DIFFERENTIABLE CHANNEL SELECTION FOR SELF-ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-attention has been widely used in convolutional neural networks for computer vision tasks. 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 Channel Selection (DCS). In contrast with the conventional self-attention, DCS searches for input-dependent locations and key dimensions of channels in a continuous space by a novel differentiable searching method. Our DCS module is compatible with either fixed neural network backbone or learnable backbone with Differentiable Neural Architecture Search (DNAS), leading to DCS with Fixed Backbone (DCS-FB) and DCS-DNAS respectively. We apply DCS-FB and DCS-DNAS to three computer vision tasks, person Re-IDentification methods (Re-ID), object detection, and image classification, with competitive results on standard benchmarks and compact architecture compared to competing methods, revealing the advantage of DCS.

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 (Wang et al., 2018) 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 (Wang et al., 2018; Li et al., 2020) 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 Channel Selection (DCS) module which searches for the most informative channels in a differentiable manner. Figure 2 illustrates the difference between the vanilla self-attention and the proposed DCS module. The main contributions of this paper are as follows.

First, we propose the Differentiable Channel Selection (DCS) module. In contrast with conventional self-attention where fixed channels are used to compute pairwise similarity between input features, DCS selects the most informative channels to compute task-oriented feature affifinity, which outperforms the vanilla self-attention modules by extensive empirical study. DCS employs a novel differentiable searching algorithm which learns the position and key dimension of the most informative channels in the input features. In contrast with Gumbel-softmax based searching limited to a fixed number of options, DCS searches for the locations and key dimensions of the channels for feaure affinity computation in a continuous space. While locations and key dimensions 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 a carefully designed bilinear interpolation which enables differentiable optimization. The selected channels lie in multiple windows, and the locations and key dimensions are the locations and size of these windows.

Second, DCS modules are incorporated into either Fixed neural network Backbone or learnable backbone with Differentiable Neural Architecture Search (DNAS) algorithm, leading to DCS-FB and DCS-DNAS respectively. DCS-DNAS is a new neural architecture search method jointly learning the network backbone and the architecture of DCS, that is, the location and key dimension of channels. We apply DCS to two computer vision tasks, person Re-ID and image classification with extensive empirical study. DCS-FB and DCS-DNAS not only outperform current state-of-the-art, but also render much more compact architecture compared to competing methods. Notably, DCS achieves the mean Average Precision and top-1 accuracy of 61.2% 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 DCS 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 DCS enjoys smaller parameters than their counterparts with vanilla self-attention.

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 (Li et al., 2018a; 2014b) enforce the attention mechanism using convolutional operations with small receptive fields on feature maps. There are also works (Zhao et al., 2017; Zheng et al., 2019) 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 (Vaswani et al., 2017) and its adaption to computer vision tasks (Wang et al., 2018), recent studies (Zhang et al., 2020; Chen et al., 2019; Quan et al., 2019) 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 (Woo et al., 2018; Zhao et al., 2020). To the best of our knowledge, all attention modules are confined to the regime of fixed channels for computing feature affinity. The proposed DCS module focuses attention on the most informative channels of input features.

Our DCS 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 (Zoph & Le, 2016) or evolution algorithms (Real et al., 2019) 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 (Liu et al., 2018; Shin et al., 2018; Liu et al., 2018) 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 (Liu et al., 2018) 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 (Wan et al., 2020; Wu et al., 2019) 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 (Wu et al., 2019; Wan et al., 2020), models each option as a term with a Gumbel Softmax mask. In contrast with existing DNAS methods, DCS searches for the architecture of attention modules in a continuous architecture space with uncountably many options for the location and key dimension of channels.

Figure 1 shows the feature affinity of a particular query to all the other features in two real images. DCS tends to give higher affinity value to more semantically similar features because only the informative channels of the query are selected and used to compute its affinity to other features.

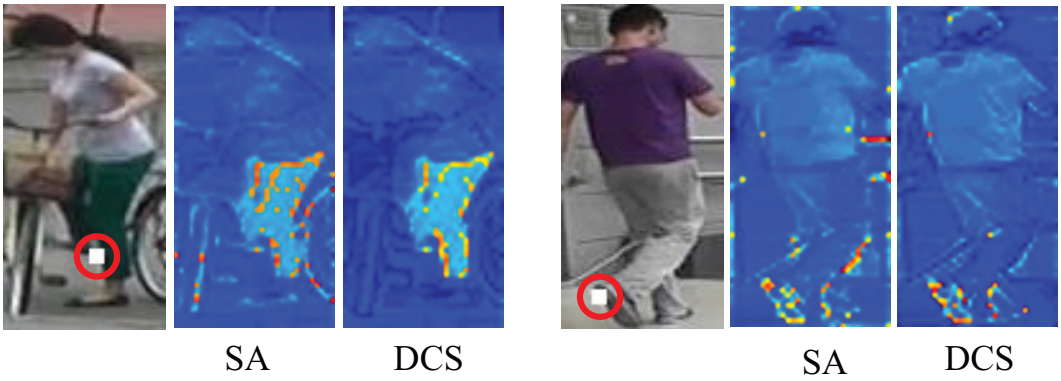

Figure 1: Illustration of feature affinity by the vanilla Self-Attention (SA) and Differentiable Channel Selection (DCS). The query is marked with a white square inside a red circle on each of the two real images. In the heatmap of the feaure affinity for SA and DCS, a redder pixel indicates a higher feature affinity value. It can be observed that DCS renders high feature affinity for features which are more semantically similar to the query, while SA often finds irrelevant features with high affinity to the query.

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 Channel Selection (DCS) module and its differentiable searching method in Section 2.2. We then introduce how we integrate DCS into fixed neural network backbones and learnable backbones, with extensive experimental results in Section 3. The right figure of Figure 2 illustrates the overview of a deep neural network with a DCS module.

2 Proposed Approach

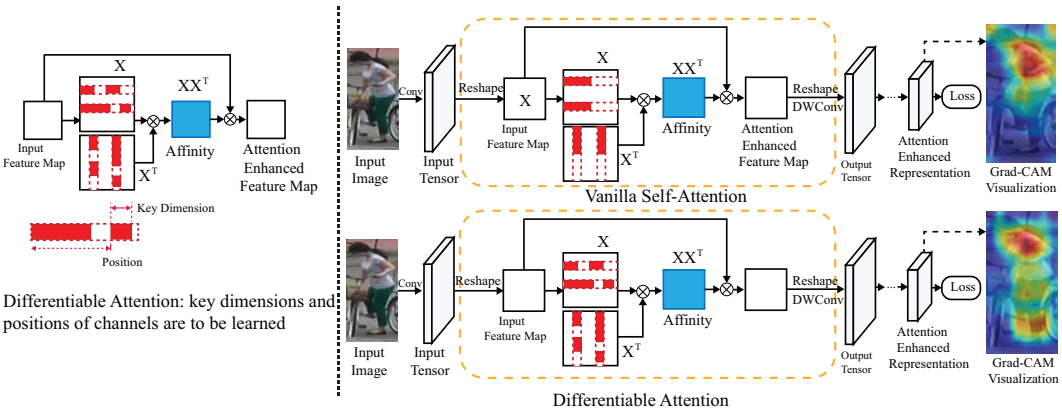

Figure 2: Left: the proposed Differentiable Channel Selection (DCS). Two different features select different channels (red boxes) to compute their affinity to all the other features. DCS automatically searches for informative channels to compute task-oriented feature affinity for each input feature, and please refer to Section 2.1 and Section 2.2 for more details. Right: Deep neural networks with the vanilla self-attention (top) and DCS (bottom) for the Re-ID task. Fixed channels are used in the vanilla self-attention to compute feature affinity for all the input features. As illustrated by the visualization of the output feature representation with Grad-CAM, DCS captures more accurate parts of human body than the vanilla self-attention. Figure 3 illustrate more visualization results.

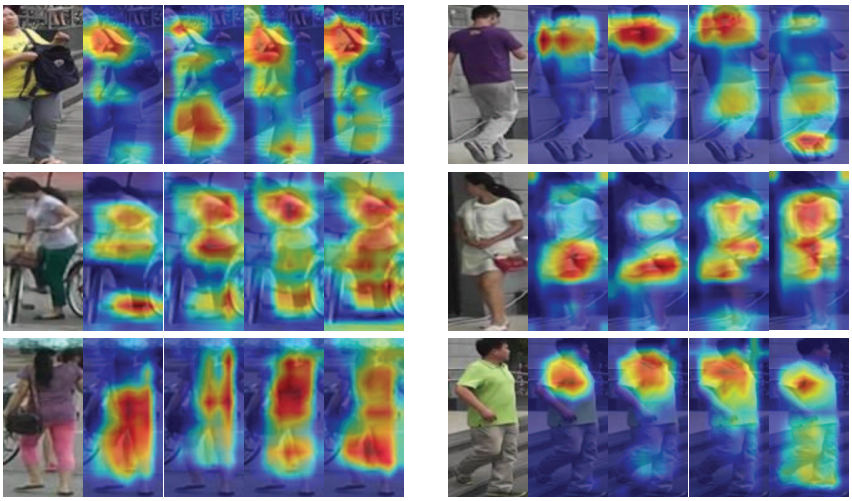

Figure 3: Grad-CAM visualization results For each image, the leftmost sub-figure is the original image, the other sub-figures are the attention visualization heatmaps generated by the Grad-CAM visualization tool. From left to right, the heatmaps are for the original MobileNetV2, MobileNetV2 with baseline self-attention, MobileNetV2 with our Single-Window Differentiable Channel Selection, and MobileNetV2 with our Differentiable Channel Selection.

2.1 Revisit Self-Attention

Vanilla Self-Attention The self-attention module applied in Transformer (Vaswani et al., 2017) is in the form of a scaled dot-product. Suppose $\mathbf{X}$ is an input feature 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 $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$. Following the success of self-attention in natural language processing, the non-local attention module (Wang et al., 2018; Li et al., 2020) produces attention-enhanced feature map by using dot-product to model the correlation between features according to

$$\text{Attention} = \frac{1}{C(\mathbf{X})} \underbrace{\mathbf{X}\mathbf{X}^\top}_{\text{Feature Affinity}} \mathbf{X}, \tag{1}$$

where $C(\mathbf{X})$ is a normalization factor. In previous non-local attention module designs (Wang et al., 2018; Li et al., 2020; Real et al., 2019), the channels used to compute the affinity between input features are usually selected by handcrafted pooling (Wang et al., 2018) or sampling (Li et al., 2020). 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 (Li et al., 2020) 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 right figure of Figure 2 for the vanilla self-attention.

2.2 Differentiable Channel Selection and Its Differential Searching Method

Clearly, different channels of the input features encode different information. For example, an informative input feature, corresponding to an important part of an object, usually incur strong response in particular channels, and using these particular channels for feature affinity computation would let that input feature select more semantically similar keys to render better attention-enhanced feature as the output. In contrast, fixed channels for feature affinity computation risk overlooking key channels important for feature affinity and using uninformative channels to generate the feature affinity matrix.

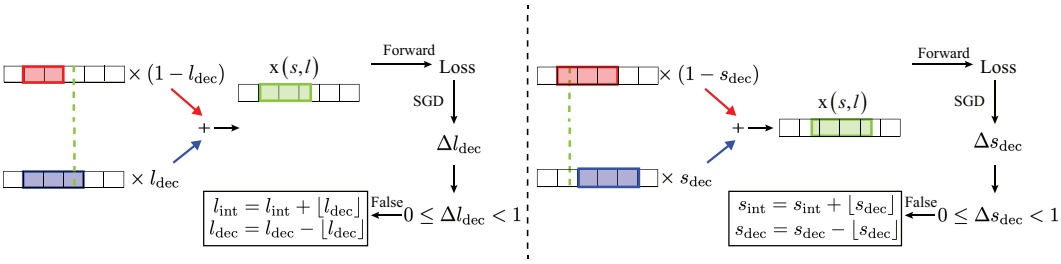

Figure 4: Interpolation process in Differentiable Channel Selection. Suppose an input feature selects channels in a window $[s, l]$, and the channel values with the fractional indices in this window are denoted by $x(s, l)$ The left figure illustrates the forward and backward processes for the key dimension l in DCS. The right figure illustrates these processes for the position s in DCS.

To solve this problem, we put forward a novel differentiable searching algorithm to search for the the most important channels to compute the feature affinity. For each input feature i in the input feature $\mathbf{X}$, the indices of its channels for feature affinity computation, $\mathbf{I}_i \subseteq \{1, 2, \ldots, c\}$ where c is the channel number, are a union of windows. That is, $\mathbf{I}_i = \bigcup_{m=1}^{M} [s_{im}, s_{im} + l_{im}]$ with M being the number of windows, and $\{s_{im}, l_{im}\}_{i=1}^{M}$ are the locations and key dimensions for the input feature i, which are the output of a 1×1 convolution layer with feature i as the input. It is noted that $[s_{im}, s_{im} + l_{im}] = \{s_{im}, s_{im} + 1, \ldots, s_{im} + l_{im}\}$, $\{s_{im}, l_{im}\}_{i=1}^{M}$ are fractional values which are normalized so that $s_{im} \geq 0$ and $s_{im} + l_{im} \leq c$.

We now explain how the channels with fractional indices in $\mathbf{I}_i$ can be computed by a bilinear interpolation process. For the illustration purpose, we assume that the $\mathbf{I}_i$ only has one window which is $[s, l]$. We let $s_{\text{int}} = \lfloor s \rfloor$, $s_{\text{dec}} = s - \lfloor s \rfloor$,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. Figure 4 illustrates how a bilinear interpolation can be used to compute the selected channel values in the fractional indices $[s, l]$ for the forward process in a neural network with one or more DCS modules. Formally, let $x(s, l)$ denotes the selected channel values, then $x(s, l)$ can be computed by the equations (3)-(5) below.

$$x(s_{\text{int}}, l) = (1 - l_{\text{dec}})x(s_{\text{int}}, l_{\text{int}}) + l_{\text{dec}}x(s_{\text{int}}, l_{\text{int}} + 1), \tag{3}$$

$$x(s_{\text{int}} + 1, l) = (1 - l_{\text{dec}})x(s_{\text{int}} + 1, l_{\text{int}}) + l_{\text{dec}}x(s_{\text{int}} + 1, l_{\text{int}} + 1), \tag{4}$$

$$x(s, l) = (1 - s_{\text{dec}})x(s_{\text{int}}, l) + s_{\text{dec}}x(s_{\text{int}} + 1, l). \tag{5}$$

In the equations (3)-(5), s_{int} and l_{int} are indices for slicing the input feature i 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 the 1×1 convolution layer which generates s and l, and update s_{int} and l_{int} whenever the decimal values of s_{dec} and l_{dec} are out of the range of $(0, 1)$. Figure 4 also illustrates the backward process for updating s_{dec} and l_{dec}. Training a neural network with DCS using the proposed differentiable searching algorithm is described in Algorithm 1. It is important to note that for illustration purpose Algorithm 1 involves the optimization of only one window. The same optimization process can be performed for all the windows of all the input features.

2.3 DCS WITH FIXED BACKBONE AND LEARNABLE BACKBONE

With our novel differentiable searching method for DCS in in Algorithm 1, the searching for the architecture of DCS can be performed by regular SGD. As a result, DCS can be incorporated into arbitrary neural network backbone, and the weights of the backbone and the architecture of DCS modules can be jointly trained by SGD. To evaluate the performance of DCS for person Re-ID, we designed two models where DCS is incorporated into popular feature extraction backbones, such as MobileNetV2, and the learnable backbone by FBNetV2 (Wan et al., 2020), leading to DCS-FB and DCS-DNAS respectively.

Algorithm 1 Training a Deep Neural Network with Differentiable Channel Selection

Input: Maximum iterations T, mini-batch of training samples $\{X_1, X_2, ..., X_N\}$, network learning rate η, DCS learning rate γ, and the loss function of the network $L(X, \Omega, \text{conv}_{1\times1})$, the percentage p used for the search of DCS in each mini-batch.
Output: The network parameters Ω, the parameters of the $\text{conv}_{1\times1}$ which generates DCS location s and key dimension l.

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 $\nabla_\Omega L(X_i, \Omega, \text{conv}_{1\times1})$
    $\Omega \leftarrow \Omega - \eta \nabla_\Omega L(X_i, \Omega, s_{\text{dec}}, l_{\text{dec}})$
  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 s_{dec} denoted by $\nabla_{s_{\text{dec}}} L(X_i, \Omega, \text{conv}_{1\times1})$
    Update the weights of the $\text{conv}_{1\times1}$ using $\nabla_{s_{\text{dec}}} L(X_i, \Omega, \text{conv}_{1\times1})$
    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 l_{dec} denoted by $\nabla_{l_{\text{dec}}} L(X_i, \Omega, \text{conv}_{1\times1})$
    Update the weights of the $\text{conv}_{1\times1}$ using $\nabla_{l_{\text{dec}}} L(X_i, \Omega, \text{conv}_{1\times1})$
    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

3 EXPERIMENTS

In this section, we demonstrate the performance of DCS for the Re-ID task in Section 3.1, the object detetion task in Section 3.2, and the image classification task on the ILSVRC-12 dataset (Russakovsky et al., 2015) in Section 3.3.

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 (Li et al., 2018b)	Inception	160 × 64	4.5	0.55	79.9	92.3	63.8	80.5	-	-
OSNet (Zhou et al., 2019)	OSNet	256 × 128	2.2	0.98	81.0	93.6	68.6	84.7	43.3	71.0
Auto-ReID (Chen et al., 2019)	ResNet50	384 × 128	13.1	2.05	74.6	90.7	-	-	-	-
RGA (Zhang et al., 2020)	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
Pre-trained on ImageNet										
AANet (Tay et al., 2019)	ResNet50	256 × 128	>23.5	-	85.3	94.7	75.3	84.0	-	-
CAMA (Yang et al., 2019)	ResNet50	256 × 128	>23.5	-	84.5	94.7	72.9	85.8	-	-
BAT-Net (Fang et al., 2019)	ResNet50	256 × 128	>23.5	-	87.4	95.1	77.3	87.7	-	-
ABD-Net (Chen et al., 2019)	ResNet50	384 × 128	69.17	14.1	88.28	95.6	78.59	89.0	60.8	82.3
Auto-ReID (Quan et al., 2019)	ResNet50	384 × 128	13.1	2.05	85.1	94.5	-	-	52.5	78.2
OSNet (Zhou et al., 2019)	OSNet	256 × 128	2.2	0.98	84.9	94.8	73.5	88.6	52.9	78.7
RGA (Zhang et al., 2020)	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**	**61.2**	**82.7**
DCS-DNAS(ours)	-	384 × 128	13.2	1.239	88.2	95.6	79.5	90.6	61.0	82.5

3.1 DCS WITH FIXED BACKBONES AND LEARNABLE BACKBONES FOR PERSON RE-ID

Datasets and Evaluation Metrics. we evaluate our proposed DCS modules on three public person Re-ID datasets, i.e., Market-1501 (Zheng et al., 2015a), DukeMTMC-reID (Ristani et al., 2016), MSMT17 (Wei et al., 2018). Market-1501 (Zheng et al., 2015a) consists of $32,668$ annotated person images of $1,501$ identities, in which $12,936$ images of 751 identities are used for training and $19,732$ images of 750 identities are used for testing. All images are shot from 6 different cameras. DukeMTMC-reID (Ristani et al., 2016) was originally proposed for video-based Re-ID. It has $16,522$ training images of 702 identities. The other $2,228$ query images of 702 identities and $17,661$ gallery images are in the test set. MSMT17 (Wei et al., 2018) is the largest and the most challenging public person re-ID dataset, which includes $126,441$ person images of $4,101$ identities detected by Faster R-CNN (Ren et al., 2015). The training set has $32,621$ person images of $1,041$ identities, and the test set has $93,820$ images of the other $3,060$ identities. CUHK03 (Li et al., 2014a) consists of $13,164$ person images of $1,467$ identities, of which images of 767 identities are in the training set and the remaining 700 identities are in the test set.

Standard Re-ID metrics top-1 accuracy (R1), and the mean Average Precision (mAP) are used to evaluate the performance of DCS and baseline models. Note that for the fairness of comparison, re-ranking (Zhong et al., 2017) and multi-query fusion (Zheng et al., 2015b) were not used.

DCS modules are compatible with popular manually designed feature extraction CNN backbones, such as InceptionNet (Szegedy et al., 2015), ResNet (He et al., 2016b), and MobileNetV2 (Sandler et al., 2018). In our experiments, we evaluate the performance of DCS 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 (Zhang et al., 2020; Chen et al., 2019), the attention modules are added after each convolution stage. Table 1 shows the performance of DCS-FB and competing baselines on the three person Re-ID datasets. The fixed backbone can be pre-trained on the ImageNet dataset (Russakovsky et al., 2015) or not. It can be observed that DCS-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, DCS-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 79.3% and 90.1% on DukeMTMC-reID. This model is so far the most compact model with best performance on this dataset, to the best of our knowledge. Table 1 also shows that DCS-FB achieves state-of-the-art performance on Market1501 when pre-trained on ImageNet. To integrate learnable backbone with DCS, we adopt the supergraph proposed in FBNetV2 (Wan et al., 2020) and jointly train the network backbone and DCS modules following Section A.1. 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. On MSMT17, DCS-DNAS achieve the best mAP and R1 with only 12.8% of the FLOPs required by the second best model in accuracy, ABD-Net (Chen et al., 2019). For each neural backbone, a DCS module is inserted after each of its four stages. Section A.2 of the appendix includes the detailed architecture of MobileNetV2,MobileNetV2-200 and the supergraph of FBNetV2 used in the experiments. For DCS-FB and DCS-DNAS, the architecture of DCS, 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.

3.1.1 TRAINING DETAILS OF DCS WITH FIXED BACKBONES AND LEARNABLE BACKBONES

DCS 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 DCS 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 fiipping, and random erasing to augment the data. Both identification loss with label smoothing (Szegedy et al., 2016) and triplet loss with hard mining (Hermans et al., 2017) 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 DCS with fxied backbone is shown in Tabel 1 of the main paper, where DCS-FB denotes our model. In the experiments, DCS 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.

DCS 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 DCS 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 fiip, 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.

3.2 DCS FOR OBJECT DETECTION

We evaluate the performance of DCS for the task of object detection. Following (Liu et al., 2022), we choose Cascade Mask R-CNN (Cai & Vasconcelos, 2018) with ConvNeXt backbones for object detection on COCO. We insert the DCS block after the four convolutional stages of the ConvNeXt backbones. In the search phase of the experiment, we train our model with AdamW optimizer for 50 epochs. The weight decay is 0.05. The batch size is set to 16. The initial learning rate is 0.001, which is divided by 10 at epochs 25, 35, and 45. In each epoch, the network weights are trained with 50% of the training samples, and the architecture parameters are trained with the remaining 50% of the training samples. After the search finished, we pre-train the backbone with searched DCS on ImangeNet, following the same settings for the training of ConvNeXt on ImageNet in (Liu et al., 2022). Next, we finetune our model on the COCO dataset with AdamW optimizer for 70 epochs. The weight decay is 0.05. The batch size is set to 16. The initial learning rate is 0.0001, which is divided by 10 at epochs 35, 50, and 60. The mAP of the detection and the architecture details are shown in Table 2. It should be emphasized that DCS usually leads to better mAP with a compact architecture. For example, ConvNeXt-S+**DCS** achieves the same mAP as the larger ConvNeXt-B model while maintaining the smaller architecture of ConvNeXt-S, evidencing the advantage of DCS.

Table 2: Performance of DCS for Object Detection. SA stands for the vanilla self-attention

Methods	FLOPs(M)/#Params(M)	mAP on COCO
ConvNeXt-T (Liu et al., 2022)	86.43/741.6	50.4
ConvNeXt-T+SA	86.79/741.7	51.0
ConvNeXt-T+**DCS**	86.91.1/741.7	51.4
ConvNeXt-S (Liu et al., 2022)	108/827	51.9
ConvNeXt-S+SA	109.45/827.8	52.2
ConvNeXt-S+**DCS**	110.53/828.9	**52.7**
ConvNeXt-B (Liu et al., 2022)	146/964	**52.7**

3.3 DCS FOR IMAGE CLASSIFICATION ON IMAGENET

We also evaluate DCS for the task of large-scale image classification. In the searching phase, we randomly sample 100 classes from the original 1000 classes of the ImageNet dataset as the training data. Both the architecture parameters and the regular network weights are trained for 100 epochs with a batch size of 128. In each epoch, the network weights are firstly trained with 80% of training data. Then, the architecture parameters are trained on the rest 20% of training data. For the series of models with MobileNetV2 backbone, Adam is used to optimize both the network weights and architecture parameters. The initial learning rate for network weights and architecture parameters is set to 0.05 and 0.01, and it decays following a cosine decaying schedule. For the series of models with FBNetV2 backbone, SGD is used to optimize the network weights with an initial learning rate of 0.1, and Adam is used to optimize the architecture parameters with an initial learning rate of 0.01.

In the training phase, the series of models with MobileNetV2 backbone are trained with Adam with a batch size of 128 for 150 epochs. The initial learning rate is set to 0.05, and it decays following a cosine decaying schedule. The series of models with FBNetV2 backbone are trained with SGD with a batch size of 128 for 350 epochs. We set the initial learning rate to be 0.05, and it is divided by 10 at 100, 200, and 300 epochs. We demonstrate the performance of DCS with a larger supergraph of FBNetV2, which is denoted by FBNetV2-L, in Table 3. It can be observed that FBNetV2-L with DCS still improves the performance of FBNetV2-L with the vanilla self-attention.

Table 3 shows the performance of DCS with two neural backbones, MobileNetV2 and FBNetV2, on the ILSVRC-12 dataset (Russakovsky et al., 2015). The architecture of the MobileNetV2 and the supergraph of FBNetV2 and the location where the vanilla Self-Attention (SA)/DCS (DCS) modules are inserted into the corresponding neural backbones are the same as that used for the Re-ID task, 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, DCS further improves the Top-1 accuracy of SA. Notably, such accuracy improvement of DCS over SA is promising, as the resultant model even enjoys slightly less FLOPs by removing the redundant channels not selected by DCS.

Table 3: Performance of DCS for Image Classification on ImageNet. SA stands for the vanilla self-attention

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+**DCS**	329.6/3.51	**72.5**	**91.5**
FBNetV2	330.3/7.50	75.7	92.5
FBNetV2+SA	333.2/7.50	75.9	92.9
FBNetV2+**DCS**	331.7/7.50	**76.4**	**93.0**
FBNetV2-L+**DCS**	626/9.99	**78.1**	**94.0**

4 CONCLUSION

We presented Differentiable Channel Selection (DCS), which searches for the informative channels when computing the feature affinity matrix in attention modules. In contrast with conventional self-attention modules, DCS searches for the location and key dimension of channels in a continuous space comprising uncountably infinite options. DCS with fixed or learnable neural backbones outperforms other competing methods for three computer vision tasks, person Re-IDentification methods (Re-ID), object detection, and image classification, with compact neural architectures with the help of neural architecture search methods.

REFERENCES

Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 6154–6162, 2018.

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*, pp. 8351–8361, 2019.

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*, pp. 8030–8039, 2019.

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

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*, pp. 770–778, 2016b.

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

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.

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*, pp. 1492–1500, 2018a.

Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014a.

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*, pp. 152–159, 2014b.

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*, pp. 2285–2294, 2018b.

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*, pp. 10297–10306, 2020.

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

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11976–11986, June 2022.

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*, pp. 3750–3759, 2019.

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, pp. 4780–4789, 2019.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

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*, pp. 17–35. Springer, 2016.

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.

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*, pp. 4510–4520, 2018.

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

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*, pp. 1–9, 2015.

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*, pp. 2818–2826, 2016.

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*, pp. 7134–7143, 2019.

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*, pp. 5998–6008, 2017.

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*, pp. 12965–12974, 2020.

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*, pp. 7794–7803, 2018.

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*, pp. 79–88, 2018.

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 (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pp. 3–19. Springer, 2018.

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*, pp. 10734–10742, 2019.

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*, pp. 1389–1398, 2019.

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*, pp. 3186–3195, 2020.

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*, pp. 1077–1085, 2017.

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*, pp. 10073–10082. IEEE, 2020.

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*, pp. 1116–1124, 2015a.

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*, pp. 1116–1124, 2015b.

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.

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*, pp. 1318–1327, 2017.

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*, pp. 3702–3712, 2019.

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

A More Details about Experiments

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
DCS Module Inserted					
bottleneck	6	32	3	21	2
DCS Module Inserted					
bottleneck	6	64	4	19	2
bottleneck	6	96	3	18	1
DCS Module Inserted					
bottleneck	6	160	3	3	2
bottleneck	6	320	1	1	1
DCS Module Inserted					
conv2d	-	1280	1	1	1
avgpool	-	-	1	1	-
conv2d	-	-	-	-	

Table 4: 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.

A.1 DCS 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.

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
DCS 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
DCS 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
DCS 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
DCS Module Inserted					
conv2d	-	1984	1	1	1
avgpool	-	-	1	1	1
fc	-	-	1	-	-

Table 5: 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).

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
DCS 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
DCS 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
DCS 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
DCS Module Inserted				
conv2d	-	1984	1	1
avgpool	-	-	1	1
fc	-	-	-	-

Table 6: 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).

MobileNetV2 (Howard et al., 2017; Sandler et al., 2018) 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 (Wang et al., 2018; Zhang et al., 2020), we insert DCS modules to the end of each convolution stage of MobileNetV2. To further explore the potential of DCS 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 (He et al., 2016a). Table 4 shows the structure of MobileNetV2 and MobileNetV2-200 as well as the positions of DCS modules.

We also combined DCS modules and the supergraph of FBNetV2 (Wan et al., 2020) to propose DCS-DNAS where the neural network backbone and the DCS modules are jointly trained. FBNetV2 employs Differentiable Neural Architecture Search (DNAS) algorithm to learn the backbone archi-

tecture 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 DCS modules into the supergraph of FBNetV2, and the supergraph of FBNetV2 and the positions of DCS modules are shown in Table 6.

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 DCS modules can be jointly performed by SGD. The proposed DCS-DNAS jointly searches for the backbone architecture and the architecture of DCS 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 5. Combining our DCS module with DNAS algorithm, we are able to search for models of different size. To test the performance of our DCS 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 6 shows the supergraph of the deeper FBNetV2-Large.

A.2 Component Analysis and Ablation Study of DCS

To diagnose the effectiveness of our DCS module, we perform the ablation studies on Market1501.

Differentiable Channel Selection vs. Baselines To verify the effectiveness of the search of position and key dimension in DCS, we perform a step-by-step ablation study. Various attention searching mechanisms are incorporated into the conventional attention module on MobileNetV2 with the same training setting used before. The conventional self-attention module serves as the baseline, which is termed as SA in Tabel 7. As shown in Table 7, either the search of position or key dimension improves the performance compared to the baseline. The proposed DCS module, which combines the search of position and key dimension, leads to the best result in Tabel 7.

Table 7: Ablation study of differentiable attention

Model	Search Options	Market-1501	
		mAP	R1
MobileNetV2	baseline	78.9	92.0
MobileNetV2+SA	baseline	82.3	92.8
MobileNetV2+DCS	position	82.9	93.1
MobileNetV2+DCS	key dimension	83.2	93.0
GSA-FB	position + key dimension	83.1	93.2
DCS-FB	position + key dimension	83.6	93.6
DCS-DNAS	position + key dimension	87.1	94.7

Differentiable Channel Selection vs. Attention Search by Gumble Softmax based DNAS

Model	Backbone	Market-1501		DukeMTMC-reID		MSMT17	
		mAP	R1	mAP	R1	mAP	R1
MobileNetV2+SA	MobileNetV2	82.3	92.8	71.4	84.4	36.2	62.8
GSA-FB	MobileNetV2	83.1	93.2	72.5	85.3	36.3	62.7
DCS-FB	MobileNetV2	83.6	93.6	73.3	85.5	36.5	63.0

Table 8: Ablation study of differentiable attention

To further demonstrate the advantage of our novel differentiable searching algorithm over the existing Gumbel Softmax based DNAS algorithm. We designed a baseline attention module that searches for the position and key dimension with the Gumbel Softmax based DNAS algorithm proposed in FBNetV2(Wan et al., 2020), which is termed Gumbel Softmax Attention (GSA). We designed 4 candidate downsampling ratios, $\{1, 1/2, 1/4, 1/8\}$, for the key dimension of GSA. For each downsampling ratio $r \in \{1, 1/2, 1/4, 1/8\}$, we set the candidate start positions to $\{0, \lfloor c \times r \rfloor, ..., \lfloor c \times (\frac{1}{r} - 1)r \rfloor\}$, where c is the number of channels. As a result, there are 15 options for each GSA module.

The comparison between GSA-FB (GSA with Fixed Backbones) and DCS-FB on all the three public person Re-ID benchmarks are shown in Table 8, and DCS-FB always outperforms GSA-FB. As shown in Table 8, DCS-FB outperforms GSA-FB, which reveals the advantage of our novel differentiable searching algorithm over the existing Gumbel Softmax based DNAS algorithm. SA standards for Self-Attention in Table 8.

B RUNNING TIME

We performed our experiments on four Tesla-V100 GPUs. The batch size is set to 64 for both search and training phase. We searched for the architecture for 300 epochs and fine-tuned the model for another 600 epochs. Table 9 shows the GPU hours of searching and fine-tuning on the Market1501 data set with the input size of 256×128.

Model	Backbone	Searching	Training
DCS-FB	MobileNetV2	3.6	5.1
DCS-FB	MobileNetV2-200	6.2	8.3
DCS-DNAS	FBNetV2	9.6	10.1

Table 9: Running time of different models on Market1501

C DIFFERENTIABILITY

We explained in Section 2.2 of the main paper that the loss function L of the neural network is differentiable with respect to the decimal parts of the location s and the key dimension l, namely s_{dec} and l_{dec}. In fact, L is differentiable with respect to s_{dec} when $0 < s_{\text{dec}} < 1$. In practice, we can always slightly adjust the learning rate γ in Algorithm 1 so as to avoid the case that s_{dec} is 1 or 0. The same argument is applied to l_{dec}. Moreover, as one could expect, we never observed that s_{dec} or l_{dec} is exactly 1 or 0 throughout our experiments, and such cases are almost surely impossible in practice.