Differentiable Attention

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-attention has been widely used in deep learning, and recent efforts have been devoted to incorporating self-attention modules into convolutional neural networks for computer vision. Previous approaches usually use fixed channels to compute feature affinity for self-attention, which limits the capability of selecting the most informative channels for computing such feature affinity and affects the performance of downstream tasks. In this paper, we propose a novel attention module termed Differentiable Attention (DA). In contrast with conventional self-attention, DA searches for the locations and key dimension of channels in a continuous space by a novel differentiable searching method. Our DA module is compatible with either fixed neural network backbone or learnable backbone with Differentiable Neural Architecture Search (DNAS), leading to DA with Fixed Backbone (DA-FB) and DA-DNAS respectively. We apply DA-FB and DA-DNAS to two computer vision tasks, person Re-IDentification methods (Re-ID) and image classification, with state-of-the-art results on standard benchmarks and compact architecture compared to competing methods, revealing the advantage of DA.

1 Introduction

Self-attention, with its success in natural language processing, has recently drawn increasing interest beyond the NLP literature. Efforts have been made to introduce self-attention to deep convolutional neural networks (CNNs) for computer vision tasks with compelling results. The success of self-attention in computer vision is arguably attributed to its capability of capturing fine-grained cues and important parts of objects in images, which is particularly helpful for downstream tasks such as person Re-IDentification methods (Re-ID) and image classification. For example, non-local neural network [1] employs self-attention to aggregate input features to attention enhanced features by weighted summation of the input features. The weights in the weighted summation are the pairwise feature affinity, which is computed as the dot product between input features. Lacking an effective way of selecting channels, previous works [1, 2] use fixed channels to computer such feature affinity, and such fixed channels are selected by handcrafted pooling and sampling. As a result, the selected channels may not be the most informative ones for the downstream tasks.

We argue that more informative channels should be selected in the attention modules to calculate more meaningful affinities among the features. In this paper, we propose a novel Differentiable Attention (DA) module which searches for the most informative channels in a differentiable manner. Figure 1 illustrates the difference between the vanilla self-attention and the proposed DA module. The main contributions of this paper are as follows.

First, we propose the Differentiable Attention (DA) module. In contrast with conventional self-attention where fixed channels are used to compute pairwise similarity between input features, DA selects the most informative channels to compute task-oriented pairwise affinity, which outperforms the vanilla self-attention modules by extensive empirical study. DA employs a novel differentiable searching algorithm which learns the position and key dimension of the most informative channels

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

in the input features. In contrast with Gumbel-softmax based searching limited to a fixed number of options, DA searches for the location and key dimension in a continuous space comprising uncountably infinite options for these two parameters. While location and key dimension are integers with respect to which the loss function of the neural network is not differentiable, we extend these two parameters to real value domain by carefully designed bilinear interpolation, which enables differentiable optimization. The location and key dimension of the selected channels form a window, and the channels inside the window found by DA are used in the inference process of neural networks with DA modules. Since DA with a single window risks loosing informative channels, we further extend Single-Window DA to Multi-Window DA so as to further boost the performance of DA. A natural window merging process is used to merge overlapping windows obtained by Multi-Window DA, which adaptively infers the number of final disjoint windows.

Second, DA modules are incorporated into either Fixed neural network Backbone or learnable backbone with Differentiable Neural Architecture Search (DNAS) algorithm, leading to DA-FB and DA-DNAS respectively. DA-DNAS is a new neural architecture search method jointly learning the network backbone and the architecture of DA, that is, the location and key dimension of channels. We apply DA to two computer vision tasks, person Re-ID and image classification with extensive empirical study. DA-FB and DA-DNAS not only outperform current state-of-the-art, but also render much more compact architecture compared to competing methods. Notably, DA achieves the mean Average Precision and top-1 accuracy of 61.0% and 82.7% with only 12.8% of the FLOPs of the model with best precision so far. We also have interesting findings which are of independent interest. For example, we find DA tends to be more selective in channel selection in higher layers than it does in bottom layers, reflecting the fact that only a few channels have the useful semantic information for prediction. By pruning unselected channels, neural networks with DA enjoys smaller parameters than their counterparts with vanilla self-attention. Our experiments also suggest that Multi-Window DA further improves the performance of Single-Window DA with almost the same neural network size and FLOPs.

1.1 Related Work

Integrating attention mechanism into CNN models also achieved great success in person Re-ID and image classification. Existing works in Re-ID [3, 4] enforce the attention mechanism using convolutional operations with small receptive fields on feature maps. There are also works [5, 6] exploring external clues of human semantics (pose or mask) as attention or to use them to guide the learning of attention. The explicit semantics which represent human structures is helpful for determining the attention. However, the external annotation or additional model for pose/mask estimation is usually required. Following the success of self-attention in natural language processing [7] and its adaption to computer vision tasks [1], recent studies [8, 9, 10] in Person Re-ID also adopted self-attention modules and non-local blocks, which aims at enhancing the features of the target position via aggregating information from all positions. Self-attention is also used to enhance CNNs for image classification and recognition [11, 12]. To the best of our knowledge, all attention modules are confined to the regime of fixed channels for computing feature affinity. The proposed DA module focuses attention on the most informative channels of input features.

Our DA module falls in the class of Neural Architecture Search (NAS) methods in the sense that the architecture of attention modules, i.e. the channels used to compute feature affinity, is learned. Existing NAS methods can be grouped into two categories by optimization scheme, namely Differentiable NAS (DNAS) and Non-differentiable NAS. NAS methods heavily rely on controllers based reinforcement learning [13] or evolution algorithms [14] to discover better architecture. The search phase of such methods usually cost thousands of GPU hours. Recently, DNAS have shown promising results with improved efficiency. DNAS frameworks are able to save a huge amount of GPU hours in the search phase. So far all the DNAS methods [15, 16, 15] search for optimal options for architecture in a handcrafted and finite option set. They transform the discrete network architecture space into a continuous space over which differentiable optimization is feasible, and use gradient descent techniques to search the continuous space. However, the continuous space is for the coefficients used to interpolating finite architecture options, not for architecture itself. For example, DARTS [15] relaxes the originally discrete optimization problem of NAS to a continuous problem in terms of the option interpolation coefficients, enabling efficient optimization by Stochastic Gradient Descent (SGD). In a similar manner, almost all the other DNAS methods [17, 18] adapt Softmax or Gumbel Softmax to search among a finite set of candidate operations. For example, to search for the best filter numbers at different convolution layers, FBNet [18, 17], models each option as a term with a Gumbel Softmax mask. In contrast with existing DNAS methods, DA searches for the architecture of attention modules in a continuous architecture space with uncountably many options for the location and key dimension of channels.

The rest of this paper is organized as follows. We first revisit the vanilla self-attention in Section 2.1. Then we introduce our proposed Differentiable Attention (DA) module and its differentiable searching method in Section 2.2. We then introduce the multi-window extension of DA in Section A.1. Lastly, we introduce how we integrate DA into fixed neural network backbones and learnable backbones in Section B.5, with extensive experimental results in Section 3. The right figure of Figure 1 illustrates the overview of a deep neural network with a DA module.

2 Proposed Approach

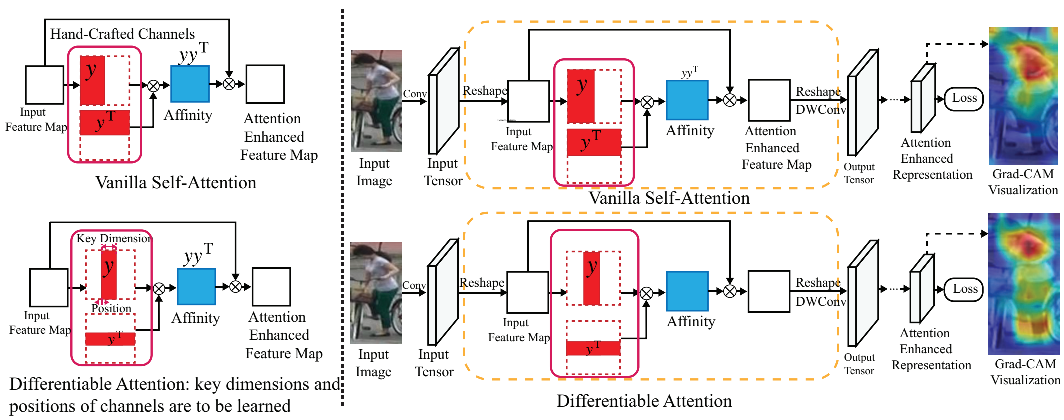


Figure 1: Left: Comparison between the vanilla self-attention and the proposed Differentiable Attention (DA). The channels used in these two types of attention modules are illustrated as boxes in red with notation $\mathbf{y}$. While fixed channels are used in the vanilla self-attention to compute feature affinity, DA automatically searches for informative channels to compute task-oriented feature affinity. Please refer to Section 2.1 and Section 2.2 for more details. Right: Deep neural networks with the vanilla self-attention (top) and DA (bottom) for the Re-ID task. As illustrated by the visualization of the output feature representation with Grad-CAM, DA captures more accurate parts of human body than the vanilla self-attention. Please refer to Section A.6 of the supplementary for more visualization results.

2.1 Revisit Self-Attention and Non-local Block

Vanilla Self-Attention The self-attention module applied in Transformer [7] is in the form of a scaled dot-product. Suppose $\mathbf{X}$ is an input feature of shape $h \times w \times c$ which is reshaped as a matrix $\mathbf{X} \in \mathbb{R}^{hw \times c}$. The vanilla self-attention module applies three projections to $\mathbf{X}$ to obtain key (K), query (Q), and value (V) representations. The output is computed as a weighted sum over the value V by $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$. Following the success of self-attention in natural language processing, non-local block [1] is put forward to integrate self-attention mechanism into CNNs for computer vision tasks. In a non-local block, the attention module can be formulated as $\mathrm{Attention} = f(\theta(\mathbf{X}), \phi(\mathbf{X}))g(\mathbf{X})$, where $\theta(\cdot)$, $\phi(\cdot)$ and $g(\cdot)$ are transformations applied on the input $\mathbf{X}$. Recent work [2] demonstrates that θ, ϕ, and g in this equation can be removed due to CNNs' strong capability of function approximation. By using dot-product to model the correlation between features, the non-local attention module can be simplified as

$$\mathrm{Attention} = \frac{1}{C(\mathbf{X})}\mathbf{X}\mathbf{X}^\top\mathbf{X}, \tag{1}$$

where matrix $\mathbf{X} \in \mathbb{R}^{hw \times c}$ is reshaped from the input tensor $\mathbf{X}$. $C(\mathbf{X})$ is a normalization factor similar to the softmax function in self-attention. As dot-product is used to compute feature affinity, $C(\mathbf{X})$ equals to the number of positions in $\mathbf{X}$. In previous non-local attention module designs [1, 2, 14],

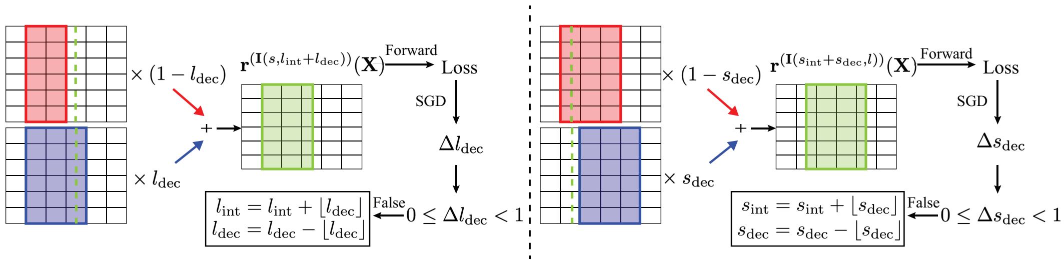


Figure 2: Interpolation process in Differentiable Attention. The left figure illustrates the forward and backward processes for the key dimension l of DA. The right figure illustrates these processes for the position s of DA.

the channels used to compute the affinity between input features are usually selected by handcrafted pooling [1] or sampling [2]. As a result, the feature affinity matrix is expressed as

$$\mathbf{r}^{(\mathbf{I}_0)}(\mathbf{X}) = \mathbf{X}^{(\mathbf{I}_0)} {\mathbf{X}^{(\mathbf{I}_0)}}^\top, \tag{2}$$

where $\mathbf{X}^{(\mathbf{I}_0)}$ is a submatrix of $\mathbf{X}$ formed by aggregating columns of $\mathbf{X}$ with column indices in a set $\mathbf{I}_0$. For example, LightNL [2] compresses $\mathbf{X}$ by setting $\mathbf{I}_0$ to $\{1, 2, \ldots, \lfloor r \times c \rfloor\}$, where r is a fixed ratio, such as 0.5. Such $\mathbf{X}^{(\mathbf{I}_0)}$ is the channels marked in red in the left figure of Figure 1 for the vanilla self-attention, with $\mathbf{y} = \mathbf{X}^{(\mathbf{I}_0)}$.

2.2 Differentiable Attention and Its Differential Searching Method

Clearly, different channels of the input features encode different information. As a result, handcrafted sampling or pooling methods risk overlooking key channels important for feature affinity and using uninformative channels to generate the feature affinity matrix $\mathbf{r}$.

To solve this problem, we propose a novel differentiable searching algorithm to search for the the most important channels to compute the feature affinity. In other words, we propose to search for an optimal $\mathbf{I}$ and the compute the feature affinity matrix by $\mathbf{r}^{(\mathbf{I})}(X) = \mathbf{X}^{(\mathbf{I})} {\mathbf{X}^{(\mathbf{I})}}^\top$. Note that $\mathbf{I}$ is a subset of all the indices of columns of $\mathbf{X}$. For convenience of optimization, we restrict $\mathbf{I}$ to consecutive indices so that it can be parameterized by a staring location s and a size, or key dimension l. In this manner, $\mathbf{I}$ is expressed as $\mathbf{I}_{(s,l)} = \{s, s+1, \ldots, s+l\}$, and our goal is reduced to searching for s and l. The correlation matrix can be expressed as

$$\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) = \mathbf{X}^{(\mathbf{I}_{(s,l)})} {\mathbf{X}^{(\mathbf{I}_{(s,l)})}}^\top, \tag{3}$$

where s and l are integers in the conventional setting of attention modules. Note that $\mathbf{X}^{(\mathbf{I}_{(s,l)})}$ forms a window of channels, so our goal is to search for a window to potentially improve the performance of the vanilla self-attention. In the left figure of Figure 1 for DA, $\mathbf{X}^{(\mathbf{I}_{(s,l)})}$ is illustrated as a red box of learnable location and size/key dimension, where $\mathbf{y} = \mathbf{X}^{(\mathbf{I}_{(s,l)})}$. Existing DNAS methods [15, 18] treat different choices of s and l as different options and adopt Softmax or Gumbel Softmax based searching methods to search among a handcrafted and finite option set. The performance of such methods highly depends on the choice of option set. Moreover, the searching process of representaive DNAS methods, such as FBNetV2 [17], involve a separately tuned temperature parameter which is not trained by SGD used to optimize the network weights. As a result, there could be inconsistency between searching for architecture and training of neural network weights in the searching process. To this end, we propose to search for the starting location s and the key dimension l for DA by a novel differentiable searching method where s and l can take fractional values and the network loss function is differentiable with respect to s and l . The advantages of the differentiable searching method are two-fold. First, it searches for the optimal s and l among uncountably infinite options because s and l are in a continuous domain. Second, s and l are optimized by the same SGD used to optimize the network weights. As a result, the searching for s and l, or equivalently the searching for the architecture of the DA module, is seamlessly incorporated into the optimization of other network weights by the regular SGD, so there is no inconsistency between optimization of architecture parameters and network weights.

To further explore the result of the searching of s and l over the feature map $\mathbf{X}$, we visualized an input feature map of DA with searched s and l in Figure 3 deferred to the supplementary. We have also marked the $\mathbf{X}^{(\mathbf{I}_0)}$ where $\mathbf{I}_0$ is the first half of all the channels of $\mathbf{X}$ suggseted by LightNL. As shown in this figure, the red-boxed area of $\mathbf{X}^{(\mathbf{I}_{(s,l)})}$ is potentially more informative since it exhibits more variation in patterns than other areas selected by $\mathbf{X}^{(\mathbf{I}_0)}$.

To optimize s and l with SGD, we need to first calculate the gradient of the loss function w.r.t. s and l. While the loss function of the neural network is not differentiable with respect to integer s and l under the conventional setting, DA employs bilinear interpolation to define the correlation matrix with fractional s and l. The $\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ with fractional s and l can be expressed as

$$\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) = \begin{bmatrix} 1 - s_{\text{dec}} & s_{\text{dec}} \end{bmatrix} \mathbf{R}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) \begin{bmatrix} 1 - l_{\text{dec}} \\ l_{\text{dec}} \end{bmatrix}, \tag{4}$$

where

$$\mathbf{R}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) = \begin{bmatrix} \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}})})}(\mathbf{X}) & \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}}+1)})}(\mathbf{X}) \\ \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}})})}(\mathbf{X}) & \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}}+1)})}(\mathbf{X}) \end{bmatrix}. \tag{5}$$

In equation (5), $s_{\text{int}} = \lfloor s \rfloor$, $s_{\text{dec}} = s - \lfloor s \rfloor$,and $l_{\text{int}} = \lfloor l \rfloor$, $l_{\text{dec}} = l - \lfloor l \rfloor$ are integral part and decimal part of s and l respectively, where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. With the bilinear interpolation, we are now able to compute the gradients of the loss function w.r.t. s_{dec} and l_{dec} by

$$\nabla_{s_{\text{dec}}} L = \nabla_{\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})} L \cdot \nabla_{s_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}), \tag{6}$$

$$\nabla_{l_{\text{dec}}} L = \nabla_{\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})} L \cdot \nabla_{l_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}). \tag{7}$$

$\nabla_{s_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ and $\nabla_{l_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ are the gradients of correlation matrix w.r.t. s_{dec} and l_{dec}. With equation (4), the gradients of correlation matrix w.r.t. s_{dec} and l_{dec} are computed by

$$\begin{aligned} \nabla_{s_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) = &(1 - l_{\text{dec}})(\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}})})}(\mathbf{X}) - \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}})})}(\mathbf{X})) \\ &+ l_{\text{dec}}(\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}}+1)})}(\mathbf{X}) - \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}}+1)})}(\mathbf{X})), \end{aligned} \tag{8}$$

and

$$\begin{aligned} \nabla_{l_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X}) = &(1 - l_{\text{dec}})(\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}}+1)})}(\mathbf{X}) - \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}})})}(\mathbf{X})) \\ &+ l_{\text{dec}}(\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}}+1)})}(\mathbf{X}) - \mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}})})}(\mathbf{X})). \end{aligned} \tag{9}$$

Note that in equation (5), $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}}+1)})}(\mathbf{X})$ and $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}}+1)})}(\mathbf{X})$ contain one more element than $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}})})}(\mathbf{X})$ and $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}})})}(\mathbf{X})$. To make their size compatible in the bilinear interpolation, we pad one zero after $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}},l_{\text{int}})})}(\mathbf{X})$ and $\mathbf{r}^{(\mathbf{I}_{(s_{\text{int}}+1,l_{\text{int}})})}(\mathbf{X})$.

In the above formulation, s_{int} and l_{int} are indices for slicing $\mathbf{X}$ to compute spatial correlation. Our key observation is that, while the network loss function is not differentiable with respect to s_{int} and l_{int}, it is indeed differentiable with respect to s_{dec} and l_{dec} based on our calculation. Therefore, we can apply regular SGD to optimize s_{dec} and l_{dec}, and update s_{int} and l_{int} whenever the decimal values of s_{dec} and l_{dec} are out of the range of $(0, 1)$. Training a neural network with DA using the proposed differentiable searching algorithm is described in Algorithm 1. Moreover, Figure 2 shows how $\mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ is computed by bilienar interpolation in the forward process, and how l_{int}, l_{dec}, s_{int}, s_{dec} are updated in the backward process.

So far we have introduced our new differentiable searching algorithm for DA which searches for a single window of channels, or Single-Window DA. In order to avoid the potential risk of loosing informative channels by using only one window and enhance the flexibility of the window searching process, we extend the Single-Window DA to Multi-Window DA where multiple windows are searched for, which is detailed in Section A.1 of the supplementary.

2.3 DA with Fixed Backbone and Learnable Backbone

With our novel differentiable searching method for DA in in Algorithm 1 deferred to the supplementary, the searching for the architecture of DA can be performed by regular SGD. As a result, DA can be incorporated into arbitrary neural network backbone, and the weights of the backbone and the architecture of DA modules can be jointly trained by SGD. To evaluate the performance of DA for person Re-ID, we designed two models where DA is incorporated into popular feature extraction backbones, such as MobileNetV2, and the learnable backbone by FBNetV2 [17], leading to DA-FB and DA-DNAS respectively. Both Single-Window DA and Multi-Window DA can be used for DA-FB or DA-DNAS.

3 Experiments

In this section, we demonstrate the performance of DA for the Re-ID task. Table 1 shows the performance of DA-FB with single-window DA and competing baselines on the three person Re-ID datasets. In particular, Table 1 demonstrates that DA-DNAS archives superior results on all the standard Re-ID benchmark datasets over other competing Re-ID methods with a compact neural architecture. With only 1.809 GFLOPs, DA-DNAS achieves much better performance than existing methods which require higher GFLOPs, such as Auto-ReID [10] and ABD-Net [9].

More results and details are deferred to the supplementary. In the supplementary, Section B.1 introduces the datasets and evaluation metrics, and Section B.2- Section B.3 detail our results for single-window and multi-window DA. Section B.4 further shows the performance of multi-window DA on the image classification task on the ILSVRC-12 dataset [19].

Table 1: Performance of DCS-FB with comparisons to state-of-the-art Re-ID models

Methods	Backbones	Input Size	Params(M)	FLOPs(G)	Market1501		DukeMTMC-reID		MSMT17	
					mAP	R1	mAP	R1	mAP	R1
Trained from scratch										
HACNN [20]	Inception	160×64	4.5	0.55	79.9	92.3	63.8	80.5	-	-
OSNet [21]	OSNet	256×128	2.2	0.98	81.0	93.6	68.6	84.7	43.3	71.0
Auto-ReID [9]	ResNet50	384×128	13.1	2.05	74.6	90.7	-	-	-	-
RGA [8]	MobileNetV2	256×128	5.13	2.63	81.5	92.9	-	-	-	-
Baseline (ours)	MobileNetV2	256×128	2.22	0.380	78.9	92.0	-	-	-	-
DCS-FB (ours)	MobileNetV2	256×128	2.23	0.382	84.5	93.9	73.6	85.5	36.9	63.6
DCS-FB (ours)	MobileNetV2 - 200	256×128	5.09	0.884	87.3	95.1	77.2	88.6	45.9	72.3
DCS-FB (ours)	OSNet	256×128	2.2	0.98	84.0	93.8	73.2	85.2	36.7	63.4 -
Pre-trained on ImageNet										
AANet [22]	ResNet50	256×128	>23.5	-	85.3	94.7	75.3	84.0	-	-
CAMA [23]	ResNet50	256×128	>23.5	-	84.5	94.7	72.9	85.8	-	-
BAT-Net [24]	ResNet50	256×128	>23.5	-	87.4	95.1	77.3	87.7	-	-
ABD-Net [9]	ResNet50	384×128	69.17	14.1	88.28	95.6	78.59	89.0	60.8	82.3
Auto-ReID [10]	ResNet50	384×128	13.1	2.05	85.1	94.5	-	-	52.5	78.2
OSNet [21]	OSNet	256×128	2.2	0.98	84.9	94.8	73.5	88.6	52.9	78.7
RGA [8]	ResNet50	256×128	28.3	-	87.5	96.0	-	-	57.5	80.3
DCS-FB (ours)	MobileNetV2	256×128	2.23	0.382	85.0	94.7	75.2	86.7	52.7	78.2
DCS-FB (ours)	MobileNetV2 - 200	256×128	5.09	0.884	87.1	95.1	78.6	89.1	54.8	78.5
DCS-FB (ours)	OSNet	256×128	2.2	0.98	86.4	94.6	75.4	88.7	54.6	79.3
DCS-FB (ours)	MobileNetV2	384×128	2.23	0.571	86.3	95.0	76.4	88.2	56.2	79.9
DCS-FB (ours)	MobileNetV2 - 200	384×128	5.09	1.32	88.3	95.3	79.3	90.1	57.8	80.5
DCS-FB (ours)	ResNet50	384×128	23.5	6.55	**88.4**	**96.0**	76.3	88.4	56.2	80.3
DCS-DNAS(ours)	-	384×128	24.5	**1.809**	**88.3**	**95.7**	**79.8**	**91.1**	**62.9**	**83.6**
DCS-DNAS(ours)	-	384×128	13.2	1.235	88.2	95.6	79.5	90.6	62.1	82.8

4 Conclusion

We presented Differentiable Attention (DA), which searches for the informative channels when computing the feature affinity matrix in attention modules. In contrast with conventional self-attention modules, DA searches for the location and key dimension of channels in a continuous space comprising uncountably infinite options. We also extend DA to a Multi-Window design, which further improves the performance of DA. DA with fixed or learnable neural backbones outperforms other competing methods for two computer vision tasks, person Re-ID on three public benchmark datasets and image classification on the ILSVRC-12 dataset, in terms of prediction accuracy and size/FLOPs of the resultant models.

References

[1] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages

7794–7803, 2018.

[2] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10297–10306, 2020.

[3] Kai Li, Zhengming Ding, Kunpeng Li, Yulun Zhang, and Yun Fu. Support neighbor loss for person re-identification. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1492–1500, 2018.

[4] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 152–159, 2014.

[5] Haiyu Zhao, Maoqing Tian, Shuyang Sun, Jing Shao, Junjie Yan, Shuai Yi, Xiaogang Wang, and Xiaoou Tang. Spindle net: Person re-identification with human body region guided feature decomposition and fusion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1077–1085, 2017.

[6] Liang Zheng, Yujia Huang, Huchuan Lu, and Yi Yang. Pose-invariant embedding for deep person re-identification. *IEEE Transactions on Image Processing*, 28(9):4500–4509, 2019.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[8] Zhizheng Zhang, Cuiling Lan, Wenjun Zeng, Xin Jin, and Zhibo Chen. Relation-aware global attention for person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3186–3195, 2020.

[9] Tianlong Chen, Shaojin Ding, Jingyi Xie, Ye Yuan, Wuyang Chen, Yang Yang, Zhou Ren, and Zhangyang Wang. Abd-net: Attentive but diverse person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8351–8361, 2019.

[10] Ruijie Quan, Xuanyi Dong, Yu Wu, Linchao Zhu, and Yi Yang. Auto-reid: Searching for a part-aware convnet for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3750–3759, 2019.

[11] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2018.

[12] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10073–10082. IEEE, 2020.

[13] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[14] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[16] Richard Shin, Charles Packer, and Dawn Song. Differentiable neural network architecture search. 2018.

[17] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020.

[18] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[20] Wei Li, Xiatian Zhu, and Shaogang Gong. Harmonious attention network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2285–2294, 2018.

[21] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3702–3712, 2019.

[22] Chiat-Pin Tay, Sharmili Roy, and Kim-Hui Yap. Aanet: Attribute attention network for person re-identifications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7134–7143, 2019.

[23] Wenjie Yang, Houjing Huang, Zhang Zhang, Xiaotang Chen, Kaiqi Huang, and Shu Zhang. Towards rich feature discovery with class activation maps augmentation for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1389–1398, 2019.

[24] Pengfei Fang, Jieming Zhou, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Bilinear attention networks for person retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8030–8039, 2019.

[25] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[26] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[27] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 79–88, 2018.

[28] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1318–1327, 2017.

[29] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[33] 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.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[36] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

Supplementary for Differentiable Attention

A Additional details in Section 2

In this section, we first show the visualization of the input feature map of DA with searched s and l in Figure 3, and the algorithm for training a deep neural network with Differentiable Attention in Algorithm 1.

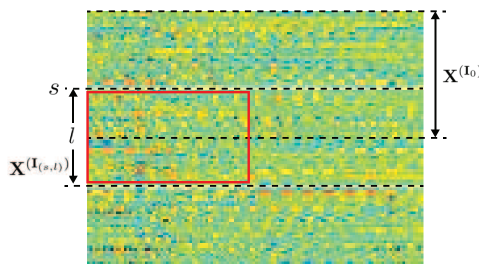


Figure 3: The visualization of $\mathbf{X}^{(\mathbf{I}_{(s,l)})}$ where the feature map $\mathbf{X}$ is the input to the third DA module in MobileNetV2-200 on an image from the DukeMTMC-reID dataset for the Re-ID task. Here $s = 29.049$ and $l = 37.013$, and the shape of $\mathbf{X}$ is 96×128.

Algorithm 1 Training a Deep Neural Network with Differentiable Attention

Input: Maximum iterations T, mini-batch of training samples $\{X_1, X_2, ..., X_N\}$, network learning rate η, DA learning rate γ, and the loss function of the network $L(X, \Omega, s_{\text{dec}}, l_{\text{dec}})$, the percentage p used for the search of DA in each mini-batch.
Output: The network parameters Ω, DA location parameters l_{dec}, l_{int}, and DA key dimension parameters s_{dec}, s_{int}

for $t = 1, 2, ..., T$ **do**
  for $i = 1, 2, ..., \lfloor N \times (1-p) \rfloor$ **do**
    Compute the loss $L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    Obtain the gradient of Ω denoted by $\nabla_{\Omega} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    $\Omega \leftarrow \Omega - \eta \nabla_{\Omega} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
  end for
  for $i = \lfloor N \times (1-p) \rfloor + 1, ..., N$ **do**
    Compute the loss $L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    Compute the gradient of correlation matrix w.r.t. s_{dec} denoted by $\nabla_{s_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ with equation (8)
    Compute the gradient of s_{dec} denoted by $\nabla_{s_{\text{dec}}} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$ with equation (6)
    Update $s_{\text{dec}} \leftarrow s_{\text{dec}} - \gamma \nabla_{s_{\text{dec}}} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    if $0 \leq s_{\text{dec}} < 1$ **then**
      continue
    else
      $s_{\text{dec}} \leftarrow s_{\text{dec}} - \lfloor s_{\text{dec}} \rfloor$
      $s_{\text{int}} \leftarrow s_{\text{int}} + \lfloor s_{\text{dec}} \rfloor$
    end if
    Compute the loss $L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    Compute the gradient of correlation matrix w.r.t. l_{dec} denoted by $\nabla_{l_{\text{dec}}} \mathbf{r}^{(\mathbf{I}_{(s,l)})}(\mathbf{X})$ with equation (9)
    Compute the gradient of l_{dec} denoted by $\nabla_{l_{\text{dec}}} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$ with equation (7)
    Update $l_{\text{dec}} \leftarrow l_{\text{dec}} - \gamma \nabla_{l_{\text{dec}}} L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
    if $0 \leq l_{\text{dec}} < 1$ **then**
      continue
    else
      $l_{\text{dec}} \leftarrow l_{\text{dec}} - \lfloor l_{\text{dec}} \rfloor$
      $l_{\text{int}} \leftarrow l_{\text{int}} + \lfloor l_{\text{dec}} \rfloor$
    end if
  end for
end for

A.1 Multi-Window Differentiable Attention

Section 2 introduces our new differentiable searching algorithm for DA which searches for a single window of channels, or Single-Window DA. In order to avoid the potential risk of loosing informative channels by using only one window and enhance the flexibility of the window searching process, we extend the Single-Window DA to Multi-Window DA where multiple windows are searched for. We herein define the features with channels selected by each window as $\mathbf{X}^{(i)} = \mathbf{X}^{(\mathbf{I}_{(s_i,l_i)})}$. We use (s_i, l_i) to denote the starting location and the size or key dimension of each window, where $i \in \{1, 2, ..., M\}$ and M is the number of windows. Because channels selected by Multi-Window DA are the union of channels selected by all the windows, we need to combine overlapping windows together. Overlapping windows are defined as follows. Let O be a sequence of windows, if the union of all the windows in O is a window with starting position s_O and a key dimension l_O, then O is a sequence of overlapping windows. Let $\{(s_{k_1}, l_{k_1}), (s_{k_2}, l_{k_2}), ..., (s_{k_P}, l_{k_P})\}$ be the starting positions and key dimensions of the windows in such O, then s_O and l_O are computed by

$$s_O = \min_{j=1,\dots,P} s_{k_j}, \quad l_O = \max_{j=1,\dots,P} (s_{k_j} + l_{k_j}) - s_O. \tag{10}$$

In this way, all the M windows of Multi-Window DA can be merged into T disjoint windows with $T \le M$. Suppose that $\{\mathbf{X}^{(O_1)}, \mathbf{X}^{(O_2)}, ..., \mathbf{X}^{(O_T)}\}$ are the final disjoint windows after merging, then the features with channels selected by the union of all the M windows in Multi-Window DA are

$$\mathbf{X}^{\{M\}} = \text{Concat}\left[\mathbf{X}^{(O_1)}, \mathbf{X}^{(O_2)}, ..., \mathbf{X}^{(O_T)}\right]. \tag{11}$$

Then, the correlation matrix in Multi-Window DA can be computed by

$$\mathbf{r}^{\{M\}}(\mathbf{X}) = \mathbf{X}^{\{M\}} {\mathbf{X}^{\{M\}}}^{\top}. \tag{12}$$

Similar to Single-Window DA, the loss function of a neural network equipped with Multi-Window DA is still differentiable with respect to all the starting locations and key dimensions of the M windows, which are $\{s_i, l_i\}_{i=1}^{M}$. Algorithm 1 can be used to train a neural network with Multi-Window DA, and the integral and decimal parts of each s_i and l_i are updated according to the same updating rules in Algorithm 1 for s_{int}, s_{dec} and l_{int}, l_{dec}. By the aforementioned window merging process, the number of final windows T is adaptively inferred. In Section A.5 of the supplementary, we provide statistics on how T varies with comparison to M in different settings, and it is showed that Multi-Window DA can always learn a very compact set of disjoint final windows which contain potentially richer information than than that of Single-Window DA.

B Additional Experimental Results

B.1 Datasets and Evaluation Metrics

In this section, we evaluate our proposed DA modules on three public person Re-ID datasets, i.e., Market-1501 [25], DukeMTMC-reID [26], MSMT17 [27]. Standard Re-ID metrics top-1 accuracy (R1), and the mean Average Precision (mAP) are used to evaluate the performance of DA and baseline models. Note that for the fairness of comparison, re-ranking [28] and multi-query fusion [29] were not used.

B.2 Single-Window DA

B.2.1 DA with Fixed Backbones and Learnable Backbones

DA modules are compatible with popular manually designed feature extraction CNN backbones, such as InceptionNet [30], ResNet [31], and MobileNetV2 [32]. In our experiments, we evaluate the performance of DA modules on widely used lightweight CNN backbone, MobileNetV2, with width 1.0. Similar to ResNet, MobileNetV2 is also built upon bottleneck structures. Following the common practices in person Re-ID [8, 9], the attention modules are added after each convolution stage. Table 1 shows the performance of DA-FB with Single-Window DA and competing baselines on the three person Re-ID datasets. The fixed backbone can be pre-trained on the ImageNet dataset

Table 2: Performance of DA-DNAS on MSMT17

Methods	Params(M)	FLOPs(G)	mAP	R1
Auto-ReID [10]	13.1	2.05	52.5	78.2
RGA [8]	28.3	-	57.5	80.3
ABD-Net [9]	69.17	14.1	60.8	82.3
DA-DNAS	13.2	1.235	62.1	82.8
DA-DNAS	24.5	**1.809**	**62.9**	**83.6**

Table 3: Performance of Multi-Window DA

Methods	Backbones	Market1501		DukeMTMC-reID		MSMT17	
		mAP	R1	mAP	R1	mAP	R1
Single-Window DA							
DA-FB	MobileNetV2	86.3	95.0	76.4	88.2	56.2	79.9
DA-FB	MobileNetV2 - 200	88.3	95.3	79.3	90.1	57.8	80.5
DA-DNAS	FBNetV2	**88.3**	**95.7**	**79.8**	**91.1**	**62.9**	**83.6**
Multi-Window DA							
DA-FB	MobileNetV2	86.2	95.0	76.4	88.2	56.0	79.8
DA-FB	MobileNetV2 - 200	88.2	95.4	79.2	90.3	57.8	80.8
DA-DNAS	FBNetV2	**88.6**	**96.0**	**80.1**	**91.5**	**63.1**	**83.9**

[19] or not. It can be observed that DA-FB outperforms the corresponding baseline network with the same manually designed backbone, reflecting the advantage of searching for the location and key dimension of attention models automatically in a continuous space. Notably, with pre-training on ImageNet, DA-FB with the backbone of MobileNetV2-200 leads to a model of only 5.09M parameters and 1.32G FLOPs achieving mAP and R1 of 79.2% and 90.1% on DukeMTMC-reID. To integrate learnable backbone with DA, we adopt the supergraph proposed in FBNetV2 [17] and jointly train the network backbone and DA modules following Section B.5. The learned backbone is a subgraph of the supergraph learned by the DNAS algorithm. The inverted residual bottleneck with 3×3 convolution kernel is used as the basic building block of the supergraph. DNAS algorithm is used to search the channel number of each convolution layer. The experiment results of DA with learnable backbone on MSMT17 is shown in Tabel 2. On MSMT17, DA-DNAS achieve the best mAP and R1 with only 12.8% of the FLOPs required by the second best model in accuracy, ABD-Net [9]. For each neural backbone, a DA module is inserted after each of its four stages. Section B.5 of the supplementary includes the detailed architecture of MobileNetV2,MobileNetV2-200 and the supergraph of FBNetV2 used in the experiments. For DA-FB and DA-DNAS, the architecture of DA, which includes location and key dimension of channels, and the architecture of the neural backbone (if applicable) are learned during the search phase, and the learned architecture is then trained again to obtain the final performance. The training details are included in Section B.6 of the supplementary.

B.3 Multi-Window DA

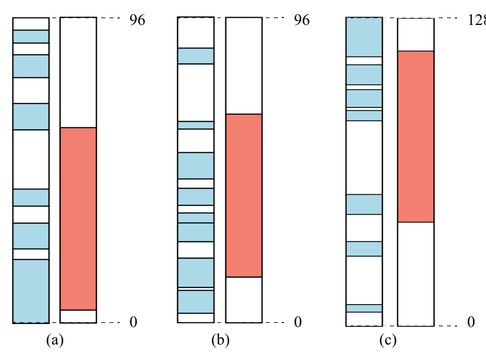


Figure 4: Comparisons between windows learned by Multi-Window DA and Single-Window DA inserted after the third stage in **(a)** MobileNetV2, **(b)** MobileNetV2-200 and **(c)** FBNetV2 on the Market-1501 dataset

In this section, we demonstrate the performance of Multi-Window DA with comparison to its single-window counterpart, that is, Single-Window DA. To this end, we replace every Single-Window DA in the MobileNetV2, MobileNetV2-200 and FBNetV2 in the previous section with a Multi-Window DA. The length of each window in the Multi-Window DA is initialized to l_0. The starting positions of different windows are initialized as $\{0, l_0 \times 1, l_0 \times 2, ..., l_0(M-1)\}$, where c is the number of channels of the input feature and the initial window number is $M = \lceil \frac{c}{l_0} \rceil$. Note that the size of the last window is set to $c - l_0(M-1)$ which may not be l_0. l_0 is empirically set to 8 in our experiments. In this manner, the windows are initialized to cover all the channels of the input features before searching. It is worthwhile to emphasize that 8 is an empirical choice of the initial window size which should be a small number, so that Multi-Window DA can start with many small windows. By the window merging process introduced in Section A.1, windows will be merged and the final window number can be automatically inferred. The optimization settings for the searching phase and training phase are the same as that we used for Single-Window DA in Section B.2. The experiment results of Multi-Window DA with comparison to Single-Window DA on three public benchmarks are shown in Table 3. The results manifest that Multi-Window DA outperforms Single-Window DA on the public benchmarks. The best accuracy on MSMT17, (mAP,R1) of (63.1%,83.9%), is achieved with even smaller FLOPs of 1.798M after removing channels not selected by Multi-Window DA, compared to 1.809M in Table 2. We also visualize the windows learned by Multi-Window DA and Single-Window DA inserted after the third stage of MobileNetV2 and MobileNetV2-200 on the Market-1501 dataset in Figure 4, where the windows are marked in colors. Figure 4 illustrates that the windows learned by Multi-Window DA sparsely spread over all the channels and select channels not chosen by its Single-Window counterpart. The distribution of windows also suggests that Multi-window DA has more flexibility in searching for the most informative channels for computing pairwise affinity of the input features. In Section B.7, we further study the windows learned by these two variants of DA, and show 1) the statistics including the sum of the length of windows compared to the channel number of input features; 2) the histogram of window length for Multi-Window DA. We observe that Multi-Window DA can adaptively select more compact set of channels in higher layers. Additionally, the sum of the window length of Multi-Window DA is usually smaller than the window length of Single-Window DA. Notably, In Table 8 and Table 9 of Section B.7 showing the window statistics, we observe that while DA selects most channels after Stage 1 of MobileNetV2 and FBNetV2, it only selects less than 48.6% channels after Stage 4 for all neural backbones. This suggests that Multi-Window DA is capable of searching for the inherent sparsity structure of the channels that contribute to capturing task-oriented feature correlation. Such observation is of independent interest, since it provides a principled way of identifying redundant channels in models with attention modules.

B.4 Multi-Widow DA for Image Classification on ImageNet

Table 4: Performance of Multi-Window DA for Image Classification on ImageNet. SA stands for the vanilla self-attention, and MW-DA stands for Multi-Window DA

Methods	FLOPs(M)/#Params(M)	Top-1	Top-5
MobileNetV2	327.7/3.51	71.8	90.5
MobileNetV2+SA	330.2/3.51	72.1	91.1
MobileNetV2+MW-DA	**329.1/3.51**	**72.4**	**91.2**
FBNetV2	330.3/7.50	75.7	92.5
FBNetV2+SA	333.2/7.50	75.9	92.9
FBNetV2+MW-DA	**331.7/7.50**	**76.1**	**93.0**

We also evaluate DA for the task of large-scale image classification. Table 4 shows the performance of Multi-Window DA with two neural backbones, MobileNetV2 and FBNetV2, on the ILSVRC-12 dataset [19]. The architecture of the MobileNetV2 and the supergraph of FBNetV2 and the location where the vanilla Self-Attention (SA)/Multi-Window DA (MW-DA) modules are inserted into the corresponding neural backbones are the same as that used for the Re-ID task in Section B.3, expect for slight adjustment in the last layer for classification purpose. It can be observed that while SA improves the accuracy of the corresponding baseline, MW-DA further improves the Top-1 accuracy of SA by the same amount as the improvement of SA over the baseline (0.3% for MobileNetV2, 0.2% for FBNetV2). Notably, such accuracy improvement of MW-DA over SA is promising, as the resultant model even enjoys slightly less FLOPs by removing the redundant channels not selected by MW-DA.

B.5 DA with Fixed Backbones and Learnable Backbones

We introduce the details about the architecture of MobileNetV2, MobileNetV2-200 and the supergraph of FBNetV2 used in the experiments of the main paper.

MobileNetV2 [33, 32] is an efficient neural network model with depthwise separable convolution and inverted residual block as building blocks. The original MobileNetV2 has 53 convolution layers. Following the convention of incorporating non-local attention blocks into neural networks [1, 8], we insert DA modules to the end of each convolution stage of MobileNetV2. To further explore the potential of DA with deeper backbone, we designed a deeper variant of MobileNetV2 with 200 layers termed MobileNetV2-200, which is inspired by the design of 200-layer ResNet [34]. Table 5 shows the structure of MobileNetV2 and MobileNetV2-200 as well as the positions of DA modules.

We also combined DA modules and the supergraph of FBNetV2 [17] to propose DA-DNAS where the neural network backbone and the DA modules are jointly trained. FBNetV2 employs Differentiable Neural Architecture Search (DNAS) algorithm to learn the backbone architecture by choosing options in a supergraph. FBNetV2 designs a search space with building blocks inspired by the design of MobileNetV2. It features a masking mechanism based on Gumbel Softmax for feature map reuse so that it can efficiently search for the number of filters of each convolution layer. We insert DA modules into the supergraph of FBNetV2, and the supergraph of FBNetV2 and the positions of DA modules are shown in Table 7.

Operator	t	c	n		s
			MobileNetV2	MobileNetV2-200	
conv2d	-	32	1	1	2
bottleneck	1	16	1	1	1
bottleneck	6	24	2	3	2
DA Module Inserted					
bottleneck	6	32	3	21	2
DA Module Inserted					
bottleneck	6	64	4	19	2
bottleneck	6	96	3	18	1
DA Module Inserted					
bottleneck	6	160	3	3	2
bottleneck	6	320	1	1	1
DA Module Inserted					
conv2d	-	1280	1	1	1
avgpool	-	-	1	1	-
conv2d	-	-	-	-	

Table 5: The backbone structure of MobileNetV2 and MobileNetV2-200 Each line describes a sequence of 1 or more identical layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. The expansion factor t is always applied to the input feature.

Operator	e	f	n		s
			FBNetV2	FBNetV2-Large	
conv2d	1	16	1	1	2
bottleneck	1	(12, 16, 4)	1	1	1
bottleneck	(0.75, 3.25, 0.5)	(16, 28, 4)	1	1	2
bottleneck	(0.75, 3.25, 0.5)	(16, 28, 4)	2	6	1
DA Module Inserted					
bottleneck	(0.75, 3.25, 0.5)	(16, 40, 8)	1	3	2
bottleneck	(0.75, 3.25, 0.5)	(16, 40, 8)	2	6	1
DA Module Inserted					
bottleneck	(0.75, 3.25, 0.5)	(48, 96, 8)	1	3	2
bottleneck	(0.75, 3.25, 0.5)	(48, 96, 8)	2	6	1
bottleneck	(0.75, 4.5, 0.75)	(72, 128, 8)	4	12	1
DA Module Inserted					
bottleneck	(0.75, 4.5, 0.75)	(112, 216, 8)	1	3	2
bottleneck	(0.75, 4.5, 0.75)	(112, 216, 8)	3	3	1
DA Module Inserted					
conv2d	-	1984	1	1	1
avgpool	-	-	1	1	1
fc	-	-	1	-	-

Table 6: The supergraph of FBNetV2 and FBNetV2-Large (3 $\times$ depth), with block expansion rate e, number of filters f, number of blocks n, and stride of first block s Tuples of three values in the column of expansion rate e and number of filters f represent the lowest value, highest, and steps between options (low, high, steps).

The backbone architecture of FBNetV2 is learned in a differentiable manner by SGD, therefore, the searching for the backbone architecture and the architecture of our DA modules can be jointly performed by SGD. The proposed DA-DNAS jointly searches for the backbone architecture and the architecture of DA modules. Similar to the design of MobileNetV2-200, we also designed a deeper search space for FBNetV2, termed as FBNetV2-Large. The depth of FBNetV2-Large is 3 times the depth of the original FBNetV2. The structure of FBNetV2-Large is also shown in Table 6. Combining our DA module with DNAS algorithm, we are able to search for models of different size. To test the performance of our DA module with larger backbone, we also designed a deeper version of FBNetV2-Large, which is approximately $6\times$ the depth of the original FBNetV2. Table 7 shows the supergraph of the deeper FBNetV2-Large. The performance of DA with FBNetV2-Large (DA-DNAS ($3\times$)) and the deeper FBNetV2-Large (DA-DNAS ($6\times$)) is shown in Table 2. It can be observed that DA-DNAS ($6\times$) outperforms ABD-Net[9] in terms of top-1 accuracy, while the model size and FLOPs are fractions of that of ABD-Net.

Operator	e	f	n	s
conv2d	1	16	1	2
bottleneck	1	(12, 16, 4)	1	1
bottleneck	(0.75, 6.25, 0.5)	(16, 28, 4)	1	2
bottleneck	(0.75, 6.25, 0.5)	(16, 28, 4)	12	1
DA Module Inserted				
bottleneck	(0.75, 6.25, 0.5)	(16, 40, 8)	6	2
bottleneck	(0.75, 6.25, 0.5)	(16, 40, 8)	12	1
DA Module Inserted				
bottleneck	(0.75, 6.25, 0.5)	(48, 96, 8)	6	2
bottleneck	(0.75, 6.25, 0.5)	(48, 96, 8)	12	1
bottleneck	(0.75, 7.5, 0.75)	(72, 128, 8)	24	1
DA Module Inserted				
bottleneck	(0.75, 7.5, 0.75)	(112, 216, 8)	6	2
bottleneck	(0.75, 7.5, 0.75)	(112, 216, 8)	6	1
DA Module Inserted				
conv2d	-	1984	1	1
avgpool	-	-	1	1
fc	-	-	-	-

Table 7: The supergraph of FBNetV2-Large (6 $\times$ depth), with block expansion rate e, number of filters f, number of blocks n, and stride of first block s Tuples of three values in the column of expansion rate e and number of filters f represent the lowest value, highest, and steps between options (low, high, steps).

B.6 Training Details of DA with Fixed Backbones and Learnable Backbones

B.6.1 DA with Fixed Backbones

During the search phase, we use input size of 256×128 with a batch size of 64. Momentum SGD is used to optimize both the architecture parameters and network parameters for 300 epochs. In each epoch, the network weights are trained with 80% of training samples, and the parameters for positions and key dimensions of all DA modules are trained with the remaining 20% of training samples. The initial learning rate for architecture parameter is set to 0.3, and cosine schedule is applied. The initial learning for network parameters is set to 0.035, and we decay it by 10 at the 150-th and 240-th epoch. The architecture parameters and network parameters are updated iteratively during the search phase.

In the training phase, the size of input images is 256×128 for all datasets. Following common practice, we also use random cropping, horizontal flipping, and random erasing to augment the data. Both identification loss with label smoothing [35] and triplet loss with hard mining [36] are used to supervise the training. All models are trained with momentum SGD for 600 epochs. The momentum for SGD is set as 0.9. The initial learning rate is set to 0.035, and we decay the learning rate by 10 at the 300-th and 500-th epoch. We set the weight decay of SGD to 0.0005.

The experiment results of DA with fxied backbone is shown in Tabel 1 of the main paper, where DA-FB denotes our model. In the experiments, DA is integrated into multiple CNN backbones including OSNet, MobileNetV2, and ResNet50. We compare the performance of our model with other state-of-the-art methods on three datasets. For fair comparisons with some state-of-the-art methods like Auto-ReID, we have also tried input size of 384×128.

B.6.2 DA with Learnable Backbones

During the search phase, we use input size of 384×128 with a batch size of 64. Both the architecture parameters and network parameters are trained for 300 epochs. In each epoch, the network weights are trained with 80% of training samples by SGD. The Gumbel Softmax sampling parameters in the supergraph and the parameters for positions and key dimensions of all DA modules are trained with the remaining 20% using Adam. The initial learning rate for optimizing architecture parameters is set to 0.03, and cosine learning rate schedule is applied. The initial learning rate for network parameters is set to 0.035, and it is decayed by 10 at the 150-th and 240-th epoch.

After the search phase, we pre-train the model on ImageNet. Then we fine-tune the model on the Re-ID datasets. During the fine-tuning process, the input images are augmented by random horizontal flip, normalization, random erasing, and mixup. Adam is used to fine-tune the network. The initial learning rate is set to 0.00035. A warmup strategy is used in the fine-tune process. In the beginning, the backbone weights are frozen and only the weights associated with classifiers are trained. After 10 epochs, all layers are freed for training for the remaining 390 epochs. The learning rate is decayed by 10 after 200 and 300 epochs.

Table 8: Window statistics of Single-Window DA and Multi-Window DA for Re-ID

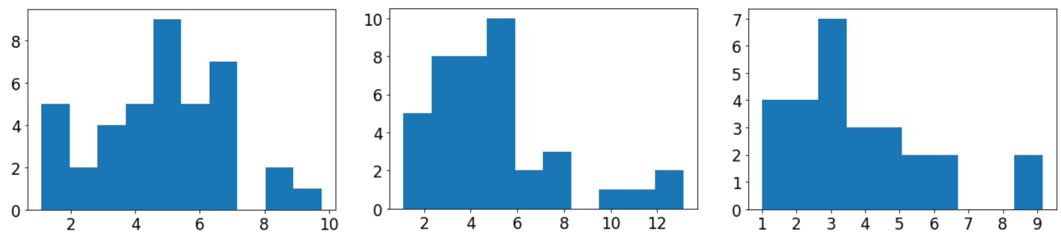
Methods	Backbones	Stage 1		Stage 2		Stage 3		Stage 4	
		sum of window length	channel number	sum of window length	channel number	sum of window length	channel number	sum of window length	channel number
Single-Window DA									
DA-FB	MobileNetV2	23.392	24	30.903	32	57.005	96	126.326	320
DA-FB	MobileNetV2 - 200	23.907	24	31.790	32	51.215	96	137.993	320
DA-DNAS	FBNetV2	27.011	28	38.033	40	50.016	128	120.360	216
Multi-Window DA									
DA-FB	MobileNetV2	21.371 ($3 \to 2$)	24	30.209 ($4 \to 3$)	32	52.955 ($12 \to 6$)	96	133.623 ($40 \to 15$)	320
DA-FB	MobileNetV2 - 200	19.603 ($3 \to 1$)	24	28.317 ($4 \to 3$)	32	46.669 ($12 \to 8$)	96	119.270 ($40 \to 13$)	320
DA-DNAS	FBNetV2	26.367 ($4 \to 3$)	28	37.015 ($5 \to 2$)	40	50.901 ($16 \to 8$)	128	87.216 ($27 \to 9$)	216

Table 9: Window statistics of Multi-Window DA for ImageNet Classification

Methods	Backbones	Stage 1		Stage 2		Stage 3		Stage 4	
		sum of window length	channel number	sum of window length	channel number	sum of window length	channel number	sum of window length	channel number
DA-FB	MobileNetV2	22.293 ($3 \to 1$)	24	31.107 ($4 \to 2$)	32	47.692 ($12 \to 6$)	96	128.919 ($40 \to 13$)	320
DA-DNAS	FBNetV2	26.739 ($4 \to 2$)	28	37.861 ($5 \to 2$)	40	63.031 ($16 \to 8$)	128	105.397 ($27 \to 10$)	216

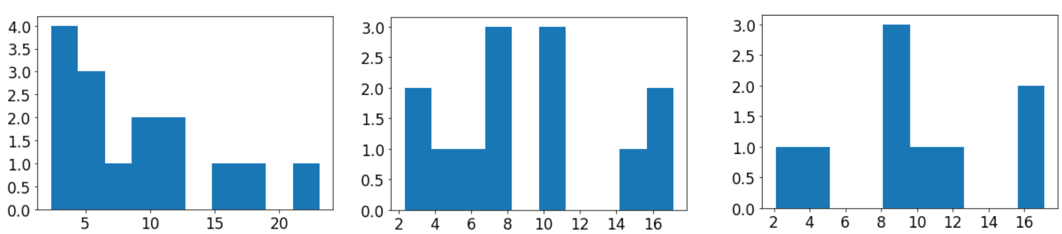
B.7 More Details about Multi-Window DA for Re-ID

The experiment results of Multi-Window DA with comparison to Single-Window DA on three public benchmarks are shown in Table 3 of the main paper. The results manifest that Multi-Window DA outperforms Single-Window DA on the public benchmarks. To further study the windows learned by these two variants of DA, statistics including the sum of the length of windows, and the channel number of input features are shown in Table 8 and Table 9 for Re-ID and image classification respectively. For Multi-Window DA, the number of windows before and after merging is also provided. The sum of window length for Multi-Window DA is the sumber of windows after the window merging process. We observe that Multi-Window DA can adaptively learn a compact of number of windows in DA. Additionally, the sum of the window length of Multi-Window DA is usually smaller than the window length of Single-Window DA. This suggests that Multi-Window DA is more capable of searching for the sparsity structure of the channels that are necessary for capturing task-oriented feature similarity. Figure 5 and Figure 6 illustrate the histograms of window lengths before and after the window merging process in Multi-Window DA inserted after the fourth stage in MobileNetV2, MobileNetV2-200 and FBNetV2 on the Market1501 dataset for the ReID task. We can see that even after the window merging process, most of the final disjoint windows still have a length less than 10. This observation also indicates that Multi-Window DA is able to capture sparse structure of the channels that are important for calculating pairwise affinity of the input features.



(a) (b) (c)

Figure 5: Histograms of window length before merging in Multi-Window DA inserted after the fourth stage for **(a)** MobileNetV2, **(b)** MobileNetV2-200 and **(c)** FBNetV2. In (a) and (b), the histogram is drawn for 40 windows. In (c), the histogram is drawn for 27 windows. In each histogram of this figure and Figure 6, the horizontal axis denotes the length of windows and the vertical axis denotes the number of windows.



(a) (b) (c)

Figure 6: Histograms of window length after the window merging process in Multi-Window DA inserted after the fourth stage for **(a)** MobileNetV2, **(b)** MobileNetV2-200 and **(c)** FBNetV2. The number of windows in (a), (b) and (c) are 15, 13 and 9, respectively. The vertical axis denotes the number of final disjoint windows after the window merging process described in Section 2.3 of the main paper.