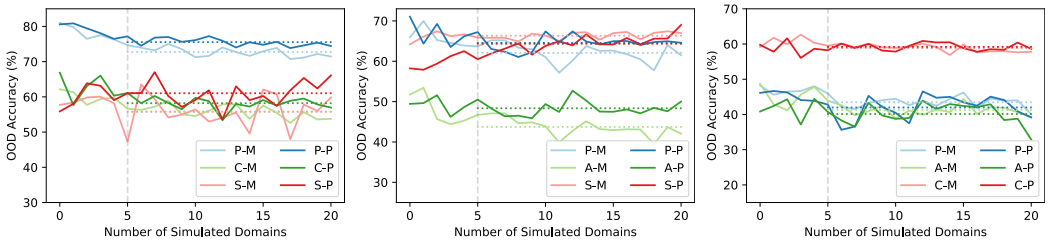


(a) Source: Art (b) Source: Cartoon (c) Source: Sketch

Figure 4: OOD accuracy (%) on PACS (Additional)

A Experimental Results

A.1 Experiments on PACS (Continued)

Here we present the results of additional experiments with the PACS benchmark.

Previous experiments on the PACS benchmark only used the Photo dataset as the source domain. In the following section, we report other cases where the source domain is changed (e.g., Art, Cartoon, Sketch). Here, we will denote each experiment as *Art as source*, *Cartoon as source*, and *Sketch as source*, respectively.

Table 5: sDG accuracy on PACS (Full).

Method	P	A	C	S	Avg.
Source: Photo					
Ours (AN+P)	____	52.46	50.29	66.79	**56.52**
Ours (AN+M)	____	57.54	46.89	64.93	56.45
Source: Art					
Ours (AN+P)	**78.07**	____	**66.04**	**63.15**	**69.09**
Ours (AN+M)	77.53	____	59.39	60.04	65.65
Source: Cartoon					
Ours (AN+P)	64.57	**50.02**	____	**69.00**	**62.04**
Ours (AN+M)	65.20	47.10	____	65.81	59.37
Source: Sketch					
Ours (AN+P)	46.25	44.31	61.60	____	50.72
Ours (AN+M)	**48.03**	**47.83**	60.32	____	**52.06**

In Table 5, we report the sDG accuracy of our two methods, MDAR and PROF, where AN, M, and P stands for AlexNet, MDAR, and PROF, respectively. Each row in the table displays the source domain, backbone type, and the training method (M/P). In cases where Art or Cartoon is used as source domain, training with our oracle regularization PROF marked higher OOD accuracy then its counterpart. On the other hand, PROF suffered when Sketch was set as the source domain, falling behind the baseline MDAR. Our hypothesis is that this behavior is triggered by the subpar performance of the oracle. To elaborate, the oracle used on the *Sketch as source* experiment displayed low OOD accuracy on the target domains, unsuitable for effective oracle regularization (Photo: 51.61%, Art: 39.39%, Cartoon: 56.85%).

Next, we present the analysis on mid-train OOD fluctuation in each experimental configuration. When the source domain is set as Art, employing PROF resulted in yielded a stabilization of the OOD performance, effectively mitigating fluctuations. The fluctuation was quantified as the reduction in variance across the target domain accuracy in $K > 5$. When compared with the conventional *augment & align* method MDAR, our regularization method PROF displayed large reductions in variance (Photo: 1.71→1.17, Cartoon: 3.13→2.97, Sketch: 21.50→11.22). The mid-train OOD fluctuation when source is set as Art, is depicted in Figure 4a.

Similarly, when the source domain is configured as Cartoon, PROF displays similar stabilization of the mid-train OOD performance. Using PROF allows a reduction in fluctuation, measured as variance (Photo: 5.15 → 3.06, Art: 5.00 → 3.07, Sketch: 0.70 → 3.91). We note that the stabilization effect in Sketch is relatively lower than that of other target domains, even lower than our *augment & align* baseline MDAR. The mid-train fluctuation is demonstrated in Figure 4b.

Lastly, we report the experimental results where the source was set as Sketch. In the *Sketch as source* experiment, we observe that PROF not only suffers in terms of performance but also exhibits instability. PROF displayed high variance in mid-train performance when compared to the baseline (Photo: 2.46 → 10.41, Art: 2.33 → 7.99, Cartoon: 1.01 → 1.04). The fluctuation is illustrated in Figure 4c. While a clear explanation is absent, we view that this phenomenon is caused by the underperformance of the oracle in the *Sketch as source* experiment. This result displays a clear example of the problems associated with the obstacles regarding the oracle, where obtaining an oracle may not be readily available. We further discuss the issue with oracles in the following section, Appendix D.

A.2 EXPERIMENTAL RESULTS ON DIGITS (CONTINUED)

Here we continue our analysis on the results of the Digits Experiment. In Section 5, we demonstrated that our regularization method PROF successfully mitigates issues of OOD fluctuation, measured as variance. This is illustrated in Figure 5 (M and P are from MDAR and PROF.). One notable observation is the significant increase in OOD generalization accuracy (81.82) when using PROF, in Table 3. As mentioned in the footnote, we do not claim this score to be state-of-the-art, as the true oracle is used. From the perspective of knowledge distillation, this is anticipated as the true oracle is already generalized to the target domains. In comparison, the approximated oracle in PACS does not guarantee robustness in the target domains, despite its higher generalizability. This confirms that a gap between the approximated oracle and the true oracle exists, which is a limitation that we acknowledge. We provide further analysis on the oracle in Appendix D.

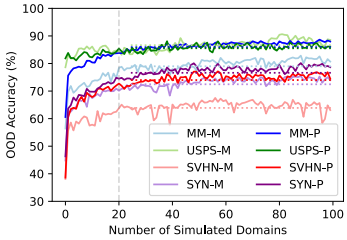


Figure 5: OOD accuracy (%) on Digits

Next, we discuss the results of our baseline experiment using MDAR. As mentioned in the main paper, our baseline surpassed state-of-the-art in Digits. In SVHN and SYNDIGIT (S-D), we show large improvement, while results in MNIST-M (M-M) show slight deficiency. Similar to existing methods, we refrain from using any form of manual data augmentation. We find that in Digits, increasing the number of simulated domains (K) helps OOD generalization. Both our baseline (MDAR) and PROF benefited from long training ($K > 100$).

A.3 EXPERIMENTAL RESULTS ON OFFICE-HOME (CONTINUED)

Here we continue our analysis of the results of the Office-Home Experiment. The Office-Home benchmark is not commonly used in the sDG literature, but we include the benchmark to bring attention to an important question: Is augmentation reliable for sDG?

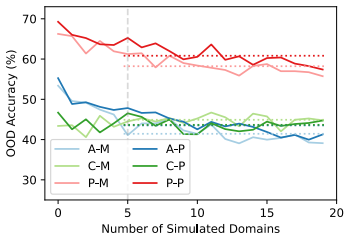


Figure 6: OOD accuracy (%) on Office-Home

As described in Table 4, augmentation-based approaches do show a boost in OOD accuracy. However, the effect gradually disappears with a sharp decline in OOD accuracy, as depicted in Figure 6. (A, C, and P are abbreviations of Art, Clipart, and Product domains, while M and P are from MDAR and PROF.) This downward trend is also spotted on other benchmarks, but not as intense. We believe that this phenomenon aligns with our analysis of the uncertainty of utilizing augmentation for OOD generalization. Our hypothesis is that the distributional gap within the Office-Home benchmark may be more intense than conventional sDG benchmarks (e.g., Digits, Corrupted CIFAR-10, PACS). The phenomenon brings novel questions on the efficacy of augmentation-based generalization methods. We believe that further research is required. Nonetheless, even in this case, PROF continues to stabilize the learning process, showing a smaller variance than our baseline (MDAR).

A.4 A SYNERGISTIC APPROACH: COMBINED USE OF MDAR AND PROF

In this section, we report the effect of using MDAR and PROF simultaneously. While PROF was designed for use without an alignment term (e.g., MDAR), we tested the effect of combining the two terms together. We observe that the synergistic method of PROF and MDAR triggered some differences in the training process.

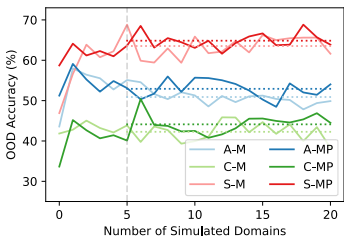


Figure 7: OOD accuracy (%) on PACS (MDAR + PROF)

Regarding the OOD accuracy, the synergistic method marked Art: 58.96%, Cartoon: 45.86%, Sketch: 64.57%, an average of 56.46% with AlexNet, as seen in Table 1. While the accuracy is slightly higher than using MDAR alone (56.45%), we view that the synergistic method does not significantly benefit the OOD performance. On the other hand, applying the synergistic method with a ResNet18 backbone showed a rise in OOD accuracy by a large

gap 1. Further research is necessary to provide an understanding of this behavior as no definitive explanation currently exists, while our hypothesis is that the model architecture may have caused the phenomenon.

Regarding the mid-train OOD fluctuation, the synergistic method was not able to reduce fluctuations across Art and Cartoon, while reducing the fluctuation in Sketch. (Art: 3.39→4.50, Cartoon: 5.22→5.86, Sketch: 7.23→3.52) Similar to previous experiments, the mid-train OOD fluctuation was quantified with the variance across the target domain accuracy in $K > 5$. The mid-train OOD fluctuation is depicted in Figure 7 (A, C, and S are from PACS and M and MP from MDAR and MDAR+PROF, the synergistic method.). Our hypothesis is that the two terms may have disrupted each other, while a clear explanation for this phenomenon remains elusive. We believe that additional research is needed to produce an effective synergy of both methods.

A.5 Study of Hyperparameters (Continued)

We explore our method's sensitivity to hyperparameters. (λ_{PROF}): λ_{PROF} is the hyperparameter used for PROF that operates as the balancing weight of the two functions in Equation (4). We begin with the value in the original paper of Zbontar et al. (2021) with $\lambda_{\text{PROF}} = 0.005$, and an alternate value $\frac{1}{d}$ introduced in Tsai et al. (2021) where d is the length of a vector in $\mathcal{D}$ (distillation head output space). We observe that our method is resilient to the switch between two candidate values of λ_{PROF} although we cannot guarantee they are optimal. (λ_{MDAR} and λ_{adv}): The study on λ_{MDAR} and λ_{adv} is processed similar to λ_{PROF}. Switching between $\lambda = 0.005$ and $\frac{1}{p}$ posed no notable impact on the learning process, where p is the length of a vector in $\mathcal{P}$ (projection head output space). While we cannot guarantee an optimal value. (w_{adv}, w_{cyc}, w_{div}): We optimize the hyperparameters w_{adv}, w_{cyc}, w_{div} using grid search. We find that as long as the weight-multiplied loss (wL) is situated on the $(0, 1)$ range, there is no significant impact on performance.

B Implementation Detail

In this section, we report the implementation details of our method.

B.1 Model Architecture

We report the details of model architectures used in our experiments. All models were built to match the architecture used in previous studies.

Task Model The task model architecture varies in each experiment. For each experiment, we report the feature extractor H, including an additional layer (i.e. buffer) used to match the feature extractor's output dimension to the oracle's.

The model used in the PACS experiment is AlexNet (Krizhevsky et al., 2012). The model consists of 5 convolutional layers with channels of $\{96, 256, 384, 384, 256\}$, followed by two fully-connected layers of size 4096 units. The buffer is a 2-layered MLP that maps the output dimension 4096 to that of the oracle (RegNetY-16GF), which is 3024. Hence, the final output dimension of the feature extractor is 3024.

The model used in the Corrupted CIFAR-10 experiment is a Wide Residual Network (i.e. WRN) of width $w = 4$ and depth 16 (Zagoruyko & Komodakis, 2016). WRN is a model that boosts its performance by widening the network by a certain factor w. The model consists of 4 network blocks with channels incrementally increasing as $\{16, 16w, 32w, 64w\}$. Specifically, the 4 blocks refer to an initial convolutional layer, followed by three additional network blocks. We further follow the original WRN design and set the dropout rate as 0.3. The buffer is a 2-layered MLP that maps the output dimension 256 to that of the oracle (ResNet50), which is 2048. Hence, the final output dimension of the feature extractor is 2048.

For the model used in the Digits experiment, please refer to Section 5.1. The architecture consists of two 5×5 convolutional layers, with 64 and 128 channels respectively. Each convolutional layer is followed by a MaxPooling layer (2×2). The network also includes two fully connected layers with sizes of 1024, 1024 being the final output dimension of the feature extractor. Since we do not employ oracle for the Digits experiment, a buffer was not added.

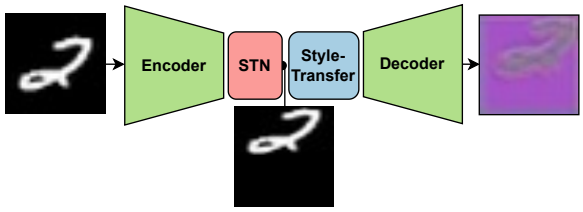


Figure 8: The illustration of the Generator.

Generator In this section, we describe the generator in detail. While the design of the generator slightly varies in each experiment, the basic architecture is the same. The generator consists of an encoder and a decoder, with a spatial transformer network (STN) and a style-transfer module in between the encoder and the decoder. The four components are placed in the order of `Encoder - STN - Style-Transfer - Decoder`.

We begin by illustrating the overall process of how an image is augmented by the generator. First, the input image is passed through the encoder to get a feature representation vector. The feature vector is then passed through the STN and the style-transfer module for modification. The modified vector is then reconstructed via a decoder, returning an augmented image. The mentioned process is illustrated in Figure 8. In the figure, we depict how each module modifies the input image.

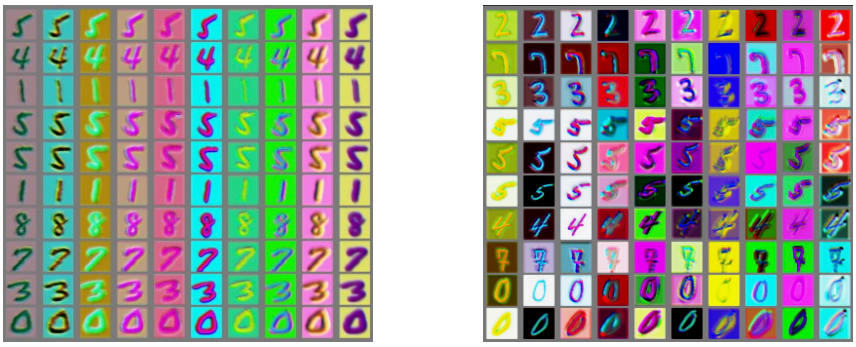
STN is a module that learns to perform spatial transformations on the input (Jaderberg et al., 2015). During the process, the STN module learns transformation parameters, where the parameters each define the magnitude of spatial transformations (e.g., rotation, scaling, translation). The STN module can be inserted at any point in the generator, allowing the generator to selectively transform the data up to a degree that is label-preserving. We place the STN right after the Encoder, following the experimental results of the original paper (Jaderberg et al., 2015). In Figure 8, we can see that the STN performs spatial transformations, creating the modified image at the middle. An advantage of STN is that no additional requirements are needed for training the module.

The style-transfer module modifies the features of the input image by adjusting the mean and standard deviation of the image features. This is performed using a normalization technique called Batch-Instance Normalization (i.e. BIN) (Nam & Kim, 2018). BIN selectively normalizes the features of the input image that are of less significance, while preserving features that are important. Note that this module is a modified version of the AdaIN method introduced in Huang & Belongie (2017), where we switched the normalization method from Instance Normalization (Ulyanov et al., 2016) to BIN for effective style transfer.

We share the results of applying these modifications in Figure 9. Whilst previous augmentation methods (Li et al., 2021; Wang et al., 2021) were limited to manipulating certain attributes (e.g., color, stroke), our method further allows spatial manipulations (e.g., shape, location). For instance, in the right image of Figure 9, we can observe that the images generated using our method displayed a large variance in shape, position, and color. This modification is inspired by recent studies on domain shift (Kaur et al., 2022; Wiles et al., 2021), which revealed that domain shift occurs on a variety of levels. However, an observable limitation is that the STN cannot transform complex images as in PACS, as small spatial modifications vastly change the semantics of the image. As depicted in Figure 10, the effect of the spatial modification is limited on PACS images.

Oracle Here, we report the architecture of the oracle. The oracle varies on the type of the experiment, (1) a RegNetY-16GF for the PACS and Office-Home experiment, (2) a ResNet50 for the corrupted CIFAR-10 experiment.

The RegNetY-16GF is a variant of the RegNet family, a line of models introduced in (Radosavovic et al., 2020) for image classification. The name of the model indicates its configurations, where the "Y" indicates the convolution method, and the "16GF" represents the model's capacity or complexity. We implement the model, and its model weights using the torchvision (Falbel, 2023) library. We used the weights pretrained via end-to-end fine-tuning of the original SWAG (Singh et al., 2022) weights on the ImageNet-1K data (Russakovsky et al., 2014). We then fine-tuned the pretrained model again with the Photo domain of PACS for 200 epochs, with a learning rate of $1e-4$ using



(a) Conventional Style-transfer (b) STN + Style-transfer

Figure 9: The illustrated comparison of the generators.

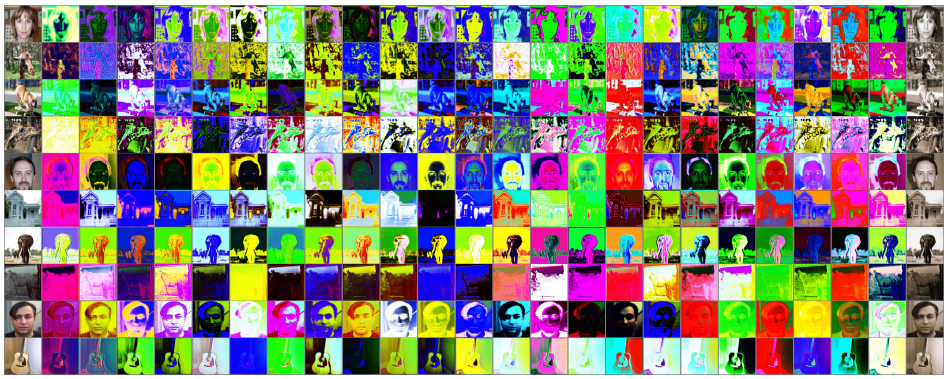


Figure 10: The illustration of generated images (PACS).

the SGD optimizer and the Cosine Annealing learning rate scheduler, a batch size of 64. For the Office-Home, we fine-tuned the pretrained model with the Real World domain of Office-Home for 30 epochs, using the SGD optimizer and the Cosine Annealing scheduler, a batch size of 16.

The ResNet50 is a variant of the ResNet family, a series of image classification models introduced in He et al. (2016). The name of the model indicates its depth, where "50" marks the number of layers. We implemented the model and its model weights using the torchvision library. For ResNet50, we used the weights pretrained with the ImageNet-1K dataset. We finetuned the pretrained ResNet50 with the CIFAR-10 dataset, the source domain of the corrupted CIFAR-10 experiment. In detail, we trained for 100 fine-tuning epochs, with a learning rate of $1e-4$ with the SDG optimizer and the Cosine Annealing learning rate scheduler, a batch size of 64.

B.2 Model Training

In this section, we elaborate on the details of the training process. We explicitly state the training hyperparameters (e.g., number of simulated domains (K), number of inner training loops for each generator, learning rate, the type of the optimizer, learning rate scheduler, and batch size). We further state the configurations of the projection heads (e.g., projection dimension ($\mathcal{Z}$) of the projection head P, projection dimension ($\mathcal{D}$) of the distillation head D).

PACS For the PACS experiment, we set K as 20, training each generator with 30 inner loops. During the first 15 inner loops we train the generator, and stop the training during the last 15 loops. We manually set the number of epochs by analyzing the training behavior of the generators. We set the learning rate as $1e-4$, using the Adam optimizer (Kingma & Ba, 2015). The batch size was set as 64. Regarding the model architecture, both the projection dimension ($\mathcal{Z}$) and the distillation head projection dimension ($\mathcal{D}$) were set as 1024.

Corrupted CIFAR-10 For the Corrupted CIFAR-10 experiment, we set K as 20, and 20 inner loops. During half (10) of the inner loops, we trained the generator and stopped the training during the remaining 10 inner loops. We set the learning rate as $1e-4$, with the Adam optimizer. The batch size was set as 256. The projection dimension ($\mathcal{Z}$) and the distillation head projection dimension ($\mathcal{D}$) were both set as 512.

Digits For the Digits experiment, we set K as 100, with 10 inner loops. Similar to the above two experiments, we trained the generator for 5 epochs and stopped the training for the other 5. Furthermore, the learning rate was tuned as $1e-4$, using the Adam optimizer. The batch size was set as 128. Finally, both the projection dimension ($\mathcal{Z}$) and the distillation head projection dimension ($\mathcal{D}$) were as 128.

Office-Home For the Office-Home experiment, we set K as 20, training each generator with 30 inner loops. During the first 15 inner loops we train the generator, and halted training for the remaining 15 loops. Similar to other cases, we set the number of epochs by analyzing the training behavior of the generators. The learning rate was set as $1e-4$, using the Adam optimizer. The batch size was set as 64. Regarding the model architecture, both the projection dimension ($\mathcal{Z}$) and the distillation head projection dimension ($\mathcal{D}$) were set as 512.

B.3 Model Pretraining

In this section, we report the information regarding the pretraining process. As mentioned above, we pretrained our task model with the source domain prior to the main training procedure. We announce the number of pretraining epochs, the learning rate, the optimizer, the learning rate scheduler, and the batch size.

PACS We pretrained the AlexNet with the train data of the Photo domain, using the train split introduced in the original paper (Li et al., 2017). We pretrained the model for 60 epochs, with a learning rate of $5e-3$ using the SGD optimizer. We further used the Step learning rate scheduler with a gamma rate (i.e. the strength of the learning rate decay) of 0.5. The batch size was set as 32.

Corrupted CIFAR-10 For the corrupted CIFAR-10 experiment, we pretrained the WRN with the train split of CIFAR-10. The pretraining epochs was set as 200, with a learning rate of $1e-1$ using the SGD optimizer. We used the Multi-Step LR scheduler, setting the gamma rate as $2e-1$, with milestones set as $\{60, 120, 160\}$. Hence, every time the training epoch reaches the milestone, the learning rate was reduced to one-fifth of the previous rate. The batch size was set as 128.

Digits Lastly, for the Digits experiment, we set the number of pretraining epochs as 100, with a learning rate of $1e-4$ using the Adam optimizer. The batch size was set as 256.

Office-Home We pretrained the ResNet18 with the train split of the Real World domain. We pretrained the model for 100 epochs, with a learning rate of $1e-4$ using the SGD optimizer. We used no learning rate scheduler. The batch size was set as 64.

B.4 Hyperparameters

In this part, we state the hyperparameters used in our experiments.

λ_{PROF} λ_{PROF} is a balancing coefficient for L_{PROF}, an objective adopting the feature-decorrelation loss introduced in Zbontar et al. (2021). We tuned λ_{PROF} using experimental results of the original paper and (Tsai et al., 2021). In the original paper, the author reported the optimal value of the balancing term as 0.005, which remains consistent under varying projection dimensions. We set this as a starting point for hyperparameter tuning. We find that if λ_{PROF} balances the off-diagonal term (i.e. redundancy reduction term) and the diagonal term (i.e. alignment term) to a similar degree, no significant differences are observed. Furthermore, switching λ_{PROF} to $\frac{1}{d} \approx 0.0001$ showed no significant changes to the learning process. Here, d denotes the projection dimension of the distillation head $\mathcal{D}$ (distillation head output space). While we cannot guarantee an optimal value for λ_{PROF}, we set $\lambda_{\text{PROF}} = 0.005$ for our two experiments using PROF.

$\lambda_{\text{MDAR}}, \lambda_{adv}$ The hyperparameters λ_{MDAR} and λ_{adv} is used together for adversarial learning, hence we report the two together. λ_{MDAR} was set in a similar way as λ_{PROF}. For our experiments, λ_{adv} was set as 0.005. λ_{adv} was searched under a fixed value of $\lambda_{\text{MDAR}} = 0.005$. We experimented with varying values of λ_{adv}: $\{0.005, 0.05, 0.5\}$, which showed no significant difference to the training process, while 0.05 showed slightly better results in the validation set of the source domain. Hence, in our experiments, λ_{adv} was set to 0.05. To explicate, generally, L_{adv} displayed a value approximately 10 times larger than L_{MDAR}. We believe that this behavior is correlated to 0.05 being a good value for λ_{adv} under a fixed value of $\lambda_{\text{MDAR}} = 0.005$.

All other hyperparameters (e.g., w_{cyc}, w_{div}, w_{adv}, w_{PROF}) are searched with a similar method to Li et al. (2021). For all experiments, we set w_{cyc} as 20.0, w_{cyc} as 2.0, and w_{adv} as 0.1 in Digits, and 0.02 in PACS and Corrupted CIFAR-10. Finally, w_{PROF} was set as 0.1. The values were tuned such that the weighted losses (i.e. wL) are situated in a similar range.

C On Domain Gaps

In previous works, there exist different mentions regarding the domain gap within the experimental datasets. We begin this section by comparing such views.

There are contradicting views on the domain gap within the PACS dataset, the authors of Wan et al. (2022) view that the domain gap is significant between the Art domain and the source domain (Photo), while relatively smaller with the Sketch and Cartoon domain. In contrast, Wang et al. (2021) viewed that the domain gap is the largest between the source and the Sketch domain, due to its vastly abstracted shapes. On the contrary, there exists a shared consensus regarding the domain gap within corrupted CIFAR-10 dataset, where researchers view that the domain gap between the source (CIFAR-10) and the target (corruption datasets) is defined by the severity level of the corruption (Li et al., 2021; Qiao et al., 2020; Wang et al., 2021; Wan et al., 2022). Concerning the Digits dataset, the authors of Qiao et al. (2020); Wang et al. (2021); Li et al. (2021) view that USPS displays the smallest domain gap with the source domain (MNIST). This is very similar to the view of Wan et al. (2022) that USPS and SYNDIGIT datasets are closer to the source, while there is a large domain gap between the MNIST-M and the source domain.

In our paper, we used a different measure to observe the domain gap between datasets: the OOD classification accuracy on unseen domains. Our view on domain discrepancy is that it can be indirectly observed through the downstream task performance. This is closely tied to realistic settings, where task performance is the leading motive behind the study of sDG. The method is simple: using a fixed model, we train the model with the train split of the source domain. Then, using the trained model, we test the classification accuracy on unseen domains. We reported the results in Section 5.2. Using the baseline OOD accuracy as a measure for domain gap matches the view of many existing works, while differences exist. For instance, USPS displays the highest OOD accuracy, matching the view of previous works that USPS shows the smallest discrepancy with the source (Qiao et al., 2020; Wang et al., 2021; Li et al., 2021; Wan et al., 2022). In PACS, the Sketch domain displays the lowest baseline OOD accuracy, which is in line with the view of some previous works (Wang et al., 2021), while different from the view of Wan et al. (2022).

D On Oracles

In this section, we discuss the implementation of the oracle using pretrained models. Using pretrained models for OOD generalization is not an entirely novel idea (Li et al., 2023; Cha et al., 2022), but first for the task of sDG.

We selected the pretrained RegNetY-16GF as an oracle for PACS. In Cha et al. (2022), a pretrained RegNetY-16GF model displayed high MI with the true oracle, a model that is trained on all source and target domains). The authors reported that the true oracle displayed an average validation accuracy of 98.4% on all PACS domains.

Similar to this, our implementation of the oracle with a pretrained RegNetY-16GF finetuned on the source domain (i.e. Photo in PACS, MNIST in Digits, Real World in Office-Home) displayed high validation accuracies across all target domains. To be specific, in PACS, the finetuned RegNetY-16GF marked 75.16%, 75.30%, 69.00% on Art, Cartoon, Sketch, and an average validation accuracy

of 73.15. While the average accuracy is lower than the true oracle in Cha et al. (2022), this is an expected behavior as our oracle used only the Photo domain, while the true oracle in (Cha et al., 2022) utilized all four domains of PACS.

However, we empirically confirm that the RegNetY-16GF is not universally available for use as the oracle. For instance, using the RegNetY-16GF to implement the oracle for the Corrupted CIFAR-10 experiment was not satisfactory. When finetuned with the source domain (i.e. CIFAR-10), RegNetY-16GF marked low validation accuracy in the target domain with an average of 60.65%. This is similar for the implementation with ResNet50, which marked an average accuracy of 61.25% on the target domains, performing worse than the task model. We believe that this difference is derived from the difference between the two datasets. For instance, PACS is a collection of images without any distortion, while the Corrupted CIFAR-10 is a dataset generated by vastly distorting CIFAR-10. As the RegNetY-16GF is not specifically trained to withstand distortions, its performance decrease in Corrupted CIFAR-10 is understandable. Similarly, the RegNetY-16GF does not fit well with the Digits benchmark due to the large gap between the pretrained dataset of the RegNetY-16GF and the Digit classification datasets.

This issue can be explained with the work of Wolpert & Macready (1997), where the authors demonstrate that there exists a trade-off between a model's performance on a certain task and the performance on all remaining tasks. We believe this to be a crucial limitation of our method, and aspire to investigate further.