network connect to the attention modules, while “Org” is short for “Original”. Many popular attention modules adopt this way to connect the ResNet backbone [9, 15, 27]. We denote the attention module in the ℓ^{th} block as $M(\cdot; W_\ell)$, where W_ℓ are the parameters. Then the attention will be formulated as $M(f_\ell(x_\ell); W_\ell)$ which consists of the extraction and processing operations introduced in Section 1. In the recalibration step, the attention is applied to the residual output $f_\ell(x_\ell)$, *i.e.*,

$$x_{\ell+1} = x_\ell + M(f_\ell(x_\ell); W_\ell) \odot f_\ell(x_\ell), \tag{2}$$

where $\ell = 1, \cdots, m$ and $\odot$ is the element-wise multiplication. Eqn. 2 indicates that the computational cost and number of parameters grow with the increasing number of blocks m.

3.2. Share-full Attention Network

We denote an attention network as a Share-full network (Fig. 2 (c)) if the blocks within one stage are connected to the same attention module, which is defined for the stage. The idea of Share-full network is first proposed in Huang et al. [11]. We denote attention module defined in the stage k as $M(\cdot; W_k)$. If the ℓ^{th} block belongs to the k_ℓ stage, then the attention is modeled as $M(f_\ell(x_\ell); W_{k_\ell})$. The building block becomes

$$x_{\ell+1} = x_\ell + M(f_\ell(x_\ell); W_{k_\ell}) \odot f_\ell(x_\ell), \tag{3}$$

where $\ell = 1, \cdots, m$. Distinct from the Org-full attention network, the number of extra parameters of the Share-full network depends on the number of stages, instead of the number of blocks m. Typically, a ResNet has 3∼4 stages but has tens of blocks, which indicates a Share-full network can significantly reduce the extra parameters.

Algorithm 1 Searching optimal connection scheme

```
Input: Training set $D_{\text{train}}$; validation set $D_{\text{val}}$; a Share-full network $\Omega(\mathbf{x}|\mathbf{1})$; learning rate $\eta$; pre-training step $K$; searching step $T$; time step $h$ to apply PPO.
Output: The trained controller $\chi_\theta(x_0)$.
                                          ▷ Pre-train the supernet
for $t$ from 1 to $K$ do
    $\mathbf{a} \sim [Bernoulli(0.5)]^m$
    train $\Omega(\mathbf{x}|\mathbf{a})$ with $D_{\text{train}}$
end for
                                          ▷ Policy-gradient-based search
for $t$ from 1 to $T$ do
    $\mathbf{p}_\theta \leftarrow \chi_\theta(x_0)$
    $\mathbf{a} \sim \mathbf{p}_\theta$
    $g_{\text{spa}} \leftarrow$ Eqn. 6
    $g_{\text{val}} \leftarrow \Omega(D_{\text{val}}|\mathbf{a})$, $g_{\text{rnd}} \leftarrow \|\sigma_1(\mathbf{a}) - \sigma_2(\mathbf{a};\phi)\|_2^2$
    calculate the reward $G(\mathbf{a})$ by Eqn. 7
    update $\theta$ by Eqn. 5
    update $\phi$ by minimizing $\|\sigma_1(\mathbf{a}) - \sigma_2(\mathbf{a};\phi)\|_2^2$
    put $(\mathbf{p}_\theta, \mathbf{a}, G(\mathbf{a}))$ into replay buffer
                                          ▷ Update $\theta$ from buffer
    if t $\geq$ h then
        sample $(\mathbf{p}_\theta, \mathbf{a}, G(\mathbf{a}))$ from replay buffer
        update $\theta$ by Eqn. 8
    end if
end for
return $\chi_\theta(x_0)$
```

4. Proposed Method

In this section, we systematically introduce the proposed Efficient Attention Network (EAN) framework, which consists of two parts: First, we pre-train a supernet as the search space, which has the same network structure as a Share-full network. Second, we use a policy-gradient-based method to search for an optimal connection scheme from the supernet. The basic workflow of our method is shown in Alg. 1.

4.1. Problem Description

We consider a supernet $\Omega(\mathbf{x}|\mathbf{a})$ with m blocks and input $\mathbf{x}$, which has the same network structure as a Share-full network. A sequence $\mathbf{a} = (a_1, a_2, \cdots, a_m)$ denotes an attention connection scheme, where $a_i = 1$ when the i^{th} block is connected to the shared attention module, otherwise it is equal to 0. A sub-network specified by a scheme $\mathbf{a}$ can be formulated as follows:

$$x_{\ell+1} = x_\ell + \Big(a_\ell \cdot M(f_\ell(x_\ell); W_{k_\ell}) + (1 - a_\ell)\mathbf{1}\Big) \odot f_\ell(x_\ell), \tag{4}$$

where $\mathbf{1}$ denotes an all-one vector and ℓ is from 1 to m. In particular, $\Omega(\mathbf{x}|\mathbf{a})$ becomes a Share-full network if $\mathbf{a}$ is all-one vector, or a vanilla ResNet while $\mathbf{a}$ is a zero vector.

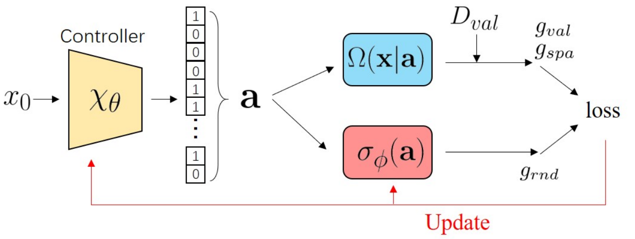

Figure 3: The illustration of our policy-gradient-based method to search an optimal scheme.

Our goals are: (1) to find a connection scheme $\mathbf{a}$, which is sparse enough for less computation cost, from 2^m possibilities; (2) to ensure that the network $\Omega(\mathbf{x}|\mathbf{a})$ possesses good generalization.

4.2. Pre-training the Supernet

To determine the optimal architecture from the pool of candidates, it is costly to evaluate all their individual performance after training. In many related works on NAS, candidates' validation accuracy from a supernet serve as a satisfactory performance proxy [6, 28, 5]. Similarly, to obtain the optimal connection scheme for the attention module, we propose to train the supernet as the search space following the idea of co-adaption [19]. We consider the validation performance of the sampled sub-networks as the proxy for their stand-alone performance[1].

Specifically, given a dataset, we split all training samples into the training set D_{train} and the validation set D_{val}. To train the supernet, we activate or deactivate the attention module in each block of it randomly during optimization. We first initialize a supernet $\Omega(\mathbf{x}|\mathbf{a}^{(0)})$, where $\mathbf{a}^{(0)} = (1, \cdots, 1)$. At iteration t, we randomly draw a connection scheme $\mathbf{a}^{(t)} = (a_1^t, \cdots, a_m^t)$, where a_i^t is sampled from a Bernoulli distribution $B(0.5)$. Then, we train sub-network $\Omega(\mathbf{x}|\mathbf{a}^{(t)})$ with the scheme $\mathbf{a}^{(t)}$ from $\Omega(\mathbf{x}|\mathbf{a}^{(0)})$ on D_{train} via weight-sharing.

4.3. Training Controller with Policy Gradient

In this part, we introduce the step to search the optimal scheme, which uses a controller to generate connection schemes and updates the controller by policy gradient, as illustrated in Fig. 3.

We use a fully connected network as controller $\chi_\theta(x_0)$ to produce the connection schemes, where θ are the learnable parameters, and x_0 is a constant vector $\mathbf{0}$. The output of $\chi_\theta(x_0)$ is $\mathbf{p}_\theta$, where $\mathbf{p}_\theta = (p_\theta^1, p_\theta^2, ..., p_\theta^m)$ and p_θ^i represents the probability of connecting the attention to the i^{th} block. A realization of $\mathbf{a}$ is sampled from the controller output, *i.e.*, $\mathbf{a} \sim \mathbf{p}_\theta$. The probability associated with the scheme $\mathbf{a}$ is $\hat{\mathbf{p}}_\theta = (\hat{p}_\theta^1, \hat{p}_\theta^2, ..., \hat{p}_\theta^m)$, where $\hat{p}_\theta^i = (1 - a_i)(1 - p_\theta^i) + a_i p_\theta^i$.

[1] Train the sub-networks from scratch

Dataset	Model	Test Accuracy (%)			Parameters (M)			Relative Inference Time Increment (%)		
		Org-full	Share-full	EAN	Org-full	Share-full	EAN	Org-full	Share-full	EAN
CIFAR100	Org	74.29	-	-	1.727	-	-	0.00	-	-
	SE [9]	75.80	76.09	**76.93**	1.929	1.739	**1.739**	54.46	52.09	**23.52**
	SGE [15]	75.75	76.17	**76.36**	1.728	1.727	**1.727**	93.60	93.41	**50.49**
	DIA [11]	-	**77.26**	77.12	-	1.946	**1.946**	-	121.11	**65.46**
ImageNet	Org	76.01	-	-	25.584	-	-	0.00	-	-
	SE [9]	77.01	77.35	**77.40**	28.115	26.284	**26.284**	25.94	25.92	**10.35**
	SGE [15]	77.20	77.51	**77.62**	25.586	25.584	**25.584**	40.60	40.50	**19.66**
	DIA [11]	-	77.24	**77.56**	-	28.385	**28.385**	-	27.26	**16.58**

Table 1: Comparison of relative inference time increment (see Eqn. 9), number of parameters, and test accuracy between various attention models on CIFAR100 and ImageNet 2012. "Org" stands for ResNet164 backbone in CIFAR100 and ResNet50 backbone in ImageNet. EAN networks have faster inference speed among the networks with the same type of attention module and reduce over 39% growth rate of inference time compared with the same type Share-full attention network.

We denote $G(\mathbf{a})$ as a reward for $\mathbf{a}$. The parameter set θ can be updated via policy gradient with learning rate η, *i.e.*,

$$\begin{aligned} R_\theta &= G(\mathbf{a}) \cdot \sum_{i=1}^{m} \log \hat{p}_\theta^i, \\ \theta &= \theta + \eta \cdot \nabla R_\theta. \end{aligned} \tag{5}$$

In this way, the controller tends to output the probability that results in a large reward G. Therefore, designing a reasonable G can help us search for a good structure.

Sparsity Reward. One of our goals is to accelerate the inference of the attention network. To achieve, we complement a sparsity reward g_{spa} to encourage the controller to generate the schemes with fewer connections between attention modules and backbone. We define g_{spa} by

$$g_{\text{spa}} = 1 - \frac{\|\mathbf{a}\|_0}{m}, \tag{6}$$

where $\|\cdot\|_0$ is a zero norm that counts the number of nonzero entities, and m is the number of blocks.

Validation Reward. The other goal is to find the schemes with which the networks can maintain the original accuracy. Hence, we use the validation accuracy of the sub-network $\Omega(\mathbf{x}|\mathbf{a})$ sampled from the supernet as a reward, which depicts the performance of its structure. The accuracy of $\Omega(\mathbf{x}|\mathbf{a})$ on D_{val} is denoted as g_{val}. In fact, it is popular to use validation accuracy of a candidate network as a reward signal in NAS [21, 30, 6, 31, 28]. Furthermore, it has been empirically proven that the validation performance of the sub-networks sampled from a supernet can be positively correlated to their stand-alone performance [1].

Curiosity Bonus. To encourage the controller to explore more potentially useful connection schemes, we add the Random Network Distillation (RND) curiosity bonus [2] in our reward. Two extra networks with input $\mathbf{a}$ are involved in the RND process, including a target network $\sigma_1(\cdot)$ and a predictor network $\sigma_2(\cdot;\phi)$, where ϕ is the parameter set. The parameters of $\sigma_1(\cdot)$ are randomly initialized and fixed after initialization, while $\sigma_2(\cdot;\phi)$ is trained with the connection schemes collected by the controller.

The basic idea of RND is to minimize the difference between the outputs of these two networks, which is denoted by term $\sigma_\phi(\cdot) = \|\sigma_1(\cdot) - \sigma_2(\cdot;\phi)\|_2^2$, over the seen connection schemes. If the controller generates a new scheme $\mathbf{a}$, $\sigma_\phi(\mathbf{a})$ is expected to be larger because the predictor $\sigma_2(\cdot;\phi)$ never trains on scheme $\mathbf{a}$. Then, we denote the term $\|\sigma_1(\mathbf{a}) - \sigma_2(\mathbf{a};\phi)\|_2^2$ as g_{rnd}, which is used as curiosity bonus to reward the controller for exploring a new scheme. Besides, in Fig. 5, we empirically show that RND bonus mitigates the fast convergence of early training iterations, leading to exploration for more schemes.

To sum up, our reward $G(\mathbf{a})$ becomes

$$G(\mathbf{a}) = \lambda_1 \cdot g_{\text{spa}} + \lambda_2 \cdot g_{\text{val}} + \lambda_3 \cdot g_{\text{rnd}}, \tag{7}$$

where $\lambda_1, \lambda_2, \lambda_3$ are the coefficient for each bonus.

Data Reuse. To improve the utilization efficiency of sampled connection schemes and speed up the training of the controller, we incorporate Proximal Policy Optimization (PPO) [23] in our framework. As shown in Alg. 1, after the update of parameter θ and ϕ, we put the tuple $(\mathbf{p}_\theta,\ \mathbf{a},\ G(\mathbf{a}))$ into a buffer. At the later step, we retrieve some used connection scheme and update θ as follows:

$$\begin{aligned} \kappa &= \mathbb{E}_{\mathbf{a}\sim\mathbf{p}_{\theta_{old}}} \left[G(\mathbf{a}) \sum_{i=1}^{m} \frac{\hat{p}_\theta^i}{\hat{p}_{\theta_{old}}^i} \nabla_\theta \log \hat{p}_\theta^i \right], \\ \theta &= \theta + \eta \cdot \kappa, \end{aligned} \tag{8}$$

where η is learning rate and the θ_{old} denotes the θ sampled from buffer.

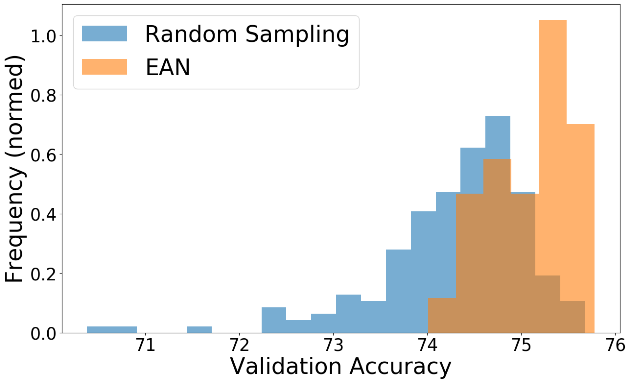

Figure 4: Comparison of the validation accuracy distribution between EAN and Random sampling for SE module. The validation accuracy is obtained by training from scratch the model specified by the connection scheme from these methods on CIFAR100 with ResNet164 backbone.

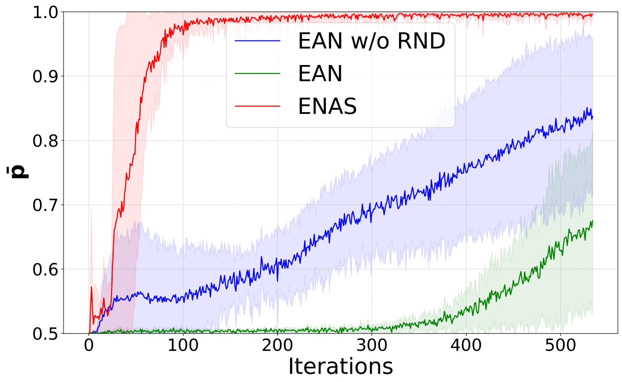

Figure 5: Comparison of the convergence speed between ENAS and EAN. The controller tends to generate a deterministic scheme when $\bar{p}$ is close to 1.

5. Experiments

5.1. Datasets and Settings

On CIFAR100 [13] and ImageNet 2012 [22] datasets, we conduct experiments on ResNet [7] backbone with different attention modules, including SE (Squeeze-Excitation) [9], SGE (Spatial Group-wise Enhance) [15] and DIA (Dense-Implicit-Attention) [11] modules. In our Supplementary, we describe these modules as well as the training settings of controller and networks. Since the networks with attention modules have extra more computational cost from the vanilla backbone inevitably, we formulate the inference time increment to represent the relative speed of different attention networks, *i.e.*,

$$\frac{I_t(w.\ Attention) - I_t(wo.\ Attention)}{I_t(wo.\ Attention)} \times 100\%, \qquad (9)$$

where $I_t(\cdot)$ denotes the inference time of the network and the notation $w/wo.\ Attention$ represents the network with the attention module or the vanilla backbone network. All results are measured by forwarding the data of batch size 50 for 1000 times on the server with Intel(R) Xeon(R) Gold 5122 CPU @ 3.60GHz and 1 Tesla V100 GPU.

CIFAR100. CIFAR100 consists of 50k training images and 10k test images of size 32 by 32. In our implementation, we choose 10k images from the training images as a validation set (100 images for each class, 100 classes in total), and the remainder images as a sub-training set. Regarding the experimental settings of ResNet164 [7] backbone with different attention modules, the supernet is trained for 150 epochs, and the search step T is set to be 1000.

ImageNet 2012. ImageNet 2012 comprises 1.28 million training images, which we split 100k images (100 from each class and 1000 classes in total) as the validation set and the remainder as the sub-training set. The testing set includes 50k images. Besides, the random cropping of size 224 by 224 is used in ImageNet experiments. Regarding the experimental settings of ResNet50 [7] backbone with different attention modules, the supernet is trained for 40 epochs, and the search step T is set to be 300.

5.2. Results

The concrete connection schemes found by EAN are presented in our Supplementary. Table 1 shows the test accuracy, the number of parameters, and relative inference time increment on CIFAR100 and ImageNet 2012. Fig. 1 visualizes the ImageNet results from Table 1. Since EAN and Share-full network use sharing mechanism [11] for the attention module, over the vanilla ResNet, they both have fewer parameters increment than the Org-full network. Note that EAN networks have faster inference speed among the networks with the same type of attention and reduce over 39% growth rate of inference time compared with the same type Share-full attention network. Furthermore, Share-full networks have higher accuracy than Org-full networks, but in most cases, the accuracy of EAN networks surpass that of Share-full networks. It implies that disconnecting the interaction between the attention and backbone in the appropriate location can maintain or even improve the performance of attention models.

6. Analysis

6.1. Comparison with Random Sampling

In this part, we illustrate that our EAN framework can find efficient attention structures with good performance. For the SE module with CIFAR100, we compare the 180 different connection schemes obtained by random sampling

Method	Stage1	Stage2	Stage3	Test Accuracy (%)
ENAS (a)	001001001001001001	001001001001001001	001001001001001001	75.80
ENAS (b)	100100101100100100	101101100100101101	100100101101100100	75.11
ENAS (c)	110110110110110110	110110110110110110	110110110110110110	76.08
EAN (a)	001100100101110101	001100000111001111	101100000111110001	**76.93**
EAN (b)	001100000001010111	011100001000010111	101000100110000000	**76.71**

Table 2: The connection schemes searched by ENAS [21] or EAN. The experiment is conducted on CIFAR100 with SE module and ResNet164 backbone.

Dataset	Model	MAE/MSE			Relative Inference Time Increment (%)		
		Org-full	Share-full	EAN	Org-full	Share-full	EAN
SHHB	SE [9]	9.5/15.93	8.9/14.6	8.6/14.7	19.19	19.19	**6.16**
	DIA [11]	-	9.1/14.9	8.2/13.9	-	16.93	**8.71**
SHHA	SGE [15]	93.9/144.5	91.6/143.1	88.4/140.0	58.98	58.85	**30.55**
	SE [9]	89.9/140.2	89.9/140.2	79.4/127.7	49.50	49.00	**21.07**
	DIA [11]	-	92.5/130.4	90.3/141.6	-	51.75	**29.43**

Table 3: Transfer the optimal architecture searched by EAN from image classification to crowd counting task.

and 40 connection schemes found by EAN. We train networks specified by these schemes from scratch on D_{train} and evaluate their performances on D_{val}. Fig. 4 shows the validation accuracy distribution between EAN and random sampling. The validation accuracy associated with EAN (average: 75.10) is greater than random sampling (average: 74.29) with P-value $4 \times 10^{-7} < 0.05$ under t-test. Besides, standard derivation of EAN (std: 0.45) is smaller than random sampling (std: 0.81), suggesting that the EAN framework can find good structures stably.

6.2. Comparison with ENAS

In this part, we compare our EAN with ENAS [21], a prevailing method in NAS, which trains the target network and RNN controller alternatively as well. However, it is infeasible to solve our problem of searching for an optimal scheme by implementing ENAS directly. We compare EAN and ENAS on CIFAR100 with Share-full-SE and ResNet164 backbone.

From our empirical results, the controller of ENAS tends to converge to some periodic-alike schemes at a fast speed. In this case, it will conduct much less exploration of the potential efficient structures. The majority of the schemes searched by ENAS are "111...111" (Share-full network) or "000...000" (Vanilla network), which shows that it can not get the balance between the performance and inference time. The list of schemes searched by ENAS is presented in Supplementary. In Table 2, the minority of the periodic-alike schemes searched by ENAS are shown, e.g., "001" in ENAS (a). Such schemes may come from the input mode of ENAS, *i.e.*, for a connection scheme $\mathbf{a} = (a_1, a_2, ..., a_m)$, the value of component a_l depends on $a_{l-1}, a_{l-2}, ..., a_1$. Such strong sequential correlations let the sequential information dominate in the RNN controller instead of the policy rewards. Compared with the periodic-alike connection schemes from ENAS, the schemes from EAN demonstrate better performance.

Besides, our experiment indicates that ENAS explores a much smaller number of candidate schemes. We quantify the convergence of the controller using $\bar{\mathbf{p}} = \frac{1}{m}\sum_{i=1}^{m} \hat{p}_\theta^i$, which is the mean of the probability $\hat{\mathbf{p}}$ associated with the scheme. When $\bar{\mathbf{p}}$ is close to 1, the controller tends to generate a deterministic scheme. Fig. 5 shows the curve of $\bar{\mathbf{p}}$ with the growth of searching iterations, where $\bar{\mathbf{p}}$ of ENAS shows the significant tendency for convergence in 20 iterations and converges very fast within 100 iterations. Generally speaking, methods in NAS [21, 30] require hundreds or thousands of iterations for convergence.

6.3. Transferring Connection Schemes

To further investigate the generalization of EAN, we conduct experiments on transferring the optimal architecture from image classification to crowd counting task [29, 3, 17, 8]. Crowd counting aims to estimate the density map and predict the total number of people for a given image, whose efficiency is also crucial for many real-world applications, *e.g.*, video surveillance and crowd analysis. However, most state-of-the-art works still rely on the heavy pretrained backbone networks [20] for obtaining satisfactory performance on such dense regression problems. The experiments show that our EAN serves as an efficient backbone network and can extract the representative features for crowd counting.

The networks pre-trained on the ImageNet dataset serve as the backbone of crowd counting models. We evaluated the transferring performance on the commonly-used Shanghai Tech dataset [29], which includes two parts. Shanghai Tech part A (SHHA) has 482 images with 241,677 people counting, and Shanghai Tech part B (SHHB) contains 716 images with 88,488 people counting. Following the previous works, SHHA and SHHB are split into train/validation/test set with 270/30/182 and 360/40/316 images, respectively. The performance on the test set is reported using the standard Mean Square Error (MSE) and Mean Absolute Error (MAE), as shown in Table 3. Our EAN can outperform the baseline (Org-full and Share-full) while reducing the inference time increment by over 40% compared with the baseline.

6.4. Capturing Discriminative Features

To study the ability of EAN in capturing and exploiting features of a given target, we apply Grad-CAM [24] to compare the regions where different models localize on with respect to their target prediction. Grad-CAM is a technique to generate the heatmap highlighting network attention by the gradient related to the given target. Fig. 6 shows the visualization results and the softmax scores for the target with vanilla ResNet50, Share-full-SE, and EAN-SE on the validation set of ImageNet 2012. The red region indicates an essential place for a network to obtain a target score while the blue region is the opposite. The results show that EAN-SE can extract similar features as Share-full-SE, and in some cases, EAN can even capture much more details of the target associating with higher confidence for its prediction. This implies that the searched attention connection scheme may have a more vital ability to emphasize the more discriminative features for each class than the two baselines (Vanilla ResNet and Share-full-SE). Therefore it is reasonable to bring additional improvement on the final classification performance with EAN in that the discrimination is crucial for the classification task, which is also validated from ImageNet test results in Table 1.

6.5. Removing Batch Normalization Layer

Batch Normalization (BN) [12] is widely used to stabilize the training by normalizing each layerwise input in the deep network. Even after removing all BNs, the DIA attention module can stabilize the training and achieve a good generalization due to the dense connections [11].

In this part, we argue that the EAN-DIA can also inherit that property to some extent. Table 4, we compare the performance of Share-full DIA, EAN-DIA, and vanilla ResNet in different settings of BNs removal. Since the connection scheme searched by our EAN possesses certain sparsity, it has sacrificed the capability of stabilizing the optimization to some degree. However, the results show that a part of such capability is still reserved in our EAN-DIA, *e.g.*, in config1 and config2.

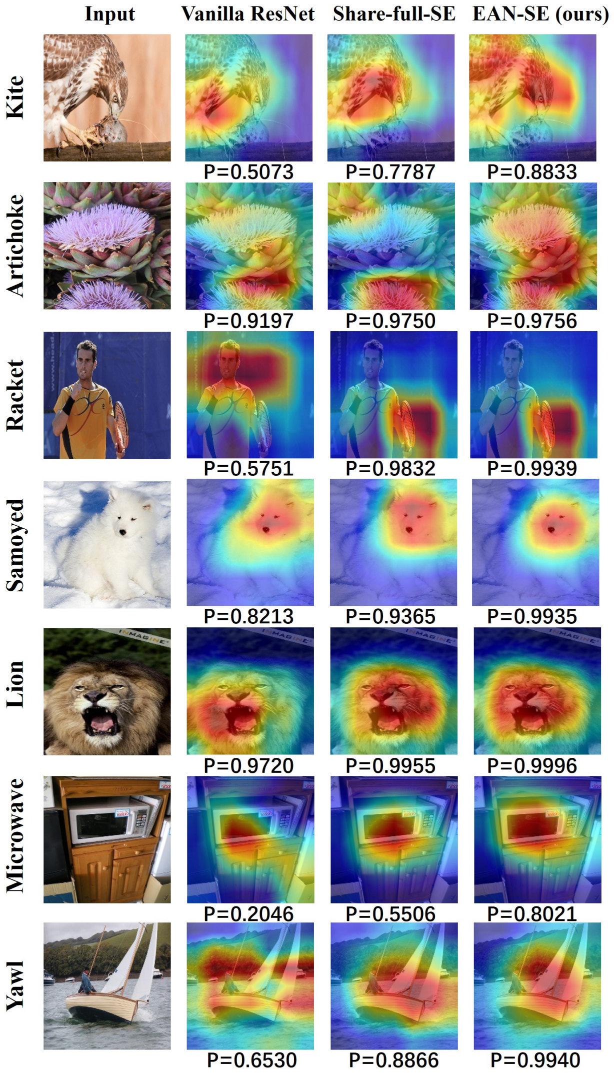

Figure 6: Grad-CAM visualization of different attention models. The red region indicates an essential place for a network to obtain a target score ($\mathbf{P}$) while the blue region is the opposite.

7. Conclusion

To improve the efficiency of using the attention module in a network, we propose an effective EAN framework to search for an optimal connection scheme to plug the modules. Our numerical results show that the attention network searched by our framework can preserve the original accuracy while reducing the extra parameters and accelerating the inference. We empirically illustrate that our attention

Model	Config1	Config2	Config3
Vanilla ResNet	73.95$_{(\pm 0.52)}$	71.73$_{(\pm 0.82)}$	nan
Share-full-DIA	76.95$_{(\pm 0.13)}$	76.59$_{(\pm 0.14)}$	73.80$_{(\pm 0.45)}$
EAN-DIA	76.24$_{(\pm 0.59)}$	76.02$_{(\pm 0.60)}$	nan

Table 4: The test accuracy of different attention network after removing the BN layer with different configurations in each Bottleneck block. The experiments are conducted on CIFAR100 with ResNet164 backbone. "nan" indicates the numerical explosion. (Config1: remove the first BN layer; Config2: remove the first two BN layers; Config3: remove all the BN layers.)

networks have the capacity of transferring to other tasks and capturing the informative features.

References

[1] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 550–559, 2018. 5

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. 5

[3] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. Scale aggregation network for accurate and efficient crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 7

[4] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 1, 3

[5] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 3, 4

[6] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 3, 4, 5

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 6

[8] Mohammad Hossain, Mehrdad Hosseinzadeh, Omit Chanda, and Yang Wang. Crowd counting using scale-aware attention networks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1280–1288. IEEE, 2019. 7

[9] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1, 2, 3, 5, 6, 7, 10

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2

[11] Zhongzhan Huang, Senwei Liang, Mingfu Liang, and Haizhao Yang. Dianet: Dense-and-implicit attention network. In *AAAI*, pages 4206–4214, 2020. 1, 2, 3, 5, 6, 7, 8, 10

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 8

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 6

[14] HyunJae Lee, Hyo-Eun Kim, and Hyeonseob Nam. Srm: A style-based recalibration module for convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1854–1862, 2019. 1

[15] Xiang Li, Xiaolin Hu, and Jian Yang. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*, 2019. 1, 2, 3, 5, 6, 7, 10

[16] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10297–10306, 2020. 3

[17] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018. 7

[18] Senwei Liang, Zhongzhan Huang, Mingfu Liang, and Haizhao Yang. Instance enhancement batch normalization: An adaptive regulator of batch noise. In *AAAI*, pages 4819–4827, 2020. 1, 2, 3

[19] Senwei Liang, Yuehaw Khoo, and Haizhao Yang. Drop-activation: Implicit parameter reduction and harmonic regularization. *arXiv preprint arXiv:1811.05850*, 2018. 4

[20] Lingbo Liu, Jiaqi Chen, Hefeng Wu, Tianshui Chen, Guanbin Li, and Liang Lin. Efficient crowd counting via structured knowledge transfer. In *ACM International Conference on Multimedia*, 2020. 7

[21] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104, 2018. 5, 7

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

[24] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 8

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[26] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 1, 3

[27] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 1, 3

[28] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1999–2008, 2020. 3, 4, 5

[29] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016. 7, 8

[30] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3, 5, 7

[31] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 3, 5

8. Supplementary

8.1. Review of Attention Modules

In this part, we review the attention modules used in our paper, *i.e.*, SE [9], SGE [15] and DIA [11]. We follow some notations of Section 3 in the main text. Let x_ℓ be the input of the ℓ^{th} block, $f_\ell(\cdot)$ be the residual mapping, and $M(\cdot; W_\ell)$ be the attention module in the ℓ^{th} block with the parameters W_ℓ. The attention is formulated as $M(f_\ell(x_\ell); W_\ell)$. We denote $f_\ell(x_\ell)$ as $X^{(\ell)}$ of size $C \times H \times W$, where C, H and W denote channel, height and width respectively. For simplicity, we denote $X^\ell_{chw} = X^\ell[c, h, w]$ as the value of pixel (h, w) at the channel c and $X^\ell_c = X^\ell[c, :, :]$ as the tensor at the channel c.

SE Module. SE module utilizes average pooling to extract the features and processes the extracted features by a one-hidden-layer fully connected network.

First, the SE module squeezes the information of channels by the average pooling,

$$m^\ell_c = \text{AVG}(X^\ell_c) = \frac{1}{H \cdot W} \sum_{h=1}^{H} \sum_{w=1}^{W} X^\ell_{chw}, \tag{10}$$

where $c = 1, \cdots, C$. Then, an one-hidden-layer fully connected layer $\text{FC}(\cdot; W_\ell)$ with ReLU activation is used to fuse the information of all the channels and here W_ℓ is the parameter. The hidden layer node size is $C//r$, where "$//$" is exact division and "r" denotes reduction rate. The reduction rate is 16 in our experiments. Finally, a Sigmoid function(i.e., $\text{sig}(z) = 1/(1 + e^{-z})$) is applied to the processed features and we get the attention as follows,

$$[\delta_1; \cdots; \delta_C] = \text{sig}(\text{FC}([m^\ell_1; \cdots; m^\ell_C]; W_\ell)). \tag{11}$$

DIA Module. DIA module integrates the block-wise information by an LSTM (Long Short-Term Memory). Let m^ℓ_c be the output of average pooling as Eqn. 10. Then m^ℓ_c is passed to LSTM along with a hidden state vector $h_{\ell-1}$ and a cell state vector $c_{\ell-1}$, where h_0 and c_0 are initialized as zero vectors. The LSTM generates h_ℓ and c_ℓ at the ℓ^{th} block, *i.e.,*

$$(h_\ell, c_\ell) = \text{LSTM}([m^\ell_1; \cdots; m^\ell_C], h_{\ell-1}, c_{\ell-1}; W), \tag{12}$$

where W is the trainable parameter of the LSTM. The hidden state vector h_t is used as attention to recalibrate feature maps. The reduction ratio within LSTM introduced in [11] is 4 for CIFAR100 or 20 for ImageNet 2012.

SGE Module. SGE divides the feature maps into different groups and then utilizes the global information from the group to recalibrate its features. Let G be the number of groups and then each group has $C//G$ feature maps. Denote Y^ℓ of size $(C//G) \times H \times W$ as a group of feature maps within X^ℓ. The extracted feature for the group Y^ℓ is

$$g_c^\ell = \text{AVG}(Y_c^\ell) = \frac{1}{H \cdot W} \sum_{h=1}^{H} \sum_{w=1}^{W} Y_{chw}^\ell. \tag{13}$$

Let g be $[g_1^\ell; \cdots; g_{C//G}^\ell]$. The importance coefficient for each pixel (h, w) is defined as

$$p_{hw} = g \cdot Y[:, h, w], \tag{14}$$

where $\cdot$ is dot product. Then p_{hw} is normalized by

$$\hat{p}_{hw} = \frac{p_{hw} - \mu}{\sigma + \epsilon}, \tag{15}$$

where the mean μ and variance σ^2 are defined by

$$\mu = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} p_{hw}, \sigma^2 = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} (p_{hw} - \mu)^2. \tag{16}$$

An additional pair of parameters (γ, β) are introduced for the group Y^ℓ to rescale and shift the normalized features, and SGE modules get the attention for $Y[:, h, w]$ as follows,

$$\text{sig}(\gamma \hat{p}_{hw} + \beta). \tag{17}$$

The G is set to be 4 for CIFAR100 or 64 for ImageNet 2012 experiments.

8.2. Connection Scheme Searched by EAN

We show the connection schemes searched by EAN in Table 5 and Table 6.

8.3. List of Schemes Searched by ENAS

We show the list of connection schemes by ENAS (an example) in Table 7. From our empirical results, the controller of ENAS tends to converge to some periodic-alike schemes. In this example, the majority of the schemes searched by ENAS are “111...111” (Share-full network).

8.4. Training Setting for Controller

CIFAR100. We optimize the controller for 1000 iterations with momentum SGD. The learning rate is set to be $5{\times}10^{-2}$. The time step h to apply PPO is 10.

ImageNet 2012. We optimize the controller for 300 iterations with momentum SGD. The learning rate is set to be $5{\times}10^{-2}$. The time step h to apply PPO is 10.

8.5. Training Setting for Stand-alone Performance

In this part, we introduce the parameter setting for the model trained from scratch. In our experiments, we use cross-entropy loss and optimize the model by SGD with momentum 0.9 and initial learning rate 0.1. The weight decay is set to be 10^{-4}.

CIFAR100. On CIFAR, we use ResNet164 backbone. The model is trained for 164 epochs with the learning rate dropped by 0.1 at 81, 122 epochs.

ImageNet 2012. On ImageNet 2012, we use the ResNet50 backbone. The model is trained for 120 epochs with the learning rate dropped by 0.1 at every 30 epochs.

Dataset	Model	Stage1	Stage2	Stage3	Stage4	Test Accuracy (%)
ImageNet 2012	SE	010	0010	110011	001	77.40
	SGE	100	1011	010011	011	77.62
	DIA	110	0011	110010	011	77.56

Table 5: The connection scheme searched by EAN with ResNet50 backbone and different attention modules on ImageNet 2012. ResNet50 has 4 stages and each stage has 3, 4, 6 and 3 blocks respectively.

Dataset	Model	Stage1	Stage2	Stage3	Test Accuracy (%)
CIFAR100	SE	001100100101110101	001100000111001111	101100000111110001	76.93
	SGE	010101101111011010	101110101011000000	101101101110100010	76.36
	DIA	111000101111000110	100000010010010110	000111111000000001	77.12

Table 6: The connection scheme searched by EAN with ResNet164 backbone and different attention modules on CIFAR100. ResNet164 has 3 stages and each stage has 18 blocks respectively.

Iteration	**Connection Scheme**	**Sparse**	$\bar{\mathbf{p}}$
0	000110000001111101110010000001000110111110110110010011	0.52	0.50
5	100100111101010011110001110101011111011100110001000011	0.44	0.51
10	100001001011110110001000110011011101110111110111000011	0.44	0.50
15	111001000110011110111000111001011011111011011110111001	0.37	0.57
20	111110011111111110001110011110010111011001111111110111	0.26	0.67
25	101111111111111100111111110000010001101111111110011111	0.26	0.64
30	011110011111111111111110001111111111110011111111101111	0.17	0.85
35	111111110001111111111011111111111111111111111111111111	0.07	0.91
40	101111111111111111111111111111111111111111111111111111	0.02	0.96
45	111111111111111101111111111111111111111111111111111111	0.02	0.98
50	011111111111111111111000111111111111111111111111111111	0.07	0.98
55	111111111100111111111111111111111111111111111111111111	0.04	0.98
60	111111111111111111111111111111111111111111111111111111	0.00	0.98
65	111111111111111111111111111111111111111111111111111111	0.00	0.99
70	111111111111111100111111111111111111111111111111111111	0.04	0.96
75	011111111111111111111111111111111111111111111111111111	0.02	0.99
80	111111111111111111111111111111111111111111111111111111	0.00	1.00
85	111111111111111111111111111111111111111111111111111111	0.00	1.00
90	111111111111111111111111111111111111111111111111111111	0.00	1.00
95	111111111111111111111111111111111111111111111111111111	0.00	0.98
100	111111111111111111111111111111111111111111111111111111	0.00	0.99
105	111111111111111111111111111111111111111111111111111111	0.00	1.00
110	111111111111111111111111111111111111111111111111111111	0.00	1.00
115	111111111111111111111111111111111111111111111111111111	0.00	1.00
120	111111111111111111111111111111111111111111111111111111	0.00	1.00
125	111111111111111111111111111111111111111111111111111111	0.00	1.00
130	111111111111111111111111111111111111111111111111111111	0.00	1.00
135	111111111111111111111111111111111111111111111111111111	0.00	0.99

Table 7: The connection scheme searched by ENAS. $\bar{\mathbf{p}}$ is the average of the probability associated with the scheme. The controller tends to generate a deterministic scheme when $\bar{\mathbf{p}}$ is close to 1. The experiment is conducted on CIFAR100 with ResNet164 and SE modules.