# Exploiting Position Information in Convolutional Kernels for Structural Re-parameterization

**Tianxiang Hao**[1,2], **Hui Chen**[3] and **Guiguang Ding**[1]
[1]School of Software, Tsinghua University
[2]Hangzhou Zhuoxi Institute of Brain and Intelligence
[3]Beijing National Research Center for Information Science and Technology (BNRist)
beyondhtx@gmail.com, jichenhui2012@gmail.com, dinggg@tsinghua.edu.cn

## Abstract

In order to boost the performance of a convolutional neural network (CNN), several approaches have shown the benefit of enhancing the spatial encoding of feature maps. However, few works paid attention to the positional properties of convolutional kernels. In this paper, we demonstrate that different kernel positions are of different importance, which depends on the task, dataset and architecture, and adaptively emphasizing the informative parts in convolutional kernels can lead to considerable improvement. Therefore, we propose a novel structural re-parameterization Position Boosting Convolution (PBConv) to exploit and enhance the position information in the convolutional kernel. PBConv consists of several concurrent small convolutional kernels, which can be equivalently converted to the original kernel and bring no extra inference cost. Different from existing structural re-parameterization methods, PBconv searches for the optimal re-parameterized structure by a fast heuristic algorithm based on the dispersion of kernel weights. Such heuristic search is efficient yet effective, well adapting the varying kernel weight distribution. As a result, PBConv can significantly improve the representational power of a model, especially its ability to extract fine-grained low-level features. Importantly, PBConv is orthogonal to procedural re-parameterization methods and can further boost performance based on them.

## 1 Introduction

Convolutional neural network (CNN) is widely used in various visual tasks [Krizhevsky *et al.*, 2012; Ren *et al.*, 2015; Toshev and Szegedy, 2014] and produces impressive results, greatly promoting the development of computer vision. How to improve the performance of various visual models becomes a very popular topic [Hu *et al.*, 2018; Ding *et al.*, 2021c; Ding *et al.*, 2019; Wang *et al.*, 2024b; Xiong *et al.*, 2024; Shen *et al.*, 2024; Hao *et al.*, 2024]. As is known to all, the convolutional operator aggregates neighbor information to obtain feature representation. Many prior works [Chen *et al.*, 2017; Xu *et al.*, 2015; Hu *et al.*, 2018] have found that the position information of features has a significant impact on the aggregation results, especially for fine-level image analysis tasks such as pixel-level semantic segmentation [Fu *et al.*, 2019; Zhong *et al.*, 2020]. For example, spatial-wise attention [Chen *et al.*, 2017] promotes the model by modeling the importance of feature maps. However, few works focus on the role of different positions of convolutional kernels when designing convolution-based networks. While, in this paper, we deeply analyze such a characteristic of the convolutional kernel and improve the performance of CNNs. We find the positional distributions of convolutional kernels are diverse and are influenced by tasks, architectures, and internal layer order as in Fig. 2. In addition, we find that pruning weight on the position with larger magnitude leads to more performance drop as shown in Fig. 3. Therefore, it is appropriate for us to use magnitude to represent the importance of positions. If a kernel position has a large magnitude, then this position is more important compared with the position with a smaller magnitude. Furthermore, we notice that the position distribution of kernel weights is approximately uniform after the beginning random initialization, but rapidly evolves towards dispersion as the training progresses. Based on this observation, we hypothesize that reinforcing the strong and weakening the weak of convolutional kernel weights' positions would help improve the model's performance, and design PBConv to enlarge such gap. Afterward, Fig. 9 will demonstrate that our goal has indeed been achieved.

Motivated by such observation, we propose a novel Position Boosting Convolution (PBConv) to replace the original convolution. A PBConv adaptively learns the importance of kernel positions, promoting the important ones and suppressing the unimportant ones. Our PBConv follows the principle of re-parameterization (rep for short ) [Ding *et al.*, 2019; Ding *et al.*, 2021c; Ding *et al.*, 2021d; Hu *et al.*, 2022; Huang *et al.*, 2022; Ding *et al.*, 2022; Ding *et al.*, 2024; Ding *et al.*, 2021a; Hao *et al.*, 2023a; Hao *et al.*, 2023b; Hao *et al.*, 2023c; Guo *et al.*, 2020; Wang *et al.*, 2024a] mechanism, which trains an over-parameterized model but merge redundant parameters before inference, ultimately retaining only the necessary structure. In this way, these techniques can fully exploit the capacity of the model during training to improve performance without adding any inference overhead. However, existing structural re-parameterization techniques [Ding *et al.*, 2019; Ding *et al.*, 2021c] ignore the in-

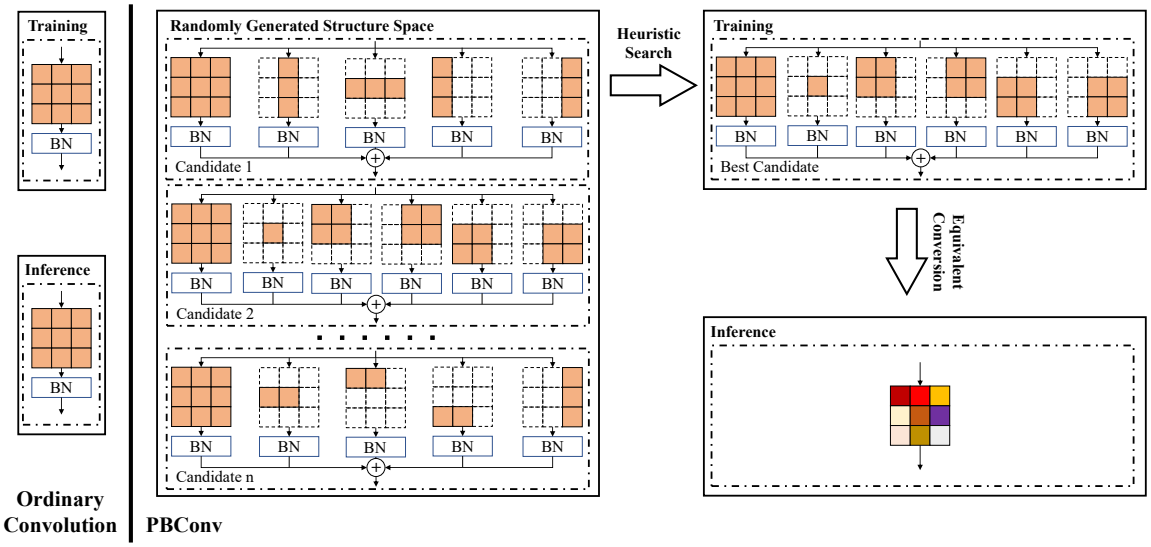


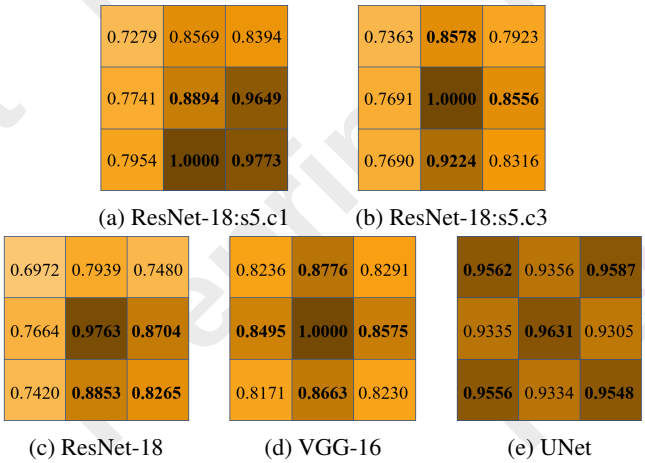

Figure 1: Overview of PBConv. For an off-the-shelf model, we replace each convolutional layer with our PBConv. The structure of PBConv for a particular model is decided by heuristic search. First, we randomly generate a series of legal candidates of PBConv whose small kernels could cover all regions of the original kernel. We train the models with each PBConv candidate replacing the original convolution for quite few iterations and choose the best candidate according to the dispersion of the positional distribution of the average of each layer's equivalent convolutional kernel. The total time cost for candidate selection is negligible compared with training. After training, all branches are merged into the original kernel. Finally, PBConv restores an enhanced single convolutional layer in inference and brings *free* improvements.

ternal properties of convolutional kernels and lead to inferior performance. In contrast, as in Fig. 1, PBConv uses multiple concurrent small convolutional kernels and adaptively emphasizes the informative parts of the original kernel. PBConv follows a divide-and-conquer philosophy. Given a convolutional kernel, we regard it as a combination of sub-regions. We tactfully configure several smaller kernels over it. Each of them is responsible for adjacent positions. Such a structure can enable small kernels to learn detailed and discriminative information about the relations between neighbor pixels in the local receptive field. As a result, PBConv strengthens the *fine-grained low-level feature* extraction, leading to significant improvements. To find the optimal structure of PBConv, we first randomly generate a set of legal candidate that covers the whole region and thus can adaptively learn the importance of each position, and then choose the best candidate according to the dispersion of the position distribution after a quick few iterations training, which is strongly correlated with the final performance as in Fig. 6. Such a heuristic search scheme enables us to adaptively select structures with superior performance compared to those designed by existing re-parameterization methods like [Ding *et al.*, 2019; Ding *et al.*, 2021c]. Besides, benefiting from the linearity as in prior re-parameterization works, our PBConv can trigger the aggregation among kernel weights. After training, PBConv could easily merge the complex structures into the simplest one, and enjoy improvement without any extra inference cost. Better still, our PBConv performs layer-level re-parameterization design and is orthogonal to procedure-level re-parameterization methods like [Ding *et al.*, 2023; Huang *et al.*, 2022] and architecture-level re-parameterization methods like [Ding *et al.*, 2021d]. Experiments show PBConv can further improve such orthogonal methods.

We conduct extensive experiments on four vision tasks, ranging from image-level tasks, and object-level tasks to pixel-level tasks. Experimental results show that PBConv can consistently achieve superior performance compared with existing state-of-the-art methods, improving plenty of architectures on various datasets and tasks. The experiments well demonstrate the superiority of PBConv, indicating a promising application in more practical scenarios.

Overall, we summarize our contributions as follows.

- We discover the positional distributions of convolutional kernel weights are diverse. Weights in different positions are of different importance. Motivated by such observation, we propose to selectively emphasize informative positions and suppress less useful ones.
- We propose PBConv to replace the convolutional layer. PBConv consists of parallel small kernels that aim at the extraction of low-level visual features, showing superiority, especially on pixel-level and object-level tasks.
- We find the dispersion of the convolutional kernel's position distribution is strongly correlated with the final performance, even in very early training. Such findings inspire us to design a heuristic search algorithm to adaptively acquire the optimal PBConv structure with negligible cost. PBConv follows the re-parameterization schema and thus brings no extra inference cost.
- We do extensive experiments on various visual tasks, datasets, and models. PBConv consistently outperforms state-of-the-art layer-level re-parameterization methods, and is verified to be orthogonal to procedure-level and architecture-level re-parameterization method.




(a) ResNet-18:s5.c1 (b) ResNet-18:s5.c3

(c) ResNet-18 (d) VGG-16 (e) UNet

Figure 2: Overall positional importance distributions of $3 \times 3$ convolutional layers in several models. The numbers corresponding to the most important positions are bolded. (a)(b) are the distributions of the 1st and 3rd $3 \times 3$ convolutional layers separately in 5th stage of ResNet-18 pre-trained on ImageNet. (c)(d)(e) are the average positional distributions across every layer of the three convolutional models separately. In the same model((a)(b)), different positions exhibit distinct importance properties in a convolutional layer, and the same position in different layers may exhibit distinct importance as well. In different tasks and models((c)(d)(e)), such average position distributions show diverse patterns.

## 2 Related Work

### 2.1 Attention mechanism

Attention [Chen *et al.*, 2017; Hu *et al.*, 2018; Chen *et al.*, 2020; Xu *et al.*, 2015] is a popular technique that calculates an importance score for the features and influences the update of corresponding parameters. The importance score can be spatial-wise [Xu *et al.*, 2015], channel-wise [Hu *et al.*, 2018] or other mixed metrics [Chen *et al.*, 2017]. Intuitively, attention can derive informative parts by weighting the importance of input features. In contrast, our work aims to emphasize the positional importance of convolutional kernels, instead of features.

### 2.2 Re-parameterization mechanism

[Ding *et al.*, 2021c; Ding *et al.*, 2019; Guo *et al.*, 2020; Ding *et al.*, 2021d; Ding *et al.*, 2021b; Huang *et al.*, 2022; Hu *et al.*, 2022; Ding *et al.*, 2022; Ding *et al.*, 2024] belong to re-parameterization methods. These works train a large over-parameterized network and then convert the complex structure to a simple one in inference time to improve performance without increasing inference cost. [Ding *et al.*, 2021d; Ding *et al.*, 2021b; Ding *et al.*, 2022; Ding *et al.*, 2024] design a whole architecture with strong performance, while [Ding *et al.*, 2019; Ding *et al.*, 2021c] and our PBConv aim to design a generic convolution block suitable for various models to replace the ordinary convolution to improve the performance. Besides, [Huang *et al.*, 2022; Hu *et al.*, 2022; Ding *et al.*, 2023] introduce procedure-level design during training of previous re-parameterization structure [Ding *et al.*, 2021c; Ding *et al.*, 2021d]. In Experiments section, we will show our PBConv is a better generic block than previous layer-level re-parameterization methods [Ding *et al.*, 2019; Ding *et al.*, 2021c], and PBConv is orthogonal to procedure-level re-parameterization methods and architecture-level re-parameterization methods and can lead to further improvement if combined together.

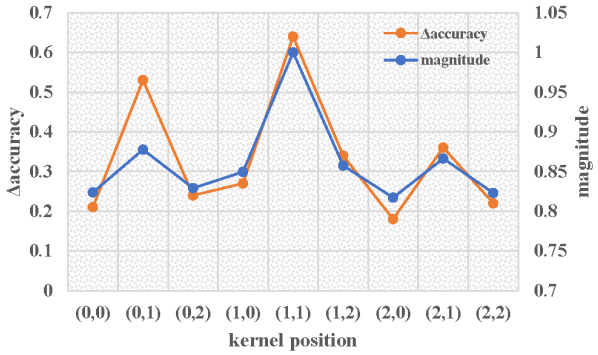



Figure 3: Relationship between position magnitude and position pruning accuracy drop. Clearly, positions with larger magnitude are more important, and pruning them causes more performance drop.

## 3 A Closer Look at Positional Distribution of Convolutional Kernels

Before demonstrating our method, we start by deeply analyzing the property of the positional distribution of the weights of convolutional kernels.

We first define the *importance* of a position in the convolutional kernel. Motivated by prior works [Han *et al.*, 2015; Li *et al.*, 2016; Ding *et al.*, 2019], we use the average magnitude of the kernel weights to describe such property. Formally, for a convolutional layer with output channel $D$, input channel $C$ and kernel size $K \times K$, we study the positional distribution of $\boldsymbol{W} \in \mathbb{R}^{D \times C \times K \times K}$ by calculating:

$$\boldsymbol{M} = \frac{\sum_{d=1}^{D} \sum_{c=1}^{D} |\boldsymbol{W}_{d,c,:,:}|}{max(\sum_{d=1}^{D} \sum_{c=1}^{D} |\boldsymbol{W}_{d,c,:,:}|)} \tag{1}$$

where $|\boldsymbol{W}|$ computes the element-wise magnitude of weight matrix $\boldsymbol{W}$. Then we can get an average magnitude matrix $\boldsymbol{M} \in \mathbb{R}^{K \times K}$ with entries $\in [0, 1]$.

We choose some widely used models and analyze the positional importance distributions. As in Figs. 2a and 2b, there are significant differences in the distribution of kernel positions between different layers of the same model. We also average $\boldsymbol{M}$ of every $3 \times 3$ convolutional layer of the model to obtain an overall inspection. Seen from Figs. 2c to 2e, different positions exhibit distinct importance properties for different models. For example, the largest region in ResNet-18 [He *et al.*, 2016] is the bottom right $2 \times 2$ square. In VGG-16 [Simonyan and Zisserman, 2014], it is the central cross while 4 corners' weights are the smallest. However, in U-Net [Ronneberger *et al.*, 2015] there is quite an opposite pattern to that in VGG-16, *i.e.*, the largest weights are located in 4 corners and center.

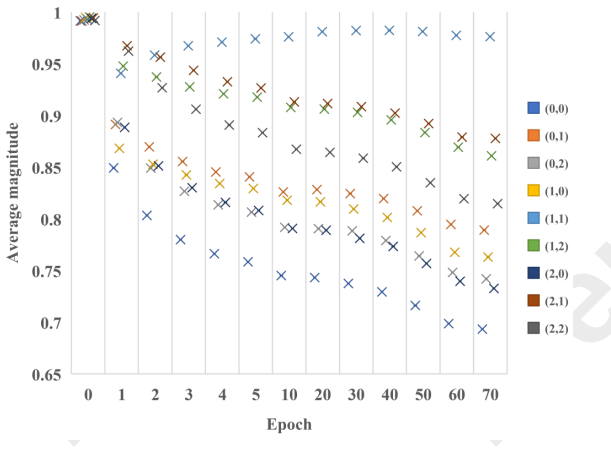



Figure 4: The variation curve of the kernel position distribution during the training of ResNet-50 on ImageNet. The distribution is roughly uniform at the beginning, and then rapidly disperses.

To investigate how such property influences performance, we prune each position of 3×3 convolutional layers of VGG-16 on CIFAR-10 [Krizhevsky *et al.*, 2009], respectively, by masking corresponding weights to zeros. The relationship between performance drop and position magnitude is shown in Fig. 3. Pruning the position with a higher magnitude will generally cause more damage. Therefore, it is reasonable to use magnitude to represent positional importance. Positions with higher magnitudes are more important.

Furthermore, we visualize the variation of the positional distribution of 3×3 convolutional kernels during the training process of ResNet-50 on ImageNet in Fig. 4. We use different colors for the nine positions in the 3×3 kernel. The distribution is roughly uniform at the beginning since the model is randomly initialized. But it rapidly disperses as the training goes on, *i.e.* the gap between important positions and unimportant positions enlarges consistently.

According to the observations above, we can conclude that the positional importance distribution in the convolutional kernel presents a large variety. It can show different features due to different visual tasks, different networks, and even different layers in the same network. The positional distribution is approximately uniform at the beginning, but rapidly disperses in the training. As the training goes on, important positions tend to be more important, while unimportant positions tend to be more unimportant.

# 4 Position Boost Convolution

## 4.1 Preliminary

Suppose a convolutional layer with weights $\boldsymbol{W} \in \mathbb{R}^{D\times C\times K\times K}$ and bias $\boldsymbol{b} \in \mathbb{R}^{D}$ takes feature map $\boldsymbol{F} \in \mathbb{R}^{C\times H\times W}$ as input, and its output is denoted by $\boldsymbol{O} \in \mathbb{R}^{D\times H'\times W'}$. We use $\boldsymbol{B} \in \mathbb{R}^{D\times H'\times W'}$ to represent replicas of $\boldsymbol{b}$ among the extra two dimensions. Formally,

$$\boldsymbol{O} = \boldsymbol{F} \circledast \boldsymbol{W} + \boldsymbol{B} \tag{2}$$

where $\circledast$ denotes convolution operation, and $+$ denotes element-wise addition. From the perspective of sliding window, the output at $(h, w)$ on $d$-th output channel is given by

$$\boldsymbol{O}_{d,h,w} = \sum_{c=1}^{C}\sum_{i=1}^{K}\sum_{j=1}^{K} \boldsymbol{S}_{c,i,j}^{h,w}\boldsymbol{W}_{d,c,i,j} + \boldsymbol{b}_d \tag{3}$$

where $\boldsymbol{S}^{h,w} \in \mathbb{R}^{C\times K\times K}$ denotes the sliding window in $\boldsymbol{F}$ corresponding to $\boldsymbol{O}_{d,h,w}$. The relationship between $\boldsymbol{S}^{h,w}$ and $\boldsymbol{F}$ depends on the convolutional layer.

**Fuse convolutional layer and BN.** In modern CNNs, there is usually a normalization layer following a convolutional layer. Batch Normalization (BN) [Ioffe and Szegedy, 2015] is one of the most popular modules, which would continuously update its running mean $\boldsymbol{\mu}$, running variance $\boldsymbol{\sigma}^2$, scaling factor $\boldsymbol{\gamma}$ and bias $\boldsymbol{\beta}$ in training. These values are fixed after training, and thus BN turns into a linear transformation in inference and we could fuse it into the convolutional layer. Particularly, for a conv-BN sequence, the output $\hat{\boldsymbol{O}} \in \mathbb{R}^{D\times H'\times W'}$ at $(h, w)$ on $d$-th output channel is given by

$$\hat{\boldsymbol{O}}_{d,h,w} = \boldsymbol{\gamma}_d \frac{\boldsymbol{O}_{d,h,w} - \boldsymbol{\mu}_d}{\boldsymbol{\sigma}_d} + \boldsymbol{\beta}_d \tag{4}$$

Therefore, we can equivalently transform a conv-BN sequence to a single convolutional layer by Eqs. (3) and (4). New $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{b}}$ for any filter $d$ is given by

$$\hat{\boldsymbol{W}}_{d,:,:,:} = \frac{\boldsymbol{\gamma}_d \boldsymbol{W}_{d,:,:,:}}{\boldsymbol{\sigma}_d}, \ \hat{\boldsymbol{b}}_d = \frac{\boldsymbol{\gamma}_d(\boldsymbol{b}_d - \boldsymbol{\mu}_d)}{\boldsymbol{\sigma}_d} + \boldsymbol{\beta}_d \tag{5}$$

Similar with $\boldsymbol{B}$, we use $\hat{\boldsymbol{B}} \in \mathbb{R}^{D\times H'\times W'}$ to denote replicas of $\hat{\boldsymbol{b}}$. Eventually, the conv-BN sequence is equivalently converted to a single convolutional layer.

$$\boldsymbol{F} \circledast \hat{\boldsymbol{W}} + \hat{\boldsymbol{B}} = BatchNorm(\boldsymbol{F} \circledast \boldsymbol{W} + \boldsymbol{B}) \tag{6}$$

## 4.2 Full coverage with parallel small kernels

For a convolutional kernel, we use multiple parallel small convolutional kernels to boost its positional distribution.

**Positional embedding for the convolutional kernel.** For a convolutional layer with weights $\boldsymbol{W}$, we aim to emphasize its informative parts and suppress less useful parts by learning an embedding vector for each position. We use $\boldsymbol{E}_{i,j} \in \mathbb{R}^{D\times C}$ to denote the embedding vector of $\boldsymbol{W}_{:,:,i,j}$, and $\widetilde{\boldsymbol{W}}$ to denote enhanced weights, which is the element-wise sum of the original weights and embedding vectors.

$$\widetilde{\boldsymbol{W}}_{:,:,i,j} = \boldsymbol{E}_{i,j} + \boldsymbol{W}_{:,:,i,j} \tag{7}$$

By replacing $\boldsymbol{W}$ with $\widetilde{\boldsymbol{W}}$, we could enhance the weights on more important parts to help aggregate local information. Finally, we acquire significant improvement without increasing any computation cost. We employ a multi-branch structure to learn such embedding vector during training and add it to the original kernel weights via the re-parameterization mechanism.

For a $K \times K$ convolutional layer, $\forall 1 \le h \le K, 1 \le w \le K, 1 \le k_h \le K - h + 1$ and $1 \le k_w \le K - w + 1$, it is feasible to adaptively learn positional embeddings for a rectangular area denoted by top left corner $(h, w)$ and bottom right corner $(h+k_h-1, w+k_w-1)$ via a concurrent convolutional branch with proper configuration. The key is making the new

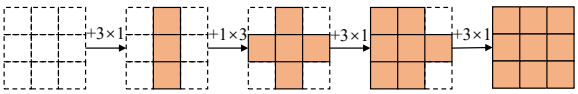



Figure 5: An example to help demonstrate the rules for candidate generation. Suppose the original kernel is 3×3, then this figure shows the only legitimate PBConv candidate if the randomly generated adding sequence is "3×1,1×3,3×1,3×1".

sliding window become a part of the original one and thus the fusion is possible due to additive and associative properties. The kernel size of the new branch should be $(k_h, k_w)$, and other settings should be the same as those of the original one, except for the padding. Note that new padding values might be negative, which means cropping. Subsequently, we can sum up their weights and biases in advance, instead of forwarding all branches and then summing up their outputs.

Furthermore, it is feasible to merge concurrent small convolutional branches into different regions of the original kernel. We use $\boldsymbol{W^0}, \boldsymbol{b^0}(\boldsymbol{B^0})$ to denote weights and biases of the original convolutional layer, and $\boldsymbol{W^l}, \boldsymbol{b^l}(\boldsymbol{B^l})$ to denote those of $l$-th extra branch. Suppose there are $n$ extra branches. After training, we can integrate all the branches into a single one with parameters $\widetilde{\boldsymbol{W}}$ and $\widetilde{\boldsymbol{B}}$ due to linear property as follows. First, $\forall\ 0 \leq l \leq n$, convert $l$-th conv-BN branch to a single convolutional layer with parameters $\hat{\boldsymbol{W}}^{\boldsymbol{l}}$ and $\hat{\boldsymbol{b}}^{\boldsymbol{l}}(\hat{\boldsymbol{B}}^{\boldsymbol{l}})$ via Eq. (5). Then $\forall 1 \leq i \leq K,\ 1 \leq j \leq K$ we have

$$\widetilde{\boldsymbol{W}}_{:,:,i,j} = \hat{\boldsymbol{W}}^{\boldsymbol{0}}{}_{:,:,i,j} + \sum_{l=1}^{n} \boldsymbol{I^l}(i,j),\ \widetilde{\boldsymbol{b}} = \sum_{l=0}^{n} \hat{\boldsymbol{b}}^{\boldsymbol{l}} \tag{8}$$

$$\boldsymbol{I^l}(i,j) = \begin{cases} \hat{\boldsymbol{W}}^{\boldsymbol{l}}{}_{:,:,i-h^l+1,j-w^l+1},\ if\ (i,j) \in \boldsymbol{\Omega^l} \\ \boldsymbol{0},\ otherwise \end{cases} \tag{9}$$

where $\boldsymbol{\Omega^l} = \{(x,y) \mid h^l \leq x \leq h^l + k_h^l - 1\ and\ w^l \leq y \leq w^l + k_w^l - 1\}$. The target positional embedding $\boldsymbol{E}$ can be easily derived: $\boldsymbol{E}_{i,j} = \sum_{l=1}^{n} \boldsymbol{I^l}(i,j)$. Eventually,

$$\boldsymbol{F} \circledast \widetilde{\boldsymbol{W}} + \widetilde{\boldsymbol{B}} = \sum_{l=0}^{n} BatchNorm^l(\boldsymbol{F} \circledast \boldsymbol{W^l} + \boldsymbol{B^l}) \tag{10}$$

An important rule for boosting positions is "full coverage", *i.e.* every region of the original kernel should be covered by at least one small kernel. Full coverage ensures PBConv can adaptively learn the embedding of every position and enhance or suppress corresponding weights. Formally, $\forall 1 \leq i \leq K, 1 \leq j \leq K, \exists 1 \leq l \leq n$, s.t. $\boldsymbol{I^l}(i,j) \neq \boldsymbol{0}$.

### 4.3 Heuristic search for optimal structure

In the previous section, we have demonstrated how to use multiple concurrent small convolutional kernels to enhance the positional distribution of the original large kernel. For a $K \times K$ convolutional kernel, each kernel smaller than $K \times K$ with proper padding can enhance a specific region of the original kernel. Due to the extremely large number of combinations of small kernels, it is not feasible to train and compare the final performance for each combination to get the best. We thus design a fast heuristic search algorithm to quickly select the optimal structure that can maximize the model's performance from a set of randomly generated candidates.

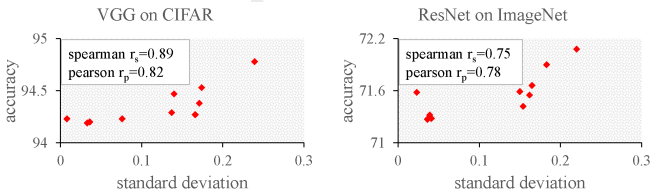



Figure 6: Correlation between standard deviation of the kernel position distribution and accuracy. Each point is a randomly generated PBConv candidate. The standard deviation is calculated after only few iterations from training, and the accuracy is the final accuracy after the whole training process. We find the final accuracy is strongly correlated with the standard deviation at the very beginning of training, indicating that we can accurately estimate the performance of each PBConv candidate within little time cost.

First, we introduce a random candidate generation algorithm. For a $K \times K$ kernel, there are $K^2 - 1$ types of small kernels in different sizes,*e.g.* 1×1, 1×2, 1×3, 2×1, 2×2, 2×3, 3×1, 3×2 are all the legal components for a 3×3 kernel. For simplicity, we randomly select $t \in \{1, 2, 3\}$ types of kernels for candidate generation, *i.e.* the possibility that each candidate has 1/2/3 types of small kernels is 33%/33%/33%. We aim to cover all the regions of the original kernel with such $t$ types of chosen kernels. We continue adding small kernels over the original kernel, until all the regions are covered.

Every time a new small kernel is added, we set its coverage area based on the following three rules. The priority of these rules decreases from the first to the last. That is to say, only when the preceding rules provide multiple areas of equal priority will we resort to the subsequent rules for priority.

1. Coverage rule: the kernel should be placed onto areas that contain more uncovered positions.
2. Center rule: the kernel should be placed across the central position of the original kernel, *e.g.*, since we consistently observe extremely large importance in the center position as shown in Fig. 2.
3. Alternative rule: the kernel should be placed from left to right, from top to bottom if the priority can not be determined by former rules.

We give an example in Fig. 5 for better clarification.

Now we have designed a complete mechanism to randomly generate some candidates. Next, we use the aforementioned algorithm to randomly generate $c$ candidates, where $c$ is set to 10 in practice. Motivated by the observation that the positional distribution of convolutional kernel rapidly disperses at the very beginning of training as in Fig. 4, we further study the relationship between standard deviation of kernel positions and model performance. In Fig. 6, the standard deviation of the positional distribution after few training iterations is strongly correlated to the final performance. Therefore, we can leverage such a heuristic search scheme to quickly select the best candidate from massive randomly generated candidates. The overall design of PBConv is in Fig. 1. The heuristic search process is very efficient and fast, typically accounting for only 1% to 10% of the total training time.

| Dataset | Model | Top-1 accuracy (%) | | | |
|---|---|---|---|---|---|
| | | Base | ACB | DBB | PBConv |
| CIFAR-10 | VGG-16 | 94.12 | 94.36 | 94.64 | **94.78** |
| ImageNet | MobileNetv2 | 71.27 | 71.60 | 71.83 | **72.26** |
| | ResNet-18 | 70.75 | 71.36 | 71.77 | **72.08** |
| | ResNet-50 | 75.79 | 76.02 | 76.07 | **76.43** |
| | ResNet-101 | 77.12 | 77.74 | 77.85 | **78.25** |

Table 1: Results for image classification.

| Rep Level | Model | Accuracy (%) |
|---|---|---|
| Architecture | RepVGG-A0 [Ding *et al.*, 2021d] | 87.03 |
| | **+PBConv** | **87.75** |
| Procedure | DyRep-VGG-16 [Huang *et al.*, 2022] | 95.22 |
| | **+PBConv** | **95.44** |
| Procedure | RepOpt-VGG-B1 [Ding *et al.*, 2023] | 90.10 |
| | **+PBConv** | **90.42** |

Table 2: Combination between PBConv and state-of-the-art architecture-level and procedure-level re-parameterization.

# 5 Experiments

## 5.1 Datasets and Competitors

We do evaluation on CIFAR [Krizhevsky *et al.*, 2009] and ImageNet [Deng *et al.*, 2009] classification, Cityscapes [Cordts *et al.*, 2016] segmentation, GoPro [Nah *et al.*, 2017] deblurring and COCO [Lin *et al.*, 2014] detection.

The layer-level reparameterization methods ACB [Ding *et al.*, 2019] and DBB [Ding *et al.*, 2021c] are the most related competitors. We first build a baseline and then replace its conv-BN sequence with ACB/DBB/PBConv, and train all models with identical configurations for a fair comparison. Notably, we also combine PBConv with newest architecture-level [Ding *et al.*, 2021d] and procedure-level [Huang *et al.*, 2022; Ding *et al.*, 2023] re-parameterization methods, and earn consistent improvements.

## 5.2 Image Classification

We first evaluate PBConv on image classification. Results are shown in Section 5.1. On CIFAR-10, PBConv improves VGG-16 by 0.66%. On the large scale ImageNet dataset, PBConv improves MobileNetv2 by 0.66%, ResNet-18 by 1.33%, ResNet-50 by 0.64% and ResNet-101 by 1.13%, significantly outperforming competitors like ACB and DBB.

Furthermore, we combine PBConv with state-of-the-art procedure-level and architecture-level re-parameterization methods. Results on CIFAR are in Section 5.2. Clearly, PBConv is orthogonal to these re-parameterization algorithms and they could be combined for further improvement.

To explore how PBConv helps CNN aggregate feature representations, we use Grad-CAM [Selvaraju *et al.*, 2017] to show the attention maps produced by ResNet-18 and PBConv-ResNet-18. In Fig. 7, the attention maps given by PBConv more precisely locate the target objects and keep the background out of attention. No matter what size and shape the target objects are, the attention given by PBConv is better confined to the semantic regions and thus more integral objects are identified in more precise locations.

## 5.3 Semantic Segmentation

Next, we experiment on pixel-level tasks. We start with high-level semantic segmentation. Models are trained with ImageNet pre-trained backbone weights on Cityscapes.

On various segmentation architectures with various backbone models, our PBConv consistently outperforms all competitors by a significant margin. Seen from Section 5.2, when using ResNet-18 as backbone, PBConv improves PSPNet by 1.50% mIoU, which is 0.29/0.67% higher than ACB/DBB, Deeplabv3 by 0.88% mIoU, which is 0.68/0.78% higher than ACB/DBB, and Deeplabv3+ by 0.57% mIoU, which is 0.49/0.63% higher than ACB/DBB. When using lightweight MobileNetv2 as the backbone, PBConv still outperforms all competitors by improving PSPNet by 0.59% mIoU, which is 0.49/0.31% higher than ACB/DBB. The advantages are still kept when we use a heavy ResNet-50 as the backbone.

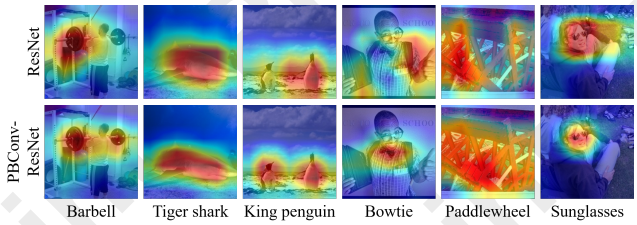



Figure 7: Attention maps of ResNet-18 generated by Grad-CAM. Clearly, our PBConv recognizes the objects with different sizes and shapes more precisely and keeps the background out of attention.

## 5.4 Image Deblurring

We then evaluate PBConv on image deblurring, where normalization is not in usual deblurring models [Nah *et al.*, 2017; Cho *et al.*, 2021]. So is ACB/DBB/PBConv here.

As in Section 5.2, PBConv improves 1.05, 0.55, and 0.59 PSNR for UNet, DeepDeblur, and MIMO-UNet respectively.

## 5.5 Object Detection

For object detection, PBConv improves mAP of [Duan *et al.*, 2019] by 1.20% on COCO, far better than others.

| Model (backbone) | mIoU (%) | | | |
|---|---|---|---|---|
| | Base | ACB | DBB | PBConv |
| PSPNet (ResNet-18) | 73.24 | 74.45 | 74.07 | **74.74** |
| Deeplabv3 (ResNet-18) | 76.12 | 76.32 | 76.22 | **77.00** |
| Deeplabv3+ (ResNet-18) | 77.07 | 77.15 | 77.01 | **77.64** |
| PSPNet (ResNet-50) | 76.93 | 77.91 | 77.60 | **78.20** |
| Deeplabv3 (ResNet-50) | 77.75 | 77.82 | 77.90 | **79.47** |
| Deeplabv3+ (ResNet-50) | 78.40 | 78.58 | 78.37 | **79.58** |
| PSPNet (MobileNetv2) | 72.84 | 72.92 | 73.02 | **73.43** |

Table 3: Results on Cityscapes for semantic segmentation.

| Model | PSNR | | | |
|---|---|---|---|---|
| | Base | ACB | DBB | PBConv |
| UNet | 25.07 | 25.74 | 25.82 | **26.12** |
| DeepDeblur | 28.29 | 28.62 | 28.63 | **28.84** |
| MIMO-UNet | 30.57 | 30.99 | 30.92 | **31.16** |

Table 4: Results on GoPro for image deblurring.

| Model (backbone) | mAP (%) | | | |
|---|---|---|---|---|
| | Base | ACB | DBB | PBConv |
| CenterNet (ResNet-18) | 29.30 | 27.30 | 29.70 | **30.50** |

Table 5: Results on COCO for object detection.

## 5.6 Analysis

**Ablation studies** In Section 5.2, we study the effect of each separate rule for PBConv generation on VGG-16 on CIFAR. Besides, we find the number of branches and parameters don't show significant influence in PBConv. PBConv uses from 85% to 130% of the parameters of DBB for different models, but consistently outperforms DBB.

**Why PBConv is effective?** We do dimension reduction by the t-SNE algorithm for the feature maps of all stages of a ResNet-18 and PBConv-ResNet-18. In Fig. 8, for low-level features (stages 1 and 2), PBConv shows clear superiority against baseline, aggregating the features with the same labels while decoupling those with different labels. For high-level features (stages 3 and 4), the gap is marginal, relatively. Therefore, the effectiveness of PBConv mainly comes from the strong low-level feature extraction ability, *e.g.*, the ability to catch edges and textures as seen from Fig. 7. As a result, PBConv is naturally more suitable for fine-grained tasks.

**Does PBConv truly enhance the positional distribution?** The equivalent kernel after training for ResNet-18 is in Fig. 9. Both layers 3-1 and 5-2 obey our assumption that PBConv indeed enlarges the gap between informative parts with larger magnitude and uninformative parts with smaller magnitude. The position distribution before final fusion is in Fig. 10. A position's value in a branch is negatively correlated with the number of times the position is covered and positively correlated with the original value at the corresponding position.

## 6 Conclusion

Motivated by the observation of positional distributions of convolutional kernels, we propose PBConv, an architecture-neutral block, to improve CNN. By learning positional embedding vectors for kernel weights, we selectively emphasize the informative parts and suppress less useful ones. We integrate small kernels in PBConv to better extract fine-grained low-level features. To find an optimal structure, we propose a heuristic search algorithm based on the correlation analysis between kernel positional distribution and performance. Extensive experiments demonstrate the effectiveness of PBConv across various visual tasks, datasets, and architectures. Moreover, PBConv brings no extra cost in inference for it could be equivalently converted to a simplest convolutional layer after training. Through controlled experiments, we show the key of PBConv is the mechanism to enhance positional weight distributions by learning embedding vectors adaptively. Such a paradigm shows fairly strong low-level feature extraction capacity. Furthermore, our PBConv is a layer-level re-parameterization design, and thus is orthogonal to architecture-level and procedure-level re-parameterization methods, *i.e.* the model performance could be further improved if we combine PBConv with them together.

| | Coverage | Center | Alternative | Accuracy |
|---|---|---|---|---|
| PBConv | ✓ | ✓ | ✓ | 94.58 |
| | ✗ | ✓ | ✓ | 94.21 |
| | ✗ | ✗ | ✓ | 94.05 |

Table 6: Ablation studies of rules for PBConv generation.

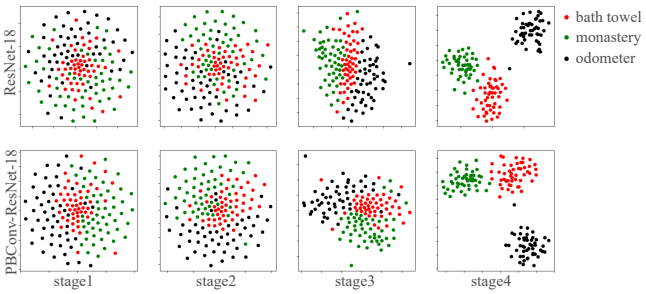
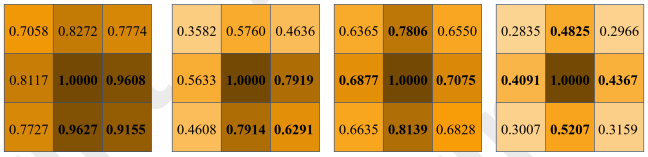



Figure 8: Visualization of different stages' output features of ResNet by t-SNE. The effectiveness of PBConv primarily stems from its enhanced ability to extract low-level features.




(a) base:3-1 (b) ours:3-1 (c) base:5-2 (d) ours:5-2

Figure 9: Positional distribution of stage3.conv1 (3-1) and stage5.conv2 (5-2) of baseline ResNet-18 and ours.

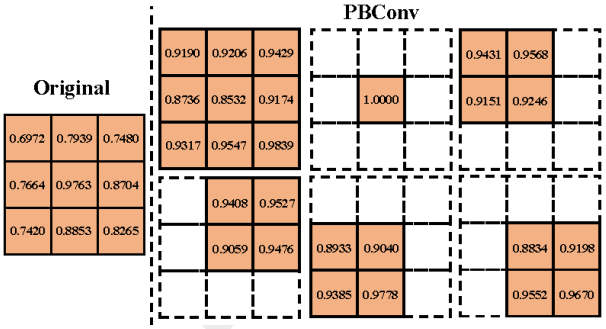



Figure 10: Weight of the PBConv for RN-18 before fusion.

## Acknowledgements

This work was supported by Pioneer and Leading Goose R&D Program of Zhejiang (No. 2024C01142), Beijing Natural Science Foundation (No. L22302) and National Natural Science Foundation of China (Nos. 62271281, 62441235, 62021002).

## References

[Chen *et al.*, 2017] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5659–5667, 2017.

[Chen *et al.*, 2020] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[Cho *et al.*, 2021] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4641–4650, 2021.

[Cordts *et al.*, 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Ding *et al.*, 2019] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1911–1920, 2019.

[Ding *et al.*, 2021a] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4510–4520, 2021.

[Ding *et al.*, 2021b] Xiaohan Ding, Chunlong Xia, Xiangyu Zhang, Xiaojie Chu, Jungong Han, and Guiguang Ding. Repmlp: re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv preprint arXiv:2105.01883*, 2021.

[Ding *et al.*, 2021c] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895, 2021.

[Ding *et al.*, 2021d] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021.

[Ding *et al.*, 2022] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022.

[Ding *et al.*, 2023] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Kaiqi Huang, Jungong Han, and Guiguang Ding. Re-parameterizing your optimizers rather than architectures. In *The Eleventh International Conference on Learning Representations*, 2023.

[Ding *et al.*, 2024] Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. Unireplknet: A universal perception large-kernel convnet for audio, video, point cloud, time-series and image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

[Duan *et al.*, 2019] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[Fu *et al.*, 2019] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[Guo *et al.*, 2020] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. Expandnets: Linear over-parameterization to train compact convolutional networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[Han *et al.*, 2015] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626*, 2015.

[Hao *et al.*, 2023a] Tianxiang Hao, Hui Chen, Yuchen Guo, and Guiguang Ding. Consolidator: Mergable adapter with group connections for visual adaptation. In *The Eleventh International Conference on Learning Representations*, 2023.

[Hao *et al.*, 2023b] Tianxiang Hao, Xiaohan Ding, Jungong Han, Yuchen Guo, and Guiguang Ding. Manipulating identical filter redundancy for efficient pruning on deep and complicated cnn. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[Hao *et al.*, 2023c] Tianxiang Hao, Mengyao Lyu, Hui Chen, Sicheng Zhao, Jungong Han, and Guiguang Ding. Reparameterized low-rank prompt: Generalize a vision-

language model within 0.5 k parameters. *arXiv preprint arXiv:2312.10813*, 2023.

[Hao *et al.*, 2024] Tianxiang Hao, Xiaohan Ding, Juexiao Feng, Yuhong Yang, Hui Chen, and Guiguang Ding. Quantized prompt for efficient generalization of vision-language models. In *European Conference on Computer Vision*, pages 54–73. Springer, 2024.

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

[Hu *et al.*, 2018] 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.

[Hu *et al.*, 2022] Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiaojin Gong, and Xian-Sheng Hua. Online convolutional re-parameterization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 568–577, 2022.

[Huang *et al.*, 2022] Tao Huang, Shan You, Bohan Zhang, Yuxuan Du, Fei Wang, Chen Qian, and Chang Xu. Dyrep: bootstrapping training with dynamic re-parameterization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 588–597, 2022.

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

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Krizhevsky *et al.*, 2012] 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.

[Li *et al.*, 2016] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[Nah *et al.*, 2017] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891, 2017.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[Selvaraju *et al.*, 2017] 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.

[Shen *et al.*, 2024] Leqi Shen, Tianxiang Hao, Tao He, Sicheng Zhao, Yifeng Zhang, Pengzhang Liu, Yongjun Bao, and Guiguang Ding. Tempme: Video temporal token merging for efficient text-video retrieval. *arXiv preprint arXiv:2409.01156*, 2024.

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

[Toshev and Szegedy, 2014] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[Wang *et al.*, 2024a] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Repvit: Revisiting mobile cnn from vit perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15909–15920, 2024.

[Wang *et al.*, 2024b] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 37:107984–108011, 2024.

[Xiong *et al.*, 2024] Yizhe Xiong, Hui Chen, Tianxiang Hao, Zijia Lin, Jungong Han, Yuesong Zhang, Guoxin Wang, Yongjun Bao, and Guiguang Ding. Pyra: Parallel yielding re-activation for training-inference efficient task adaptation. *arXiv preprint arXiv:2403.09192*, 2024.

[Xu *et al.*, 2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[Zhong *et al.*, 2020] Zilong Zhong, Zhong Qiu Lin, Rene Bidart, Xiaodan Hu, Ibrahim Ben Daya, Zhifeng Li, Wei-Shi Zheng, Jonathan Li, and Alexander Wong. Squeeze-and-attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13065–13074, 2020.