Crafting Distribution Shifts for Validation and Training in Single Source Domain Generalization - Supplementary

Nikos Efthymiadis Giorgos Tolias Ondřej Chum
VRG, Faculty of Electrical Engineering, Czech Technical University in Prague
efthynik@fel.cvut.cz toliageo@fel.cvut.cz chum@cmp.felk.cvut.cz

1. Dataset Details

Digits dataset details. It is a collection of five digit-recognition datasets: MNIST, MNIST-M, SVHN, SYN, and USPS. MNIST-M combines the original MNIST handwritten digit database with random patches of the BSDS500 dataset. SVHN is a dataset of real-world house number images from Google Street View. SYN is a synthetic dataset created from different Windows fonts after applying geometric transformations and blurring. USPS is a dataset of scanned digits from U.S. Postal Service envelopes. To compare with the literature, we use only the first $10,000$ images of the MNIST training set. We use 90% for training and 10% for validation. We are evaluating on the test set of the rest domains.

PACS dataset details. It is a domain generalization dataset that includes four domains: photo, art paintings, cartoon, and sketch. It consists of seven classes and $9,991$ images. In the experiments presented in the main paper, the photo domain is used as the source, and the remaining domains are used for evaluation. This Supplementary provides an additional experiment where each domain is used as the source. The photo domain consists of $1,670$ images. We use the official partition of 10% for validation and the rest 90% for training.

Mini-DomainNet dataset details. It is a subset of the domain generalization dataset DomainNet [17]. It consists of $140,006$ images, 126 classes, and four domains: clipart, painting, real, and sketch. We use the real domain as source, and we evaluate on the rest. The real domain has $64,979$ images and we are using the official split that includes $58,482$ images for training and $6,497$ images for validation.

NICO++ dataset details. It is an out-of-distribution generalization dataset including $88,866$ natural images of 60 categories, where the following contexts serve as the domains: autumn, dim light, grass, outdoor, rock, and water.

Camelyon17 dataset details. It is a medical dataset focused on tumor detection, with data from five different hospitals. Data from hospitals 1, 2, and 3 are treated as the source domain, and data from hospitals 4 and 5 are the target. Camelyon17 consists of two classes: cancerous and non-cancerous tissue, and it contains $455,954$ images.

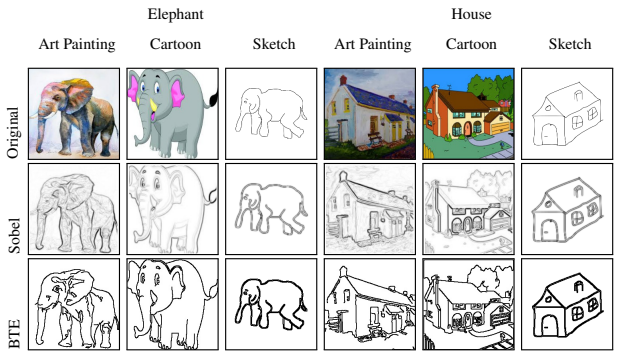

Figure 1. The output of the shape extraction process for BTE and Sobel-based edge maps from the different target domains in the PACS dataset. BTE removes texture cues more effectively, making it a better choice for increasing the shape bias.

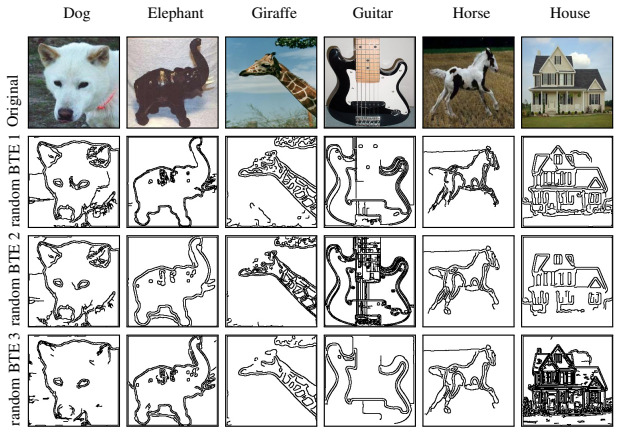

Figure 2. The output of the shape extraction process for BTE edge maps on images of the photo domain in PACS during training. BTE introduces randomization at multiple steps of edge detection, enriching the training set with edge maps that vary in level of detail.

16-class-ImageNet dataset details. The 16-class-ImageNet dataset [8] is a subset of the ImageNet dataset that maps 231 of the original classes to 16 new ones, closer to the level of abstraction a human could guess. Although the 16-class-ImageNet consists of $213,555$ images, we sub-sample to 500 images per class. We test the trained models to the texture-

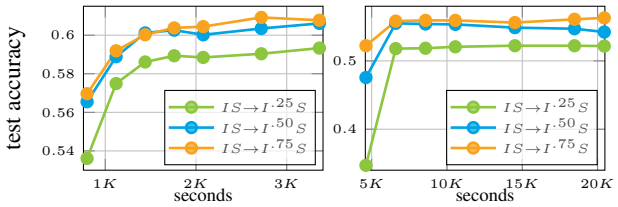

Figure 3. **Accuracy vs Time:** Training on PACS (left) and on Mini-DomainNet (right) with different number of BTEs in the batch. The training batch includes 64 RGB images plus 0, 8, 16, 24, 32, 48, and 64 BTEs (dots).

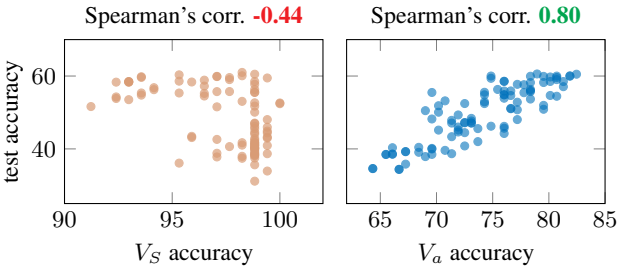

Figure 4. **Augmented validation without increasing the validation set size:** Correlation between the validation and test accuracy using the standard validation and the augmented one of the same size. This experiment shows that it is not the larger size of V_A that makes the difference. The experiment is on the PACS dataset with a ResNet-18 as a backbone.

shape cue conflict stimuli dataset [7], which is a dataset that shares the same 16 classes and consists of $1,280$ images.

2. Implementation Details

Shape extraction. The pipeline for BTEs and their randomization is adopted from [6] with the exception that we use Sobel instead of the learnable edge detectors, eliminating the need for additional training data. The pipeline is as follows: First, the image is blurred using a Gaussian filter with kernel size 5 and sigma equal to 1.0. Next, the Sobel operator is applied for edge detection, followed by non-maximum suppression to thin the edge map. Finally, the edge map is binarized using adaptive hysteresis. The upper and lower bounds of hysteresis are chosen as $1.5t$ and $0.5t$, respectively, where the threshold t is selected using Otsu's method on the edge map before thinning.

In the randomized variant used for training, the standard deviation of the Gaussian blurring is chosen randomly from 0,1, and 2, with 0 corresponding to no blur. The thresholding method is randomly picked among Yen [10], Otsu [16], Isodata [18], Li [13], and the mean method [9]. Additional random noise is introduced in both the threshold value t and the hysteresis bounds, enriching the training set.

In the variant using Sobel edge maps, the process is simplified in blurring and applying the Sobel operator. Randomization is only through the standard deviation of the Gaussian blurring. Examples of Sobel-based edge maps and BTEs are shown in Figures 1 and 2.

Method	Art	Cartoon	Photo	Sketch	Avg.
ERM	68.80	70.00	38.90	39.40	54.30
JiGen [1]	67.70	72.23	41.70	36.83	54.60
ADA [19]	72.43	71.97	44.63	45.73	58.70
SelfReg [11]	72.59	76.56	43.46	45.76	59.59
SagNet [15]	73.20	75.67	48.53	50.07	61.90
GeoTexAug [14]	72.07	78.70	49.07	59.97	65.00
L2D [20]	76.91	77.88	52.29	53.66	65.18
XDED [12]	76.50	77.20	59.10	53.10	66.50
CADA [4]	76.33	79.08	61.59	56.65	68.41
ITTA [3]	74.60	77.10	60.80	**61.20**	68.40
ProRandC [5]	76.98	78.54	62.89	57.11	68.88
MCL [2]	77.13	**80.14**	62.55	59.60	69.86
ABA_{3l}	75.34	77.49	58.86	53.76	66.36
$IS \rightarrow I^{\cdot 75} S$ (Ours)	**80.67** $\pm$ 0.4	76.53 $\pm$ 1.1	**65.85** $\pm$ 0.5	58.41 $\pm$ 1.3	**70.37** $\pm$ 0.5

Table 1. **Comparison with state-of-the-art approaches on PACS with a ResNet18 backbone.** Each column corresponds to a different source domain, reporting average performance when testing on the three remaining domains as target domains.

Implementation details for our approach. The *basic augmentations* include cropping with relative size in $[0.8, 1.0]$, an aspect ratio in $[\frac{3}{4}, \frac{4}{3}]$, resizing to 224×224, and horizontal flipping with a probability of 0.5. Digits is an exception where the resize is 32×32, and the flipping is skipped as it conflicts with the task. The *extra augmentations* from the ImgAug library are from the following groups: arithmetic, artistic, blur, color, contrast, convolutional, edges, geometric, segmentation, and weather.

For our PACS, Digits, and Mini-DomainNet experiments, the *learning rate* is tuned using a grid search among 33 equidistant values on a logarithmic scale in the range of $[10^{-5}, 1]$. The *loss weight* λ is tuned using a grid search among 17 equidistant values in $[0, 1]$. The *exponent* w is a test-time parameter, and it is tuned among the values 0, 0.25, 0.5, 0.75, and 1.0 for all of our experiments. The first and last values correspond to the variants S and I, respectively. Experiments on PACS and Digits for the comparison with the state-of-the-art are repeated 30 times. In all other experiments on PACS, Digits, and Mini-DomainNet, we use 5, 5, and 3 seeds, respectively.

Camelyon17 and NICO++ require longer training because of the larger size of the former and the randomly initialized training of the latter. Therefore, we perform a learning rate grid search for 9 equidistant values on a logarithmic scale, in the range of $[10^{-5}, 1]$, while we train for 3 different seeds.

For the 16-class-ImageNet experiment, we perform a grid search for our IS method's loss weight λ for 16 different values in the range of $[0.0625, 1]$.

We always use stochastic gradient descent with an exponential scheduler that decreases the learning rate by two magnitudes by the end of the training. We tune the number of epochs and the learning rate jointly for the $IS \rightarrow IS$ variant. This experiment provides us with the number of epochs to use for all variants, which remains fixed for all the follow-up experiments described in the main paper. We train our models for 10, 40, 50, 300, 300, and 700 epochs on Camelyon17, Mini-DomainNet, 16-class-ImageNet, Digits,

	PACS-ViT-S		PACS-RN18		MiniDN-RN18		MiniDN-Alexnet		NICO++-RN18		Digits-LeNet		Cam17-RN50	
Val	Method	Acc	Method	Acc	Method	Acc	Method	Acc	Method	Acc	Method	Acc	Method	Acc
V_O	$\hat{\imath}_{S\to IS}$	75.68	$\hat{\imath}_{S\to I^{.75}S}$	66.19	$\hat{\imath}_{S\to I^{.75}S}$	57.89	$\hat{\imath}_{S\to I^{.75}S}$	49.10	$\hat{\imath}_{S\to I^{.75}S}$	29.12	$\hat{\imath}_{S\to I^{.50}S}$	83.83	$\hat{\imath}_{S\to I}$	94.47
V_S	*ties**	68.70	$\imath_{S_{\times 2}\to I^{.50}S}$	51.75	$\imath_{S_{\times 2}\to I^{.75}S}$	52.98	$\imath_{I\to I}$	39.82	$\imath_{S\to I^{.75}S}$	26.86	$\imath_{S_{sob}\to I}$	72.51	$\imath_{I\to I}$	78.73
V_A	$\hat{\imath}_{S\to I^{.75}S}$	74.48	$\hat{\imath}_{S\to I^{.75}S}$	65.85	$\hat{\imath}_{S\to I^{.75}S}$	57.35	$\hat{\imath}_{S\to I^{.75}S}$	48.85	$\hat{\imath}_{S\to I^{.75}S}$	29.12	$\hat{\imath}_{S\to I^{.75}S}$	82.61	$\imath_{S\to I}$	93.56
gain		5.78		14.10		4.37		9.03		2.26		10.10		14.83

Table 2. **Method selection based on the validation set:** Test accuracy is reported after tuning and selecting the best method among all proposed variants according to different val sets – *i.e.*, oracle, standard, and augmented. The method chosen and the performance gain between the augmented and the standard validation set are reported. For ViT-S PACS, V_S ties across seven variations; we report the average.

			tune learning rate				tune loss weight λ			
		Train→Test	V_S	V_A	V_O	Gain	V_S	V_A	V_O	Gain
PACS	RN18	$IS\to I^{.25}S$	58.0	57.6	61.0	-0.4	37.9	56.1	57.7	18.3
		$IS\to I^{.50}S$	58.0	59.7	61.3	1.7	35.8	57.1	59.4	21.4
		$IS\to I^{.75}S$	56.7	59.2	61.3	2.5	49.0	57.5	60.4	8.5
PACS	ViT-S	$IS\to I^{.25}S$	62.6	65.4	65.9	2.8	63.1	64.9	65.2	1.8
		$IS\to I^{.50}S$	67.4	68.5	69.4	1.1	68.0	68.1	68.9	0.1
		$IS\to I^{.75}S$	69.4	71.2	71.5	1.8	70.4	70.6	71.2	0.2
MiniDN	Alexnet	$IS\to I^{.25}S$	45.3	45.5	45.9	0.2	43.6	44.9	45.4	1.3
		$IS\to I^{.50}S$	46.9	47.5	48.3	0.6	45.9	47.5	47.9	1.6
		$IS\to I^{.75}S$	47.8	47.8	48.1	0.0	46.1	48.0	48.6	1.9
MiniDN	RN18	$IS\to I^{.25}S$	51.8	51.6	51.9	-0.2	47.3	50.8	51.0	3.5
		$IS\to I^{.50}S$	55.0	55.0	55.5	0.1	48.1	53.9	54.6	5.8
		$IS\to I^{.75}S$	55.5	55.3	56.0	-0.2	50.1	54.7	54.9	4.6
Digits	LeNet	$IS\to I^{.25}S$	78.0	78.8	78.9	0.8	75.4	76.0	76.8	0.7
		$IS\to I^{.50}S$	77.8	78.8	79.1	0.9	76.1	76.3	77.2	0.2
		$IS\to I^{.75}S$	76.2	78.6	78.9	2.4	76.4	76.7	77.6	0.3
NICO++	RN18	$IS\to I^{.25}S$	23.4	23.2	23.6	-0.2	23.6	23.7	23.8	0.1
		$IS\to I^{.50}S$	25.8	25.7	26.2	-0.1	26.0	26.5	26.7	0.5
		$IS\to I^{.75}S$	26.1	26.0	26.6	-0.1	26.2	26.8	26.9	0.6
Cam17	RN50	$IS\to I^{.25}S$	92.0	92.3	92.4	0.3	92.1	92.2	92.4	0.1
		$IS\to I^{.50}S$	93.3	93.4	93.9	0.1	93.7	93.7	93.9	0.0
		$IS\to I^{.75}S$	93.8	94.3	94.5	0.5	94.1	94.1	94.5	0.0
Avg gain						0.7				3.4

Table 3. **Learning rate and loss weight tuning per validation:** Test accuracy for our method variants is reported after tuning according to different validation sets – *i.e.*, oracle, standard, and augmented. The performance gain between the augmented versus the standard validation set is also presented. As expected, the method is more effective for tuning hyperparameters related to domain generalization, such as the shape loss weight.

PACS, and NICO++ respectively.

Implementation details for the literature methods. Regarding SelfReg, SagNet, L2D, and ACVC, we follow all the implementation details – optimizer, schedulers, augmentations, and hyperparameters – from the original works, except for the learning rate. To determine the number of training epochs, we first set the learning rate to the value reported in the publication of the respective method and tune the number of epochs to maximize validation accuracy. Ties are resolved by picking the smaller number, while we never go for more than 800 or 100 epochs on PACS and Mini-DomainNet, respectively. Once the number of epochs is tuned, we tune the learning rate to maximize validation accuracy.

3. Extra Experiments

Performance vs Time: For the proposed validation V_A, there is no matter of time-performance trade-off. We argue that the standard validation V_S is completely incapable of predicting test performance in the context of domain generalization. This can be seen from Figure 5 of the main paper: PACS shows an accuracy drop of 22.2 if V_S is used over V_A. Methods that do not use the proposed augmentations in training, such as L2D and SelfReg, do not require the 2-fold cross-validation. In such cases, the time overhead of V_A over V_S is only the performance of a random augmentation. For methods that use augmentations, the extra time compared to V_S is approximately doubled because of the 2-fold cross-validation.

For the proposed recognition method, the performance vs training time trade-off is shown in Figure 3. Even with roughly half of the training time, when only 25% of the batch images have their BTEs (16 BTEs) used, the test accuracy decrease is approximately 1%. The time measurements were conducted on a single Tesla A100 40GB GPU.

V_A vs V_S: Effective because it is larger? The proposed validation method V_A increases the variability in the validation set as well as the size of the validation set by a factor of 10, which is given by the 10 groups of augmentations. To demonstrate that the benefit does not come from the larger validation set, we create an additional set by augmenting each image only once by randomly picking one of the 10 augmentation groups per image. The result is an augmented set of the same size as the original validation set, which we denote as V_a. We perform the same experiments as in Figure 4 of the main paper for PACS with a ResNet-18, but we exclude all variations that use augmentations during training to avoid overestimation. Figure 4 shows that validation V_a is still significantly better than V_S.

Experiments with each domain as the source domain. In Table 1 we report the performance on the PACS dataset while using each domain as the source domain. We consider this an invalid setup due to the ImageNet pre-training. The networks have seen both real images during the pre-training phase and also cartoons, artworks, or sketches during training, making it similar to an MSDG task. Additionally, evaluating on the photo domain no longer corresponds to testing on an unseen domain. Nevertheless, we report results following the example of the literature, and our method is the top performing. Training from scratch would make these setups valid for SSDG, but the literature lacks results for comparison.

Method selection and hyperparameter tuning. We summarize the results of our experiments for method selection in Table 2 and for hyperparameter tuning in Table 3.

Extra augmentations: Examples from the PACS dataset of all 76 extra augmentations are shown in Figures 5-7.

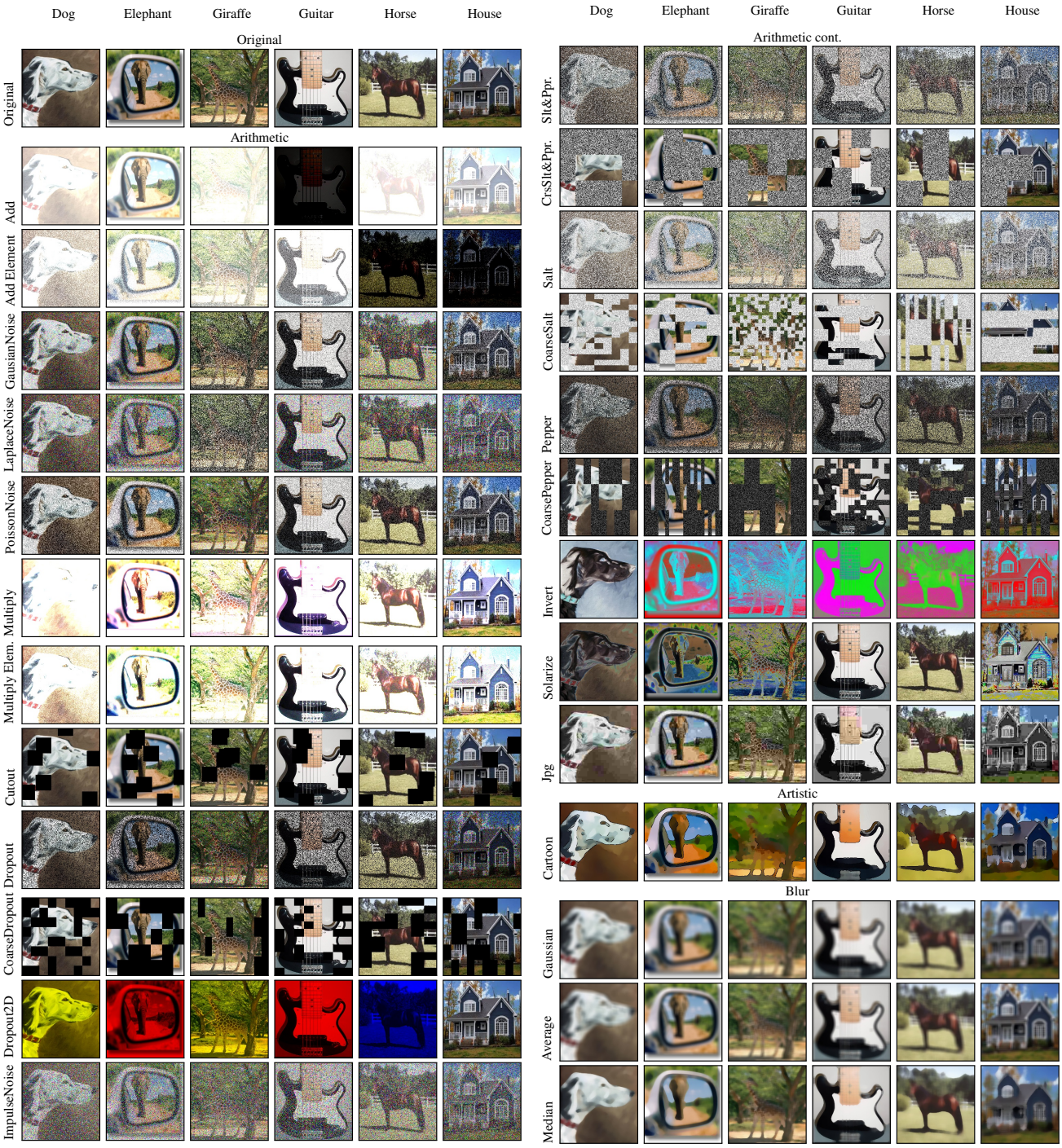

Figure 5. Examples of augmentations used from each augmentation category.

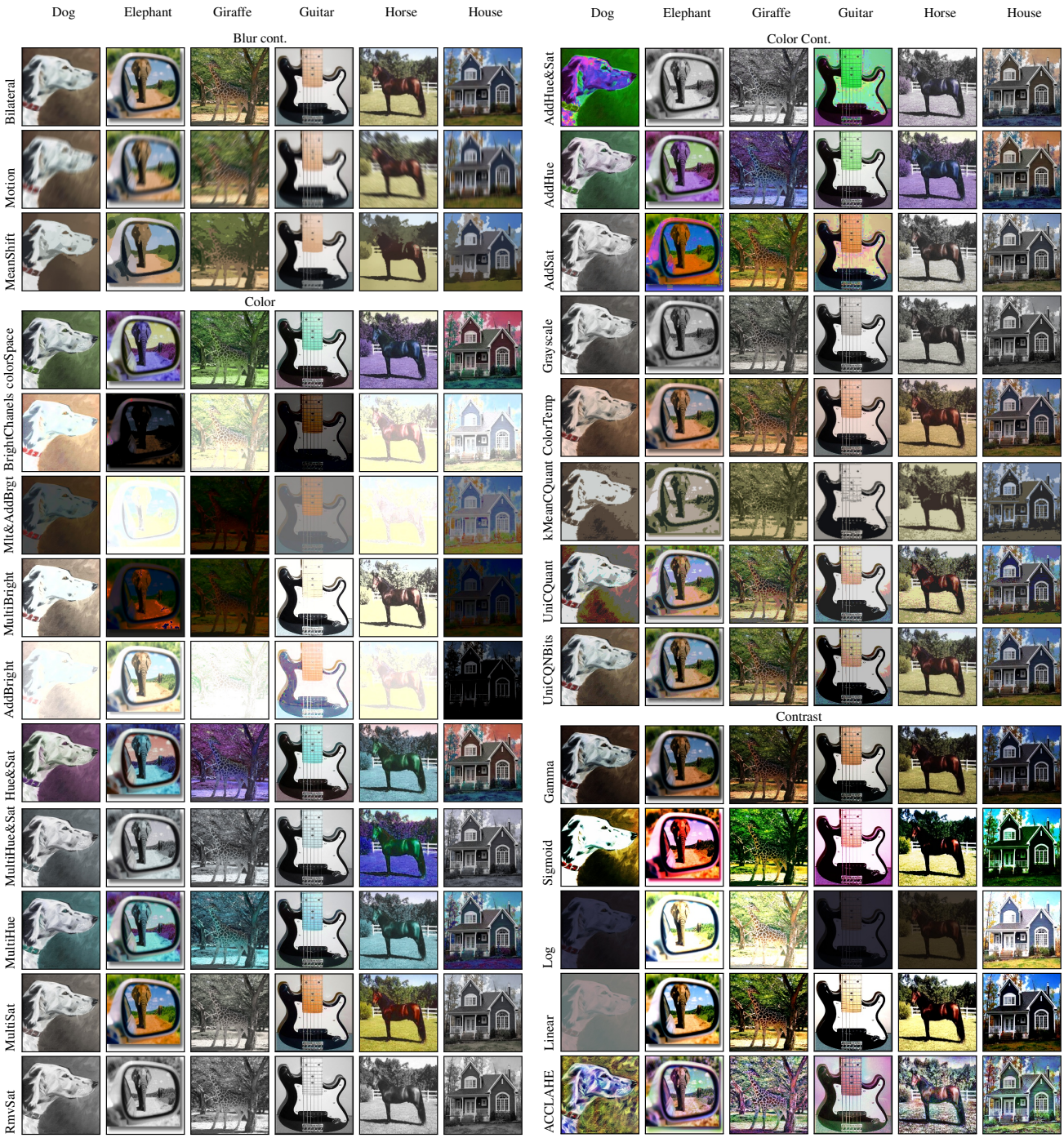

Figure 6. Examples of augmentations used from each augmentation category.

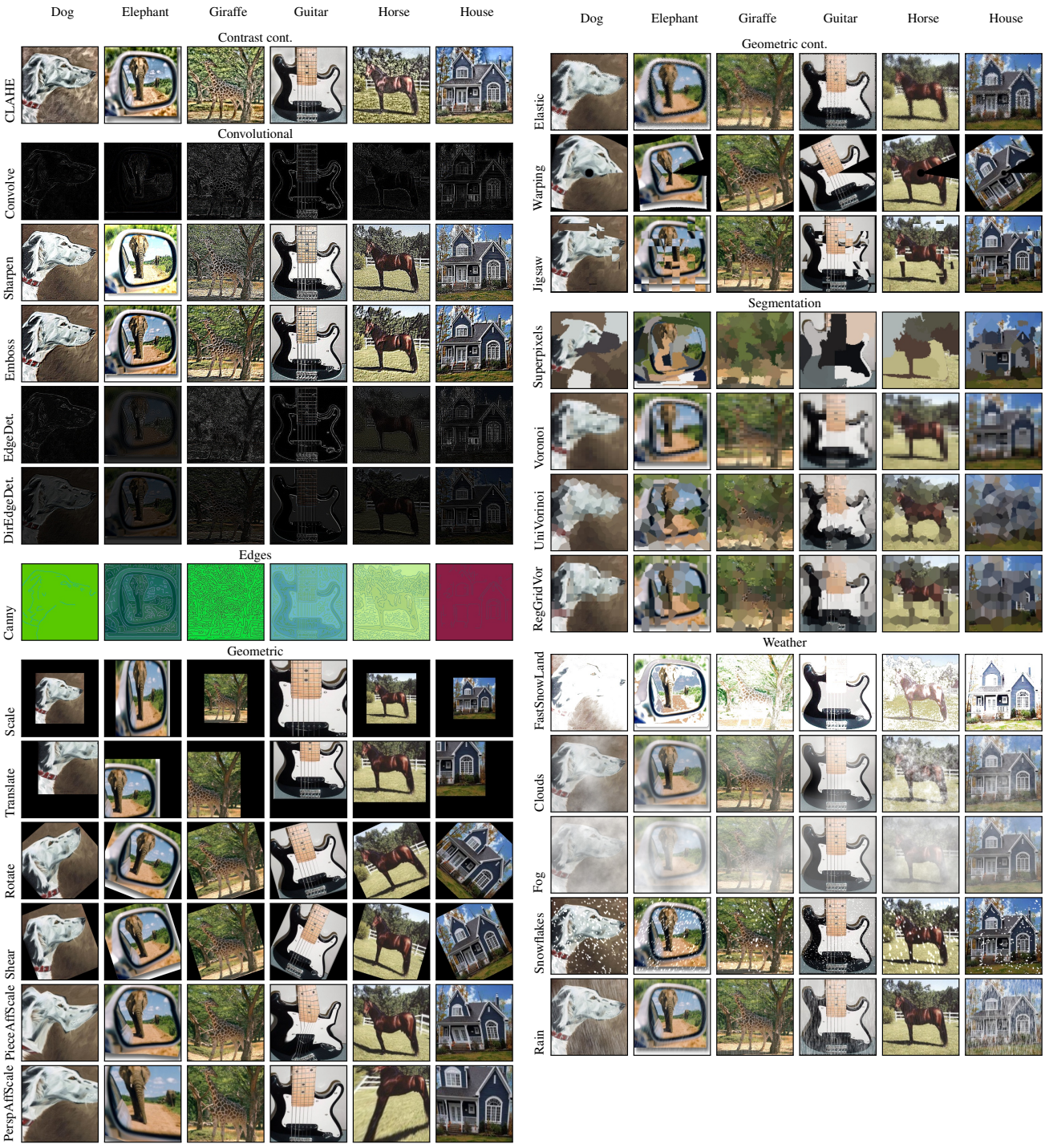

Figure 7. Examples of augmentations used from each augmentation category.

References

[1] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019. 2

[2] Jin Chen, Zhi Gao, Xinxiao Wu, and Jiebo Luo. Meta-causal learning for single domain generalization. In *CVPR*, 2023. 2

[3] Liang Chen, Yong Zhang, Yibing Song, Ying Shan, and Lingqiao Liu. Improved test-time adaptation for domain generalization. In *CVPR*, 2023. 2

[4] Tianle Chen, Mahsa Baktashmotlagh, Zijian Wang, and Mathieu Salzmann. Center-aware adversarial augmentation for single domain generalization. In *WACV*, 2023. 2

[5] Seokeon Choi, Debasmit Das, Sungha Choi, Seunghan Yang, Hyunsin Park, and Sungrack Yun. Progressive random convolutions for single domain generalization. In *CVPR*, 2023. 2

[6] Nikos Efthymiadis, Giorgos Tolias, and Ondrej Chum. Edge augmentation for large-scale sketch recognition without sketches. In *ICPR*, 2022. 2

[7] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019. 2

[8] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In *NeurIPS*, 2018. 1

[9] Chris A. Glasbey. An analysis of histogram-based thresholding algorithms. *CVGIP*, 1993. 2

[10] Yen Jui-Cheng, Chang Fu-Juay, and Chang Shyang. A new criterion for automatic multilevel thresholding. IEEE *Transactions on Image Processing*, 1995. 2

[11] Daehee Kim, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021. 2

[12] Kyungmoon Lee, Sungyeon Kim, and Suha Kwak. Cross-domain ensemble distillation for domain generalization. In *ECCV*, 2022. 2

[13] Chun Hung Li and CK Lee. Minimum cross entropy thresholding. *Pattern Recognition*, 1993. 2

[14] Xiao-Chang Liu, Yong-Liang Yang, and Peter Hall. Geometric and textural augmentation for domain gap reduction. In *CVPR*, 2022. 2

[15] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *CVPR*, 2021. 2

[16] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1979. 2

[17] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. 1

[18] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *Transactions on Systems, Man, and Cybernetics*, 1978. 2

[19] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *NeurIPS*, 2018. 2

[20] Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *ICCV*, 2021. 2