Journal of Computers Vol. 36 No. 5, October 2025, pp. 39-57
https://doi.org/10.63367/19911599202510360500 3

# PPENet: For Parameterized Position Encoding Learning in Image Recognition

Xiang Li, Bin Dai*, and Lin Li

Department of Computer and information engineering, Xiamen University of Technology, Xiamen, Fujian

872769541@qq.com, bdai@xmut.edu.cn

*Received 23 September 2024; Revised 26 December 2024; Accepted 25 January 2025*

**Abstract.** In the research of lightweight models for mobile or resource-constrained devices, it has been found that convolutional neural networks still exhibit certain advantages over Transformers in terms of performance and application conditions. We propose a lightweight network architecture design incorporating variable convolutional kernels and parameterized position encoding. This is a reconstruction approach that integrates the characteristics and parameters of position information encoding in Transformers into convolutional neural networks, further enhancing the overall performance of the network. Specifically, during the training process of the model, effective correlations were established between the feature information obtained through convolution and the positional feature information from convolutions with different kernels. This enhancement allows convolutional processes to gain improved positional awareness of feature information compared to previous methods. We have integrated the significant advantages of other models to create a plug-and-play module. Experimental results demonstrate that, on traditional computer vision image datasets, our proposed lightweight model architecture design exhibits superior inference speed and overall performance compared to other CNN-based lightweight network models under similar parameter conditions. Scientific experimental results from analysis in image classification and semantic segmentation domains show significant improvements. For instance, compared to MobileNet series, EfficientNet, and FasterNet on traditional public datasets, our proposed model achieves the highest classification accuracy improvement of 7.2% and enhances training speed. Compared to segmentation models like Unet, LinkNet, and SegNet on segmentation datasets, our proposed PPENet achieves the highest mIOU performance improvement of 20.48% while drastically reducing parameter count.

**Keywords:** parameterized position encodings, lightweights, information position awareness, computer vision

## 1 Introduction

In the field of computer vision [1], with the rapid development and application of Transformer models [2]. They have shown excellent performance in various computer vision recognition tasks. such as image classification, image segmentation, and object detection, compared to other types of neural networks [3-5]. Therefore, many researchers consider it the best alternative algorithm to replace neural networks. However, while Transformer models improve performance, they also introduce a large number of parameters. This leads to issues such as high latency, high computational cost, and difficulty in training. As a result, their application on mobile devices is limited. If the adaptability of mobile devices is improved by reducing the size of the model, the advantages of such performance are often not as good as those of lightweight neural networks. It is undeniable that in model applications on mobile devices, ConvNets offer better hardware support, are easier to train, and possess the ability to learn discriminative visual re-presentations [6]. Researchers have also developed re-parameterization methods to improve the performance of ConvNet models. For example, the application of the RepVGG [5] re-parameterization technique has significantly improved the performance of VGGNet. However, research has found that the RepVGG re-parameterization method overlooks the control of parameters and FLOPs. As a result, its perfor-

* Corresponding Author

mance on mobile devices is inferior to that of lightweight networks such as the MobileNet series (V1, V2, V3) [7-9].

Therefore, we believe that the advantages of Transformers (ViTs) and ConvNets are both crucial and indispensable in the research of re-parameterization algorithms. The main focus of this research is to combine the advantages of Transformers and ConvNets to develop a re-parameterization method suitable for mobile devices, aiming to improve the adaptability of lightweight networks and ultimately enhance overall performance.

In this paper, we propose a position encoding re-parameterization method that enhances the sensitivity to positional information [10]. Specifically, the proposed position encoding re-parameterization consists of two main components: the initial weight update and the position encoding. This approach addresses, to some extent, the limitations of re-parameterization for mobile device applications.

The main idea of the position encoding re-parameterization method we propose is as follows: First, position information parameters are generated by position encoding processing. Then use the basic original weight $W_o$ processing to generate the transformation into $W_b$. Finally, the $W_b$ processed feature information parameters and position coding information are integrated for computational reasoning. The final results show that the proposed model achieves better performance in lightweight models while maintaining the same inference cost. For example, compared to models like EfficientNet [11] and FasterNet [12]. the accuracy improved by up to 7.2%, and the training loss was also reduced. In the field of semantic segmentation, the proposed PPENet model achieved a segmentation performance of 79.82% with fewer than 1 million parameters. Additionally, the control over parameters and FLOPs has also shown favorable results.

Our contributions are summarized as follows:

1. We propose a position encoding re-parameterization method. By establishing connections with the parameters of existing models, it enhances the image representation capability. As a parameterized form, position encoding information can capture complex features from different locations, thereby improving the model's performance.
2. The proposed PPE block can be used as a replacement for convolution or embedded within existing model architectures. Experimental results verify that it improves performance without a significant increase in parameters or computational cost. Additionally, it can be applied to tasks such as semantic segmentation.
3. We have confirmed that PPE Block can reduce the depth of the model while ensuring the expression of model features

## 2 Related Work

### 2.1 The High-performance Architectural Design Provides Versatile Application Methods

In the development process of CNN, in order to adapt the neural network to the inference impairment on different hardware, many researchers have proposed visual neural networks (CNNs) [13-15] for various mobile devices. Various architectural designs [16]. For example, in the design of CNN-based model structures, there are comprehensive designs similar to ResNet, SKNet [17], and others, as well as module structures like SEBlock [18], ECABblock [19], which enhance model capabilities by embedding into existing models [20]. However, many current network model designs focus solely on creating superior structures or studying modules/plugins that enhance performance. It is crucial for current model research to explore a structure that combines the advantages of both aspects. Through comparing the method of parameterizing positional encoding with variable convolutions in this study, we emphasize how to incorporate additional positional encoding information into image features during model execution. This approach aims to enhance the performance of the model structure. In summary, the method of parameterizing positional encoding with variable convolutions not only enhances the representation capability of image features without altering the original model structure but also independently constructs a model with positional information in image features. This approach can be applied to various computer vision tasks such as image classification [21], semantic segmentation [22], and others.

### 2.2 Re-parameteriation Universe

Studies in structural reparameterization [23-26] have shown that multi-branch structures generally outperform single-branch structures, while single-branch structures are more advantageous for deployment purposes. The

reparameterization approach proposed by Ding X et al. [5], such as REPVGG, has significantly enhanced the performance of traditional VGGNet [14] networks. Experimental results have demonstrated the feasibility of using a multi-branch structure for identity mapping during training and a single-branch structure for inference. However, reparameterization experiments have shown that networks operating in a multi-branch structure during training require more memory and training time. On the other hand, the parameterized position encoding method of variable convolution can effectively expand the receptive field to a certain extent. It operates by convolving with original weights during training and enriching feature maps with positional information through position encoding. Empirical evidence confirms its ability to enhance image representation while maintaining computational efficiency.

### 2.3 Lightweight on Mobile Devices

With the rapid advancement of technology and the widespread application of mobile devices, there is a growing demand for equipping them with suitable capabilities. Over time, convolutional neural network models have gained significant attention for deployment on mobile devices. Extensive efforts by researchers have led to the development of numerous lightweight convolutional neural networks such as ShuffleNets [27], MobileNets, EfficientNet, ESPNetv2 [28], and others. To achieve a more balanced trade-off between accuracy and efficiency in lightweight models, ShiftNet [29] has been developed, which introduces shift operations interleaved with point-wise convolutions to replace spatial convolutions. Models like ShuffleNet, MobileNet series, and others offer advantages such as fewer parameters, lower computational costs, and faster inference speeds, making them highly effective on mobile devices.

Despite the considerable advantages of existing lightweight models, there remains a significant gap between these models and their deeper counterparts. Moreover, many studies have focused only on feature maps [30-32], with subsequent research outcomes falling short of expectations. In recent years, researchers have explored a series of studies combining Transformer and CNN technologies. Models that integrate Transformer-based techniques into CNN architectures have demonstrated advantages in local feature extraction and global information capture [33, 34]. Our primary research goal is to develop a model that is better adapted for mobile devices and enhances operational performance. Our work extensively considers research focuses such as enlarging receptive fields, parameterization, and positional encoding of image features.

### 2.4 Applications of Encoders

Encoders are widely used in machine learning and deep learning for data dimensionality reduction, feature extraction, denoising, and various other applications. Although stacked autoencoders are a more fundamental model in deep learning, this encoder obtains different types of input representation signals by stacking multiple layers of autoencoding networks. The features extracted in this way can help to replicate the input signals as closely as possible during the model training process, thereby improving the model's performance to some extent.

Various experimental studies have demonstrated that convolutional neural networks (CNNs) possess strong feature extraction capabilities. Will further integrating CNN algorithms with autoencoder algorithms lead to better performance in fields such as image processing and computer vision? Our research shows that incorporating encoding information into CNNs not only enhances performance to some extent but also aligns better with the design of lightweight models, making them very friendly for deployment on mobile devices.

### 2.5 Lightweight Model and Transformers

In recent years, with the rapid development of artificial intelligence, more and more applications on mobile devices are increasingly relying on lightweight models. Research on lightweight models has become very important. Currently, many lightweight models have been designed, such as MobileNets, EfficientNet, and others. Due to their excellent compatibility with mobile devices, these models have been widely applied in mobile devices.

Although lightweight models have many advantages, there is still a significant performance gap compared to deep models. The application of parameterization methods in lightweight models is bound to bring better performance improvements. However, the parameterization method of RepVGG has unsatisfactory results in lightweight applications. Because RepVGG ignores the impact of parameters and FLops on lightweight models.

Therefore, designing a parameterization method that is better suited for lightweight models is the focus of this research.

The proposed position encoding parameterization method combines the advantages of ConvNets and Transformers, making it more mobile-device-friendly. Experiments embedded in lightweight models also verify that it has almost no impact on the number of parameters and FLOPs.

The design of the algorithm we proposed differs from other parameterization methods as follows: 1. Different architecture and approach: For example, RepVGG primarily merges multiple different types of convolutional layers into an efficient structure during training, whereas our design integrates position encoding with weight variation to form a fused structure. 2. Different problems addressed: The main goal of this design is to focus on the application of parameterization on mobile devices, reducing the impact of parameters and FLOPs, making position encoding parameterization more advantageous in lightweight models.

## 3 The Buildings Related Work

In this chapter, we will provide a detailed exposition on the design of the structure involving variable convolutional positional encoding parameterization, as well as the method of replacing convolutions in different models.

### 3.1 Model Architecture

**PPE Architecture (PPE Block).** In the initial design of the Positional Encoding Parameterization (PPE Block) module, we drew inspiration from the RefConv [20] technique for parameterization. However, we found that integrating the RefConv approach into positional encoding led to a sharp increase in parameters and did not effectively complement global information loss when processing local features through convolution operations. To enhance the model's representation capability of input data and improve its ability to capture spatial structure and positional information, we redesigned the parameterization of positional encoding information. Here, we denote the base weights as Wo and the feature information as map. Initially, the feature map is processed in two different ways. In the first approach, the tensor map is split along dimension 1 to generate Xw and Xh. Operations are then performed using Xw and Xh tensors to derive positional information Pw for height direction and Ph for width direction. Leveraging Pw and Ph, which can operate on features within the same row and column, enables comprehensive feature extraction while incorporating positional information into the input features. In the second approach, we utilize the base weights Wo to apply a convolutional operation on the original feature map using different kernels, aiming to refocus and retain the original information in the features. This step addresses the potential loss of information during convolutional processing of local features.

Ultimately, the positional information-enriched feature information from the first approach and the re-focused original feature information obtained through the second approach are combined by addition. This merging process results in feature information that incorporates spatial structure and positional cues, thereby enhancing the model's performance. As shown in Fig. 1:

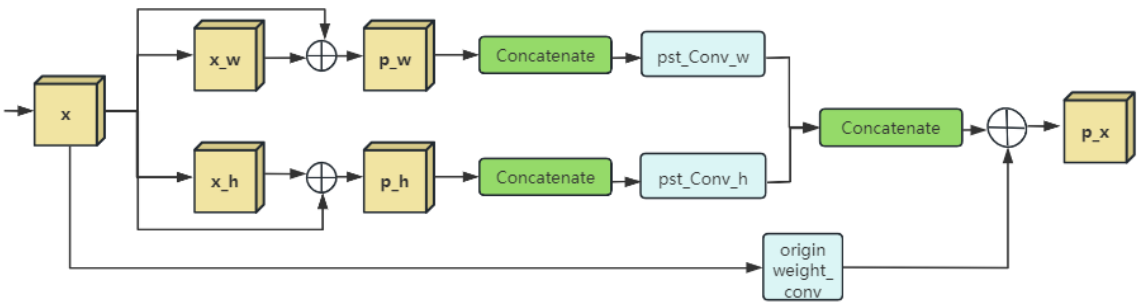


**Fig. 1.** Position coding parameterized module diagram

**Random Convolution Kernel Architecture (RK Block).** The concept of receptive field in CNN models is frequently discussed because it determines how much input information neurons perceive, and it scales proportionally with the size of the receptive field. However, increasing the receptive field size also results in significantly higher computational requirements and parameter counts. How can we address the challenges brought by increasing the receptive field while maximizing its benefits and avoiding associated issues? Our designed random convolutional kernel module offers solutions from multiple perspectives. Firstly, it departs from the traditional approach of fixed convolutional kernels by utilizing randomly variable convolutional kernels. This approach not only enhances the diversity of the network in lightweight models, making it more flexible, but also improves model generalization and robustness while effectively mitigating the risk of overfitting. Secondly, we have designed a convolution scheme that combines depth-wise separable convolutions with point-wise convolutions. Lastly, by integrating with the positional encoding parameterization module, our overall module design ensures seamless integration and embedding flexibility at any time and place. As shown in Fig. 2:

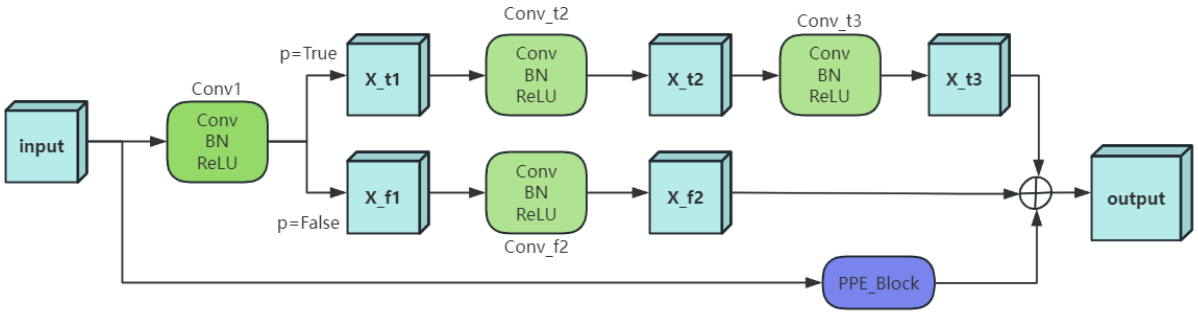


**Fig. 2.** Random convolutional Kernel module

As shown in Fig. 2, initially, we set the kernel_size dimension to 3, and define the range of random value n in the random pool from 0 to 3. The randomly generated kernel_size is 3+2×n. To ensure that the corresponding feature undergoes convolutional size changes, the padding value is set to 1+n. In Conv1, using the variable parameter α, we generate the hidden layer hidden_channel. It performs an operation where input in_Channel results inhidden_channel=in_Channel×α, with strides=1. Then, based on the given condition, different operations are executed: When p=True, this method is referred to as the parameterized large-size version. Conv_t2 executes depth-wise separable convolution with input and output channels both set to hidden_channel, with strides=1. Conv_t3 performs point-wise convolution. Finally, the output is combined with the PPE_Block (Positional Encoding Parameterization). When P = False, it represents the parameterized small-size version. In this version, to reduce parameters and improve computational efficiency, Conv_t2 executes depth-wise separable convolution directly with hidden_channel and in_Channel through point-wise convolution. Subsequent steps are similar to those in the P = True scenario.

**RKP Block Convolution.** Here we integrate PPE block and RK block together, which is called RKP block convolution.

*Receptive field.* To better cater to varying demands in image information, enhance feature diversity, increase non-linearity, and extend receptive fields to some extent, models need the capability to adaptively generate randomized convolutional kernels within defined constraints. Therefore, utilizing the pattern of large randomized convolution kernels, we construct a network capable of achieving a larger receptive field to some extent. Specifically, if we set the repetition depth of the PPE Block in the network structure to i, with the base PPE Block having a kernel size k, dilation rate a ($a \in (0,3)$), and receptive RF, then the relationship between the receptive field size and the repetition count i of the PPE Block is given by:

$$K = 3, K_0 = K, K_i = K_0 \times a, a \in (0,3), K_0 \le K_i. \tag{1}$$

$$RF_0 = K_0, RF_i = RF_{i-1} + K_i . \quad (2)$$

Because DBCBlock adopts a simple convolutional form and does not perform straightforward receptive field calculations, using RFDBC for representation, the overall relationship formula for receptive fields in the model is:

$$RF = RF_{DBC\_n} + \sum_{i=0}^{n-1} RF_{DBC\_i} + RF_i. \quad (3)$$

The choice of base convolution kernels and dilation rates can enhance the representation capability of images to a certain extent, allowing for a larger expansion of the receptive field. We have discarded the conventional fixed approach for selecting dilation rates and instead opted for a random method. Here, the dilation rates are chosen from within a fixed range, aiming to prevent information loss due to excessively large convolution kernels and to maintain the model's ability to understand global features.

*Collection of PPE characteristics.* In convolutional neural network (ConvNets) models, the gathering of features can be described as

$$y_j = \sum_{i \in d(j)} W_{j-i} - x_i. \quad (4)$$

Here, $x_i$, $y_j$ represent the input and output positions at j. d(j) denotes the local neighborhood of j. Through the formula, it becomes clear that ConvNets can only gather information within a local receptive field. Our aim is to help ConvNets overcome these limitations. To fully demonstrate the value of our research, here is a detailed introduction to the algorithm proposed in this study. The PPE (Preserve, Propagate, and Expand) that we propose primarily involves gathering relevant feature information from both horizontal and vertical directions, followed by fusion operations as part of the parameterization process.

Here's a detailed description of the algorithm focusing on the vertical direction: In the process of parameterizing positional encoding information, we experimented with various methods to extract positional encoding that aligns closely with the design intent of our algorithm and enhances image representation capabilities. During our investigation into data interpolation methods, we found that traditional methods like nearest-neighbor interpolation, bilinear interpolation, and radial basis function interpolation each have their strengths and weaknesses. We continuously compared these methods in practical scenarios and through experimentation.

Ultimately, we decided to handle tensor dimensions and positional encoding separately. We first processed the positional encoding tensor information and then fused it through convolutional operations to generate new features. Subsequently, we concatenated and activated these features.

The overall formula for the parameterized feature collection in this study can be represented as:

Here, $x_i$, $y_j$ represent the input and output positions at j. d(j) denotes the local neighborhood of j. Through the formula, it becomes clear that ConvNets can only gather information within a local receptive field. Our aim is to help ConvNets overcome these limitations. To fully demonstrate the value of our research, here is a detailed introduction to the algorithm proposed in this study. The PPE (Preserve, Propagate, and Expand) that we propose primarily involves gathering relevant feature information from both horizontal and vertical directions, followed by fusion operations as part of the parameterization process.

Here's a detailed description of the algorithm focusing on the vertical direction: In the process of parameterizing positional encoding information, we experimented with various methods to extract positional encoding that aligns closely with the design intent of our algorithm and enhances image representation capabilities. During our investigation into data interpolation methods, we found that traditional methods like nearest-neighbor interpolation, bilinear interpolation, and radial basis function interpolation each have their strengths and weaknesses. We continuously compared these methods in practical scenarios and through experimentation.

Ultimately, we decided to handle tensor dimensions and positional encoding separately. We first processed the positional encoding tensor information and then fused it through convolutional operations to generate new features. Subsequently, we concatenated and activated these features.

The overall formula for the parameterized feature collection in this study can be represented as:

$$F(x) = \varphi(x) + \lambda(x). \tag{5}$$

In ConvNets, $\varphi$(x) typically represents the traditional feature computation method, which involves extracting features from input data using convolutional layers and pooling operations.

$$\varphi(x) = \sum_{i \in d(j)} W_{j-i} - x_i. \tag{6}$$

λ(x) represents the process of parameterizing relative positional encoding. It enhances feature maps by utilizing relative offsets in height and width dimensions. This approach not only leverages the flexibility and generality of relative positional encoding but also operates efficiently across inputs of different sizes without requiring recalibration. Since pixel positions in images carry significant semantic meaning in image processing, a directional mapping method is adopted. The encoding formula is as follows:

$$R_{ij} = P^{\bar{x}}_{I^{\tilde{x}}(i,j)} + P^{\bar{y}}_{I^{\tilde{y}}(i,j)}. \tag{7}$$

$$I^{\tilde{x}}(i,j) = g(\tilde{x}_i - \tilde{x}_j). \tag{8}$$

$$I^{\tilde{y}}(i,j) = g(\tilde{y}_i - \tilde{y}_j). \tag{9}$$

So, λ(x) primarily applies a method of feature enhancement based on positional encoding and convolution to process feature information, rather than traditional interpolation methods. Next, we will explain the positional encoding in the vertical and horizontal directions.

Let's assume the defined ph (representing the vertical direction) is a tensor of shape (1, height_h, width_h), where its elements are computed based on positional encoding in the vertical direction. At position (i, j), the calculation formula for vertical positional encoding is:

$$P_h(i,j) = X_h(i,j) + h. \tag{10}$$

Where $P_h$(i,j) represents the value of $P_h$ at height i and width j. $X_h$(i,j) denotes the value of input tensor $X_h$ at height i and width j. h represents the height index, which is directly added to the value of the input tensor when computing the positional encoding, thereby introducing positional information.

Similarly, in the width direction, $p_w$ is a tensor of shape (1, height_w, width_w), where its elements are computed based on positional encoding in the width direction. At position (i, j), the calculation formula for positional encoding is:

$$P_w(i,j) = X_w(i,j) + w. \tag{11}$$

Where $P_w$(i,j) represents the value of $p_w$ at height i and width j. $X_w$(i,j) denotes the value of input tensor $X_w$ at height i and width j. w is the width index, which is directly added to the input tensor's value when computing the positional encoding, thereby introducing positional information in the width direction.

By adding the indices of height and width directly to the corresponding positions of the input tensor, the positional encoding tensors $p_h$ and $p_w$ introduce positional information for each element. This approach enables convolution operations to capture not only local features but also perceive global positional information, thereby enhancing the model's expressive power.

# 4 Experiments

## 4.1 Performance Evaluation on Traditional Datasets

The selection of datasets and model comparison. Firstly, based on the existing equipment conditions, we selected datasets such as flowerdata and skindata for extensive experiments. These experiments aim to demonstrate the effectiveness of the proposed Positional Encoding Parameterization in improving the representational capacity and performance of Convolutional Neural Networks (CNNs). Based on existing device capabilities, we conducted extensive experiments primarily using datasets such as flower datasets and skin datasets. Flower datasets and skin datasets are traditional datasets from different domains. Conducting experiments on these two distinct datasets helps to demonstrate the rigor and scientific validity of the experiments. Flowerdata consists of five categories with 3306 training samples and 364 validation samples. Skindata, on the other hand, is a dataset for classifying skin diseases in the medical field, comprising seven categories and containing 10016 skin images. Subsequently, various lightweight CNN models that are significant and representative were compared and tested. These models include MobileNetV1, V2, V3, ShuffleNe, FasterNet, and VGGNet16.

Experimental configuration. In terms of the matching value of the experiment. We uniformly use the batch size to train a total of 150 epoch. The initial value of the learning rate is 0.25, which is dynamically adjusted according to every 15 epoch. The version of the video card used for training is a tesla M20. The input image resolution is 224 × 224. This contrast experiment is carried out according to two ways. The first is the performance comparison experiment of our self-designed PPE model structure. The second is to insert our PPE Block module for experimental comparison without changing the orginal model architecture.

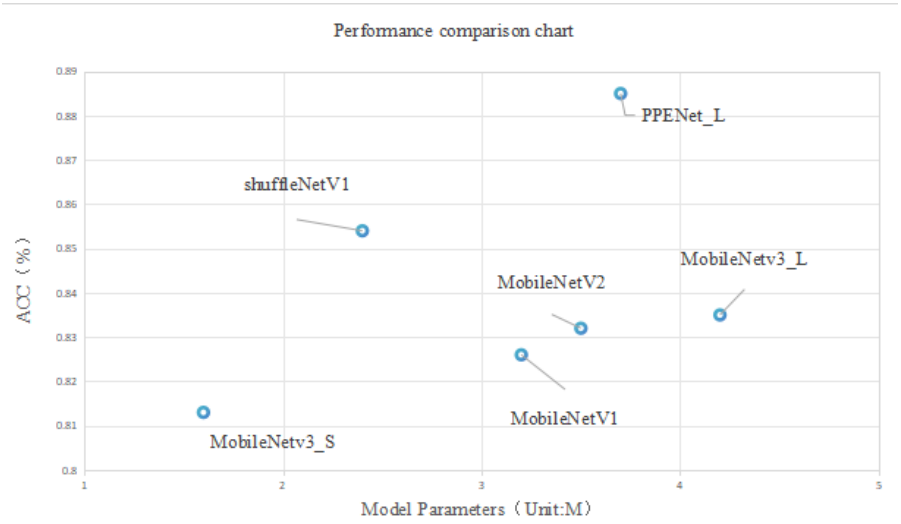


**Fig. 3.** Image classification performance comparison chart

Experimental Results 1. The experimental results for image classification are shown in Fig. 3 and Table 1. Fig. 3 Clearly, the performance of the PPENet_L model is superior to other models. In Table 1, more model metrics are compared, highlighting that our proposed PPENet model achieves performance improvement while using fewer convolutional layers and maintaining comparable parameter count and runtime. For instance, compared to the MobileNet series (V1, V2, V3), PPENet_s demonstrates the best experimental results by reducing parameter count by 42.8% while achieving 85.4% accuracy. Compared to MobileNetV3_s and MobileNetV3_L, it improves accuracy by 4.1% and 1.2%, respectively. Detailed data comparison is shown in Fig. 3, Fig. 4, Fig. 5 and Table 1.

Experimental Classification Results on the FlowerData dataset. Fig. 3 Comparison based solely on accuracy and parameter count. To maintain clarity, the comparison focused on selected models: Table 1 Comprehensive

evaluation of model performance from multiple perspectives. The models evaluated in this assessment are based on the experimental results obtained in our study.

**Table 1.** Classification performance metrics table

| Model | Top-1 | Low-1 | Param | s/epoch |
|---|---|---|---|---|
| MobileNetV1 | 0.826 | 0.241 | 3.2M | 26 s |
| MobileNetV2 | 0.832 | 0.207 | 3.5M | 30 s |
| MobileNetV3-s | 0.813 | 0.239 | 1.6M | 21 s |
| MobileNetV3-l | 0.835 | 0.179 | 4.2M | 27s |
| ShufflenetV1 | 0.856 | 0.149 | 0.85M | 37s |
| FasterNet | 0.860 | 0.287 | 13.7M | 41s |
| EfficientNet_b0 | *0.857* | 0.187 | 4.01M | 41s |
| **PPENet-s** | ***0.854*** | **0.281** | **2.4M** | **21s** |
| **PPENet-l** | ***0.885*** | **0.166** | **3.7M** | **50s** |

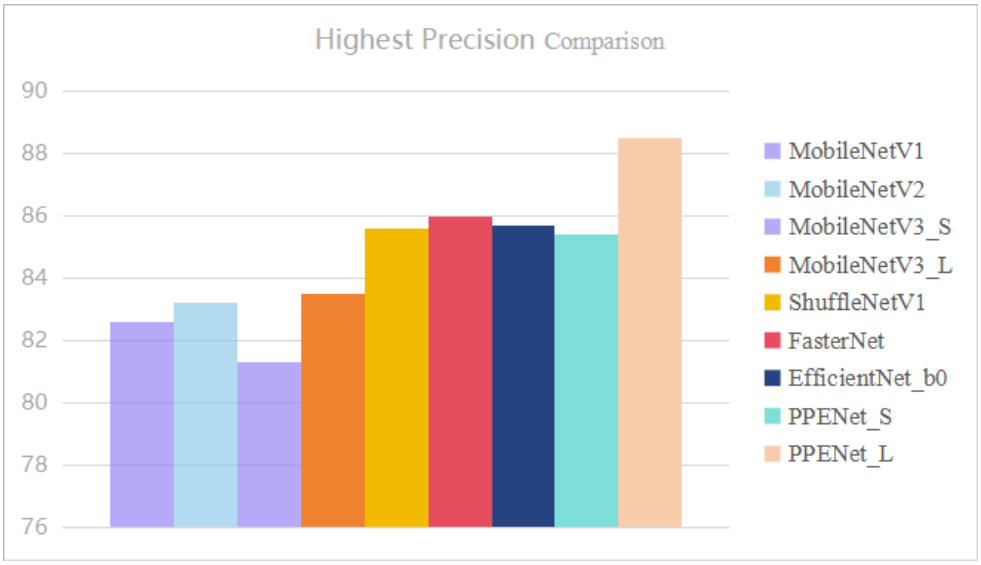


**Fig. 4.** Comparison of training accuracy for image classification

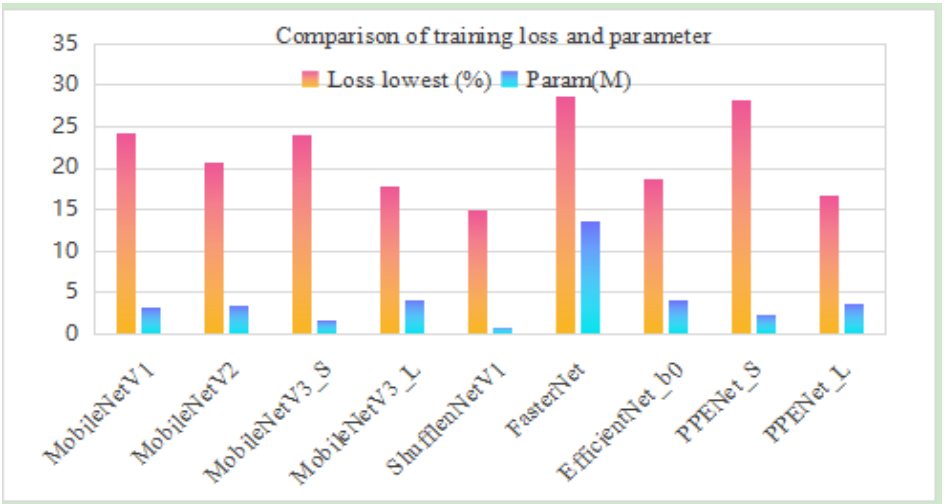


**Fig. 5.** Comparison of training loss and parameters in image classification

In order to better show the structure of the model, the corresponding complete model is given here. The structure of the model is mainly composed of DBCBlock and RKBlock, in which the function of DBCBlock is to reduce the size of the image and increase the dimension. PPEBlock is included in RKBlock, and its main function is to provide different convolutional cores and receive position coding parameterized information to realize the fusion of global information.

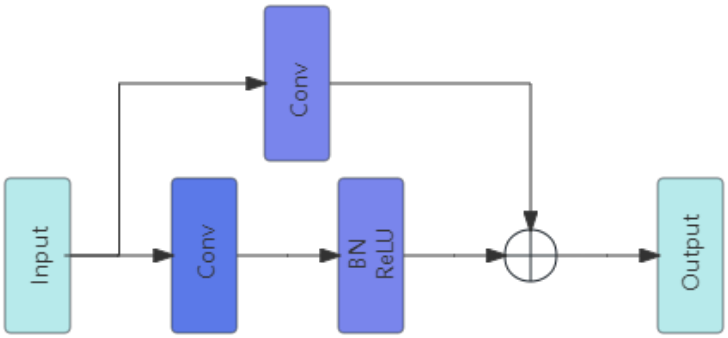


**Fig. 6.** DBCBlock description

The structure of Fig. 6 DBCBlock is very simple, mainly using residual structure and point-by-point convolution. Make a preliminary processing of the image. Then the processed image features are transferred to the RKBlock module for processing. The detailed model structure is shown in Table 2.

**Table. 2.** Chematic diagram of the PPENet model

| PPENet_L | | | | | |
|---|---|---|---|---|---|
| Block | In_channel | Out_channel | Kernel | Stride | Judge |
| DBC Block | 3 | 32 | 1 | 1 | False |
| RKP Block | 32 | 32 | | 1 | False |
| DBC Block | 32 | 48 | 1 | 1 | False |
| RKP Block | 48 | 48 | | 1 | False |
| DBC Block | 48 | 72 | 1 | 1 | True |
| RKP Block | 72 | 72 | | 1 | False |
| DBC Block | 72 | 96 | 1 | 1 | True |
| RKP Block | 96 | 96 | | 1 | True |
| DBC Block | 96 | 156 | 1 | 1 | True |
| RKP Block | 156 | 156 | | 1 | True |
| DBC Block | 196 | 196 | 1 | 1 | True |
| PPENet_S | | | | | |
| Block | In_channel | Out_channel | Kernel | Stride | Judge |
| DBC Block | 3 | 32 | 3 | 2 | False |
| RKP Block | 32 | 32 | | 1 | False |
| DBC Block | 32 | 72 | 1 | 1 | False |
| RKP Block | 48 | 48 | | 1 | True |
| DBC Block | 72 | 96 | 1 | 1 | False |
| RKP Block | 96 | 96 | | 1 | True |
| DBC Block | 96 | 128 | 1 | 1 | True |

Experimental Results 2. The performance comparison results based on integrating the PPEBlock module into the models are shown in Fig. 7, Fig. 8 and Table 3.

Fig. 7, Fig. 8 Clearly demonstrates that integrating the PPEBlock module with the models results in improved accuracy, effectively validating the efficacy of the proposed PPEBlock module.

Table 3 Comprehensive performance evaluation from multiple aspects illustrates that integrating the PPEBlock module leads to improved accuracy and effective reduction in training loss, while either maintaining or marginally increasing parameter and computational complexity. The models used in this comparative experiment are classic CNN architectures such as ResNet34, MobileNetv3, among others. The experimental results demonstrate that our designed PPE_Block can enhance overall model performance to a certain extent.

The perimental comparison of PPEBlock module insertion is carried out on the use of FlowerData datasets. (a) the accuracy is compared by base model and model+PPEBlock experiments. (b) all the data and information are obtained through our experiments, which ensures that the experimental data can prove the superiority of the model under the same equipment and training parameters.

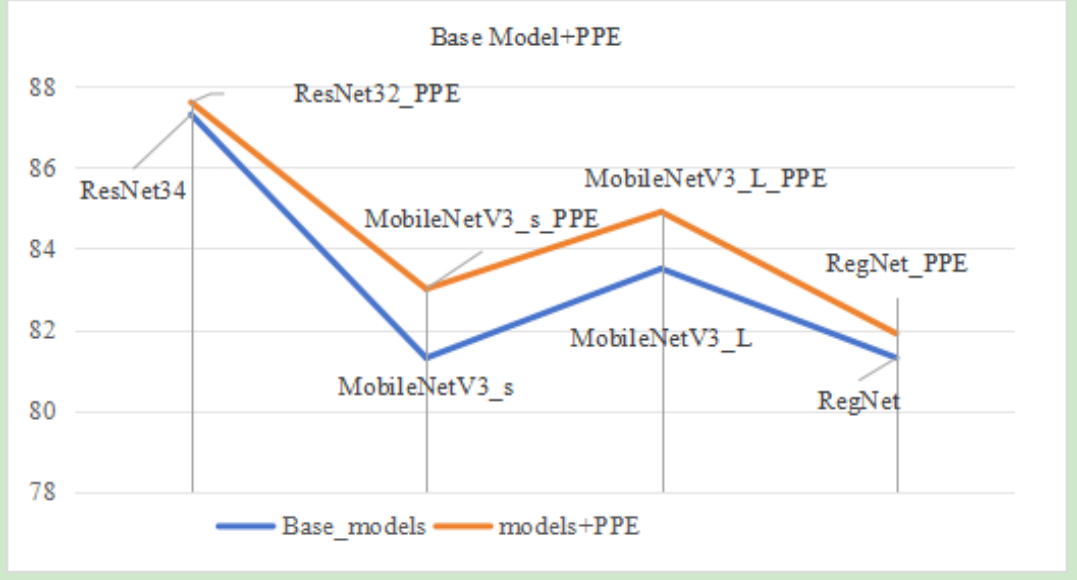


**Fig. 7.** Performance comparison of inserted image classification

**Table 3.** Performance comparison table of different models

| Heading level | Models | Top-1 ACC | Low-1 loss | Param | Flpos |
|---|---|---|---|---|---|
| 1 | ResNet34 | 87.3% | 16.0% | 21.2M | 3.67G |
| 2 | **ResNet34+PPE** | **87.6%** | **15.0%** | **21.2M** | **3.68G** |
| 3 | MobileNetV3_S | 81.3% | 23.9% | 1.66M | 0.06G |
| 4 | **MobileNetV3_S+PPE** | **83.0%** | **21.8%** | **1.67M** | **0.06G** |
| 5 | MobileNetV3_L | 83.5% | 17.9% | 4.2M | 0.24G |
| 6 | **MobileNetV3_L+PPE** | **84.9%** | **17.5%** | **4.2M** | **0.24G** |
| 7 | *RegNet* | 81.3% | 27.5% | 2.7M | 0.21G |
| 8 | ***RegNet+PPE*** | **81.9%** | **24.3%** | **2.3M** | **0.21G** |

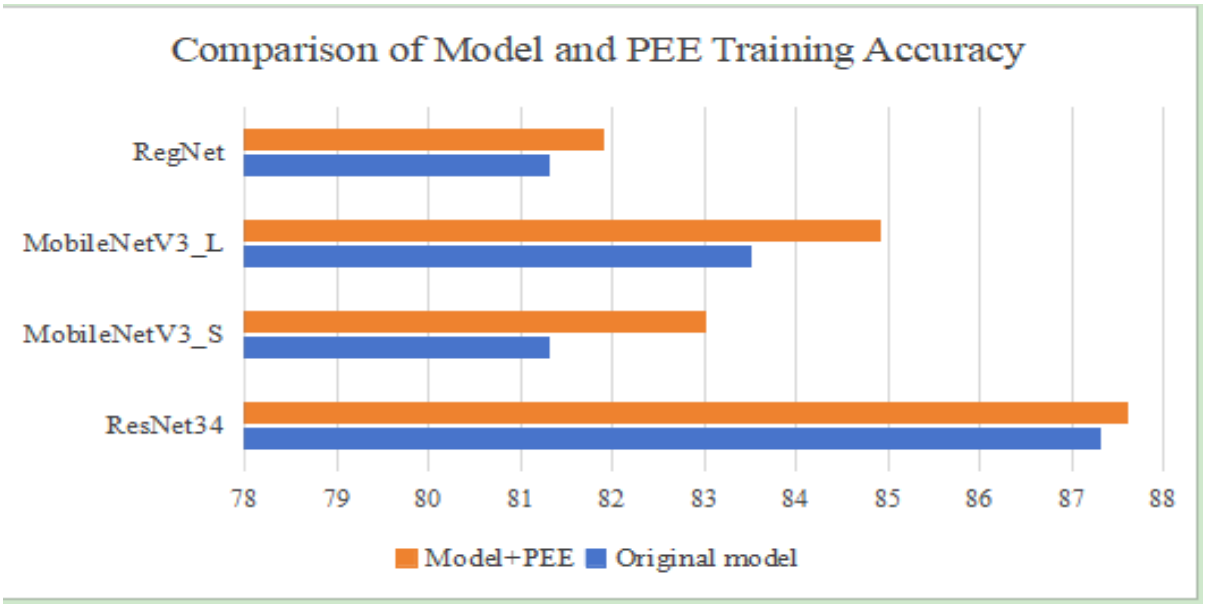


**Fig. 8.** Comparison of model and PPE training accuracy

### 4.2 Performance Evaluation in Semantic Segmentation

In the comparative experiments on semantic segmentation [35, 36] performance using the dsb2018_96 dataset [28, 29], adjustments were made to tailor our PPENet model more closely to the conditions required for semantic segmentation. During the experiment, we used the Adam optimizer with specific parameter settings: learning_rate = 1e-3, momentum = 0.9, weight decay = 1e-4, and a minimum learning rate of 1e-5. The training was conducted for 100 epochs. For data augmentation, we employed random image rotations and horizontal flips.The models tested against our PPENet included UNet, UNet++, SegNet, and others, serving as comparative experimental models for semantic segmentation.

Experimental Results. The semantic segmentation experiments are shown in Fig. 9, Fig. 10 and Table 4.

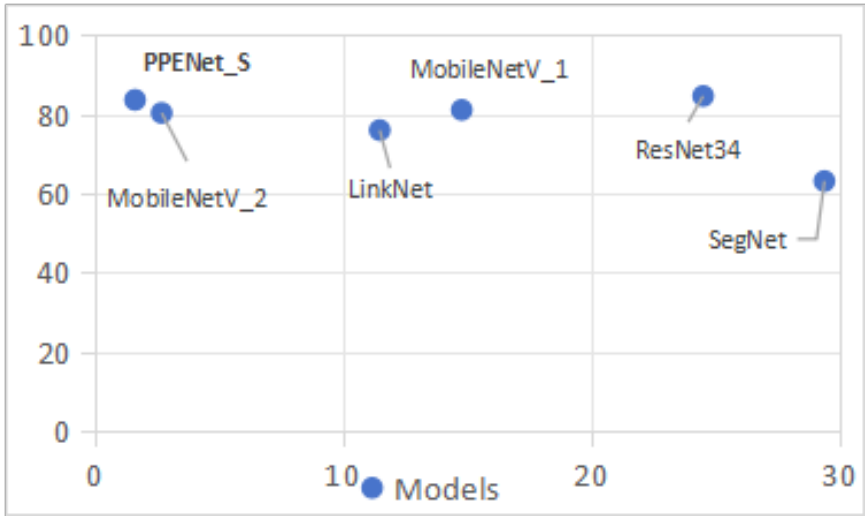


**Fig. 9.** Comparative experiment on segmentation performance

Fig. 9, Fig. 10 Using accuracy and parameter count, it is clear that the top-performing models are UNet++, UNet+PPE, and PPENet_s.

Table 4 Comprehensive performance metrics for semantic segmentation models reveal unexpected findings: our proposed PPENet_s model, with less than 1 million parameters, performs comparably to UNet++ and even achieves a Validation IOU (Intersection over Union) 0.3% higher than UNet++. Additionally, experiments inserting the PPEBlock module into the UNet model resulted in a segmentation performance of 81.97%, surpassing UNet and UNet++.

These results highlight that PPENet not only excels in image classification but also demonstrates significant potential and applicability in semantic segmentation tasks.

**Table 4.** Performance comparison table of image segmentation models

| Heading level | Models | Params (M) | mIOU |
|---|---|---|---|
| 1 | MobileNetV1 | 14.08 | 81.06% |
| 2 | MobileNetV2 | 2.72 | 80.30% |
| 3 | ResNet34 | 24.52 | 84.57% |
| 4 | SegNet | 29.4 | 63.11% |
| 5 | LinkNet | 11.5 | 75.96% |
| 6 | **PPENet_S** | **1.65** | **83.59%** |

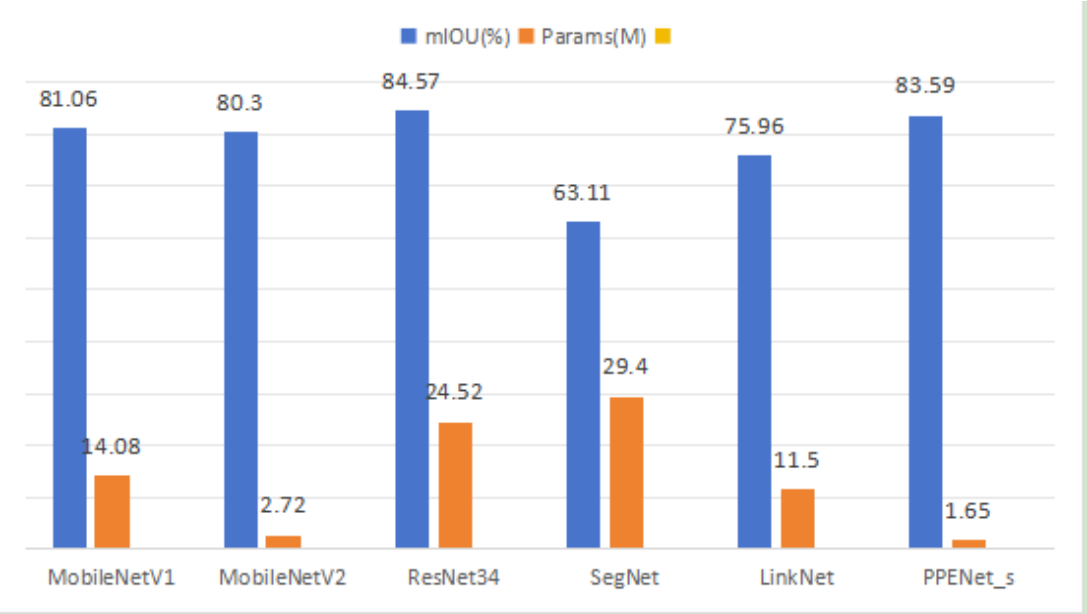


**Fig. 10.** Comparison experiment diagram of segmentation models

In the semantic segmentation experiments on the dsb2018_96 dataset:

We compared different models based solely on accuracy and model parameters to ensure clarity and conciseness of the figures without detailing specific data.

Detailed annotations were provided for the data of different models in the figures. All data presented in the figures were obtained from the experimental validation of different models in this study.

### 4.3 Ablation Study

The RKP blocks mainly consist of PPE blocks and RK blocks. In this section, we will study the main functions of each part of the RKP blocks and how they affect the overall performance of the model. The main purpose of RKP blocks is to fuse positional encoding information with diverse feature information, ensuring that the model's expressive power is enhanced while also improving its generalization and robustness. Next, we will use a

part of the RKP blocks to construct the model and conduct a thorough analysis of the importance of each component. This experiment will use the PPENet_l model, designed in this study, as the baseline model. First, we replaced the RK block in the baseline model and conducted the experiment under the same conditions as described in Section 4.1 of the previous chapter. Similarly, we also replaced the PPE blocks and performed the experiment. The experimental data is shown in Table 5:

**Table 5.** Impact of different modules on model performance

| Models | Top-1 ACC | Low-1 loss | Params (M) | Flops (G) |
|---|---|---|---|---|
| **PPENet_l** | **0.885** | **0.166** | **6.7** | **2.175** |
| PPE blocks | 0.852 | 0.252 | 0.58 | 0.147 |
| RK blocks | 0.868 | 0.188 | 5.4 | 2.115 |

It can be clearly seen from Table 5 that the experiment separating the RKP blocks, which play a vital role in the model structure, was conducted using the PPENet_l model as the baseline. We found that in the constructed model, the RK blocks are the primary source of parameters and computational complexity. Moreover, constructing the model using only a part of the RKP blocks fails to fully leverage the performance of this module.

In the process of rebuilding the model using the RKP blocks, both internal blocks are indispensable. Only by fully integrating the positional encoding information from the PPE blocks with the feature information from the RK blocks can the model's performance be fully realized.

At the same time, we also considered conducting experimental analysis by replacing either the PPE block or the RK block from the RKP blocks and applying them to other models, in order to verify the importance of inserting the PPE block into lightweight models. The experimental data table Table 6 shows:

**Table 6.** Influence of different modules on the performance of lightweight model

| Models | Bloacks | Top-1 ACC | Low-1 loss | Params (M) | Flops (G) |
|---|---|---|---|---|---|
| ResNet34 | Based | 0.86 | 0.16 | 21.2 | 3.67 |
| | PPE blocks | 0.876 | 0.15 | 21.2 | 3.68 |
| | RK block | 0.89 | 0.22 | 30.28 | 6.92 |
| MobileNetV3_s | Based | 0.813 | 0.239 | 1.6 | 0.06 |
| | PPE block | 0.83 | 0.218 | 1.6 | 0.06 |
| | RK block | 0.846 | 0.257 | 103.65 | 7.95 |
| mobileNetV3_l | Based | 0.835 | 0.179 | 4.2 | 0.24 |
| | PPE block | 0.849 | 0.218 | 4.2 | 0.24 |
| | RK block | 0.851 | 0.209 | 387.7 | 36.86 |

As shown in Table 6, experiments were conducted by embedding either the PPE block or the RK block into the backbone of different models for verification. We found that embedding the PPE block and RK block into other models resulted in accuracy improvements over the baseline model.hat is even more noteworthy is that after integrating the PPE block with other models, the performance improved without a significant increase in parameters and FLOPs; instead, they remained almost unchanged. In contrast, the RK block, while improving accuracy, also brings an increase in parameters and FLOPs. From this, we can conclude that when integrating with other lightweight models, using only the PPE block is a wiser choice, while in constructing new model architectures, it is necessary to use the combined RKP block, which integrates both the PPE block and the RK block. This is because using the RKP block in the model reconstruction process allows for better handling of the fusion between positional encoding information and image features, ensuring that the parameterization method is more effective.

### 4.4 Dynamic Study of Experimental Process

To better present the experimental process in real-time. we analyze the visualizations of different model features at various training stages, along with the training dynamics curves. Here, the MobileNet series, RepVGG and the PPENet peer model designed this time have trained 100 epochs on ImageNet. First, we compare the visualized feature maps of PPENet at different stages with those of other models. We observed that in the initial stages of

the model, PPENet is able to extract more feature information, and after processing by the backbone network, it still retains more image features containing positional encoding information. As shown in Fig. 11.

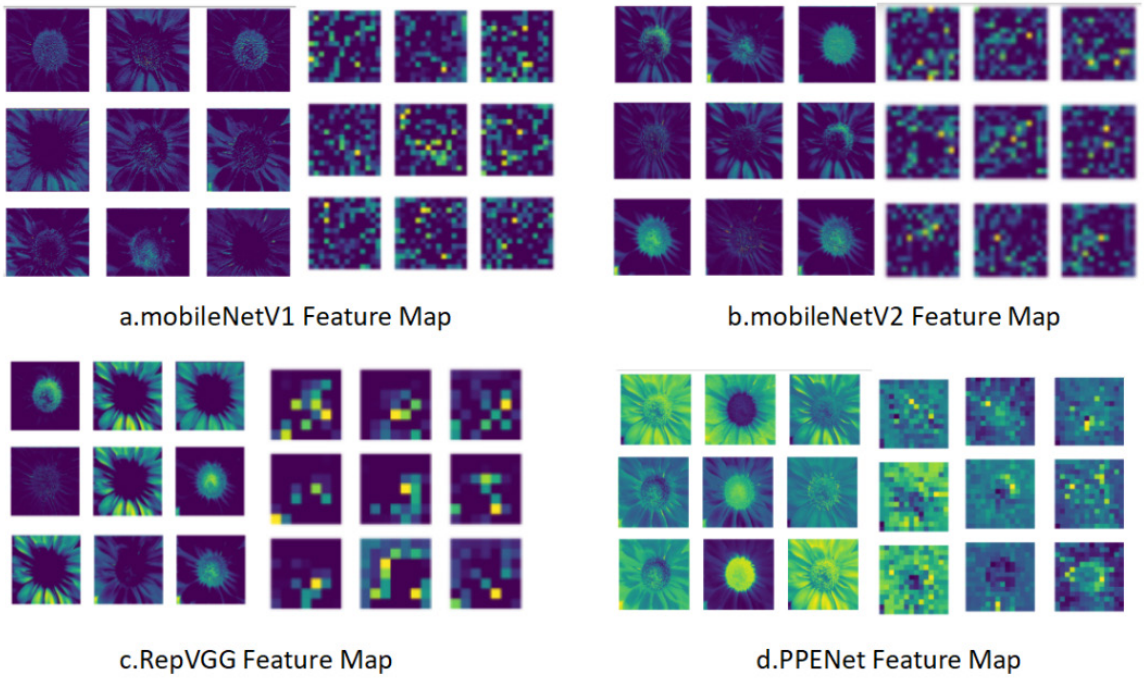


**Fig. 11.** Visual characteristic diagram of different stages

The dynamic curves plotted include the accuracy curve and the loss curve. We observed that the PPENet model demonstrates superior overall training accuracy, and its training loss converges faster compared to the other models being compared. Moreover, during the optimization process, the PPENet model, which integrates positional encoding information with feature map fusion, achieves better training accuracy and lower training loss than the compared models. As shown in Fig. 12.

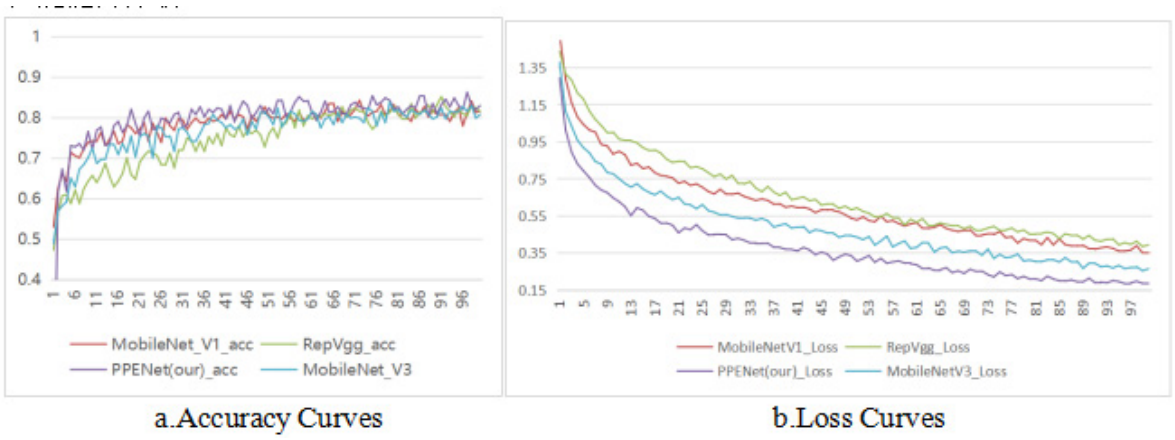


**Fig. 12.** Dynamic curve

### 4.5 Experimental Analysis

**Reasoning Time and Accuracy.** Here, we evaluate the proposed PPENet model in terms of inference time, model size, and other aspects. The inference time and other metrics of our PPENet and lightweight networks under default parameters and the same training conditions are presented in Table 7. It can be clearly seen that under the same training conditions, the proposed PPENet model outperforms the other models in terms of performance. Overall, the proposed PPENet model strikes a better balance between inference time and accuracy. Notably, compared to EfficientNet_bo, our PPENet_s demonstrates significant improvements in overall performance, especially with a 55.7% reduction in inference latency. See Table 7 for other detailed data comparison.

**Table 7.** Reasoning time and precision

| Models | Model size and lifting ratio | | Reasoning delay time and lifting ratio | | Training time and promotion proportion | |
|---|---|---|---|---|---|---|
| MobileNetV1 | 12.8MB | +25% | 7.08ms | 30.1% | 1.7h | 29.4% |
| MobileNetV2 | 14MB | 31.4% | 8.17ms | 39.4% | 1.8h | 33.3% |
| MobileNetV3_l | 16.8MB | 42.9% | 7.32ms | 32.4% | 1.4h | 14.3% |
| MobileNetV3_s | 6.4MB | -0.5% | 5.72ms | 13.5% | 1.75h | 31.4% |
| EfficientNet_b0 | 16.04MB | 40.2% | 11.17ms | 55.7% | 2.2h | 45.25% |
| **PPENet_s** | **9.6MB** | | **4.95ms** | | **1.2h** | |

**Lightweight Adaptation Analysis.** The main focus of this study is on the issue that the performance of parameterization methods is not as competitive as that of lightweight models in the application of model compression. For example:

(1) Parameterization methods similar to RepVGG can result in a large number of parameters and high FLOPs, which are not suitable for lightweight models.

(2) Performance issues, such as training accuracy, may not be as good as those of existing lightweight models.

To address this, we have developed and designed a parameterization method that incorporates the advantages of both Transformers and ConvNets, with positional encoding information, to improve the adaptability of parameterization methods in lightweight models. The lightweight adaptability of the model was analyzed. As shown in Table 8:

**Table 8.** Comparison of lightweight indicators

| Models | Based | | | **Based+PPE** | | |
|---|---|---|---|---|---|---|
| | Top-1 ACC | Param (M) | FLops (G) | Top-1 ACC | Param (M) | FLops (G) |
| RegNet | 81.3% | 2.7 | 0.21 | 81.9% | 2.7 | 0.21 |
| MobileNetV3_l | 83.5% | 4.2 | 0.24 | 84.9% | 4.2 | 0.24 |
| MobileNetV3_s | 81.3% | 1.6 | 1.6 | 83.0% | 1.6 | 0.06 |

As shown in Table 8, it can be analyzed that embedding the PPEBlock designed in this study into other lightweight models results in improved performance, with almost no change in the number of parameters and FLOPs.

By combining the analysis of Table 7, Table 8 and the previously mentioned Table 3. it can be seen that the lightweight model constructed using RKP (PPE block + RK block) shows significant improvements in both performance and parameter efficiency compared to existing lightweight models. Therefore, this demonstrates that the parameterization method combining the advantages of Transformers and ConvNets is suitable for mobile devices, aligning with the requirements of lightweight models for mobile applications.

**Limit Analysis.** To explore the limitations of the parameterized lightweight model in this study. We conducted multiple experiments under the same conditions for verification. The model PPENet, designed with the RKP block as its main backbone, demonstrates that stacking RKP blocks is not without limitations. The RK block within this module is the primary source of the increase in parameters. Therefore, blindly pursuing higher training accuracy by stacking modules indefinitely is not advisable. In the large and small versions of the PPENet model in this study, good performance was achieved with 5 and 3 modules, respectively.

In conclusion, the RKP block-based model cannot be stacked without limit, and good results can be achieved with a relatively small number of modules.

## 5 Conclusion

In this paper, we extensively described the proposed PPENet model. The model architecture integrates positional encoding parameterization into traditional convolutional model structures. This incorporation endows the model with both the characteristics of convolutional neural networks and positional awareness. By showcasing the characteristics and performance evaluations of lightweight models in the field of model compression, we conducted tests across various visual tasks such as image classification and semantic segmentation. After comparing the performance of our proposed model with various representative models, we found that our model achieves significant performance improvements in image classification and semantic segmentation tasks with lower dimensions and fewer parameters. However, this research focuses solely on the application of position encoding parameterization in lightweight models. It does not imply that it will always exhibit strong performance in deep learning tasks. In the future, we will continue to conduct in-depth research on position encoding parameterization to improve its adaptability across different models. This will better contribute our ideas and designs to the development of lightweight models for mobile devices.

## References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, in: Proc. 2021 9th International Conference on Learning Representations, 2021.

[2] H. Zhang, W. Hu, X. Wang, Parc-net: Position aware circular convolution with merits from convnets and transformer, in: Proc. 2022 European Conference on Computer Vision, 2022.
https://doi.org/10.1007/978-3-031-19809-0_35

[3] F. Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, in: Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
https://doi.org/10.1109/CVPR.2017.195

[4] J.-F. Dai, H.-Z. Qi, Y.-W. Xiong, Y. Li, G.-D. Zhang, H. Hu, Y.-C. Wei, Deformable Convolutional Networks, in: Proc. 2017 IEEE International Conference on Computer Vision, 2017.
https://doi.org/10.1109/ICCV.2017.89

[5] X.-H. Ding, X.-Y. Zhang, N.-N. Ma, J.-G. Han, G.-G Ding, J. Sun, RepVGG: Making VGG-style ConvNets Great Again, in: Proc. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
https://doi.org/10.1109/CVPR46437.2021.01352

[6] K.-M. He, X.-Y. Zhang, S.-Q. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016.
https://doi.org/10.1109/CVPR.2016.90

[7] A. Howard, M.-L. Zhu, B. Chen, D. Kalenichenko, W.-J. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. <https://arxiv.org/abs/1704.04861>, 2017 (accessed 9.11.2023).

[8] M. Sandler, A. Howard, M.-L. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, in: Proc. of the IEEE conference on computer vision and pattern recognition, 2018.
https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474

[9] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M.-T. Tan, W. Wang, Y.-K. Zhu, R. Pang, V. Vasudevan, Q. Le, H. Adam, Searching for MobileNetV3, in: Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
https://doi.org/10.1109/ICCV.2019.00140

[10] Y.-P. Chen, X.-Y. Dai, D.-D. Chen, M.-C. Liu, X.-Y. Dong, L. Yuan, Z.-C. Liu, Mobile-former: Bridging mobilenet and transformer, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition, 2022.
https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.00520

[11] M.-X. Tan, Q.V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, in: Proc. 2019 International conference on machine learning. PMLR, 2019.

[12] J.-R. Chen, S.-H. Kao, H. He, W.-P. Zhuo, S. Wen, C.-H. Lee, S.H.G Chan, Run, Don't walk: Chasing higher FLOPS for faster neural networks, in: Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023.
https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.01157

[13] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: Proc. of the 25th International Conference on Neural Information Processing Systems (NIPS), 2012.
[14] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, in: Proc. of the 3rd International Conference on Learning Representations, ICLR, 2015.
[15] C. Szegedy, W. Liu, Y.-Q. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, in: Proc. of the IEEE conference on computer vision and pattern recognition, 2015. https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594
[16] X.-Y. Lu, M. Suganuma, T. Okatani, SBCFormer: Lightweight Network Capable of Full-size ImageNet Classification at 1 FPS on Single Board Computers, in: Proc. 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024. https://doi.ieeecomputersociety.org/10.1109/WACV57701.2024.00116
[17] X. Li, W.-H. Wang, X.-L. Hu, J. Yang, Selective kernel networks. in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2019. https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00060
[18] J. Hu, S. Li, G. Sun, Squeeze-and-Excitation Networks, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. https://doi.org/10.1109/CVPR.2018.00745
[19] S. Woo, J.-C. Park, J.-Y. Lee, I.-S. Kweon, CBAM: Convolutional Block Attention Module, in: Proc. of the European conference on computer vision (ECCV), 2018.
[20] Z.-C. Cai X.-H. Ding, Q. Shen, X. Cao, Refconv: Re-parameterized refocusing convolution for powerful convnets. <https://doi.org/10.48550/arXiv.2310.10563>, 2023 (accessed 14.11.2023).
[21] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, J. Hervé, Training data-efficient image transformers & distillation through attention, in: Proc. of the 38th International Conference on Machine Learning, 2021.
[22] S.-X. Zheng, J.-C. Lu, H.-S. Zhao, X.-T. Zhu, Z.-K. Luo, Y.-B. Wang, Y.-W. Fu, J.-F. Feng, T. Xiang, P.H.S. Torr, L. Zhang, Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition, 2021. https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00681
[23] X.-H. Ding, H.-H. Chen, X.-Y. Zhang, J.-G. Han, G.-G. Ding, RepMLPNet: Hierarchical Vision MLP with Re-parameterized Locality, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition, 2022. https://doi.org/10.1109/CVPR52688.2022.00066
[24] X.-H. Ding, Y.-C. Guo, G.-G. Ding, J.-G. Han, ACNet: Strengthening the Kernel Skeletons for Powerful CNN via Asymmetric Convolution Blocks, in: Proc. of the IEEE/CVF international conference on computer vision (ICCV), 2019. https://doi.org/10.1109/ICCV.2019.00200
[25] X.-H. Ding, T.-X. Hao, J.-C. Tan, J. Liu, J.-G. Han, Y.-C. Guo, G.-G. Ding, ResRep: Lossless CNN Pruning via Decoupling Remembering and Forgetting, in: Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021. https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00447
[26] X.-H. Ding, X.-Y Zhang, J.-G. Han, G.-G. Ding, Diverse Branch Block: Building a Convolution as an Inception-like Unit, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2021. https://doi.org/10.1109/CVPR46437.2021.01074.
[27] X.-Y. Zhang, X.-Y. Zhou, M.-X. Lin, J. Sun, ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, in: Proc. of the IEEE conference on computer vision and pattern recognition (CVPR), 2018. https://doi.org/10.1109/CVPR.2018.00716
[28] S. Mehta, M. Rastegari, L. Shapiro, H. Haishirzi, ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2019. https://doi.org/10.1109/CVPR.2019.00941
[29] Z.-Y. Yan, X.-M. Li, M. Li, W.-M. Zuo, S.-G Shan, Shift-Net: Image Inpainting via Deep Feature Rearrangement, in: Proc. of the European conference on computer vision (ECCV), 2018. https://doi.org/10.1007/978-3-030-01264-9_1
[30] A. Hassani, H. Shi, Dilated Neighborhood Attention Transformer. <https://doi.org/10.48550/arXiv.2209.15001>, 2022 (accessed 8.12.2023).
[31] Z. Liu, Y.-T. Lin, Y. Cao, H. Hu, Y.-X. Wei, Z. Zhang, S. Lin, B. Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, in: Proc. of the IEEE/CVF international conference on computer vision (ICCV), 2021. https://doi.org/10.1109/ICCV48922.2021.00986
[32] Z.-Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, Y.-X. Li, MaxViT: Multi-Axis Vision Transformer, in: Proc. 2022 European conference on computer vision, 2022. https://doi.org/10.1007/978-3-031-20053-3_27

[33] Z.-H. Dai, H.-X. Liu, Q.V. Le, M.-X. Tan, CoAtNet: Marrying Convolution and Attention for All Data Sizes, in: Proc. 2021 Advances in neural information processing systems, 2021.
[34] M. Everingham, S.M.A. Eslami, L.V. Gool, C.K.I. Williams, J. Winn, A. Zisserman, The Pascal Visual Object Classes Challenge: A Retrospective, International journal of computer vision 111(1)(2015) 98-136. https://doi.org/10.1007/s11263-014-0733-5
[35] H.-Y. Wang, Y.-K. Zhu, H. Adam, A. Yuille, L.-C. Chen, MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2021. https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00542
[36] Y.-Q. Wang, Z.-L. Xu, X.-L. Wang, C.-H. Shen, B.-S. Cheng, H. Shen, H.-X. Xia, End-to-End Video Instance Segmentation with Transformers, in: Proc. of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), 2021. https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00863