International Journal of Computer Vision
https://doi.org/10.1007/s11263-024-02194-5

# Kill Two Birds with One Stone: Domain Generalization for Semantic Segmentation via Network Pruning

Yawei Luo[1] · Ping Liu[2] · Yi Yang[1]

Received: 12 October 2023 / Accepted: 8 July 2024


**Abstract**
Deep models are notoriously known to perform poorly when encountering new domains with different statistics. To alleviate this issue, we present a new domain generalization method based on network pruning, dubbed NPDG. Our core idea is to prune the filters or attention heads that are more sensitive to domain shift while preserving those domain-invariant ones. To this end, we propose a new pruning policy tailored to improve generalization ability, which identifies the filter and head sensibility of domain shift by judging its activation variance among different domains (*unary manner*) and its correlation to other filters (*binary manner*). To better reveal those potentially sensitive filters and heads, we present a differentiable style perturbation scheme to imitate the domain variance dynamically. NPDG is trained on a single source domain and can be applied to both CNN- and Transformer-based backbones. To our knowledge, we are among the pioneers in tackling domain generalization in segmentation via network pruning. NPDG not only improves the generalization ability of a segmentation model but also decreases its computation cost. Extensive experiments demonstrate the state-of-the-art generalization performance of NPDG with a lighter-weight structure.

**Keywords** Domain adaptation · Network pruning · Semantic segmentation · Adversarial training

## 1 Introduction

Deep learning-based methods have achieved remarkable success in various vision tasks when the training and testing data are independent and identically distributed (*i.i.d.*). In practical applications, however, the deep models are usually deployed to a new environment with different statistics, making their performance drop significantly. To alleviate the issue, domain generalization (DG) (Muandet et al., 2019; Zhou et al., 2021) is proposed to enhance the generalization capability of deep neural networks to unseen target distributions after training on source domain(s). Compared to domain adaptation (DA) (Zhao et al., 2020; Hoyer et al., 2022; Zhang et al., 2021), DG poses more challenging learning scenarios since it cannot acquire any data from the target domain during training.

DG has typically been studied under two different settings, i.e., multi source-based and single source-based. Multi-source DG (MSDG) (Li et al., 2017; Li & Hospedales, 2020) targets at looking across various sources for shared factors in the hypothesis that they will hold also for any new target domain.

However, the assumption of available samples from multiple domains can not always hold. A more practical scenario lies in the single-source DG (SSDG), which is attracting increasing attention in the research community and is also the focus of our work. Since only one domain can be accessed, capturing the invariances across domains becomes non-trivial. A popular trend is to create multiple augmented domains from the source domain via data augmentation or style transfer to mimic unseen domains. For example, previous works such as Yue et al. (2019), Tjio et al. (2021) utilize source domain images and style images to create stylized samples. Those prior works assume that if the network can

Communicated by Nicu Sebe.

✉ Yi Yang
yangyics@zju.edu.cn

Yawei Luo
yaweiluo@zju.edu.cn

Ping Liu
pino.pingliu@gmail.com

[1] Zhejiang University, Hangzhou, China

[2] Ping Liu is with the Department of Computer Science and Engineering, University of Nevada, Reno, NV, USA

Published online: 27 July 2024

be exposed to enough diverse domains in training, it should perform well to unseen target domain data. However, the data generation/stylization in most previous works (Yue et al., 2019; Tjio et al., 2021) is conducted independently with the downstream target task, e.g., image classification, semantic segmentation, etc., making the final results sub-optimal.

Based on the diversified source domains, prior DG works such as Yue et al. (2019), Tjio et al. (2021) tried to make the model to learn domain invariant representations to achieve a high generalization ability. Further, works such as Cai et al. (2019), Peng et al. (2019), Zou et al. (2020) argue that explicitly disentangle the latent representation into domain-relevant and domain-irrelevant groups should be more beneficial to train a generalized model. Although the disentangling strategy has been proven effective in enhancing the model generalization ability, it still has two issues that need to be solved: (1) it is necessary to carefully design network structures or loss functions to explicitly disentangle features, both of which are non-trivial; (2) the domain-irrelevant features still occupy non-negligible storage space and cost additional inference time, bringing unnecessary waste.

In this paper, we propose a new single-source domain generalization method based on network pruning, dubbed NPDG. We follow the spirit of SSDG to generate new domains and distill the domain invariant knowledge across them but introduce critical differences to prior arts. Our core idea is to prune the filters that are more sensitive to domain shift while preserving those domain-invariant ones. To this end, we propose a tailored pruning policy to improve generalization ability, which identifies the filter or attention head sensibility of domain shift by judging its activation variance among different domains (unary manner) and its correlation to other filters or heads (binary manner). Specifically, if the activation of a filter or head changes dramatically as the domain changes, and is highly correlated to other filters' activations within the same layer, we regard this filter or head as a domain-sensitive one. To better reveal those potentially sensitive filters and heads, we present a differentiable style perturbation scheme to imitate the domain variance dynamically, magnifying the different behaviors between domain variant and invariant filters or heads (Fig. 1).

Our main contributions can be summarised as follows:

- We propose a new domain generalization method for semantic segmentation based on network pruning (NPDG). NPDG is among the pioneers in tackling style shift via network pruning, which not only improves the generalization ability of a deep model but also decreases its computation cost.
- We present a differentiable style perturbation (DSP) module to imitate the domain variance dynamically, which can better reveal those potential domain-sensitive filters in our challenging single-source scenario.
- We verify the effectiveness of NPDG on the widely used cross-domain segmentation datasets. Experimental results on both CNN-based and Transformer-based backbones demonstrate the state-of-the-art generalization performance achieved by NPDG.

## 2 Related Work

### 2.1 Domain Adaptation and Generalization

Domain adaptation (DA) (Zhao et al., 2020; Long et al., 2015; Li et al., 2019; Zheng & Yang, 2021) and domain generalization (DG) (Zhou et al., 2021; Wang et al., 2021) both aim to train a model that performs well on an unlabeled target domain. The target domain is with a statistical distribution different from the source domain(s). The critical difference between DA and DG consists in the accessibility of target data during training. The existing DA strategies include: aligning distribution between domains (Tzeng et al., 2017; Luo et al., 2019, 2021; Wang et al., 2023), synthesizing labeled target samples (Luo et al., 2020), or conducting self-training based on estimated pseudo labels for target samples (Zhang et al., 2019). Compared to DA, DG is more challenging since no target data is accessible when training. Inaccessible to target data makes the previous DA methods inapplicable in DG. Based on the number of source domains during training, existing DG works can be roughly divided into multi-domain (Zhao et al., 2021; Gong et al., 2021; Fu et al., 2021) and single-domain methods (Qiao & Peng, 2021; Wang & Jiang, 2021; Huang et al., 2021; Zhao et al., 2022, 2023). Given more than one source domain, works such as Fu et al. (2021), Zhao et al. (2021), Zhong et al. (2022) assume that each domain intrinsically shares certain domain invariant information. Accordingly, training a model to distill the domain-invariant features across these source domains is expected to perform well on the unseen domains. In contrast, single-source DG restricts that the training data only contains samples from a single source domain. Recently, single-source DG (Qiao & Peng, 2021; Wang & Jiang, 2021; Huang et al., 2021; Tjio et al., 2021) has attracted increasing attention as multiple source domain collection and annotation are time-consuming and labor-expensive. This paper also focuses on the single-source DG. Diverging from conventional methods (Zhao et al., 2023; Zhong et al., 2022; Huang et al., 2021; Choi et al., 2021) that primarily emphasize providing diverse training data to enhance generalizability, NPDG takes a distinctive approach by scrutinizing the internal characteristics of a segmentation model. Specifically, it focuses on pruning filters or attention heads that are sensitive to the domain. To our knowledge, NPDG is among the pioneers in tackling domain generalization for semantic segmentation via network pruning, which not only improves the

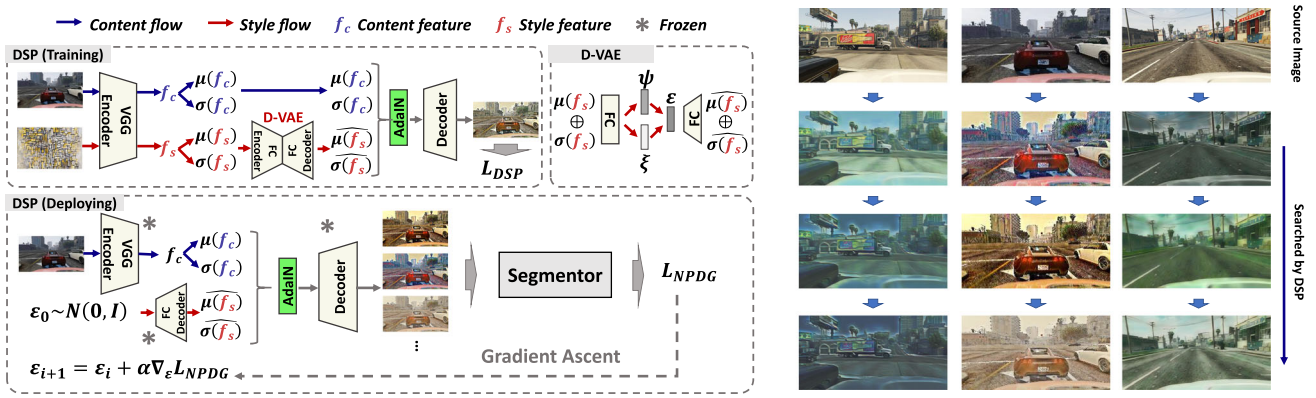


**Fig. 1** The detailed network structure of **DSP**. DSP adopts an extra D-VAE in the latent space of AdaIN Huang and Belongie (2017) in order to encode the style statistics of the training data (i.e., $\mu(f_s) \oplus \sigma(f_s)$) into a standard distribution. During the training of DSP, we train D-VAE and AdaIN together using Eq. 2. After training, we fix the parameters of DSP and deploy it as a "new domain generator" to the segmentor (in deploying stage, the FC encoder of D-VAE is abandoned and FC decoder is fed with the sampling $\varepsilon$). DSP and the segmentor can be trained in an end-to-end manner, where $\varepsilon$ is updated by *gradient ascent*, bootstrapping DSP to generate higher diversified images for the downstream segmentor pruning. On the right, we show some increasingly diversified data produced by DSP

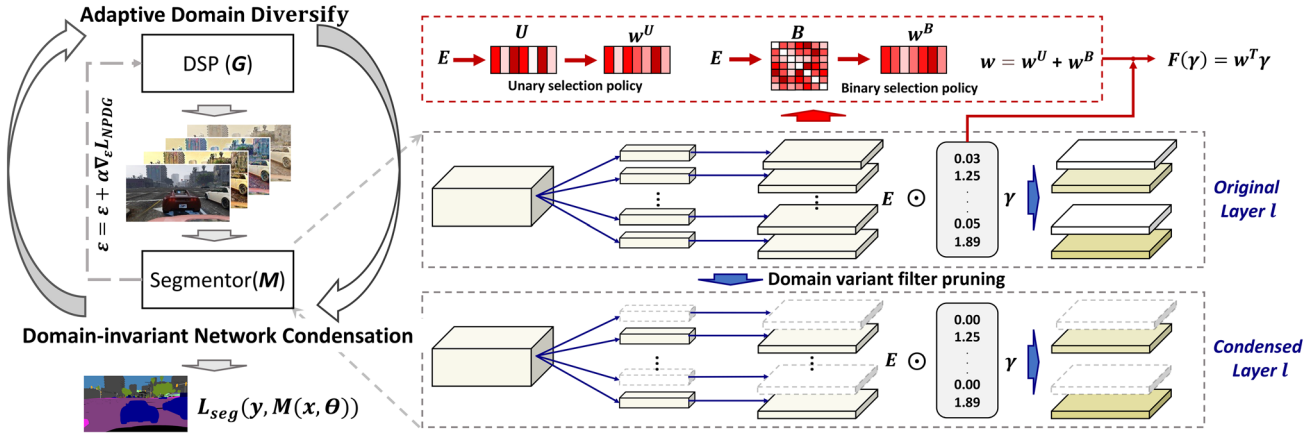


**Fig. 2** Main framework of NPDG. Generally, the domain diversifying and domain variant filter pruning are conducted in an alternative and adaptive manner. On one hand, domain diversifying is leveraged to enhance the data variations based on the sole source domain data. Benefiting from the proposed DSP module, the generated data is highly adaptive as it considers the capability of target network $\mathcal{M}$. On the other hand, based on the highly diversified data generated by DSP, the proposed unary and binary sensitivities are utilized to compress the domain-variant filters, making the pruned model more domain-generalized. The two processes collaborate with each other until a model with high generalization ability is achieved

generalization ability of a deep model but also decreases its computation cost (Fig. 2).

## 2.2 Network Pruning

Network pruning refers to removing components from a neural network to reduce network complexity (LeCun et al., 1989; Hassibi & Stork, 1992). Recently, one of the primary objectives of pruning is to reduce the number of parameters in the network for acceleration and compression while minimizing the impact on model performance. These pruning techniques can be broadly categorized into two types: unstructured (Han et al., 2015; Rosenfeld et al., 2021; Sehwag et al., 2020) and structured pruning (He & Xiao, 2023; He et al., 2019; Qian & Klabjan, 2021; He et al., 2020). Unstructured pruning involves removing individual connections (weights) within the network, resulting in unstructured sparsity. While unstructured pruning often achieves high compression rates, it typically requires specialized hardware or library support for practical acceleration. On the other hand, structured pruning involves removing entire filters from the neural network. This approach can lead to realistic acceleration and compression without specialized hardware. Luo et al. (2017) adopted the statistics information from the next layer to guide the filter selections. Dubey et al. (2018) aimed to obtain a decomposition by minimizing the

reconstruction error of training set sample activation. He et al. (2018) proposed to select filters with a "$\ell_2$-norm" criterion and softly prune those selected filters. However, most of the current methods are conducted on image classification and are not tailored for finer-grained tasks, such as segmentation. An exception lies in CAP He et al. (2021), which utilizes the contextual priors to guide the pruning of unimportant channels. Nevertheless, in these previous network pruning works, the training data and testing data were assumed *i.i.d.* and the domain gap was not taken into consideration.

More recently, some network pruning methods for cross-domain scenarios have been proposed (Cai et al., 2021; Nguyen et al., 2022; Sun, 2023; Wu et al., 2024). Cai et al. (2021) proposed to search for a subnetwork that can help with multi-source Ki67 image analysis. Chen et al. (2019) prompted the idea of transferring knowledge from a resource-rich source domain to a target domain with limited data to perform model compression. Nguyen et al. (2022) found that pruning is efficient in the domain generalization setting even with a strong compression rate for classification tasks, which is beneficial for hardware platform deployment. Long et al. (2023) proposed Discriminative Microscopic Distribution Alignment (DMDA), aiming at alleviating the inconsistency between feature generalizability and feature discriminability. Tian et al. (2022) proposed NCDG, which improves the generalization capability by maximizing the neuron coverage of DNN with the gradient similarity regularization between the original and augmented samples. Although these works mentioned above explore network pruning to enhance generalization ability, they all focus on coarse-grained visual tasks such as classification and recognition. In comparison to these existing pruning methods, NPDG stands out for two key reasons: (1) it is specifically customized for segmentation tasks, particularly addressing the domain shift caused by style changes, and (2) it demonstrates superior generalization ability on various benchmarks in previously unseen domains.

## 3 Methodology

### 3.1 Problem Settings

In the training process of SSDG, we only have access to a single-source data $X_S$ with labels $Y_S$ where $(X_S, Y_S) \sim \mathcal{P}_s$. The target data is in a different distribution $\mathcal{P}_t$ where $\mathcal{P}_s \neq \mathcal{P}_t$, and only accessible in the testing stage. Our goal is to learn a model $\mathcal{M}$ with the weights $\theta$ based on those accessible source data to correctly predict the labels $Y_T$ for the target domain $X_T$, where $(X_T, Y_T) \sim \mathcal{P}_t$.

Overall, our main idea is to find the filters behaving in a domain-invariant manner and *only* leverage those filters to learn domain-invariant knowledge. In an ideal case, if we can prune the filters sensitive to domain variances, the pruned network is expected to have a high generalization capability and inference speed. There are two cruxes to achieve this goal: 1) how to adaptively diversify the sole source domain to imitate domain variances for magnifying different behaviors between domain variant and invariant filters; 2) how to identify the filters sensitive to domain variance to achieve a compressed domain-general model. We detail our solutions in the following sections.

### 3.2 Diversify the Source Domain via Differentiable Style Perturbation

The first crucial step in our solution is to diversify the sole source domain to imitate domain variances by generating new stylized data with different variations. Most of the prior arts generate the stylized data in a static and independent manner. For example, a parameterized distribution, e.g., Gaussian distribution, is utilized to sample novel data. In the whole training process, the distribution is fixed without change, limiting the diversities of generated data. We argue that limited diversified data is not enough to identify the domain-sensitive filters. Instead, to benefit the domain-sensitive filter identification in the downstream stage, we propose a differentiable style perturbation (DSP) module to adaptively generate new data with high diversification. Unlike previous works, our DSP can dynamically produce out-of-distribution data based on the feedback from task model $\mathcal{M}$ so as to match the pruning state of $\mathcal{M}$.

DSP is motivated by AdaIN Huang and Belongie (2017). Vanilla AdaIN regards "style" as a pair of "mean $\mu(f_s)$" and "variation $\sigma(f_s)$" of the style image features $f_s$. To achieve style transfer, AdaIN scales the content features $f_c$ with $\sigma(f_s)$, and shifts it with $\mu(f_s)$:

$$\text{AdaIN}(f_c, f_s) = \sigma(f_s)\left(\frac{f_c - \mu(f_c)}{\sigma(f_c)}\right) + \mu(f_s), \tag{1}$$

where $\mu(.)$ and $\sigma(.)$ denote channel-wise mean and standard deviation operations, respectively.

At first, based on the vanilla AdaIN, we adopt an extra Domain Variational Auto-Encoder (D-VAE) in the latent space to encode the style statistics (i.e., $\mu(f_s) \oplus \sigma(f_s)$ where $\oplus$ denotes "concatenation") into a standard distribution. Then we decode a latent code $\varepsilon$ sampled from the distribution to reconstruct the style feature statistics. In the training stage, besides the conventional training scheme for AdaIN (please refer to Huang and Belongie (2017) for more details), we have two extra losses for training D-VAE. The overall training objective for DSP is to minimize the following losses:

$$\mathcal{L}_{DSP} = \mathcal{L}_{AdaIN} + \lambda_k \mathcal{L}_{KL} + \lambda_r \mathcal{L}_{Rec} \tag{2}$$

Within Eq. (2), the latter two terms form the training loss for D-VAE:

$$\mathcal{L}_{KL} = \mathrm{KL}[\mathcal{N}(\psi, \xi)||\mathcal{N}(0, I)] \tag{3}$$

$$\mathcal{L}_{Rec} = \|\mu(f_s) \odot \sigma(f_s), \widehat{\mu(f_s) \odot \sigma(f_s)}\|_2, \tag{4}$$

where $\widehat{\mu(f_s) \odot \sigma(f_s)}$ denotes the reconstructed style vector from a sampling $\varepsilon \sim \mathcal{N}(\psi, \xi)$[1].

Thanks to the merits of the Variational Auto-Encoder, DSP is able to generate arbitrary new domains by disturbing the sampled vectors $\varepsilon$ in the deploying stage. More than that, the gradient can be back-propagated directly to $\varepsilon$, making the whole generation differentiable. Particularly, when passing an inverse gradient to $\varepsilon$ (i.e., $\varepsilon \leftarrow \varepsilon + \beta \nabla_\varepsilon \mathcal{L}_{NPDG}$), DSP would produce "harder" stylized images to imitate more shifted domains. The highly diversified domain data can magnify the behaviors of those domain-sensitive filters, benefiting the downstream network pruning for domain generalization.

### 3.3 Network Pruning for Generalization

This section will describe our network pruning policy under the domain shift. Following Liu et al. (2017), we introduce a learnable scaling factor $\gamma$ for each filter. The scaling factor $\gamma$ is multiplied by the output activation map of the corresponding filter $A$, i.e., $A \odot \gamma$, to yield the final output of this layer. The filters are considered to be pruned if the scaling factors on them are near zeros after joint training. In practice, we reemploy $\gamma$ in the *Batch Normalization (BN)* Ioffe and Szegedy (2015) layer for CNN-based backbones as the scaling factor owing to the widespread adoption of BN in deep networks. Compared with CNN-based backbones, the attention heads are similar to the filters of a CNN layer. As a result, NPDG can be adapted easily to the transformer-based model, i.e., pruning those domain-sensitive attention heads. For transformer-based backbones, we assign the scaling factor to each attention activation as follows:

$$\mathrm{MHAtt}(\mathbf{x}, q) = \sum_{h=1}^{N_h} \gamma_h \mathrm{Att}_{W_k^h, W_q^h, W_v^h, W_o^h}(\mathbf{x}, q) \tag{5}$$

In our paper, we use self-attention, so the $x$ serves as the query $q$. $d$ denotes the feature dimension and $N_h$ represents the number of attention heads in a layer. The learnable weights of the attention head are denoted by $W_k^h$, $W_q^h$, $W_v^h \in \mathbb{R}^{d_h \times d}$, and $W_o^h \in \mathbb{R}^{d \times d_h}$. Additionally, $\gamma_h$ represents the learnable scaling factors, which play a similar role to those in CNN-based models.

Accordingly, the training objective of our approach can be written as

$$\mathcal{L}_{NPDG}(x, y) = \hat{\mathbb{E}}_{(x,y)\sim\mathcal{P}_s}[\mathcal{L}_{seg}(y, \mathcal{M}(x, \theta))] + \lambda_n \mathcal{F}(\gamma), \tag{6}$$

where $\mathcal{M}$ is the task model with its parameters denoted as $\theta$, $\mathcal{L}(.,.)$ denotes the task loss such as cross entropy. $\gamma = (\gamma_1, \gamma_2, ..., \gamma_m) \in \mathbb{R}^m$ denotes the vector of scaling factor and $m$ represents the number of filters. We employ a global network pruning strategy that is consistent with our baseline (Liu et al., 2017). We do not discriminate between scaling factors from different layers during sparse training. That is, in the last term in Eq. (6), scaling factors $\gamma$ come from the entire network. $\mathcal{F}(.)$ denotes the sparsity regularization function of $\gamma$. $\lambda_n$ controls the relative importance of the two terms, where we follow Liu et al. (2017) to set $\lambda_n = 1e-5$.

To achieve network pruning, the sparsity regularization function $\mathcal{F}$ is designed to push all scaling factors $\gamma_i$ to 0. For instance, *Network Slimming* Liu et al. (2017) realized this function as $L1$ regularization, i.e. $\mathcal{F}(\gamma) = \|\gamma\|_1$.

Evidently, $L1$ regularization suppresses all the factors indistinctively. But in domain generalization, a more reasonable pruning mechanism is to only suppress domain-sensitive filters or attention heads. Accordingly, we modify $\mathcal{F}(.)$ to reweight the vanilla $L1$ regularization as follows:

$$\mathcal{F}(\gamma) = w^\mathsf{T}\gamma = \sum_{i=1}^{n} w_i \gamma_i, \tag{7}$$

where $w = (w_1, w_2, ..., w_m) \in \mathbb{R}^m$ denotes the assigned weights on scaling factors during training. Negative $\gamma_i$s are clamped to 0. We design $w$ to reflect the domain sensitivity of the filters or heads. In this case, the scaling factors corresponding to the sensitive filters/heads would suffer from a more sparsity-induced penalty from $\mathcal{F}$. Specifically, we constitute $w$ from unary filter sensitivity $w^U$ and binary filter sensitivity $w^B$, where $w = \lambda w^U + (1-\lambda) w^B$ and $\lambda$ indicates the hyper-parameter to control the relative weight of unary and binary factors.

It is noteworthy here that both $w^U$ and $w^B$ are designed to capture the *relative* sensitivity for domain shift. Therefore, to eliminate the influence of the weight magnitude of a filter itself, we first pre-process the activation map $A$ by Instance Normalization (Ulyanov et al., 2016):[2]

$$\mathbf{E} = \gamma\left(\frac{\mathbf{A} - \mu(\mathbf{A})}{\sigma(\mathbf{A})}\right) + \beta, \tag{8}$$

[1] We use the notation $\psi$ for mean and $\xi$ for standard deviation in D-VAE, in order to avoid confusion to the $\mu(.)$ and $\sigma(.)$ in AdaIN.

[2] Unless otherwise noted, we employ $\mathbf{A}$ to denote the original feature maps and $\mathbf{E}$ as the standardized ones. $\mathbf{E}$ is for calculating $w$ only and is not forwarded to the next layer.

where $\mu(\mathbf{A})$ and $\sigma(\mathbf{A})$ denote the mean and standard deviation of $\mathbf{A}$ computed across spatial dimensions independently for each channel and each sample. $\gamma$ and $\beta$ are the affine parameters fixed to $\mathbf{1}$ and $\mathbf{0}$, respectively.

### 3.4 Unary Filter/Head Sensitivity

Unary filter sensitivity $w_i^U$ measures the activation variance of the $i_{th}$ filter/head under domain shift. As discussed above, in this paper we assume that the domain shift lies in the style difference while the pure semantic information is domain invariant. Based on DSP, here we forward a mini-batch of images that share the same content but different styles to the task model $\mathcal{M}$. Consequently, the filters or heads that are sensitive to the style transformation would yield more varied activations. We design Eq. (9) to capture such variance:

$$U_{E_i} = \frac{\sum_{n=1}^{N} \left\| \mathbf{E}_i^n - \overline{\mathbf{E}_i} \right\|_2}{ND} \in \mathbb{R}, \tag{9}$$

where $N$ is the batch size, $D$ is the dimension of an activation map. Specifically, $D = H_i W_i$ in a CNN-based model, while $D = T_i d_i$ in a transformer-based model ($T_i$ is the token amount). $\mathbf{E}_i^n$ denotes the standardized activation map from the $i_{th}$ filter/attention head on $n_{th}$ image in the mini-batch. $\overline{\mathbf{E}_i}$ denotes the average activation map from the $i_{th}$ filter/head across $N$ images. $H_i$, $W_i$ / $T_i$, $d_i$ denotes the spatial dimensions of the activation map from a filter/attention head.

To balance the magnitude of scaling factors across all the layers, we further normalize $U_{E_i}$ as in Eq. (10) to yield $w_i^U$:

$$w_i^U = C_l \times \frac{U_{E_i}}{\sum U_{E_i}} \texttt{ if } i \in \texttt{layer } l, \tag{10}$$

where $C_l$ is the total filter/head number in layer $l$.

### 3.5 Binary Filter/Head Sensitivity

The above unary filter sensitivity helps to identify the sensitive filter in an unary manner. In this section, we further consider the binary relation between filters under domain shift. Recent studies (Gatys et al., 2016; Choi et al., 2021) have demonstrated that feature correlations (*i.e.,* a Gram matrix or covariance matrix) capture the style information of images.

Motivated by this, we propose to iteratively pinpoint and prune the filters that are highly correlated to other filters within the same layer. Accordingly, the covariance matrix can be calculated with the following policy:

$$\mathbf{B}_{\mathbf{E}_l\mathbf{E}_l} = \frac{\sum_{n=1}^{N} ((\mathbf{E}_l^n)(\mathbf{E}_l^n)^\mathsf{T} - \mathbf{I})}{ND} \in \mathbb{R}^{C_l \times C_l}, \tag{11}$$

where $\mathbf{E}_l^n \in \mathbb{R}^{(1\times C_l)\times(D)}$ denotes the standardized activation maps from the all filters in layer $l$ on $n_{th}$ image in the mini-batch and $\mathbf{I}$ denotes the identity matrix. Specifically, $D = H_i W_i$ in a CNN-based model, while $D = T_i d_i$ in a transformer-based model ($T_i$ is the token amount). $C_l$ denotes the total filter/head number in layer $l$.

Owing to the standardization process by instance normalization, the scale of the activation map $\mathbf{E}$ is already fixed as a unit value. This enables us to measure the feature correlation degree by directly summing each row (or column since $\mathbf{B}_{\mathbf{E}_l\mathbf{E}_l}$ is symmetric) of $\mathbf{B}_{\mathbf{E}_l\mathbf{E}_l}$:

$$w_i^B = \frac{1}{C_l} \sum_{c=1}^{C_l} \mathbf{B}_{\mathbf{E}_l\mathbf{E}_l}(i,c) \texttt{ if } i \in \texttt{layer } l. \tag{12}$$

We visualize the pruning results of using binary filter sensitivity in Fig. 3. As can be observed, the correlated features (style information) can be significantly diminished in this manner.

With the above unary and binary filter sensitivity, we can obtain the final filter sensitivity $w$ by adding $w^U$ and $w^B$:

$$w = \lambda w^U + (1-\lambda) w^B \in \mathbb{R}^m \tag{13}$$

where $m$ is the total filter or attention head number in a network. $\lambda$ is a hyper-parameter to control the relative importance of $w^U$ and $w^B$, whose value is discussed in the experimental part. The yielded $w$ will be adopted to reweight the scaling factors as in Eq. (7).

**Algorithm 1:** Network Pruning for DG.

**Input:** Source domain data $X_S$; source domain label $Y_S$; a pre-trained DSP module $G = \{E, D, E_{vae}, D_{vae}\}$; segmentor $\mathcal{M}$ with parameter $\theta$; learning rate $\alpha$, $\beta$; max searching depth $n$; pruning threshold $t$.

**Output:** Optimal $\theta^*$.

Randomly initialize $\theta$;
**for** $x \in X_S$ **do**
  $f_c = E(x)$;
  Sampling $\varepsilon \sim \mathcal{N}(0, I)$;
  **for** $i = 1, ..., n$ **do**
    Reconstruct the style vector: $\widehat{\mu(f_s)} \odot \widehat{\sigma}(f_s) = D_{vae}(\varepsilon)$;
    Generate new-domain image $x_{new} = D(\text{AdaIN}(f_c, \mu(f_c), \sigma(f_c), \widehat{\mu(f_s)}, \widehat{\sigma(f_s)}))$;
    Calculate $\mathcal{F}(\gamma)$, in which $w$ is calculated by Eqs. (8)–(13);
    Calculate $\mathcal{L}_{NPDG}(x_{new}, y)$ based on Eq. (6) ;
    Update model parameters: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{NPDG}(x_{new}, y)$;
    Update sampling: $\varepsilon \leftarrow \varepsilon + \beta \nabla_\varepsilon \mathcal{L}_{NPDG}(x_{new}, y)$;
  Filter pruning: $Filter_i \leftarrow 0, \gamma_i \leftarrow 0$ if $\gamma_i < t$;
  If the target pruning rate is reached: `Stop Pruning and start Fine-tuning.`
return $\theta$ as $\theta^*$;

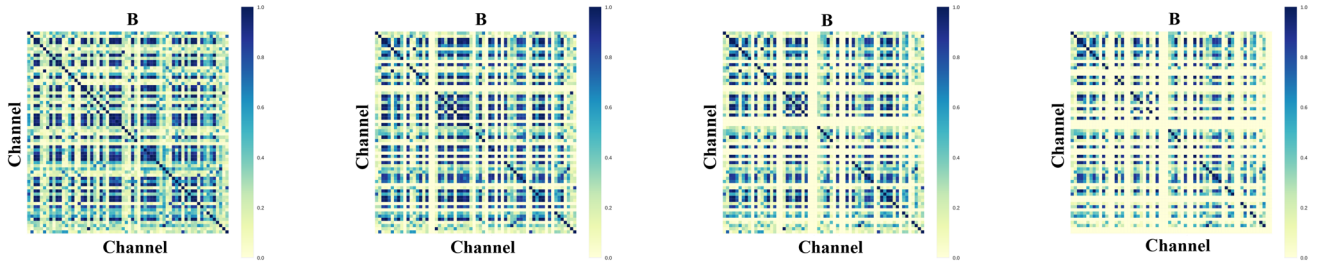


**Fig. 3** Visualization of the Filter Correlation Matrix (i.e., $\mathbf{B}_{\mathbf{E}_l\mathbf{E}_l}$ in Eq. (11). Sub-figures from left to right denote the matrix under target pruning rate 0%, 10%, 20%, 50%, respectively. The binary manner succeeds in diminishing those correlated features, but experimentally we found that an overlarge pruning rate (e.g., 50%) destroys useful information for segmentation to yield low mIoU

### 3.6 Training Pipeline

The learning process for NPDG is shown in Alg. 1. The domain diversifying and domain invariant filter pruning are conducted in an alternative and adaptive manner. On the one hand, domain diversifying is leveraged to enhance the data variations based on the sole source domain data. Benefiting from the proposed DSP module, the generated data is highly adaptive as it considers the capability of target network $\mathcal{M}$. On the other hand, based on the highly diversified data generated by DSP, the proposed unary and binary domain sensitivities are employed to compress the domain-variant filters, making the pruned model more domain-generalized. The two processes collaborate with each other until a model with high generalization ability is achieved. It should be noted that the pruning strategy is performed uniformly on all the layers. Since we have normalized the activation maps from the filters/heads before calculating the unary and binary terms, the magnitude of scaling factors is balanced across all the layers.

We follow the most of NP methods to divide the training process into pruning and fine-tuning stages. Once reaching the target pruning rate, e.g. 20%, we stop the pruning stage and start the fine-tuning process. In fine-tuning, the term $\mathcal{F}(\gamma)$ in Eq. (6) is deactivated while other settings are kept the same.

## 4 Experiments

### 4.1 Datasets Details and Evaluation Protocols

We evaluate NPDG on the DG tasks between 5 datasets, i.e., **GTA5** Richter et al. (2016), **SYNTHIA** Ros et al. (2016), **Cityscapes** Cordts et al. (2016), **BDD100K** Yu et al. (2020), and **Mapillary** Neuhold et al. (2017), among which the synthetic dataset GTA5 or SYNTHIA is adopted as the source domain while the real-world dataset Cityscapes, BDD100K, or Mapillary is employed as the unseen target domain.

**GTA5** contains 24, 966 high-resolution images, automatically annotated into 19 classes. **SYNTHIA** contains 9, 400 synthetic images compatible with the Cityscapes annotated classes. **Cityscapes** is with 5, 000 street scenes which are divided into a training set with 2, 975 images, a validation set with 500 images and a testing set with 1, 525 images. **BDD100K** contains diverse urban street-view images with the resolution of 1280 × 720, including 7, 000 training and 1, 000 validation images. **Mapillary** is a large-scale dataset consisting of 25, 000 high-resolution street scenes with a minimum resolution of 1920 × 1080. The semantic categories of these street-view datasets are highly overlapped but the style statistics differ from each other, while NPDG is tailored for such learning scenarios.

Besides the DG datasets, we extra leverage some data as the "style images" to pre-train DSP. Here we follow Huang and Belongie (2017) to use a painting dataset mostly collected from **WikiArt** Nichol (2016). However, there is no limit to using any other website data since no annotation is required for style images.

In terms of the evaluation metrics, we leverage Intersection over Union (IoU) to measure the segmentation performance and use the parameter amount and FLOPs of the pruned model to measure the pruning performance.

### 4.2 Implementation Details

We use PyTorch (Paszke et al., 2017) for our implementation. The training process is composed of two stages. In the first stage, we use source images and style images to train the DSP module. Here we follow Huang and Belongie (2017) to use a dataset of paintings mostly collected from WikiArt. However, there is no limitation in choosing any other website data since no annotation is required for style images. In our best DSP model, we set the hyper-parameters in Eq. 2 as $\lambda_k = 1.0$ and $\lambda_r = 5.0$, respectively.

In the second stage, we fix DSP and train the segmentation model within the NPDG framework. We leverage ResNet-101 He et al. (2016)-based DeepLab-v3+ Chen et al. (2018),

**Table 1** Domain generalization performance of the ***synthetic-2-real* tasks.** G,S,C,B,D denote *GTA5*, *SYNTHIA*, *Cityscapes*, *BDD100k* and *Mapillary*, respectively. P-rate denotes the target pruning rate for filters, and GFLOPS denotes the number of flops $\times 10^9$. The best and second-best results are marked with Bold and Underline, respectively

| Method | mIoU | | | GFLOPs | #Param. | mIoU | | | GFLOPs | #Param. |
|---|---|---|---|---|---|---|---|---|---|---|
| | G → *C* | G → *B* | G → *M* | | | S → *C* | S → *B* | S → *M* | | |
| Basel. Chen et al. (2018) | 34.6 | 27.7 | 29.6 | 84.7 | 168 M | 33.7 | 31.9 | 32.5 | 84.7 | 168 M |
| IBN-Net Pan et al. (2018) | 40.3 | 35.6 | 35.9 | 84.8 | 168 M | 37.5 | 33.0 | 33.7 | 84.8 | 168 M |
| SW Pan et al. (2019) | 36.1 | 36.6 | 32.6 | 84.7 | 168 M | 31.6 | 35.5 | 29.3 | 84.7 | 168 M |
| DRPC Yue et al. (2019) | 42.5 | 38.7 | 38.0 | 110.4 | 178 M | 37.6 | 34.3 | 34.1 | 110.4 | 178 M |
| RobustNet Choi et al. (2021) | 36.6 | 35.2 | 40.3 | 84.7 | 168 M | 35.8 | 31.6 | 30.8 | 84.7 | 168 M |
| GTR Peng et al. (2021) | 43.7 | 39.6 | 39.1 | 84.7 | 168 M | 39.7 | 35.3 | 36.4 | 84.7 | 168 M |
| FSDR Huang et al. (2021) | 44.8 | 41.2 | 43.4 | 95.5 | 171 M | 40.8 | 37.4 | 39.6 | 95.5 | 171 M |
| ATF Li et al. (2023) | 44.8 | 42.3 | 45.6 | 84.7 | 168 M | 41.3 | 36.9 | 39.1 | 84.7 | 168 M |
| SHADE Zhao et al. (2022) | 46.7 | **43.6** | 45.5 | 84.7 | 168 M | 40.1 | 37.8 | 39.2 | 84.7 | 168 M |
| NPDG P-rate=0% (DSP only) | $44.2_{\pm 0.3}$ | $41.0_{\pm 0.5}$ | $43.2_{\pm 0.6}$ | 84.7 | 168 M | $39.1_{\pm 0.3}$ | $37.1_{\pm 0.2}$ | $38.6_{\pm 0.3}$ | 84.7 | 168 M |
| NPDG P-rate=10% | $46.1_{\pm 0.3}$ | $43.3_{\pm 0.2}$ | $45.1_{\pm 0.3}$ | 78.1 | 151 M | $\mathbf{41.4}_{\pm 0.4}$ | $\mathbf{39.1}_{\pm 0.2}$ | $\mathbf{41.1}_{\pm 0.5}$ | 79.5 | 153 M |
| NPDG P-rate=20% | $\mathbf{46.9}_{\pm 0.5}$ | $43.5_{\pm 0.3}$ | $\mathbf{46.1}_{\pm 0.4}$ | 67.7 | 133 M | $41.0_{\pm 0.3}$ | $\mathbf{39.1}_{\pm 0.5}$ | $39.8_{\pm 0.2}$ | 69.6 | 135 M |
| NPDG P-rate=30% | $45.8_{\pm 0.2}$ | $42.9_{\pm 0.2}$ | $45.8_{\pm 0.4}$ | **55.9** | **110 M** | $40.3_{\pm 0.2}$ | $38.6_{\pm 0.1}$ | $39.2_{\pm 0.4}$ | **56.6** | **112 M** |

**Table 2** Domain generalization performance of the ***real-2-synthetic* and *cross-real* tasks.** G,S,C,B,D denote *GTA5*, *SYNTHIA*, *Cityscapes*, *BDD100k* and *Mapillary*, respectively. P-rate denotes the target pruning rate for filters, and GFLOPS denotes the number of flops $\times 10^9$. The best and second-best results are marked with Bold and Underline, respectively

| Method | mIoU | | | | | Mean GFLOPs | Mean #Param. |
|---|---|---|---|---|---|---|---|
| | **real-2-syn** | | **cross-real** | | | | |
| | C → *G* | C → *S* | C → *B* | B → *M* | M → *C* | | |
| Baseline Chen et al. (2018) | 40.43 | 22.81 | 48.01 | 46.33 | 47.61 | 84.7 | 168 M |
| IBN-Net Pan et al. (2018) | 45.06 | 26.14 | 48.56 | 48.16 | 51.22 | 84.8 | 168 M |
| SW Pan et al. (2019) | 44.87 | 26.10 | 48.49 | 47.30 | 50.35 | 84.7 | 168 M |
| DRPC Yue et al. (2019) | 43.55 | 25.91 | 49.44 | 49.06 | 51.33 | 110.4 | 178 M |
| RobustNet Choi et al. (2021) | 45.00 | 26.20 | 50.73 | 48.35 | 50.22 | 84.7 | 168 M |
| GTR Peng et al. (2021) | 43.80 | 24.11 | 47.57 | 48.11 | 50.51 | 84.7 | 168 M |
| FSDR Huang et al. (2021) | 44.50 | 26.61 | 49.02 | 47.69 | 48.22 | 95.5 | 171 M |
| SHADE Zhao et al. (2022) | **48.61** | 27.62 | 50.95 | 50.32 | **52.49** | 84.7 | 168 M |
| NPDG P-rate=0% (DSP only) | $44.26_{\pm 0.3}$ | $25.91_{\pm 0.2}$ | $48.38_{\pm 0.3}$ | $47.11_{\pm 0.3}$ | $48.02_{\pm 0.2}$ | 84.7 | 168 M |
| NPDG P-rate=10% | $45.77_{\pm 0.3}$ | $26.33_{\pm 0.5}$ | $50.68_{\pm 0.3}$ | $49.11_{\pm 0.5}$ | $50.74_{\pm 0.4}$ | 79.1 | 152 M |
| NPDG P-rate=20% | $47.18_{\pm 0.5}$ | $27.78_{\pm 0.3}$ | $52.00_{\pm 0.3}$ | $50.24_{\pm 0.3}$ | $51.86_{\pm 0.6}$ | 68.8 | 134 M |
| NPDG P-rate=30% | $46.74_{\pm 0.6}$ | $\mathbf{27.91}_{\pm 0.4}$ | $\mathbf{52.35}_{\pm 0.3}$ | $\mathbf{50.43}_{\pm 0.5}$ | $51.50_{\pm 0.6}$ | **54.1** | **110 M** |

VGG-16-based DeepLab-v3+ Chen et al. (2018), or MiT-B5-based Segformer (Xie et al., 2021) as the backbone of segmentor, respectively. To reduce the memory footprint, we resize the original source-domain image to $1,280 \times 720$ and random crop $960 \times 480$ as the input. For the CNN-based segmentor, we use SGD (Bottou, 2012) with a momentum of 0.9 and a weight decay of $5e$-4 as the optimizer. The initial learning rates for SGD is set to $2.5e$-4 and is decayed by a poly policy, where the initial learning rate is multiplied by $(1 - \frac{iter}{max\_iter})^{power}$ with $power = 0.9$. We train the network for a total of $100k$ iterations, with the first $5k$ as the warm-up stage. For the Transformer-based segmentor, we employ AdamW as the optimizer with the learning rate $6e$-5 for the encoder and $6e$-4 for the decoder, and weight decay 0.01. Linear learning rate warmup by $1.5k$ iterations is first adopted, and then the learning rate linearly decays. All models are trained with a batch size of 2 for 40K iterations. In our best model, we set hyper-parameters $\lambda = 0.2$ and pruning threshold $t = 0.1$, respectively.

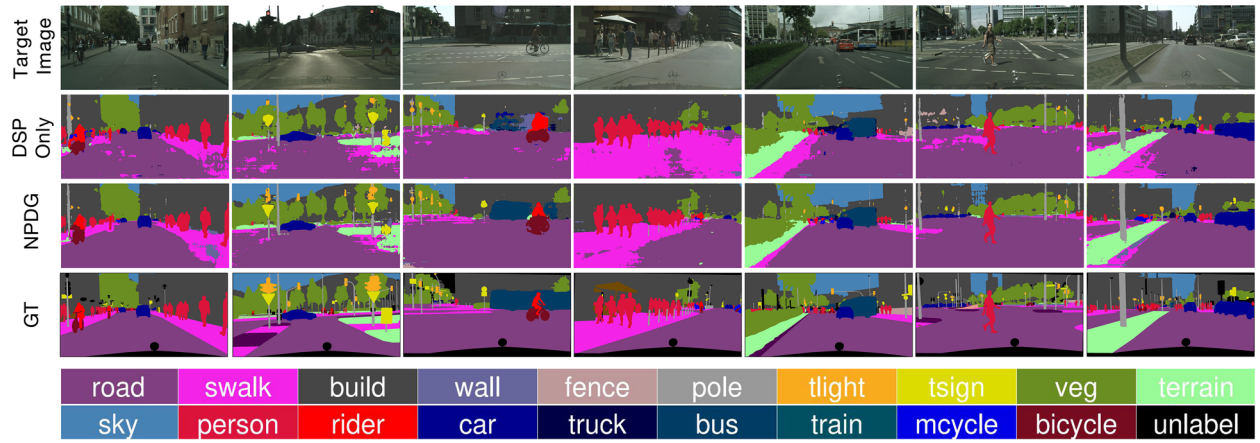


**Fig. 4** Visualization comparisons of using DSP only and the full NPDG

**Table 3** Domain generalization performance using transformer-based backbone. G,S,C,B,D denote *GTA5*, *SYNTHIA*, *Cityscapes*, *BDD100k* and *Mapillary*, respectively. P-rate denotes the target pruning rate for filters, and GFLOPS denotes the number of flops $\times 10^9$. The best and second-best results are marked with Bold and Underline, respectively. The segmentation model is chosen as transformer-based MiT B5 Xie et al. (2021)

| Method | mIoU | | | | | | | Mean GFLOPs | Mean #Param. |
|---|---|---|---|---|---|---|---|---|---|
| | syn-2-real | | | real-2-syn | | cross-real | | | |
| | G → *C* | G → *B* | G → *M* | C → *G* | C → *S* | C → *B* | C → *M* | | |
| Basel. Chen et al. (2018) | 45.60 | 44.86 | 47.72 | 48.36 | 25.28 | 46.66 | 55.54 | 11.9 | 84.7M |
| MixStyle Zhou et al. (2020) | 43.29 | 44.23 | 46.51 | 50.95 | 26.33 | 48.39 | 57.12 | 11.9 | 84.7M |
| CrossNorm Tang et al. (2021) | 46.41 | 44.69 | 50.21 | 48.91 | 25.13 | 47.91 | 58.33 | 11.9 | 84.7M |
| AdvStyle Zhong et al. (2022) | 46.56 | 45.10 | 48.35 | 50.41 | 28.07 | 49.28 | 59.09 | 11.9 | 84.7M |
| SHADE Zhao et al. (2023) | **53.27** | <u>48.19</u> | **54.99** | <u>52.80</u> | **30.27** | **52.77** | 59.82 | 11.9 | 84.7M |
| NPDG P-rate=0% (DSP only) | $47.60_{\pm 0.2}$ | $45.24_{\pm 0.4}$ | $48.61_{\pm 0.5}$ | $50.69_{\pm 0.2}$ | $28.79_{\pm 0.5}$ | $47.75_{\pm 0.1}$ | $57.11_{\pm 0.2}$ | 12.0 | 84.7M |
| NPDG P-rate=10% | $48.33_{\pm 0.3}$ | $47.11_{\pm 0.3}$ | $49.65_{\pm 0.6}$ | $52.03_{\pm 0.3}$ | $28.99_{\pm 0.2}$ | $50.01_{\pm 0.2}$ | $59.89_{\pm 0.4}$ | 11.2 | 78.3M |
| NPDG P-rate=20% | <u>$49.61_{\pm 0.4}$</u> | **$48.39_{\pm 0.5}$** | $51.83_{\pm 0.3}$ | **$53.11_{\pm 0.4}$** | <u>$29.87_{\pm 0.5}$</u> | <u>$52.38_{\pm 0.3}$</u> | **$60.36_{\pm 0.5}$** | <u>10.0</u> | <u>71.1M</u> |
| NPDG P-rate=30% | $49.34_{\pm 0.3}$ | $47.82_{\pm 0.5}$ | <u>$52.06_{\pm 0.4}$</u> | $52.74_{\pm 0.2}$ | $29.61_{\pm 0.5}$ | $51.68_{\pm 0.7}$ | <u>$60.30_{\pm 0.5}$</u> | **8.9** | **62.2M** |

## 4.3 Comparative Studies

### 4.3.1 Compared with DG-Based Methods

We present the generalization results on tasks GTA5 → {*Cityscapes* (*C*), *BDD100k* (*B*) and *Mapillary* (*M*)} and SYNTHIA → {*Cityscapes* (*C*), *BDD100k* (*B*) and *Mapillary* (*M*)} in Table 1, with comparisons to the state-of-the-art DG methods (Pan et al., 2019; Yue et al., 2019; Choi et al., 2021; Peng et al., 2021; Huang et al., 2021; Li et al., 2023; Zhao et al., 2022). Besides the commonly used comparison metric, i.e., mIoU, we also provide the *GFLOPS* and parameter size to evaluate the computation and memory cost of the enrolled methods. First, as can be observed in Table 1, adopting DSP alone can largely improve the baseline method, bringing at least +9% in terms of mIoU. Based on DSP, using the domain-sensitive filter pruning can further improve the generalization ability (e.g., +2.7%, +2.5%, +2.9% in the three target domains when pruning 20% of the filters). These results demonstrate the effectiveness of the proposed DSP and NPDG. Compared to the state-of-the-art method SHADE Zhao et al. (2022), NPDG outperforms it in 5 situations out of 6 experiments with a lighter-weight structure and fewer computation costs.

We also give more comprehensive experimental results on real-to-synthetic and cross-real DG tasks. Seen from tabulated results 2, it becomes evident that NPDG effectively boosts the model's generalization capabilities while concurrently minimizing its overall size in the context of the domain generalization task-from the real-to-synthetic domain, as well as the cross-real domain. Notably, NPDG can steadily outperform the baseline by 2–5% in terms of mIoU, and outperforms current state-of-the-art techniques SHADE Zhao et al. (2023) on $C \rightarrow S$, $C \rightarrow B$ and $B \rightarrow M$ tasks while using fewer computation costs. However, NPDG exhibits inferior performance compared to SHADE on the $C \rightarrow G$

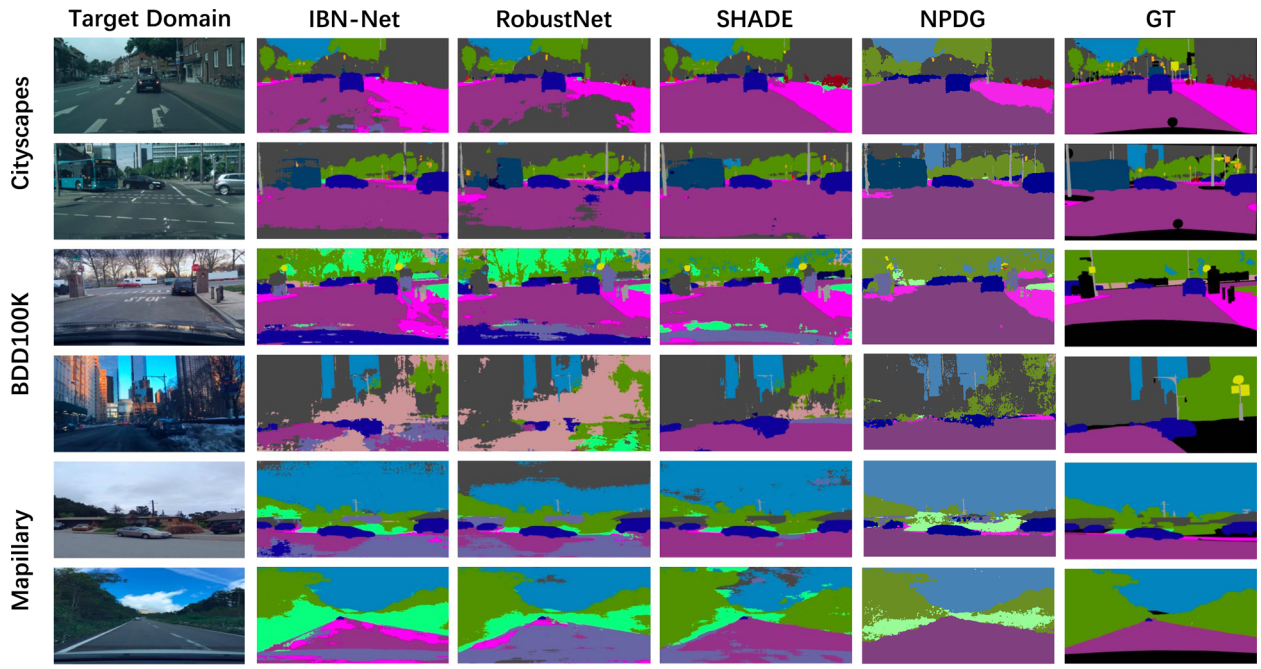


**Fig. 5** DGSS performance of CNN-based methods on different benchmarks, including Cityscapes, BDD100K and Mapillary

and $M \rightarrow C$ tasks (47.18% versus 48.61%, and 51.86% versus 52.49%, respectively). We attribute this discrepancy to the significant style gap between these domains, which is relatively large. Consequently, more advanced style transfer methods would be better suited to address such scenarios effectively. The visualization comparisons can be found in Fig. 5.

When utilizing the transformer as the segmentation backbone, NPDG surpasses the baseline model on all target datasets by at least 4%, achieving state-of-the-art results in 3 out of 7 DG tasks with a lighter-weight structure and reduced computation costs, as shown in Table 3. Since transformers inherently offer greater generalizability and effectiveness for computer vision tasks, NPDG ultimately achieves an average improvement of 5% over CNN-based backbones. These findings illustrate that NPDG is applicable to transformer-based structures, enhancing the generalization ability of segmentation models. We give more visualization samples using different pruning rates (10%, 20% and 50%, respectively) on a transformer, which can be found in Fig. 6. As can be observed, within a certain pruning rate range, NPDG has improved the generalization of the semantic segmentation model compared with the baseline backbone DAFormer. However, if the pruning rate continues to increase, the fine-grained structure and small objects in the output will disappear, affecting the final segmentation result. The results of these visualizations are consistent with the results of our previous quantitative analysis: An excessively high pruning rate (e.g., 90% in the classification task) in segmentation tasks could lead to edge blurring, adversely affecting the overall segmentation performance. Our experiments consistently pointed to an optimal pruning ratio falling within the range of around 20% (Table 2).

#### 4.3.2 Robustness

We provide standard deviations to the presented experiments to evaluate the robustness of NPDG. Because each pruning iteration may result in slightly different model structures, minor fluctuations in these metrics may occur. However, we found that these variances are not substantial. In comparison experiments, these values hover at approximately 0.3, with a minimum value of 0.1 and a maximum value of 0.6. In ablation experiments, the standard deviations are at approximately 0.3, whereas using random only yields the maximum value of 0.6 since such a strategy introduces much uncertainty in generating stylized samples. Furthermore, we observed that employing a higher pruning rate results in greater deviations, typically around +0.15, as a consequence of generating a wider array of variant model structures. By comparing to the DSP-only model, we can infer that the notable enhancements over previous methodologies primarily stem from domain-sensitive filter pruning. Conclusively, our approach has demonstrated efficacy and robustness across diverse domain generalization scenarios, including *syn-to-real*, *real-to-syn*, and *cross-real* settings.

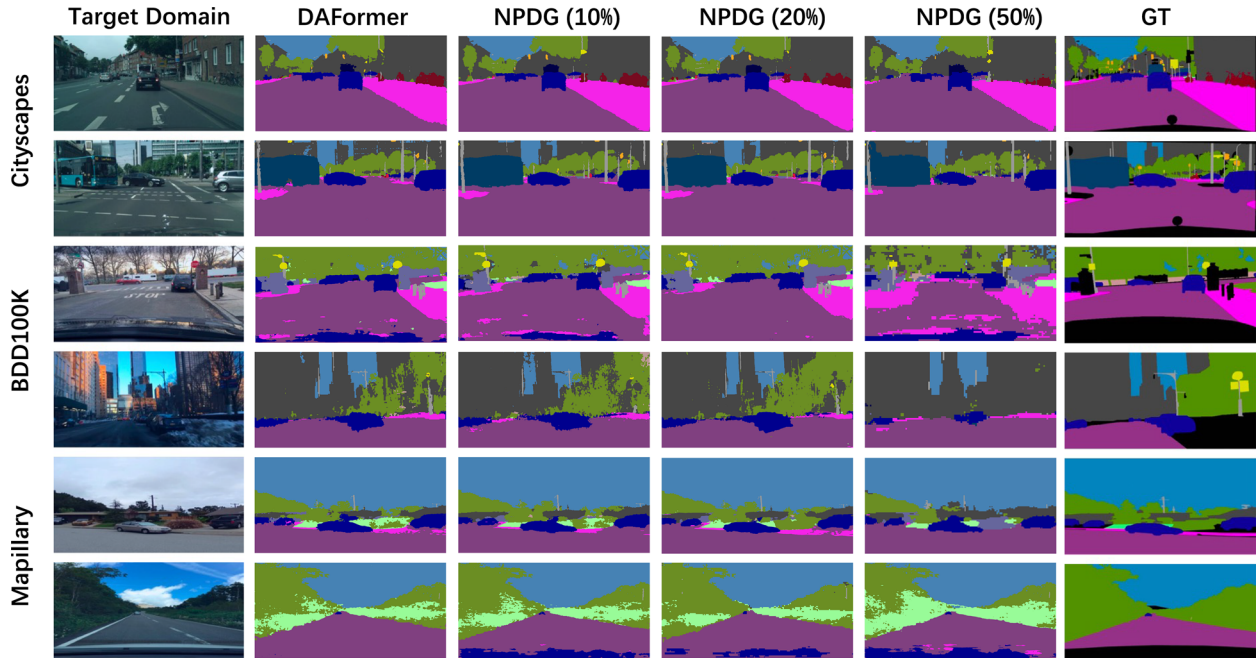


**Fig. 6** Visualization results on MiT-5B using different pruning rates (10%, 20% and 50%, respectively)

#### 4.3.3 Efficiency

Compared with the existing DG methods, NPDG can achieve state-of-the-art segmentation accuracy using a lighter-weight model, saving over 17 GFLOPS and 35$M$ parameter amount. However, further increasing the pruning rate (over 30%) would hurt the generalization performance, which contrasts with the common performance on the classification tasks. We own the reason for the semantic segmentation task itself. Because segmentation is a finer-grained task compared to the classification, pruning too many filters (even some of them are potentially domain sensitive) would drop the segmentation accuracy on the semantic boundaries.

### 4.4 Compared with Network Pruning (NP) Methods

To the best of our knowledge, few NP methods have been conducted on the cross-domain segmentation task. We thus select some popular NP methods that have been proven effective for the vanilla classification (He et al., 2018; Liu et al., 2017; Zhuang et al., 2020) and segmentation tasks (He et al., 2021). We evaluate the pruning performance on both ResNet-101 and VGG-16. The performance comparisons are reported in Table 4. Since these methods do not consider the domain shift during filter or weight pruning, it is unsurprising that they cause the performance drop in new domains compared to the unpruned baseline model. Among these methods, SFP He et al. (2018) achieves the lowest GFLOPS and memory cost. This is because SFP prunes each layer with an equal pruning rate in an explicit manner. Network Slimming based methods (Liu et al., 2017; Zhuang et al., 2020), including ours, are designed to prune the filters in an implicit training process. We observed that in such an implicit scheme, a large part of pruned filters are in the shallow layers. Since the deeper layers usually contain more parameters, the explicit manner would benefit more when the goal is to prune a fixed number of filters.

The visualization result of DSP and NPDG can be referred to in Fig. 4. We also provide some segmentation results of hard examples from BDD100K in Fig. 13, including the dark environments and the adverse weather.

### 4.5 Ablation Studies

The core components of NPDG consist of the DSP module, the unary and binary filter sensitivity. To assess the importance of these components, we conducted an ablation study on them. The results are reported in Table 5. Generally, all of the proposed modules are beneficial to the baseline for better generalization performance. Generating new variant domains brings a huge performance improvement, within which the domain sampled by DSP outperforms the random sampling (RS) strategy ($\varepsilon \sim \mathcal{N}(0, I)$). Based on the domain variance generator, using unary and binary filter sensitivity brings an improvement of 0.7% and 1.3% to DSP in terms of mIoU. Combining both sensitivities would yield the highest mIoU of 47.2, indicating the complementary function of the two filter sensitivities. Combining the RS with unary and binary filter sensitivity only brings a slight improvement (+1.0%) for RS. This indicates that DSP surpasses

**Table 4** Domain generalization performance on $G \rightarrow C$ compared with network pruning (NP) methods using VGG-16 and ResNet-101. P-rate denotes the target filters pruning rate, and GFLOPS denotes the number of flops $\times 10^9$. † indicates our own implementation

| Method | Struc. | P-rate | mIoU | GFLOPs | #Param. | Struc. | P-rate | mIoU | GFLOPs | #Param. |
|---|---|---|---|---|---|---|---|---|---|---|
| Basel. (DSP only) | VGG | 0% | $36.9_{\pm 0.3}$ | 50.1 | 101 M | ResNet | 0% | $44.2_{\pm 0.4}$ | 84.7 | 168 M |
| NS Liu et al. (2017) | VGG | 10% | $36.2_{\pm 0.2}$ | 47.0 | 94 M | ResNet | 10% | $42.2_{\pm 0.2}$ | 80.2 | 153 M |
| | | 20% | $35.5_{\pm 0.4}$ | 42.7 | 84 M | | 20% | $41.5_{\pm 0.3}$ | 71.6 | 140 M |
| | | 50% | $32.7_{\pm 0.5}$ | 27.8 | 55 M | | 50% | $37.7_{\pm 0.4}$ | 48.8 | 92 M |
| SFP He et al. (2018) | VGG | 10% | $35.8_{\pm 0.2}$ | 45.2 | 91 M | ResNet | 10% | $41.4_{\pm 0.2}$ | 75.5 | 152 M |
| | | 20% | $35.4_{\pm 0.2}$ | 40.1 | 81 M | | 20% | $39.5_{\pm 0.4}$ | 65.4 | 137 M |
| | | 50% | $31.1_{\pm 0.5}$ | **25.6** | **50 M** | | 50% | $33.1_{\pm 0.6}$ | **43.1** | **85 M** |
| PR Zhuang et al. (2020) | VGG | 10% | $36.5_{\pm 0.4}$ | 46.2 | 93 M | ResNet | 10% | $42.6_{\pm 0.2}$ | 77.7 | 154 M |
| | | 20% | $36.2_{\pm 0.2}$ | 41.1 | 83 M | | 20% | $42.2_{\pm 0.2}$ | 69.2 | 141 M |
| | | 50% | $34.4_{\pm 0.6}$ | 26.8 | 54 M | | 50% | $39.6_{\pm 0.5}$ | 45.9 | 90 M |
| CAP† He et al. (2021) | VGG | 10% | $36.7_{\pm 0.4}$ | 45.4 | 91 M | ResNet | 10% | $43.2_{\pm 0.4}$ | 76.4 | 152 M |
| | | 20% | $36.6_{\pm 0.3}$ | 41.0 | 82 M | | 20% | $42.5_{\pm 0.1}$ | 69.5 | 141 M |
| | | 50% | $35.2_{\pm 0.3}$ | 26.4 | 53 M | | 50% | $40.5_{\pm 0.2}$ | 44.9 | 89 M |
| NPDG (ours) | VGG | 10% | $37.8_{\pm 0.2}$ ↑ | 45.9 | 92 M | ResNet | 10% | $46.1_{\pm 0.2}$ ↑ | 75.9 | 151 M |
| | | 20% | **38.7**$_{\pm 0.2}$ ↑ | 41.3 | 83 M | | 20% | **47.0**$_{\pm 0.2}$ ↑ | 68.0 | 133 M |
| | | 50% | $34.5_{\pm 0.5}$ | 26.5 | 52 M | | 50% | $42.3_{\pm 0.3}$ | 44.6 | 89 M |

The indicates IoU or lowest memory cost is marked with Bold

RS for the pruning-based DG. We also give the visualization results when using partial or full pruning strategy on a transformer-based segmentor (MiT-5B as backbone), as shown in Fig. 7. The baseline method utilizes the vanilla non-TopK pruning strategy. "NPDG w/o u_NP" indicates that we consider binary attention head sensitivity only, while "NPDG w/o b_NP" means that we merely use unary sensitivity. In "NPDG full", we employ both unary and binary sensitivity. Firstly, it can be observed that for DGSS tasks, both the unary policy and the binary policy alone outperform the direct removal of non-TopK filters. Secondly, the segmentation performance of the binary strategy alone is slightly better than that of the unary strategy alone. This indicates that for segmentation tasks, the binary strategy is more effective in removing domain-sensitive information, suggesting that domain change information is primarily present in style factors. Finally, the combination of the unary and binary strategies achieves the best results, demonstrating the complementarity of the two approaches.

## 4.6 Hyper-Parameter Studies

In NPDG, the key hyper-parameters primarily consist of the ratio $\lambda$ in Eq. 13, the pruning rate $r$, and the pruning threshold $t$.

*(1) Ratio between unary and binary weights* To determine an appropriate ratio between unary and binary weights, we conducted a grid search ranging from 0.2 to 0.8. Extreme values such as 0 and 1 would result in an overlap of the ablation study. The results indicate that both ResNet101-based and MiT-B5-based models achieve their optimal performance when $\lambda = 0.4$. This demonstrates that both unary and binary sensitivity pruning contribute, with binary pruning exhibiting a slightly more pronounced effect. The detailed results can be seen in table 6.

**Table 5** Ablation study on $G \rightarrow C$. The pruning rate is set as 20%

| Random | DSP | Unary | Binary | mIoU | #Paramter |
|---|---|---|---|---|---|
| | | | | $34.7_{\pm 0.2}$ | 168 M |
| √ | | | | $42.8_{\pm 0.6}$ | 168 M |
| | √ | | | $44.2_{\pm 0.2}$ | 168 M |
| | √ | √ | | $44.9_{\pm 0.3}$ | 135 M |
| | √ | | √ | $45.5_{\pm 0.2}$ | **127 M** |
| √ | | √ | √ | $43.8_{\pm 0.5}$ | 135 M |
| | √ | √ | √ | **47.2**$_{\pm 0.3}$ | 133 M |

The indicates IoU or lowest memory cost is marked with Bold

*(2) Pruning rate r* Similar to many existing network pruning methodologies, the pruning rate is inherently a flexible parameter. There is no universally established standard for its values that guarantees optimal efficiency and performance across diverse domains. As demonstrated in the baseline method, i.e. network slimming (Liu et al., 2017), the pruning rates are chosen spanning from 10 to 90% to comprehensively explore the network's performance landscape. Consequently, we also conducted a grid-like search strategy in our experiments, setting the rate from 10 to 50%. Our extensive experimentation has revealed nuanced insights,

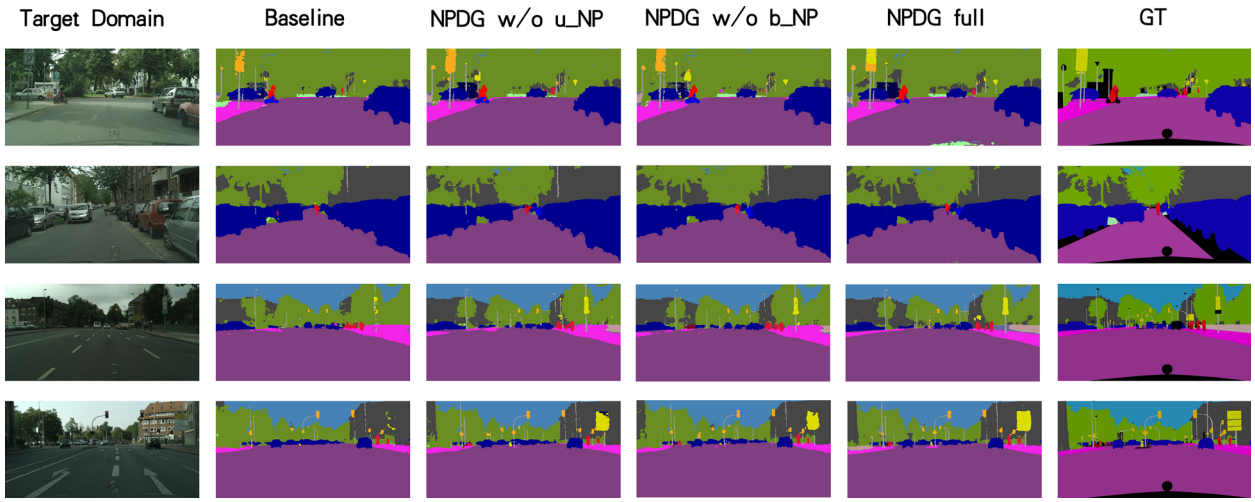


**Fig. 7** Qualitative results and analysis about the pruning strategy for transformer-based structure. The baseline method utilizes the vanilla non-TopK pruning strategy. "NPDG w/o u_NP" indicates that we consider binary attention head sensitivity only, while "NPDG w/o b_NP" means that we merely use unary sensitivity. In "NPDG full", we employ both unary and binary sensitivity. Firstly, it can be observed that for DGSS tasks, both the unary policy and the binary policy alone outperform the direct removal of non-TopK filters. Secondly, the segmentation performance of the binary strategy alone is slightly better than that of the unary strategy alone. This indicates that for segmentation tasks, the binary strategy is more effective in removing domain-sensitive information, suggesting that domain change information is primarily present in style factors. Finally, the combination of the unary and binary strategies achieves the best results, demonstrating the complementarity of the two approaches

particularly in the context of segmentation tasks compared to coarse-grained classification tasks. We observed that an excessively high pruning rate (e.g., 90% in the classification task) in segmentation tasks could lead to edge blurring, adversely affecting the overall segmentation performance. Our experiments consistently pointed to an optimal pruning ratio falling within the range of 20% to 40%, with approximately 30% yielding the best generalization outcomes.

Employing a heuristic approach to use a 30% pruning rate has proven effective in most cases. Nevertheless, finding an exact trade-off between pruning rate and mIoU is a more robust and practical way, with the employment of a validation set. Recognizing the challenge in constructing an effective validation set, posed by unseen target domains in DG tasks, we have integrated an off-the-shelf style generator (DSP) into our framework, as illustrated in Fig. 8. Specifically, we allocated 10% of the source dataset for validation purposes. Styles not encountered during training were sampled within the style space (in terms of new sampled $\mu$ and $\sigma$) to stylize this 10% validation subset, thereby constituting our final validation set.

Throughout our experiments, spanning pruning rates from 0 to 90%, we computed the mean Intersection over Union (mIoU) across various tasks. The corresponding performance metrics are presented in Figs. 8, 9, 10, 11. Interestingly, our findings reaffirm our earlier conclusion drawn from the heuristic strategy, indicating that the optimal pruning rate typically lies around 20% for both CNN-based and Transformer-based backbones. However, this revised approach aligns more closely with conventional machine learning practices by leveraging a dedicated validation set for hyperparameter tuning. Consequently, in practical appli-

**Table 6** Hyper-parameter study of $\lambda$ on task $G \rightarrow C$

| $\lambda$ | mIoU (ResNet-101) | mIoU (MiT-B5) |
|---|---|---|
| 0.2 | $46.74_{\pm 0.4}$ | $51.10_{\pm 0.2}$ |
| 0.4 | $\mathbf{47.17}_{\pm 0.3}$ | $\mathbf{52.35}_{\pm 0.4}$ |
| 0.6 | $46.12_{\pm 0.3}$ | $51.65_{\pm 0.2}$ |
| 0.8 | $45.29_{\pm 0.5}$ | $49.47_{\pm 0.5}$ |

The indicates IoU or lowest memory cost is marked with Bold

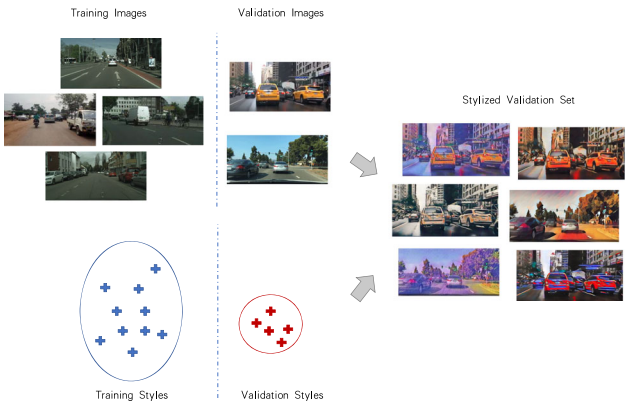


**Fig. 8** Validation set construction using the off-the-shelf DSP

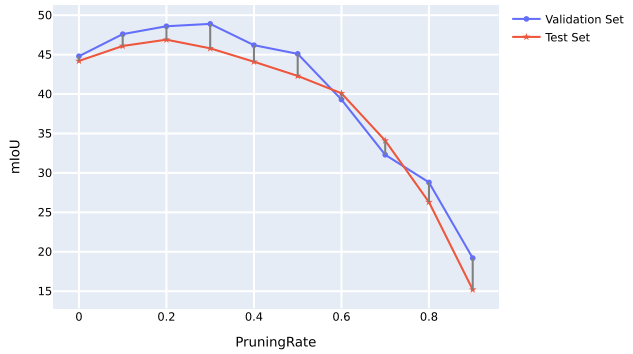


**Fig. 9** G → C (CNN)

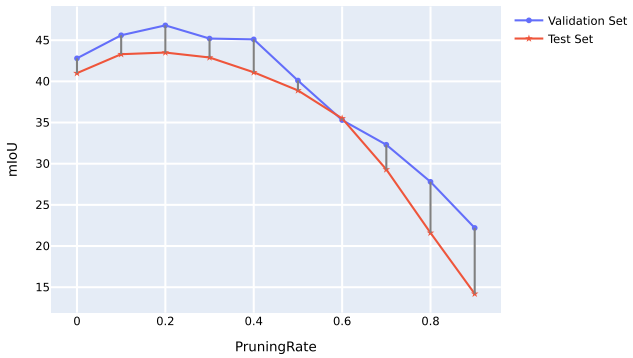


**Fig. 10** G → B (CNN)

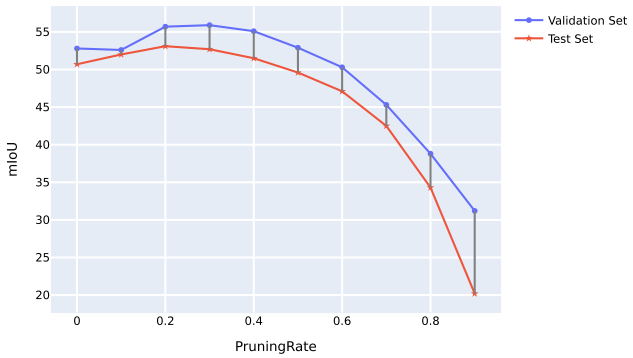


**Fig. 11** C → G (Transformer)

cations, we recommend adopting a constructed validation first. If unavailable, using a pruning ratio of around 30% is fine for most cases.

*(3) Pruning threshold t* A pruning threshold $t$ is required to be set in advance. Generally, we found the pruning threshold t is not a very sensitive hyper-parameter and can be valued in a certain range. Given sufficient training time, certain scaling factors gradually approach zero. In other words, regardless of whether the pruning threshold is set to 0.1 or 0.01, as long as the sparse training iterations are sufficient, a sufficient number of filters satisfying the requirements will be identified. Once the pruning rate is reached, the sparse training will be ended. This inherent adaptability allows our approach

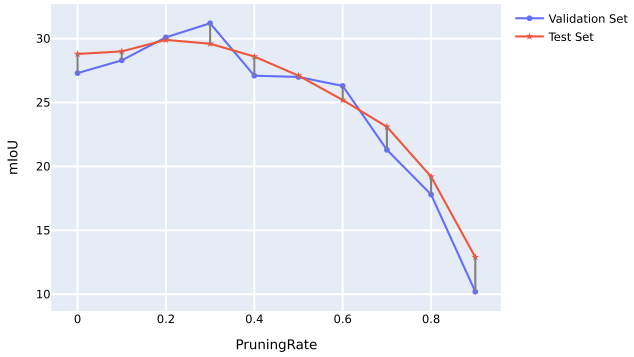


**Fig. 12** C → S (Transformer)

to effectively handle variations in pruning thresholds and ensures the identification of filters that align with the desired requirements (Fig. 15).

It should be noted that our claim regarding the pruning threshold not being very sensitive does not imply that the parameter can be arbitrarily selected. A clear principle is that the threshold should ensure that the initial values of most scaling factors are greater than the threshold. This criterion is not difficult to meet. As illustrated in Fig. 14, threshold values ranging from 0.001 to 0.2 are acceptable according to this principle. Generally, we set the pruning threshold $t = 0.1$, which is consistent with Liu et al. (2017).

## 4.7 Analysis and Discussion

*Performance on the source dataset* We reported the performance of the pruned model (ResNet-101 as the backbone) on GTA5 in Table 7, to evaluate NPDG and other methods on the source domain. Compared to the unpruned model, NPDG slightly affects the mIoU by −0.9 when pruning 30% filters, which remains comparable performance to other SOTA pruning methods.

*Statistics of scaling factors* We visualize the statistics of scaling factors before and after the sparsity training. As can be observed in Fig. 14a, the statistics of the scaling factors in the pre-trained model are close to a Gaussian distribution. After sparsity training, the factors on the domain-sensitive filters are pushed to zeros. Filters corresponding to tiny scaling factors (e.g. $\leq 0.1$) are then pruned.

*Pruning state of shallow/deep layers* We visualize the final pruning state of each bottleneck (Btnk) in ResNet-101. As can be observed in Fig. 14b, as the sparsity training proceeds, the pruning rate of shallow layers (e.g., in Btnk 1 and 2) is higher than the target rate while the deep layers (e.g., in Btnk 3 and 4) are gotten less pruned. This is because the shallow layers capture the low-level information, which are more sensitive to the domain shifts e.g. style variance for segmentation task, and thus punished more by $w$.

**Table 7** Performance on the **source domain** (GTA5)

| Unpruned | NPDG | NS Liu et al. (2017) | CAP He et al. (2021) |
|---|---|---|---|
| $77.8_{\pm 0.5}$ | $76.9_{\pm 0.5}$ (0.9↓) | $75.3_{\pm 0.6}$ (2.5↓) | $76.7_{\pm 0.5}$ (1.1↓) |

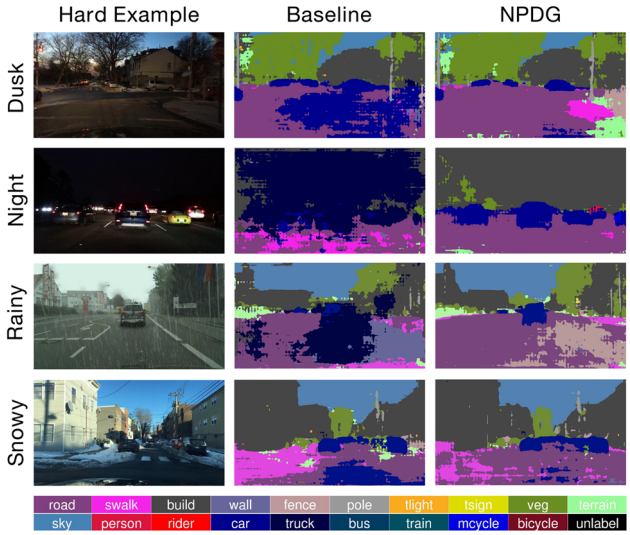


**Fig. 13** Comparisons of baseline and NPDG on hard examples, including images of dusk, night and adverse weathers

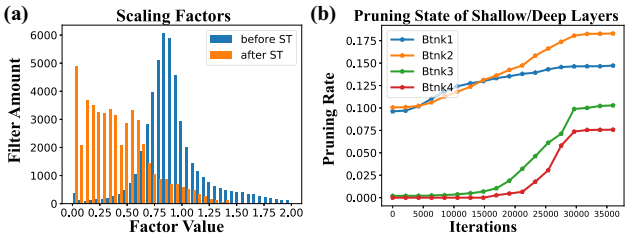


**Fig. 14** **a** Statistics for the scaling factors before/after sparse training. **b** Visualization of the pruning stage of each bottleneck (Btnk) in ResNet-101 as the sparse training proceeding

*Computational Cost during training & inference* (1) Training Computational Cost: The training of NPDG involves a two-stage process. Firstly, we train DSP. Once DSP is trained, it remains fixed and is employed in the subsequent Network Pruning & Domain Diversifying loop without retraining. As DSP is pre-trained and static during the Network Pruning & Domain Diversifying loop, the overall computational cost is more manageable than expected. We conducted thorough analyses to quantify the impact of each component on the model's training speed. Specifically, each iteration during training incurs a cost of 892ms. The incorporation of DSP accounts for 21 ms, while the computation of unary and binary weights accounts for 5 ms. (2) Inference Computational Cost: During inference, DSP is no longer required, and the segmentation model operates in its pruned form, resulting in a notably faster inference speed.

*Comparing with Non-Top-K neuron pruning* We conducted a referenced study on Li et al. (2024), wherein the authors demonstrated that Top-K neurons exhibit a stronger inclination towards shape characteristics compared to their counterparts (i.e., Non-top-K neurons). Their findings suggested that pruning non-Top-K neurons could mitigate model bias towards texture, which is also helpful in our domain generalization setting. We posit that non-Top-K pruning aligns with a "training free" post-hoc pruning strategy, while ours resembles a more "learnable" way to prune the filters. From this perspective, our work and Li et al. (2024) are not mutually exclusive but can be complementary since the pruning is done at different stages. Moreover, a comparative analysis of the pruned neurons from Li et al. (2024) and our study reveals an overlap of approximately 70%, indicating that both strategies achieve similar effectiveness in pruning the style-(domain-) sensitive filters.

Although achieving 70% overlapped pruned neurons, the experimental results of Li et al. (2024) in segmentation tasks demonstrate worse Intersection over Union (IoU) performance compared to ours. For instance, on the G → C task, the IoU stands at $37.3_{\pm 0.2}$. The possible reason is that neurons have different duties in the two tasks of classification and segmentation, where the training-free post-hoc pruning strategy would potentially hurt the semantic boundaries and object details. We give some visualization results to support our hypothesis. Following Li et al. (2024), we used a texture synthesis approach (Gatys et al., 2016) with ablation of the remaining filters' responses to explore the information contained in them, as shown in Fig. 15. These results further indicate that the adoption of Li et al. (2024) leads to blurred semantic boundaries.

## 5 Conclusions

In this work, we introduce a novel domain generalization approach, termed NPDG, which leverages network pruning techniques. NPDG strategically prunes filters or attention heads that exhibit heightened sensitivity to domain shifts while retaining those deemed domain-invariant. The innovation of NPDG is twofold. Firstly, we propose a tailored pruning policy specifically designed to enhance generalization capabilities. This policy discerns filter sensitivity to domain shifts through both unary and binary analyses. Secondly, we introduce a differentiable style perturbation scheme, which dynamically mimics domain variations, thereby aiding in the

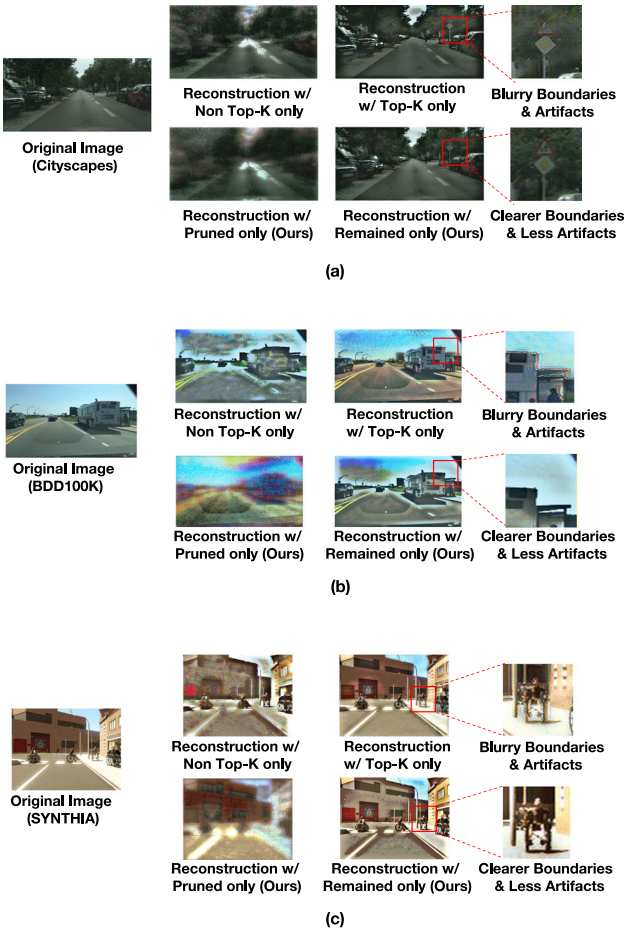


**Fig. 15** Visualizing the retained and pruned neurons by optimizing input images to match their activations. Subfigures (**a**), (**b**), and (**c**) present the reconstruction results for samples from the Cityscapes, BDD100K, and SYNTHIA datasets, respectively. It is evident that NPDG retains filters that depict fine-grained object structures more effectively compared to using only the TopK filters. Since semantic segmentation is a fine-grained scene understanding task that is very sensitive to object edges, NPDG performs better for out-of-distribution generalization of semantic segmentation

identification of potentially sensitive filters. To the best of our knowledge, we are among the pioneering efforts in addressing domain generalization via network pruning methods. Extensive experiments on both CNN- and Transformer-based architectures demonstrates that NPDG achieves state-of-the-art performance in segmentation generalization, even with lighter-weight models.

*Limitations & Future work* For most segmentation DG benchmarks such as "day-to-night" and "summer-to-winter", style difference is one of the vital causes for the domain shifts. While it is true that other factors also contribute to these shifts, it is practically impossible to identify all of them in the absence of target data. As a future direction, we aim to address the general domain shift problem comprehensively. Nevertheless, the paradigm of NPDG could be a positive inspiration for tackling other factors.

Currently, we rely on empirical values derived from known datasets to select hyper-parameters for network pruning, which fortunately demonstrate applicability across various domains. However, as we anticipate facing increasingly complex domains in the future, the effectiveness of these parameters may diminish. Therefore, we recognize the necessity of developing a method for determining hyper-parameters in a test-time training manner in future research endeavors.

**Acknowledgements** This work was supported by the National Natural Science Foundation of China ( 62293554, 62206249, U2336212 ), Natural Science Foundation of Zhejiang Province, China ( LZ24F020002 ), Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), and the Fundamental Research Funds for the Central Universities(No. 226-2022-00051).

**Data availability** The datasets that support the findings of this study are publicly available. **GTA5** Richter et al. (2016) is available at https://download.visinf.tu-darmstadt.de/data/from_games. Synthia Ros et al. (2016) can be downloaded from https://synthia-dataset.net/. BDD100K Yu et al. (2020) is available at https://doc.bdd100k.com/download.html. Cityscapes Cordts et al. (2016) can be available at https://www.cityscapes-dataset.com/ after registration. Mapillary Neuhold et al. (2017) is available at https://www.mapillary.com/datasets after registration.

## Declarations

**Competing interests** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

Bottou, Léon. (2012). *Stochastic gradient descent tricks*. In Neural networks: Tricks of the trade.

Cai, J., Zhu, C., Cui, C., Li, H., Wu, T., Zhang, S., & Yang, L. (2021). Generalizing nucleus recognition model in multi-source ki67 immunohistochemistry stained images via domain-specific pruning. In *MICCAI*, pages 277–287.

Cai, R., Li, Z., Wei, P., Qiao, J., Zhang, K., & Hao, Z. (2019). Learning disentangled semantic representation for domain adaptation. In *IJCAI*, pages 2060–2066.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *T-PAMI, 40*(4), 834–848.

Chen, S., Wang, W., & Pan, S. J. (2019). Cooperative pruning in cross-domain deep neural network compression. In *IJCAI*, pages 2102–2108.

Choi, S., Jung, S., Yun, H., Kim, J. T., Kim, S., & Choo, J. (2021). Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, pages 11580–11590.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223.

Dubey, A., Chatterjee, M., & Ahuja, N. (2018). Coreset-based neural network compression. In *ECCV*, pages 454–470.

Fu, Y., Zhang, M., Xu, X., Cao, X., Ma, C., Ji, Y., Zuo, K., & Lu, H. (2021). Partial feature selection and alignment for multi-source domain adaptation. In *CVPR*.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2016). Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423.

Gong, R., Li, W., Chen, Y., Dai, D., & Van Gool, L. (2021). Dlow: Domain flow and applications. *International Journal of Computer Vision, 129*(10), 2865–2888.

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *NeurIPS*, 28.

Hassibi, B., & Stork, D.(1992). Second order derivatives for network pruning: Optimal brain surgeon. In *NeurIPS*, page 164–171.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

He, W., Wu, M., Liang, M., & Lam, S. (2021). Cap: Context-aware pruning for semantic segmentation. In *WACV*, pages 960–969.

He, Y., Ding, Y., Liu, P., Zhu, L., Zhang, H. & Yang, Y. (2020). Learning filter pruning criteria for deep convolutional neural networks acceleration. In *CVPR*, pages 2009–2018.

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, pages 2234–2240.

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Yi (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, pages 4340–4349.

He, Y., & Xiao, L. (2023). Structured pruning for deep convolutional neural networks: A survey. *TPAMI*.

Hoyer, L., Dai, D., & Van Gool, L. (2022). Hrda: Context-aware high-resolution domain-adaptive semantic segmentation. In *ECCV*, pages 372–391.

Huang, J., Guan, D., Xiao, A., & Lu, S. (2021). Fsdr: Frequency space domain randomization for domain generalization. In *CVPR*, pages 6891–6902.

Huang, J., Guan, D., Xiao, A., & Lu, S. (2021). Fsdr: Frequency space domain randomization for domain generalization. In *CVPR*, pages 6891–6902.

Huang, X., & Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456.

LeCun, Y., Denker, J., & Solla, S. (1989). Optimal brain damage. In *NeurIPS*, page 598–605.

Li, D., & Hospedales, T. (2020). Online meta-learning for multi-source and semi-supervised domain adaptation. In *ECCV*, pages 382–403.

Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. M. (2017). Deeper, broader and artier domain generalization. In *ICCV*, pages 5542–5550.

Li, T., Wen, Z., Li, Y., & Lee, T. S. (2024). Emergence of shape bias in convolutional neural networks through activation sparsity. *NeurIPS*.

Li, X., Li, M., Wang, Y., Ren, C.-X., & Guo, X. (2023). Adaptive texture filtering for single-domain generalized segmentation. arXiv preprint arXiv:2303.02943.

Li, Y., Yuan, L., & Vasconcelos, N. (2019). Bidirectional learning for domain adaptation of semantic segmentation. In *CVPR*, pages 6936–6945.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2736–2744.

Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105.

Long, S., Zhou, Q., Ying, C., Ma, L., & Luo, Y. (2023). Rethinking domain generalization: Discriminability and generalizability. arXiv preprint arXiv:2309.16483.

Luo, J.-H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *CVPR*, pages 5058–5066.

Luo, Y., Liu, P., Guan, T., Yu, J., & Yang, Y. (2020). Adversarial style mining for one-shot unsupervised domain adaptation. In *NeurIPS*, pages 20612–20623.

Luo, Y., Liu, P., Zheng, L., Guan, T., Yu, J., & Yang, Y. (2021). Category-level adversarial adaptation for semantic segmentation using purified features. *T-PAMI*.

Luo, Y., Zheng, L., Guan, T., Yu, J., & Yang, Y. (2019). Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *CVPR*, pages 2507–2516.

Muandet, K., Balduzzi, D., & Schölkopf, B. (2019). Domain generalization via invariant feature representation. In *ICML*, pages 10–18.

Neuhold, G., Ollmann, T., Bulo, S. R. & Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, pages 4990–4999.

Nguyen, B., Moëllic, P.-A., & Blayac, S. (2022). Domain generalization on constrained platforms: On the compatibility with pruning techniques. In *Global IoT Summit*.

Nichol, K. (2016). Painter by numbers, wikiart. https://www.kaggle.com/c/painter-by-numbers.

Pan, X., Luo, P., Shi, J., & Tang, X. (2018). Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, pages 464–479.

Pan, X., Zhan, X., Shi, J., Tang, X., & Luo, P. (2019). Switchable whitening for deep representation learning. In *ICCV*, pages 1863–1871.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In *NeurIPS*.

Peng, D., Lei, Y., Liu, L., Zhang, P., & Liu, J. (2021). Global and local texture randomization for synthetic-to-real semantic segmentation. *T-IP, 30*, 6594–6608.

Peng, X., Huang, Z., Sun, X., & Saenko, K. (2019). Domain agnostic learning with disentangled representations. In *ICML*, pages 5102–5112.

Qian, X., & Klabjan, D. (2021). A probabilistic approach to neural network pruning. In *ICML*, pages 8640–8649.

Qiao, F., & Peng, X. (2021). Uncertainty-guided model generalization to unseen domains. In *CVPR*, pages 6790–6800.

Richter, S. R., Vineet, V., Roth, S., & Koltun, V. (2016). Playing for data: Ground truth from computer games. In *ECCV*, pages 102–118.

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., & Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, pages 3234–3243.

Rosenfeld, J. S., Frankle, J., Carbin, M., & Shavit, N. (2021). On the predictability of pruning across scales. In *ICML*, pages 9075–9083.

Sehwag, V., Wang, S., Mittal, P., & Jana, S. (2020). Hydra: Pruning adversarially robust neural networks. In *NeurIPS*, pages 19655–19666.

Sun, X. (2023). Pruning for better domain generalizability. arXiv preprint arXiv:2306.13237.

Tang, Z., Gao, Y., Zhu, Y., Zhang, Z., Li, M., & Metaxas, D. N. (2021). Selfnorm and crossnorm for out-of-distribution robustness. In *ICCV*.

Tian, C. X., Li, H., Xie, X., Liu, Y., & Wang, S. (2022). Neuron coverage-guided domain generalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(1), 1302–1311.

Tjio, G., Liu, P., Zhou, J. T., & Goh, R. S. M. (2021). Adversarial semantic hallucination for domain generalized semantic segmentation. *CoRR*.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176.
Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.
Wang, J., Lan, C., Liu, C., Ouyang, Y., Zeng, W., & Qin, T. (2021). Generalizing to unseen domains: A survey on domain generalization. arXiv preprint arXiv:2103.03097.
Wang, J., & Jiang, J. (2021). Learning across tasks for zero-shot domain adaptation from a single source domain. *T-PAMI*.
Wang, W., Zhong, Z., Wang, W., Chen, X., Ling, C., Wang, B., & Sebe, N. (2023). Dynamically instance-guided adaptation: A backward-free approach for test-time domain adaptive semantic segmentation. In *CVPR*, pages 24090–24099.
Wu, K., Tang, F., Liu, N., Deussen, O., Dong, W., & Lee, T.-Y., et al. (2024). Lighting image/video style transfer methods by iterative channel pruning. In *ICASSP*, pages 3800–3804. IEEE.
Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., & Luo, P. (2021). Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*.
Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., & Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, pages 2636–2645.
Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A., Keutzer, K., & Gong, B. (2019). Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, pages 2100–2110.
Zhang, P., Zhang, B., Zhang, T., Chen, D., Wang, Y., & Wen, F. (2021). Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In *CVPR*, pages 12414–12424.
Zhang, Q., Zhang, J., Liu, W., & Tao, D. (2019). Category anchor-guided unsupervised domain adaptation for semantic segmentation. In *NeurIPS*.
Zhao, S., Yue, X., Zhang, S., Li, B., Zhao, H., Bichen, W., Krishna, R., Gonzalez, J. E., Sangiovanni-Vincentelli, A. L., Seshia, S. A., & Keutzer, K. (2020). A review of single-source deep unsupervised visual domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems, 33*(2), 473–493.
Zhao, Y., Zhong, Z., Yang, F., Luo, Z., Lin, Y., Li, S., & Sebe, N. (2021). Learning to generalize unseen domains via memory-based multi-source meta-learning for person re-identification. In *CVPR*, pages 6277–6286.
Zhao, Y., Zhong, Z., Zhao, N., Sebe, N., & Lee, G. H. (2022). Style-hallucinated dual consistency learning for domain generalized semantic segmentation. In *ECCV*, pages 535–552. Springer.
Zhao, Y., Zhong, Z., Zhao, N., Sebe, N., & Lee, G. H. (2023). Style-hallucinated dual consistency learning: A unified framework for visual domain generalization. *IJCV*.
Zheng, Z., & Yang, Y. (2021). Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. *International Journal of Computer Vision, 129*(4), 1106–1120.
Zhong, Z., Zhao, Y., Lee, G. H., & Sebe, N. (2022). Adversarial style augmentation for domain generalized urban-scene segmentation. *NeurIPS, 35*, 338–350.
Zhou, K., Liu, Z., Qiao, Y., Xiang, T., & Loy, C. C. (2021). Domain generalization: A survey. arXiv preprint arXiv:2103.02503.
Zhou, K., Yang, Y., Qiao, Y., & Xiang, T. (2020). Domain generalization with mixstyle. In International conference on learning representations
Zhuang, T., Zhang, Z., Huang, Y., Zeng, X., Shuang, K., & Li, X. (2020). Neuron-level structured pruning using polarization regularizer. In *NeurIPS*, pages 9865–9877.
Zou, Y., Yang, X., Yu, Z., Kumar, B. V. K. V., & Kautz, J. (2020). Joint disentangling and adaptation for cross-domain person re-identification. In *ECCV*, pages 87–104.