ATTENTION MEETS NORMALIZATION AND BEYOND

Xu Ma, Jingda Guo, Qi Chen, Sihai Tang, Qing Yang, Song Fu

University of North Texas, Denton, TX 76203

ABSTRACT

To make Convolutional Neural Networks (CNNs) more efficient and accurate, various lightweight self-attention modules have been proposed. In this paper, we systematically study state-of-the-art attention modules in CNNs and discover that self-attention mechanism can be closely related to normalization. Based on this observation, we propose a novel attention module, named Normalization-Attention module (NA module in short), which is almost parameter-free. The NA module calculates the mean and standard deviation of intermediate feature maps and processes the feature context with normalization, which makes a CNN model easier to be trained and more responsive to informative features.

Our proposed Normalization-Attention module can be integrated into various base CNN architectures, and used for many computer vision tasks, including image recognition, object detection, and more. Experimental results on ImageNet and MS COCO benchmarks show that our method outperforms state-of-the-art works using fewer parameters. Codes are made publicly available[1].

***Index Terms*—** Convolutional Neural Network, Self-attention, Normalization, Object detection, Image Recognition.

1. INTRODUCTION

Convolutional Neural Networks have shown promising performance in many vision tasks and applications [1, 2] due to the powerful feature representation ability. To further improve the performance of CNN architectures, one popular approach is to integrate self-attention mechanisms [3] into CNN models.

A self-attention mechanism is designed as a lightweight external module independent of the base CNN model. By aggregating the attention module, a base model is informed on where or what is most informative. That is, the attention mechanism distinguishes the importance of each feature in a feature map and biases toward the features with learned importance. In this vein, the self-attention mechanism works as emphasizing the importance between constituent features through appropriate weighting functions. However, recent studies on attention's interpretability [4, 5, 6] in NLP find that the attention weights are not closely related to features' importance. Experimental results show that *attention does not necessarily correspond to importance; they are only weakly and inconsistently related* [5]. Additionally, recent research in computer vision conducts empirical analysis of self-attention mechanisms. Rather than considering the importance of intermediate features, works in [7, 8, 9] design self-attention modules by utilizing normalization methods. Without bells and whistles, those designs achieve comparable performance as SENet and GENet, using fewer parameters and computation overhead.

Inspired by recent studies [7, 8, 9, 10, 11], we investigate the relations between self-attention modules in CNN models and normalization techniques and discover that there exists dependency between them. Following this observation, we develop a Normalization-Attention module (NA module in short) which is extremely lightweight (i.e., only introducing 32 parameters *vs.* 25.56M parameters in ResNet50). The NA module is conceptually interpretable and empirically powerful. We calculate the similarity between each query position and the global context, and normalize the obtained similarity. Then, a scaled dot-product operation is employed to aggregate the attention information and original feature maps.

Results from extensive experiments on large-scale vision benchmarks demonstrate the capacity of our NA module. With almost no additional parameters and FLOPs, our NA module improves the accuracy of ResNet50 By 1.41% on the ImageNet 2012 validation dataset. In addition to image recognition, the NA module generalizes well for other vision tasks. We improve the detection performance of Cascade R-CNN [12] by 2% mAP on the MS COCO dataset [13]. Most remarkably, our NA module improves the performance for small object detection by a large margin, i.e., a 2.2% improvement on RetinaNet and a 3.1% improvement on Cascade R-CNN. All these improvements on large-scale vision benchmarks demonstrate that the proposed Normalization-Attention module is efficient and can be applied to various vision tasks.

2. RELATED WORKS

Normalization. Normalization layers [10, 11, 14, 15] in deep neural networks have been extensively studied. In the traditional machine learning, an important assumption is that the

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

978-1-7281-1331-9/20/$31.00

data distributions of the source domain and the target domain are consistent. However, a convolutional layer could shift the feature distribution. As more and more convolutional layers are used in a deep CNN model, such a shift intensifies. Thus, there exists a departure from the basic assumption in CNN models, making the models more difficult to train. To mitigate this discrepancy, Ioffe et al. introduced Batch Normalization [10] to neural networks. By scaling and shifting the output of each convolutional layer, Batch Normalization can effectively adjust feature distribution and alleviate the gradient vanishing problem. Inspired by Batch Normalization, a series of normalization algorithms have been proposed. To overcome the problem of a small batch size in Batch Normalization, Layer Normalization [15] normalizes all channels for each sample and avoids the dependency on a batch of samples. Group Normalization [11] divides channels into several groups and normalizes each group individually.

Self-Attention. Self-attention mechanisms [3] have been used for computer vision and attract more attention. To boost representation capability of CNN models, lightweight attention modules have been proposed. SENet [16] uses a squeeze-and-excite module to investigate the channel correlation of intermediate feature maps. SKNet [17] reveals the importance of the receptive field (RF) in a convolutional layer and leverages a self-attention mechanism to adjust RF sizes adaptively. AANet [18] employs an attention module as augmentation of the vanilla convolutional layer. Although these attention modules are lightweight, the convolutional layers and fully-connected layers in the attention modules still use a large number of parameters and are compute intensive.

Most recently, researchers put more interests in developing nearly parameter-free self-attention designs. A well-known example is the Spatial Group-wise Enhance (SGE) module [7]. SGE groups a C-channel feature map into G groups and calculates a global semantic feature of each group using mean in the channel dimension. Then a normalization operation is employed to prevent bias among samples. Similar to SGE, LCT [8] and SRM [9] explore extremely lightweight attentions.

3. RETHINKING OF ATTENTION MODULES

Generally speaking, a self-attention mechanism can be described as a mapping of a query and a set of key-value pairs to an attention map [3]. A basic dot-product self-attention can be expressed as follows.

$$out = \mathbf{softmax}(\Phi(f_{\theta_q}(Q), f_{\theta_k}(K)))f_{\theta_v}(V), \qquad (1)$$

where f_{θ_q}, f_{θ_k} and f_{θ_v} and embedding functions, and Φ is a pairwise similarity function.

Non-Local module [19] advocates that each point x_i in a feature map contributes to a query point x_j. Specifically, the Non-Local module can be presented as:

$$out_j = \sum_{j=1}^{W*H} \frac{e^{f_{\theta_q}(x_i)^T f_{\theta_k}(x_j)}}{C(X)} f_{\theta_v(x_j)}, \qquad (2)$$

where $C(X)$ is a normalization factor. In [19], $C(X)$ equals $\sum_{\forall_j} \Phi(x_i, x_j)$.

Since the Non-Local module needs to learn query-independent global context, the computational overhead is high, which makes it impossible to apply the module to every block in a CNN model. To address this problem, a Global Context (GC) module [20] modifies the Non-Local module by relaxing the requirement of the key position. That is, a GC module learns a global attention map and applies it to all query position. This modification drastically reduces the compute overhead by $W \times H$ times, with minimal accuracy loss. The GC module normalizes pairwise relation between the query position and all positions, and it can be expressed by

$$out_j = \sum_{j=1}^{W*H} \frac{e^{f_{\theta_k}(x_j)}}{\sum_{i=1}^{H*W} f_{\theta_k(x_j)}} f_{\theta_v(x_j)}. \qquad (3)$$

Furthermore, the SE module removes the query position from the attention mechanism and calculates the global context by $\sum_{j=1}^{W*H} \frac{e^{f_{\theta_k}(x_j)}}{H*W}$.

Comparing Non-Local, GC, and SE modules, we discover that 1) the relation of key-query pairs becomes more relaxed, and 2) the value embedding function f_{θ_v} is never considered. Inspired by the recent works on context modeling and the those on normalization layers, we develop our Normalization-Attention module, which is extremely lightweight and empirically powerful, suitable for various vision tasks.

4. NORMALIZATION-ATTENTION MODULE

We present Normalization-Attention module (NA module), a plug-in module that is compatible with various base CNN networks. The NA module models the global context using global similarities. Then a normalization method is employed to normalize the global context. We adopt the dot-product self-attention formulation to develop our module.

Context Modeling. As described in the previous section, a self-attention module considers the relationship between key and query positions. However, learning relationship of all pairs is computationally expensive and may compromise the effectiveness for aggregation to each convolutional block, for example in a Non-Local Network. In order to get the contextual information, we model the contextual similarity. For each query position x_i, we quantify the similarity between the query position and the global context $\mathbf{C}$ as follows.

$$\mathbf{C} = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{X}_{h,w}, \qquad (4)$$

where h and w indicate the spatial position. We define our contextual similarity function $\Phi(\cdot)$ as :

$$\Phi(x_i, \mathbf{C}) = -\left\|x_i - \mathbf{C}\right\|_1. \tag{5}$$

Other similarity functions, such as dot product $x_i^{\mathrm{T}}\mathbf{C}$ and squared similarity $-\left(x_i - \mathbf{C}\right)^2$, can also be explored. We use a generic approach for context aggregation and compute the final context g by

$$g = \sum_{i}^{H\times W} \frac{\Phi(x_i, \mathbf{C})}{\sum_{i}^{H\times W} \Phi(x_j, \mathbf{C})}. \tag{6}$$

Equation (6) can be easily implemented using a softmax function. In the self-attention formulation, i.e., Equation (1), we note that our global context modeling is a typical implementation of the self-attention mechanism.

Normalization. Different from the traditional processing of g that uses fully-connected layers or convolutional layers, we explore normalization approaches, as they are widely used [10, 11, 7, 9].

Given the global context $g \in \mathbb{R}^C$, we calculate the mean μ and standard deviation σ using

$$\mu = \frac{1}{C}\sum_{m}^{C} g_m, \quad \sigma = \left(\frac{1}{C}\sum_{m}^{C}\left(g_m - \mu\right)^2 + \epsilon\right)^{\frac{1}{2}}. \tag{7}$$

Here, ϵ is a small constant for numerical stability and we set $\epsilon = 1e^{-5}$. We normalize g by an affine transformation function:

$$\hat{g} = \gamma\frac{g - \mu}{\sigma} + \beta, \tag{8}$$

where γ and β are two learnable parameters and they are the only additional parameters introduced in our design. When γ equals the standard deviation and β equals the mean, then $\hat{g}$ becomes identical to g. We obtain the output $\mathbf{Y}$ of the NA module with the dot-product attention as

$$\mathbf{Y} = \mathbf{X} \odot sigmoid(\hat{g}), \tag{9}$$

where $\odot$ is element-wise multiplication.

Efficiency. Our NA module is extremely lightweight and each module adds only two parameters. When applying the NA module to ResNet50 in a case study, we only add *32 parameters* (i.e., 2×16) in total. Compared to the *25.56M parameters* in vanilla ResNet50, the number of extra parameters is negligible. The NA module is computationally efficient as well. The most time-consuming operation is the calculation of the standard deviation. Compared to the fully-connection layers or convolutional layers in other self-attention modules, the computation overhead of the NA module is negligible since we only calculate the mean value and standard deviation in our module. In addition to parameter and computation efficiency, the proposed NA module is implementation-friendly. It can be easily implemented in the main CNN frameworks, such as PyTorch, TensorFlow, and MXNet.

5. PERFORMANCE EVALUATION

We have conducted comprehensive experiments to evaluate the effectiveness of our NA module for improving the performance of CNN models across a number of tasks and datasets. Specifically, we experiment on ImageNet [21] to verify the improvement of the NA module on the state-of-the-art base CNN models for image recognition, and compare with other self-attention modules. In pursuit of profound insight, we conduct ablation experiments on the NA module for better understanding of its characteristics. We also conduct experiments on the MS COCO [13] benchmark to examine the effectiveness of our NA module for object detection.

5.1. Image Classification on ImageNet

We evaluate the object recognition performance of the NA module on ImageNet. The ImageNet dataset consists of 1K classes with 1.3M training images and 50k testing images. Following the common practice in [2], we perform random-size cropping of images to 224×224, and horizontally randomly flip images with a probability of 0.5. We train networks from scratch using synchronous SGD with a weight decay of 0.0001 and a momentum of 0.9. All experiments are conducted on a server with 8 Tesla V100 GPUs. For ResNet [2] and its variants, we train the networks for 100 epochs with a batch size of 256 (i.e., 32 images per GPU), stating at a learning rate of 0.1 and decreasing it by 10 every 30 epochs. For lightweight CNN models such as MobileNetV2 [22], we train the networks for 150 epochs with a batch size of 512 (i.e., 64 images per GPU), stating at a learning rate of 0.1 and adjusting it using a cosine decay method [23].

We select two base CNN models: ResNet50 and MobileNetV2, as representations of a normal model and a lightweight model, respectively. For comparison, we select several state-of-the-art self-attention modules, including SENet [16], GENet [24], SKNet[17], and CBAM[25]. The metrics that We measure are Top-1 and Top-5 accuracy for evaluating the performance of the aforementioned modules, and the numbers of FLOPs and parameters for comparing the efficiency of the tested modules.

From the results in Table 1, we find that a simple NA-module offers a comparable performance as the SE module (slightly beyond) with almost no additional computation overhead. Compared to other competitive self-attention modules, the NA module introduces the least number of parameters and FLOPs while achieving a promising performance. Compared to vanilla ResNet50, we achieve an improvement of 1.41% Top-1 accuracy on the ImageNet validation dataset. Moreover, our NA-MobileNetV2 outperforms the vanilla MobileNetV2 by 1.23%. The appealing results indicate that our NA module provides a new schema for self-attention mechanism design, which is not only efficient but also powerful.

Model	Top-1 Acc (%)	Top-5 Acc (%)	GFLOPs	Parameters
ResNet50 [2]	75.8974	92.7224	4.122	25.557M
SE-ResNet50 [16]	77.2877	**93.6478**	4.130	28.088M
GE-ResNet50 [24]	76.2357	92.9847	4.127	25.557M
SK-ResNet50 [17]	**77.3657**	93.5256	4.187	26.154M
CBAM-ResNet50 [25]	77.2840	93.6005	4.139	28.090M
NA-ResNet50 (ours)	77.3078	93.6096	**4.122**	**25.557M**
MobileNetV2 [2]	71.0320	90.0670	0.320	3.505M
SE-MobileNetV2 [16]	72.0482	90.5812	0.321	3.563M
GE-MobileNetV2 [24]	**72.2776**	**90.9120**	0.322	3.555M
CBAM-MobileNetV2 [25]	71.9069	90.5114	0.324	3.565M
NA-MobileNetV2 (ours)	72.2596	90.7498	**0.320**	**3.50M**

Table 1. Single crop classification accuracy (%) on the ImageNet validation set. We train all models using the PyTorch framework. The best performance is highlighted in "**bold**" and the second-best are highlighted in "blue".

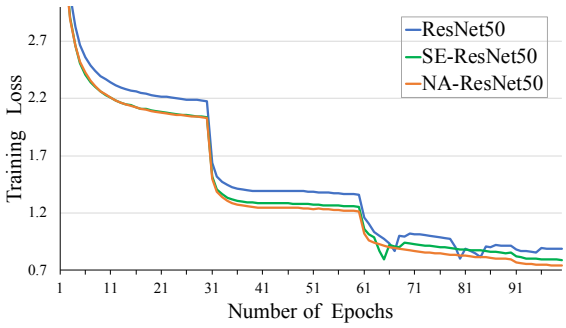

Fig. 1. Training loss curves on the ImageNet validation dataset. Our proposed NA module makes ResNet50 easier to train and achieve lower training loss.

Model	Top-1	GLOPs	Parameters
ResNet50 [2]	75.8974	4.122	25.557M
NA (w embedding)	77.2183	4.123	25.558M
NA (w/o embedding)	77.3078	4.122	25.557M

Table 2. Embedding ablation.

Model	Top-1	GLOPs	Param
ResNet50 [2]	75.8974	4.122	25.557M
NA (mean)	76.901	4.122	25.557M
NA (ours)	77.3078	4.122	25.557M

Table 3. Context modeling ablation.

5.2. Ablation Studies

Next, we investigate the impact of NA module on the performance of CNN models. To this end, we thoroughly evaluate each component of the NA module.

Training loss. We measure the training loss of vanilla ResNet50, SE-ResNet50, and our NA-ResNet50. As illustrated in Figure 1, our NA-ResNet50 yields a lower training loss compared to both SE-ResNet50 and vanilla ResNet50, which infers that our NA-ResNet50 possesses a better feature representation capability. We also find that during the later epochs (see the 60th to 100th epochs), the training loss of SE-ResNet50 and vanilla ResNet50 fluctuates. In contrast, the loss curve of our NA-ResNet50 remains stable and smooth during the entire training period. This is attributed to the fundamental normalization step in our NA module.

Embedding *vs.* No-Embedding. As presented in Section 4, we calculate the context similarity without embedding key positions and the global context to a new feature space. To study the influence of embedding function f_{θ_q} and f_{θ_k} in our NA module, we implement the embedding function using two convolutional (we set the kernel size to 3.) layers and a ReLU activation function[2]. The comparison results are presented in Table 2. To our surprise, the embedding NA module performs slightly inferior (i.e., 77.2183% *vs.* 77.3087%) than the NA module without embedding function, although it is still better than vanilla ResNet50 by 1.32%.

Context modeling *vs.* Mean value. We compute the context modeling by calculating the similarities between query positions and global context. A native substitution is directly employing the mean value of the input feature map. As shown in Table 3, both pipelines outperform the vanilla ResNet50, suggesting that the main contribution comes from our normalization operation in self-attention module. Meanwhile, our context modeling method achieves 0.4% better performance than using mean values.

Heatmap visualization. We next visually investigate our NA module. We present the heatmap of the last convolutional layers in a ResNet50 model and corresponding variants using Grad-CAM [27]. We compare our NA module with SE module and the vanilla ResNet50. The results are plotted in Figure 2. Consistent with our expectations, all models correctly pay

[2]We note that using a point-wise convolutional layer for the embedding function may cause our module to become non-convergent. Hence, we employ the channel sharing convolutional embedding.

Detector	Backbone	$AP_{50:95}$	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	GMac	Parameters
RetinaNet [26]	ResNet50 [2]	36.2	55.9	38.5	19.4	39.8	48.3	239.32	37.74M
RetinaNet [26]	SE-ResNet50 [16]	**37.4**	57.8	**39.8**	20.6	40.8	**50.3**	239.43	40.25M
RetinaNet [26]	NA-ResNet50 (ours)	37.3	**57.9**	39.5	**21.6**	**40.9**	49.2	239.32	37.74M
Cascade R-CNN [12]	ResNet50 [2]	40.6	58.9	44.2	22.4	43.7	54.7	234.71	69.17M
Cascade R-CNN [12]	GC-ResNet50 [20]	41.1	59.7	44.6	23.6	44.1	54.3	234.82	71.69M
Cascade R-CNN [12]	NA-ResNet50 (ours)	**42.6**	**61.3**	**46.4**	**25.5**	**46.2**	**55.3**	234.71	69.17M

Table 4. Object detection performance (%) with different backbones on the MS-COCO validation dataset. The best performance is marked as "**bold**" and the second best is marked as "blue".

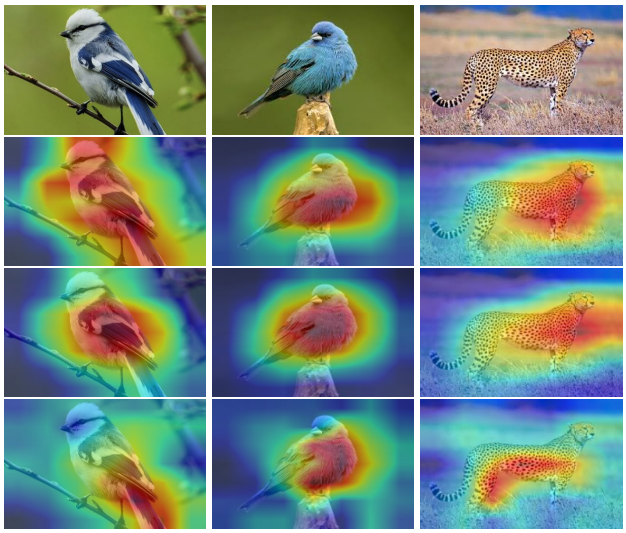

Fig. 2. Heatmap visualization. **Top line:** original images; **Second line:** heatmap of vanilla ResNet50; **Third line:** heatmap of SE-ResNet50; **Bottom line:** heatmap of our NA-ResNet50. Best viewed in color.

attention to the target objects and ignore the trivial regions. However, we emphasize that while it is easy to pay attention correctly, learning the most discriminative features are difficult for CNN models. We can see that vanilla ResNet50 pays attention to a vast region that encompasses the target object, while SE-ResNet50 pays attention to the central part of an object. Different from them, our NA-ResNet50 strictly focuses on the most discriminative parts of an object. It biases the location of the most informative and discriminative features and simultaneously suppresses the irrelevant regions.

5.3. Object Detection on MS COCO

Our NA module likewise improves the performance of CNN models in image recognition tasks, but further benefits high-level vision tasks. To demonstrate the generalizability of our NA module, we conduct experiments on MS COCO dataset for the object detection task. We employ two state-of-the-art architectures, RetinaNet [26] and Cascade R-CNN [12], as our detectors, using the open-sourced mmdetection framework [28]. We measure the average precision of bounding box detection on the challenging COCO 2017 dataset [13]. The input images are resized to make the shorter side to be 800 pixels. We adopt the settings used in [1] and train all models with a total of 16 images per mini-batch (that is 2 images per GPU). We select ResNet50, and its variants as the detection backbones and all backbones are pre-trained using ImageNet benchmark and directly taken from Table 1. All detection models are trained for 24 epochs using synchronized SGD with a weight decay of 0.0001 and a momentum of 0.9. We initialize the learning rate to 0.02 for Cascade R-CNN and 0.01 for RetinaNet as previous work [12, 26]. The learning rate is reduced by a factor of 10 at the 18th and 22nd epochs during the training.

We present detection results in Table 4. With only additional 32 parameters and negligible computational overhead, our NA module improves RetinaNet by 1.1% mAP and Cascade R-CNN by 2.0% mAP. Unambiguously, our NA module performs better than other self-attention modules (SE module and GC module) by a large margin. An important property of our approach is that it is more sensitive to small objects. Using RetinaNet, we improve the small object detection performance by 1% mAP (21.6% *vs.* 20.6%) compared to SE-ResNet50. Using Cascade R-CNN, we improve the small object detection performance by 1.9% mAP (25.5% *vs.* 23.6%) compared to GC-ResNet50.

6. CONCLUSION

In this paper, we systematically study several state-of-the-art attention modules for improving CNN feature representation ability and propose an extremely lightweight attention module, named as Normalization-Attention module, to improve the performance of base CNN models with negligible additional parameters and FLOPs. Instead of computing each pair of key and query positions in the traditional self-attention mechanism, our method computes the similarity between each query position and global context. Next, A straight-forward normalization is applied to the integrated global similarities. Normalization-Attention module is computationally efficient but empirically powerful. In particular, our Normalization-Attention module improves ResNet50 by 1.41% on the ImageNet image recognition task, with 32 pa-

rameters increase and almost no additional FLOPs. When applying our NA module to high-level vision tasks like object detection, we improve Cascade R-CNN by 2.0% mAP.

7. REFERENCES

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[4] Sarthak Jain and Byron C Wallace, "Attention is not explanation," in *NAACL*, 2019, pp. 3543–3556.

[5] Sofia Serrano and Noah A. Smith, "Is attention interpretable?," *arXiv preprint arXiv:1906.03731*, 2019.

[6] Sarah Wiegreffe and Yuval Pinter, "Attention is not not explanation," in *EMNLP*, 2019, pp. 11–20.

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

[8] Dongsheng Ruan, Jun Wen, Nenggan Zheng, and Min Zheng, "Linear context transform block," *arXiv preprint arXiv:1909.03834*, 2019.

[9] HyunJae Lee, Hyo-Eun Kim, and Hyeonseob Nam, "Srm: A style-based recalibration module for convolutional neural networks," *arXiv preprint arXiv:1903.10829*, 2019.

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

[11] Yuxin Wu and Kaiming He, "Group normalization," in *ECCV*, 2018, pp. 3–19.

[12] Zhaowei Cai and Nuno Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *CVPR*, 2018, pp. 6154–6162.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, 2014, pp. 740–755.

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

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

[16] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[17] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang, "Selective kernel networks," in *CVPR*, 2019, pp. 510–519.

[18] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le, "Attention augmented convolutional networks," *arXiv preprint arXiv:1904.09925*, 2019.

[19] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He, "Non-local neural networks," in *CVPR*, 2018, pp. 7794–7803.

[20] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu, "GCNet: Non-local networks meet Squeeze-Excitation networks and beyond," *arXiv preprint arXiv:1904.11492*, 2019.

[21] 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*, vol. 115, no. 3, pp. 211–252, 2015.

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[23] Ilya Loshchilov and Frank Hutter, "SGDR: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[24] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi, "Gather-Excite: Exploiting feature context in convolutional neural networks," in *NIPS*, 2018, pp. 9401–9411.

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

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2980–2988.

[27] 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*, 2017, pp. 618–626.

[28] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al., "Mmdetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.