Reappraising Domain Generalization in Neural Networks

Sarath Sivaprasad*[1,2] Akshay Goindani*[1] Vaibhav Garg[1]
Ritam Basu[1] Saiteja Kosgi[1] Vineet Gandhi[1]
[1]Kohli Centre on Intelligent Systems, IIIT Hyderabad [2]TCS Research, Pune

{sarath.s, akshay.goindani, saiteja.k, ritam.basu}@research.iiit.ac.in
vaibhav.garg@students.iiit.ac.in, vgandhi@iiit.ac.in

Abstract

Domain Generalization (DG) is perceived as a front face of OOD generalization. We present empirical evidence to show that the primary reason for generalization in DG is the presence of multiple domains while training. Furthermore, we show that methods for generalization in IID are equally important for generalization in DG. Tailored methods fail to add performance gains in the Traditional DG (TDG) evaluation. Our experiments prompt if TDG has outlived its usefulness in evaluating OOD generalization? To further strengthen our investigation, we propose a novel evaluation strategy, ClassWise DG (CWDG), where for each class, we randomly select one of the domains and keep it aside for testing. We argue that this benchmarking is closer to human learning and relevant in real-world scenarios. Counter-intuitively, despite being exposed to all domains during training, CWDG is more challenging than TDG evaluation. While explaining the observations, our work makes a case for more fundamental analysis around the DG problem before exploring new ideas to tackle it.

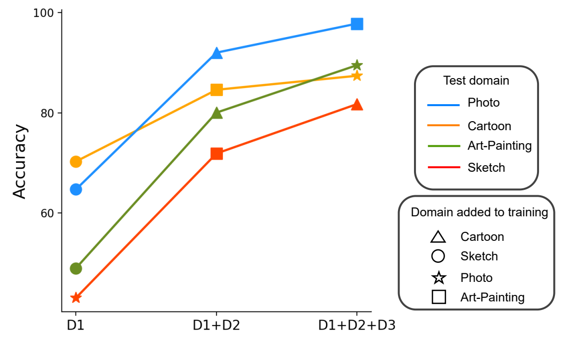


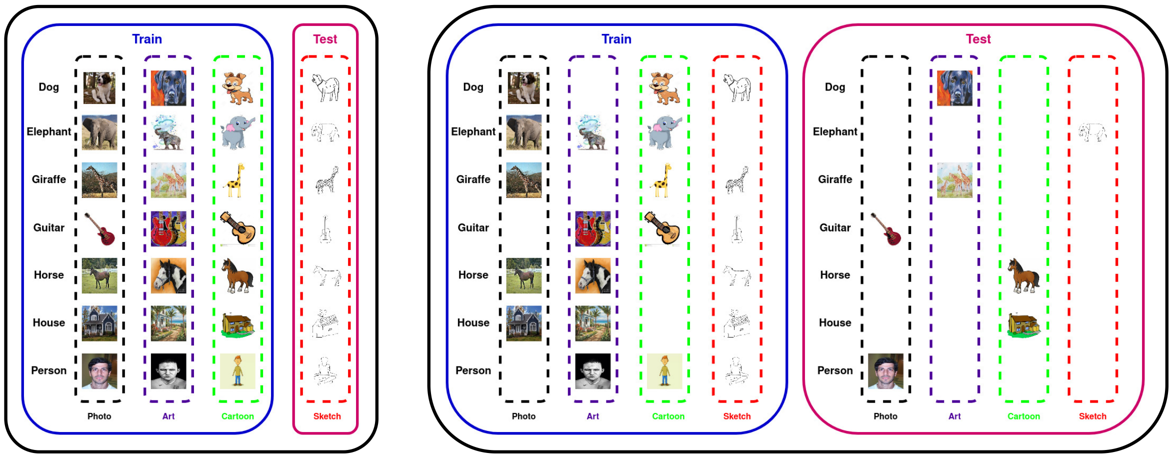
Figure 1. Test performance of Inception-Resnet backbone trained on different subsets of PACS dataset. The color of the line denotes the domain kept out for testing. At every tick on X-axis a new domain is added to train data. For instance, when the test domain is 'Cartoon'(yellow line), the model is first trained on just 'Sketch'(circle). In subsequent steps 'Art-painting'(square) and 'Photo'(star) are added to training data. The graph shows the increase in test performance on adding domains to train data.

1. Introduction

Generalization is a key goal in machine learning. Empirical results show us that sufficiently parameterized networks can completely fit any training data in any configuration of labels [46]. Hence, the most rudimentary notion of evaluation is to measure performance on unseen Independent and Identically Distributed (IID) data. This notion is brittle for real-world settings; for instance, minimal perturbations can substantially deteriorate the performance of models trained in IID setting [14]. Significant efforts have been made to improve the generalization of the neural network across perturbations [26]. However, more and more evidence is arising that robustness from synthetic image perturbations like noise, simulated weather artifacts, adversarial examples, etc., does not necessarily improve performance on distribution shift arising in real data [40]. Furthermore, performance on IID test data does not necessarily imply that the network has learned the expected underlying distributions [9]. Hence, evaluating generalization across Out of Distribution Data (OOD) is desirable for machine learning models.

Domain Generalization (DG) goes beyond perturbations and is a common way to formally evaluate OOD generalization. It focuses on scenarios where target domains have *distinct characteristics* but *no data for training*. DG aims to extract domain-agnostic features by learning from multiple domains, which can then be applied to an unseen domain. For instance, learn a model using labeled data of photos, paintings, cartoons and then apply it on sketches (traditional

*equal contribution



(a) TDG (b) CWDG

Figure 2. The figure shows the difference in train test split with Traditional DG (TDG) setting and ClassWise-DG (CWDG) setting. (a) shows one of the four splits in TDG, (b) shows one train test split out of the 16384 possible samplings. The open entry in the train set of CWDG corresponds to the entry in the test set.

DG setup is illustrated in Figure 2(a)). The motivation of DG is to produce human-like classification/representation models with deeper semantic sharing across domains - a horse is a horse irrespective of its form of depiction (a photo, cartoon, painting, or a sketch).

A myriad of *inventive methods* has been proposed for Traditional Domain Generalization (TDG). Some of the notable efforts include reverse gradients to obtain domain agnostic features [6]; kernel methods [28] to learn to map all domains to a common representation; style transfer for data augmentation [3]; jointly training domain agnostic and domain-specific models [18] and inhibiting features corresponding to the highest gradients in each domain [17]. Recently, Gulrajani and Lopez-Paz [11] show that an Empirical Risk Minimization (ERM) baseline gives a competent performance on TDG benchmarks, and none of the tailored methods give *any clear advantage* over the baseline.

In this paper, we explore the properties which are decisive for TDG. We discover that factors that aid IID generalization (optimization algorithm, augmentation, backbone) also play a key role in TDG. We find that DG tailored methods add no value over an optimally trained ERM. We observe that generalization properties of SGD [4] hold strong in TDG, and the robustness of a backbone in DG is proportional to its IID generalization. Our analysis helps us achieve a new State-Of-The-Art (SOTA) on six different DG datasets. Moreover, since initial explorations, TDG has been evaluated with training on *multiple domains*, which we uncover as the primary contributing factor to the performance on the unseen domain. Figure 1 shows that successively adding newer domains to the training data significantly improves performance on the unseen test domain.

Furthermore, as a key contribution, we propose an alternate setting to evaluate the generalization of neural networks: ClassWise-DG (CWDG). In this setting, a randomly selected domain from each class is kept aside for testing, as illustrated in Figure 2(b). Overall the model sees all domains during training, but only a subset of domains are seen for each class. We argue that CWDG benchmarking is closer to human learning. For example, a child in the early years may see some objects in photos and sketches, some in paintings and cartoons; but may not necessarily see all classes in all domains. In contrast, the TDG setting assumes that all class categories are seen in all source domains, and the target domain is never seen (suddenly, you are exposed to sketches one day!). Moreover, in the real world, the availability of class annotations is not uniform across domains; therefore, performance on the CWDG benchmark will show the model's efficacy in leveraging different kinds of supervision available in the real world. Contrary to the intuitive expectation, we observe that neural networks struggle to retain their performance despite seeing all the domains during training.

We argue that CWDG creates an incentive to learn domain-specific features and makes generalization difficult. In Figure 3 we show that how domain shifts across classes can lead to shortcut learning [9]. Learning to discriminate among domains creates a prior in CWDG, and hence learning domain-agnostic features becomes difficult. In contrast, in TDG, since all classes see a good proportion of all domains, learning domain-agnostic features is convenient for the network. This explains the observation in [11] that why ERM already gives robust output, which cannot be improved upon by tailored methods. We further experiment and show that using reverse gradients to learn domain-agnostic features gives 6% performance gains in CWDG over ERM. We believe CWDG emulates domains shifts present in real-world datasets and is a more robust benchmark to evaluate the ability to learn domain-agnostic features. Overall, our work makes the following contributions:

- Our work explains the findings in [11]. We uncover *why ERM works* and *why other methods fail to improve*. The analysis leads to a new SOTA on six DG benchmarks.
- We propose CWDG and perform thorough benchmarking using popular algorithms.
- Our analysis shows that the challenge of OOD generalization lies beyond TDG formulation. The distribution shift across classes appears as the primary hurdle.

2. Related Work

We discuss the prior art in two components. We first address the common notion of generalization in IID data. Subsequently, we discuss the previous works on TDG and motivate the need for the new CWDG setting.

Generalization in IID setting: Sufficiently parameterized networks can completely fit any training data [46]. Hence, a bare essential way to evaluate a neural network is to train on a randomly selected portion of the data and test on the unseen part. Popular methods like dropout [36], weight decay, early stopping, and regularization techniques [8, 42] have shown to improve this notion of generalization. It is common wisdom that spatial transforms in image data help improve generalization [26]. Constraining networks [2, 35] have also shown to improve generalization in IID setting. The optimizer also plays a role in generalization; specifically, SGD is shown to achieve better generalization than adaptive algorithms [44].

Recently Taori *et al.* [40] suggests considering generalization beyond IID setting with perturbations. They report a thorough study with 204 ImageNet models, showing that robustness from synthetic image perturbations like noise, simulated weather artifacts, adversarial examples, etc., does not improve the performance on distribution shift arising in real-world data. Moreover, Recht *et al.* [31, 32] expose the problems in using a specific part of IID distribution as test data. They show a drop in performance when tested on new test data collected from the same distribution, motivating the evaluation beyond the IID setting.

Domain generalization: Our work focuses on the DG in deep neural networks, and for pre-deep learning efforts, we refer the reader to the review by Moreno *et al.* [27]. Furthermore, we limit our discussion to DG in image classification. TDG formulation involves learning from multiple domains and testing on an unseen domain. TDG on image classification is commonly evaluated on six datasets: [1, 5, 10, 20, 29, 41].

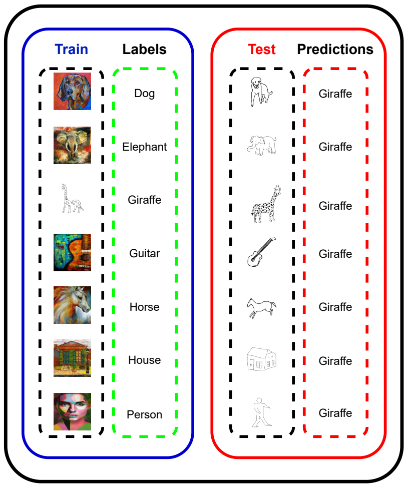


Figure 3. The figure shows a synthetic data setting, where a domain shift is created on a specific class: ‘Giraffe’. All train samples are chosen from the domain ‘Art-Painting’ except the class ‘Giraffe’, which is sampled from ‘Sketch’. At inference, ‘Sketch’ images from across classes are predicted as ‘Giraffe’.

Learning domain agnostic features using the TDG formulation has seen significant interest in recent years. The problem has been approached from many different directions like data augmentation [3, 45, 48], gradient manipulation [6, 17], ensemble learning [24], and feature disentanglement [18, 30]. For a comprehensive list, readers can refer to the recent surveys [43, 47]. It is worth noting that several of these ideas [6] have found widespread success beyond the TDG framework.

Gulrajani and Lopez-paz [11] suggest that inconsistencies in experimental conditions (datasets and training protocols) render fair comparisons difficult. They propose DomainBed, a unifying benchmark for TDG. They empirically show that a carefully implemented ERM outperforms the state-of-the-art in terms of average performance. A natural thought that arises is that despite the proposition of numerous inventive ideas for TDG, why none of them improves over the baseline ERM. In this work, we claim that TDG is not an appropriate formulation to measure the efficacy of a model to learn domain agnostic features. We argue that in TDG formulation, learning domain agnostic features is the most convenient thing for the network, not a challenge.

Consequently, we propose CWDG, a more challenging DG formulation, which leaves room for shortcut learning [9]. Our work interestingly contrasts with [25] which proposes further *constraints* on TDG by introducing unseen classes in the test domain. We instead *relax* the assumptions and expose all domains during training.

Dataset	# Domains	Domains	# Classes	# Images
RMNIST [10]	6	0°, 15°, 30°, 45°, 60°, 75°	10	70000
CMNIST [1]	2	Red, Green	2	120000
DomainNet [29]	6	Clipart, Infograph, Painting, Quickdraw, Real, Sketch	345	586575
PACS [20]	4	Photo, Art-Painting, Cartoon, Sketch	7	9991
VLCS [5]	4	Caltech101, LabelMe, SUN09, VOC2007	5	10729
Office-Home [41]	4	Art, Clipart, Product, Photo	65	15558

3. Method

We follow the notations presented in [20]. We consider S available domains containing samples from a total of C classes. Let c_i represent the class c corresponding to i^{th} domain. Furthermore, let N_{c_i} be the number of labeled samples in class c and domain i. $y_{c_i}(k)$ represents the label corresponding to k^{th} such sample and $\hat{y}_{c_i}(k)$, the label prediction. ℓ is the loss computed between the true and predicted labels. The objective for minimization in the TDG setting is as follows:

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{S-1} \sum_{i=1:S_i \neq S_{test}}^{S} \frac{1}{C} \sum_{(c_i \in C)} \frac{1}{N_{c_i}} \sum_{k=1}^{N_{c_i}} \ell\left(\hat{y}_{c_i}(k), y_{c_i}(k)\right),$$

where Θ are the network parameters.

In the proposed CWDG setting, the optimization problem is modified as follows:

$$\underset{\Theta}{\operatorname{argmin}} \frac{1}{S} \sum_{i=1}^{S} \frac{1}{C_{S_i}} \sum_{(c_i \in C : c_i \neq c_{i_{test}})} \frac{1}{N_{c_i}} \sum_{k=1}^{N_{c_i}} \ell\left(\hat{y}_{c_i}(k), y_{c_i}(k)\right).$$

Where, C_{S_i} is the number of classes present in the domain S_i for train. Assume Θ_{*_1} is obtained post minimization in TDG setting, and Θ_{*_2} is the solution obtained under CWDG. Θ_{*_1} is evaluated on unseen domain S_{test}. On the other hand, Θ_{*_2}, which has been optimized over all domains in S, is evaluated on unseen class instances $c_{i_{test}}$. The solution in CWDG, Θ_{*_2}, intuitively needs to be able to discriminate classes agnostic to which domain it appears in S, but our empirical evidence show otherwise.

We look at the type of shift incurred in CWDG compared to classical TDG setting following the study of datashift by Moreno-Torres *et al.* [27]. They study various data shift and broadly classify them into four different categories, namely:

- Covariate shift: $P_{test}(x) \neq P_{train}(x)$ but $P_{test}(y|x) = P_{train}(y|x)$
- Prior probability shift: $P_{test}(y) \neq P_{train}(y)$ but $P_{test}(y|x) = P_{train}(y|x)$
- Concept shift : $P_{test}(x) = P_{train}(x)$ but $P_{test}(y|x) \neq P_{train}(y|x)$
- Dataset shift: $P_{test}(x, y) \neq P_{train}(x, y)$ but none of the above hold.

CWDG falls under the category of dataset shift, while TDG formulation falls under covariate shift. Both measures of robustness are distinct and desirable for corresponding real-world applications.

To analyse the performance of networks in CWDG, we use Frechet Inception Distance (FID) [15] as a measure of distribution shift between the train and test split of the datasets. FID measures the distance between two multinomial Gaussians d as,

$$d\left((m_1, C_1), (m_2, C_2)\right) = \\ \left(\|m_1 - m_2\|_2^2 + \operatorname{Tr}\left(C_1 + C_2 - 2\left(C_1 C_2\right)^{\frac{1}{2}}\right)\right)^{\frac{1}{2}}$$

where $d\left((m_1, C_1), (m_2, C_2)\right)$ is the Frechet distance, m_1, m_2 are the means, C_1 and C_2 are the covariances of two gaussian distributions, Tr denotes the trace of a matrix.

We compute the distance d between the embeddings of the final layer of an InceptionV3 network [39] pre-trained on ImageNet. To compute the FID score in each experimental setting, we pass the train and test split through the pre-trained network and calculate the distance d of the embeddings.

4. Experiments

In this section, we present our three experimental frameworks. First, we perform an ablation study to explore the effect of model backbone, augmentation, and optimizer in TDG setting using PACS dataset [20]. Then we compare our best-performing model and training strategy against ten benchmark algorithms across six datasets from DomainBed [11] in the TDG setting. In our third experiment, we report the performance of ten DG algorithms on the CWDG formulation of the six datasets.

4.1. Exploring effects of different modelling choices

In the ablation experiments, we use PACS [20] dataset. Following TDG, we keep aside one domain for testing in each fold and train on the other three. We use an oracle for model selection. To measure the effect of each intervention in the ablation, we keep all the other modeling choices

	Adam with augmentation				SGD without augmentation				SGD with augmentation			
Backbones	**Photo**	**Sketch**	**Art**	**Cartoon**	**Photo**	**Sketch**	**Art**	**Cartoon**	**Photo**	**Sketch**	**Art**	**Cartoon**
Alexnet	28.06	19.3	27.9	35.39	88.26	60.42	65.645	70.065	87.69	69.17	66.91	69.28
Vgg-19_BN	25.88	20.38	21.09	21.34	88.41	77.63	76.31	78.47	90.27	82.12	76.78	79.6
Resnet-18	39.95	40.02	26.8	38.42	87.96	74.38	75.7	77.38	87.34	80.29	73.64	76.91
Resnet-50	35.06	34.95	26.28	33.15	88.53	78.21	72.99	79.28	89.57	81.02	74.91	77.72
DenseNet-121	34.13	36.5	28.39	30.18	87.50	79.1	73.4	76.44	88.9	80.42	74.31	78.15
Inc-Res	53.12	52.16	37.8	50.69	96.12	81.36	84.96	83.69	95.06	87.35	88.8	84.8

Table 1. The table shows DG results on PACS and the effect of various modeling choices. The compiled results show the disparity in accuracy between the two optimizing algorithms, the different backbone models, and augmentation. Our best-performing backbone 'Inc-Res' corresponds to Inception-Resnet

the same and run the experiment five times and report the mean value. Unless specifically mentioned otherwise, the training protocol and hyper-parameters are the same as in DomainBed [11].

We train a simple ERM with six different backbones, namely: AlexNet [19], VGG-19 [34], ResNet-18 [12], ResNet-50 [13], DenseNet-121 [16] and Inception-Resnet [38]. We run all four folds of PACS on these backbones with SGD and ADAM optimizer. As the next intervention, we run all the aforementioned backbones with and without augmentations with an SGD optimizer. For augmentation, we use standard image perturbations as used in [17]. We select random crops from train images and resize them into model input size. We randomly flip half the images in every batch horizontally (with 0.5 probability) and use color-jitter augmentation by randomly scaling brightness, contrast, saturation, and hue. These scales are sampled randomly from a distribution between 0.6 and 1.4. We randomly choose images from each batch and grayscale them (with a probability of 0.1). We normalize all images with ImageNet mean and standard deviation.

We train each model for 30 epochs with a batch size of 32. We use a learning rate of 0.01 with no scheduler across all the runs, except for Alexnet (learning rate of 0.001) since it does not converge at higher values. We use ImageNet pre-trained weights for all the backbone models and fine-tune the last layer with a categorical-cross-entropy loss function. We use a weight decay of 0.0005 and momentum of 0.9.

4.2. Comparisons on TDG benchmark

We compare the performance of the best model from Section 4.1 across different datasets using DomainBed [11]. We select the backbone and the hyperparameters using one of the datasets (PACS dataset). We use the same choices for the other datasets. We argue that doing so reduces the chances of overfitting the training strategy on other datasets. We compare our best performing model with the existing methods on six different datasets: PACS [20], VLCS [5], Office-Home [41], DomainNet [29], CMNIST [1] and RMNIST [10]. The statistics of the datasets are given in Table **??**.

We use the hyperparameters and random-seed used in DomainBed to maintain uniformity of comparison. We follow the augmentation strategy used in DomainBed. This follows that MNIST datasets, namely CMNIST [1] and RMNIST [10] are not augmented. We compare our results against popular methods, namely: C-DANN [23], IRM [1], MLDG [21], DRO [33], MMD [22], ERM [11], CORAL [37], Mixup [45], RSC [17], DANN [7] and GRL [6]. Outside of DomainBed, we also use multi-branch reverse-gradient [6] model on the Inception-ResNet backbone. We run these experiments five times and use and report the mean accuracies. We use oracle for model selection.

To make the comparison of a neural network with all the methods in DomainBed [11] more rigorous, we apply all the methods on top of the the best performing ERM (Inception-Resnet backbone). We then evaluate the performances in the Domainbed style of training across the six different datasets.

4.3. Benchmarking CWDG

In this setting, we follow the CWDG formulation of the aforementioned datasets (Section 4.2). We keep out a random domain from each class to create a train test split. We do this split multiple times. Results of one of the splits are presented in Table 3. The other results are presented in the supplementary material. We use the same augmentations and hyper-parameters as in DomainBed. We evaluate the efficacy of the methods specified in Section 4.2 on CWDG.

We explore the efficacy of reverse-gradient [6] in CWDG formulation. We branch the model before the last fully connected layer. One branch has four outputs (corresponding to the four domains in the train data), and the other branch has seven outputs (corresponding to the seven classes). We flip the gradients flowing through the domain arm by multiplying with a factor of -1.

Algorithms	PACS	VLCS	Office-home	Domain-Net	CMNIST	RMNIST	Average
IRM	82.9	77.2	66.7	32.6	59.16	97.7	69.37
GRL	83.69	77.38	70.2	37.4	50.5	98.49	69.61
MMD	82.8	76.7	67.1	28.4	73.35	98.1	71.07
DANN	84	77.7	65.5	38.1	73.03	89.1	71.23
C-DANN	81.7	74	64.7	37.9	73.03	96.3	71.27
DRO	83.1	77.5	67.1	33.4	73.35	97.9	72.05
RSC	84.77	78.8	70.8	39.2	61.2	98.23	72.16
MLDG	82.4	77.1	67.6	41.6	71.64	98	73.05
Mixup	83.7	78.6	68.2	38.7	73.34	98.1	73.44
CORAL	83.6	77	68.6	40.2	73.35	98.1	73.47
ERM-Inc-Resnet	89.11	78.84	71.95	43.2	74.35	99.2	76.10

Table 2. Comparing ERM-Inc-Resnet with other algorithms in DomainBed. The algorithms are sorted by their average performance across the six datasets.

Algorithms	PACS	VLCS	Office-home	Domain-Net	CMNIST	RMNIST	Average
IRM	64.8	63.1	55.77	28.8	61.58	71.2	57.53
RSC	79.3	64.5	65.2	25.3	50.5	98.7	63.91
MMD	73.8	60.2	65.46	25.8	73.05	98.5	66.13
DANN	74.4	64.2	62.95	24.6	72.05	98.8	66.16
MLDG	73	62.97	65.87	25.73	71.93	98.5	66.3
CORAL	77.06	60.2	65.46	25.8	73.5	98.5	66.75
C-DANN	77.7	63.77	64.58	24.04	72.5	98.9	66.91
ERM-Inc-Resnet	79.6	60.86	66.1	25.8	71.15	99.8	67.21
Mixup	77.6	64.2	66.02	25.1	73.05	98.3	67.35
DRO	79.38	64.77	66.1	25.15	73.05	98.5	67.82
GRL	86.2	67.87	66.9	26.9	74.15	99.3	70.22

Table 3. Comparing the performance of different algorithms in DomainBed against Inception-Resnet-ERM and GRL(with Inception-Resnet backbone) in CWDG setting. Algorithms are sorted by their average performance across the six datasets.

5. Results

Our method involves simply training a backbone (trained on ImageNet) with SGD and standard data augmentation (and nothing else!). Table 1 shows the effect of different backbones and motivates the use of SGD and data augmentation. Next, we compare our plain baseline against the state-of-the-art methods in TDG. We then report the performance of algorithms covered in DomainBed under the CWDG setting.

5.1. Exploring effects of different modeling choices

Effect of optimizer: Table 1 compiles the accuracy of different backbones under SGD and ADAM optimization over all domains in PACS. SGD outperforms ADAM by a significant margin for a given learning rate for all the six backbones across all domains. Comparing the average performance difference between domains, we observe that ADAM gives a lower performance on 'Sketch' and 'Art-Painting', domains farther from the domains the backbones were trained on (ImageNet). For the 'Art-Painting' domain, Inception-Resnet backbone with SGD gives more than twice the accuracy of ADAM. Averaged over the four domains, SGD gives 89.00% accuracy compared to 48.44% accuracy given by ADAM. The results demonstrate that SGD has a clear advantage over ADAM in the studied scenario. The observation may stem from the fact that fine-tuning a large ImageNet model on a relatively small dataset like PACS reduces to a simple overparameterized problem. Moreover, previous work [44] has suggested that for simple overparameterized problems, adaptive methods can find drastically different solutions than SGD.

Effect of augmentation: Table 1 compares the accuracy of different backbones trained using SGD, with and without augmentation. We observe that augmentations almost always improve the performance of networks. For instance, with the Inception-Resnet model, the average performance of the model across all four domains with augmentation is 89.00%, which is higher than the average accuracy without augmentation 86.53%.

Algorithms	PACS	VLCS	Office-home	CMNIST	RMNIST	Average
IRM	88.9	75.8	56.6	61.54	79.2	72.40
GRL	86.4	75.4	65.8	51.6	98.49	75.53
MLDG	76.4	72.6	67.6	73.4	98.1	77.62
RSC	85.87	77.8	70.8	62.5	98.23	79.04
C-DANN	85.7	75.8	66.78	74.25	97.95	80.9
MMD	88.3	78.2	67.8	73.89	98.3	81.3
CORAL	87.5	77.8	68.89	74.25	98.2	81.32
DRO	88.9	78.5	66.7	73.89	98.2	81.34
Mixup	88.8	78.7	67.8	74.2	97.89	81.48
DANN	89.0	78.7	68.8	74.25	97.8	81.77
ERM	89.11	78.84	71.95	74.35	99.2	82.7

Table 4. Comparing ERM-Inc-Resnet with other methods in DomainBed implemented with inception-Resnet backbone. The algorithms are sorted by their average performance across the five datasets.

Effect of choice of backbone: Table 1 shows the significance of backbone in DG. Across all domains and irrespective of augmentation and other choices, the Inception-Resnet backbone outperforms all other backbones. Zhou *et al*. [47] challenges the common perception that models that perform on ImageNet will learn domain-generalizable features and hence argues for DG specific technique methods. In contrast, we observe that the better performing backbone for DG is the better performing model on the ImageNet IID benchmark and not necessarily the backbone with more parameters.

5.2. Comparing with baselines in TDG

In Table 2 we compare the ERM baselines against other algorithms in DomainBed [11]. The proposed ERM baseline with Inception-Resnet backbone (ERM-Inc-Resnet) not only outperforms other methods on an average but also outperforms the best performing model in every dataset. This consistent improvement across datasets further proves that inception-Resnet should be the preferred backbone for DG tasks. On the PACS dataset, we get a margin of above 5% from the next best performing model. This comparison shows that neural networks trained with a robust backbone, augmentation, and optimizer under TDG setting do not need any additional method to learn domain agnostic features. Furthermore, Table 4 shows the comparison of ERM, with all methods implemented in Inception-Resnet backbone. This table further proves that none of the methods outperform ERM despite having a robust backbone. In fact, some of the methods simply constraint the learning, reducing test performance. This motivates us to think if by solving deficits of neural networks in TDG setting, are we fixing a system that is not broken?

5.3. Benchmarking CWDG

Table 3 shows the performance of different algorithms under DomainBed in the CWDG setting. A one-to-one comparison between TDG results and CWDG is not meaningful, but it is curious to note that despite seeing all domains during training, the models perform worse than when tested on an unseen domain. The drop in performance in the CWDG setting (compared to TDG) suggests fundamental challenges beyond seeing unseen domains. We further observe that the idea of gradient reversal (GRL) holds much merit in the CWDG. It is worth noting that GRL did not give any improvements in TDG, further motivating the need to evaluate DG from varied perspectives.

Backbones	Our model	with RSC	with GRL
Resnet-18	71.295	70.42	74.6
Resnet-50	76.6	72.32	79.3
Inc-Res	79.6	78.71	**86.2**

Table 5. Results for CWDG on PACS dataset

We compare different backbones with GRL and RSC on CWDG in Table 5. The numbers are reported as an average over five runs, wherein every run, we sample a different domain into the test set for each class. RSC brings down the performance in each case. In contrast, employing GRL improves the performance of all three backbones.

We present three analytical experiments to build further insights on how Neural Networks learn with domains shifts. We explore how useful the learning from each domain is while classifying on other domains using pairwise experiments. We further compare dataset shifts with test performance, while adding domains to the training set. In the third experiment, we make a case for going beyond distribution shifts across train-test splits to explain OOD performance.

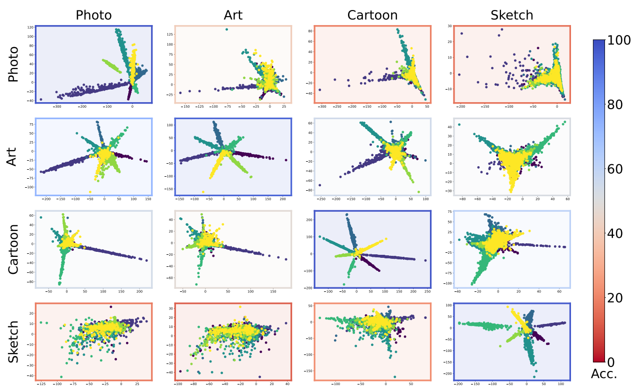


Figure 4. The color of each tile shows the accuracy of the model trained on the domain corresponding to the row name and tested on the domain corresponding to the column name. The plot within each tile, shows the embeddings learned using an additional network with a two-node bottleneck. Embeddings from different classes are represented with different colors.

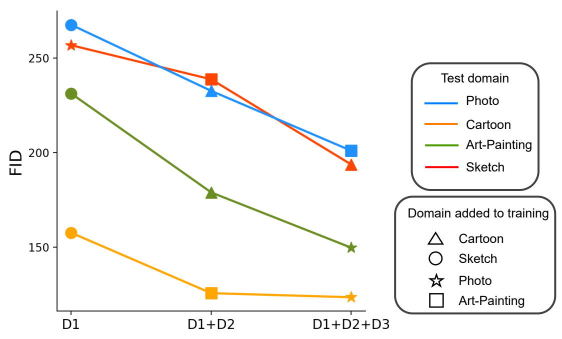


Figure 5. The graph shows the FID distance between the train and test data on adding domains to train data. The graph contrasts to Figure 1, showing that adding domains to train data decreases the FID distance.

6. Analysis and Discussion

6.1. PACS vs PACS

We analyze the interplay of variances within domains in Figure 4. We train the Inception-Resnet [38] backbone (pre-trained on Imagenet) four times, each run corresponding to each of the four domains in the PACS dataset. We keep out a part of the training domain for model selection and evaluation. We evaluate each of the four models on all four domains using the kept out data for the domain it is trained on and all samples from other domains. The accuracy is indicated as the color of each tile in Figure 4.

To further investigate the efficacy of features to discriminate classes in other domains, we train each of the four models with a two-node bottleneck layer before the classification head. The embeddings obtained on the test split of each of the sixteen pairwise experiments are plotted inside the corresponding tiles in Figure 4. The embeddings corresponding to each of the classes are plotted with a different color.

Expectedly, the features learned from a domain are most effective in the same domain (diagonal plots in Figure 4). Also, from the accuracy and the separation in classes, we can see that features learned from a domain like 'Art-Painting' help discriminate samples of almost all four domains. However, features learned from the 'Sketch' domain are not adequate for discriminating samples in any other domain. Also, the accuracy metrics of the pair-wise experiments are almost symmetrical.

6.2. Train-Test distributions shifts

In this section, we explain the correlation between accuracy of models and the corresponding distribution shift (FID distance) between train and test splits. We freeze the test set and observe the performance while varying the training set. For each run, we report the test accuracy and the corresponding FID distance.

The experiments are performed using the PACS dataset. We start with a single domain each in the train and test splits and subsequently keep adding the remaining domains to the train set (until there are all but the test domain in the train data). For instance, we start with the 'Sketch' domain as test and 'Photo' as train. We add 'Art-Painting' and 'Cartoon' to the train data in the subsequent two steps. We compare test accuracy (on 'Sketch') between the three runs. To make a comparison unbiased of the train data size, we randomly sub-sample after adding each domain such that the size of train data is the same for all experiments for each iteration. We perform this experiment with all four domains as test data with a randomly selected order of adding domains.

Figure 1 presents the obtained results. Despite some domains being poor at contributing discriminative features to other domains (as seen in Section 6.1), we find a significant advantage of having multiple domains in the training set. The results clearly show that adding domains to the train data, however different from the test domain, aid domain agnostic feature learning and improves test performance.

Comparing Figure 1 with Figure 5, we can see that adding a new domain to the train data decreases the FID distance between the train and test splits and increases the accuracy of the trained model. The experiment empirically suggests that in the TDG formulation, the accuracy of the model is entirely explained by the distribution shift between the train and test splits (Figure 5).

6.3. Classwise priors

This section demonstrates that dataset shifts alone cannot explain OOD generalization, and class-wise distribution shifts also pose a challenge. We formulate a synthetic setting using two domains ('Art-Painting' and Sketch') from the PACS dataset. For the first model, the training data is created with a domain shift on a specific class. All train samples are chosen from the domain 'Art-Painting' except the class 'Giraffe,' which is sampled from 'Sketch' (as illustrated in Figure 3). We then train a second network using all classes from a single domain ('Art-Painting'). Both models are evaluated on the 'Sketch' domain. We use Frechet Inception Distance (FID) [15] to measure the dataset shift among train-test splits in the two settings.

When the training data for all the classes are picked from a single domain ('Art-Painting'), we get a test performance of 61.67% on the test data ('Sketch'). By replacing a single class ('Giraffe') in training from the 'Sketch' domain, all test images are predicted as Giraffe (the accuracy drops to 13%, the fraction of samples from the class Giraffe). The observation is intuitive as the network fits the easier discriminative variances that differentiate domains and use them to classify.

Unlike in typical TDG setup, this behavior cannot be explained by distribution shifts among the train and test split (Figure 5). The train data is now closer to test data as they share samples from a class. The FID score of train and test data in the first setting is 147.50, and the second setting is 95.96. The performance drops despite the reduction in the dataset shift.

The above observation shows that there are factors other than distribution shifts that determine the OOD generalization of neural networks. The tendency of a network to fit the domain-specific variances which are used to distinguish classes in train data is caused by the priors in train data (different domains see a different subset of classes). The model's ability to escape fitting such prior can be evaluated by extending this synthetic experiment, which precisely motivates the need for CWDG evaluation. In the CWDG setting, different seeds for sampling domains from classes to create test data can evaluate the different variances that a model fits.

7. Conclusion

In this work, we find that a carefully trained ERM outperforms all existing state-of-the-art methods on six standard datasets when evaluated in traditional DG setting. The findings are similar to [11]. However, we disentangle the performance of ERM and discuss the role of backbone, data augmentation, and optimization algorithm. We hope the analysis will help better plan future experiments on DG. We introduce a novel evaluation benchmark called CWDG benchmark, which can be constructed using any existing DG dataset by modifying the train and test split. The benchmark aims to extend the horizon of evaluation of DG in terms of dataset shifts. Contrasting TDG with CWDG, we explain the two previously counter-intuitive observations: (a) none of the inventive methods improve DG in neural networks, and (b) despite seeing all domains in train data, networks fail to perform in CWDG setting. Unlike TDG, in the CWDG framework, the network, if left to its own devices, will learn domain-specific features. Through a detailed analysis, we show that the performance of networks on test data cannot be fully explained by the dataset shift between train and test data but also by the priors introduced by distribution shift across classes. We thoroughly evaluate the proposed benchmark and find that, unlike TDG, classical methods like reverse gradients aid performance when added on various backbones in the CWDG setup.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 3, 4, 5

[2] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017. 3

[3] Francesco Cappio Borlino, Antonio D'Innocente, and Tatiana Tommasi. Rethinking domain generalization baselines. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9227–9233. IEEE, 2021. 2, 3

[4] P. Chaudhari, A. Choromanska, S. Soatto, and Y. LeCun. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016. 2

[5] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1657–1664, 2013. 3, 4, 5

[6] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 2, 3, 5

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016. 5

[8] Xavier Gastaldi. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 3

[9] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020. 1, 2, 3

[10] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015. 3, 4, 5

[11] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020. 2, 3, 4, 5, 7, 9

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 5

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 5

[14] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018. 1

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 4, 9

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 5

[17] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. *ECCV*, 2020. 2, 3, 5

[18] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012. 2, 3

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 5

[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. 3, 4, 5, 12

[21] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 5

[22] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018. 5

[23] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018. 5

[24] Massimiliano Mancini, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Best sources forward: domain generalization through source-specific nets. In *2018 25th IEEE international conference on image processing (ICIP)*, pages 1353–1357. IEEE, 2018. 3

[25] Udit Maniyar, Aniket Anand Deshmukh, Urun Dogan, Vineeth N Balasubramanian, et al. Zero shot domain generalization. *arXiv preprint arXiv:2008.07443*, 2020. 3

[26] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018. 1, 3

[27] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012. 3, 4

[28] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18. PMLR, 2013. 2

[29] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1406–1415, 2019. 3, 4, 5

[30] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *International Conference on Machine Learning*, pages 7728–7738. PMLR, 2020. 3

[31] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018. 3

[32] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 3

[33] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019. 5

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[35] Sarath Sivaprasad, Ankur Singh, Naresh Manwani, and Vineet Gandhi. The curious case of convex neural networks. *arXiv preprint arXiv:2006.05103*, 2020. 3

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3

[37] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016. 5

[38] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 5, 8, 12

[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 4

[40] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *arXiv preprint arXiv:2007.00644*, 2020. 1, 3

[41] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for

unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5018–5027, 2017. 3, 4, 5

[42] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013. 3

[43] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021. 3

[44] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017. 3, 6

[45] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6502–6509, 2020. 3, 5

[46] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 1, 3

[47] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *arXiv preprint arXiv:2103.02503*, 2021. 3, 7

[48] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *ICLR*, 2021. 3

A. Appendix

In this document we present the supplementary materials to support the main text. Here we present our two discussions. First section shows the results of different seeds for train test split in CWDG. In the subsequent section we discuss the challenge in model selection in DG.

B. Different seeds for CWDG

In this section we present results for the different train test splits in CWDG setting of the PACS dataset. We observe comparable performance across the different seeds of CWDG(Table 6). Each column in the table correspond to one run and the last row in each column shows the accuracy of that run. That is, each row correspond to a class and each element shows the domain kept out for the particular class. The first column shows the split used in the main text. The accuracies shows that as long as the domains are evenly spread such that there is a clear prior in the train split the performance of neural network stays signficantly below the TDG setting.

C. Model selection in DG

In the TDG setting, model selection is traditionally done using the test data (the entirety of the unseen domain). However, the formulation of DG implies that the test domain is not known during training. Hence, using an oracle for model selection is not appropriate in the strict sense. To further analyze the challenge in model selection in the TDG setting, we contrast the best and worst models (based on their test performance), picked after the training saturates (the loss curve saturates).

We run Inception-Resnet-ERM [38] in the TDG setting of the PACS [20] dataset. For each run, we keep out one domain for test. We also create a validation set by randomly sampling 5% data from each of the three training domains and monitor the model performance on this validation set after each training step. We select the best-performing model on this validation set. We report the accuracy of this model on the test domain (A_{sel}). We also report the performance of the model selected by an oracle for each run as A_{max} (directly using test data for validation). A_{min} is the accuracy of the model that gives the lowest test performance once the training accuracy saturates. We report A_{max}, A_{min} and A_{sel} with all four folds of PACS data.

From Table 7 we can see that model selection in the TDG setting is not trivial. Performance on a validation set (sampled from train split) does not guarantee performance on the unseen domain. To evaluate the possible model selection for the CWDG setting, we run Inception-Resnet-ERM in the CWDG setting of the PACS dataset. The A_{sel}, A_{max} and A_{min} value for the CWDG setting are 79.1%, 79.4% and 66.3% respectively. The difference between the A_{max} and A_{min} in the CWDG setting is higher than the average difference in the TDG setting.

Classes	**Domain**	**Domain**	**Domain**	**Domain**	**Domain**
Guitar	Photo	Art	Cartoon	Sketch	Art
Person	Photo	Cartoon	Art	Photo	Cartoon
Horse	Cartoon	Sketch	Cartoon	Art	Photo
Elephant	Sketch	Photo	Art	Sketch	Art
Dog	Photo	Art	Sketch	Cartoon	Cartoon
Giraffe	Art	Photo	Cartoon	Art	Photo
House	Cartoon	Cartoon	Photo	Sketch	Sketch
Accuracy	79.6	79.38	78.81	79.12	79.18

Table 6. performance of NN on different train test splits in CWDG setting. Last row of each column shows results of one run and each row of the column corresponds to the domain kept out for the corresponding class.

Train domains	**Test domain**	A_{min}	A_{max}	A_{sel}
SAC	P	94.01	96.4	96.3
PSC	A	78.7	86.3	84
PAS	C	81.1	87.8	83
PAC	S	64.96	85.09	83.2

Table 7. Model selection in TDG