Position-Aware Recalibration Module: Learning From Feature Semantics and Feature Position

Xu Ma , Song Fu
Department of Computer Science and Engineering
University of North Texas
Denton, Texas 76203, USA
xuma@my.unt.edu, Song.Fu@unt.edu

Abstract

We present a new method to improve the representational power of the features in Convolutional Neural Networks (CNNs). By studying traditional image processing methods and recent CNN architectures, we propose to use positional information in CNNs for effective exploration of feature dependencies. Rather than considering feature semantics alone, we incorporate spatial positions as an augmentation for feature semantics in our design. From this vantage, we present a Position-Aware Recalibration Module (PRM in short) which recalibrates features leveraging both feature semantics and position. Furthermore, inspired by multi-head attention, our module is capable of performing multiple recalibrations where results are concatenated as the output. As PRM is efficient and easy to implement, it can be seamlessly integrated into various base networks and applied to many position-aware visual tasks. Compared to original CNNs, our PRM introduces a negligible number of parameters and FLOPs, while yielding better performance. Experimental results on ImageNet and MS COCO benchmarks show that our approach surpasses related methods by a clear margin with less computational overhead. For example, we improve the ResNet50 by absolute 1.75% (77.65% vs. 75.90%) on ImageNet 2012 validation dataset, and 1.5%∼1.9% mAP on MS COCO validation dataset with almost no computational overhead. Codes are made publicly available[1].

1 Introduction

Humans learn visual scenes by considering both distinct objects and the surrounding environment [Yu *et al.*, 2018; Borji and Itti, 2012]. By analyzing the target representation and considering the surrounding context, people can infer visual concepts effectively.

In computer vision, such attributes can be obtained by analyzing the feature dependencies on feature maps [Wang *et al.*, 2018]. However, directly querying for the dependencies over the entire image for each pixel would radically increase the computation overhead, making it infeasible to integrate with each block in a convolutional neural network [Wang *et al.*, 2018]. In addition, such an approach is inefficient since many pixels are useless and even detrimental, making little to no contribution to feature extraction.

To mitigate this problem, researchers propose to query the dependencies in a more succinct fashion; instead of querying for every pixel, they query the global context by using a query-independent operation [Hu *et al.*, 2018b; Cao *et al.*, 2019]. The change from query-specific to query-independent operations significantly reduces computation overhead while achieving comparable performance. This design presents a satisfying result for a range of vision tasks, but it suffers from the inherent weakness: the spatial position information is discarded. *Even if we disrupt the spatial position of the features, there will be no change in the results for query-specific or query-independent operations* (e.g., in GC [Cao *et al.*, 2019] and Non-Local [Wang *et al.*, 2018] modules).

While positional information is barely exploited in prevalent CNN models, it has never gone out of favor in literature and gained more traction recently. An inspiring example is the Transformer [Vaswani *et al.*, 2017] for machine translation tasks. To remedy the position-agnostic property of self-attention mechanisms, a lightweight fully-connected network is applied by the Transformer to each position separately and identically. More recently, Local-Relation Network [Hu *et al.*, 2019] for image recognition tasks considers geometric priors by using a small sub-network. However, the local relation network is difficult to implement since it possesses little compatibility with the common deep learning frameworks. We also notice that convolutional layers can implicitly learn the positional information from the widely used zero-padding operations [Islam *et al.*, 2020]. Nevertheless, the simple zero-padding operations are insufficient when compared with direct position learning.

In this paper, we emphasize that feature positional knowledge makes a significant contribute to many vision tasks in addition to feature semantics. Following this direction, we propose a new lightweight module that distills this insight into a combination pipeline. To ensure networks can effectively construct feature presentations, we calculate similarities and relative positions for the most discriminative features, and combine them to recalibrate the intermediate fea-

[1] https://github.com/13952522076/PRM

ture maps. In addition, we leverage the mean of each channel as a global context for supplement. To further improve the performance, we extend our method to a reformed multi-head version, similar to that in [Vaswani *et al.*, 2017], but leveraging group operations.

Experimental results show that our position-aware recalibration module outperforms the baseline and related modules by a clear margin, with reduced computation complexity and model size. We evaluate our module on multiple vision tasks including image recognition and object detection. Compared with plug-in modules in [Hu *et al.*, 2018b; Hu *et al.*, 2018a; Woo *et al.*, 2018; Cao *et al.*, 2019; Li *et al.*, 2019a; Li *et al.*, 2019b], we achieve results either on par or better with fewer parameters and FLOPs. Meanwhile, when applying our method to object detection, our PRM consistently achieves at least 1.5% $\sim$ 1.9% absolute mAP improvement on the MS COCO benchmark.

The main contributions of this paper are as follows.

- Experimental results show that it is practical to calculate the dependencies between the feature map and the most representative query points for the key-query operations. To further reduce parameters and FLOPs, we process the dependencies by leveraging normalization statistics rather than applying learnable layers.
- We introduce the relative positional information to the learning procedure, which is crucial in the human visual system, but overlooked in existing work.
- To balance accuracy and computational overhead, we develop and operate multi-head PRM in groups, which achieves a stronger capability with minimal parameter increase compared to the original PRM.

2 Related Work

2.1 Position Encoding

Position encoding encodes the geometric position and distance information of pixels in an image [González and Woods, 1981]. Typically, it encodes the spatial dependency between a pair of pixels based on their distance. Recent technologies in neural language processing employ position embedding to explore word dependencies, such as Transformer [Vaswani *et al.*, 2017]. Most recently, position encoding has attracted more interests in vision community. For example, AANet [Bello *et al.*, 2019] incorporates positional information into self-attention, Hu et al. [Hu *et al.*, 2019] exploit features' spatial distances to enhance the feature representation ability. Although position encoding improves the representation ability of local relation networks, the sliding window and short range of receptive field make it difficult to learn the global geometric dependency. In contrast, we encode the relative position information between a selected point and global feature maps.

2.2 Self-Attention Mechanisms

To our knowledge, self-attention was first proposed to explore global dependencies in machine translation [Vaswani *et al.*, 2017]. For each position, self-attention calculates the weighted response of the global context in a mapping space. However, traversing the global dependency for all points is prohibitively expensive. To this end, Global Context [Cao *et al.*, 2019] applies a query-independent formulation, which reduces the computation overhead while achieving a comparable performance. Inspired by that work, we compute the similarities between entire feature map and the most characteristic features. To further improve our method, we present a multi-head attention design that calculates dependencies in groups and concatenates the results as the output.

2.3 Normalization

Normalization has been an indispensable component in state-of-the-art deep learning models. In order to mitigate distribution shifting in a deep neural network, Ioffe et al. introduced Batch Normalization (BN) [Ioffe and Szegedy, 2015], which scales and shifts the output of each convolutional layer. However, the performance of BN is affected by the batch size. When using a small batch size which is common for training object detection and segmentation networks, BN suffers from dramatic performance degradation. To this end, various improved normalization layers are proposed to overcome this problem [Ba *et al.*, 2016]. In this paper, we leverage the normalization statistics in our design to process obtained feature dependencies.

3 Our Proposed Method

In this section, we describe the proposed position-aware recalibration module (PRM) in detail. The input and output of a PRM are denoted as $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ and $\mathbf{y} \in \mathbb{R}^{C\times H\times W}$, where C is the channel number and $H \times W$ indicates the spatial resolution, and the batch dimension is not included for brevity. Inspired by self-attention mechanisms, we consider feature dependencies for each key-query pair. We note that calculating the relationship for every pair of key-query is compute intensive, as pointed out in [Wang *et al.*, 2018]. For better efficiency, we opt to calculate the relationship between a feature map $\mathbf{x}$ and the most representative query point $q \in \mathbb{R}^C$. We select the query point q by the index of max-mean value along the channel dimension of each position in $\mathbf{x}$. Then, we encode the relative geometric position to reweight the dependencies. As we choose only the most discriminative point, it may not be sufficient to present the whole feature map, as demonstrated in [Woo *et al.*, 2018]. For this reason, we also calculate the dependencies between the feature map $\mathbf{x}$ and a global context $z \in \mathbb{R}^C$ (which is obtained by global average pooling) and use them in PRM. Our position-aware recalibration module (PRM) can be formulated as

$$\begin{aligned} &\mathbf{y} = sigmoid\left(\mathcal{N}\left(\mathbf{S}\right)\right) \otimes \mathbf{x} \\ &s.t. \quad \mathbf{S} = \alpha\phi\left(\mathbf{x}, q\right) * \mathbf{D} + \beta\phi\left(\mathbf{x}, z\right), \\ &\qquad \mathbf{D} = f_p\left(\left|p_{\mathbf{x}} - p_q\right|\right), \end{aligned} \tag{1}$$

where $\phi(\cdot)$ is a similarity function; α and β are learnable parameters to balance the importance of dependencies to the most discriminative point and the dependencies to the global context; $p_{\mathbf{x}} \in \mathbb{R}^{2\times H*W}$ and $p_q \in \mathbb{R}^{2\times 1\times 1}$ are the position indexes of feature map $\mathbf{x}$ and query point q; $f_p\left(\cdot\right)$ is an encoding function of relative geometric position; function $\mathcal{N}$ means the similarity normalization; $\otimes$ indicates element-wise

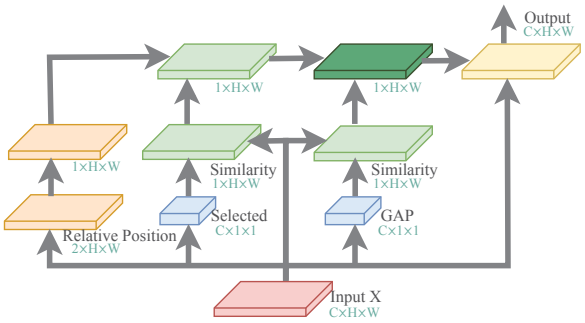

Figure 1: A basic Position-Aware Recalibration Module, which can be simply extended to multi-head PRM as presented in Section 3.5.

dot-product. For convenience, we calculate the relative geometric position between each point in $\mathbf{x}$ and query point q by absolute distance, that is $|p_{\mathbf{x}} - p_q|$. An example position-aware recalibration module is illustrated in Figure 1.

The central idea of Equation (1) is to learn the recalibration coefficients from both feature semantics and feature position. We then scale the coefficients to a range of $(0, 1)$ using a sigmoid function to recalibrate the input $\mathbf{x}$. In the following subsections, we will comprehensively describe each component and analyze the detail design.

3.1 Similarity Function

We model the feature dependencies of feature map $\mathbf{x}$ and query point q (or global context z) via their similarity $\phi(\mathbf{x}, q)$. Following the state-of-the-art work [Hu *et al.*, 2019; Wang *et al.*, 2018], we present several instantiations of the pairwise similarity function ϕ:

Cosine similarity. As a common approach used in information retrieval, we calculate the similarity as:

$$\phi(\mathbf{x}_i, q) = \frac{\mathbf{x}_i^{\mathsf{T}} q}{\max\left(\|\mathbf{x}_i\|_2 * \|q\|_2, \epsilon\right)}, \tag{2}$$

where ϵ is a small value to avoid division by zero. As an example, we set ϵ to 10^{-8}.

L1-norm similarity. A naive implementation of similarity is to measure the difference between two vectors. In this vein, the similarity function can be written as:

$$\phi(\mathbf{x}_i, q) = \sum_{c=1}^{C'} -\left|\mathbf{x}_i^c - q^c\right|, \tag{3}$$

where c is the channel index. We sum up the similarity of each channel to obtain the final similarity.

Dot-product similarity. We also consider the dot-product similarity, which can be presented as:

$$\phi(\mathbf{x}_i, q) = \mathbf{x}_i^{\mathsf{T}} q. \tag{4}$$

The above entities of similarity function ϕ showcase the flexibility of PRM. While various similarity functions can be explored, we empirically demonstrate that it is insensitive to the performance (see Table 3), indicating the robustness of our PRM. In the following experiments, we opt to choose Equation (4) as our default setting, unless otherwise noted.

3.2 Position Encoding

In addition to feature semantics, another essential aspect contributes to the presentation ability is the feature position, which, however, is overlooked in most networks. To this end, we explore the position encoding for our PRM.

To study the intrinsic properties of feature position, we first compute the pairwise relative position between all key point positions in $\mathbf{x}$ and the query point position to form a relative position map $\mathbf{D} \in \mathbb{R}^{2\times H\times W}$. For each pair of key-query point, the distance can be formulated as $\mathbf{D}_i = |p_{\mathbf{x}_i} - p_q|$, where p denotes the (x, y) geometric position. Next, we encode the distance map to a new embedding space. Driven by the observed phenomena in [Luo *et al.*, 2016] that the impact in a receptive field follows a Gaussian distribution, we encode the distance using a probability density function. We regard this transformation as our position encoding f_p:

$$f_p\left(|p_{\mathbf{x}} - p_q|\right) = \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\theta|p_{\mathbf{x}} - p_q|}{d}\right)^2}, \tag{5}$$

where learnable parameter $\theta \in \mathbb{R}^2$ redresses the scale of $|p_{\mathbf{x}} - p_q|$, and the learnable scalar d regulates the density curve of the Gaussian distribution. Compared with the learnable layers for positional encoding [Vaswani *et al.*, 2017; Hu *et al.*, 2019], our design is more interpretable and friendly for training. The encoded relative geometric position is then aggregated with the dependency $\phi(\mathbf{x}_i, q)$ by multiplication.

3.3 Semantic Normalization

We next regulate the semantic similarity using normalization statistics. Suppose the feature similarity map is $\mathbf{S} \in \mathbb{R}^{H\times W}$, we compute the mean μ and standard deviation σ by

$$\mu = \frac{1}{HW}\sum_{i=1}^{HW} \mathbf{S}_i, \quad \sigma = \left(\frac{1}{HW}\sum_{i=1}^{HW}\left(\mathbf{S}_i - \mu\right)^2\right)^{\frac{1}{2}}. \tag{6}$$

We normalize the similarity $\mathbf{S}$ over the spatial dimension, as employed in recent works [Li *et al.*, 2019a]:

$$\mathbf{S} = f_s(\mathbf{S}) = \frac{\mathbf{S} - \mu}{\sigma + \epsilon}, \tag{7}$$

where constant $\epsilon = 1e^{-5}$ is for numerical stability. Next, we apply an affine transformation which can be formulated as

$$\mathbf{S} = \lambda\mathbf{S} + \xi, \tag{8}$$

where λ and ξ are a pair of learnable parameters to avoid distribution shifting.

3.4 Recalibration

Finally, we recalibrate the intermediate feature map using normalized $\mathbf{S}$. We re-scale the normalized $\mathbf{S}$ to a range of $(0, 1)$ via a sigmoid function and recalibrate the input $\mathbf{x}$ using element-wise multiplication. Note that during the whole pipeline of PRM, we only query the dependencies for the selected query point q and global context z, which introduces few parameters. This indicates that our method is lightweight (see Table 1 for detailed complexity analysis). Thus, it is possible to integrate PRM into each block in a network.

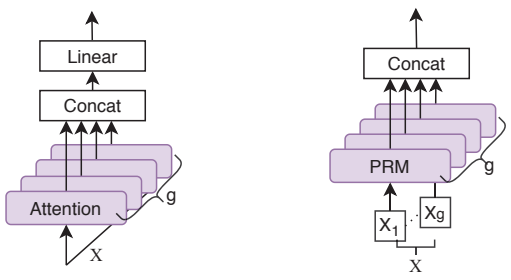

Figure 2: Left: multi-head attention; Right: multi-head PRM.

3.5 Multi-Head PRM

Our PRM learns the dependencies between the entire key map and a selected query point. We note that while it is relatively efficient to query the entire feature map for only one point, similar to [Cao *et al.*, 2019], the performance is compromised compared with that by querying for all key points. In this section, we close the gap between querying for particular one query point (e.g., [Hu *et al.*, 2018b; Cao *et al.*, 2019]) and for entire query map (e.g., [Wang *et al.*, 2018]).

Driven by the success of multi-head attention modules in machine translation [Vaswani *et al.*, 2017] and image recognition [Bello *et al.*, 2019], we consider the use of multi-head PRM, but operate in groups. The number of heads is denoted by g. We divide the input $\mathbf{x}$ into g groups, that is $\mathbf{x} = [x_1, x_2, ..., x_g]$. For each group $x_g \in \mathbb{R}^{\frac{C}{g} \times H \times W}$, we perform the PRM operation as aforementioned. We then concatenate the output of each group as the final recalibrated result. Figure 2 demonstrates the general modification. Qualitatively, assume we have g heads in our PRM, the number of parameters we introduced grows linearly g times, which is still negligible compared to the base network. By default, we set g to 64 in our experiments. In the sense of group operation, our multi-head PRM is similar to the groups design taken by [Li *et al.*, 2019a] and allows us to improve the performance further while providing a glimpse on its inherent distinction of each group.

4 Experiments

We present comprehensively experimental results to demonstrate the power of our method. For brevity, we evaluate the effectiveness of PRM on both low-level vision task (image recognition) and position-aware vision task (object detection). We compare our method with related state-of-the-art work and report the classification/detection results in this section. Our method has prevailed on these comparisons by a clear margin. Comprehensive ablation studies are conducted to provide an inherent insight into our method.

4.1 Image Recognition on ImageNet

We first evaluate our module for Image recognition task on ImageNet 2012 classification dataset [Russakovsky *et al.*, 2015]. We train all models on the training set using PyTorch [Paszke *et al.*, 2019] framework, and report the top-1 and top-5 classification accuracy on the validation set. For training, we follow the standard practice that randomly crops the images to a spatial resolution of 224×224 and horizontal flip images by a possibility of 50%. The input images are normalized by mean channel subtraction for both training and testing. We train all models from scratch using synchronous SGD with momentum 0.9 and weight decay 0.0001 for 100 epochs. All models are conducted on a server with 8 Tesla V100 GPUs, and each GPU has 32 images in a mini-batch (256 in total). The learning rate is initialized to 0.1 and divided by a factor of 10 every 30 epochs.

As shown in Table 1, the top-1 classification accuracy of ResNet is lifted by 1.75%. Compare with other work, we achieve a comparable or better performance with fewer parameters and FLOPs. For instance, we surpass SE-ResNet50 by 0.36% (77.64% *vs.* 77.28%) with even fewer parameters and FLOPs. In the context of same FLOPs and number of parameters, our PRM outperforms GE module and SGE module by a clear margin. Remarkably, our PRM-ResNet50 achieves a result that on pair with ResNet101(77.8659%), but only requires about half of the parameters and FLOPs.

We have showcased that PRM provides a broad performance gain for ResNet. To further explore the compatibility of our method, we next consider the introduction of PRM to various other CNN architectures. Without losing the generalization, we evaluate the effect of PRM on several different CNN architectures, including ResNet50 [He *et al.*, 2016], MobileNetV2 [Sandler *et al.*, 2018], and MnasNet_1.0 [Tan *et al.*, 2019]. In table 2, we see that our PRM can consistently improve the classification accuracy with minimal computational overhead, independent of the choice of base network architectures. All these results in Table 1 and Table 2 unambiguously demonstrate the efficiency of our PRM. In following subsections, we will present detail insights in our PRM.

4.2 Ablation Studies

Similarity functions. Our PRM is not limited to a particular similarity functions. We investigate three kinds of similarity functions, including cosine, L1-norm and dot-product similarity, as presented in Section 3.1. For generality, we evaluate the effect of similarity function on ResNet and MobileNetV2, and presented the results in Table 3. Across all three implementations we observe that the performances of different similarity functions are similar, just a tiny fluctuation of less than 0.06%. Similar phenomena are also shown in MobileNetV2 architecture. This observation indicates that our module is not sensitive to the selection of similarity functions, suggesting that the performance gain comes from the design rather than certain similarity instance.

Selected query point *vs.* Global Average Pooling. Recall the formulation of PRM in Equation 1, the similarity metric is mainly composed of two parts: the similarity to selected query point q and the similarity to the global average pooling z. Here, we explore the contribution rate of $\phi(\mathbf{x}, q) * \mathbf{D}$ and $\phi(\mathbf{x}, z)$. We quantify the contribution rate of similarity for selected query point as $rate = \frac{1}{g}\sum_i^g \frac{|\alpha_i|}{|\alpha_i|+|\beta_i|}$, where α_i and β_i indicates the α and β in ith group. We plot the contribution rate of each PRM (total 16) for ResNet50 in Figure 3. Somewhat surprisingly, the contribution rate is closely related

Model	top-1 acc.	top-5 acc.	FLOPs (G)	Parameters (M)
ResNet50 [He *et al.*, 2016]	75.8974	92.7224	4.122	25.557
SE-ResNet50 [Hu *et al.*, 2018b]	77.2877	93.6478	4.130	28.088
GE-ResNet50 [Hu *et al.*, 2018a]	76.2357	92.9847	4.127	25.557
CBAM-ResNet50 [Woo *et al.*, 2018]	77.2840	93.6005	4.139	28.090
SK-ResNet50 [Li *et al.*, 2019b]	77.3657	93.5256	4.187	26.154
GC-ResNet50 [Cao *et al.*, 2019]	74.8966	92.2812	4.130	28.105
SGE-ResNet50 [Li *et al.*, 2019a]	77.5072	**93.6783**	4.127	25.560
PRM-ResNet50 (ours)	**77.6474**	93.6418	4.128	25.560

Table 1: Comparison results of classification accuracy (%) and complexity on ImageNet. The best performances are marked in **bold**.

Models	PRM	top-1	FLOPs	Parameters
ResNet50	w/o	75.8974	4.122G	25.56M
	w/	77.6474	4.128G	25.56M
MobileNetV2	w/o	71.0320	0.320G	3.51M
	w/	72.5466	0.321G	3.51M
MnasNet	w/o	71.7195	0.330G	4.38M
	w/	73.0147	0.331G	4.38M

Table 2: The performance of PRM on different CNN architectures.

network	similarity function	top-1	top-5
Resnet50	Cosine	77.6517	93.6711
	L1-norm	77.6012	93.5944
	Dotproduct	77.6474	93.6418
MobileNetV2	Cosine	72.5374	90.8714
	L1-norm	72.5570	90.7993
	Dotproduct	72.5466	90.8960

Table 3: Ablation on similarity functions.

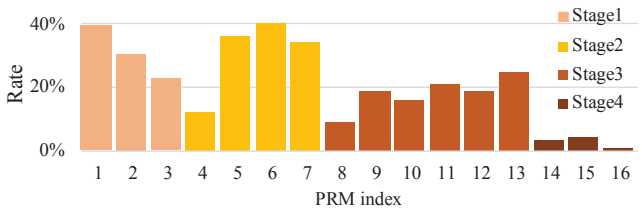

Figure 3: Contribution rate in PRM.

to the stage of the network. As can be observed, the rate is decreasing in stage 1; in stage2, it is always at a high level; in stage3, the proportion is relatively low and very stable, only about 20%; in the final stage4, the similarity of GAP almost dominates the entire similarity. We argue that this change is consistent with the philosophy of CNN intrinsically. In the shallow layers, the features produced by the layers should be very discriminative [Szegedy *et al.*, 2015]; while in the deep layers, all the features are semantically strong [Lin *et al.*, 2017a], indicating that all features contributes fairly.

Multi-head numbers. We test the influence of the head number in PRM. We test the numbers $\{8, 16, 32, 64, 128\}$ and analyze the influence on model performance [2]. All the variants of PRM are integrated to ResNet18. We plot the performance in Figure 4. Holistically, PRM consistently outperforms the baseline by a large margin, regardless of the choice of the head number. Meanwhile, the accuracy increases progressively as more heads are employed. Then it becomes saturated when g reaches 64. Thus, we set the heads number g to 64 by default in our experiments.

[2]We note that when g is too small, it takes much longer time for PRM to converge due to the similarity variance.

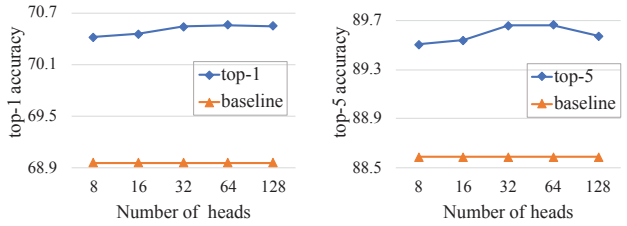

Figure 4: Influence of the head number in PRM.

Effectiveness of each component. We conduct experiments to study the intrinsic properties of our PRM design by incorporating each component sequentially to the base network, ResNet18. We evaluate the effectiveness of the following three components: feature dependency, position encoding, and normalization, which can be considered as independent components. Table 5 disentangles the effect of each component in our module. In line with our expectations, all the ablative variants of PRM are instrumental, improving the performance of original ResNet18 by a large margin. The ablation studies provide the following insights.

- Most of the performance gain comes from our feature dependency. Multiplying the weighted feature dependency can yield over absolute 1% improvement.
- The position encoding consistently improves the performance of PRM. In particular, our position encoding is highly convenient and succinct, only two learnable parameters are introduced for the Gaussian distribution.
- Combining all these components together, we achieve the best performance. This suggests that the streamline of our proposed module be empirically effective.

4.3 Object Detection on MS COCO

We further investigate PRM for object detection on the MS COCO benchmark. Following the discussion in [Ren *et al.*, 2015], we report the mean Average-Precision (mAP) for the

Detector	Backbone	$AP_{50:95}$	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	GMac	Parameters(M)
RetinaNet	ResNet50 [He *et al.*, 2016]	36.2	55.9	38.5	19.4	39.8	48.3	239.32	37.74
RetinaNet	SE-ResNet50 [Hu *et al.*, 2018b]	37.4	57.8	**39.8**	20.6	**40.8**	50.3	239.43	40.25
RetinaNet	PRM-ResNet50 (ours)	**37.7**	**58.4**	39.7	**21.4**	40.6	**50.7**	239.32	37.74
Cascade R-CNN	ResNet50 [He *et al.*, 2016]	40.6	58.9	44.2	22.4	43.7	54.7	234.71	69.17
Cascade R-CNN	GC-ResNet50 [Cao *et al.*, 2019]	41.1	59.7	44.6	23.6	44.1	54.3	234.82	71.69
Cascade R-CNN	PRM-ResNet50 (ours)	**42.5**	**61.2**	**46.2**	**24.2**	**45.8**	**56.4**	234.71	69.17

Table 4: Detection performance (%) using different backbones on the MS-COCO validation dataset. The best ones are highlighted in "**bold**".

Feature dependency	Position encoding	Normal -ization	top-1	top-5
			68.9632	88.5902
✓			70.1863	89.4491
✓	✓		70.2388	89.4970
✓		✓	70.4467	89.5653
✓	✓	✓	70.5616	89.6635

Table 5: Ablation studies on each component of PRM. We conduct these ablation experiments based on ResNet-18.

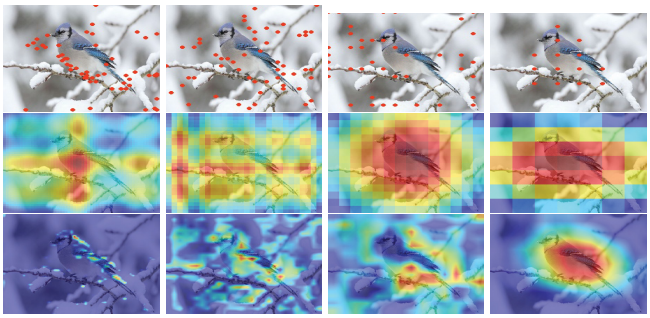

Figure 5: Query points visualization. **From top to bottom:** selected query points, the positional mask from all groups, and the attention map generated by Grad-CAM. **From left to right:** the corresponding results of each stage in ResNet50. Best viewed in color.

detection performance. We employ RetinaNet [Lin *et al.*, 2017b] and Cascade R-CNN [Cai and Vasconcelos, 2018] as detectors with a ResNet-50 backbone in the detection experiments. We apply our PRM and other modules to the ResNet backbone, modifying them in a similar way as what we have done in the ImageNet classification experiments. All models are trained for 24 epochs with pre-trained backbones that are listed in Table 1.

Without increasing the complexity, PRM improves RetinaNet by 1.5% mAP and Cascade R-CNN by 1.9% mAP. Our module performs better than others (SE module and GC module) by a large margin, with even fewer parameters and FLOPs. In particular, our PRM outperforms GC module by 1.4% mAP (42.5% *vs.* 41.1%) based on Cascade R-CNN detector. The promising results on object detection show that our PRM is conducive to significantly improve the performance of position-aware visual tasks.

4.4 Visualization and Analysis

So far, we evaluate the efficiency of PRM for different vision tasks and present experimental results to test the intrinsic properties. Next, we visually showcase the impact of PRM.

Since our module is a query-specific design, we first visually disentangle the selected query points in our module. To this end, we plot the selected query points in the intermediate feature maps. The allocations of query points are plotted in Figure 5 (top line as red points – some are overlapped). Clearly, most points locate closely to the target objects. Specifically, most points overlap in the last stage and enclose the target object. This indicates our selected query points are inherently discriminative and position-aware.

We continue to verify the efficiency of our simple position encoding. To calculate the positional mask, we leverage the average of all encoded relative positions (as presented in Equation 5) among all groups. We use linear interpolation to match the sizes of the positional mask and the original image. The positional masks at different stages are plotted in Figure 5 (the middle line). As a comparison, we also plot the attention map generated by Grad-CAM [Selvaraju *et al.*, 2017] (the bottom line). As we can see, the positional mask and attention map are generally similar, both caring more about the most distinct regions and ignoring the unrelated regions. This result shows that even a simple position encoding can effectively encode feature's positional information.

5 Conclusion

We present a new module to recalibrate the convolutional neural networks, which is named position-aware recalibration module (PRM). PRM selects the most discriminative query point(s) in the embedded query space and calculates the dependencies based on the feature semantics and feature position. While PRM is conceptually sophisticated, it is computationally efficient. With PRM integrated, we significantly improve the performance of various CNN modes on ImageNet and MS COCO benchmarks. Remarkably, our PRM outperforms SE, SK, *etc* by an observable margin, with fewer parameters and FLOPs. Comprehensive experiments and ablation studies have demonstrated and validated the efficiency of PRM. On all validation tests, integration of PRM yields noticeable improvement over baseline models. As a future work, we plan to investigate other position encoding designs and a new formulation of our multi-head PRM.

Acknowledgments

This research was supported in part by an NSF grant (CNS-1852134) and a grant from Fujitsu Labs of America. We thank Qi Chen, Jingda Guo, Sihai Tang, Dr. Qing Yang, Dr. Nannan Wang, and Dr. Xi Wang for their help and comments/suggestions. We also thank Google for providing access to the Google Cloud Platform.

References

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[Bello *et al.*, 2019] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, pages 3286–3295, 2019.

[Borji and Itti, 2012] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *TPAMI*, 35(1):185–207, 2012.

[Cai and Vasconcelos, 2018] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018.

[Cao *et al.*, 2019] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV Workshops*, pages 0–0, 2019.

[González and Woods, 1981] Rafael C. González and Richard E. Woods. Digital image processing. *TPAMI*, PAMI-3:242–243, 1981.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Hu *et al.*, 2018a] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *NIPS*, pages 9401–9411, 2018.

[Hu *et al.*, 2018b] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018.

[Hu *et al.*, 2019] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, pages 3464–3473, 2019.

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

[Islam *et al.*, 2020] Md Amirul Islam, Sen Jia, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *ICLR*, 2020.

[Li *et al.*, 2019a] Xiang Li, Xiaolin Hu, and Jian Yang. Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. *arXiv preprint arXiv:1905.09646*, 2019.

[Li *et al.*, 2019b] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *CVPR*, pages 510–519, 2019.

[Lin *et al.*, 2017a] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017.

[Lin *et al.*, 2017b] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[Luo *et al.*, 2016] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NIPS*, pages 4898–4906, 2016.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, pages 8024–8035, 2019.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626, 2017.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[Tan *et al.*, 2019] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2018] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018.

[Woo *et al.*, 2018] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, pages 3–19, 2018.

[Yu *et al.*, 2018] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, pages 5505–5514, 2018.