ConAM: Confidence Attention Module for Convolutional Neural Networks

Yu Xue*
School of Computer and Software
Nanjing University of Information Science and Technology
China
xueyu@nuist.edu.cn

Ziming Yuan
School of Computer and Software
Nanjing University of Information Science and Technology
China
YZM@nuist.edu.cn

Ferrante Neri
School of Computer Science
University of Nottingham
U.K.
ferrante.neri@nottingham.ac.uk

Abstract

The so-called "attention" is an efficient mechanism to improve the performance of convolutional neural networks. It uses contextual information to recalibrate the input to strengthen the propagation of informative features. However, the majority of the attention mechanisms only consider either local or global contextual information, which is singular to extract features. Moreover, many existing mechanisms directly use the contextual information to recalibrate the input, which unilaterally enhances the propagation of the informative features, but does not suppress the useless ones. This paper proposes a new attention mechanism module based on the correlation between local and global contextual information and we name this correlation as confidence. The novel attention mechanism extracts the local and global contextual information simultaneously, and calculates the confidence between them, then uses this confidence to recalibrate the input pixels. The extraction of local and global contextual information increases the diversity of features. The recalibration with confidence suppresses useless information while enhancing the informative one with fewer parameters. We use CIFAR-10 and CIFAR-100 in our experiments and explore the performance of our method's components by sufficient ablation studies. Finally, we compare our method with a various state-of-the-art convolutional neural networks and the results show that our method completely surpasses these models. We implement ConAM with the Python library, Pytorch, and the code and models will be publicly available.

1. Introduction

The rapid development of convolutional neural networks (CNNs) [1, 2] in the past ten years has greatly promoted various research fields of deep learning, such as image [3, 4, 5], time series [6], video [7, 8], audio [9], text [10], natural language [11] and generative adversarial networks [12, 13]. CNNs became the backbone because of their powerful representation ability. With the goal of enhancing the representation of CNNs, researchers have put in a lot of effort and obtained remarkable achievements. Based on the factors that affect the representation ability of CNNs, the existing methods can be divided into quantity-based and quality-based methods. Quantity-based methods can be viewed as "stacking". For example, width, depth, and cardinality are the three main factors that affect CNNs' performance. Stacking by increasing the width [14, 15], which means increasing the number of feature maps in a convolutional layer, is a kind of method to further reinforce the performance of the entire network. Expanding depth [16] of CNNs by stacking more convolutional layers, which is to extract high-dimensional and abstract features, is another method to enhance the representation ability. Different from the increase in width and depth, the increase in cardinality [17] broadens the convolution operations of a convolutional layer to enrich features. Hence, the cardinality can be seen as the third dimension of quantity-based methods to improve CNNs in addition to width and depth. Quality-based methods are the upgrading of original CNNs' components or the creation of new ones which are plug-and-play. For example, skip connection [19, 20] breaks the bottleneck of direct connection network design, so enables interactive fusion of features between different layers, and strengthens the features' propagation to enhance the representation. Another quality-based method focuses on designing new convolution operations, such as depth separable convolution [21, 22], dilated convolution [23], etc.

The quantity-based and quality-based operations can strengthen the representation of CNNs, but they have some drawbacks. First of all, the stacking of feature maps or convolutional layers greatly increases the number of parameters which is not beneficial to the device. Skip connection has no extra parameters, but during inference, the skip part requires additional storage memory. Depth-wise convolution can reduce the number of parameters, but because the channels are separated from the spatial dimension, spatial features on different channels cannot be considered, so the network performance may be compromised. Taking into account the above shortcomings, the attention mechanism is proposed to improve representation without increasing or increasing a few parameters. The attention mechanism simulates the human visual system that is paying more attention to the interesting area. The attention mechanism extracts the distribution of the image as contextual information and forwards it to the network. In other words, the attention mechanism biases the network toward valuable information instead of the background. More importantly, the attention mechanism can improve the performance of CNNs with significantly fewer parameters.

Most attention mechanisms use the mapping of local or global contextual information to directly polish all pixels of the input, which can intensify the spread of valuable information and make the network pay more attention to meaningful features. For example, SE-Net [24] is a representative attention mechanism which extracts global average pooling (GAP) of every channel as contextual information and recalibrates the input with the embedding of the contextual information, which can zoom in on the meaningful features. Different from SE-Net [24], GE-Net [25] focuses on local contextual information and uses its mapping to recalibrate the input, which is a more general style. Similar to GE-Net [25], SPA-Net [26] also considers local feature responses but from three different spatial scales to extract contextual information. In addition to the extraction of contextual information from the channel dimension performed by SPA-Net [26], CBAM [27] and BAM [28] extract contextual information from the extra spatial dimension and merge them, then recalibrate the input with the merged result.

Recalibration methods of these attention mechanisms are using multilayer perceptron (MLP) to map the extracted global or local contextual information and then perform pixel-based multiplication with the input feature maps, which allows each pixel to contain the whole image's contextual information. Therefore, the valuable global information is magnified, so the subsequent feature extraction pays more attention to the valuable information. These attention mechanisms only unilaterally amplify meaningful contextual information but do not propose how to suppress invalid ones. Obviously, the better mechanism should suppress ineffective features while highlighting valuable ones, and most attention mechanisms cannot perform both at the same time, because they do not take into consideration the relationship between local and global contextual information.

Based on the above analysis, this paper proposes a new plug-and-play attention module, ConAM, which is based on the correlation between local and global contextual information, and we call this correlation the "confidence" which serves as the similarity between local and global features. ConAM extracts the local and global contextual information, calculates the confidence between them, and finally recalibrates the input with the confidence. In detail, first, the input is divided into non-overlapping patches, and ConAM takes the average value of each patch and the entire input as the local and global contextual information, respectively, at the same time. Hence ConAM considers not only the global contextual information but also the local, which does not exist in any of existing attention mechanisms. Secondly and more importantly, ConAM calculates the confidence between the local and global contextual information by the inner product of them, which means ConAM focuses on the relationship between the two kinds of contextual information instead of the two kinds of information. We claim that this involvement of confidence is first proposed in CNNs. Finally, different from the existing attention mechanisms, ConAM uses the confidence to recalibrate the input patch by patch, which leads to a fewer number of needed parameters. In other words, the input recalibration relies on the correlation of local and global contextual information instead of the embedding of them. Namely, the more relevant the local features are to the global features, the more they are amplified, and the less relevant the more they are suppressed. Accordingly, the network not only pays attention to the informative information, but also further suppresses useless information.

1.1. Contribution

The main contribution of this paper can be summarized as follows:

1. We enrich features of CNNs by simultaneously extracting local contextual information and global contextual information.

2. We propose a novel input recalibration that relies on the correlation between local and global contextual information.

1.2. Organization

The remainder of this paper is organized as follows: Section 2 presents the related background information of ConAM. Section 3 demonstrates the mechanism of ConAM in detail. Section 4 shows the experiment results and analysis. Finally, Section 4.2 summarizes our work of ConAM.

2. Related Work

2.1. Deep Network Architectures

CNNs have been the focus of machine learning. AlexNet [29] first proved the high robustness of the CNN in visual classification. VGG [14] and Inception [15] further proved that the depth of CNNs is a crucial factor affecting performance by stacking convolutional layers. However, simply stacking the convolutional layers beyond a certain limit may degrade the network. Contrary to stacking, ResNet [19] proved that skip connections can significantly improve performance without increasing the number of layers and other parameters, and laid an important foundation for the development of CNNs. Based on ResNet, WideReNet [16] proved that increasing the network width can also enhance the network's performance. Different from WideReNet [16], ResNeXt [17] enriches the convolution styles by adding different convolution groups, i.e., cardinality, so as to increase the diversity of features. Most recent deep network architecture designs target how to broaden a factor, such as depth, width, and cardinality, which increases many parameters.

2.2. Attention Mechanism

The attention mechanism is recognized as an effective technique to improve CNNs' performance. It imitates the human visual system, that is, focusing on the area of interest instead of the panorama. Therefore, the attention mechanism makes the network inclined to learn more valuable information, rather than useless noise. SE-Net [24] is the most typical attention mechanism. It uses the mean value of each channel as global contextual information and integrates its embedding into the input, then flows this global knowledge into the network, so that the network can be aware of this knowledge. On the basis of SE-Net, various improvements have been made. For example, ECA-Net [30] proposed an adaptive convolution kernel method to achieve local interactions, thereby significantly reducing model complexity. CBAM [27] and BAM [28] add additional information about the spatial dimension. Different from extracting global contextual information, SPA-Net [26] and GE-Net [25] extract local contextual information to make contextual information more general. In this paper, we consider both local and global contextual information, and recalibrate the input based on the correlation between the local and global information.

3. Confidence Attention Module

In this section, we elaborate on ConAM. Here, we give a schematic of ConAM in Fig. 1. ConAM is a plug-and-play module, which means ConAM can be embedded in any convolutional layer without changing any hyperparameters in this layer. The confidence attention in ConAM consists of four operations, namely extraction of local and global contextual information, mapping of contextual information, confidence calculation, and recalibration. For clarity, detailed diagrams and formula are provided.

3.1. Extraction of Local and Global Contextual Information

Extracting the local and global contextual information is the premise for calculating the confidence between them. The local contextual information can be viewed as the feature response or the descriptor of the local receptive field and represent a general distribution. In detail, we stipulate that the acquisition of local contextual information satisfies three points. First, the average value is used to represent a feature response of the local receptive field. Second, different fields do not overlap each other. We define a local receptive field as a patch, and its width and height are P, so the size of each patch is P^2. For an input $\in \mathbb{R}^{H\times W}$, the number of patches is $(HW)/P^2$. Third, each channel has a feature response on the same spatial patch. Given an input $\in \mathbb{R}^{C\times H\times W}$, we can get a local contextual information matrix $(lc) \in \mathbb{R}^{(HW/P^2)\times C}$ as follows:

$$lc = \begin{bmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1C} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1} & \delta_{n2} & \cdots & \delta_{nC} \end{bmatrix}, \tag{1}$$

where $\delta_{ij} = AvgPool(patch_{ij})$, n denotes the number of patches and equals to $(HW)/P^2$, C denotes the number of channels and δ_{ij} denotes the feature response of $patch_{ij}$ which is the j^{th} patch in the i^{th} channel, and $patch_{ij}$ is $\in \mathbb{R}^{P\times P}$.

The global contextual information can be viewed as the feature response or the descriptor of the global receptive field and represents a general distribution. We also take the average value as a feature response which is obtained by each channel. In the same way as to achieve the local contextual information, the global contextual information matrix $(gc) \in \mathbb{R}^{C}$ is described as follows:

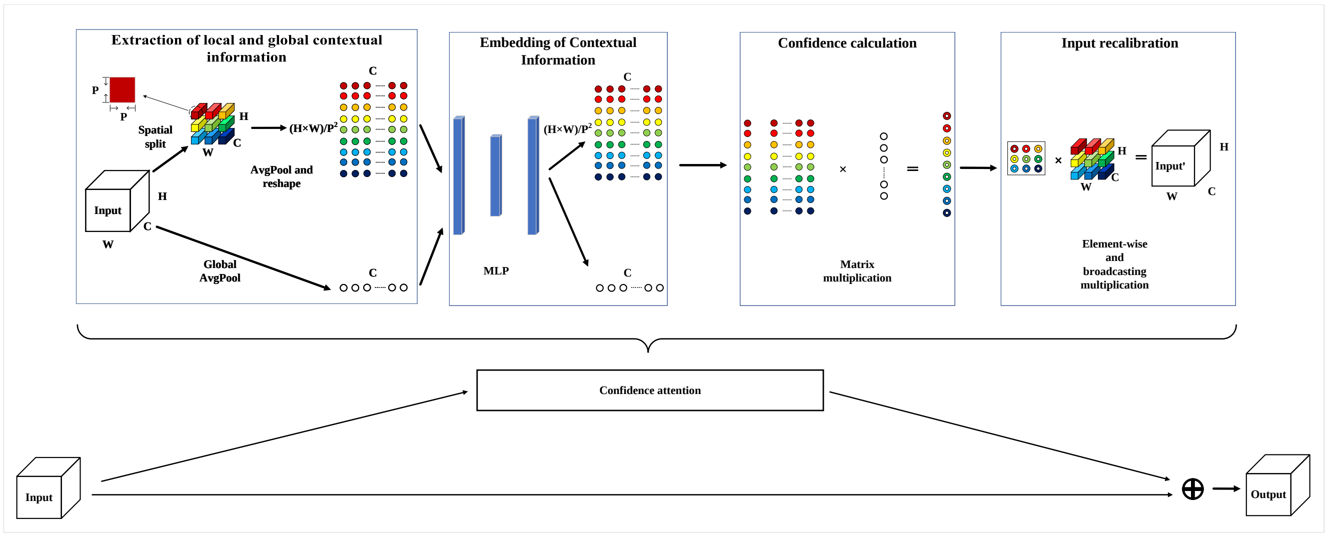


Figure 1: The overall process of ConAM which contains confidence attention and skip connection. The confidence attention process is composed of four parts and they are extraction of local and global contextual information, embedding of contextual information, confidence calculation, and input recalibration.

$$gc = \begin{bmatrix} \widehat{\delta_1} \\ \widehat{\delta_2} \\ \vdots \\ \widehat{\delta_C} \end{bmatrix}, \tag{2}$$

where $\widehat{\delta_i} = AvgPool(channel_i)$, C denotes the number of channels and $\widehat{\delta_i}$ denotes the global response (AvgPool) of $channel_i$ which is the i^{th} channel, and $channel_i$ is $\in \mathbb{R}^{H \times W}$.

3.2. Embedding of Contextual Information

Contextual information embedding [24, 27] is an indispensable process. It further refines the extracted contextual information, which can fuse the contextual information of all channels on the same receptive field and map more abstract information. Specifically, to save parameters and computational complexity, we perform a shared two-layer MLP to generate the embedding of the local and global contextual information. The first layer is used for dimension reduction and maps $\mathbb{R}^{C} \mapsto \mathbb{R}^{\widehat{C}}$. The second layer is used to increase the dimension to be consistent with the input and maps $\mathbb{R}^{\widehat{C}} \mapsto \mathbb{R}^{C}$. Meanwhile, to improve the robustness, each layer is followed by an activation function, i.e., ReLu [31]. The following formula $f(\bullet)$ expresses such a process:

$$f(X) = \sigma(W_2(\sigma(W_1 X))), \tag{3}$$

where σ denotes the activation function, W_1 and W_2 denote the first layer and second layer of MLP, respectively, and X denotes the contextual information, lc or gc, and for brevity, the bias is omitted.

3.3. Confidence Calculation

Confidence calculation is the core of ConAM. Its purpose is to calculate the similarity between local and global contextual information. We consider that the object to be recognized and the background are in different distributions and their distributions have a wide gap. Meanwhile, most objects to be recognized occupy most of the area of the image, so the distribution of the whole image is mainly affected by these objects, not the background. We believe that the distribution of these objects is similar to that of the whole image, while the background distribution is not. Therefore, we use the distribution of the entire image as a benchmark to expand the original difference between the object and the background, so that the network pays more attention to this difference in the learning process to highlight the object while suppressing the background.

We divide the image into non-overlapping patches and use the average value of each patch and entire image to represent their the distribution, then calculate the correlation of the embedding of distributions between the patch and the entire image to capture their similarity. We call the correlation the *confidence*. We use the dot product between the embedding of the patch and image distribution to acquire the $confidence$. Formally, given an input $\in \mathbb{R}^{C \times H \times W}$, we extract the distribution, lc and gc, then product their embedding matrix, $f(lc)$ and $f(gc)$, and perform matrix multiplication on these two matrices as shown below:

$$confidence = f(lc) \times f(gc)$$
$$= \begin{bmatrix} \delta_{11}^{'} & \delta_{12}^{'} & \cdots & \delta_{1C}^{'} \\ \delta_{21}^{'} & \delta_{22}^{'} & \cdots & \delta_{2C}^{'} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n1}^{'} & \delta_{n2}^{'} & \cdots & \delta_{nC}^{'} \end{bmatrix} \times \begin{bmatrix} \widehat{\delta_1}^{'} \\ \widehat{\delta_2}^{'} \\ \vdots \\ \widehat{\delta_C}^{'} \end{bmatrix} = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix}, \tag{4}$$

where η_i denotes the $confidence_i$, and it represents the similarity between $patch_i$ ($\in \mathbb{R}^{C\times P\times P}$) and the entire image ($\in \mathbb{R}^{C\times H\times W}$), $[\,\delta_{i1}^{'}\ \delta_{i2}^{'}\ \cdots\ \delta_{iC}^{'}\,]$ denotes the embedding of local contextual information and $[\,\widehat{\delta_1}^{'}\ \widehat{\delta_2}^{'}\ \cdots\ \widehat{\delta_C}^{'}\,]^T$ denotes the embedding of global contextual information.

Finally, we normalize them with the normalization function *norm*(●) to generate a $confidence$ matrix as follows:

$$confidence = norm(confidence), \tag{5}$$

3.4. Recalibration

Image recalibration is the last step of ConAM which multiplies each patch with its corresponding confidence as shown below:

$$\begin{aligned} input &= input \times confidence \\ &= (patch_1, patch_2, ..., patch_m) \circ (\eta_1, \eta_2, ..., \eta_m), \end{aligned} \tag{6}$$

where $m = (HW)/P^2$, $\circ$ is the Hadamard product and η_i needs to be broadcast from $\mathbb{R}$ to $\mathbb{R}^{C\times P\times P}$ to match $patch_i$. Fig. 2 shows the process of recalibration when the number of patches is nine. Firstly, the image is split into non-overlapping patches, which is the same as the process of generating $confidence$. Then these patches are multiplied by $confidence$, before which the η_i ($\in \mathbb{R}$) in $confidence$ should be broadcast to $\mathbb{R}^{C\times P\times P}$ to match the size of $patch_i$. Finally, these recalibrated patches are recombined into feature maps.

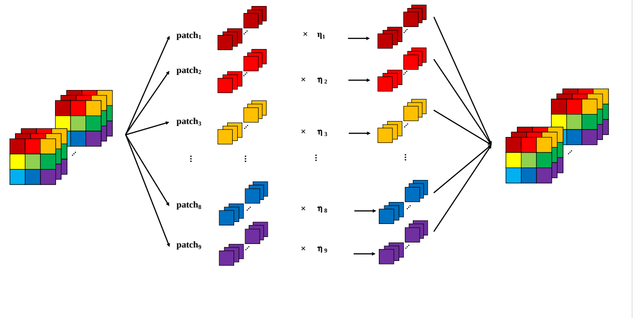


Figure 2: The overall process of recalibration.

4. Experiments and Results Analysis

To thoroughly validate the performance of our module, we first perform ablation studies, and then compare ConAM with four different kinds of state-of-the-art CNNs, including classical models, models generated by neural architecture search (NAS) [32, 33, 34], which is divided into semi-automatic NAS and full-automatic NAS according to their search method, and the models based on the popular existing attention mechanisms. Each experiment is performed three times due to the high computational overhead, and the average results are presented. Besides, to facilitate the exploration of the influence of different patch sizes, we modify the first layer with filter size of 7 and stride of 2 into the layer with filter size of 3 and stride of 1. Meanwhile, the following MaxPooling is deleted. Hence the input spatial size of each block in ResNet [19] is transformed from [8, 8, 4, 2] to [32, 32, 16, 8] and this gives greater flexibility in designing different patch sizes to study the impact of patch size. Every model in all experiments follows this modification.

4.1. Ablation studies

In this section, we show our design choice in the process of studying ConAM. For the ablation studies, we use datasets of CIFAR-10 [18] and CIFAR-100 [18]. The CIFAR [18] dataset is the most used dataset in deep learning. It is divided into CIFAR-10 [18] with 10 categories and CIFAR-100 [18] with 100 categories. Each dataset contains 60k RGB images, with 50k used for training and 10k used for testing. Our module design process includes exploring three kinds of attention strategies, exploring the normalization and activation, and verifying on other different models to study the impact of different patch sizes on ConAM.

4.1.1 Experiment of three attention strategies

Experiment design: To better understand the details of ConAM, we design three different strategies to extract the $confidence$. The first strategy calculates the $confidence$ with the pure contextual information without mapping as shown below:

$$confidence = lc \times gc, \tag{7}$$

The second strategy obtains the $confidence$ without mapping, but the $confidence$ is fused with the local contextual information as shown below:

$$confidence = lc \times gc \tag{8}$$

and

$$confidence = confidence \circ lc, \tag{9}$$

where $\circ$ denotes the Hadamard product. This strategy aims to allow the network to consider both local contextual information and $confidence$.

The third strategy uses the shared two-layer MLP without activation to map the local and global contextual information. We follow the setting of CBAM [27], set the first layer to reduce the input to 1/16 of the original, then calculate the $confidence$ with their embeddings as shown below:

$$confidence = (W_2\,(W_1\,(lc))) \times (W_2\,(W_1\,(gc))) \tag{10}$$

Implementation: CIFAR-10 [18] and ResNet-50 [19] are used to test the three strategies. We embed ConAM at the beginning of each block and take the size of the local receptive field as 1/2 of the input spatial size of each block. An NVIDIA 2080TI graphic processing unit (GPU) is used to implement the experiment. Data augmentation involves random cropping, random horizontal flipping, and Cutout [35]. Random cropping fills four zeros on all borders of the image, and then randomly crops the image with a size of 32 $\times$ 32. We use the SGD optimizer with a momentum of 0.9 and a weight decay of 5e-4. A total of 250 epochs is set for training, with a batch size of 128. The training accuracy is recorded every 30 iterations in each epoch, and the test accuracy is recorded once in each epoch. To clearly understand the performance of these three strategies, two kinds of experiments with different learning rate strategies are designed as shown in TABLE 1 and each experiment is conducted three times.

Table 1: Learning rate setting.

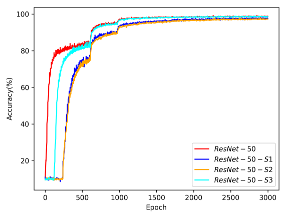
Epoch	0∼50	50∼80	80∼120	120∼250
Learning rate$_1$	0.1	0.01	0.001	0.0008
Epoch	0∼60	60∼120	120∼170	170∼250
Learning rate$_2$	0.1	0.01	0.001	0.0008

Result and analysis: Fig. 3 shows the training and test curves of ResNet-50 [19] on CIFAR-10 [18] with these two learning rates. (a) and (b) are the first experiment with Learing_rate$_1$, (c) and (d) are the second experiment with Learing_rate$_2$. To clearly compare the performance of these three strategies, we only show the baseline curve in the first experiment. TABLE 2 shows the accuracy of the first experiment.

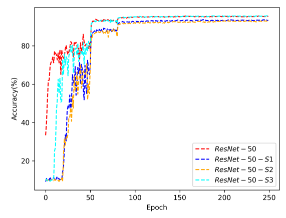
Table 2: The result of the first experiment.

Model	ResNet-50 [19]	ResNet-50-S1	ResNet-50-S2	ResNet-50-S3
Accuracy (%)	95.21	93.21	92.50	**95.38**

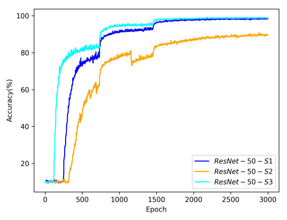
It can be seen from TABLE 2 that ResNet-50-S3 achieves the highest accuracy among the three strategies.



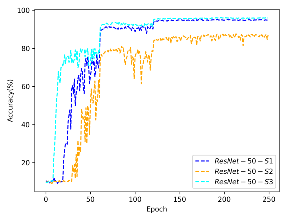
(a) Training$_1$



(b) Test$_1$



(c) Training$_2$



(d) Test$_2$

Figure 3: Accuracy curves of training and test during CIFAR-100 [18] training. (a) and (b) are the curves of Learning rate$_1$, while (c) and (d) are the curves of Learning rate$_2$.

ResNet-50-S1 and ResNet-50-S2 have the lowest accuracy, but the accuracy of ResNet-50-S1 is greater than that of ResNet-50-S2.

In terms of convergence, it can be inferred from Fig. 3 that ResNet-50-S1, ResNet-50-S2, and ResNet-50-S3 all show fitting stagnation in the early stage, but the fitting stagnation does not exist in the baseline. In the first experiment, the stagnation of ResNet-50-S3 is about half that of ResNet-50-S1 and ResNet-50-S2, and the stagnation of ResNet-50-S1 and ResNet-50-S2 is the same. In the second experiment, the stagnation of ResNet-50-S3 is still shorter than that of ResNet-50-S1 and ResNet-50-S2, while the stagnation of ResNet-50-S2 is the longest. Compared to the first experiment, the performance of ResNet-50-S2 is severely degraded.

We can infer that the performance of ResNet-50-S3 is the best among these three strategies. Although it stagnates in the early stage, the final accuracy is higher than that of the baseline. The performance and convergence of ResNet-50-S3 are better than those of ResNet-50-S1 and ResNet-50-S2, which is thanks to the feature mapping of ResNet-50-S3 using MLP. We argue that any distribution extracted from the input is noise if it is not mapped. The $confidence$ of ResNet-50-S1 directly uses the unprocessed local and global contextual information. Although the $confidence$ is the correlation between the local and global contextual information, it does not participate in the learning process of the network through some certain parameters, so its information cannot be recognized in the early stage.

Hence, for any other parameters that are learned and updated, such $confidence$ is just additional noise at the beginning. As for ResNet-50-S2, its $confidence$ is multiplied by the local contextual information, i.e., $confidence = confidence \circ lc$, which leads to the destruction of the original $confidence$. In addition, neither of them participates in the learning process, so the performance of ResNet-50-S2 is the worst and unstable. ResNet-50-S3 has undergone feature mapping by MLP, and the embedding of the local and global distribution information flows into other parameters in the network through the gradient, which leads to an increase in performance.

4.1.2 Experiment of normalization and activation

Experiment design: We verify the role of normalization and activation functions in ConAM with ResNet-50-S3. Meanwhile, to reflect the performance of ConAM on a more complex dataset, we use CIFAR-100 [18] and adopt ResNet-50 [19] as the baseline. We design three strategies. The first strategy, ResNet-50-S3-S, allows the $confidence$ to be processed by the normalization function, Softmax, as shown below:

$$confidence = Softmax\left(\left(W_2\left(W_1\left(lc\right)\right)\right) \times \left(W_2\left(W_1\left(gc\right)\right)\right)\right) \tag{11}$$

The second strategy, ResNet-50-S3-SR, is based on the former, adding the activation function, i.e., ReLu [31], to each layer of the shared two-layer MLP as shown below:

$$confidence = Softmax\left(\sigma\left(W_2\left(\sigma\left(W_1\left(lc\right)\right)\right)\right) \times \sigma\left(W_2\left(\sigma\left(W_1\left(gc\right)\right)\right)\right)\right) \tag{12}$$

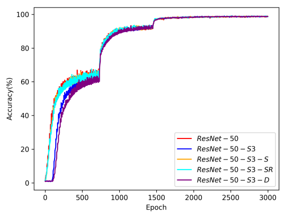
The third strategy, ResNet-50-S3-D, is based on ResNet-50-S3, adding regularization method, Dropout [36], to each layer of the shared two-layer MLP, and the property of Dropout [36] is set to 0.3.

Implementation: We still use the hyperparameters of training in Section 4.1.1, but the learning rate uses $\mathrm{Learning\ rate}_2$.

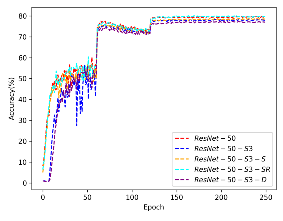
Table 3: The result of CIFAR-100 [18] training.

Model	ResNet-50 [19]	+ResNet-50-S3	ResNet-50-S3-S	ResNet-50-S3-SR	ResNet-50-S3-D
Accuracy (%)	79.39	77.96	78.41	**79.72**	77.03

Result and analysis: Fig. 4 demonstrates the training curve of the baseline, ResNet-50-S3 and these three models, and TABLE 3 shows the accuracy result. The training accuracy of these five models is almost the same. The performance of ResNet-50-S3-SR surpasses the baseline and is the highest, the accuracy of ResNet-50-S3-D is the



(a) Training



(b) Test

Figure 4: Accuracy curves of training and test during CIFAR-100 [18] training. (a) is the curve of training and (b) is the curve of test.

lowest, the accuracy of ResNet-50-S3 and ResNet-50-S3-S is second only to the baseline, and the performance of ResNet-50-S3-S exceeds that of ResNet-50-S3. In the early stage of training, the fitting stagnation phenomenon appears in ResNet-50-S3 and ResNet-50-S3-D. The fitting stagnation disappears in ResNet-50-S3-S and ResNet-50-S3-SR. At the same time, between the 40^{th} and 60^{th} epoch, the accuracy of ResNet-50-S3 fluctuates violently. This phenomenon does not exist in the other models but in CIFAR-10 [18] training as shown in Section 4.1.1.

Combining the results in Section 4.1.1, we argue that this stagnation originates from the "cognitive speed" of the network for confidence. If there is no mapping like MLP or Softmax, the network will understand this knowledge (confidence) very slowly, and with the addition of MLP, this knowledge is transmitted to the entire network by parameters, so the network can quickly recognize and absorb this knowledge and accelerate the convergence of the network. On this basis, the Softmax function further accelerates this "cognitive speed", hence both ResNet-50-S3-S and ResNet-50-S3-SR converge at the beginning and outperform ResNet-50-S3 in the initial convergence and final performance. Moreover, we believe that Softmax and Dropout [36] can alleviate the fluctuation of accuracy, which is why there is no fluctuation except for ResNet-50-S3 in the 40^{th} to 60^{th} epoch. In addition, the reason why ResNet-50-S3-S is not better than the baseline is over-fitting caused by MLP, so the performance surpasses the baseline after adding the ReLu function [31].

4.1.3 Experiment of patch size

Experiment design: This section verifies the impact of different patch sizes and normalization functions on ConAM performance. To validate the general effectiveness of ConAM for different models on different datasets, we use the classic models, ResNet-18 [19], ResNet-34 [19], ResNet-50 [19] and ResNet-101 [19], as baselines, and train them on CIFAR-10 [18] and CIFAR-100 [18].

The patch is the basic unit to extract local contextual information. In the above experiment, we adopt ConAM at the beginning of each block of ResNet-50 [19], and the patch size is set to [16, 16, 8, 4]. However, the network width is gradually increasing, and the spatial size is gradually shrinking, so the deeper the layer, the larger the area where its receptive field radiates to the front layer. Considering that the early feature maps mainly contain low-level semantic features, such as the edges and corners, the extracted average value can reflect the local distribution, but the later feature maps contain high-level semantic features, and the average value extracted lose the meaning of local distribution. Therefore, we design three combinations of patch sizes, i.e., [8, 4, 2, 0], [8, 8, 2, 0] and [8, 8, 4, 0]. 0 means no ConAM is added. In addition, we employ the other function, Sigmoid, to verify the normalization impact on the performance of ConAM.

Implementation: All models are based on ResNet-50-S3-SR. ResNet-101 [19] is trained on an NVIDIA V-100 GPU, and the rest are trained on an NVIDIA 2080TI GPU. Data augmentation still uses the above method. We train ResNet-50 [19] and ResNet-101 [19] with two learning rate settings, and the learning rate and batch size are set as shown in TABLE 4 and 5.

Table 4: Batch size setting.

Model	ResNet-18 [19] and ResNet-34 [19]		ResNet-50 [19] and ResNet-101 [19]	
Dataset	CIFAR-10 [18]	CIFAR-100 [18]	CIFAR-10 [18]	CIFAR-100 [18]
Batch size	250		150	250

Table 5: Learning rate setting.

Model	ResNet-18 [19] and ResNet-34 [19]				ResNet-50 [19] and ResNet-101 [19]			
Dataset	CIFAR-10 [18] and CIFAR-100 [18]				CIFAR-10 [18] and CIFAR-100 [18]			
Epoch_1	0-60	60-100	100-150	150-250	0-40	40-70	70-100	100-150/250
Learning rate_1	0.1	0.01	0.001	0.0008	0.1	0.01	0.001	0.0008
Epoch_2	-	-	-	-	0-60	60-120	120-150	150-250
Learning rate_2	-	-	-	-	0.1	0.01	0.001	0.0008

Result and analysis: TABLE 6 shows the result of ConAM on ResNet-18 [19], ResNet-34 [19], ResNet-50 [19], and ResNet-101 [19] with different datasets, normalization functions, patch sizes, and learning rates. We can infer from the results that the accuracy of the proposed ConAM on all models and dataset reaches the maximum value. The combination of Softmax and [8, 8, 4, 0] achieves the best performance on CIFAR-100 [18] for all models and learning rates. For CIFAR-10 [18], the combination of Sigmoid and [8 ,8, 4, 0] can achieve the best effect on almost all models. Especially for ResNet-18 [19], our ConAM with almost all patch sizes and normalization functions surpasses the baseline. As for ResNet-50 [19] with Learning rate_1, the performance of ConAM with almost all patch sizes and normalization functions on CIFAR-100 [18] outperforms the baseline. The performance of ResNet-101 [19] enhanced by ConAM on CIFAR-10 [18] almost completely exceeds the original model. In other models, ConAM also achieves competitive results. In the verification results of ResNet-34 [19] on CIFAR-100 [18] and ResNet-50 [19] on CIFAR-10 [18], ConAM obtains better performance than the original model.

4.2. Comparison with popular CNNs

Experiment design: We compare ConAM with currently popular models. To fully demonstrate the excellent performance of ConAM, we compare it with four different groups of CNNs, including the classic models designed manually, the models searched by NAS, and the models based on the attention mechanism. The models searched by NAS are divided into semi-automatic and full-automatic searched models according to the search method. We use ConAM-ResNet-101, in which ConAM with ReLu [31] uses two combinations, Softmax with [8, 8, 4, 0] and Sigmoid with [8, 8, 4, 0], because these two combinations get the best performance on CIFAR-10 [18] and CIFAR-100 [18] in the above experiments.

Implementation: We employ random cropping, random horizontal flipping, and random erasing [37]. The learning rate uses the Learning rate_2 in Section 4.1.3, and the other hyperparameters remain unchanged. Meanwhile, we re-implement the two attention mechanism based models, SE-ResNet-101 [24] and CBAM-ResNet-101 [27], with the same hyperparameters.

Result and analysis: TABLE 7 shows the comparison results. It is divided into 4 parts, which are the classic CNNs designed by hand, the CNNs searched by semi-automatic NAS, the CNNs searched by full-automatic NAS, and the two most popular networks based on attention mechanism, SE-ResNet-101 [24] and CBAM-ResNet-101 [27]. The symbol '-' means there is no result publicly reported by the corresponding competitor. The table also gives the parameters of these models. It can be inferred that the proposed ConAM outperforms all types of CNNs. ConAM's accuracy on CIFAR-10 [18] and CIFAR-100 [18] is 1.30 and 3.18 higher than the highest accuracy of the manually designed network, 94.78 and 77.70, respectively. Compared with the semi-automatic searched network, the accuracy of ConAM on CIFAR-10 [18] and CIFAR-100 [18] is 0.31 and 1.53 higher than the highest accuracy of 95.77 and 79.35, respectively. It is 0.38 and 1.73 higher than the best accuracy of the full-automatic searched network, 95.70 and 79.15 respectively. It is 0.33 and 1.62 higher than the highest values of the two networks based on the attention mechanism. It can be seen that ConAM performs better on more complex datasets. More importantly, in the attention group, not only is the performance of ConAM better than SE-ResNet-101 [24] and CBAM-ResNet-101 [27], but also the

Table 6: The result of ConAM on ResNet-18 [19], ResNet-34 [19], ResNet-50 [19], and ResNet-101 [19] with different datasets, normalization functions, patch sizes, and learning rates.

Model		ResNet-18 [19]		ResNet-34 [19]	
Dataset		CIFAR-10 [18]	CIFAR-100 [18]	CIFAR-10 [18]	CIFAR-100 [18]
Baseline		95.38	76.31	95.69	77.57
+Softmax	[8, 4, 2, 0]	**95.44**	**76.38**	95.68	76.80
	[8, 8, 2, 0]	**95.51**	**76.37**	95.67	77.18
	[8, 8, 4, 0]	95.23	**76.60**	**95.76**	**78.20**
+Sigmoid	[8, 4, 2, 0]	**95.42**	**76.41**	95.50	**77.81**
	[8, 8, 2, 0]	**95.51**	**76.82**	95.46	77.06
	[8, 8, 4, 0]	**95.39**	76.10	95.61	**77.73**

Model		ResNet-50 [19]				ResNet-101 [19]			
Learning rate		Learning rate$_1$		Learning rate$_2$		Learning rate$_1$		Learning rate$_2$	
Dateset		CIFAR-10 [18]	CIFAR-100 [18]	CIFAR-10 [18]	CIFAR-100 [18]	CIFAR-10 [18]	CIFAR-100 [18]	CIFAR-10 [18]	CIFAR-100 [18]
Baseline		94.94	77.43	95.26	79.33	95.10	78.88	95.47	79.78
+Softmax	[8, 4, 2, 0]	93.81	77.37	95.00	78.83	94.94	78.34	**95.67**	79.58
	[8, 8, 2, 0]	94.27	**77.76**	**95.43**	77.25	94.67	78.66	**95.75**	79.19
	[8, 8, 4, 0]	94.59	**77.63**	94.83	**79.76**	94.93	78.55	95.35	**79.87**
+Sigmoid	[8, 4, 2, 0]	94.30	**78.05**	**95.61**	78.86	94.93	78.16	**95.87**	79.15
	[8, 8, 2, 0]	94.82	**78.02**	94.98	79.31	**95.30**	78.78	**95.82**	78.93
	[8, 8, 4, 0]	**95.07**	**78.77**	**95.35**	78.88	**95.16**	**78.95**	**95.82**	79.09

Table 7: Comparison between the proposed ConAM and the state-of-the-art peer competitors in terms of the classification accuracy, the number of parameters on the dataset CIFAR-10 [18] and CIFAR-100 [18].

Accuracy (%) / Dataset / Model	CIFAR-10 [18]	CIFAR-100 [18]	Parameter(M)	
DenseNet(k=12) [20]	94.76	75.58	1.0	Hand-crafted
Maxout [38]	93.57	61.40	-	Hand-crafted
VGG [14]	93.34	71.95	20.04	Hand-crafted
Network in Network [39]	91.19	64.32	-	Hand-crafted
Highway Network [40]	92.28	67.61	-	Hand-crafted
All-CNN [41]	92.75	66.29	-	Hand-crafted
FractalNet [42]	94.78	77.70	38.6	Hand-crafted
Genetic CNN [43]	92.90	70.95	-	Semi-automatic
EAS [44]	95.77	-	23.4	Semi-automatic
Block-QNN-S [45]	95.62	79.35	6.1	Semi-automatic
Large-scale Evolution [46]	94.60	-	5.4	Full-automatic
Large-scale Evolution [46]	-	77.00	40.4	Full-automatic
CGP-CNN [47]	94.02	-	2.64	Full-automatic
NAS [48]	93.99	-	2.5	Full-automatic
MetaQNN [49]	93.08	72.86	-	Full-automatic
AE-CNN [50]	95.70	-	2.0	Full-automatic
AE-CNN [50]	-	79.15	5.4	Full-automatic
SE-ResNet-101 [24]	95.34	-	47.29	Attention
SE-ResNet-101 [24]	-	79.22	47.48	Attention
CBAM-ResNet-101 [27]	95.75	-	47.29	Attention
CBAM-ResNet-101 [27]	-	79.26	47.48	Attention
ConAM(Softmax+[8, 8, 4, 0])(Ours)	**96.08**	-	42.56	Attention
ConAM(Sigmoid+[8, 8, 4, 0])(Ours)	-	**80.88**	42.74	Attention

number of parameters on CIFAR-10 [18] and CIFAR-100 [18] are about 4.7 M (10%) lower than that of SE-ResNet-101 [24] and CBAM-ResNet-101 [27].

sectionConclusion The goal of this paper is to develop a new attention module based on confidence in CNNs, ConAM, which considers local and global contextual information at the same time, calculates the confidence between them, and uses the confidence to recalibrate the input. We explore in detail various factors that affect the performance of ConAM, including MLP, normalization function, patch size, etc., and come up with the best combination of them to optimize ConAM performance. We compared ConAM with typical CNNs and state-of-the-art CNNs including the models searched by NAS and the popular attention mechanism-based models on CIFAR-10 and CIFAR-100. The results show that the performance of ConAM surpasses all these models and achieves the highest accuracy. More importantly, experiments show that ConAM can use fewer parameters to obtain higher performance gains compared to its peer attention mechanism-based models.

References

[1] Wang, Xuesong and Bao, Achun and Cheng, Yuhu and Yu, Qiang. Multipath ensemble convolutional neural network. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(2): 298–306, 2021. 1

[2] Liang, Stephen D, ‘ Optimization for deep convolutional neural networks: how slim can it go?’ *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2): 171–179, 2020. 1

[3] Guo, Haonan and Li, Shenghong and Qi, Kaiyue and Guo, Ying and Xu, Zhengwu. Learning automata based competition scheme to train deep neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2) : 151–158, 2020. 1

[4] Liu, Li and Ouyang, Wanli and Wang, Xiaogang and Fieguth, Paul and Chen, Jie and Liu, Xinwang and Pietikainen, Matti. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2): 261–318, 2020. 1

[5] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Evolution of image segmentation using deep convolutional neural network: a survey. *Knowledge-Based Systems*, 201:106062, 2020. 1

[6] Harbola, Shubhi and Coors, Volker. One dimensional convolutional neural network architectures for wind prediction. *Energy Conversion and Management*, 195: 70–75, 2019.

[7] T. Chi Hsuan and K. Yan Fu. Detecting and counting harvested fish and identifying fish types in electronic monitoring system videos using deep convolutional neural networks *ICES Journal of Marine Science*, 77(4): 1367–1378, 2019.

[8] S. Rencheng, Z. Senle, L. Chang, Z. Yunfei, C. Juan and C. Xun, Heart rate estimation from facial videos using a spatiotemporal representation with convolutional neural networks. *IEEE Transactions on Instrumentation and Measurement*, 69(10): 7411–7421, 2020.

[9] W. Junqi, C. Bolin, L. Weiqi and F. Yanmei. Audio steganography based on iterative adversarial attacks against convolutional neural networks. *IEEE Transactions on Information Forensics and Security*, 15: 2282–2294, 2020.

[10] Yousef Mohamed, Hussain Khaled F and Mohammed Usama S. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks *Pattern Recognition*, 108: 107482, 2020. 1

[11] Gimenez Maite, Palanca Javier and Botti Vicent, Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. A case of study in sentiment analysis. *Neurocomputing*, 378: 315–323, 2020. 1

[12] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021. 1

[13] H. M. Dipu Kabir, Abbas Khosravi, Saeid Nahavandi, and Abdollah Kavousi-Fard. Partial adversarial training for neural network-based uncertainty quantification. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(4):595–606, 2021. 1

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

[17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 1

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. 1, 3, 9

[19] 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

[20] 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. 1, 3

[21] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1, 3

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5, 6, 7, 8, 9, 10

[23] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 1, 3, 5, 6, 7, 8, 9

[24] 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, 9

[25] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. *arXiv preprint arXiv:1810.12348*, 2018. 2

[26] Jingda Guo, Xu Ma, Andrew Sansom, Mara McGuire, Andrew Kalaani, Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Spanet: Spatial pyramid attention network for enhanced image recognition. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020. 2

[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. 2

[28] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018. 2, 3, 4, 8, 9, 10

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 2, 3

[30] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11531–11539, 2020. 2, 3

[31] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011. 2, 3, 4, 6, 8, 9, 10

[32] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 2, 3

[33] Damien O'Neill, Bing Xue, and Mengjie Zhang. Evolutionary neural architecture search for high-dimensional skip-connection structures on densenet style networks. *IEEE Transactions on Evolutionary Computation*, 2021. 3

[34] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics*, 50(9):3840–3854, 2020. 3

[35] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 4, 7, 8

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 5

[37] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 5

[38] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013. 5

[39] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 6

[40] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 7

[41] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 8

[42] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 9

[43] Lingxi Xie and Alan Yuille. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1379–1388, 2017. 9

[44] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 9

[45] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2423–2432, 2018. 9

[46] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017. 9

[47] M. Suganuma, S. Shirakawa, and T. Nagao,. A genetic programming approach to designing convolutional neural network architectures. *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504, 2017. 9

[48] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 9

[49] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 9

[50] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. Completely automated cnn architecture design based on blocks. *IEEE transactions on neural networks and learning systems*, 31(4):1242–1254, 2019. 9

9

9

9