Aggregation with Feature Detection

Shuyang Sun[1] Xiaoyu Yue[2] Xiaojuan Qi[3] Wanli Ouyang[4] Victor Prisacariu[1] Philip Torr[1]
[1]University of Oxford [2]Centre for Perceptual and Interactive Intelligence
[3]University of Hong Kong [4]The University of Sydney

Abstract

Aggregating features from different depths of a network is widely adopted to improve the network capability. Lots of modern architectures are equipped with skip connections, which actually makes the feature aggregation happen in all these networks. Since different features tell different semantic meanings, there are inconsistencies and incompatibilities to be solved. However, existing works naïvely blend deep features via element-wise summation or concatenation with a convolution behind. Better feature aggregation method beyond summation or concatenation is rarely explored. In this paper, given two layers of features to be aggregated together, we first detect and identify where and what needs to be updated in one layer, then replace the feature at the identified location with the information of the other layer. This process, which we call DEtect-rePLAce (DEPLA), enables us to avoid inconsistent patterns while keeping useful information in the merged outputs. Experimental results demonstrate our method largely boosts multiple baselines e.g. *ResNet, FishNet and FPN on three major vision tasks including ImageNet classification, MS COCO object detection and instance segmentation.*

1. Introduction

Representation learning is considered to be the engine for most applications in the field of computer vision. By stacking basic blocks with different connectivities, deep networks can be designed for various tasks, *e.g.* image classification [15, 18, 46, 34, 35, 16, 48, 25, 7, 45], object detection [20, 34] and semantic segmentation [30].

Deep networks [15, 18, 34, 20, 30, 46] that fuses features from different depths have benefited many applications in computer vision. It is often the case that features from deeper layers are better geared towards the final prediction because they contain high-level semantics harvested from large contextual region. However, recent works have found that there may be information loss when forward-propagating data throughout the network [15, 13, 18]. Thus skip connections [15] or dense connections [18] were proposed to consecutively fuse features from previous layers to prevent such loss of information and better leverage the capacity of the network. The above facts indicate that the features at all depths are crucial for building up a network with better performance.

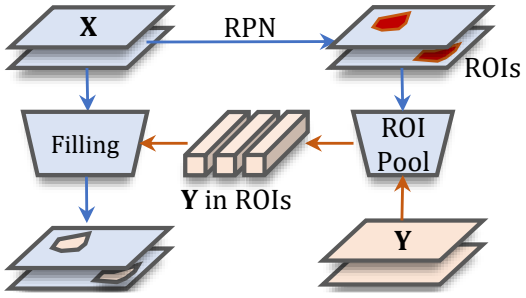

Figure 1: **Brief description of DEPLA.** Given an identity feature $\mathbf{X}$ and a candidate feature $\mathbf{Y}$, our method could first locate where and what need to be updated in the identity feature $\mathbf{X}$, and then migrate the information from the candidate feature $\mathbf{Y}$ using the learnt locations.

Existing methods [15, 18, 20, 46, 22] however largely focus on features of which layers should be combined, and paid little attention to *how* to combine them. To the best of our knowledge, state-of-the-art backbones [28, 36] for feature extraction still follow the conventional design in [15]. They use either the element-wise sums to fuse the features together, or a concatenation/convolution [18]. However, as the semantic meaning of different features may vary, network activation at the same position but on different feature maps may capture very different information. The aforementioned element-wise sum or concatenation cannot discover the inconsistency or the incompatibility between features during the feature aggregation. A method that can identify *which location to aggregate* is needed.

In this study, we aim to address this problem by designing a pipeline that can learn to detect the most appropriate part of feature for aggregation. As shown in Figure 1, we reformulate the entire feature aggregation process as a two-stage process. Specifically, we first detect where to update, then aggregate the selected locations using the detected fea-

tures. With our detect-then-aggregate design, the information retained in the summarized features is only related to the feature within the ROIs, skipping the irrelevant and inconsistent patterns outside the ROIs. The features related to the ROIs of one layer will be finally merged into the features of another layer by Back-Filling Network (BFN). We name the entire module as DEtect-rePLAce (DEPLA) according to the process.

We choose to embed our network into the current prevalent ResNet [15], SE-ResNet [17], FishNet [34] and the Feature Pyramid Network [20] to evaluate our method for short-range, long-range and multi-scale feature aggregation. Experimental results on ImageNet [8] and COCO [21] show that, by simply inserting our module into these baselines, our approach can steadily boost their accuracy on major vision tasks, including image classification, object detection and instance segmentation. Notably, when embedded in ResNet, our method can outperform all other ResNet-based counterparts compared under similar computational complexity.

2. Related Works

Deep Network Architecture Design. The success on the challenge ILSVRC [8] of AlexNet [19] signalled the resurgence of the deep learning era. However, sequential connected networks like [32, 35] soon meet the bottleneck of depth. In order to solve this problem, He et al. [15, 44] introduced residual learning that sums up the features within the same resolution into deep networks to enable networks to be extremely deep. An alternative approach to solve this problem is to densely concatenate the layers in the same stage [18]. As a combination, DPN [5] integrates both the connectivity of the DenseNet and ResNet. Recent works follow further improve the design of [15] by adjusting the hyper-parameters like height, width, and bottleneck ratio. All these works propose to aggregate features via a element-wise summation or concatenation.

Long-Range Feature Aggregation in Deep Networks. Apart from aggregating features of the same resolution, DLA [46] concatenates the information from different sources together. U-Net [30] also proposes to concatenate features from the low-level to the high-level for medical image segmentation, which also achieved great success in that field. FishNet [34] as a versatile backbone preserves features by concatenation and then let them refine each other through a simple residual block [15]. To obtain better features for object detection, Feature Pyramid Networks (FPN) [20] fuse both the transformed features from the top-down lateral convolutions and the bottom-up weighted pyramid through a simple sum operation. Based on FPN, several extensive works [11, 41, 22, 26] explore new possibility on connectivity between scales. Some works like [23] discuss the problems in feature's magnitude, which now could be solved by applying normalization onto the merged features. To summarize, all these networks directly merge the features from variant depths without any guidance, which causes noise and useful fragments will not be distinguished.

Attention Models. Our method can be also regarded as a variant of the attentive models. Attention has been proven to be effective in many application in the field of deep learning [17, 38, 9, 39, 6, 42, 4]. By calculating and applying a context-based encoding summarized from a specific dimension of the feature itself, all these methods are named as self attention models. Works including SENet [17], A^2Net [4] and GloRe [6] explore how to gather and learn the encodings on channel dimension, resulting in less redundancy and imbalance among channels. The non-local neural networks [42] and Attention Augmented networks (AA) [2] integrate a transformer-like [38] structure into CNN and associate the pixel-wise information of a feature map. SAN [49] is a novel architecture that only learns to extract features from images with attention modules and adopts a patch-based local relative attention module to replace the convolution kernels defined in conventional CNNs. All these models are complementary to our method as they are designed for self-refinement, while we aim at aggregating features from two different layers.

3. Aggregation with Feature Detection

The inputs of our DEPLA module are two feature maps $\mathbf{X}$ and $\mathbf{Y}$ from two layers. Here, $\mathbf{X} \in \mathbb{R}^{C_1 \times H \times W}$ is called the identity feature and $\mathbf{Y} \in \mathbb{R}^{C_2 \times H \times W}$ is called the candidate feature. C_1 and C_2 respectively denote the number of channels for $\mathbf{X}$ and $\mathbf{Y}$. The goal of DEPLA is to use the candidate feature $\mathbf{Y}$ to refine the identity feature $\mathbf{X}$ and produce the aggregated feature. As shown in Figure 2, the DEPLA module can be roughly divided into three submodules: the soft Region Proposal Network (SoftRPN), Soft ROI Pooling, and Back-Filling Network (BFN). First, SoftRPN detects where the features should be aggregated and produces the RoI Maps $\mathbf{M}$. Then, soft ROI Pooling summarizes feature map $\mathbf{Y}$ into a compact RoI-wise feature $\mathbf{Z}$ based on the ROI maps $\mathbf{M}$. Further, BFN back-fills the summarized feature $\mathbf{Z}$ to corresponding learnt spatial locations. The output of BFN is finally aggregated with the identity feature $\mathbf{X}$.

3.1. Soft Region Proposal Network

As shown in Figure 2, the Soft Region Proposal Network (SoftRPN) takes the identity feature $\mathbf{X}$ as input and outputs a set of ROI maps $\mathbf{M}$ with size $L \times D \times H \times W$, where $H \times W$ is the spatial size and L represents the number of ROI groups. Each ROI group contains D candidate ROI maps for each channels. The value at each location implies how likely this location is going to be replaced by features from $\mathbf{Y}$.

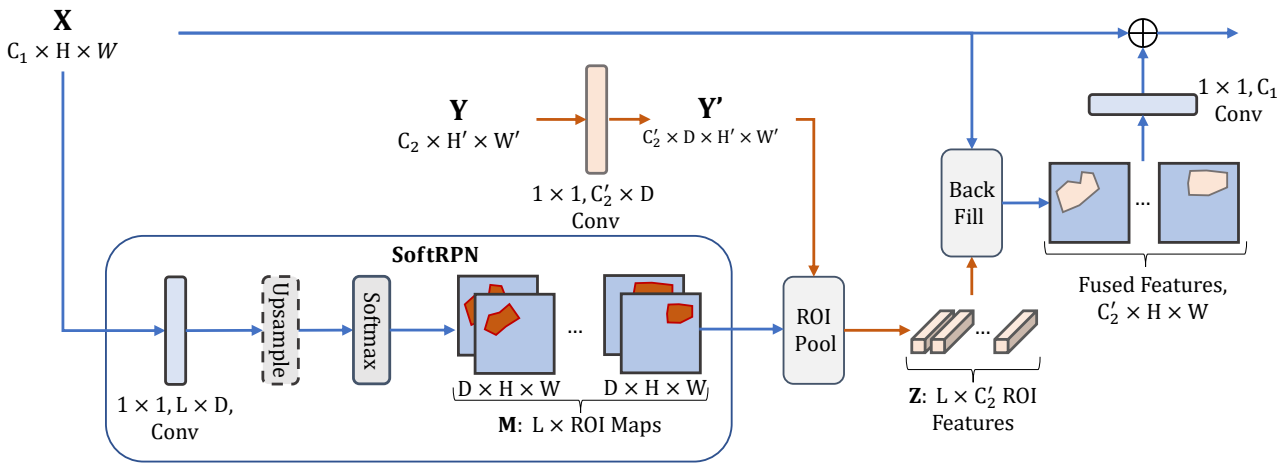

Figure 2: **Overview of the DEPLA module.** The whole process can be divided into three parts: (1) The identity feature $\mathbf{X}$ is fed into SoftRPN to determine where and what features in $\mathbf{X}$ need to be updated by the candidate feature $\mathbf{Y}$. (2) The generated ROI maps $\mathbf{M}$ are then fed into the soft ROI Pooling to extract the corresponding features from $\mathbf{Y}$ and output the summarized features $\mathbf{Z}$. (3) The summarized features $\mathbf{Z}$ from soft ROI Pooling are processed by the Back Filling Network (BFN). "$1 \times 1, L \times D$ Conv" represents a convolution layer with kernel size (1×1) and number of output channels $(L \times D)$. The symbol $\bigoplus$ represents element-wise summation. Blocks with dashed boundary represent optional operations.

To generate these ROI maps $\mathbf{M}$, we feed $\mathbf{X}$ to a 1×1 convolution layer to produce score maps $\tilde{\mathbf{M}}$. When the spatial sizes of $\mathbf{X}$ and $\mathbf{Y}$ are different, we will apply a bilinear up-sampling layer to the smaller one to align their spatial sizes. A Softmax operation with temperature T is applied to the reshaped 2D score maps $\tilde{\mathbf{M}} \in \mathbb{R}^{(L\times D)\times(H\times W)}$ to yield the confidence of each pixel as Equation (1), where a higher value implies a higher probability of the pixel belonging to a ROI.

$$\mathbf{M}_{i,j} = \frac{\exp\left(\tilde{\mathbf{M}}_{i,j}\right)}{\sum_{l}^{H\times W} \exp\left(\frac{\tilde{\mathbf{M}}_{l}}{T}\right)}, \tag{1}$$

where $\mathbf{M} \in [0,1]^{(L\times D)\times(H\times W)}$ represents the ROI maps. M is also the output of SoftRPN after reshaping to $L \times D \times H \times W$. The hyper-parameter T controls the distribution of the confidence values. When T becomes lower, the confidence distribution will become sharper, making the area of the activated ROI smaller.

But the regions with low confidence scores contribute little to aggregating features in the following step of Soft ROI Pooling. Therefore, low-confidence ROIs will have a low influence during the feature aggregation.

3.2. Soft ROI Pooling

The aim of Soft ROI Pooling is to summarize the information from the candidate features $\mathbf{Y}$ using the positional clues $\mathbf{M}$ learnt from SoftRPN. As shown in Figure 2, the module taking the following two inputs: 1) the ROI maps $\mathbf{M}$ of size $(L \times D) \times H \times W$, which is the output of SoftRPN; and 2) the transformed candidate feature $\mathbf{Y}' \in \mathbb{R}^{C'_2\times D\times H\times W}$, which is transformed from $\mathbf{Y}$ by a 1×1 convolution. The output is the summarized ROI-wise features $\mathbf{Z} \in \mathbb{R}^{L\times C'_2}$ which captures what our method believes to be the most useful parts in $\mathbf{Y}'$ for aiding $\mathbf{X}$ during the feature aggregation. In this way, the inconsistent patterns in $\mathbf{Y}'$ that do not lie in ROIs are ignored and hence have low influence on feature aggregation.

As shown in Figure 3, we mask the feature $\mathbf{Y}'$ using the ROI maps $\mathbf{M}$ as: 1) each $D \times H \times W$ feature from $\mathbf{Y}'$ is element-wisely multiplied with each $D \times H \times W$ confidence map $\mathbf{M}$; and 2) each element-wisely multiplied result of size $D \times H \times W$ are summed into a scalar that contains the summarized feature that aggregates the information from all $H \times W$ spatial locations and D ROIs. For the L groups of ROI maps and C'_2 groups of features, there are $L \times C'_2$ summarized features as the output of the Soft ROI Pooling. With the element-wise multiplication, only confident ROIs contribute to feature summarization while those identified to be less useful with lower confidences in the ROI maps will be less counted.

3.3. Back Filling Network (BFN)

Equipped with the summarized ROI-wise feature $\mathbf{Z}$, we design a back-filling operation to map $\mathbf{Z}$ back into appropriate spatial locations of the identity feature $\mathbf{X}$ such that the feature aggregation can be finalized.

To merge $\mathbf{Z}$ into $\mathbf{X}$, one can simply add the summarized feature to all pixels of $\mathbf{X}$, however, this ignores the fact that the summarized features $\mathbf{Z}$ are relevant to the ROIs and can

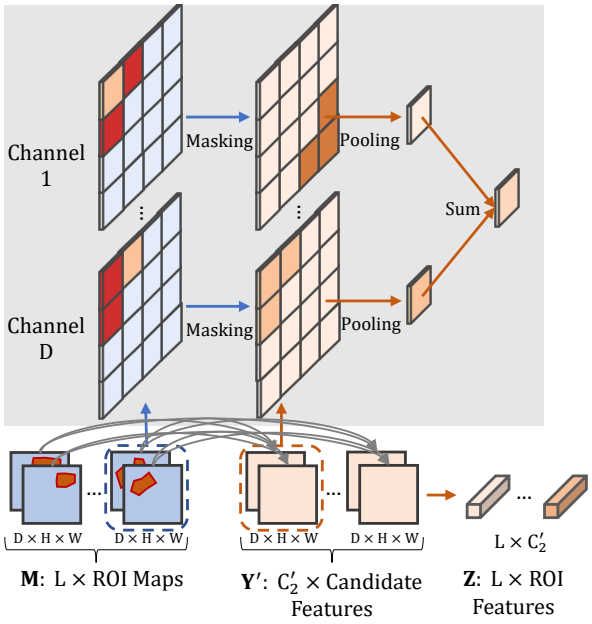

Figure 3: **Soft ROI Pooling.** Element-wise multiplication between the ROI maps generated from SoftRPN and the transformed candidate feature $\mathbf{Y}'$ are used for selecting features from ROIs. Features for each group of ROI maps $\mathbf{M}$ (a ROI group containing D ROI maps) and each group of candidate features will be summed up to a single scalar. There are L groups of ROI maps and C_2' groups of transformed candidate features $\mathbf{Y}'$. Therefore, the output is $L \times C_2'$ summarized features.

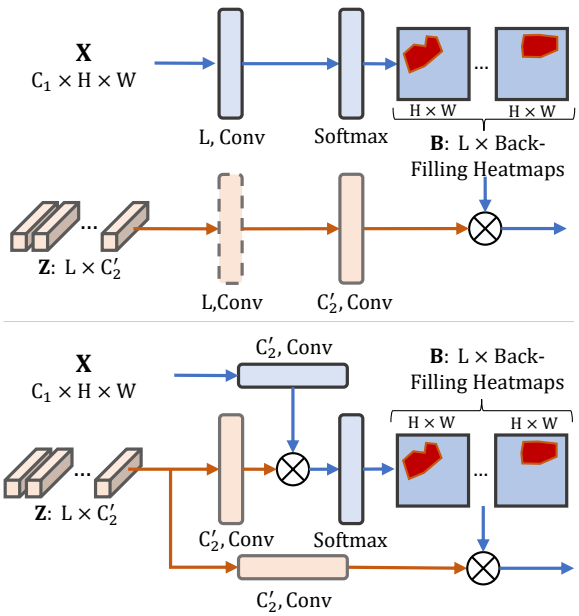

Figure 4: **Back-Filling Network (BFN).** The back-filling heatmap is generated in this network for guiding the $L \times C_2'$ summarized features (in orange) to the appropriate spatial locations. The guidance is achieved by multiplying the back-filling heatmap with the summarized feature. Top: plain BFN that learns to generate the back-filling heatmaps only based on the confidence maps of ROIs. Bottom: Adaptive BFN that calculates the back-filling heatmaps according to the affinity between the identity features $\mathbf{X}$ and the summarized features. $\otimes$ denotes the matrix multiplication, and all convolutions are with kernel size 1×1. Operations including transpose, reshape *etc*. are omitted in the figure.

be only useful for aggregation at suitable locations of $\mathbf{X}$. To achieve this goal, we design a Back-Filling Network (BFN) to generate a set of heatmaps which projects the summarized features back to the suitable spatial locations. Similar to the soft ROIs, each location of these *back-filling heatmaps* has a confidence value indicating how likely the summarized features from $\mathbf{Y}$ will be set here. Information from $\mathbf{Z}$ are repeated over the spatial dimensions of the back-filling heatmaps, and are multiplied with the confidence at each location so that the information from the summarized features are placed at appropriate locations, which are further refined by a 1×1 convolution layer and combined with the identity feature $\mathbf{X}$ to produce the final output of the DEPLA module. We propose two strategies to generated the back-filling heatmaps, namely, *Plain BFN* and *Adaptive BFN*, which are elaborated as follows.

Plain BFN. As shown on the top of Figure 4, Based on the confidence maps of ROIs $\mathbf{M}$ (output of SoftRPN), we generate the back-filling heatmaps $\mathbf{B} \in [0,1]^{L \times H \times W}$ using one convolution layer followed by softmax normalization. Note that the Softmax operation is applied on the spatial dimension. Another 1×1 convolution is applied here to transform $\mathbf{Z}$. After the back-filling heatmaps $\mathbf{B}$ are obtained, we need to map the ROI features $\mathbf{Z}$ accordingly based on the location clues given by $\mathbf{B}$. If we regard the set of heatmaps as a convolution kernel with shape $(H \times W) \times L$, where $H \times W$ represents the spatial kernel size and L represents the number of channels, the back-filling operation can be also treated as a de-convolution process with a dynamically generated kernel $\mathbf{B}$ on the transformed $\mathbf{Z}$.

Adaptive BFN. In Plain BFN, the pooled candidate feature may still have some inconsistency with the identity feature $\mathbf{X}$ as they only contain information about $\mathbf{X}$. Therefore, we design Adaptive BFN as shown at the bottom row of Figure 4. We first transform the identity feature $\mathbf{X}$ and also the ROI-wise feature $\mathbf{Z}$ into a same latent space with C_2' channels using two independent 1×1 convolutions. Then we can generate the back-filling heatmaps by calculating a dot-product between the two transformed features. With a softmax operation applied on the spatial dimension, the back-filling heatmaps $\mathbf{B} \in [0,1]^{L \times H \times W}$ are obtained. The lateral back-filling operation, which can be regarded as a

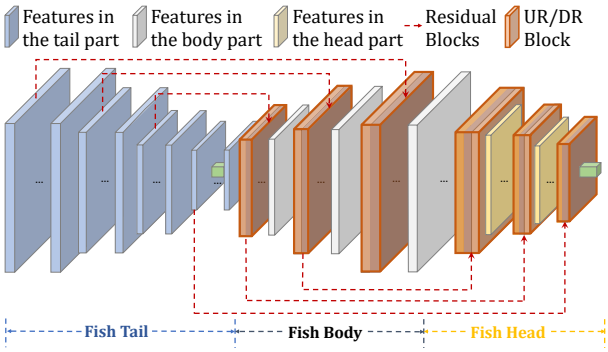

Figure 5: Overview structure of the FishNet. We insert our DEPLA module in the UR and DR blocks (rendered in red) where the feature aggregation happens.

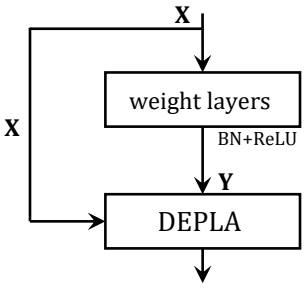

Figure 6: **Integrate DEPLA into residual block**

de-convolution operation treating $\mathbf{B}$ as the de-convolution kernel and the transformed $\mathbf{Z}$ as the input, is identical to the one proposed in plain BFN. Ablation studies on these alternative designs are conducted in the experimental section.

4. Embedding DEPLA into CNNs

To validate that DEPLA can be generalized to different feature aggregation approaches, we choose to embed DEPLA into ResNet [15], SE-ResNet [17], FishNet [34] and the Feature Pyramid Network [20] to evaluate DEPLA for short-range, long-range and multi-scale feature aggregation.

4.1. Integration with ResNet

Here we present how DEPLA can be integrated popular CNN architectures. We first choose the ResNet [15] as one of our baselines and reconstruct the ResNet and use DEPLA module to merge the features with the same resolution but varied in depths. In this paper, we divide the ResNet into 4 independent stages by their resolution from high to low. Specifically, we insert the DEPLA module into three residual blocks of the stage 3 regardless of the depth of the network. The way we integrate DEPLA into the residual block is shown in 6, where the identity of the residual block serves as $\mathbf{X}$ and the outputs of the residual weight layers serve as $\mathbf{Y}$. As for the hyper-parameters, we set L=16 C_2'=16 and

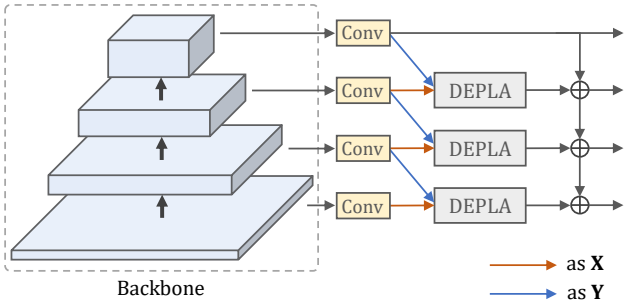

Figure 7: **Integrating DEPLA with FPN.** We add the DEPLA between every two adjacent scales. The notation Conv here means 1×1 convolution layer for dimension reduction.

D=4. Temperature of the softmax function is set to 1 as default for all ResNet-based models.

4.2. Integration with FishNet

We next embed our module into FishNet [34] to study the effect of DEPLA module in long-range feature aggregation. FishNet is divided into three parts, including the tail, body, and head, where long-range feature aggregation is applied among the three parts to help the network learn multi-scale representations. As shown in 5, we embed the DEPLA into the UR and DR-blocks of the FishNet as the low-level and the high-level features will be aggregated in these blocks. Typically, features in UR and DR blocks can have about 20-60 convolution layers in between.

4.3. Integration with FPN

We show how DEPLA can be integrated into the Feature Pyramid Network architecture [20] to study the effect of DEPLA on multi-scale features for object detection. Details are shown in Figure 7. Since FPN generates features of all scales for the final prediction, and merges the information reversely from the high-level to the low-level, for each scale, we adopt the feature from the higher level as the candidate feature $\mathbf{Y}$, and the feature at the current scale as the identity feature $\mathbf{X}$. Note that we adopt the adaptive BFN instead of the plain BFN to generate the back-filling heatmaps in FPN.

5. Experiments

5.1. Implementation Details on ImageNet

We conduct experiments for image classification on the ImageNet 2012 classification dataset [31] that includes 1000 classes. There are 1.2 million images for training and 50 thousands images for validation (denoted by ImageNet-1k val). We implement our method using the popular deep learning framework PyTorch [27]. We apply two different

Model	FLOPS (G)	Top-1 Acc (%)
ResNet-50 [15]	4.1	76.2
ResNet-101 [15]	7.8	77.4
ResNet-152 [15]	11.6	78.3
SEResNet-50 [17]	4.1	76.9
SEResNet-50++ [33]	4.1	79.4
SEResNet-101 [17]	7.9	77.7
SEResNet-101++ [33]	7.9	81.4
RegNetY-4G [28]	4.0	79.4
RegNetY-4G++ [37]	4.0	80.0
RegNetY-8G [28]	8.0	79.9
RegNetY-8G++ [37]	8.0	81.7
LambdaResNet50 [1]	6.1	79.3
LambdaResNet101 [1]	13.5	81.9
ResNet-50-DEPLA	4.5	**80.6**
SEResNet-50-DEPLA	4.5	**81.3**
SEResNet101-DEPLA	8.4	**82.1**

Table 1: Comparison with other ResNet-based models. ++ means well-tuned re-implementations.

training strategies to take both better performance and fair comparison into account.

Standard Training Strategy. For naïve implementations, our training and testing details mainly follow what adopted in [28], including the 100 epochs training schedule, with the cosine annealing optimization policy [24], standard data augmentation *e.g*. random resized crop and random flipping, label smoothing[7] *etc*. Note that the results of all re-implemented baselines in Table 2 are better than those reported in the original papers.

Better Training Strategy. As lots of works adopt better training strategy for better performance [43, 12, 17], we also report the results using the following tricks to the standard training srtategy:

(1) Longer training schedule with 400 epochs.
(2) Auto-augmentation.
(3) Warming up the learning rate from 0 for 5 epochs.
(4) Label smoothing [7] with smooth factor 0.1.
(5) Mixup training [47].

Note that in this paper, only results for DEPLA-based models lying in Table 1 are trained using the better training strategy. All other results on ImageNet are trained with standard training strategy.

5.2. Experimental results on ImageNet

DEPLA outperforms other ResNet-based counterparts. As shown in Table 1, with better training strategy, the DEPLA module can achieve 4.4% gain in terms of the Top-1 accuracy when integrated with the prevalent baseline ResNet-50. When embedded with the highly competitive baseline SEResNet-50, our method can still achieve 4.7% gain. We also compare with the SEResNet-50++, which are well-tuned strong baselines with much better performance (2% higher than what reported in the original paper) using heavy data augmentation and regularization [33]. We observe that even when the baselines are strong, the SEResNet50-DEPLA can still outperform the improved baseline by a remarkable 1.9%, which makes it even comparable to SEResNet-101++ with much lower FLOPS. Notably, SEResNet101-DEPLA can also outperform the SEResNet101 by a remarkable 1.7%, which indicates that DEPLA module is complementary to the Squeeze-Excitation module.

Model	FLOPS (G)	Top-1 Acc (%)
ResNet-50* [15]	4.1	76.9
SEResNet-50* [17]	4.1	77.5 (+0.6)
AA-ResNet-50[2]	4.1	77.7 (+0.8)
LambdaResNet-50 [1][1]	6.1	78.2 (+1.3)
GloRe-ResNet-50 [6]	5.2	78.4 (+1.5)
ResNet-50-DEPLA	4.5	78.6 (+1.6)
SEResNet-50-DEPLA	4.5	79.1 (+2.2)

Table 2: Comparison of attention-based models on ImageNet classification with a ResNet50 architecture trained under standard training strategy. * indicates our re-implementation using the standard training strategy. [1] The FLOPS of LambdaResNet50 is calculated under a smaller scope size $\|m\|$=7 $\times$ 7. The default setting in the original paper will lead to 10G FLOPS.

For the comparison with the state-of-the-art baselines of the ResNet family under similar model capacity, we compare our model with the latest LambdaResNet and RegNetY (*w/* SE). As shown in Table 1, our ResNet50-DEPLA can outperform the LambdaResNet50 by a clear 1.3% while being 1.7G lower in FLOPS. SEResNet50-DEPLA and SEResNet101-DEPLA can also outperform the corresponding well-tuned RegNet counterparts (RegNetY-4G++ and RegNetY-8G++) that are also embedded with SE modules. Concretely, the top-1 accuracy of SEResNet50-DEPLA is higher than RegNetY-4G++ by 0.6% and SEResNet101-DEPLA can surpass RegNetY-8G++ by 0.4%. Note that the key differences between ResNet and RegNet are just the network hyper-parameters (design space) like width, depth and bottleneck ratio. It is a promising direction to apply DEPLA into RegNet architectures for future work.

Comparison under the standard training strategy. Table 2 exhibits the results comparing DEPLA under the standard training strategy with other counterparts of the ResNet family. Note that all the listed results are trained under similar schedule, regularization and data augmentations. Both ResNet50-DEPLA and SEResNet50-DEPLA can out-

Model	FLOPS (G)	Top-1 Acc (%)
ResNet-50 [34]	4.1	76.9
w/ DEPLA-Plain-BFN	4.5	78.6 (+1.7)
w/ DEPLA-Adaptive-BFN	4.9	78.8 (+1.9)

Table 3: Comparison between the Plain-FRN and the Adaptive-FRN on ImageNet. We choose Plain FRN due to the cost-accuracy trade-off.

	T	Object Detection $AP^d/AP^d_S/AP^d_M/AP^d_L$
ResNet50 *w/* FPN [14]	-	38.0/21.7/41.4/50.6
w/ FPN-DEPLA	0.2	38.9/**23.8**/42.7/49.7
	0.5	**39.1**/23.6/**42.9**/**50.4**
	1.0	39.0/23.5/42.7/49.9

Table 4: Comparison between the FPN and the FPN-DEPLA with different temperature T. The ROI pooling is going to be sharper when T becomes lower.

perform all other counterparts and show a clear boost (1.6% and 2.2%) over the baseline results.

5.3. Plain BFN *vs*. Adaptive BFN

Shown in Table 3, the adaptive BFN is able to outperform the plain BFN by 0.2% on ImageNet. Despite the better performance in terms of accuracy, the adaptive BFN has higher FLOPS compared with the plain BFN. Therefore, we choose the plain BFN as the final setting when DEPLA is embedded into ResNet.

5.4. Experimental investigations on MS COCO

We conduct experiments on object detection and instance segmentation to demonstrate the generalization capability of DEPLA based on the FPN [20] and FishNet [34]. All experiments are implemented on mmdetection [3].

Dataset and Evaluation Metrics. Experiments are conducted on the challenging MS COCO [21] dataset. It consists of 115k images for training (*train-2017*) and 5k images (*val-2017*) for validation. We train models on *train-2017* and report the results on *val-2017*. All reported results follow standard COCO-style Average Precision (AP) metrics which include AP (averaged over IoU thresholds) and AP_S, AP_M, AP_L (AP at different scales).

Implementation Details for FPN-DEPLA. We integrate DEPLA into FPN to study the effect of DEPLA when applied on multi-scale feature aggregation. When the DEPLA is embedded into FPN [20], we take 16 images in one batch for 12 epochs (1x) in the training phase, with the base learning rate of 0.02. The learning rate is decayed after 8 and 11 epochs by a factor of 0.1. A ResNet-50 is applied before the FPN for feature extraction. Note that here the ResNet-50 is **not** integrated with DEPLA. Without loss of generality, experiments are conducted base on the FPN Mask R-CNN with ResNet50 [14] as the backbone, we simply replace the FPN with our FPN-DEPLA to verify the effectiveness of the FPN-DEPLA module.

Implementation Details for FishNet150-DEPLA. FishNet-DEPLA is proposed to validate the efficacy of DEPLA for long-range feature aggregation. The entire model is first pretrained on ImageNet, and the learning rate for FishNet-DEPLA is set to 0.015. Sync-batch-norm is applied here to stabilize the training process. We train detectors for 24 epochs with an initial learning rate of 0.02, and decrease it by 0.1 after 20 and 22 epochs respectively. All other hyper-parameters follow the settings in the mmdetection [3] if not specifically noted.

DEPLA can be well generalized on multi-scale feature aggregation for object detection. Table 4 compares our proposed FPN-DEPLA module with FPN. We observed that equipped with FPN-DEPLA, Mask R-CNN outperforms its FPN counterpart by 0.9% with only a small additional computational cost. Considering that only minor changes are applied to the standard FPN architecture, the improvement of AP demonstrates the generalization capability of our DEPLA module. The improvement in small objects is considerably significant (1.6% - 1.9%), which further proves the efficacy of the feature aggregation that DEPLA applies.

The effect of the temperature T. As shown in Table 4, we also conduct experiments to study the effects when different temperature T is applied to soft ROI Pooling defined in Equation 1. When T is lower, the distribution of heatmap is closer to a one-hot tensor, and the activated area will become more restricted, which simulates the max-pooling. The results shown in Table 4 validates our assumption that as the AP of small objects comes to be higher when T is lower.

DEPLA can be well generalized on long-range feature aggregation for object detection and instance segmentation. We evaluate the effect of DEPLA when integrated with FishNet-150 that aggregating features from varying depths. We insert FishNet150-DEPLA into two frameworks Faster-RCNN and Mask-RCNN as backbones for object detection and instance segmentation. The experimental results on object detection are implemented on FPN-based Faster R-CNN [29]. The lateral connections and top-down pathway in FPN are attached to the same place as FishNet. Note that here DEPLA is not embedded into FPN for fair comparison. According to the results shown in Table 5, FishNet150-DEPLA obtains a 2.5% absolute AP increase compared with FishNet150, and a 2.0% absolute AP increase when compared with ResNet-101. Note that our model is lighter than all models compared in Table 5 except for FishNet-150.

		Instance Segmentation	Object Detection	
		Mask R-CNN	Mask R-CNN	Faster R-CNN
Backbone	GFLOPS	$AP^s/AP^s_S/AP^s_M/AP^s_L$	$AP^d/AP^d_S/AP^d_M/AP^d_L$	$AP^d/AP^d_S/AP^d_M/AP^d_L$
ResNet-101 [15]	7.8	37.7/20.0/41.3/51.9	42.2/24.4/46.1/55.5	41.1/24.0/44.9/53.7
ResNeXt-101($32 \times 4d$) [44]	8.0	37.8/19.8/41.4/51.9	42.2/23.5/46.1/56.0	41.2/23.9/44.9/54.3
HRNetv2p-W32 [40]	8.3	37.8/20.8/40.9/51.5	42.5/24.7/46.1/55.6	41.4/24.1/44.8/53.6
HRNetv2p-W40 [40]	16.1	38.2/20.5/41.2/52.0	42.8/24.9/46.2/56.2	42.1/24.7/45.7/55.0
Res2Net-101 [10]	8.3	38.7/20.6/42.0/53.2	43.6/24.8/47.2/57.9	43.0/25.0/47.2/56.3
FishNet150 [34]	6.45	37.0/19.8/40.2/50.3	41.5/24.1/44.9/55.0	40.6/23.3/43.9/53.7
FishNet150-DEPLA	7.3	**39.4/22.4**/42.9/52.6	**44.1/27.1**/48.0/56.2	**43.1/27.0**/46.8/55.0

Table 5: Average Precision (%) of instance segmentation and object detection with different backbones on MS COCO *val-2017* . AP^s_* and AP^d_* denote the average precision for segmentation and detection respectively, and AP^*_S, AP^*_M, AP^*_L denote the AP for small, medium and large objects respectively. The FPN column are integrated with Faster-RCNN [29]. The GFLOPS are calculated with input size 224×224. We show that FishNet-DEPLA is the best at solving the small cases compared with other backbones.

We further use FishNet-DEPLA as the backbone of Mask R-CNN. As shown in Table 5, by embedding the DEPLA into the FishNet-150, the network can remarkably improve both the mask AP and bbox AP for small objects ($1.4\%, 1.3\%$ *vs*. FishNet and $2.4\%, 2.7\%$ *vs*. ResNet-101), demonstrating the effectiveness of merging features. As for the average precision, our FishNet-DEPLA could achieve higher performance on both tasks compared with the a series state-of-the-art hand-crafted backbones *e.g*. HRNet [40] and Res2Net [10].

FishNet150-DEPLA shows great performance dealing with the hard small objects. As shown in Table 5, we can observe that the FishNet150-DEPLA is particularly good at finding the hard small cases compared with other backbones. Specifically, FishNet150-DEPLA can outperform FishNet-150, ResNeXt101-($32\times4d$) by 3% and 3.6% for object detection when embedded into Mask-RCNN.

Above all, all these experimental results showcase that DEPLA is also valid and helpful for all major tasks including image classification, object detection and instance segmentation. Besides, DEPLA can be successfully applied onto backbones designed for different purpose, which further demonstrates the generalizability of DEPLA.

5.5. Visualization

We visualize feature maps of FishNet150-DEPLA to see how DEPLA works between the high-level and low-level features. From Figure 8 we can see that DEPLA can effectively merge the precise information around the ski pole from the low-level information to the high-level features. This demonstrates that DEPLA can selectively find useful features from different layers for aggregation instead of naïvely summing them together.

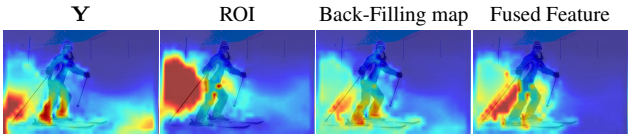

Figure 8: Visualization of FishNet-DEPLA. DEPLA module migrates the precise information around the ski pole from the low-level candidate feature $\mathbf{Y}$ while ignoring the noise and inconsistency on the bottom-right corner.

6. Conclusion

In this study, we propose DEtect-rePLAce (DEPLA) that learns to keep the helpful features while removing those inconsistent patterns during feature aggregation. Extensive experiments show that our DEPLA module can be successfully integrated with backbones using short-range (ResNet, SEResNet), long-range (FishNet) and multi-scale (FPN) feature aggregation. Experiments also show that DEPLA steadily boost the baseline performance on all three major visual tasks including image classification, object detection and instance segmentation. We hope our research could shed some lights on the problem of feature aggregation.

Acknowledgement. We would like to thank Jiangmiao Pang and Andrew Gambardella for proofreading and helpful comments. This work is supported by the EPSRC Turing AI Fellowship EP/W002981/1, EPSRC/MURI grant EP/N019474/1. We would also like to thank the Royal Academy of Engineering and FiveAI. Wanli Ouyang is supported by Australian Research Council Grant DP200103223, FT210100228, Australian Medical Research Future Fund MRFAI000085, and SenseTime.

References

[1] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *ICLR*, 2021. 6

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, pages 3286–3295, 2019. 2, 6

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *NeurIPS*, pages 352–361, 2018. 2

[5] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *NIPS*, pages 4470–4478, 2017. 2

[6] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Shuicheng Yan, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *CVPR*, 2019. 2, 6

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1251–1258, 2017. 1, 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. IEEE, 2009. 2

[9] Jun Fu, Jing Liu, Haijie Tian, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. *arXiv preprint arXiv:1809.02983*, 2018. 2

[10] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *T-PAMI*, 2019. 8

[11] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019. 2

[12] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[13] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 1

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2980–2988. IEEE, 2017. 7

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 2, 5, 6, 8

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CVPR*, 2018. 2, 5, 6

[18] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 2

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2, 5, 7

[21] 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*, pages 740–755. Springer, 2014. 2, 7

[22] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 1, 2

[23] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 2

[24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018. 1

[26] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, pages 821–830, 2019. 2

[27] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[28] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, pages 10428–10436, 2020. 1, 6

[29] 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. 7, 8

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 1, 2

[31] 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. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[33] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 6

[34] Shuyang Sun, Jiangmiao Pang, Jianping Shi, Shuai Yi, and Wanli Ouyang. Fishnet: A versatile backbone for image, region, and pixel level prediction. In *NeurIPS*, pages 762–772, 2018. 1, 2, 5, 7, 8

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019. 1

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 6

[38] 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. 2

[39] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, pages 3156–3164, 2017. 2

[40] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *T-PAMI*, 2020. 8

[41] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *CVPR*, pages 11943–11951, 2020. 2

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, pages 7794–7803, 2018. 2

[43] Junyuan Xie, Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, and Mu Li. Bag of tricks for image classification with convolutional neural networks. *arXiv preprint arXiv:1812.01187*, 2018. 6

[44] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995. IEEE, 2017. 2, 8

[45] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *ICCV*, pages 1284–1293, 2019. 1

[46] Fisher Yu, Dequan Wang, and Trevor Darrell. Deep layer aggregation. *arXiv preprint arXiv:1707.06484*, 2017. 1, 2

[47] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 6

[48] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018. 1

[49] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, pages 10076–10085, 2020. 2