International Journal of Science and Engineering Applications
Volume 14-Issue 04, 08 - 17, 2025, ISSN:- 2319 - 7560
DOI:10.7753/IJSEA1404.1003

MDA-RepXNet: Multi-Scale Object Detection with Multi-Dimensional Information

Shengzhe Liu
School of Electronic
Information and Electrical
Engineering
Yangtze University
Jingzhou 434023, China

Abstract: Objects of different scales exhibit varied characteristics within images, necessitating network architectures capable of handling multi-scale information for optimal performance in complex scenarios. However, ResNet struggles with detecting small or occluded objects. In this work, we introduce the Multi-Dimensional Attention- Reparameterized Multi-Branch Network (MDA-RepXNet), a novel multi-scale network designed to enhance performance in challenging conditions. MDA-RepXNet employs a Structurally Reparameterized Multi-Branch Network (RepXNet) as its backbone, incorporating multi-scale branches to manage different feature scales and thus improving small object recognition. Additionally, we propose a Multi-Dimensional Attention (MDA) mechanism to enhance feature fusion. This design significantly improves image feature extraction and object detection while maintaining high computational efficiency. Experimental results demonstrate that our approach increases the mean Average Precision (mAP) by 3.9% on the MS COCO dataset compared to the baseline, underscoring the effectiveness and superiority of the proposed method.

Keywords: Structural reparameterization, Dynamic networks, Multi-scale, Object detection

1. INTRODUCTION

In recent years, deep learning has achieved significant advances in the field of computer vision, especially through the application of Convolutional Neural Networks (CNNs). CNNs have demonstrated outstanding performance in various visual tasks, including image classification [1][2][3], object detection [4][5][6][7] and semantic segmentation [8][9][10]. Since the introduction of AlexNet [1] in 2012, which revolutionized computer vision research by proving that learned features could surpass hand-engineered features, the development of CNNs has been propelled forward. Following AlexNet, VGGNet [2], composed of convolutional layers, ReLU activations, and pooling layers forming VGG blocks, employed loops and subroutines to streamline the implementation of these repetitive structures within modern deep learning architectures. Subsequently, ResNet [11] continued the design of 3x3 convolutional layers from VGGNet, effectively mitigating the problem of gradient vanishing in deep network training by introducing residual structures. ResNet has since become the foundational network for a multitude of research and applications.

With the increasing demands for network performance, a single network architecture could no longer satisfy the requirements of complex visual tasks. Consequently, researchers have proposed various improvement schemes to enhance network expressiveness and computational efficiency. For instance, ResNeXt [12] built upon ResNet by constructing residual blocks using repeated grouped convolutions, thus providing greater flexibility and scalability while maintaining a relatively small number of parameters. DenseNet [13] reused features through dense connections, DLA [14] achieved efficient feature fusion via hierarchical aggregation, and Res2Net [15] introduced a multi-scale feature fusion mechanism within residual blocks, further enhancing the network's feature representation capabilities. MRMNet [16] optimizes multi-scale feature extraction and small object information representation by introducing multi-scale extension (MSE) and contextual feature refinement (CFR) modules while maintaining the plug-and-play characteristics of the modules. However, these methods use the same network structure for both training and inference, leading to reduced overall training efficiency.

This paper proposes a novel structure, RepXNet, as the backbone network. To enhance the network's feature representation capabilities during training, we introduce a multi-scale feature fusion structure. For inference, a reparameterization process converts multi-branch convolutions into single-branch convolutions, improving inference efficiency (Figure 1).

Multi-scale backbone networks significantly enhanced the expressiveness and robustness of neural networks by integrating information from different scales during the feature extraction phase. However, relying solely on the multi-scale characteristics of the backbone network was insufficient to fully capture objects of all scales in an image. In object detection tasks, the Feature Pyramid Network (FPN) [17] and its variants [18],[19] further extended this multi-scale concept. FPN effectively fused features of different scales by constructing a bottom-up feature pyramid, enhancing the model's ability to detect multi-scale objects. Additionally, it ingeniously combined these features from various levels to create a pyramid rich in semantic information, further improving detection performance. In recent years, FPN-like networks have further developed this concept. Path Aggregation Network (PANet) [20] enhanced feature fusion through additional pathways and short connections. Feature Pyramid Grid (FPG) [21] integrated a feature pyramid network with a grid attention mechanism to improve accuracy. Gated mechanisms [22] and scale-dynamic convolution kernels [23] were introduced, enhancing detection accuracy and model robustness in multiple complex scenarios. While these methods improved feature representation in the neck network through various connections, they overlooked the finer-grained feature details at the convolutional kernel level.

To address this, we propose a novel method called Multi-Dimensional Dynamic Feature Pyramid Network (MDFPN), which adaptively generates multi-dimensional dynamic convolution kernels based on input features at various scales. The core idea is to adaptively generate corresponding convolution weights from multi-scale features and employ MDA to learn convolutional attention across multiple dimensions of the kernel space in parallel. These attention types are complementary and are sequentially applied to the respective convolutional kernels, significantly enhancing the feature extraction capability of FPN.

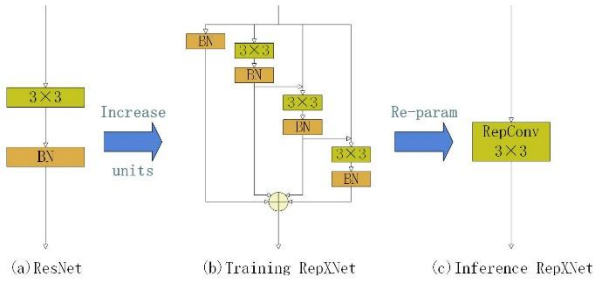

Figure 1. Increase units and structural reparameterization:(a) The second convolutional layer of the bottleneck block in ResNet. (b) The training phase of RepXNet with unit = 4, showcasing a multi-branch structure with residual connections. (c) The reparameterization inference process of RepXNet.

In conclusion, our main contributions include:

•We introduce RepXNet, a new multi-scale backbone network. RepXNet is segmented into distinct units during training, employing residual feature methods. In the inference phase, the multi-scale branches consolidate into a single convolutional structure. This design significantly enhances the network's feature representation capability and improves inference efficiency.

• We present a novel method called MDFPN, designed to dynamically generate multi-dimensional convolutional kernels based on input features at varying scales. MDFPN utilizes Multi-Dimensional Attention (MDA) to learn and apply convolutional attention across multiple dimensions simultaneously in the kernel. This strategy notably enhances the feature extraction capability of the network's neck architecture and improves model performance across diverse and complex scenarios.

•This work proposes the MDA-RepXNet architecture, a fusion of RepXNet and MDFPN, as illustrated in Figure 2. We showcase the efficacy of this approach in image classification and object detection tasks. Our method exhibits a 2.74% enhancement in Top-1 accuracy compared to the Baseline in image classification. In the downstream task of object detection, our method consistently delivers strong performance across both one-stage and two-stage detectors.

2. Related Work

Modern Convolutional Neural Networks. The development of CNNs can be traced back to the proposal of the backpropagation algorithm, a pivotal development that significantly enhanced the efficiency and feasibility of training neural networks. The journey from the seminal LeNet [24] to the more sophisticated Deep Belief Networks (DBN) [25] marked the gradual emergence of deep learning methodologies. In recent developments, Vision Transformer (ViT) [26] has harnessed the Transformer model from natural language processing [27], marking its foray into image vision processing. This paradigm shift has profoundly influenced traditional CNNs and catalyzed the development of a range of Transformer-based networks for both supervised [28],[29] and self-supervised learning [30],[31]. ViT also spurred advancements in weakly supervised learning networks, as evidenced by KMT-PLL [32], a methodology that tackled partial label learning (PLL) challenges through a fusion of K-means clustering and attention mechanisms. Window Token Transformer (WTT) [33] efficiently models long-range dependencies using learnable window tokens and the Global–Local Feedforward Network (GLFFN), while ensuring hierarchical information transfer through the Feature Inheritance Module (FIM). The emergence of Robust-ResNet [34] and ConvNeXt [35] signified a robust response, as these models integrated effective components from ViT into CNNs, achieving robustness that matched or surpassed Transformer models during testing. Therefore, this paper incorporates these efficient design elements into our architecture to leverage the advantages of ViT while enhancing processing capabilities for various tasks.

Structural Reparameterization. The reparameterization method enhances model performance by employing networks with diverse structures during the training and inference phases. Its fundamental principle lies in utilizing complex model architectures during training to augment expressive power and optimization effects while simplifying model structures during inference to enhance inference speed and reduce computational overhead. For instance, ACNet [36] incorporated asymmetric convolution blocks (ACB) during training, DBB [37] employed multiple different convolution kernels during training, and RepVGG [38] introduced additional branches and convolution operations during training. However, during the inference phase, these architectures were reparameterized into a single equivalent convolution operation, reducing computational complexity and enhancing inference speed. OREPA [39], on the other hand, leveraged a linear scaling layer to optimize online blocks more effectively, decomposing convolutions for online reparameterization, leading to a substantial reduction in training time for reparameterized models. FastViT [40] adopted a hybrid visual Transformer architecture, integrating the token mixing operator RepMixer, structural reparameterization, and large kernel convolutions. This approach eliminated skip connections in the network, further enhancing model efficiency. This paper adopts structural reparameterization technology, employing intricate network structures during training to improve model learning capability and transitioning to more efficient model structures during inference. This strategy significantly boosts model inference speed without compromising performance.

Dynamic Neural Networks. Dynamic neural networks exhibit adaptability by adjusting their structure and computational pathways based on changes in input data, thus improving computational efficiency and demonstrating strong potential in model generalization. Introducing the concept of dynamic convolution kernel [41] concept enhanced the network's expressive power and adaptability by generating convolutional kernels based on input data. DyNet [42] introduced dynamic computational graph structures to adjust network structure and computational pathways. Dynamic convolution [43] and CondConv [44] dynamically adjusted convolutional kernel weights by introducing attention mechanisms. Building upon this, WeightNet [45] added a set of hierarchical fully connected

International Journal of Science and Engineering Applications
Volume 14-Issue 04, 08 - 17, 2025, ISSN:- 2319 - 7560
DOI:10.7753/IJSEA1404.1003

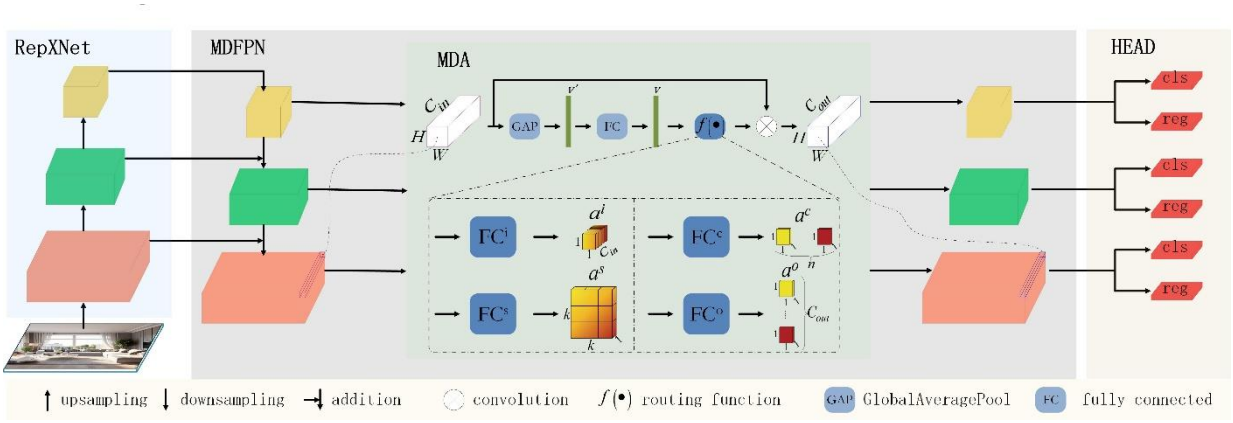

Figure 2. The MDA-RepXNet framework utilizes RepXNet as its backbone network and integrates MDFPN as its neck network. Apply the input channel attention $\boldsymbol{\alpha^i}$ to adjust each channel's importance in the input feature map. Use kernel attention $\boldsymbol{\alpha^c}$ and spatial attention $\boldsymbol{\alpha^s}$ to adjust convolutional kernel weights for convolution operations. Utilize an output channel attention $\boldsymbol{\alpha^o}$ to enhance the importance of each filter in the output feature map.

layers on the attention activation layer, unifying SENet [46] and CondConv, simplifying and enhancing the training process. DynamicViT [47] proposed a dynamic spatial sparsification framework. This framework significantly accelerated visual Transformer models by progressively pruning redundant tokens and introducing asymmetric computation. KAConv [48] embeds the attention mechanism into convolution kernels through the AG unit, adaptively adjusting kernel parameters to enhance the flexibility and information representation of the convolution. Our approach is closely related to dynamic convolution, which generates dynamic convolutional kernels based on input feature content, more effectively handling features of different scales and complexities.

3. Approach

In this section, we present an overview of the proposed multi-scale network architecture, MDA-RepXNet. To overcome the limitations of traditional methods, we introduce the RepXNet module as the backbone network in Sec 3.1. This module captures and analyzes features of different scales through a multi-scale design during the training phase and re-parameterizes the original structure during the inference phase. This reparameterization allows the network to maintain its original performance while achieving faster inference speeds. Sec 3.2, we first review dynamic convolution based on convolutional kernels, analyzing its advantages and disadvantages to design MDA and comparing its computational cost with traditional convolution methods. By employing a parallel strategy, MDA constructs MDFPN as the detection neck network, enabling the learning of multi-dimensional attention corresponding to the convolutional kernels at different levels of feature maps. This significantly enhances the network's feature representation capability.

The MDA-RepXNet framework utilizes RepXNet as its backbone network and integrates MDFPN as its neck network. This architecture effectively handles multi-scale targets while maintaining high efficiency and providing superior detection performance. In the subsequent sections, we present a series of quantitative and qualitative experiments to demonstrate the performance of MDA-RepXNet on multi-scale object detection tasks, along with analyses to reveal its intrinsic advantages.

3.1 RepXNet Module

Patchify Stem Replacement. In computer vision tasks, the stem initiates the network by processing input images to create effective feature representations. In standard ResNet-style architecture, the stem performs spatial downsampling using a 7×7 convolution layer with a stride of 2, followed by a 3×3 max-pooling layer. This process generates suitably sized feature maps for deeper layers. ViT-style architectures downsample more aggressively by dividing the input image into non-overlapping p×p patches, projecting each through a linear layer. This approach, resembling a convolutional kernel operation, enhances the robustness of ViTs. Applying this patchify method to CNNs helps bridge the robustness gap between CNNs and Transformers, as noted in [49]. To this end, we replace the ResNet-style stem with a ViT-style stem. Specifically, we utilize a convolution layer with a kernel size of 8 and a stride of 8. In stage 2, we set the stride of the first block to 1 to maintain the overall stride, ensuring that the 224×224 input image generates a 7×7 feature map before reaching the final global pooling layer (Figure 4). This substitution leverages the robust patchify operation from ViTs to enhance initial feature extraction in CNNs.

Multi-Scale Training and Re-parameterization Inference. During the training phase, complex multi-scale structures enhance the model's ability to represent data and improve training effectiveness. In the inference phase, these multi-branch structures are re-parameterized into an equivalent, more efficient single-branch structure to improve inference performance. This approach reduces computational load and memory usage during inference, maintaining high performance while increasing the model's inference speed.

During the training process, we enhanced the bottleneck blocks in ResNet by incorporating grouped convolution operations to improve feature extraction capabilities. Initially, the input x is transformed into a feature map I using a 1×1 convolution operation, resulting in $I \in \mathbb{R}^{N \times C_1 \times H_1 \times W_1}$, where C_1 is the number of channels. This feature map I serves as the input to subsequent multi-scale structures. After processing through these structures, we obtained a feature map $P \in \mathbb{R}^{N \times C_2 \times H_2 \times W_2}$, where the number of output channels is C_2.

We partition the feature map I into multiple units u, each with the same number of channels C_1/i, where i represents

International Journal of Science and Engineering Applications
Volume 14-Issue 04, 08 - 17, 2025, ISSN:- 2319 - 7560
DOI:10.7753/IJSEA1404.1003

the number of units and $i > 1$. For each unit, the input and output feature maps are denoted as $I_i \in \mathbb{R}^{N\times(C_1/i)\times H_1\times W_1}$ and $P^{(i)} \in \mathbb{R}^{N\times(C_2/i)\times H_2\times W_2}$, respectively. In the first unit, we utilize $\mu^{(0)}$, $\sigma^{(0)}$, $\gamma^{(0)}$, and $\beta^{(0)}$ as the channel mean, standard deviation, learned scaling factor, and bias term for the Batch Normalization (BN) layer, respectively. This BN layer is applied to the input I_1 to obtain the output $P^{(1)}$. Subsequently, $P^{(1)}$ is used as one of the inputs to the second unit, combined with I_2 through a residual connection to form a new input $I^{(2)}$. Then, $I^{(2)}$ is passed through a 3×3 convolution layer $W^{(3)} \in \mathbb{R}^{C_2\times C_1\times3\times3}$ with C_1 input channels and C_2 output channels, followed by a BN layer with parameters $\mu^{(3)}$, $\sigma^{(3)}$, $\gamma^{(3)}$, and $\beta^{(3)}$, to obtain the output $P^{(2)}$.

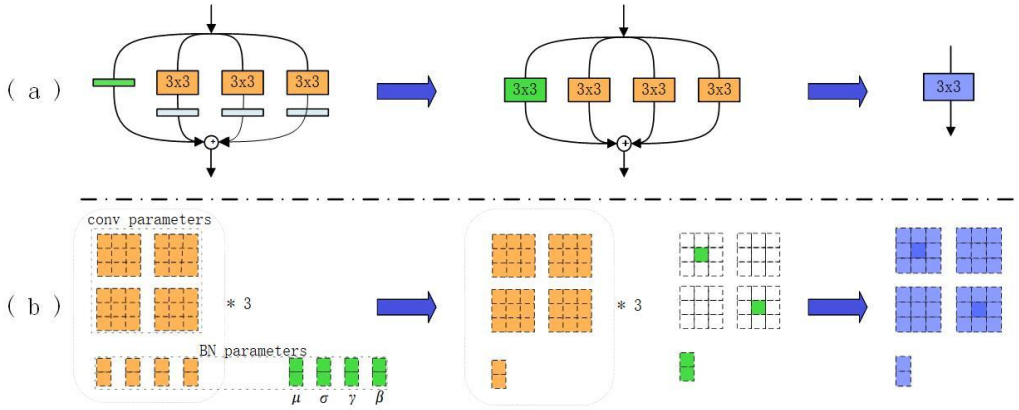

Figure 3. Visualization of Structural Reparameterization in the Bottleneck Section: (a) shows the structural form, with residual connections between branches omitted for clarity; (b) depicts the reparameterized form.

This process iterates for each subsequent unit. For each i-th unit, $P^{(i-1)}$ is combined with I_i through a residual connection to form a new input $I^{(i)}$, which is then passed through $W^{(3)}$ and the BN layer to obtain the corresponding unit output $P^{(i)}$, as shown in Figure 1(b). The above description can be formalized as:

$$\begin{aligned} P &= \mathrm{BN}(I_1,\mu^{(0)},\sigma^{(0)},\gamma^{(0)},\beta^{(0)}) \oplus \textstyle\sum_{i=1}^{i} P^{(i)}, \\ P^{(i)} &= \mathrm{BN}(I^{(i)} * W^{(3)},\mu^{(3)},\sigma^{(3)},\gamma^{(3)},\beta^{(3)}), \\ I^{(i)} &= I_i + P^{(i-1)} \end{aligned} \quad (1)$$

This improved structure enables the network to learn complex feature representations more effectively, enhancing model performance and generalization capabilities.

In the inference phase, these multi-branch structures are reparameterized into an equivalent, streamlined single-branch structure to enhance inference efficiency. This approach reduces computational load and memory usage during inference, maintaining high performance while increasing the model's inference speed. Formally, for the channel j of the non-first unit, ∀1≤j≤C_2/i, the batch normalization function during inference is converted according to the following expression:

$$\mathrm{BN}\left(I^{(i)}\right)_{:,j,:,:} = \gamma_j \odot \frac{I^{(i)}_{:,j,:,:} - \mu_i}{\sqrt{\sigma_j^2+\epsilon}} + \beta_j \quad (2)$$

where $\epsilon > 0$ is a small value to avoid division by zero. Each BN layer and the preceding convolution layer are merged into a single convolution layer with a bias vector.Let W, μ, γ, σ denote the converted convolution kernel W' and bias vector b', which are given by:

$$\begin{aligned} W'_{j,:,:,:} &\leftarrow \frac{\gamma_j \odot W_{j,:,:,:}}{\sqrt{\sigma_j^2+\epsilon}}, \\ b'_j &\leftarrow \beta_j - \frac{\gamma_j \odot \mu_j}{\sqrt{\sigma_j^2+\epsilon}}. \end{aligned} \quad (3)$$

By deriving from Eq. (2) and Eq. (3), we obtain:

$$\mathrm{BN}(I * W,\mu,\gamma,\sigma)_{:,j,:,:} = (I * W')_{:,j,:,:} + b'_j \quad (4)$$

The reparameterization method ensures that during inference, the network uses a simplified structure, thereby improving inference speed without sacrificing performance.

We extend this transformation to the first unit by treating its identity branch as a 1×1 convolution with an identity matrix kernel. Following this transformation, we obtain three 3×3 kernels, one 1×1 kernel, and four bias vectors, as illustrated in Figure 3. To maintain the size and position of the feature map, the 3×3 and 1×1 kernels must have the same stride. Consequently, we zero-pad the 1×1 kernel to match the size of the 3×3 kernels, resulting in a padded 3×3 kernel that is slightly smaller. This process involves combining the resulting 3×3 convolution kernels, including the padded 1×1 kernel, into a final 3×3 kernel, and adding the four bias vectors to produce the final bias vector.

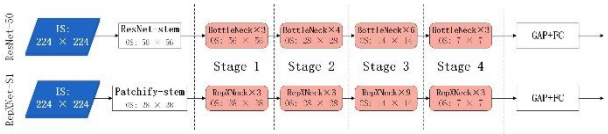

Figure 4. The structure of ResNet-50 and RepXNet-S1. Here, 'IS' denotes the input size, and 'OS' is the output size. By setting the stride of the first block in RepXNet stage 2 to 2, we ensure that the input image produces a 7×7 feature map before the final GAP layer.

Adjusting Stage Block Structure. ResNet mitigates the deep neural network degradation issue during training by incorporating residual blocks and shortcut connections. This stage-wise arrangement facilitates the development and training of deep networks without encountering challenges from gradient vanishing or explosion. Within ResNet, stages represent different network sections, each comprising multiple residual blocks, with variations in feature map sizes and

channel counting across stages. While the original ResNet stage configuration was primarily empirical, the Swin Transformer [49] follows a comparable structure but with distinct stage ratios. Recent research has extensively examined computation distribution [50],[51], hinting at possibilities for further optimized designs. This study adopts the computation ratios from [49], employing 1:1:3:1 ratio for shallower networks (as shown in Figure 4) and 1:1:9:1 ratio for deeper networks.

Activation functions. The Swish [52] activation function provides differentiability across the entire real number domain and exhibits smooth transitions near zero, effectively mitigating the vanishing gradient problem. In contrast to step functions like ReLU [53], the smoothness of Swish stabilizes model training, enhances performance, and accelerates convergence rates. Additionally, the Swish function allows for a trainable β parameter, enhancing the activation function's flexibility to adapt to various layer characteristics and data properties.

$$\text{Swish}_\beta(x) = x \bullet Sigmoid(\beta x) \quad (5)$$

While adding a trainable β parameter to the Swish activation function slightly increases the computational load, the performance benefits justify this overhead. During training, updating the additional β parameter requires just one extra multiplication operation to the overall model's parameter count. We argue that in practical scenarios, this increase is negligible, making the trade-off acceptable and beneficial.

3.2 Multi-Dimensional Dynamic Feature Pyramid Network (MDFPN)

Dynamic Convolution. Traditional convolutional layers utilize a single fixed convolution kernel, applying identical convolution operations to all input samples. The output feature y is obtained by convolving the input feature x with the fixed convolution kernel W, expressed as $y = W * x$, with the temporary exclusion of the bias term.

In contrast, dynamic convolution employs n parallel convolution kernels that aggregate into a single convolution kernel through an input-dependent attention mechanism. This method does not significantly increase computational overhead because the parallel convolutions share output channels and do not expand the network's dimensions. By adaptively fusing multiple convolution kernels based on image content, dynamic convolution reduces redundant computations and enhances feature extraction capabilities. Comprising convolution kernels W_k and attention functions α_k:

$$y = \sum_{k=1}^{n} a_k W_k * x \quad (6)$$

where $\alpha_k(x)$ is the attention weight for k-th kernel based on the input x. This mechanism allows dynamic convolution layers to adapt their operations to different inputs, enhancing the network's ability to extract meaningful features from the data.

Multi-dimensional Attention (MDA). MDA differs from existing dynamic convolution methods by focusing on designing attention mechanisms across spatial, input channel, and output channel dimensions. MDA first assigns attention scalar α^k to the parameters of each spatial position. Then, it assigns attention scalars α^i and α^o to the parameters of each input and output channel, respectively. Finally, like traditional dynamic convolution methods, attention scalar α^c is assigned to the entire convolution kernel:

$$y = \tilde{W} * x = \sum_{k=1}^{n} a_k^i \otimes a_k^s \otimes a_k^c \otimes a_k^o W_k * x \quad (7)$$

Spatial attention α^s effectively targets various locations within the input feature map, enabling the network to capture and utilize critical spatial information. This mechanism allows the network to adjust its focus dynamically to various positions within the input feature map, thereby emphasizing key spatial regions. Specifically, α^s assigns distinct attention weights to each spatial position of the convolution kernel. As the convolution kernel processes the input feature map, it dynamically applies these weights to capture significant local features more effectively, enhancing feature extraction capabilities.

Output channel attention α^o assigns varying attention weights to each output channel of a convolutional layer. These weights determine the response strength of each filter to the input features, dynamically adjusting each filter's contribution. In convolution operations, each filter corresponds to an output channel; hence, attention weights for output channels are essentially attention weights for the filters. By assigning higher attention weights to filters that are more responsive to the input features, the network can more effectively capture important features, achieving adaptive feature extraction.

Figure 2 illustrates the specific implementation process of MDA: global average pooling is applied to the input features, followed by a fully connected (FC) layer and ReLU activation function to obtain the feature vector v. This feature vector is then input into different FC layers, where the routing functions $\alpha_i = f_i(x)$ compute the corresponding attention weights. These weights are sequentially applied to the convolution kernel W, resulting in the final convolution kernel:

$$lv = \text{ReLU}(\text{FC}(GlobalAveragePool(x))), \quad a = f(x) = Sigmoid(\text{FC}(v)) \quad (8)$$

Computational Cost. For standard convolution, given an input feature map of size (C_{in}, H, W), a convolution kernel size of $k \times k$, and the number of output feature map channels C_{out}, the computational complexity is:

$$O(\bullet) = k^2 C_{in} C_{out} HW \quad (9)$$

For dynamic convolution kernels in MDFPN, the computational complexity is:

$$O(\bullet) = C_{in} HW + C_\alpha(2C_{in} + C_{out} + k^2 + K + 3) \quad (10)$$

Where, $C_\alpha < C_{\text{in}}$ and $C_\alpha < C_{\text{out}}$ represent the attention channels used to measure the attention mechanism based on the input channels. This design controls both the parameter and computational costs within the attention mechanism, thereby preventing overfitting and enhancing efficiency. K denotes the number of convolution kernels. Notably, while the attention mechanism introduces additional computational overhead, the overall increase in computational cost remains manageable because C_α is typically much smaller than C_{out}. Furthermore, when $K \ll HW$, the extra cost of computing the attention mechanism is minimal compared to the cost associated with the fixed convolution kernel. Therefore, the overall increase in computational complexity compared to regular convolution remains negligible.

MDFPN. Employing MDA for dynamic convolutions significantly improves multi-scale information processing in feature pyramid networks. This approach processes multi-scale information on the feature map and dynamically adjusts the convolution kernel weights, integrating multi-scale and multi-dimensional information. Consequently, the network can more

comprehensively capture and handle object features of varying scales. Specifically, MDFPN enhances the recognition ability of multi-scale objects by combining feature maps from different scales. Simultaneously, by dynamically adjusting convolution kernel parameters, it better adapts to multi-dimensional information of varying scales, effectively addressing detection challenges posed by changes in object scale. From the perspective of convolution kernels, MDFPN improves the network's adaptive capabilities.MDA's effectiveness extends beyond the MDFPN structure; it can also be used as a module in other networks, improving overall detection performance.

4. Experiments

To validate the performance of our proposed model, we compared RepXNet against a baseline on the ImageNet-1k dataset [54] and integrated it with MDFPN for downstream object detection tasks on the MS COCO 2017 dataset [55]. Through a series of comparative experiments and ablation studies, we demonstrated the outstanding performance of MDA-RepXNet.

4.1 RepXNet for ImageNet-1K

To prepare for training, we resize the images in the ImageNet-1K dataset to 256×256 pixels and then randomly crop them to 224×224 pixels. Our implementation of RepXNet, along with other models such as ResNet [11], ResNeXt [12], Res2Net [15], and RepVGG [38], is conducted using the PyTorch framework. We maintain consistently the same data argumentation strategy with [11] and [12], except for RepVGG. The initial learning rate is set to 0.1, with a decay rate of 10% every 30 epochs, and the training process extends over 100 epochs. For RepVGG, following [38],we initialize the learning rate to 0.1 and employ a cosine annealing strategy across 120 epochs. The training utilizes the standard SGD optimizer with globalbatchsize=256, weightdecay=0.0001, and momentum=0.9. RepXNet model configuration follows the criteria outlined in **Table 1**. All experiments utilize four NVIDIA GeForce RTX 3090 (24G) GPUs for execution.

Performance Comparison. Table 2 shows the Top-1 and Top-5 accuracies of various models on the ImageNet-1K dataset, using ResNet-50 as the baseline. Our implementation of the Res2Net model employed the Res2Net-50 configuration with "scale factor"=4s and "filter width"=26w. For the ResNeXt model, we adopted the ResNeXt-50 configuration with "Cardinality"=32 and "bottleneck width"=4. Despite a 0.5 FLOPs increase compared to ResNet-50, RepXNet-S1 maintains comparable computational efficiency. It outperforms ResNet-50 by 1.96% in Top-1 accuracy. In comparison to other multi-scale models, RepXNet-S1 achieves a 0.75% higher Top-1 accuracy than ResNeXt-50 and a 0.44% improvement over Res2Net. Although our model exhibits slightly higher computational and parameter complexity than ResNeXt, it demonstrates a 4.2 FLOP reduction and a 0.69% increase in Top-1 accuracy compared to more complex models like RepVGG. **Table 3** showcases our experimentation with various depths and units of RepXNet models. The comparison between RepXNet-S1, RepXNet-S2, and RepXNet-S3 in **Table 2** highlights that increasing the number of units can enhance the model's accuracy. However, this improvement comes with increased computational demands and longer training durations. Notably, despite RepXNet-S3 having more parameters and higher computational complexity than RepXNet-L1, it yielded inferior results. As a result, we decided against further increasing the unit under the same network depth.

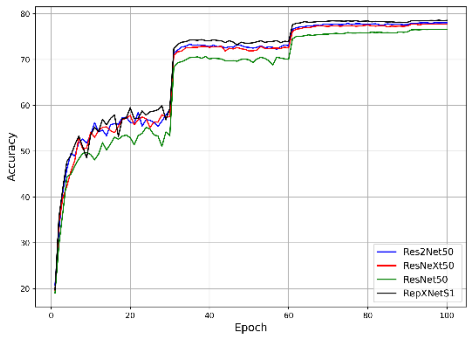

Figure 5. The Top-1 accuracy is displayed for the ImageNet-1K validation set, where all models shown were trained using identical methodologies.

Table 1. Different complexities were set for the models for comparison, with fixed filter width $= 26d$.

Name	Layer of each stage	unit
RepXNet-S1	3,3,9,3	4
RepXNet-S2	3,3,9,3	6
RepXNet-S3	3,3,9,3	8
RepXNet-L1	3,3,27,3	4

Table 2. ImageclassificationaccuracyontheImageNet-1K dataset. The speed is measured in samples per second with a global batch size of 256 on NVIDIA GeForceRTX3090.

Model	FLOPs	Speed	Top-1	Top-5
Baseline	4.1	1727	76.52	92.98
ResNeXt-50	4.3	1256	77.73	93.56
Res2Net-50	4.3	1623	78.04	93.82
RepVGG-B1g2	8.8	1864	77.79	93.77
RepXNet-S1	4.6	**1674**	78.48	94.18
RepXNet-S2	6.8	1205	78.82	94.37
RepXNet-S3	9.0	838	**79.26**	**94.55**

Table 3. Classification accuracy of deeper networks on the ImageNet-1K dataset.

Model	FLOPs	Params	Top-1	Top-5
ResNet-101	7.8	44.55	77.40	93.54
RepVGG-B2	20.4	89.02	78.54	94.18
Res2Net-101	8.1	45.21	79.13	94.44
RepXNet-S3	9.0	57.71	79.26	94.55
RepXNet-L1	8.9	49.60	79.54	94.71

International Journal of Science and Engineering Applications
Volume 14-Issue 04, 08 - 17, 2025, ISSN:- 2319 - 7560
DOI:10.7753/IJSEA1404.1003

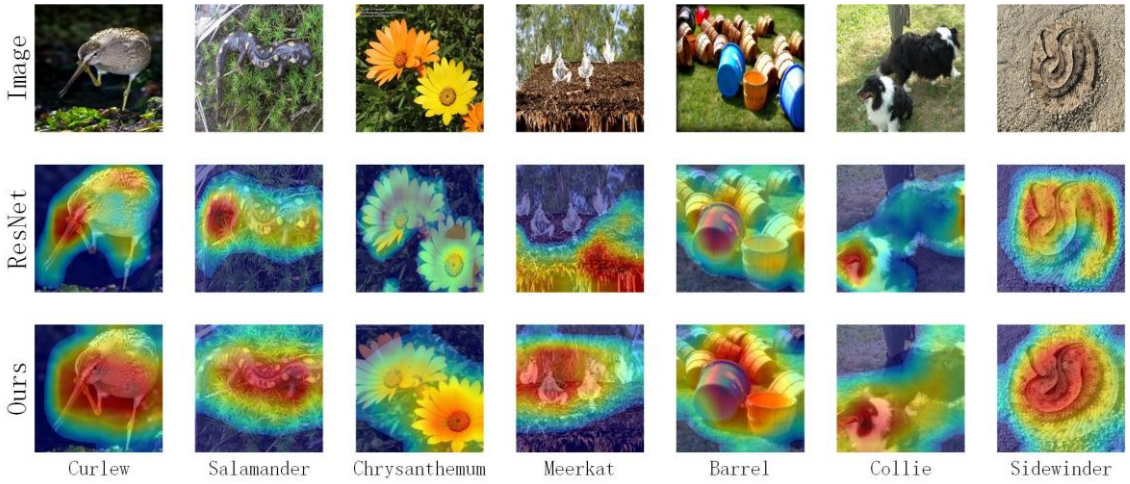

Figure 6. Comparison of CAM visualization results between ResNet-50 and RepXNet.

Visualization and CAM. Figure 5 shows the Top-1 accuracy performance of ResNet-50, ResNeXt-50, Res2Net-50, and RepXNet-S1 on the ImageNet-1K validation set. In the initial stages of training, RepXNet-S1 shows more fluctuations in accuracy compared to the other models, indicating a longer adaptation period. This increased volatility in RepXNet-S1's performance is likely due to its complex architecture, which necessitates more time to adapt and stabilize. However, when the learning rate decreases to 0.01, RepXNet-S1 demonstrates improved stability and reduced fluctuations in accuracy. This observation suggests that the model effectively fine-tunes its parameters, leading to a smoother learning curve and achieving the highest Top-1 accuracy among all models.

We employed the XGrad-CAM [56] method to visualize Class Activation Maps (CAM) and better understand the multi-scale representation capabilities of RepXNet-S1. XGrad-CAM enhances gradient weighting through normalized activations, resulting in more distinct and precise heatmaps that enhance interpretability. As illustrated in Figure 6, areas with heightened CAM responses are shown with more vivid colors. Comparing the CAM images of ResNet-50 and RepXNet-S1, we noticed a significant difference in their coverage patterns. RepXNet-S1 consistently captures the entire object comprehensively, while ResNet-50 focuses on parts of the object, such as the curlew, chrysanthemum, and collie. In an image featuring meerkats, ResNet-50 concentrated on the roof, leading to false detection, whereas RepXNet-S1 effectively encompassed all four meerkats.

4.2 MDA-RepXNet for MS-COCO

To evaluate the performance of our backbone networks in the downstream task of object detection and to validate the feasibility of MDFPN, we train the networks on the MS-COCO 2017 [55] training set and report results on the validation set. All backbone networks utilize pre-trained models from ImageNet-1k [54] and then fine-tune them on the detection dataset. We employ standard COCO evaluation criteria and select classic models such as Faster R-CNN [7] and RetinaNet [6] as base models. All experiments utilize four NVIDIA GeForce RTX 3090 (24G) GPUs for execution. The initial learning rate is set to 0.001 and decays by 10% at the 8th and 11th epochs, during a total of 12 epochs. During the first 500 iterations, the learning rate increases to 0.02. We use the standard SGD optimizer for training all models, with a batch size of 2 per GPU, a weight decay of 0.0001, and momentum=0.9.

Performance Comparison. We conducted experiments comparing ResNet-50 and RepXNet-S1 as backbone networks. Additionally, for ResNet-50, we incorporated the FPN [17], for comparative experiments with other neck networks, including PAFPN [20], BFPN [57], the lightweight FPG [21], and our proposed MDFPN. **Error! Reference source not found.** presents the experimental results obtained using different models. When using ResNet-50 as the backbone, MDFPN's mAP improved by 2.8% compared with FPN. Although MDFPN slightly underperformed FPG in medium and large object detection, it showed a minor improvement in small object detection. Furthermore, using MDA-RepXNet resulted in a 3.9% overall accuracy increase compared to the baseline.

Visualization. We showcase the detection results of ResNet w./FPN and MDA-RepXNet as network structures for object detection. In Figure 7(a), ResNet w./FPN generated false positives, detecting objects like dogs and cars that were not present. An overall comparison reveals that our method exhibits higher confidence in detecting larger objects while still making accurate predictions for occluded or relatively small objects, such as the bench in Figure 7(b) and the car in Figure 7(c).

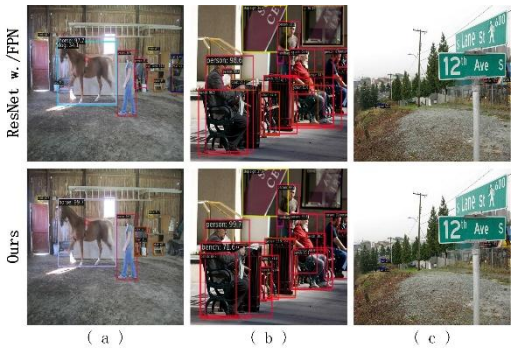

Figure 7. Example results of object detection using ResNet w./FPN and MDA-RepXNet under the Faster R-CNN framework.

4.3 Ablation Study

Why adopt the trainable Swish function as the activation function for RepXNet? Error! Reference source not found.presents the Top-1 and Top-5 accuracy rates on ImageNet-1K achieved by modifying different activation functions, using RepXNet-S1 as the backbone network. The results indicate that a fixed β value reduces accuracy and performs worse than ReLU as the activation function. Swish, a smooth function that interpolates non-linearly between the linear function and ReLU, demonstrates improved performance. Making β as a trainable parameter allows the model to control the degree of interpolation, enhancing adaptability and optimizing non-linear adjustments. This adaptability enables RepXNet to capture features more effectively and improve performance when handling complex data and tasks.

Impact of Unit (u) and Width (d) on Network Performance. In Table 2, we observed a significant reduction in model error rates as the number of units increased. To further explore their impact, we experimented with different widths and units. As shown in Table 6**Table 5**, increasing the number of units from 4 to 6 led to a noticeable improvement in the model's Top-1 accuracy. However, reducing the width from 26 to 14, while keeping 6 units, significantly decreased the model's accuracy by 0.24% compared to RepXNet-S1 with fewer units and $\text{width} = 26d$. Therefore, combined with the results in Table 2, we infer that increasing the number of units and widths can improve the model's accuracy, with the number of units having a more significant impact.

Table 4. Report on object detection results on the MS COCO 2017 dataset, 'w./' indicating 'with'.

Method	backbone	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN w./FPN	ResNet-50	37.3	57.9	40.3	19.6	40.2	49.2
Faster RCNN w./PAFPN	ResNet-50	37.8	58.7	40.8	21.4	41.0	48.6
Faster RCNN w./BFPN	ResNet-50	38.3	59.5	41.9	22.1	42.0	48.5
Faster RCNN w./FPG	ResNet-50	39.7	59.0	42.0	20.1	42.9	53.0
Faster RCNN w./MDFPN	ResNet-50	40.1	59.7	42.4	22.5	42.6	52.9
Faster RCNN w./FPN	RepXNet-S1	39.6	59.2	42.9	21.7	42.4	51.1

Table 5. Comparing the accuracy of RepXNet on ImageNet-1K using different activation functions, β^o indicates trainability.

Setting	Top-1	Top-5
ReLU	78.31	94.06
Swish $\beta = 1$	78.11	93.97
Swish β^o	78.48	94.18

Table 6. Validating the performance of RepXNet with different units and widths on the ImageNet-1K dataset.

Model	Setting	Top-1
RepXNet	4×26d	78.48
RepXNet	6×26d	78.82
RepXNet	6×14d	78.24

Comparison of Results on Different Base Models. To validate the feasibility of our network with other base models, we employed RetinaNet as the foundational network, RepXNet-S1 as the backbone, and integrated MDFPN as the neck. Table 7 shows the outcomes. These experiments confirm that our network architecture is indeed viable within the context of one-stage networks. However, it is worth noting that the overall performance is comparatively lower when compared to that observed in two-stage networks.

Table 7. Validating the feasibility of MDA-RepXNet using different foundational networks on the MS COCO dataset.

Method	mAP	AP_{50}	AP_{75}
Faster RCNN	41.2	61.6	43.7
RetinaNet	40.1	60.2	43.1

5. Conclusion

The proposed Multi-Dimensional Attention - Reparameterized Multi-Branch Network (MDA-RepXNet) combines RepXNet's multi-scale feature extraction capabilities with MDFPN's multi-dimensional feature fusion strategy. This integration helps MDA-RepXNet preserve the precise detail features of original images, significantly improving performance for image classification and object detection tasks. Its advanced feature extraction is particularly effective at detecting small objects and handling occlusions, demonstrating the method's robustness and efficacy. Future research will explore the effects of different unit configurations and width variations on multi-scale networks. Additionally, we plan to integrate these components into more advanced network structures to maximize their potential.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[3] He, L., Ai, Q., Lei, Y., Pan, L., Ren, Y., & Xu, Z. (2023). Edge enhancement improves adversarial robustness in image classification. Neurocomputing, 518, 122-132.

[4] Dong, Y., Shen, L., Pei, Y., Yang, H., & Li, X. (2023). Field-matching attention network for object detection. Neurocomputing, 535, 123-133.
[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
[6] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).
[7] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
[8] Shelhamer, E., Long, J., & Darrell, T. (2016). Fully convolutional networks for semantic segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(4), 640-651.
[9] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848.
[10] Liu, Q., Dong, Y., & Li, X. (2023). Multi-stage context refinement network for semantic segmentation. Neurocomputing, 535, 53-63.
[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
[12] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
[13] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
[14] Yu, F., Wang, D., Shelhamer, E., & Darrell, T. (2018). Deep layer aggregation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2403-2412).
[15] Gao, S. H., Cheng, M. M., Zhao, K., Zhang, X. Y., Yang, M. H., & Torr, P. (2019). Res2net: A new multi-scale backbone architecture. IEEE transactions on pattern analysis and machine intelligence, 43(2), 652-662.
[16] Dong, Y., Liu, Y., & Li, X. (2024). MRMNet: Multi-scale residual multi-branch neural network for object detection. Neurocomputing, 596, 127886.
[17] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).
[18] Guo, C., Fan, B., Zhang, Q., Xiang, S., & Pan, C. (2020). Augfpn: Improving multi-scale feature learning for object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 12595-12604).
[19] Zhang, D., Zhang, W., Li, F., Liang, K., & Yang, Y. (2023). PNANet: Probabilistic two-stage detector using pyramid non-local attention. Sensors, 23(10), 4938.
[20] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8759-8768).
[21] Chen, K., Cao, Y., Loy, C. C., Lin, D., & Feichtenhofer, C. (2020). Feature pyramid grids. arXiv preprint arXiv:2004.03580.
[22] Zhu, M. (2024, March). Dynamic feature pyramid networks for object detection. In Fifteenth International Conference on Signal Processing Systems (ICSPS 2023) (Vol. 13091, pp. 503-511). SPIE.
[23] Zhang, K., Li, Z., Hu, H., Li, B., Tan, W., Lu, H., ... & Pu, S. (2022, July). Dynamic feature pyramid networks for detection. In 2022 IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-6). IEEE.
[24] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
[25] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527-1554.
[26] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
[27] Ashish, V. (2017). Attention is all you need. Advances in neural information processing systems, 30, I.
[28] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021, July). Training data-efficient image transformers & distillation through attention. In International conference on machine learning (pp. 10347-10357). PMLR.
[29] Xue, F., Shi, Z., Wei, F., Lou, Y., Liu, Y., & You, Y. (2022, June). Go wider instead of deeper. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 8, pp. 8779-8787).
[30] Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., & Kong, T. (2021). ibot: Image bert pre-training with online tokenizer. arXiv preprint arXiv:2111.07832.
[31] Chen, X., Xie, S., & He, K. (2021). An empirical study of training self-supervised vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9640-9649).
[32] Fan, J., Huang, L., Gong, C., You, Y., Gan, M., & Wang, Z. (2024). KMT-PLL: K-means cross-attention transformer for partial label learning. IEEE Transactions on Neural Networks and Learning Systems.
[33] Mao, J., Chang, Y., & Yin, X. (2023). Window Token Transformer: Can learnable window token help window-based transformer build better long-range interactions?. Neurocomputing, 559, 126793.
[34] Wang, Z., Bai, Y., Zhou, Y., & Xie, C. (2022). Can cnns be more robust than transformers?. arXiv preprint arXiv:2206.03452.
[35] Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A convnet for the 2020s. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11976-11986).
[36] Ding, X., Guo, Y., Ding, G., & Han, J. (2019). Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1911-1920).
[37] Ding, X., Zhang, X., Han, J., & Ding, G. (2021). Diverse branch block: Building a convolution as an inception-like unit. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10886-10895).

[38] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., & Sun, J. (2021). Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 13733-13742).

[39] Li, Y., Zhang, Y., Timofte, R., Van Gool, L., Yu, L., Li, Y., ... & Wang, X. (2023). NTIRE 2023 challenge on efficient super-resolution: Methods and results. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 1922-1960).

[40] Vasu, P. K. A., Gabriel, J., Zhu, J., Tuzel, O., & Ranjan, A. (2023). Fastvit: A fast hybrid vision transformer using structural reparameterization. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 5785-5795).

[41] Jia, X., De Brabandere, B., Tuytelaars, T., & Gool, L. V. (2016). Dynamic filter networks. Advances in neural information processing systems, 29.

[42] Zhang, Y., Zhang, J., Wang, Q., & Zhong, Z. (2020). Dynet: Dynamic convolution for accelerating convolutional neural networks. arXiv preprint arXiv:2004.10694.

[43] Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic convolution: Attention over convolution kernels. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11030-11039).

[44] Yang, B., Bender, G., Le, Q. V., & Ngiam, J. (2019). Condconv: Conditionally parameterized convolutions for efficient inference. Advances in neural information processing systems, 32.

[45] Ma, N., Zhang, X., Huang, J., & Sun, J. (2020, August). Weightnet: Revisiting the design space of weight networks. In European Conference on Computer Vision (pp. 776-792). Cham: Springer International Publishing.

[46] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[47] Rao, Y., Liu, Z., Zhao, W., Zhou, J., & Lu, J. (2023). Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(9), 10883-10897.

[48] Shan, X., Ma, T., Shen, Y., Li, J., & Wen, Y. (2022). KAConv: Kernel attention convolutions. Neurocomputing, 514, 477-485.

[49] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 10012-10022).

[50] Radosavovic, I., Johnson, J., Xie, S., Lo, W. Y., & Dollár, P. (2019). On network design spaces for visual recognition. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1882-1890).

[51] Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., & Dollár, P. (2020). Designing network design spaces. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10428-10436).

[52] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

[53] Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[54] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[55] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13 (pp. 740-755). Springer International Publishing.

[56] Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., & Li, B. (2020). Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. arXiv preprint arXiv:2008.02312.

[57] Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. (2019). Libra r-cnn: Towards balanced learning for object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 821-830).