Domain Generalization Guided by Gradient Signal to Noise Ratio of Parameters Supplemental Material

Mateusz Michalkiewicz
University of Queensland

Masoud Faraki
NEC Labs America

Xiang Yu
Amazon *

Manmohan Chandraker
NEC Labs America,
University of California, San Diego

Mahsa Baktashmotlagh
University of Queensland

We provide additional material to supplement our work. Section 1 describes additional experimental results that cover Digits-DG and VLCS benchmarks, as well as applicability of our approach to different backbones. In Section 2 we address two key aspects. Firtly, we demonstrate that by iteratively dropping the most predictive parameters, the model is forced to learn less dominant features. Secondly, we extend the scope of the ablation study to encompass the intermediate dropout masks. We report on the impact of ImageNet pre-training in Section 3 and provide details on baseline implementation in Section 4. Lastly, code listings are appended in Section 5.

1. Additional experiments

Digits-DG. We expand the experimental validation of our method over Digits-DG [29] dataset which covers digit images with various styles, colors and backgrounds coming from MNIST [19], MNIST-M [10], SVHN [22] and SYN [10] datasets. Table 1 shows that our method obtains best performance averaged across all 4 domains.

Table 1: Classification accuracy (%) on the Digits-DG dataset [29]. The bold numbers indicate the best performance averaged across all test domains.

Digits-DG	MNIST	MNIST-M	USPS	SVHN	Synthetic	Avg ↑
Baseline	86.15	74.44	90.07	81.29	94.46	85.28
DSBN [3]	87.01	71.20	91.18	78.23	94.30	84.38
SN [21]	89.28	78.40	88.54	79.12	95.66	86.20
DSON [23]	89.62	79.00	91.63	81.02	95.34	87.32
Ours	96.97	84.38	90.82	80.11	96.21	**89.69**

VLCS. We conclude our classification experiments with VLCS dataset [24] which spans 10 729 images grouped into 5 categories. Each image belongs to one of the following domains: SUN09 [28], LabelMe [25], PASCAL VOC 2007 [8], Caltech-101 [9]. Results presented in Table 2 further demonstrate the generalization capability of our method.

Table 2: Classification accuracy (%) on the VLCS dataset [24]. The bold numbers indicate the best performance averaged across all test domains.

VLCS	Caltech	LabelMe	VOC2007	SUN09	Avg ↑
Baseline	96.25	59.72	70.58	64.51	72.76
JiGen [2]	96.93	60.90	70.62	64.30	73.19
RSC [14]	97.61	61.86	73.93	68.32	75.43
Ours	97.49	65.47	73.82	68.43	**76.30**

DNNs. We show the versatility of our approach by applying it to various backbones. Table 3 shows that the performance gap increases with the depth of the backbones: growing from 0.89% on AlexNet [18] to 2.25% on ResNet50 [13].

Table 3: Classification accuracy (%) on PACS [20] using various backbones. The bold numbers indicate the best performance averaged across all test domains.

PACS		artpaint	cartoon	sketch	photo	Avg ↑
AlexNet	MASF [7]	70.35	72.46	67.33	90.68	75.21
	DMG [4]	64.65	69.88	71.42	87.31	73.32
	RSC [14]	70.93	71.62	71.35	90.23	76.03
	Ours	72.25	73.23	70.69	91.52	**76.92**
ResNet18	MASF [7]	80.29	77.17	71.68	94.99	81.03
	DMG [4]	76.90	80.38	75.21	93.35	81.46
	RSC [14]	80.73	79.22	81.48	94.16	83.90
	Ours	83.64	80.03	84.37	95.32	**85.84**
ResNet50	MASF [7]	82.89	80.49	72.29	95.01	82.67
	DMG [4]	82.57	78.11	78.32	94.49	83.37
	ITL-Net [11]	87.1	83.3	96.1	79.3	86.4
	EoA [1]	90.5	83.4	98.0	82.5	88.6
	DNA [5]	89.8	83.4	97.7	82.6	88.4
	Style Neophile [16]	90.35	84.20	96.73	85.18	89.11
	RSC [14]	84.08	84.59	83.76	95.56	86.99
	Ours	87.93	85.53	86.68	96.83	**89.24**

*Work done while Xiang was at NEC Labs America

2. Additional analysis

Less dominant features assumption. Similar to the RSC algorithm, our method learns more generalizable features by muting the feature representations associated with the highest loss gradient, such that the network is forced to predict the labels through alternative features. Therefore, it is worthwhile to study loss difference at every iteration of the training algorithm. Loss difference can be expressed as $\Gamma(\hat{\theta}(t)) = |h(\hat{\theta}(t), \mathbf{z}_t) - h(\hat{\theta}(t), \tilde{\mathbf{z}}_t)|$, where $\hat{\theta}$ denotes the estimated parameters of the model at time t, $\tilde{\mathbf{z}}_t$ are masked features $\mathbf{z}$ and h denotes the task component of the backbone f defined as $h(\hat{\theta}, \mathbf{z}) = \sum_{(\mathbf{z},\mathbf{y})} l(f(\mathbf{z}; \hat{\theta}); \mathbf{y})$ with y being labels and l a generic loss function. We show that Γ is decreasing over training time of 30 epochs in Figure 1. Notice, that Γ is the empirical approximation of ξ, a key component in the generalization bound (see Corollary 1 in the RSC paper for detailed discussion) and therefore lower

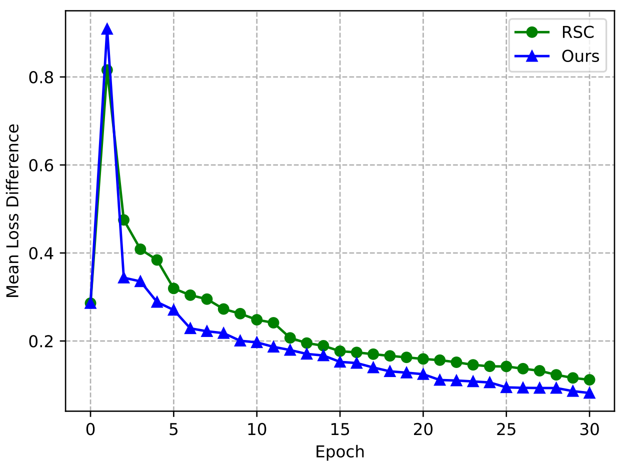

Figure 1: Temporal evolution of the loss difference $\Gamma(\theta(t))$ on PACS dataset.

Ablation of $M^{(1)}$ and $M^{(2)}$. We demonstrate the impact of the intermediate masks $M^{(1)}$ and $M^{(2)}$ of our approach in Table 4. If $M^{(1)}$ or $M^{(2)}$ is set to 0, all ResNet blocks will be zeroed and the network is unable to learn any relevant features. If we set $M^{(1)}$ to 1, the regularization process is guided solely by a Bernoulli distribution, resulting in a similar performance to the baseline. By setting $M^{(2)}$ to 1, our approach collapses to *DB+gsnr* , where the dropout ratios have to be tuned manually. With $M^{(1)}$ and $M^{(2)}$ both set to 1, no regularization occurs and the performance becomes similar to baseline.

3. Impact of ImageNet Pre-training

Following common practice, we pre-train our classification model on ImageNet [6]. For completeness, we also

Table 4: Ablation study: impact of the intermediate masks on the proposed *Meta-GSNR* approach. For simplicity, we denote 0 and 1 as all-zeros and all-ones matrices.

PACS	artpaint	cartoon	sketch	photo	Avg ↑
Meta-GSNR w/$M^{(1)} \cdot M^{(2)} = 0$	10.88	21.84	04.42	16.64	13.44
Meta-GSNR w/$M^{(1)} = 1$	78.12	76.57	75.16	94.49	81.08
Meta-GSNR w/($M^{(2)} = 1$	80.85	80.73	81.85	94.79	84.55
Meta-GSNR w/$M^{(1)} = M^{(2)} = 1$	77.49	77.21	72.67	92.45	79.95
Meta-GSNR	83.64	80.03	84.37	95.32	**85.84**

report accuracies obtained on PACS dataset using models trained from scratch in Table 5. Our approach outperforms RSC in both settings by approximately 2%. Similar to [17], we observe that a pre-trained baseline outperforms other domain generalization methods that are trained from scratch, indicating that both settings should be reported in future research.

Table 5: Classification accuracy (%) on the PACS dataset [20]. The bold numbers indicate the best performance in each setting.

PACS	artpaint	cartoon	sketch	photo	Avg ↑
Pre-trained on ImageNet					
Baseline	78.63	75.27	68.72	96.08	79.68
RSC [14]	80.73	79.22	81.48	94.16	83.90
Ours	83.64	80.03	84.37	95.32	**85.84**
Trained from scratch					
Baseline	52.19	65.01	68.23	75.38	65.20
RSC [14]	56.49	65.74	80.05	68.64	67.72
Ours	60.55	69.50	72.74	78.14	**70.23**

4. Reproduced Results

For fair comparison, in the experimental section of the main paper, we have reported the HTER and AUC metrics of *SSDG* [15], *SSAN* [26], and *EPCR* [27] methods obtained by running the code from the official repositories [1] [2] [3] . Similarly, we used the official code repository of RSC [14] [4] to report *baseline*, *RSC*, and *ours*.

5. Code Listings

Listing 1 illustrates how to obtain gradients in a typical forward pass, and Listing 2 shows how our dropout procedure can be applied. Our approach is based on TorchVision [5] implementation of DropBlock [12].

[1] https://github.com/taylover-pei/SSDG-CVPR2020
[2] https://github.com/wangzhuo2019/SSAN
[3] https://github.com/clks-wzz/EPCR
[4] https://github.com/DeLightCMU/RSC
[5] https://pytorch.org/vision/stable/index.html

```python
# forward pass to obtain gradients
class_logit, ResNetBlocks = model(images, labels, grads=None)
loss = criterion(class_logit, labels)
grads = [ torch.autograd.grad(loss, ResNetBlock)[0] for ResNetBlock in ResNetBlocks]
# forward pass to apply GSNR-guided dropout
class_logit, _ = model(data, labels, grads)
loss = criterion(class_logit, labels)
loss.backward()
optimizer.step()
```

Listing 1: Obtaining gradients in a forward pass

```python
def drop_block2d_gsnr(
        input: Tensor, grads: Tensor, p: float, p_gsnr: float, block_size: int,
        inplace: bool = False, eps: float = 1e-06, training: bool = True
) -> Tensor:
    """
    Args:
        input (Tensor[N, C, H, W]): The input tensor or 4-dimensions with the first one
                    being its batch i.e. a batch with ``N`` rows.
        grads (Tensor[N, C, H, W]): Gradients of the loss function with respect to
                    the input tensor.
        p (float): Probability of an element to be dropped.
        p_gsnr (float): Probability of dropout to be applied.
        block_size (int): Size of the block to drop.
        inplace (bool): If set to ``True``, will do this operation in-place. Default: ``False``.
        eps (float): A value added to the denominator for numerical stability. Default: 1e-6.
        training (bool): apply dropblock if is ``True``. Default: ``True``.

    Returns:
        Tensor[N, C, H, W]: The randomly zeroed tensor after dropblock.
    """
    def calc_gsnr(grads, eps=10**-7):
        ''' computes batch-wise gsnr '''
        grads_mean = grads.reshape(grads.shape[0], -1).mean(dim=0)
        grads_var = grads.reshape(grads.shape[0], -1).var(dim=0)
        gsnr = grads_mean**2/(grads_var+eps)
        return gsnr

    if p < 0.0 or p > 1.0:
        raise ValueError(f"drop probability has to be between 0 and 1, but got {p}.")
    if input.ndim != 4:
        raise ValueError(f"input should be 4 dimensional. Got {input.ndim} dimensions.")
    if not training or p == 0.0:
        return input

    assert grads.shape == input.shape
    N, C, H, W = input.size()
    block_size = min(block_size, W, H)

    gamma = (p * H * W) / ((block_size**2) * ((H - block_size + 1) * (W - block_size + 1)))
    gsnr = calc_gsnr(grads).reshape(grads.shape[1:]).unsqueeze(dim=0)
    thresh_idx = C*(H - block_size + 1) * (W - block_size + 1) * gamma * block_size**2
    thresh_idx = int(thresh_idx)
    thresh_val = torch.sort(gsnr.flatten(), descending=True)[0][thresh_idx]
    window_size = H - block_size + 1
    noise = gsnr[:,:,block_size-1:block_size-1+window_size,block_size-1:block_size-1+window_size]
    noise = (noise >= thresh_val)*1.0
    assert noise.shape == (N//N, C, H - block_size + 1, W - block_size + 1)

    noise_gsnr = torch.empty((1, C, H - block_size + 1, W - block_size + 1),
        dtype=input.dtype, device=input.device)
    noise_gsnr.bernoulli_(p_gsnr)
    noise = noise * noise_gsnr

    noise = F.pad(noise, [block_size // 2] * 4, value=0)
    noise = F.max_pool2d(noise, stride=(1, 1), kernel_size=(block_size, block_size),
        padding=block_size // 2)
    noise = 1 - noise # now high gsnr values are zeroed
    normalize_scale = noise.numel() / (eps + noise.sum())
    if inplace:
        input.mul_(noise).mul_(normalize_scale)
    else:
        input = input * noise * normalize_scale
    return input
```

Listing 2: GSNR-guided dropout strategy

References

[1] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *Advances in Neural Information Processing Systems*, 35:8265–8277, 2022. 1

[2] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019. 1

[3] Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 7354–7362, 2019. 1

[4] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *European Conference on Computer Vision*, pages 301–318. Springer, 2020. 1

[5] Xu Chu, Yujie Jin, Wenwu Zhu, Yasha Wang, Xin Wang, Shanghang Zhang, and Hong Mei. Dna: Domain generalization with diversified neural averaging. In *International Conference on Machine Learning*, pages 4010–4034. PMLR, 2022. 1

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[7] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32:6450–6461, 2019. 1

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009. 1

[9] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE, 2004. 1

[10] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 1

[11] Boyan Gao, Henry Gouk, Yongxin Yang, and Timothy Hospedales. Loss function learning for domain generalization by implicit gradient. In *International Conference on Machine Learning*, pages 7002–7016. PMLR, 2022. 1

[12] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[14] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020. 1, 2

[15] Yunpei Jia, Jie Zhang, Shiguang Shan, and Xilin Chen. Single-side domain generalization for face anti-spoofing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8484–8493, 2020. 2

[16] Juwon Kang, Sohyun Lee, Namyup Kim, and Suha Kwak. Style neophile: Constantly seeking novel styles for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7130–7140, 2022. 1

[17] Donghyun Kim, Kaihong Wang, Stan Sclaroff, and Kate Saenko. A broad study of pre-training for domain generalization and adaptation. *arXiv preprint arXiv:2203.11819*, 2022. 2

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[20] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017. 1, 2

[21] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018. 1

[22] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 1

[23] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 68–83. Springer, 2020. 1

[24] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011. 1

[25] Antonio Torralba, Bryan C Russell, and Jenny Yuen. Labelme: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484, 2010. 1

[26] Zhuo Wang, Zezheng Wang, Zitong Yu, Weihong Deng, Jiahong Li, Tingting Gao, and Zhongyuan Wang. Domain generalization via shuffled style assembly for face anti-spoofing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4123–4133, 2022. 2

[27] Zezheng Wang, Zitong Yu, Xun Wang, Yunxiao Qin, Jiahong Li, Chenxu Zhao, Xin Liu, and Zhen Lei. Consistency regularization for deep face anti-spoofing. *IEEE Transactions on Information Forensics and Security*, 18:1127–1140, 2023. 2

[28] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 1

[29] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pages 561–578. Springer, 2020. 1