
Domain Generalization using Causal Matching

Divyat Mahajan
Microsoft Research
Bangalore, India
divyatmahajan@gmail.com

Shruti Tople
Microsoft Research
Cambridge, United Kingdom
shtople@microsoft.com

Amit Sharma
Microsoft Research
Bangalore, India
amshar@microsoft.com

Abstract

Learning invariant representations has been proposed as a key technique for addressing the domain generalization problem. However, the question of identifying the right conditions for invariance remains unanswered. In this work, we propose a causal interpretation of domain generalization that defines domains as interventions under a data-generating process. Based on a general causal model for data from multiple domains, we show that prior methods for learning an invariant representation optimize for an incorrect objective. We highlight an alternative condition: inputs across domains should have the same representation if they are derived from the same *base object*. Inputs that share the same base object may be available through data augmentation or in some specific contexts, but base object information is not always available. Hence we propose an iterative algorithm called MatchDG that approximates base object similarity by using a contrastive loss formulation adapted for multiple domains. We then match inputs that are similar under the resultant representation to build an invariant classifier. We evaluate our matching-based methods on rotated MNIST, Fashion-MNIST, PACS and Chest X-ray datasets and find that they outperform prior work on out-of-domain accuracy. In particular, top-10 matches from MatchDG have over 50% overlap with ground-truth matches in MNIST and Fashion-MNIST. Code repository can be accessed here: <https://github.com/microsoft/robustdg>

1 Introduction

Machine learning models are often deployed in applications where the test (inference time) data distributions differ from their training dataset. For example, a model trained on data from one hospital is used for prediction at other hospitals or an image classification model is deployed on pictures with slightly different orientations. These applications require that a model generalizes well to new data distributions in addition to the training distribution, unlike standard machine learning tasks that focus on minimizing same-distribution error. One approach for generalizing to unseen domains is to learn representations that remain invariant across domains, by enforcing that their distributions stay the same across domains, either marginally [1, 2, 3] or conditional on the class label [4, 5]. Other methods frame invariance in terms of accuracy of the classifier on different domains [6]. However, it is unclear how to evaluate these different invariance conditions for their applicability, e.g., under class imbalance or other differences between domains.

To this end, we introduce a formal causal framework for the domain generalization task that allows an easy and coherent characterization of the invariance conditions. Specifically, we construct a model for the data generation process that assumes each input is constructed from a mix of inherent (*causal*) and domain-dependent (*non-causal*) features. Building on prior work [7, 8], we consider domain as a special intervention that changes the non-causal features of an input, and posit that an ideal classifier should be based only on the causal features. Using this model, we show that methods based on enforcing same distribution of representations across domains are inconsistent, confirming

claims from past work [6, 9]. Furthermore, we show that methods that enforce the same distribution conditional on class label are also insufficient, unless additional assumptions are made.

The same causal model also provides us with the right condition that an invariant representation should satisfy. Our invariance condition depends on a special *object* variable that defines a collection of inputs that share the same causal features. For example, photos of the same person from different viewpoints correspond to a single object, and so do augmentations of an image in different rotations, color or background. We show that the correct invariance condition is that the learnt representation be the same for each object across domains. When the object variable is available (e.g., in self-collected data or by dataset augmentation), we propose a *perfect-match* regularizer for domain generalization that minimizes the distance between representations of the same object across domains.

In practice, however, the underlying objects are not always known or replicated across domains. We therefore propose an approximation of the above invariant condition that uses class labels as a proxy, under the assumption that inputs from the same class have more similar causal features than those from different classes. Our algorithm, MatchDG has two phases. First, it constructs a representation such that inputs sharing the same causal features are closer to one another, and matches pairs of inputs that are most similar. Second, it uses these learnt matches as a regularizer when building the classifier. In datasets with data augmentations, we extend MatchDG to also use the object matches obtained from pairs of original and augmented images (MatchDGHybrid).

We evaluate our matching-based methods on rotated MNIST and Fashion-MNIST, PACS and Chest X-ray datasets. On all datasets, MatchDG and MatchDGHybrid outperform state-of-the-art methods for out-of-domain accuracy. On the rotated MNIST and Fashion-MNIST datasets where the ground-truth objects are known, MatchDG learns to makes the representation more similar to their ground-truth matches (about 50% overlap for top-10 matches), even though the method does not have access to them. Our superior results with simple matching methods show the importance of enforcing the correct invariance condition.

Contributions. To summarize, our contributions include:

- **Invariance Condition.** We propose an object-invariant condition for learning a common representation for domain generalization and justify its correctness compared to prior approaches.
- **MatchDG Algorithm.** When object information is not available, we provide a novel two-phase algorithm using contrastive loss formulation that learns a matching function between inputs and approximates object-based matches.

2 A Causal View of Domain Generalization

Consider a classification task where the learning algorithm has access to i.i.d. data from m domains, $\{(d_i, \mathbf{x}_i, y_i)\}_{i=1}^n \sim (D_m, \mathcal{X}, \mathcal{Y})^n$ where $d_i \in D_m$ and $D_m \subset \mathcal{D}$ is a set of m domains. Each training input (d, \mathbf{x}, y) is sampled from an unknown distribution $\mathcal{P}_m(D, X, Y)$. The domain generalization task is to learn a single classifier that generalizes well to unseen domains $d' \notin D_m$ and to new data from the same domains [10]. The optimum classifier can be written as: $f^* = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(d, \mathbf{x}, y) \sim \mathcal{P}} [l(y^{(d)}, f(\mathbf{x}^{(d)}))]$, where $(d, \mathbf{x}, y) \sim \mathcal{P}$ over $(\mathcal{D}, \mathcal{X}, \mathcal{Y})$.

However, we only have access to D_m domains during training. The plug-in estimator replaces \mathcal{P} by \mathcal{P}_m following the Empirical Risk Minimization (ERM) principle.

$$f_{ERM} = \arg \min_{f \in \mathcal{F}} \hat{\mathbb{E}}_{S \sim \mathcal{P}_m} [l(y^{(d)}, f(\mathbf{x}^{(d)}))] \quad (1)$$

where $S = (d, \mathbf{x}, y)^n$ is a training data of size $n = \sum_{d \in D_m} n_d$. Proof for the next Proposition is in Suppl. A.3.

Proposition 1. *The ERM estimator from (1) learns the true f^* , as the set of training domains $D_m = \mathcal{D}$ and number of data samples $n \rightarrow \infty$.*

However, when $D_m \subset \mathcal{D}$, ERM can overfit to the training domains. To avoid overfitting, a popular technique is to learn a common feature representation across all training domains that can be subsequently used to train a classifier [5, 11, 12]. Due to a lack of formal definition of the problem, different learning objectives have been proposed, such as minimizing the distributional distance

between learnt feature representations from different domains [1, 2, 3] or minimizing the distance between class-conditional feature representations [4, 5, 13]. We provide a causal framework that provides a correct invariant condition needed for domain generalization.

2.1 Data-generating process

Consider a task of classifying the type of item or screening an image for a medical condition. To build a classifier, a train set is generated by taking photos. Due to human variability or by design (using data augmentation), the data generation process yields variety of images for each class, sometimes multiple images for the *same object*. In this example, the domain generalization task is to build a classifier that is robust to different views of any new object.

Figure 1a shows a structural causal model (SCM) that describes the data-generating process. Here each view can be considered as a different *domain* D , the label for item type or medical condition as the class Y , and the image pixels as the features X . Photos of the same item or the same person correspond to a common *object* variable [8], denoted by O . To create an image, the data-generating process first samples an object and view (domain) that may be correlated to each other (shown with dashed arrows). The pixels in the photo are caused by both the object and the view, as shown by the two incoming arrows to X . The object also corresponds to high-level causal features X_C that are common to any image of the same object, which in turn are used by humans to label the class Y .

The above example is typical of a domain generalization problem; a general SCM is shown in Figure 1b. In general, the underlying *object* for each input $\mathbf{x}_i^{(d)}$ may not be observed. Analogous to the causal features X_C , we introduce a node for domain-dependent high-level features of the object X_A . Changing the domain can be seen as an intervention: for each observed $\mathbf{x}_i^{(d)}$, there are a set of (possibly unobserved) counterfactual inputs $\mathbf{x}_j^{(d')}$ where $d \neq d'$, such that all correspond to the same object (and thus share the same X_C). For completeness, we also show the true unobserved label of the object which led to its generation as Y_{true} (additional motivation for the causal graph is in Suppl. A.1). Like the object O , Y may be correlated with the domain D . Extending the model in [8], we allow that objects can be correlated with the domain conditioned on Y_{true} . As we shall see, considering the relationship of the *Object* node becomes the key piece for developing the invariant condition. We can write the following non-parametric equations corresponding to the SCM.

$$\begin{aligned} o &:= g_o(y_{true}, \epsilon_o) & \mathbf{x}_c &= g_{xc}(o) \\ \mathbf{x}_a &:= g_{xa}(d, o, \epsilon_{xa}) & \mathbf{x} &:= g_x(\mathbf{x}_c, \mathbf{x}_a, \epsilon_x) \\ & & y &:= h(\mathbf{x}_c, \epsilon_y) \end{aligned}$$

where g_o, g_{xc}, g_{xa}, g_x and h are general non-parametric functions. The error ϵ_o is correlated with domain d whereas $\epsilon_{xa}, \epsilon_x$ and ϵ_y are mutually independent error terms that are independent of all other variables. Thus, noise in the class label is independent of domain. Since x_c is common to all inputs of the same object, g_{xc} is a deterministic function of o . In addition to these equations, the SCM provides conditional-independence conditions that all data distributions \mathcal{P} must satisfy, through the concept of d-separation A.2 and the perfect map assumption [14].

Definition 1. d-separation [14]: Let A, B, C be the three non-intersecting subsets of nodes in a causal graph \mathcal{G} . For any path between two nodes, a collider is a node where arrows of the path meet head-to-head. A path from A to B is said to be blocked by C if either a non-collider on the path is in C , or there is a collider on the path and neither the collider nor its descendants are in C .

If all paths from A to B are blocked, then A is d-separated from B by C : $dsep(A, B, C) \Rightarrow A \perp\!\!\!\perp B | C$.

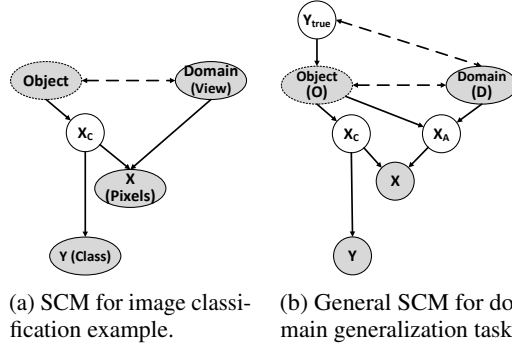


Figure 1: Structural causal models for the data-generating process. Observed variables are shaded; dashed arrows denote correlated nodes. *Object* has a dotted outline since it may not be observed.

2.2 Identifying the invariance condition

From Figure 1b, X_C is the node that causes Y . Further, by d-separation, the class label is independent of domain conditioned on X_C , $Y \perp\!\!\!\perp D | X_C$. Thus our goal is to learn y as $h(\mathbf{x}_c)$ where $h : \mathcal{C} \rightarrow \mathcal{Y}$. The ideal loss-minimizing function f^* can be rewritten as (assuming \mathbf{x}_c is known):

$$\arg \min_f \mathbb{E}_{(d, \mathbf{x}, y)} l(y, f(\mathbf{x})) = \arg \min_h \mathbb{E}[l(y, h(\mathbf{x}_c))] \quad (2)$$

Since X_C is unobserved, this implies that we need to learn it too through a representation function $\Phi : \mathcal{X} \rightarrow \mathcal{C}$. Together, $h(\Phi(x))$ leads to the desired classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$.

Conditional independencies from the SCM identify the correct learning goal for learning X_C . By the d-separation criterion, we see that X_C satisfies two conditions: **1)** $X_C \perp\!\!\!\perp D | O$, **2)** $X_C \not\perp\!\!\!\perp O$; where O refers to the object variable and D refers to a domain. The first is an invariance condition: X_C does not change with different domains for the same object. To enforce this, we stipulate that the average pairwise distance between $\Phi(x)$ for inputs across domains for the same object is 0, $\sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0$. Here $\Omega : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$ is a *matching* function that is 1 for pairs of inputs across domains corresponding to the same object, and 0 otherwise.

However, just the above invariance will not work: we need the representation to be informative of the object O too (otherwise even a constant Φ minimizes the above loss). Therefore, the second condition stipulates that X_C should be informative of the object, and hence about Y . We add the standard classification loss, leading to constrained optimization,

$$f_{\text{perfectmatch}} = \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) \text{ s.t. } \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0 \quad (3)$$

where $L_d(h(\Phi(X)), Y) = \sum_{i=1}^{n_d} l(h(\Phi(\mathbf{x}_i^{(d)})), y_i^{(d)})$. Here f represents the composition $h \circ \Phi$. For example, a neural network with $\Phi(x)$ as its r th layer, and h being the rest of the layers. The proof for the next theorem is in Suppl. A.4.

Theorem 1. *For a finite number of domains m , as the number of examples in each domain $n_d \rightarrow \infty$,*

- 1. The set of representations that satisfy the condition $\sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0$ contains the optimal $\Phi(\mathbf{x}) = X_C$ that minimizes the domain generalization loss in (2).*
- 2. Assuming that $P(X_a | O, D) < 1$ for every high-level feature X_a that is directly caused by domain, and for P -admissible loss functions [15] whose minimization is conditional expectation (e.g., ℓ_2 or cross-entropy), a loss-minimizing classifier for the following loss is the true function f^* , for some value of λ .*

$$f_{\text{perfectmatch}} = \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \lambda \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \quad (4)$$

While Theorem 1 shows the consistency of the PerfectMatch condition, it does not identify X_C uniquely since there can be multiple X_C (e.g., linear transformations) that satisfy the condition and are equally good for the predictive task. Identification of causal features is a non-trivial problem [16], especially when features are latent, that we leave for future work.

2.3 Past work: Learning common representation

Using our model, we now compare the proposed invariance condition to three main representation learning objectives: domain-invariant, class-conditional domain-invariant, and invariant-optimal-classifier representations.

Domain-invariant representations. The goal is to learn a representation Φ such that its distribution $P(\Phi(\mathbf{x}))$ is the same across domains [1, 2, 3], assuming that the ideal representation (X_C) is independent of domain. However, using d-separation on the SCM from Figure 1b, $Y \perp\!\!\!\perp D$ is not sufficient since O blocks the path between X_C and D . While [9, 17] argue that this condition fails when Y is correlated with D , our analysis shows that domain-invariant methods require a stronger condition that both class label and actual objects sampled be independent of domain.

Class-conditional domain-invariant Φ . As a better objective, class-conditional methods by [4, 5, 13, 7] aim to obtain representations such that $P(\Phi(\mathbf{x}^{(d)})|Y)$ is the same across domains, through minimizing distribution divergence measures. However, even in the ideal case where we observe Y_{true} , d-separation on the SCM reveals that $X_C \not\perp\!\!\!\perp D|Y_{true}$ due to a path through O . Thus, having the same distribution per class is not consistent with properties of X_C .

The above discussion indicates that prior representation learning methods optimize an incorrect objective: even with infinite data across domains, they will not learn the true X_C . We state it formally as a corollary with its proof in the section A.5.

Corollary 1. *The conditions enforced by domain-invariant ($\Phi(x) \perp\!\!\!\perp D$) or class-conditional domain-invariant ($\Phi(x) \perp\!\!\!\perp D|Y$) methods are not satisfied by the causal representation X_C . Thus, without additional assumptions, the set of representations that satisfy any of these conditions does not contain X_C , even as $n \rightarrow \infty$.*

Invariant-optimal-classifier Φ . Recently, [6, 18] assume that $P(Y|X_C)$ remains the same across domains and thus a single classifier over the optimal $\Phi(\mathbf{x})$ should be optimal for all domains. That is, $Y \perp\!\!\!\perp D|\Phi(\mathbf{x})$ which is also satisfied by X_C in the SCM. However, enforcing this condition is difficult in practice and thus the resultant method is restricted to a linear classifier over Φ . To this end, we propose a simple condition that supports any architecture and is consistent with the SCM.

3 MatchDG: Proposed algorithm

When object information is available, Eq. (4) provides a loss objective to build a classifier using causal features. However, object information is not always available, and in many datasets there may not be a perfect ‘‘counterfactual’’ match based on same object across domains. Therefore, we provide an invariance condition using only (X, Y) that is consistent with the conditional independencies of X_C and propose a two-phase contrastive learning method to learn such an X_C .

3.1 Match loss consistent with properties of X_C

The object-invariant condition from Section 2.2 can be interpreted as matching pairs of inputs from different domains that share the same X_C . To approximate it, our goal is to learn a matching $\Omega : \mathcal{X} \times \mathcal{X} \rightarrow \{0, 1\}$ such that pairs having $\Omega(\mathbf{x}, \mathbf{x}') = 1$ have low difference in \mathbf{x}_c and \mathbf{x}'_c . One simple way is to use the class label and match every input to a random input from the same class. Assuming X_C of inputs from the same class is bounded by δ , we show that this simple matching strategy does include X_C as a possible solution. We obtain the following *random match* regularizer.

$$f_{\text{randommatch}} = \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \lambda \sum_{\Omega_Y(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \quad (5)$$

where Ω_Y randomly matches pairs from the same class.

Theorem 2. *Assume training domains such that for any two same-class inputs $\mathbf{x}_j^{(d)}$ and $\mathbf{x}_k^{(d')}$ from domains d and d' , $\text{dist}(x_{a,j}^{(d)}, x_{a,k}^{(d')}) \geq \delta_a$ where x_a is any high-level feature that is caused directly by domain. Further, assume that the distance over X_C between same-class inputs from different domains is bounded: $\text{dist}(x_{c,j}^{(d)}, x_{c,k}^{(d')}) \leq \delta_c$ and $\delta_c < \delta_a$ ($\delta_c, \delta_a \in \mathbb{R}^+$). Then for some λ , a loss-minimizing classifier for the loss from (5) is the true function f^* , given a P -admissible loss function and a finite number of domains m with $n_d \rightarrow \infty$ in each domain.*

The proof substitutes X_C in the match condition and uses Lagrange multipliers, detailed in Suppl. A.6. Compared to class-conditional domain-invariant condition, a key distinction is that we enforce the difference in *individual* representations for same-class inputs to be low, not just to have the same distribution. That is, we minimize the variance of the class-conditional distribution. A variation of this loss is used as a contrastive regularizer in [19, 20]. However, Theorem 2 cannot guarantee that X_C will be returned by the optimization. The key parameter is δ_c . If a dataset has low δ_c , then there is a high chance of learning a good representation that is close to X_C (if $\delta_c = 0$, we obtain perfect match). But as δ_c increases, the matching condition loses any discriminative power. Hence we need to learn a matching Ω such that δ_c is minimized.

Algorithm 1 MatchDG

```

1: In: Dataset  $(d_i, x_i, y_i)_{i=1}^n$  from  $m$  domains,  $\tau, t$ 
2: Out: Function  $f : \mathcal{X} \rightarrow \mathcal{Y}$ 
3: Create random match pairs  $\Omega_Y$ .
4: Build a  $p * q$  data matrix  $\mathcal{M}$ .
5: while notconverged do ▷ Phase I
6:   for  $batch \sim \mathcal{M}$  do
7:     Minimize contrastive loss (6).
8:   if epoch %  $t == 0$  then
9:     Update match pairs using  $\Phi_{epoch}$ .
10: Compute matching based on  $\Phi$ . ▷ Phase 2.
11: Minimize the loss (5) to obtain  $f$ .

```

3.2 Two-phase method with learnt matches

To learn such a Ω , we use unsupervised contrastive learning from [21, 22] and adapt it to domain generalization in our MatchDG algorithm. MatchDG relies on the property that two inputs from the same class have more similar causal features than inputs from different classes. We optimize a contrastive representation learning loss that minimizes distance between same-class inputs from different domains in comparison to inputs from different classes across domains. This technique is likely to have lower δ_c than a simple class-based random match. Adapting the contrastive loss for a single domain [21], we consider *positive* matches as two inputs with the same class but different domains, and *negative* matches as pairs with different classes. For every positive match pair $(\mathbf{x}_j, \mathbf{x}_k)$, we propose a loss where τ is a hyperparameter, B is the batch size, and $\text{sim}(\mathbf{a}, \mathbf{b}) = \Phi(\mathbf{x}_a)^T \Phi(\mathbf{x}_b) / \|\Phi(\mathbf{x}_a)\| \|\Phi(\mathbf{x}_b)\|$ is the cosine similarity.

$$l(\mathbf{x}_j, \mathbf{x}_k) = -\log \frac{e^{\text{sim}(j,k)/\tau}}{e^{\text{sim}(j,k)/\tau} + \sum_{i=0, y_i \neq y_j}^B e^{\text{sim}(j,i)/\tau}} \quad (6)$$

Our key insight is to update matches during training. Instead of using the standard contrastive loss where matches are pre-decided, we start training with a random match based on classes. After every t epochs, we update the matches based on nearest same-class pairs in representation space, and iterate until convergence. Under assumption that inputs of same class are closer in causal features, optimizing for the initial random matches should lead to a representation wherein similarity correlates more to similarity in causal features. This completes Phase I of the algorithm. In Phase 2, we utilize the final representation to compute a new match function based on closest same-class pairs and then apply (5) to obtain a classifier regularized on those matches. In Suppl. C.6, we compare the gains due to the proposed iterative matching versus standard contrastive training.

To implement MatchDG we build a $p \times q$ data matrix containing $q - 1$ positive matches for each input and then sample mini-batches from this matrix. The last layer of the contrastive loss network is considered as the learnt representation. Algorithm 1 provides an overview; details are in Suppl. B.1. We implement MatchDG as a 2-phase method, unlike previous methods [19, 20] that employed class-based contrastive loss as a regularizer with ERM. This is to avoid the classification loss interfering with the goal of learning an invariant representation across domains (e.g., in datasets where one of the domains has many more samples than others). Therefore, we first learn the match function using only the contrastive loss. Our results in Suppl. C.4 show that the two-phase method provides better overlap with ground-truth perfect matches than optimizing classification and matching simultaneously.

4 Evaluation

We evaluate MatchDG on out-of-domain accuracy for two simulated benchmarks from CSD by [23](Rotated MNIST and Fashion-MNIST) where the same objects are rotated across domains, and on real world datasets (PACS) [24] and (ChestXRay) [25]. In addition, using the simulated datasets, we inspect the quality of matches learnt by MatchDG by comparing them to ground-truth object-based matches. For the real world datasets, we also implement MatchDGHybrid that uses augmentations commonly done while training neural networks. We compare to 1) ERM: Standard empirical risk minimization, 2) ERM-RandMatch that implements the loss from Eq. (5), 3) other state-of-the-art

methods for each dataset. For all matching-based methods, we use the cross-entropy loss for L_d and ℓ_2 distance for dist in Eq.(4), (5). Details of implementation and the datasets are in Suppl. B.1. All the numbers are averaged over 3 runs with standard deviation in brackets.

Rotated MNIST & Fashion-MNIST. The datasets contain rotations of grayscale MNIST handwritten digits and fashion article images from 0° to 90° with an interval of 15° [26], where each rotation angle represents a domain and the task is to predict the class label. Since different domains’ images are generated from the same base image (object), there exist perfect matches across domains. Following CSD, we report accuracy on 0° and 90° together as the test domain and the rest as the train domains; since these test angles, being extreme, are the hardest to generalize to.

PACS. This dataset contains total 9991 images from four domains: Photos (P), Art painting (A), Cartoon (C) and Sketch (S). The task is to classify objects over 7 classes. Following [20], we train 4 models with each domain as the target using Resnet-18, Resnet-50 and Alexnet.

Chest X-ray. We evaluate on a harder real-world dataset. We use Chest X-rays images from three different sources: NIH [25], ChexPert [27] and RSNA [28]. The task is to detect whether the image corresponds to a patient with Pneumonia (1) or not (0). For ease of interpretation, we balance the data such that there are equal number of images per class in each domain. To create spurious correlation, all images of class 0 in the training domains (NIH, ChexPert) are translated vertically downwards; no such translation is done for the test domain (RSNA).

Model Selection. While using a validation set from the test domain may improve classification accuracy, it goes against the problem motivation of generalization to unseen domains. Hence, we use only data from source domains to construct a validation set (except when explicitly mentioned in Table 3, to compare to past methods that use test domain validation).

4.1 Results: Rotated MNIST (rotMNIST) and Fashion MNIST (rotFashionMNIST)

Table 1 shows classification accuracy on rotMNIST and rotFashionMNIST for test domains 0° & 90° using Resnet-18 model. On both datasets, MatchDG *outperforms* all baselines. The last column shows the accuracy for an oracle method, ERM-PerfMatch that has access to ground-truth perfect matches across domains. MatchDG’s accuracy lies between ERM-RandMatch and ERM-PerfMatch, indicating the benefit of learning a matching function. As the number of training domains decrease, the gap between MatchDG and baselines is highlighted: with 3 source domains for rotFashionMNIST, MatchDG achieves accuracy of 45.6% whereas the next best method ERM-RandMatch achieves 38.8%.

We evaluate on a simpler 2-layer LeNet [19], and the model from [29] to compare MatchDG to prior works [30, 3, 10, 31]. Results are in Suppl. C.1, C.2.

Why MatchDG works? We compare the matches returned by MatchDG Phase I (on Resnet-18 network) to the ground-truth perfect matches and find that it has significantly higher overlap than matching based on ERM loss (Table 2). We report three metrics on the representation learnt: percentage of MatchDG matches that are perfect matches, %-age of inputs for which the perfect match is within the top-10 ranked MatchDG matches, and mean rank of perfect matches measured by distance over the MatchDG representation.

On all three metrics, MatchDG finds a representation whose matches are more consistent with ground-truth perfect matches. For both rotMNIST and rotFashionMNIST datasets, about 50% of the inputs have their perfect match within top-10 ranked matches based on the representation learnt by MatchDG Phase I. About 25% of all matches learnt by MatchDG are perfect matches. For comparison, we also show metrics for an (oracle) MatchDG method that is initialized with perfect matches: it achieves better overall and Top-10 values. Similar results for MatchDG Phase 2 are in Suppl. C.4. Mean rank for rotFashionMNIST may be higher because of the larger sample size 10,000 per domain; metrics for training with 2000 samples are in Suppl. C.5. Finally, to see how the overlap with perfect matches affects accuracy, we simulate random matches with 25%, 50% and 75% overlap with perfect matches (Suppl. Table C.3). Accuracy increases as the fraction of perfect matches increase, indicating the importance of capturing good matches.

The case with zero training error. Since neural networks often achieve zero training error, we also evaluate the effectiveness of the MatchDG regularization under this regime. Fig. 2 shows the matching loss term as training proceeds for rotMNIST and rotFashionMNIST. Even after the model achieves

Table 1: Accuracy for Rotated MNIST & Fashion-MNIST datasets on target domains of 0° and 90° . Accuracy for CSD [23], MASF [20], IRM [6] are reproduced from their code.

Dataset	Source	ERM	MASF	CSD	IRM	RandMatch	MatchDG	PerfMatch (Oracle)
Rotated MNIST	15, 30, 45, 60, 75	93.9 (0.67)	93.2 (0.2)	94.7 (0.2)	94.2 (0.32)	95.9 (0.18)	96.1 (0.34)	97.5 (0.17)
	30, 45, 60	77.9 (2.44)	69.4 (1.32)	79.2 (2.47)	77.6 (1.68)	81.4 (0.77)	86.3 (1.14)	92.0 (0.83)
	30, 45	64.6 (3.23)	60.8 (1.53)	68.7 (1.01)	63.1 (3.14)	68.4 (1.78)	74.3 (2.47)	81.7 (2.79)
Rotated Fashion MNIST	15, 30, 45, 60, 75	78.6 (1.17)	72.4 (2.9)	78.0 (1.5)	79.6 (1.82)	79.4 (0.81)	82.8 (0.27)	86.2 (0.69)
	30, 45, 60	33.7 (2.24)	25.7 (1.73)	37.2 (1.15)	35.5 (1.51)	38.8 (2.28)	45.6 (1.74)	55.3 (1.54)
	30, 45	22.1 (2.36)	20.8 (1.26)	24.9 (1.78)	24.4 (1.01)	25.1 (1.89)	34.9 (1.56)	41.4 (1.58)

Table 2: Overlap with perfect matches. top-10 overlap and the mean rank for perfect matches for MatchDG and ERM over all training domains. Lower is better for mean rank.

Dataset	Method	Overlap (%)	Top 10 Overlap (%)	Mean Rank
MNIST	ERM	18.9 (1.01)	52.4 (1.91)	25.1 (1.43)
	MatchDG (Default)	35.1 (5.23)	69.6 (5.97)	14.3 (4.16)
	MatchDG (PerfMatch)	47.6 (5.61)	81.5 (4.70)	8.2 (3.17)
Fashion MNIST	ERM	3.1 (0.20)	14.4 (0.68)	190.6 (7.92)
	MatchDG (Default)	23.9 (2.61)	50.1 (3.29)	79.7 (9.91)
	MatchDG (PerfMatch)	54.7 (4.38)	82.5 (3.07)	15.5 (3.54)

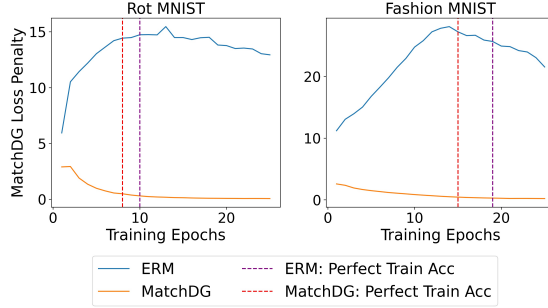
zero training error, we see that plain ERM objective is unable to minimize the matching loss (and thus MatchDG penalty is needed). This is because MatchDG regularization depends on comparing the (last layer) representations, and zero training error does not mean that the representations within each class are the same. In contrast, regularizations that are based on comparing loss between training domains such as the IRM penalty [6] can be satisfied by plain ERM as the training error goes to zero (Fig. 2b); similar to Fig. (5) from [32] where ERM can minimize IRM penalty on Colored MNIST.

4.2 PACS dataset and Chest X-rays datasets

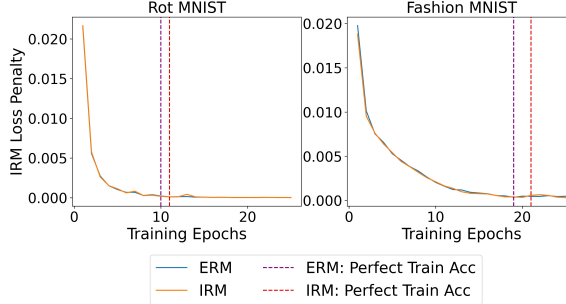
PACS. On the PACS dataset, our methods are competitive to state-of-the-art results averaged over all domains (Table 3). Given that training procedures often involve data augmentations, we use the augmentations as a separate source of “perfect” matches that are added to learnt matches from MatchDG. The resultant method, MatchDGHybrid has the highest average accuracy across domains, except compared to [33]. However it is not a fair comparison since Asadi et al. use additional style transfer data from Behance BAM! dataset during training. It is also unclear whether they used source domain or test domain data for model selection, which is why we include them in a separate category in the table. If the validation mechanism used by them includes data from the target domain during validation, then MatchDG (Test), MDGHybrid (Test) obtain better accuracy than them.

For comparison with G2DM, we compute average accuracy using the test domain as validation set, where MatchDG obtains a higher accuracy. Finally, we implement MatchDG on Resnet50 model used by the ERM in DomainBed. We find that adding MatchDG loss regularization improves the accuracy of DomainBed, from 85.7 to 87.6 with MatchDGHybrid. Also, MatchDGHybrid performs better than the prior approaches using Resnet50 architecture.

Suppl. D.1 gives the results with AlexNet network and comparison to prior work [13, 34, 24, 35, 36] and a t-SNE plot (Figure 4) to show the quality of representation learnt by MatchDG.



(a) MatchDG Penalty during the training process



(b) IRM Penalty during the training process

Figure 2: MatchDG regularization penalty is not trivially minimized even as the training error goes to zero.

Chest X-rays. When evaluated on the source domains, methods like ERM and IRM obtain higher accuracy than matching-based methods. However, on the test domain RSNA, MatchDGHybrid obtains highest classification accuracy (7 %-age points above ERM), followed by CSD and MatchDG (Table 4, last column). The models relying on spurious correlations could get high accuracy on the source domains, however, such models would not perform well on the target domain as no such correlation is present in the target domain. This indicates that approaches like ERM, IRM captured spurious correlations more than MDGHybrid, MatchDG and CSD. Details are in Suppl. E.

5 Related Work

There are four main approaches for the domain generalization task: learning a common representation, dataset augmentation, meta-learning and sharing common parameters.

Learning common representation. To learn a generalizable classifier, several methods enforce same distribution of $\Phi(\mathbf{x})$ across domains marginally or conditional on class label, using divergence measures such as maximum mean discrepancy [1, 4], adversarial training with a domain discriminator [2, 3, 13], use discriminant analysis [5, 12], and other techniques [26]. In Section 2.3, we identified limitations of the above methods. A more recent line of work [6, 18] enforces domain-invariance of the optimal $P(Y|\Phi(\mathbf{x}))$ that we compare to in the evaluation. There is work on use of causal reasoning for domain adaptation [7, 8, 16, 49] that assumes $Y \rightarrow X$ direction and other work [6, 50] on connecting causality that assumes $X \rightarrow Y$. Our SCM model unites these streams by introducing Y_{true} and labelled Y and develop an invariance condition for domain generalization that is valid under both interpretations. Perhaps the closest to our work is by Heinze-Deml and Meinshausen [8] who use the object concept in generation of input for a single domain but assume that objects are observed. We provide an algorithm that does not depend on observed objects. In doing so, we provide theoretical justification for the past uses of contrastive loss in domain generalization based on the class label [19, 20] or using augmented data [51].

Table 3: Accuracy on PACS with ResNet 18 (default), Resnet 18 with test domain validation, and ResNet 50. The results for JiGen [34], S-MLDG [37], D-SAM [38], MMLD [39], DDAIG [40] SagNet [41], DANN [3], C-DANN [13], DRO [42], Mixup [43, 44, 45], IRM [6], MLDG [35], MMD [2], CORAL [46], were taken from the DomainBed [29] paper. For G2DM [47], CSD [23], MASF [20], EpiFCR [36], MetaReg [48], it was taken from the respective paper.

	Photo	Art Painting	Cartoon	Sketch	Average.
ERM	95.84 (0.27)	77.86 (1.29)	76.91 (0.64)	75.43 (0.37)	81.51
JiGen	96.0	79.42	75.25	71.35	80.41
MASF	94.99 (0.09)	80.29 (0.18)	77.17 (0.08)	71.69 (0.22)	81.04
G2DM	93.75	77.78	75.54	77.58	81.16
CSD	94.1 (0.2)	78.9 (1.1)	75.8 (1.0)	76.7 (1.2)	81.4
EpiFCR	93.9	82.1	77.0	73.0	81.5
MetaReg	95.5 (0.24)	83.7 (0.19)	77.2 (0.31)	70.3 (0.28)	81.7
S-MLDG	94.80	80.50	77.80	72.80	81.50
D-SAM	94.30	79.48	77.13	75.30	81.55
MMLD	96.09	81.28	77.16	72.29	81.83
DDAIG	95.30	84.20	78.10	74.70	83.10
SagNet	95.47	83.58	77.66	76.30	83.25
RandMatch	96.15 (0.27)	78.99 (0.97)	78.65 (0.68)	76.70 (1.22)	82.62
MatchDG	96.47 (0.40)	79.28 (1.01)	79.47 (0.48)	76.94 (1.65)	83.04
MDGHybrid	96.77 (0.32)	81.20 (0.18)	80.38 (0.40)	77.23 (1.21)	83.89
Asadi et al.	96.93	83.01	79.39	78.62	84.46
G2DM (Test)	94.63	81.44	79.35	79.52	83.34
RandMatch (Test)	96.58 (0.32)	80.21 (1.21)	80.06 (0.71)	81.44 (0.58)	84.58
MatchDG (Test)	96.73 (0.27)	80.79 (0.39)	81.06 (0.55)	80.34 (0.43)	84.63
MDGHybrid (Test)	97.17 (0.12)	81.90 (0.22)	81.68 (0.11)	79.67 (0.35)	85.11
DomainBed (ResNet50)	97.8 (0.0)	88.1 (0.1)	77.9 (1.3)	79.1 (0.9)	85.7
MASF (ResNet50)	95.01 (0.10)	82.89 (0.16)	80.49 (0.21)	72.29 (0.15)	82.67
C-DANN (ResNet50)	97.0 (0.4)	84.0 (0.9)	78.5 (1.5)	71.8 (3.9)	82.8
MetaReg (ResNet50)	97.6 (0.31)	87.2 (0.13)	79.2 (0.27)	70.3 (0.18)	83.6
DRO (ResNet50)	98.0 (0.3)	86.4 (0.3)	79.9 (0.8)	72.1 (0.7)	84.1
Mixup (ResNet50)	97.7 (0.2)	86.5 (0.4)	76.6 (1.5)	76.5 (1.2)	84.3
IRM (ResNet50)	96.7 (0.3)	85.0 (1.6)	77.6 (0.9)	78.5 (2.6)	84.4
DANN (ResNet50)	97.6 (0.2)	85.9 (0.5)	79.9 (1.4)	75.2 (2.8)	84.6
MLDG (ResNet50)	97.0 (0.9)	89.1 (0.9)	78.8 (0.7)	74.4 (2.0)	84.8
MMD (ResNet50)	97.5 (0.4)	84.5 (0.6)	79.7 (0.7)	78.1 (1.3)	85.0
CORAL (ResNet50)	97.6 (0.0)	87.7 (0.6)	79.2 (1.1)	79.4 (0.7)	86.0
RandMatch (ResNet50)	97.78 (0.18)	83.82 (0.55)	79.44 (0.82)	80.76 (0.38)	85.33
MatchDG (ResNet50)	97.93 (0.23)	84.75 (0.78)	81.64 (0.45)	79.36 (0.25)	85.87
MDGHybrid (ResNet50)	97.98 (0.06)	86.12 (0.73)	83.59 (0.55)	82.77 (0.78)	87.57

Table 4: Chest X-Rays data. As an upper bound, training ERM on the target domain RSNA yields 73% accuracy.

	NIH (Source)	Chex (Source)	RSNA (Target)
ERM	78.9 (0.34)	84.3 (3.52)	55.2 (2.27)
IRM	79.1 (1.01)	83.4 (2.42)	56.6 (2.04)
CSD	73.2 (3.35)	83.3 (2.03)	60.5 (0.82)
RandMatch	75.3 (1.87)	83.6 (1.84)	57.4 (1.76)
MatchDG	74.7 (0.66)	82.2 (0.68)	58.4 (0.62)
MDGHybrid	74.3 (0.91)	82.4 (1.03)	62.6 (0.72)

Meta-learning. Meta-learning can be applied to domain generalization, by creating meta-train and meta-test domains within each mini-batch and ensuring that the weight updates perform well on

the meta-test domains [20, 35, 48]. While we showed that a contrastive training only can achieve promising results, combining meta-learning with our approach is an interesting future direction.

Dataset augmentation. The data augmentation methods create more out-of-domain samples, from distributions within a bounded distance [52] or on a continuous space of domain interventions [10].

Parameter Decomposition. Finally there is work that focuses on identifying common model parameters across domains [23, 24, 53], rather than a common input representation. We compared our work against one such recent method based on low-rank decomposition (CSD) [23].

6 Conclusion

We presented a causal interpretation of domain generalization and used it to derive a method that matches representations of input pairs that share causal features. We find that combining ERM with a simple invariance condition performs better on benchmarks than prior work, and hope to investigate matching-based methods with domain-dependent noise on class label and object in future work.

Acknowledgements

We would like to thank Adith Swaminathan, Aditya Nori, Emre Kiciman, Praneeth Netrapalli, Tobias Schnabel, and Vineeth Balasubramanian who provided us valuable feedback on this work. We also thank Vihari Piratla who helped us with reproducing the CSD method and other baselines.

References

- [1] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013.
- [2] 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.
- [3] 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.
- [4] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [5] Muhammad Ghifary, David Balduzzi, W Bastiaan Kleijn, and Mengjie Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1414–1430, 2016.
- [6] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.
- [7] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pages 2839–2848, 2016.
- [8] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2019.
- [9] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*, 2019.
- [10] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.
- [11] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

- [12] Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 35. NIH Public Access, 2019.
- [13] 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.
- [14] Judea Pearl. *Causality*. Cambridge university press, 2009.
- [15] John W Miller, Rod Goodman, and Padhraic Smyth. On loss functions which minimize to conditional expected values and posterior probabilities. *IEEE Transactions on Information Theory*, 39(4):1404–1408, 1993.
- [16] Sara Magliacane, Thijs van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. In *Advances in Neural Information Processing Systems*, pages 10846–10856, 2018.
- [17] Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 527–536, 2019.
- [18] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. *arXiv preprint arXiv:2002.04692*, 2020.
- [19] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017.
- [20] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pages 6447–6458, 2019.
- [21] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.
- [22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.
- [23] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. *Proceedings of the International Conference of Machine Learning (ICML) 2020*, 2020.
- [24] 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.
- [25] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.
- [26] 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.
- [27] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silvana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghighi, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.
- [28] Kaggle: Rsna pneumonia detection challenge, 2018.
- [29] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.
- [30] Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020.

- [31] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [32] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*, 2020.
- [33] Nader Asadi, Amir M Sarfi, Mehrdad Hosseinzadeh, Zahra Karimpour, and Mahdi Eftekhari. Towards shape biased unsupervised representation learning for domain generalization. *arXiv preprint arXiv:1909.08245*, 2019.
- [34] Fabio M Carlucci, Antonio D’Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.
- [35] 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.
- [36] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1446–1455, 2019.
- [37] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Sequential learning for domain generalization. *arXiv preprint arXiv:2004.01377*, 2020.
- [38] Antonio D’Innocente and Barbara Caputo. Domain generalization with domain-specific aggregation modules. In *German Conference on Pattern Recognition*, pages 187–198. Springer, 2018.
- [39] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *AAAI*, pages 11749–11756, 2020.
- [40] Kaiyang Zhou, Yongxin Yang, Timothy M Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, pages 13025–13032, 2020.
- [41] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*, 2019.
- [42] 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.
- [43] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. *arXiv preprint arXiv:1912.01805*, 2019.
- [44] Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020.
- [45] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020.
- [46] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.
- [47] Isabela Albuquerque, João Monteiro, Mohammad Darvishi, Tiago H. Falk, and Ioannis Mitliagkas. Generalizing to unseen domains via distribution matching, 2020.
- [48] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pages 998–1008, 2018.
- [49] Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.
- [50] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.
- [51] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

- [52] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, pages 5334–5344, 2018.
- [53] Hal Daumé III, Abhishek Kumar, and Avishek Saha. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59. Association for Computational Linguistics, 2010.
- [54] Joseph Paul Cohen, Mohammad Hashir, Rupert Brooks, and Hadrien Bertrand. On the limits of cross-domain generalization in automated x-ray prediction. *arXiv preprint arXiv:2002.02497*, 2020.
- [55] Yiyang Li, Yongxin Yang, Wei Zhou, and Timothy M Hospedales. Feature-critic networks for heterogeneous domain generalization. *arXiv preprint arXiv:1901.11448*, 2019.
- [56] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [57] Isabela Albuquerque, Nikhil Naik, Junnan Li, Nitish Keskar, and Richard Socher. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*, 2020.
- [58] Haohan Wang, Zexue He, Zachary C Lipton, and Eric P Xing. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.
- [59] Mohammad Mahfujur Rahman, Clinton Fookes, Mahsa Baktashmotlagh, and Sridha Sridharan. Correlation-aware adversarial domain adaptation and generalization. *Pattern Recognition*, 100:107124, 2020.

A Theory and Proofs

A.1 Constructing the causal graph

When considering classification tasks, there are two viewpoints on whether the features cause the class label, or whether the class labels cause the features. [7, 16, 49] assume a generative process where the true class label determines the features in the observed data. In contrast, [50, 6] consider a generative process where the features are used to assign a label, e.g., when manually labelling a set of images. We believe that both mechanisms are possible, depending on the context. In particular, it is plausible that the true class label Y_{true} causes the features, but it is not observed. Instead, what is observed is the output of a manual labelling process, where the features are used to label each input with its class Y [6].

Given these differences, we construct a causal graph (Figure 1) that includes both Y_{true} and Y (as in [8]), and is consistent with both viewpoints about the direction of the causal mechanism. Importantly, all d-separation results reported in the main text hold true irrespective of whether we choose Y or Y_{true} as the class label. We use Y as the label in the main text, since it corresponds to many settings where the observed class label is a result of a (possibly noisy) manual labelling process.

In addition, we chose to represent X_C and X_A as near-to-final features, that are combined using a simple operation to generate the observed features X . Under this representation, the object O does cause X_A ; X_A is produced by combination of the domain and the object. Another equally valid construction is to assume that X_A contains only the domain information, and a more complex operation generates the observed features using X_C (object information) and X_A . The corresponding causal graph will omit the edge from object O to X_A . Both these graphs are allowed by our framework. All d-separation results reported in the main text hold true irrespective of whether there exists an edge from O to X_A .

A.2 D-separation

We first expand on the d-separation definition, providing a few examples that illustrate conditional independence implications of specific graph structures in Figure 3. We use these three conditions for all the proofs below.

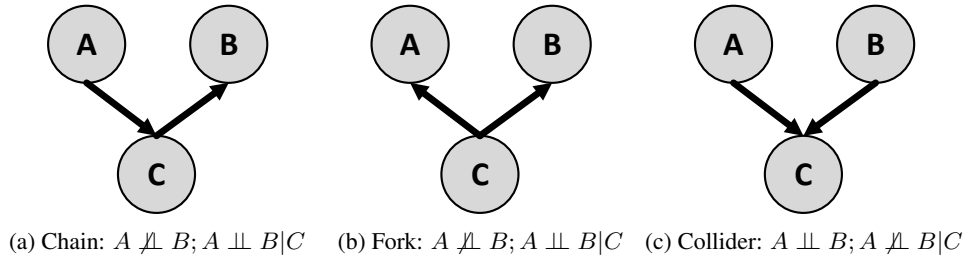


Figure 3: Causal graphs with the node C as a chain, fork, or a collider. By the d-separation criteria, A and B are conditionally independent given C in (a) and (b). In (c) however, A and B are independent but become conditionally dependent given C .

A.3 Proof of Proposition 1

Proposition 1. *The ERM estimator from (1) learns the true f^* , as the set of training domains $D_m = \mathcal{D}$ and number of data samples $n \rightarrow \infty$.*

Proof. Expanding on (1), the ERM estimator can be written as:

$$\begin{aligned}
\hat{f}_{ERM} &= \arg \min_{f \in \mathcal{F}} \frac{1}{\sum_{d=1}^m n_d} \sum_{d=1}^m \sum_{i=1}^{n_d} l(y_i^{(d)}, f(\mathbf{x}_i^{(d)})) \\
&= \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{d, \mathbf{x}, y \sim \mathcal{P}_m} l(y^{(d)}, f(\mathbf{x}^{(d)})) \\
&= \arg \min_{f \in \mathcal{F}} \hat{\mathbb{E}}_{d, \mathbf{x}, y \sim \mathcal{P}_m} l(y^{(d)}, f(\mathbf{x}^{(d)})) \\
&= \arg \min_{f \in \mathcal{F}} \hat{\mathbb{E}}_{S_n \sim \mathcal{P}_m} [l(y^{(d)}, f(\mathbf{x}^{(d)}))]
\end{aligned} \tag{7}$$

where number of samples $n = \sum_{d=1}^m n_d$ and $S_n \sim \mathcal{P}_m(D, X, Y)$ is the training dataset of size n . Since $D_m = \mathcal{D} \Rightarrow \mathcal{P} = \mathcal{P}_m$. As $n \rightarrow \infty$,

$$f_{ERM}^\infty = \arg \min_{f \in \mathcal{F}} \lim_{n \rightarrow \infty} \hat{\mathbb{E}}_{S_n \sim \mathcal{P}} l(y^{(d)}, f(\mathbf{x}^{(d)})) = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{d, \mathbf{x}, y \sim \mathcal{P}} l(y^{(d)}, f(\mathbf{x}^{(d)})) = f^* \tag{8}$$

where the last equality is due to the definition of f^* in Section 2. □

A.4 Proof of Theorem 1

Theorem 1. For a finite number of domains m , as the number of examples in each domain $n_d \rightarrow \infty$,

1. The set of representations that satisfy the condition $\sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0$ contains the optimal $\Phi(\mathbf{x}) = X_C$ that minimizes the domain generalization loss in (2).
2. Assuming that $P(X_a|O, D) < 1$ for every high-level feature X_a that is directly caused by domain, and for P -admissible loss functions [15] whose minimization is conditional expectation (e.g., ℓ_2 or cross-entropy), a loss-minimizing classifier for the following loss is the true function f^* , for some value of λ .

$$f_{\text{perfectmatch}} = \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \lambda \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \tag{4}$$

Proof. **CLAIM 1.** The matching condition can be written as:

$$C(\Phi) = \min_{\Phi} \sum_{d, d' \in D_m} \lim_{\substack{n_d \rightarrow \infty \\ n_{d'} \rightarrow \infty}} \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \tag{9}$$

where $\Omega(j, k) = 1$ for pairs of inputs \mathbf{x}_j and \mathbf{x}_k from two different domains d and d' that correspond to the same object. The distance metric dist is non-negative, so the optimal Φ is when $C(\Phi)$ is zero.

As in the SCM from Figure 1b, let X_c represent a feature vector such that it is generated based only on the object O and that it leads to the optimal classifier in (2). From Sections 2.1 and 2.2, we know that $X_c \perp\!\!\!\perp D|O$ and that $x_c = g_{xc}(o)$. Thus, x_c is the same for inputs from the same object and we can write:

$$\text{dist}(\mathbf{x}_{c,j}^{(d)}, \mathbf{x}_{c,k}^{(d')}) = 0 \quad \forall d, d' \in D_m \text{ such that } \Omega(j, k) = 1 \tag{10}$$

Hence, $\Phi(\mathbf{x}) = \mathbf{x}_c$ leads to zero regularizer term and is one of the optimal minimizers for $C(\Phi)$.

CLAIM 2. Further, we show that any other optimal Φ is either a function of \mathbf{x}_c or a constant for all inputs. We prove by contradiction.

Let X_A represent the set of unobserved high-level features that are generated based on both the object O and the domain D . From the SCM from Figure 1b, a feature vector $X_a \subseteq X_A$ is independent of X_c given the object, $X_a \perp\!\!\!\perp X_c|O$, and $x_a = g_{xa}(d, o, \epsilon_{xa})$. Further, let there be an optimal $\Phi_a(\mathbf{x})$ for $C(\Phi)$ such that it depends on some $X_a \subseteq X_A$ (and is not trivially a constant function). Since Φ_a is optimal, $\Phi_a(\mathbf{x}_j^{(d)}) = \Phi_a(\mathbf{x}_k^{(d')})$ for all d, d' such that $\Omega(j, k) = 1$, where inputs \mathbf{x}_j and \mathbf{x}_k correspond to the same object.

Let us assume that there exists at least one object o for which the effect of domain is stochastic. That is, due to domain-dependent variation, $P(X_a = x_a|D = d, O = o) < 1$. for some d and o .

Now consider a pair of inputs $\mathbf{x}_l^{(d)}$ and $\mathbf{x}_i^{(d')}$ from the same object o such that $\Omega(l, i) = 1$, and their corresponding representations are $\Phi_a(\mathbf{x}_l^{(d)})$ and $\Phi_a(\mathbf{x}_i^{(d')})$. Due to domain-dependent variation, with non-zero probability, the high-level X_a features are not the same for these two input data points, $x_{a,l}^{(d)} \neq x_{a,i}^{(d')}$. Since Φ is a deterministic function of \mathbf{x} that is not independent of X_a , if an input \mathbf{x} has a different X_a , its value of $\Phi(\mathbf{x})$ will also be different. Thus, with non-zero probability, we obtain that $\Phi(\mathbf{x}_l^{(d)}) \neq \Phi(\mathbf{x}_i^{(d')})$, unless the effect of X_a is a constant function. Hence, a contradiction and optimal Φ cannot depend on any $X_a \subseteq X_A$ that are generated based on the domain.

Therefore, an optimal solution to $C(\Phi)$ can only depend on X_c . However, any function of X_c is optimal, including trivial functions like the constant function (that will have low accuracy). Below we show that using the ERM term in (4) ensures that the optimal solution contains only those functions of X_c that also maximize accuracy.

Using (3), the empirical optimizer function can be written as (where we scale the loss by a constant $n = \sum_d n_d$, the total number of training data points):

$$\hat{f}_{pmatch} = \arg \min_{h, \Phi} \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(h(\Phi(X)), Y) \text{ s.t. } \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0 \quad (11)$$

$$\begin{aligned} &= \arg \min_{h, \psi} \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(h(\psi(X_c)), Y) \\ &= \arg \min_f \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(f(X_c), Y) \end{aligned} \quad (12)$$

where $\psi(X_c)$ denotes all functions of X_c that are optimal for (9), and the last equality is because $h \circ \psi$ can be written as $f = h \circ \psi$. Since we assume that L is a P-admissible loss function, its minimizer is the conditional expected value. Thus, for any domain d , $\arg \min_f \lim_{n_d \rightarrow \infty} \frac{1}{n_d} L_d(f(X_c), Y) = \mathbb{E}[Y|X_c, D]$. Further, by d-separation, $Y \perp\!\!\!\perp D|X_c$. Therefore, $\mathbb{E}[Y|X_c, D] = \mathbb{E}[Y|X_c]$. The above equation indicates that the loss minimizer function on any domain is independent of the domain. Thus, for the m training domains, we can write:

$$\arg \min_{f \in \mathcal{F}} \lim_{n_d \rightarrow \infty} \frac{1}{n_d} L_d(f(X_c), Y) = \arg \min_{f \in \mathcal{F}} \mathbb{E}[l(f(\mathbf{x}_c), y)] = \mathbb{E}[Y|X_c] \quad \forall d \in D_m \quad (13)$$

Now (12) can be rewritten as,

$$\hat{f}_{pmatch} = \arg \min_f \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} \frac{L_d(f(X_c), Y)}{n_d} n_d = \arg \min_f \sum_{d=1}^m \lim_{n_d \rightarrow \infty} \frac{L_d(f(X_c), Y)}{n_d} \frac{n_d}{n} \quad (14)$$

From the equation above, the loss for \hat{f}_{pmatch} can be considered as a weighted sum of the average loss on each training domain where the weights are all positive. Since $\mathbb{E}[Y|X_c]$ minimizes the average loss on each domain as $n_d \rightarrow \infty$, it will also minimize the overall weighted loss for all values of the weights. Therefore, for any dataset over m domains in D_m , $\mathbb{E}[Y|X_c]$ is the optimal function that minimizes the overall loss.

Moreover, we can also write f^* as:

$$\begin{aligned} f^* &= \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(d, \mathbf{x}, y)}[l(y, f(\mathbf{x}))] = \arg \min_{h \in \mathcal{F}} \mathbb{E}_{(d, \mathbf{x}, y)}[l(y, h(\mathbf{x}_c))] \\ &= \arg \min_{h \in \mathcal{F}} \mathbb{E}_{(\mathbf{x}, y)}[l(y, h(\mathbf{x}_c))] = \mathbb{E}[Y|X_c] \end{aligned} \quad (15)$$

where we utilize (13) and that the loss function is P-admissible. Hence, $f^* = \mathbb{E}[Y|X_c]$ is the loss-minimizing function for the loss in (12).

Finally, using a Lagrangian multiplier, minimizing the following soft constraint loss is equivalent to minimizing (11), for some value of λ .

$$\hat{f}_{\text{pmatch}} = \lim_{\forall d \in D_m, n_d \rightarrow \infty} \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \lambda \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \quad (16)$$

The result follows. \square

Comment on Theorem 1. In the case where the effect of a domain is also deterministic, it is possible that $P(X_a|O, D) = 1$ (e.g., in artificially created domains like Rotated-MNIST where every object is rotated by the *exact* same amount in each domain). In that case Theorem 1 does not apply and it is possible to learn a representation Φ_a that depends on $X_a \subseteq X_A$ and still minimizes $C(\Phi)$ to attain $C(\Phi) = 0$. For example, with two training domains on Rotated-MNIST dataset $(0^\circ, \alpha^\circ)$, it is possible to learn a representation that simply memorizes to “un-rotate” the α angle back to 0° . Such a representation will fail to generalize to domains with different rotation angles, but nonetheless minimizes $C(\Phi)$ by attaining the exact same representation for each object.

In practice, we conjecture that such undesirable Φ_a are avoided by model-size regularization during training. As the number of domains increase, it may be simpler to learn a single transformation (representation) based on X_c (and independent of X_c features like angle) than learn separate angle-wise transformations for each train domain.

A.5 Proof of Corollary 1

Corollary 1. *The conditions enforced by domain-invariant $(\Phi(x) \perp\!\!\!\perp D)$ or class-conditional domain-invariant $(\Phi(x) \perp\!\!\!\perp D|Y)$ methods are not satisfied by the causal representation X_C . Thus, without additional assumptions, the set of representations that satisfy any of these conditions does not contain X_C , even as $n \rightarrow \infty$.*

Proof. As in the SCM from Figure 1b, let X_c represent an unobserved high-level feature vector such that it is generated based only on the object O and that it leads to the optimal classifier in (2). From Sections 2.1 and 2.2, we know that $X_c \perp\!\!\!\perp D|O$ and that $x_c = g_{x_c}(o)$. Following a similar proof to Theorem 1 (Claim 1), we check whether $\Phi(\mathbf{x}) = \mathbf{x}_c$ satisfies the invariance conditions required by the two methods.

1. **Domain-invariant:** The required condition for a representation is that $\Phi_{\text{DI}}(\mathbf{x}) \perp\!\!\!\perp D$. But using the d-separation criteria on the SCM in Figure 1b, we find that $X_c \not\perp\!\!\!\perp D$ due to a path through Object O .
2. **Class-conditional domain-invariant:** The required condition for a representation is that $\Phi_{\text{CDI}} \perp\!\!\!\perp D|Y$. However using the d-separation criteria on the SCM, we find that $X_c \not\perp\!\!\!\perp D|Y$ due to a path through Object O that is not blocked by Y (nor by Y_{true} if it is observed).

Therefore, under the conditions proposed by these methods, X_c or any function of X_c is not an optimal solution without making any additional assumptions. Hence, even with infinite samples, a method optimizing for these conditions will not retrieve X_c . \square

A.6 Proof of Theorem 2

Theorem 2. *Assume training domains such that for any two same-class inputs $\mathbf{x}_j^{(d)}$ and $\mathbf{x}_k^{(d')}$ from domains d and d' , $\text{dist}(x_{a,j}^{(d)}, x_{a,k}^{(d')}) \geq \delta_a$ where x_a is any high-level feature that is caused directly by domain. Further, assume that the distance over X_C between same-class inputs from different domains is bounded: $\text{dist}(x_{c,j}^{(d)}, x_{c,k}^{(d')}) \leq \delta_c$ and $\delta_c < \delta_a$ ($\delta_c, \delta_a \in \mathbb{R}^+$). Then for some λ , a loss-minimizing classifier for the loss from (5) is the true function f^* , given a P -admissible loss function and a finite number of domains m with $n_d \rightarrow \infty$ in each domain.*

Proof. Since δ_c is a constant for a dataset, the loss from (5) can be written as,

$$f_{\text{randommatch}} = \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \lambda \sum_{\Omega_Y(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) - J\delta_c \quad (17)$$

where we added a scaling constant J for the total number of matches. For some value of λ , the above optimization is equivalent to,

$$\arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) \text{ s.t. } \sum_{\Omega_Y(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \leq J\delta_c \quad (18)$$

where δ_c is the maximum difference in X_C between two inputs with the same class but different domains.

We first show that $\Phi(\mathbf{x}) = \mathbf{x}_c$ satisfies the constraint in (18), where \mathbf{x}_c is any feature vector that is generated based only on the object ($\mathbf{x}_c = g_{xc}(o)$), as defined in the proof of Theorem 1. Now since $\text{dist}(\mathbf{x}_{c,l}^{(d)}, \mathbf{x}_{c,i}^{(d')}) \leq \delta_c$ for any two inputs \mathbf{x}_l and \mathbf{x}_i with the same class and different domains, the constraint from (18) is satisfied by $\Phi(\mathbf{x}) = \mathbf{x}_c$.

In addition, let \mathbf{x}_a be a feature vector that is generated based on both the object and a domain ($\mathbf{x}_a = g_{xa}(d, o, \epsilon_{xa})$) and that is conditionally independent of \mathbf{x}_c , $X_A \perp\!\!\!\perp X_C | O$. Since we assume that $\text{dist}(\mathbf{x}_{a,l}^{(d)}, \mathbf{x}_{a,i}^{(d')}) \geq \delta_a$ and $\delta_c \leq \delta_a$, such an \mathbf{x}_a cannot satisfy the constraint in (18).

Hence, the solution to the constraint in (18) can only include \mathbf{x}_c features that are not caused by the domain (or some function of \mathbf{x}_c , $\psi(\mathbf{x}_c)$). Thus, (18) can be rewritten as,

$$\begin{aligned} &= \arg \min_{h, \psi} \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(h(\psi(X_c)), Y) \\ &= \arg \min_f \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(f(X_c), Y) \end{aligned} \quad (19)$$

Note that the above equation is the same as (12) from Theorem 1 proof. Following the same steps as in Theorem 1 proof, we find that $f^* = \mathbb{E}[Y | X_c]$ is the loss-minimizing classifier for (18), and hence for (5) at some value of λ . \square

B Evaluation and implementation details

In this section we describe implementation details for our proposed methods. We also discuss the evaluation protocol, including details about hyperparameters and cross-validation.

B.1 Implementation details

For the implementation of ERM-RandMatch in Eq. (5), ERM-PerfMatch in Eq. (4); we use the cross-entropy loss for L_d and l_2 distance for dist in Eq. (4, 5). For both methods, we consider the representation $\Phi(\mathbf{x})$ to be the last layer of the network. That is, we take h to be identity function in Eq. (4, 5) for simplicity. It is also possible to use the second-last or any other previous layer as a representation, but the last layer performed well in our experiments.

We use SGD to optimize the loss for all the datasets, with details about learning rate, epochs, batch size, weight decay etc. provided in the section B.3 ahead. For all the different methods, we sample batches from the data matrix consisting of data points matched across domains; hence we ensure an equal number of data points from each source domain in a batch. When training with MatchDG, the underlying architecture for Phase 2 is kept the same for ERM, RandMatch, PerfMatch for the respective task; with the details mentioned below for each dataset. The details for the Phase-1 architecture are specified in section B.3, Table 6.

Rotated MNIST & Fashion-MNIST. The datasets contain rotations of grayscale MNIST handwritten digits and fashion article images from 0° to 90° with an interval of 15° [26], where each rotation

angle represents a domain and the task is to predict the class label. For Table 1, we follow the setup in CSD [23], we report accuracy on 0° and 90° together as the test domain and the rest as the train domains. We use 2,000 and 10,000 training samples from each domain for rotated MNIST and Fashion-MNIST, and train models using Resnet-18 architecture (without pre training). We choose this as our primary setup and select 0° and 90° as our target domain, since these are known to be the most difficult domains to generalize [23, 19].

Further, we also evaluate on other setups of Rotated MNIST in prior works [19, 29], which involve six domains (0° , 15° , 30° , 45° , 60° , 75°), and evaluate for each domain being the target domains with the remaining five used as source domains. We sample 1000 data points for each domain and evaluate using the LeNet architecture (Table 8) as per the setup proposed by [19]. Similarly, we sample all the 70,000 images in MNIST and evaluate using the custom architecture (Table 9) as per the setup proposed by [29].

Another important distinction between different setups above is the use of different digits for the source and the target domains ([23], [29]), as opposed to the use of same digits across the source and the target domains in setup of [19] which makes the task easier as it leaks information about the target domains.

Finally, for all the different setups proposed above, we create an additional validation set for each domain with 20% percent size as of the training set for that domain. We use the validation set from the source domains for hyper parameter tuning.

PACS. This dataset contains total 9991 images from four domains: Photos (P), Art painting (A), Cartoon (C) and Sketch (S). The task is to classify objects over 7 classes. Following [20], we train 4 models with each domain as the target using Resnet-18 (Table 3), Resnet-50 (Table 3) and Alexnet (Table 14), with each architecture pre-trained on ImageNet. We also the following data augmentations [29] while training: Random Crop, Horizontal Flip, Color Jitter, and Random Gray Scale.

Chest X-ray. We use Chest X-rays images from three different sources: NIH [25], ChexPert [27] and RSNA [28]. The task is to detect whether the image corresponds to a patient with Pneumonia (1) or not (0). For ease of interpretation, we balance the data such that there are equal number of images per class in each domain. Since majority of the images in each domain correspond to the class (0), we sample a subset of the images to ensure that there is no class imbalance in each domain. For each domain, we first utilize all the images for the class (1) and choose 30% percent of them for test set, with the remaining 70% for the training set. Further, we create a validation set from the training set with 25% size as that of training set. Then for each of the training/validation/test split, we sample an equal number of images from class (0) to remove class imbalance. The dataset size for the different splits on each domain are described below:

- NIH: Train (800), Validation (200), Test (430)
- ChexPert: Train (2618), Validation (654), Test (1402)
- RSNA: Train (3368), Validation (841), Test (1803)

Following prior works [54], we use the pre-trained DenseNet-121 architecture for classification. We use the following data augmentations: Random Crop, Horizontal Flip, and Affine Transformations. We further create spurious correlations, all the images of the class 0 in the training domains (NIH, ChexPert) are translated vertically downwards; while no such translation is done for the test domain (RSNA). We translate the images in each source domain by a fixed amount, which varies over different source domains. For the images in the source domain NIH, we translate all the pixels of each image downwards by 45 units, whereas for the source domain ChexPert, we translate 30 units. This leads to a downward shift in the position of lungs in the images for the class 0 as compared to those for class 1, which could lead to models utilizing this spurious relative difference in position of lungs for the classifications task.

B.1.1 MatchDG implementation details:

The MatchDG algorithm proceeds in two phases.

Initialization: We construct matches of pairs of same-class data points from different domains.

Hence, given each data point we randomly select another data point with the same class from another domain. The matching for each class across domains is done relative to a base domain; which is chosen by taking the domain that has the highest number of samples for that class. This is done to avoid missing out on data points when there is class imbalance across domains. Specifically, we iterate over classes and for each class, we match data points randomly across domains w.r.t a base domain for that class. This leads to matrix \mathcal{M} of size (N', K) , where N' refers to the updated domain size (sum of the size of base domain for all the classes) and K refers to the total number of domains. We describe the two phases below:

Phase 1: We sample batches (B, K) from the matched data matrix \mathcal{M} , where B is the batch size. For each data point x_i in the batch, we minimize the contrastive loss from (6) by selecting its matched data points across domains as the positive matches and consider every data point with a different class label from x_i to be a negative match.

After every t epochs, we periodically update the matched data matrix by using the representations learnt by contrastive loss minimization. We follow the same procedure of selecting a base domain for each class, but instead of randomly matching data points across domains, we find the nearest neighbour for the data point in base domain among the data points in the other domains with the same class label based on the l_2 distance between their representations.

At the end of Phase I, we update the matched data matrix based on l_2 distance over the final representations learnt. We call these matches as the *inferred* matches.

Phase 2: We train using the loss from Eq. (5), but instead of random matches, we use the inferred matches generated from Phase 1 (ERM + Inferred Match). We train the network from scratch in Phase 2 and use the representations learnt in Phase 1 to only update the matched data matrix.

The updated data matrix based on representations learnt in Phase 1 may lead to many-to-one matches from the base domain to the other domains. This can lead to certain data points being excluded from the training batches. Therefore, we construct batches such that each batch consists of two parts. The first is sampled as in Phase 1 from the matched data matrix. The second part is sampled randomly from all train domains. Specifically, for each batch (B, K) sampled from the matched data matrix, we sample an additional part of size B with data points selected randomly across domains. The loss for the second part of the batch is simply ERM, along with ERM + InferredMatch Loss on the first part of the batch.

B.2 Metrics for evaluating quality of learnt matches

Here we describe the three metrics used for measuring overlap of the learnt matches with ground-truth “perfect” matches.

Overlap %: Percentage of matches (j, k) as per the perfect match strategy Ω that are also consistent with the learnt match strategy Ω' .

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \Omega'(j, k)}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (20)$$

Top-10 Overlap %: Percentage of matches (j, k) as per the perfect match strategy Ω that are among the Top-10 matches for the data point j w.r.t the learnt match strategy Ω' i.e. $S_{\Omega'}^{10}(j)$

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \mathbb{1}[k \in S_{\Omega'}^{10}(j)]}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (21)$$

Mean Rank: For the matches (j, k) as per the perfect match strategy Ω , compute the mean rank for the data point j w.r.t the learnt match strategy Ω' i.e. $S_{\Omega'}(j)$

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \text{Rank}[k \in S_{\Omega'}(j)]}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (22)$$

B.3 HyperParameter Tuning

To select hyperparameters, prior works [20, 34, 35] use leave-one-domain-out validation, which means that the hyperparameters are tuned after looking at data from the unseen domain. Such a setup is violates the premise of the domain generalization task that assumes that a model should have no access to the test domain. Therefore, in this work, we construct a validation set using only the source domains and use it for hyper parameter tuning. In the case of PACS, we already have access to the validation indices for each domain and use them to construct a validation set based on the source domains. For Rotated & Fashion MNIST, Chest X-ray datasets, we create validation set for each source domain as described in the section B.1 above. Hence, the model does not have access to the data points from the target/test domains at the time of training and validation.

We perform a grid search over pre-defined values for each hyper parameter and report the optimal values along with the values used for grid search in Table 5. Further, we do early stopping based on the validation accuracy on source domains and use the models which obtain the best validation accuracy.

For the case of MatchDG Phase-1, we do not perform grid search and use default values for each hyper parameter (Table 6). We still do early stopping for MatchDG Phase-1, based on the metric Top-10 Overlap (Section B.2) over the validation set of source domains. Since we require perfect matches for the evaluation of the metric Top-10 Overlap, we create perfect matches using the self augmentations (Section B.1) for each dataset.

B.4 Reproducing Results from Prior Work

MNIST and Fashion MNIST The results for MASF, CSD in Table 1 are taken from [23]; except IRM where we used the implementation available online ¹ to generate the results. The results for prior approaches in Table 8 are taken from [10], [30]. For the results using DomainBed setup in Table 9, the results for prior approaches are taken from [29].

For ResNet-18 architecture (Table 1); the results for the reduced number of domains for CSD and MASF were computed using their code which is available online ²³. The MASF code was hardcoded to run for PACS dataset; which has 3 source domains that gets divided into 2 meta train and 1 meta test domain. Their code requires atleast 2 meta train domains; which leads to an issue for only 2 source domains (30, 45). In Table 1 when there are only 2 source domains; their code considers only 1 meta train domain. To resolve this issue; we create a copy of the 1 meta train domain and thus run MASF for source domains 30, 45 on MNIST.

PACS We did not generate results for the prior approaches for PACS by developing or using existing implementations. All the results for the prior approaches on PACS were taken from the respective papers as specified in the Table 3, 14.

Chest X-ray The results for the prior approaches CSD, IRM were generated using the implementations of both of the methods available on github ^{1,2}.

C Additional Evaluation on Rotated MNIST and Fashion-MNIST

Here we present results for additional experiments on Rotated MNIST and Fashion-MNIST datasets using MatchDG.

C.1 Comparing MatchDG with prior work on the LeNet Network

Table 8 compares the accuracy results for MatchDG with prior work on the LeNet architecture [19]. In this setup, there are six domains in total (0° , 15° , 30° , 45° , 60° , 75°). For each test domain, the remaining five domains are used as source training domains. We observe that matching-based training methods RandMatch and MatchDG outperform prior work. They achieve accuracy almost equal to

¹<https://github.com/facebookresearch/InvariantRiskMinimization>

²<https://github.com/vihari/CSD>

³<https://github.com/biomed-mira/masf>

Table 5: Hyper parameter selection details for all the datasets. We mention the Optimal Value for each hyper parameter and the Range used for grid search. We leave the optimal value for Epochs as blank since we do early stopping based on validation loss, with the total number of epochs for model training specified in the Range column. For the dataset PACS, since the optimal values differ for different test domains, we represent them separately in Table 7

Dataset	Hyper Parameter	Optimal Value	Range
Rotated & Fashion MNIST Table 1 (ResNet-18)	Total Epochs	-	25
	Learning Rate	0.01	[0.01]
	Batch Size	16	[16]
	Weight Decay	0.0005	[0.0005]
	Match Penalty	0.1	[0.1, 1.0]
	IRM Penalty	1.0 (RotMNIST); 0.05 (FashionMNIST)	[0.05, 0.1, 0.5, 1.0, 5.0]
	IRM Threshold	5 (RotMNIST), 0 (FashionMNIST)	[0, 5, 15, 20]
Rotated MNIST Table 8 (LeNet)	Total Epochs	-	50
	Learning Rate	0.01	[0.01]
	Batch Size	16	[16]
	Weight Decay	0.0005	[0.0005]
	Match Penalty	1.0	[0.1, 1.0]
Rotated MNIST Table 9 (DomainBed)	Total Epochs	-	25
	Learning Rate	0.01	[0.01]
	Batch Size	64	[16, 32, 64]
	Weight Decay	0.0005	[0.0005]
	Match Penalty	1.0	[0.1, 1.0]
PACS Table 3, 14 (ResNet-18, ResNet-50, AlexNet)	Total Epochs	-	50
	Learning Rate	Table 7	[0.01, 0.001, 0.0005]
	Batch Size	16	[16]
	Weight Decay	0.001	[0.001, 0.0005]
	Match Penalty	Table 7	[0.01, 0.1, 0.5, 1.0, 5.0]
Chest X-ray Table 4 (DenseNet-121)	Total Epochs	-	40
	Learning Rate	0.001	[0.01, 0.001]
	Batch Size	16	[16]
	Weight Decay	0.0005	[0.001, 0.0005]
	Match Penalty	10.0 (RandMatch), 50.0 (MatchDG, MDGHybrid)	[0.1, 1.0, 10.0, 50.0]
	IRM Penalty	10.0	[0.1, 1.0, 10.0, 50.0]
	IRM Threshold	5	[0, 5, 15, 20]

the oracle case PerfMatch for target angles (15° to 60°) that lie in between the source domains. For the harder extreme angles of 0° and 75° , MatchDG establishes new state-of-the-art accuracy results when compared to all other prior work [3, 10, 30, 26, 19, 31].

C.2 Comparing MatchDG on Domain Bed Benchmark

Table 9 compares the accuracy results for MatchDG with prior work on the setup proposed by [29]. This setup is similar to the setup in the section C.1, however, it uses a custom CNN architecture and all the 70,000 images for each domain. We observe that RandMatch and MatchDG outperform prior work on different test domains. MatchDG obtains the highest overall accuracy 98.3%, and provides significant improvement for the test domain 0° (97.1%) as compared to the best prior approach MMD [2] (96.6%).

C.3 Accuracy Results using a fraction of perfect matches

To show the importance of learning a good match function, we present the results of approaches with match function capturing some fixed percentage of perfect matches in the Table 10. For both Rotated & Fashion MNIST, we observe that the approaches that contain a higher proportion of perfect matches perform better in terms of accuracy on target domains. Hence, the quality of the match function leads to monotonic effect on the generalization performance of the matching approaches.

Interestingly, the accuracy for the range between 25-50% overlap roughly predicts the reported accuracy for MatchDG (which had about 35% and 25% overlap on the two datasets). These results confirm that in addition to high accuracy, Phase I of MatchDG allows for learning a representation where inputs with the same causal features are closer.

Table 6: MatchDG Phase 1 training details for all the datasets. We did not do hyper parameter tuning as we did for other methods, hence we mention the default value for each hyper parameter that we used. Please note we still did early stopping, the Total Epochs in the table reflects the max budget for training. The specific architecture used for Phase 1 training is also mentioned for each dataset.

Dataset	Hyper Parameter	Default Value
Rotated & Fashion MNIST Table 1, 8, 9	Total Epochs	100
	Learning Rate	0.01
	Batch Size	256 (Rotated MNIST), 64 (Fashion MNIST)
	Weight Decay	0.0005
	τ	0.05
	Architecture	ResNet-18
PACS Table 3, 14	Total Epochs	100
	Learning Rate	0.01
	Batch Size	64
	Weight Decay	0.0005
	τ	0.05
	Architecture	ResNet-50
Chest X-ray Table 4	Total Epochs	100
	Learning Rate	0.01
	Batch Size	64
	Weight Decay	0.0005
	τ	0.05
	Architecture	DenseNet-121

Table 7: Optimal values for hyper parameters on PACS. Batch Size (16), Weight Decay (0.001) was consistent across different cases. The Match Penalty for the method MDGHybrid corresponds to (MatchDG penalty, PerfMatch penalty).

Architecture	Hyper Parameter	Test Domain	ERM	RandMatch	MatchDG(Phase 2)	MDGHybrid
ResNet-18 Table 3	Learning Rate	Photo	0.0005	0.001	0.0005	0.0005
		Art Painting	0.01	0.01	0.001	0.001
		Cartoon	0.001	0.01	0.001	0.001
		Sketch	0.01	0.01	0.01	0.01
	Match Penalty	Photo	0	0.1	0.1	(0.1, 0.1)
		Art Painting	0	0.5	0.5	(0.01, 0.1)
		Cartoon	0	0.1	1.0	(0.1, 0.1)
		Sketch	0	0.1	0.5	(0.5, 0.1)
ResNet-50 Table 3	Learning Rate	Photo	0.0005	0.0005	0.0005	0.0005
		Art Painting	0.001	0.001	0.001	0.001
		Cartoon	0.001	0.0005	0.0005	0.0005
		Sketch	0.01	0.01	0.001	0.001
	Match Penalty	Photo	0	0.1	0.1	(0.1, 0.1)
		Art Painting	0	0.5	0.1	(0.01, 0.1)
		Cartoon	0	0.5	1.0	(0.01, 0.1)
		Sketch	0	0.1	0.5	(0.01, 0.1)
AlexNet Table 14	Learning Rate	Photo	0.0005	0.001	0.0005	0.0005
		Art Painting	0.001	0.001	0.001	0.001
		Cartoon	0.001	0.001	0.001	0.001
		Sketch	0.001	0.001	0.001	0.001
	Match Penalty	Photo	0	0.1	0.5	(0.1, 0.1)
		Art Painting	0	0.1	0.1	(0.01, 0.1)
		Cartoon	0	0.5	0.1	(0.01, 0.1)
		Sketch	0	0.5	0.1	(0.01, 0.1)

Table 8: Accuracy for Rotated MNIST datasets using the LeNet architecture as proposed in [19]. The results for the prior approaches CCSA [19], D-MTAE [26], LabelGrad [31], DAN [3], and CrossGrad [10] are taken from Table 9 in [10]. The results for DIVA [30] are taken from the Table 1 in their paper.

Algorithm	0	15	30	45	60	75	Average
ERM	85.6 (1.2)	99.5 (0.2)	98.9 (0.2)	98.7 (0.2)	99.3 (0.3)	88.4 (1.1)	95.1
CCSA	84.6	95.6	94.6	82.9	94.8	82.1	89.1
D-MTAE	82.5	96.3	93.4	78.6	94.2	80.5	87.6
LabelGrad	89.7	97.8	98.0	97.1	96.6	92.1	95.2
DAN	86.7	98.0	97.8	97.4	96.9	89.1	94.3
CrossGrad	88.3	98.6	98.0	97.7	97.7	91.4	95.3
DIVA	93.5 (0.3)	99.3 (0.1)	99.1 (0.1)	99.2 (0.1)	99.3 (0.1)	93.0 (0.4)	97.2
RandMatch	92.5 (0.8)	99.8 (0.1)	99.5 (0.23)	99.5 (0.1)	99.5 (0.1)	93.5 (0.4)	97.4
MatchDG	94.2(0.3)	99.8 (0.1)	99.1 (0.2)	98.9 (0.4)	99.1 (0.3)	94.1 (0.4)	97.5
PerfMatch	96.1 (0.6)	99.6 (0.1)	99.6 (0.1)	99.5 (0.3)	99.5 (0.2)	96.4 (0.3)	98.5

Table 9: Accuracy for Rotated MNIST datasets using the DomainBed setup as proposed in [29]. The results for the approaches IRM [6], DRO [42], Mixup [43, 44, 45], MLDG [35], CORAL [46], MMD [2], DANN [3], C-DANN [13] are taken from [29].

Algorithm	0	15	30	45	60	75	Average
ERM	95.6 (0.1)	99.0 (0.1)	98.9 (0.0)	99.1 (0.1)	99.0 (0.0)	96.7 (0.2)	98.0
IRM	95.9 (0.2)	98.9 (0.0)	99.0 (0.0)	98.8 (0.1)	98.9 (0.1)	95.5 (0.3)	97.9
DRO	95.9 (0.1)	98.9 (0.0)	99.0 (0.1)	99.0 (0.0)	99.0 (0.0)	96.9 (0.1)	98.1
Mixup	96.1 (0.2)	99.1 (0.0)	98.9 (0.0)	99.0 (0.0)	99.0 (0.1)	96.6 (0.1)	98.1
MLDG	95.9 (0.2)	98.9 (0.1)	99.0 (0.0)	99.1 (0.0)	99.0 (0.0)	96.0 (0.2)	98.0
CORAL	95.7 (0.2)	99.0 (0.0)	99.1 (0.1)	99.1 (0.0)	99.0 (0.0)	96.7 (0.2)	98.1
MMD	96.6 (0.1)	98.9 (0.0)	98.9 (0.1)	99.1 (0.1)	99.0 (0.0)	96.2 (0.1)	98.1
DANN	95.6 (0.3)	98.9 (0.0)	98.9 (0.0)	99.0 (0.1)	98.9 (0.0)	95.9 (0.5)	97.9
C-DANN	96.0 (0.5)	98.8 (0.0)	99.0 (0.1)	99.1 (0.0)	98.9 (0.1)	96.5 (0.3)	98.0
RandMatch	96.6 (0.3)	99.1 (0.2)	99.1 (0.1)	98.8 (0.1)	99.1 (0.2)	95.9 (0.2)	98.1
MatchDG	97.1 (0.1)	99.1 (0.1)	98.9 (0.1)	99.2 (0.2)	99.0 (0.2)	96.5 (0.2)	98.3
PerfMatch	97.8 (0.2)	99.2 (0.2)	99.1 (0.1)	99.2 (0.1)	99.1 (0.1)	97.2 (0.3)	98.6

C.4 Quality of representation learnt in the classification phase

In addition to Table 2 that shows metrics for Phase 1 of MatchDG, we compute the metrics for the classification phase (Phase 2) of MatchDG. Specifically, we compute the Overlap, Top-10 overlap and the Mean Rank metrics (Section B.2) for matched pairs of inputs based on the representation learnt at the end of the classification phase.

Table 11 shows the matching metrics for MatchDG and compares it to the matches based on the representations (last layers) learnt by the ERM-PerfMatch and ERM-RandMatch methods. For both Rotated-MNIST and Fashion-MNIST datasets, MatchDG obtains mean rank, Top 10 overlap and total overlap between ERM-PerfMatch and ERM-RandMatch. As the Fashion-MNIST dataset is more complex than the digits dataset, we observe that the mean rank with different training techniques is higher than the corresponding values for the Rotated-MNIST dataset.

C.5 Matching metrics for Fashion-MNIST dataset with 2000 training samples per domain

In the main text (Table 2), we computed matching metrics for MatchDG (Phase 1) over the Fashion-MNIST dataset with 10000 samples per domain. Here we compute the same metrics for a smaller dataset with 2000 samples per domain.

We compute the metric for the default instantiation of Phase 1 of MatchDG initialized with random matches and compare it to an *oracle* version of MatchDG initialized with perfect matches. In addition, we compare the metrics for matches generated using baseline ERM (last layer of the network) in

Table 10: Accuracy results using a fraction of perfect matches during training

	MNIST	Fashion-MNIST
RandMatch	95.9 (0.18)	79.4 (0.81)
Approx 25%	96.2 (0.35)	80.2 (1.21)
Approx 50%	96.3 (0.51)	81.8 (0.91)
Approx 75%	96.6 (0.10)	83.3 (0.63)
PerfMatch (100%)	97.5 (0.17)	86.2 (0.69)

Table 11: Mean rank, Top-10 overlap, and overlap metrics for the matches learnt in the classification phase (Phase 2), when trained on all five source domains in the Rotated MNIST and FashionMNIST datasets.

Dataset	Method	Overlap (%)	Top 10 Overlap (%)	Mean Rank
Rotated MNIST	RandMatch	2.7 (0.58)	13.7 (1.88)	76.2 (4.61)
	MatchDG (Phase 2)	23.9 (2.24)	48.7 (4.01)	34.6 (3.66)
	PerfMatch (Oracle)	88.6 (1.57)	98.5 (0.31)	0.59 (0.13)
Fashion MNIST (10k)	RandMatch	1.0 (0.13)	5.6 (0.42)	346.0 (12.80)
	MatchDG (Phase 2)	2.3 (0.03)	11.1 (0.06)	241.0 (6.14)
	PerfMatch (Oracle)	7.4 (0.40)	26.9 (0.93)	114.7 (3.04)

order to understand its effectiveness as a matching strategy in Phase 1. Table 12 shows the metrics for Phase 1 of MatchDG with 2K images from the Fashion-MNIST dataset, and reproduces the metrics for the 10K dataset from Table 2 for ease of comparison. We observe that the mean rank of perfect matches improves for the smaller dataset. Similarly, the overlap and top-10 overlap also increase for the smaller dataset. A possible reason is that there are fewer alternative matches to the perfect match as the number of samples is reduced. That said, while the overlap with perfect matches may decrease as sample size increases, the accuracy of the resultant classifier may still increase due to higher sample size.

Table 12: Metrics computed at MatchDG (Phase 1) for Fashion-MNIST dataset with 2K and 10K sample size used for training. Lower is better for mean rank.

Dataset	Method	Overlap (%)	Top 10 Overlap (%)	Mean Rank
Fashion MNIST (2k)	ERM	8.7 (0.14)	36.0 (1.41)	36.1 (1.66)
	MatchDG (Default)	61.5 (2.11)	88.2 (0.91)	5.9 (0.54)
	MatchDG (PerfMatch)	69.5 (10.8)	91.9 (5.8)	3.3 (2.4)
Fashion MNIST	ERM	3.1 (0.20)	14.4 (0.68)	190.6 (7.92)
	MatchDG (Default)	23.9 (2.61)	50.1 (3.29)	79.7 (9.91)
	MatchDG (PerfMatch)	54.7 (4.38)	82.5 (3.07)	15.5 (3.54)

C.6 Iterative updating of matches in Phase-1 of MatchDG

In Section 3.2, we proposed Phase 1 of the MatchDG algorithm with iterative updates to the computed matches. Here we compare the quality of matches learnt at the end of Phase 1 with or without using the iterative updating. Without the iterative updates, the matches always remain the same as the random matches with which the algorithm was initialized.

Table 13 shows metrics computed at the end of Phase 1 of MatchDG using both an iterative approach vs. a non-iterative approach. The iterative approach provides a $3\times$ improvement on the overlap with perfect matches for rotated MNIST and Fashion-MNIST datasets. Since higher overlap in the inferred

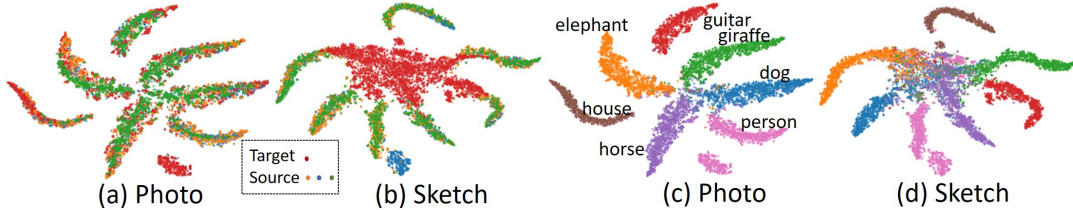


Figure 4: The t-SNE plots for visualizing features learnt in MatchDG Phase 1. (a)-(c) are for Photo as the target domain and (b)-(d) are for Sketch.

matches results in better classification accuracy in Phase 2 (as shown in Table 10), we conclude that using the iterative approach improves the domain generalization capability of MatchDG.

Table 13: Overlap with perfect matches. top-10 overlap and the mean rank for perfect matches for Iterative and Non Iterative MatchDG over all training domains. Lower is better for mean rank.

Dataset	Method (Phase 1)	Overlap (%)	Top 10 Overlap (%)	Mean Rank
MNIST	MatchDG(Iterative)	35.1 (5.23)	69.6 (5.97)	14.3 (4.16)
	MatchDG(Non Iterative)	13.5 (0.80)	38.0 (1.27)	39.9 (1.99)
Fashion	MatchDG(Iterative)	23.9 (2.61)	50.1 (3.29)	79.7 (9.91)
MNIST (10k)	MatchDG(Non Iterative)	7.3 (0.41)	23.7 (1.55)	150.9 (7.77)

D Additional Evaluation on PACS

D.1 AlexNet Results

Finally, we compare RandMatch and MatchDG to prior work on generalization accuracy for the PACS dataset using the AlexNet architecture. As in Table 3, the task is to generalize to a test domain after training on the remaining three domains.

For all test domains, Table 14 shows that both RandMatch and MatchDG outperform the baseline ERM method. Averaging over the test domains, MatchDG obtains slightly higher accuracy than RandMatch (72.08 versus 71.18). Also, MDGHybrid provides improvement over MatchDG (72.73 versus 72.08). Moreover, on average MatchDG, MDGHybrid are better than many previous approaches [26, 24, 35, 13, 36, 55, 32], but some other methods like MASF [20] achieve higher accuracy than MatchDG. Since MatchDG outperforms all prior work on the same dataset when trained using ResNet-18, ResNet-50 architecture (Table 3), we speculate that MatchDG requires a powerful underlying network architecture to use matches effectively for classification.

D.2 T-SNE Plots

Beyond accuracy, we investigate the quality of representations learnt by MatchDG using t-SNE [56] in Figure 4. Comparing the Phase I models for the easiest (*Photo*) and hardest (*Sketch*) unseen domains (Figs. 4a,b), we find that MatchDG achieves a higher overlap between train and test domains for *Photo* than *Sketch*, highlighting the difficulty of generalizing to the *Sketch* domain, even as classes are well-separated in the training domains for both models (Figs. 4c,d).

E Evaluation on Chest X-ray

Table 4 provides the results for evaluation on Chest X-ray dataset, with NIH, ChexPert as the source domains and RSNA being the target domains. Since we have the spurious correlations among the source domains, it could lead to models relying on the unstable trend related to the difference in the position of lungs between class (0) and class (1). This could lead to high accuracy on the source domains, however, the models would not perform well on the target domain as no such correlation

Table 14: Accuracy results on the PACS dataset trained with Alexnet (Default), Alexnet with test domain validation. The results for DBADG [24], MTSSL [57], CIDDG [13], HEX [58], Feature-Critic [55], MLDG [35], REx [32], CAADG [59], Epi-FCR [36], MASF [20] were taken from the DomainBed [29] paper. The results for DANN [3], IRM [6], G2DM [47] were taken from the G2DM paper. The results for D-MTAE [26], MetaReg [48], JiGen [34] were taken from the respective paper.

Method/Test Domain	Photo	Art Painting	Cartoon	Sketch	Average
ERM	86.4 (0.41)	64.3 (1.03)	66.9 (0.43)	62.4 (1.63)	70.0
D-MTAE	91.12	60.27	58.65	47.68	64.45
DANN	88.10	63.20	67.50	57.00	69.00
DBADG	89.50	62.86	66.97	57.51	69.21
MTSSL	84.31	61.67	67.41	63.91	69.32
CIDDG	78.65	62.70	69.73	64.45	69.40
HEX	87.90	66.80	69.70	56.30	70.20
FeatureCritic	90.10	64.40	68.60	58.40	70.40
MLDG	88.00	66.23	66.88	58.96	70.71
REx	89.74	67.04	67.97	59.81	71.14
CAADG	89.16	65.52	69.90	63.37	71.98
Epi-FCR	86.1	64.7	72.3	65.0	72.0
IRM	89.97	64.84	71.16	63.63	72.39
MetaReg	91.07 (0.41)	69.82 (0.76)	70.35 (0.63)	59.26 (0.31)	72.62
JiGen	89.00	67.63	71.71	65.18	73.38
G2DM	88.12	66.60	73.36	66.19	73.55
MASF	90.68	70.35	72.46	67.33	75.21
RandMatch	87.4 (0.52)	66.5 (1.14)	68.4 (1.62)	62.4 (1.35)	71.18
MatchDG	88.67 (0.86)	67.14 (0.83)	69.08 (0.53)	63.40 (1.87)	72.08
MDGHybrid	88.94 (0.42)	67.58 (0.31)	69.57 (0.36)	64.84 (1.98)	72.73
RandMatch (Test)	89.0 (0.47)	67.3 (0.32)	70.7 (0.56)	65.6 (1.08)	73.15
MatchDG(Test)	88.93 (0.21)	67.95 (0.34)	70.97 (0.29)	66.25 (1.49)	73.53
MDGHybrid (Test)	89.02 (0.18)	68.85 (0.66)	71.99 (0.28)	66.48 (0.22)	74.09

in present in the target domain. We observe in Table 4, models like ERM, IRM get higher accuracy on source domains, while they obtain worse accuracy on the target domain as compared to the other methods. This implies that ERM, IRM rely more on the unstable correlations present in the source domains.

The failure of IRM to capture stable trends can be attributed to the presence of spurious correlations without any noise or variation across source domains. This leads to obtaining optimal accuracy on source domains by relying on the spurious features, which would satisfy the IRM penalty as it relies on the loss on the source domains to learn invariant predictors. Hence, minimizing the IRM penalty does not lead to recovering invariant features in this case. In the case of evaluation of IRM on colored MNIST as in the original paper [6], the spurious correlation among source domains was present with some noise, which led to IRM penalty recovering the stable/invariant features. In contrast, matching-based methods learn to build a representation using more of the causal features since the regularization objective aims to reduce the differences in representation between inputs belonging to the same object.