

A. Synthetic Data (Slab Dataset and Simple Counter-example)

A.1. Implementation Details for the Slab Dataset

Dataset The synthetic slab dataset (Section 3.2) consists of a binary label y and 2-dimensional features; one feature has a linear relationship with y while the other has a more complex “slab” relationship with y . The features vary in their simplicity, a measure of the simplicity of the feature is given by the number of linear pieces in the optimal classification/decision curve (Figure 1, (Shah et al., 2020)). Hence, the linear features are simpler as they only have 1 linear piece in the optimal decision boundary, as opposed to the slab features that have k linear pieces in the piecewise linear optimal decision boundary.

The synthetic slab dataset was introduced for detecting simplicity bias in neural networks (Shah et al., 2020), to demonstrate that neural networks trained with SGD learn the simpler linear feature as opposed to the slab feature. We extend this dataset for the domain generalization (DG) task, by making the linear block features spurious due to domain-dependent noise addition as described below. The effect of the slab feature on y remains the same across domains. Presence of the spurious linear feature should enable an ideal DG algorithm to differentiate it from the stable slab feature, and break the simplicity bias in neural networks. However, the invariance introduced by different DG methods can have a big impact. Using this dataset, we show that the class-conditional distribution matching constraint is not sufficient (section 3.2); it is possible to satisfy the constraint using the spurious linear feature too.

The linear block features contain the positive ($y=1$) and the negative ($y=0$) labels sampled from uniform distributions $U(0.1, 1.0)$ and $U(-1.0, -0.1)$ respectively. To make the linear features spurious, we add noise to the linear features s.t. data points are sampled from $U(-0.1, 0.1)$ with probability p , and sampled from $U(0.1, 1.0)$ ($y=1$) or $U(-1.0, -0.1)$ ($y=0$) with probability $1 - p$. The k -slab feature ranges from $[-1, 1]$ and within it has different “slabs” corresponding to uniform distributions of the feature’s value, conditioned on class label y . The labels for these slabs alternate between positive ($y=1$) and negative ($y=0$) as the numeric value of the slab feature increases. The length for each slab is given by $\frac{2-m*(k-1)}{k}$ where k is the total number of slabs, and m is the margin between two slabs.

Linear Block Feature:

$$p_l(x|y=0) = \begin{cases} U(-0.1, 0.1) & \text{with prob. } p \\ U(-1.0, -0.1) & \text{with prob. } 1 - p \end{cases}$$

$$p_l(x|y=1) = \begin{cases} U(-0.1, 0.1) & \text{with prob. } p \\ U(0.1, 1.0) & \text{with prob. } 1 - p \end{cases}$$

Slab Block Feature:

Let k be the total number of slabs, and m be the margin between two slabs.

$$\text{Slab length: } L = \frac{2-m*(k-1)}{k}$$

$$\text{Start index: } I(i) = -1 + i * L$$

$$p_s(x|y=0) = \begin{cases} U(I(i), I(i) + L) & \text{for } i \in \{0, 2, 4, \dots\} \end{cases}$$

$$p_s(x|y=1) = \begin{cases} U(I(i), I(i) + L) & \text{for } i \in \{1, 3, 5, \dots\} \end{cases}$$

We also add a constant (domain independent) noise to the relationship between the slab feature and the class label by flipping the original label with probability p_s .

Source and Target Domains We generate two source data domains with noise probabilities p as 0.0 and 0.1, and generate the target domain with complete noise $p = 1.0$, rendering the linear block feature not informative of the label in the target domain. However, the slab block features have a stable relationship with the labels across the multiple source and target domains. We choose $k = 7$, $m = 0.1$, and $p_s = 0.1$ for the slab block features in our experiments. We sample $1k$ data points per domain, which leads to $2k$ training data points (source domain with p as 0.0 and 0.1), and $1k$ test data points (target domain with p as 1.0). Also, for hyperparameter tuning (model selection), we sample additional 250 data points per source domain as the validation set.

Model Architecture The overall architecture consists of a representation network along with a classification network, detailed below. Input Dim refers to the input data dimension, which is 2 dimensional (linear block feature, slab block feature). Num Classes refers to the total number of output classes, which is binary classification for the synthetic slab dataset. We refer a fully connected dense layer by FC layer, with the input and output dimensions for that layer in brackets.

Representation Network

- FC layer: (Input Dim, 100)
- ReLU activation

Classification Network

- FC layer: (100, 100)

Table 7. Hyper parameter selection details for the slab dataset. 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 the validation loss, with the total number of epochs for each model as 100.

Method	Hyper Parameter	Optimal Value	Range
DANN	Lambda	0.01	[0.01, 0.1, 1.0, 10.0, 100.0]
	Gradient Penalty	0.1	[0.01, 0.1, 1.0, 10.0]
	Discriminator Steps	4	[1, 2, 4, 8]
CDANN	Lambda	0.01	[0.01, 0.1, 1.0, 10.0, 100.0]
	Gradient Penalty	1.0	[0.01, 0.1, 1.0, 10.0]
	Discriminator Steps	2	[1, 2, 4, 8]
MMD	Lambda	0.1	[0.1, 1.0, 10.0]
C-MMD	Lambda	0.1	[0.1, 1.0, 10.0]
CORAL	Lambda	0.1	[0.1, 1.0, 10.0]
C-CORAL	Lambda	0.1	[0.1, 1.0, 10.0]
RandMatch	Lambda	1.0	[0.1, 1.0, 10.0]
PerfMatch	Lambda	1.0	[0.1, 1.0, 10.0]

- FC layer: (100, Num Classes)

For methods like DANN (Ganin et al., 2016), and CDAAN (Li et al., 2018d), which also require domain discriminators, we use the same architecture for them as that of the classification network.

Methods We use Cross-Entropy for the classification loss in ERM and all the other methods. The regularization penalty of all the methods is placed on the output of the representation network.

For the methods DANN (Ganin et al., 2016), CDANN (Li et al., 2018d), MMD (Li et al., 2018b), CORAL (Sun & Saenko, 2016), we used their implementation available in DomainBed (Gulrajani & Lopez-Paz, 2020). We extended the implementation of MMD and CORAL from DomainBed to their class conditional versions, C-MMD, and C-CORAL. The extension to class conditional version was done by computing their respective penalty over domains conditioned on a particular class label.

For RandMatch (Motiian et al., 2017) and PerfMatch, we use l_2 distance for dist in (Eq: 3). The match function Ω in the RandMatch algorithm is defined as randomly matching any two data points across domains with the same class label. For the PerfMatch algorithm, the match function Ω accepts two data points across domains with the same slab id as valid matches (the slab id corresponds to the value of the causal feature: two inputs with the same slab id have similar causal features).

Note on method selection Our objective with the synthetic slab dataset is to compare the performance of conditional distribution matching (CDM) methods to that of

Perfect Match. We chose the above mentioned CDM methods for experimentation since the other CDM methods (Li et al., 2018c; Ghifary et al., 2016; Hu et al., 2019) mentioned in the related works (Section 2, main paper) did not have their implementation publicly available. We found the implementation of CDANN (Li et al., 2018d) in the DomainBed (Gulrajani & Lopez-Paz, 2020) repository, which also provided implementation for the unconditional distribution matching methods like MMD (Li et al., 2018b), and CORAL (Sun & Saenko, 2016). Hence, we extended MMD and CORAL to their class-conditional variant using their original implementation from the DomainBed repository.

Hyperparameter Tuning All the methods were trained using SGD, with batch size 128, learning rate 0.1 and weight decay $5e-4$. We train each method for 100 epochs and do early stopping based on the validation loss.

Further details regarding the tuning of hyperparameters specific to each method’s regularization technique are provided in Table 7. The loss objective of all the methods can be written as $ERM + \lambda * \text{Regularization Penalty}$; and we provide the optimal values and grid range for the hyperparam λ in Table 7. Also, some methods like DANN, C-DANN have additional hyperparams, which are specified in the same table. The grid search range for methods that were implemented using DomainBed (Gulrajani & Lopez-Paz, 2020) is taken from the Table 8 in the DomainBed paper.

A.2. Simple counter-example and its relationship to the MatchDG assumption.

The MatchDG method depends on Assumption 1 (Section 5) which requires that same-class inputs across domains are closer in causal features than different-class inputs. Note

that the example in Section 3.1 does not satisfy this assumption. However, there exist many variations of the setup that do follow the MatchDG assumption, and still class-conditional methods cannot recover the true causal feature. For instance, by setting $|x'_c| = |x_c| + \kappa$ where $\kappa > 1.5$ and $\alpha_1 = \kappa + 1, \alpha_2 = \kappa + 2$ for domain 1 and domain 2 respectively, the train domains satisfy the MatchDG assumption.

Overall, the goal of the simple example in Section 3.1 is to show that there exist datasets where class-conditional methods would not work, but Perfect-Match does. MatchDG’s assumption works in a subset of these datasets. In future work, matching-based methods can be developed that relax the MatchDG assumption.

B. Theory and Proofs

B.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. (Gong et al., 2016; Magliacane et al., 2018; Rojas-Carulla et al., 2018) assume a generative process where the true class label determines the features in the observed data. In contrast, (Peters et al., 2016; Arjovsky et al., 2019) 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 (Arjovsky et al., 2019).

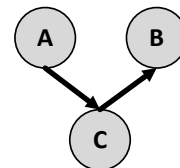
Given these differences, we construct a causal graph (Figure 2) that includes both Y_{true} and Y (as in (Heinze-Deml & Meinshausen, 2019)), and is consistent with both viewpoints about the direction of the causal mechanism. Importantly, all d-separation results reported in the main text hold true irrespective 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

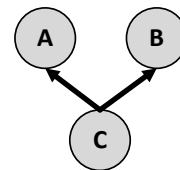
text hold true irrespective of whether there exists an edge from O to X_A .

B.2. D-separation

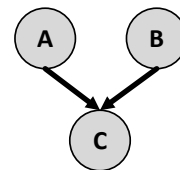
We first expand on the d-separation definition, providing a few examples that illustrate conditional independence implications of specific graph structures in Figure 4. We use these three conditions for the proofs below.



(a) Chain: $A \not\perp B$
 $A \perp B | C$



(b) Fork: $A \not\perp B$
 $A \perp B | C$



(c) Collider: $A \perp B$
 $A \not\perp B | C$

Figure 4. 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 .

Definition 1. d-separation (Pearl, 2009): 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 : $dsep(A, B, C) \Rightarrow A \perp B | C$.

B.3. Proof of Proposition 1

Proposition 1 relates to the domain generalization setup, as described in Section 3.1, where the causal feature determines y label without any noise. The distribution of the non-causal feature varies across domains. The proof uses

the entropy formulation of distribution-matching methods, as done by Akuzawa et al. (2019).

Proposition 1. *Under the domain generalization setup as above, if $P(X_c|Y)$ remains the same across domains where x_c is the stable feature, then the class-conditional domain-invariant objective for learning representations yields a generalizable classifier such that the learnt representation $\Phi(\mathbf{x})$ is independent of the domain given x_c . Specifically, the entropy $H(d|x_c) = H(d|\Phi, x_c)$.*

Proof. We can write class-conditional invariant models as optimizing two objectives: minimize the error on the training data (ERM objective), and learn a representation $\Phi(\mathbf{x})$ that is independent of domain given the class label (class-conditional invariant).

Let us focus on the second objective, which be interpreted as 2) maximizing the entropy of domain given class label and representation $H(d|y, \Phi(\mathbf{x}))$. Let Φ_2 be the optimal representation for the class-conditional invariant. We can write,

$$\Phi_2 = \arg \max_{\Phi} H(d|y, \Phi(\mathbf{x})) \quad (5)$$

Since $H(d|y, \Phi(\mathbf{x})) \leq H(d|y)$ using the property of entropy, the optimal Φ_2 satisfies,

$$H(d|y, \Phi_2(\mathbf{x})) = H(d|y) \quad (6)$$

Now two cases arise; $x_c \perp\!\!\!\perp D|Y$ or $x_c \not\perp\!\!\!\perp D|Y$. We assume the former. If X_C is independent of domain conditioned on the class label, then

$$H(d|y) = H(d|y, x_c) \quad (7)$$

Here domain d is independent of both x_c and $\Phi_2(\mathbf{x})$, conditional on y . Since causal features x_c cannot be caused by the representation $\Phi_2(\mathbf{x})$, it cannot be a collider (Definition 1) in any graph connecting d , x_c and $\Phi_2(\mathbf{x})$. Therefore, conditioning on it does not remove the independence between d and $\Phi_2(\mathbf{x})|y$ (conditioned on y). Hence, we condition on Eq 6 with x_c and obtain,

$$H(d|y, x_c) = H(d|y, x_c, \Phi_2(\mathbf{x})) \quad (8)$$

Plugging it into the above equations, we obtain

$$H(d|y) = H(d|y, x_c) = H(d|y, x_c, \Phi_2(\mathbf{x})) \quad (9)$$

Also since there is no label noise, x_c can achieve zero error for predicting the label y . That is, x_c contains all information about y , and thus we can remove y from the above equation,

$$H(d|x_c) = H(d|\Phi_2(\mathbf{x}), x_c) \quad (10)$$

This implies that the learnt representation $\Phi_2(\mathbf{x})$ is independent of the domain given x_c ; thus $\Phi_2(\mathbf{x})$ depends on x_c and not on any other feature that changes with domain. \square

B.3.1. REMARKS BASED ON PROPOSITION 1

If x_c is not independent of domain given class label.

However, if X_C is not independent of domain given the class label (i.e., $P(x_c|y)$ changes across domains), then $H(d|y) > H(d|y, x_c)$. Using the equality from Eq 8, we obtain,

$$\begin{aligned} H(d|y) &= H(d|y, \Phi_2(\mathbf{x})) > H(d|y, x_c) \\ H(d|y, \Phi_2(\mathbf{x})) &\geq H(d|\Phi_2(\mathbf{x}), y, x_c) \end{aligned} \quad (11)$$

After removing y as in Eq. 10, $H(d|x_c)$ and $H(d|\Phi_2, x_c)$ may not be equal. In particular, the ground-truth representation $\Phi_{GT}(\mathbf{x}) = x_c$ does not satisfy the class-conditional invariant: $H(d|y, \Phi_{GT}(\mathbf{x})) = H(d|y, x_c) \neq H(d|y)$.

Hence, to learn x_c as the representation, we need a separate constraint, $H(d|x_c) = H(d|\Phi, x_c)$; i.e. domain and representation should be independent conditioned on x_c .

Implications for the slab dataset (Section 3.2). In the slab dataset, $x_c = x_2$ and x_2 is independent of the domain given the class label ($X_2 \perp\!\!\!\perp D|Y$). We also see that $\Phi(\mathbf{x}) = x_2$ satisfies the class-conditional invariant: $H(d|y, x_2) = H(d|y)$ since $X_2 \perp\!\!\!\perp D|Y$. By Proposition 1, the class-conditional invariant should lead to a representation that satisfies $H(d|x_c) = H(d|\Phi(\mathbf{x}), x_c)$. However, the same constraint can also be achieved by setting $\Phi(\mathbf{x}) = x_1$ by shifting the distribution of x_1 slightly. And since there is a simple, linear correlation between x_1 and the class label y , empirically class-conditional methods end up learning a representation dependent on x_1 .

B.4. Proof of Proposition 2

Proposition 2. *Given observed data distribution $P(Y, X, D, O)$ that may also include data obtained from interventions on domain D , multiple values of X_C yield exactly the same observational and interventional distributions and hence X_C is unidentifiable.*

Proof. To prove non-identifiability, it is sufficient to show a counter-example where the same structural equations (and hence same observed and interventional distributions over Y, X, D, O) correspond to two different values of X_C .

From Section 4.1, the SCM leads to the following structural equations,

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

Substituting for \mathbf{x}_c in the SCM equations, we obtain,

$$\begin{aligned} y &= h(g_{x_c}(o), \epsilon_y) \\ \mathbf{x} &= g_x(g_{x_c}(o), g_{x_a}(d, o, \epsilon_{x_a}), \epsilon_x) \end{aligned} \quad (12)$$

Given a value of object variable o , note that g_{x_c} determines x_c . We now proceed to show that different values of g_{x_c} are possible given the same structural equations between the observed variables Y, X, D, O . Specifically, by choosing g_x and h appropriately, different values of g_{x_c} can lead to the same observed values for (y, d, o, x) .

A simple counter-example. Suppose the following SCM equations,

$$\begin{aligned} y &= h(g_{x_c}(o)) \\ x &= g_1(g_{x_c}(o)) + g_2(o, d) \end{aligned} \quad (13)$$

Introducing $h^* = h \circ g_{x_c}$ and $g_1^* = g_1 \circ g_{x_c}$, we can rewrite the above equations as,

$$\begin{aligned} y &= h^*(o) \\ x &= g_1^*(o) + g_2(o, d) \end{aligned} \quad (14)$$

then any g_{x_c} is applicable as long as we set h such that $h(g_{x_c}(o)) = h^*(o)$ and set g_1 such that $g_1(g_{x_c}(o)) = g_1^*(o)$. In particular, if the SCM equations are $y = o, x = o + o * d$, and we define $h = g_1 = g_{x_c}^{-1}$, then g_{x_c} can be any invertible function. Hence, different values of $x_c = g_{x_c}(o)$ will lead to the same structural equations over Y, X, D, O , and therefore the same observed and interventional distributions. \square

B.5. 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 (1).*

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 (Miller et al., 1993) 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 .*

$$\begin{aligned} f_{\text{perfectmatch}} &= \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) + \\ &\lambda \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \end{aligned} \quad (3)$$

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

$$C(\Phi) = \min_{\Phi} \sum_{d, d' \in D_m} \lim_{n_d \rightarrow \infty} \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \quad (15)$$

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 2(b), 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 (1). From Sections 4.1 and 4.2, we know that $X_c \perp\!\!\!\perp D | O$ and that $x_c = g_{x_c}(o)$. Thus, x_c is the same for inputs from the same object and we can write:

$$\text{dist}(\mathbf{x}_{c,j}^{(d)}, \mathbf{x}_{c,k}^{(d')}) = 0 \quad \forall d, d' \in D_m \text{ such that } \Omega(j, k) = 1 \quad (16)$$

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 2(b), a feature vector $X_a \subseteq X_A$ is independent of X_c given the object, $X_a \perp\!\!\!\perp X_c | O$, and $x_a = g_{x_a}(d, o, \epsilon_{x_a})$. 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 (3) ensures that the optimal solution contains only those functions of X_C that also maximize accuracy.

Using (2), 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):

$$\begin{aligned} \hat{f}_{pmatch} &= \arg \min_{h, \Phi} \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} L_d(h(\Phi(X)), Y) \quad (17) \\ \text{s.t.} \quad &\sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) = 0 \end{aligned} \quad (18)$$

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

where $\psi(X_c)$ denotes all functions of X_c that are optimal for (15), 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:

$$\begin{aligned} \arg \min_{f \in \mathcal{F}} \lim_{n_d \rightarrow \infty} \frac{1}{n_d} L_d(f(X_c), Y) &= \arg \min_{f \in \mathcal{F}} \mathbb{E}[l(f(\mathbf{x}_c), y)] \\ &= \mathbb{E}[Y|X_c] \quad \forall d \in D_m \end{aligned} \quad (20)$$

Now (19) can be rewritten as,

$$\begin{aligned} \hat{f}_{pmatch} &= \arg \min_f \frac{1}{n} \sum_{d=1}^m \lim_{n_d \rightarrow \infty} \frac{L_d(f(X_c), Y)}{n_d} n_d \\ &= \arg \min_f \sum_{d=1}^m \lim_{n_d \rightarrow \infty} \frac{L_d(f(X_c), Y) n_d}{n_d n} \end{aligned} \quad (21)$$

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:

$$\begin{aligned} f^* &= \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(d, \mathbf{x}, y)} [l(y, f(\mathbf{x}))] \\ &= \arg \min_{h \in \mathcal{F}} \mathbb{E}_{(d, \mathbf{x}, y)} [l(y, h(\mathbf{x}_c))] \quad (22) \\ &= \arg \min_{h \in \mathcal{F}} \mathbb{E}_{(\mathbf{x}, y)} [l(y, h(\mathbf{x}_c))] = \mathbb{E}[Y|X_c] \end{aligned}$$

where we utilize (20) and that the loss function is P-admissible. Hence, $f^* = \mathbb{E}[Y|X_c]$ is the loss-minimizing function for the loss in (19).

Finally, using a Lagrangian multiplier, minimizing the following soft constraint loss is equivalent to minimizing (18), for some value of λ .

$$\begin{aligned} \hat{f}_{pmatch} &= \lim_{\forall d \in D_m, n_d \rightarrow \infty} \arg \min_{h, \Phi} \sum_{d=1}^m L_d(h(\Phi(X)), Y) \\ &\quad + \lambda \sum_{\Omega(j,k)=1; d \neq d'} \text{dist}(\Phi(\mathbf{x}_j^{(d)}), \Phi(\mathbf{x}_k^{(d')})) \quad (23) \end{aligned}$$

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.

B.6. Proof of Proposition 3

Domain-invariant representations. ($\Phi(\mathbf{x}) \perp\!\!\!\perp D$) (Muan-det et al., 2013; Li et al., 2018b; Ganin et al., 2016). Using d-separation on the SCM from Figure 2(b), $X_C \perp\!\!\!\perp D$ is not sufficient since O blocks the path between X_C and D . While (Zhao et al., 2019) 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. ($\Phi(\mathbf{x}) \perp\!\!\!\perp D|Y$) (Li et al., 2018c; Ghifary et al., 2016; Li et al., 2018d) 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 .

Below we prove these results formally.

Proposition 3. *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 2(b), 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 (1). From Sections 4.1 and 4.2, 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 $\Phi_{DI}(\mathbf{x}) \perp\!\!\!\perp D$. But using the d-separation criteria on the SCM in Figure 2(b), 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 $\Phi_{CDI} \perp\!\!\!\perp D|Y$. 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

C. 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.

C.1. Implementation details

For the implementation of ERM-PerfMatch in Eq. (3); we use the cross-entropy loss for L_d and l_2 distance for dist in Eq. (3). Similarly, we implement the ERM-RandMatch with a match function Ω in Eq. (3) that randomly matches data points across domains with the same class. 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. (3) 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.

Also, given a fixed data point, the match function Ω could select multiple data points as potential matches for it. In this case we use Eq. (3) with stochastic matching, where we randomly select one match out of the potential multiple matches.

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 C.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 C.3, Table 9.

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° (Ghifary et al., 2015), where each rotation angle represents a domain and the task is to predict the class label. For Table 2, we follow the setup in CSD (Piratla et al., 2020), 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 (Piratla et al., 2020; Motiian et al., 2017).

Further, we also evaluate on other setups of Rotated MNIST in prior works (Motiian et al., 2017; Gulrajani & Lopez-Paz, 2020), which involve six domains ($0^\circ, 15^\circ, 30^\circ, 45^\circ, 60^\circ, 75^\circ$), 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 11) as per the setup proposed by (Motiian et al., 2017). Similarly, we sample all the 70,000 images in MNIST and evaluate using the custom architecture (Table 12) as per the setup proposed by (Gulrajani & Lopez-Paz, 2020).

Another important distinction between different setups above is the use of different digits for the source and the target domains ((Piratla et al., 2020), (Gulrajani & Lopez-Paz, 2020)), as opposed to the use of same digits across the source and the target domains in setup of (Motiian et al., 2017) 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 (Dou et al., 2019), we train 4 models with each domain as the target using Resnet-18 (Table 4), Resnet-50 (Table 5) and Alexnet (Table 18), with each architecture pre-trained on ImageNet. We also the following data augmentations (Gulrajani & Lopez-Paz, 2020) 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 (Wang et al., 2017), ChexPert (Irvin et al., 2019) and RSNA (rsn, 2018). 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. The dataset size for the different splits on each domain are described below:

- NIH: Train (800), Validation (200), Test (400)
- ChexPert: Train (800), Validation (200), Test (400)
- RSNA: Train (800), Validation (200), Test (400)

Following prior works (Cohen et al., 2020), we use the pre-trained DenseNet-121 architecture for classification. We use the following data augmentations: Random Crop and Random Horizontal Flip. We further create spurious correlations, all the images of the class 0 in the training domains are translated vertically downwards; while no such translation is done for the test domain. We translate the images in each source domain by a fixed amount, which varies over different source domains (NIH (45), ChexPert (35), RSNA (15)). 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.

C.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 samples 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 (4) 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. (3), with the match function Ω based on 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.

C.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”

Table 8. 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 10

Dataset	Hyper Parameter	Optimal Value	Range
Rotated & Fashion MNIST Table 2 (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 IRM Threshold	1.0 (RotMNIST); 0.05 (FashionMNIST) 5 (RotMNIST), 0 (FashionMNIST)	[0.05, 0.1, 0.5, 1.0, 5.0] [0, 5, 15, 20]
Rotated MNIST Table 11 (LeNet)	Total Epochs	-	100
	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 12 (DomainBed)	Total Epochs	-	25
	Learning Rate	0.01	[0.01]
	Batch Size	128	[16, 32, 64, 128]
	Weight Decay	0.0005	[0.0005]
	Match Penalty	1.0	[0.1, 1.0]
PACS Table 17, 18 (ResNet-18, ResNet-50, AlexNet)	Total Epochs	-	50
	Learning Rate	Table 10	[0.01, 0.001, 0.0005]
	Batch Size	16	[16]
	Weight Decay	0.0005	[0.0005]
	Match Penalty	Table 10	[0.01, 0.1, 0.5, 1.0, 5.0]
Chest X-ray Table 6 (DenseNet-121)	Total Epochs	-	40
	Learning Rate	0.001	[0.01, 0.001]
	Batch Size	16	[16]
	Weight Decay	0.0005	[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]

matches.

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

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \Omega'(j, k)}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (24)$$

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. $S_{\Omega'}^{10}(j)$

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \mathbb{1}[k \in S_{\Omega'}^{10}(j)]}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (25)$$

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. $S_{\Omega'}(j)$

$$\frac{\sum_{\Omega(j,k)=1; d \neq d'} \text{Rank}[k \in S_{\Omega'}(j)]}{\sum_{\Omega(j,k)=1; d \neq d'} 1} \quad (26)$$

C.3. HyperParameter Tuning

To select hyperparameters, prior works (Dou et al., 2019; Carlucci et al., 2019; Li et al., 2018a) 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 MINST, 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

Table 9. 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 2, 11, 12	Total Epochs	50
	Learning Rate	0.01
	Batch Size	64 (Table 2), 512 (Table 11, Table 12)
	Weight Decay	0.0005
	τ	0.05
	Architecture	ResNet-18 (Table 2), LeNet (Table 11), Custom CNN (Table 12)
PACS Table 17, 18	Total Epochs	50
	Learning Rate	0.01
	Batch Size	32
	Weight Decay	0.0005
	τ	0.05
	Architecture	ResNet-50
Chest X-ray Table 6	Total Epochs	50
	Learning Rate	0.01
	Batch Size	32
	Weight Decay	0.0005
	τ	0.05
	Architecture	DenseNet-121

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 8. 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 9). 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.

C.4. Reproducing Results from Prior Work

MNIST and Fashion MNIST The results for MASF, CSD, and IRM in Table 2 were computed using their code which is available online¹²³. The MASF code was hard-coded 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 2 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.

The results for prior approaches in Table 11 are taken from (Shankar et al., 2018), (Ilse et al., 2020). For the results using DomainBed setup in Table 12, the results for prior approaches are taken from (Gulrajani & Lopez-Paz, 2020).

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 17, 18.

Chest X-ray The results for the prior approaches CSD, IRM were generated using the implementations of both of the methods available on github^{1,3}.

D. Additional Evaluation on Rotated MNIST and Fashion-MNIST

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

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

Table 11 compares the accuracy results for MatchDG with prior work on the LeNet architecture (Motiian et al., 2017). In this setup, there are six domains in total

¹<https://github.com/vihari/CSD>

²<https://github.com/biomed-mira/masf>

³<https://github.com/facebookresearch/InvariantRiskMinimization>

Table 10. Optimal values for hyper parameters on PACS. Batch Size (16), Weight Decay (0.0005) 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 4, 17	Learning Rate	Photo	0.001	0.001	0.0005	0.0005
		Art Painting	0.01	0.01	0.001	0.001
		Cartoon	0.01	0.001	0.001	0.001
		Sketch	0.01	0.01	0.01	0.01
	Match Penalty	Photo	0	5.0	1.0	(0.1, 0.1)
		Art Painting	0	0.1	5.0	(0.01, 0.1)
		Cartoon	0	5.0	1.0	(0.1, 0.1)
		Sketch	0	0.5	0.5	(0.01, 0.1)
ResNet-50 Table 5, 17	Learning Rate	Photo	0.0005	0.0005	0.0005	0.0005
		Art Painting	0.01	0.01	0.001	0.001
		Cartoon	0.01	0.01	0.001	0.0005
		Sketch	0.01	0.01	0.0005	0.001
	Match Penalty	Photo	0	5.0	0.01	(0.1, 0.1)
		Art Painting	0	0.1	0.1	(0.01, 0.1)
		Cartoon	0	0.01	0.01	(0.01, 0.1)
		Sketch	0	0.1	5.0	(0.01, 0.1)
AlexNet Table 18	Learning Rate	Photo	0.0005	0.0005	0.0005	0.0005
		Art Painting	0.001	0.001	0.001	0.001
		Cartoon	0.001	0.001	0.001	0.001
		Sketch	0.0005	0.001	0.001	0.001
	Match Penalty	Photo	0	0.1	0.1	(0.1, 0.1)
		Art Painting	0	0.1	1.0	(0.01, 0.1)
		Cartoon	0	0.5	1.0	(0.01, 0.1)
		Sketch	0	0.5	0.1	(0.01, 0.1)

(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 on the all the domains except the test domain 0, where MatchDG is competitive to the best performing approach DIVA. They also achieve accuracy almost equal to the oracle case PerfMatch for target angles (15° to 60°) that lie in between the source domains.

D.2. Comparing MatchDG on Domain Bed Benchmark

Table 12 compares the accuracy results for MatchDG with prior work on the setup proposed by (Gulrajani & Lopez-Paz, 2020). This setup is similar to the setup in the section D.1, however, it uses a custom CNN architecture and all the 70,000 images for each domain. For a fair comparison, we use the same custom CNN architecture for learning the match function during the MatchDG Phase-I. Even under this constraint, MatchDG average accuracy is only 0.5% percent behind the best performing approaches (CORAL, MMD). As supported by our experiments before (Table 2, 3) we believe that using more powerful architectures (ResNet-18, ResNet-50) during the MatchDG Phase-1 should help in learning a better match function and consequently better average accuracy.

D.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 13. 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.

D.4. Quality of representation learnt in the classification phase

In addition to Table 3 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 C.2) for matched pairs of inputs based on the representation learnt at the end of the classification phase.

Table 14 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-

Table 11. Accuracy for Rotated MNIST datasets using the LeNet architecture as proposed in (Motiian et al., 2017). The results for the prior approaches CCSA (Motiian et al., 2017), D-MTAE (Ghifary et al., 2015), LabelGrad (Goodfellow et al., 2014), DAN (Ganin et al., 2016), and CrossGrad (Shankar et al., 2018) are taken from Table 9 in (Shankar et al., 2018). The results for DIVA (Ilse et al., 2020) are taken from the Table 1 in their paper.

Algorithm	0	15	30	45	60	75	Average
ERM	88.2 (1.0)	98.6 (0.5)	97.7 (0.6)	97.5 (0.3)	97.0 (0.1)	85.6 (2.1)	94.1
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	91.0 (0.9)	99.7 (0.2)	99.6 (0.1)	99.4 (0.1)	99.7 (0.1)	93.1 (1.1)	97.1
MatchDG	93.0 (0.5)	99.5 (0.3)	99.9 (0.1)	99.4 (0.1)	99.7 (0.3)	93.3 (1.1)	97.4
PerfMatch	96.5 (0.6)	99.1 (0.3)	99.2 (0.3)	98.6 (0.7)	98.6 (1.0)	94.9 (1.8)	97.8

Table 12. Accuracy for Rotated MNIST datasets using the DomainBed setup as proposed in (Gulrajani & Lopez-Paz, 2020). The results for the approaches IRM (Arjovsky et al., 2019), DRO (Sagawa et al., 2019), Mixup (Xu et al., 2019; Yan et al., 2020; Wang et al., 2020), MLDG (Li et al., 2018a), CORAL (Sun & Saenko, 2016), MMD (Li et al., 2018b), DANN (Ganin et al., 2016), C-DANN (Li et al., 2018d) are taken from (Gulrajani & Lopez-Paz, 2020).

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	95.4 (0.4)	98.2 (0.1)	97.9 (0.5)	98.5 (0.1)	98.1 (0.1)	94.3 (0.3)	97.1
MatchDG	95.9 (0.1)	98.4 (0.1)	98.6 (0.2)	98.9 (0.2)	98.7 (0.1)	95.1 (0.3)	97.6

Table 13. Accuracy results using a fraction of perfect matches during training

	MNIST	Fashion-MNIST
RandMatch	93.4 (0.26)	77.0 (0.42)
Approx 25%	93.8 (0.48)	77.8 (0.79)
Approx 50%	94.0 (0.42)	78.0 (0.78)
Approx 75%	94.7 (0.14)	78.9 (0.31)
PerfMatch (100%)	96.0 (0.41)	81.6 (0.46)

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.

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

In the main text (Table 3), 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 order to understand its effectiveness as a matching strategy in Phase 1. Table 15 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 3 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

Table 14. 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	2.2 (0.18)	13.5 (0.36)	75.5 (1.65)
	MatchDG (Phase 2)	17.7 (0.97)	41.8 (2.89)	39.6 (3.58)
	PerfMatch (Oracle)	78.2 (1.91)	95.5 (1.37)	1.84 (0.67)
Fashion MNIST (10k)	RandMatch	0.5 (0.04)	3.2 (0.17)	420.0 (7.27)
	MatchDG (Phase 2)	1.8 (0.13)	8.5 (0.56)	296.5 (9.94)
	PerfMatch (Oracle)	9.2 (0.21)	30.5 (0.38)	114.7 (3.29)

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.

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

In Section 5.1, 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 16 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 $2\times$ improvement on the overlap with perfect matches for rotated MNIST and Fashion-MNIST datasets. Since higher overlap in the inferred matches results in better classification accuracy in Phase 2 (as shown in Table 13), we conclude that using the iterative approach improves the domain generalization capability of MatchDG.

Table 15. 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)	38.5 (2.11)	71.2 (0.91)	15.9 (0.54)
	MatchDG (PerfMatch)	69.5 (10.8)	91.9 (5.8)	3.3 (2.4)
Fashion MNIST (10k)	ERM	2.1 (0.12)	11.1 (0.63)	224.3 (8.73)
	MatchDG (Default)	17.9 (0.62)	43.1 (0.83)	89.0 (3.15)
	MatchDG (PerfMatch)	56.2 (1.79)	87.2 (1.48)	7.3 (1.18)

Table 16. 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)	28.9 (1.24)	64.2 (2.42)	18.6 (1.59)
	MatchDG (Non Iterative)	12.3 (0.28)	37.9 (0.27)	37.8 (0.47)
Fashion MNIST (10k)	MatchDG (Iterative)	17.9 (0.62)	43.1 (0.83)	89.0 (3.15)
	MatchDG (Non Iterative)	7.7 (0.28)	23.8 (0.78)	153.9 (9.63)

E. Additional Evaluation on PACS

E.1. ResNet Results

Table 17 extends the evaluation on PACS with ResNet-18, ResNet-50 (Table 4, 5) in the main text by adding comparison with more prior approaches. We observe that MDGHybrid beats most of the prior approaches on both the ResNet-18 and ResNet-50 evaluation, except DDEC (Asadi et al., 2019), and RSC (Huang et al., 2020). However, as stated in the main paper, DDEC (Asadi et al., 2019) rely on data from additional source like Behance BAM! dataset and we are also not sure about the validation mechanism used by them. 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.

E.2. 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 17, the task is to generalize to a test domain after training on the remaining three domains.

For all test domains, Table 18 shows that both RandMatch and MatchDG outperform the baseline ERM method. Averaging over the test domains, MDGHybrid provides improvement over MatchDG (70.46 versus 69.91). Moreover, on average MatchDG, MDGHybrid are better than many previous approaches D-MTAE (Ghifary et al., 2015), DBADG (Li et al., 2017), CIDDG (Li et al., 2018d), HEX (Wang et al., 2019) and FeatureCritic (Li et al., 2019b), but some other methods like MASF (Dou et al., 2019), DGER (Zhao et al., 2020), RSC (Huang et al., 2020) achieve higher accuracy than MatchDG. Since MatchDG outperforms most of the prior work on the same dataset when trained using ResNet-18, ResNet-50 architecture (Table 17), we speculate that MatchDG requires a powerful underlying network architecture to use matches effectively for classification.

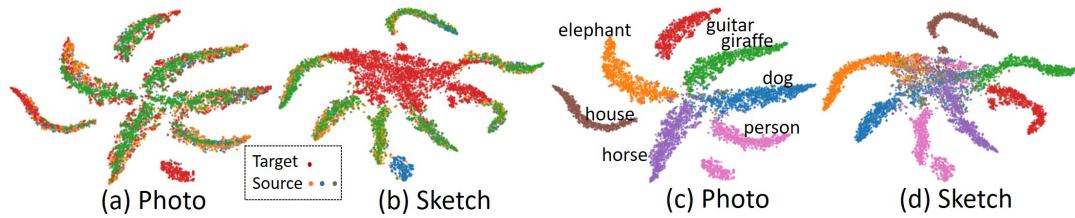


Figure 5. The t-SNE plots for visualizing features learnt in MatchDG Phase 1. (a)-(c) are for Photo as the target domain and (b)-(d) are for Sketch.

E.3. T-SNE Plots

Beyond accuracy, we investigate the quality of representations learnt by MatchDG using t-SNE (Maaten & Hinton, 2008) in Figure 5. Comparing the Phase I models for the easiest (*Photo*) and hardest (*Sketch*) unseen domains (Figs. 5a,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. 5c,d).

Domain Generalization using Causal Matching

Table 17. Accuracy on PACS with ResNet 18 (default, top row set), Resnet 18 with test domain validation (middle row set), and ResNet 50 (bottom row set). The results for JiGen (Carlucci et al., 2019), S-MLDG (Li et al., 2020), D-SAM (D’Innocente & Caputo, 2018), MMLD (Matsuura & Harada, 2020), DDAIG (Zhou et al., 2020) SagNet (Nam et al., 2019), DDEC (Asadi et al., 2019), DANN (Ganin et al., 2016), C-DANN (Li et al., 2018d), DRO (Sagawa et al., 2019), Mixup (Xu et al., 2019; Yan et al., 2020; Wang et al., 2020), IRM (Arjovsky et al., 2019), MLDG (Li et al., 2018a), MMD (Li et al., 2018b), CORAL (Sun & Saenko, 2016), were taken from the DomainBed (Gulrajani & Lopez-Paz, 2020) paper. For G2DM (Albuquerque et al., 2020a), DGER (Zhao et al., 2020), CSD (Piratla et al., 2020), MASF (Dou et al., 2019), EpiFCR (Li et al., 2019a), MetaReg (Balaji et al., 2018), RSC (Huang et al., 2020) it was taken from their respective paper.

	P	A	C	S	Average.
ERM	95.38 (0.86)	77.68 (0.35)	78.98 (0.59)	74.75 (1.70)	81.70
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
DGER	96.65 (0.21)	80.70 (0.71)	76.40 (0.34)	71.77 (1.27)	81.38
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
DDEC	96.93	83.01	79.39	78.62	84.46
RSC	95.99	83.43	80.31	80.85	85.15
RandMatch	95.37 (0.25)	78.16 (1.51)	78.83 (1.18)	75.13 (1.90)	81.87
MatchDG	95.93 (0.21)	79.77 (0.12)	80.03 (0.03)	77.11 (0.35)	83.21
MDGHybrid	96.15 (0.40)	81.71 (0.75)	80.75 (0.50)	78.79 (1.25)	84.35
G2DM (Test)	94.63	81.44	79.35	79.52	83.34
RandMatch (Test)	95.57 (0.17)	79.09 (1.09)	79.37 (0.89)	77.60 (0.87)	82.91
MatchDG (Test)	96.53 (0.05)	81.32 (0.38)	80.70 (0.54)	79.72 (1.01)	84.56
MDGHybrid (Test)	96.67 (0.20)	82.80 (0.32)	81.61 (0.06)	81.05 (1.01)	85.53
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
DGER	98.25 (0.12)	87.51 (1.03)	79.31 (1.40)	76.30 (0.65)	85.34
CORAL (ResNet50)	97.6 (0.0)	87.7 (0.6)	79.2 (1.1)	79.4 (0.7)	86.0
RSC (ResNet50)	97.92	87.89	82.16	83.35	87.83
RandMatch (ResNet50)	97.89 (0.11)	82.16 (0.19)	81.68 (0.45)	80.45 (0.19)	85.54
MatchDG (ResNet50)	97.94 (0.27)	85.61 (0.81)	82.12 (0.69)	78.76 (1.13)	86.11
MDGHybrid (ResNet50)	98.36 (0.06)	86.74 (1.01)	82.32 (0.76)	82.66 (0.48)	87.52

Table 18. Accuracy results on the PACS dataset trained with Alexnet (default, top row set), and Alexnet with test domain validation (bottom row set). The results for DBADG (Li et al., 2017), MTSSL (Albuquerque et al., 2020b), CIDDG (Li et al., 2018d), HEX (Wang et al., 2019), FeatureCritic (Li et al., 2019b), MLDG (Li et al., 2018a), REx (Krueger et al., 2020), CAADG (Rahman et al., 2020), Epi-FCR (Li et al., 2019a), MASF (Dou et al., 2019) were taken from the DomainBed (Gulrajani & Lopez-Paz, 2020) paper. The results for DANN (Ganin et al., 2016), IRM (Arjovsky et al., 2019), G2DM (Albuquerque et al., 2020a) were taken from the G2DM paper. The results for D-MTAE (Ghifary et al., 2015), MetaReg (Balaji et al., 2018), JiGen (Carlucci et al., 2019), DGER (Zhao et al., 2020) and RSC (Huang et al., 2020) were taken from the respective paper.

Method/Test Domain	Photo	Art Painting	Cartoon	Sketch	Average
ERM	85.29 (0.22)	64.23 (0.18)	66.61 (0.88)	59.25 (0.83)	68.85
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
DGER	89.92 (0.42)	71.34 (0.87)	70.29 (0.77)	71.15 (1.01)	75.67
RSC	90.88	71.62	75.11	66.62	76.05
RandMatch	85.42 (0.52)	65.54 (1.14)	68.41 (1.62)	59.46 (1.35)	69.71
MatchDG	85.41 (0.41)	66.21 (0.64)	68.47 (1.10)	59.56 (1.24)	69.91
MDGHybrid	85.67 (0.67)	66.89 (1.23)	68.89 (1.08)	60.39 (2.13)	70.46
RandMatch (Test)	86.04 (0.47)	67.35 (0.32)	69.71 (0.56)	64.66 (1.08)	71.94
MatchDG(Test)	86.52 (0.43)	67.99 (0.56)	69.92 (0.09)	65.64 (1.48)	72.52
MDGHybrid (Test)	87.03 (0.29)	67.97 (0.79)	71.06 (0.43)	67.19 (0.44)	73.31

References

- Kaggle: RsnA pneumonia detection challenge, 2018. URL <https://www.kaggle.com/c/rsna-pneumonia-detection-challenge>.
- Ahuja, K., Shanmugam, K., Varshney, K., and Dhurandhar, A. Invariant risk minimization games. *arXiv preprint arXiv:2002.04692*, 2020.
- Akuzawa, K., Iwasawa, Y., and Matsuo, Y. Adversarial invariant feature learning with accuracy constraint for domain generalization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 315–331. Springer, 2019.
- Albuquerque, I., Monteiro, J., Darvishi, M., Falk, T. H., and Mitliagkas, I. Generalizing to unseen domains via distribution matching, 2020a.
- Albuquerque, I., Naik, N., Li, J., Keskar, N., and Socher, R. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *arXiv preprint arXiv:2003.13525*, 2020b.
- Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.
- Asadi, N., Sarfi, A. M., Hosseinzadeh, M., Karimpour, Z., and Eftekhari, M. Towards shape biased unsupervised representation learning for domain generalization. *arXiv preprint arXiv:1909.08245*, 2019.
- Balaji, Y., Sankaranarayanan, S., and Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pp. 998–1008, 2018.
- Carlucci, F. M., D’Innocente, A., Bucci, S., Caputo, B., and Tommasi, T. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.
- Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.
- Christiansen, R., Pfister, N., Jakobsen, M. E., Gnecco, N., and Peters, J. A causal framework for distribution generalization. *arXiv e-prints*, pp. arXiv–2006, 2020.
- Cohen, J. P., Hashir, M., Brooks, R., and Bertrand, H. On the limits of cross-domain generalization in automated x-ray prediction. *arXiv preprint arXiv:2002.02497*, 2020.
- Dou, Q., de Castro, D. C., Kamnitsas, K., and Glocker, B. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pp. 6447–6458, 2019.
- D’Innocente, A. and Caputo, B. Domain generalization with domain-specific aggregation modules. In *German Conference on Pattern Recognition*, pp. 187–198. Springer, 2018.
- Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.
- Ghifary, M., Bastiaan Kleijn, W., Zhang, M., and Balduzzi, D. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pp. 2551–2559, 2015.
- Ghifary, M., Balduzzi, D., Kleijn, W. B., and Zhang, M. 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.
- Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., and Schölkopf, B. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pp. 2839–2848, 2016.
- Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.
- He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.
- Heinze-Deml, C. and Meinshausen, N. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2019.
- Hu, S., Zhang, K., Chen, Z., and Chan, L. 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.
- Huang, Z., Wang, H., Xing, E. P., and Huang, D. Self-challenging improves cross-domain generalization. *arXiv preprint arXiv:2007.02454*, 2, 2020.

- Ilse, M., Tomczak, J. M., Louizos, C., and Welling, M. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pp. 322–348. PMLR, 2020.
- Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpan-skaya, K., 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, pp. 590–597, 2019.
- Johansson, F. D., Sontag, D., and Ranganath, R. Support and invertibility in domain-invariant representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 527–536, 2019.
- Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Bin-as, J., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). *arXiv preprint arXiv:2003.00688*, 2020.
- Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.
- Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.
- Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.-Z., and Hospedales, T. M. Episodic training for domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1446–1455, 2019a.
- Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. Sequential learning for domain generalization. *arXiv preprint arXiv:2004.01377*, 2020.
- Li, H., Jialin Pan, S., Wang, S., and Kot, A. C. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5400–5409, 2018b.
- Li, Y., Gong, M., Tian, X., Liu, T., and Tao, D. Domain generalization via conditional invariant representations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018c.
- Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018d.
- Li, Y., Yang, Y., Zhou, W., and Hospedales, T. M. Feature-critic networks for heterogeneous domain generalization. *arXiv preprint arXiv:1901.11448*, 2019b.
- Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.
- Magliacane, S., van Ommen, T., Claassen, T., Bongers, S., Versteeg, P., and Mooij, J. M. Domain adaptation by using causal inference to predict invariant conditional distributions. In *Advances in Neural Information Processing Systems*, pp. 10846–10856, 2018.
- Matsuura, T. and Harada, T. Domain generalization using a mixture of multiple latent domains. In *AAAI*, pp. 11749–11756, 2020.
- Miller, J. W., Goodman, R., and Smyth, P. On loss functions which minimize to conditional expected values and posterior probabilities. *IEEE Transactions on Information Theory*, 39(4):1404–1408, 1993.
- Motiiian, S., Piccirilli, M., Adjeroh, D. A., and Doretto, G. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5715–5725, 2017.
- Muandet, K., Balduzzi, D., and Schölkopf, B. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pp. 10–18, 2013.
- Nam, H., Lee, H., Park, J., Yoon, W., and Yoo, D. Reducing domain gap via style-agnostic networks. *arXiv preprint arXiv:1910.11645*, 2019.
- Pearl, J. *Causality*. Cambridge university press, 2009.
- Peters, J., Bühlmann, P., and Meinshausen, N. 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.
- Piratla, V., Netrapalli, P., and Sarawagi, S. Efficient domain generalization via common-specific low-rank decomposition. *Proceedings of the International Conference of Machine Learning (ICML) 2020*, 2020.
- Rahman, M. M., Fookes, C., Baktashmotlagh, M., and Sridharan, S. Correlation-aware adversarial domain adaptation and generalization. *Pattern Recognition*, 100:107124, 2020.
- Rojas-Carulla, M., Schölkopf, B., Turner, R., and Peters, J. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.
- Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

- Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. *arXiv preprint arXiv:2006.07710*, 2020.
- Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., and Sarawagi, S. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.
- Sun, B. and Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pp. 443–450. Springer, 2016.
- Volpi, R., Namkoong, H., Sener, O., Duchi, J. C., Murino, V., and Savarese, S. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, pp. 5334–5344, 2018.
- Wang, H., He, Z., Lipton, Z. C., and Xing, E. P. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.
- Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. 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*, pp. 2097–2106, 2017.
- Wang, Y., Li, H., and Kot, A. C. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3622–3626. IEEE, 2020.
- Xu, M., Zhang, J., Ni, B., Li, T., Wang, C., Tian, Q., and Zhang, W. Adversarial domain adaptation with domain mixup. *arXiv preprint arXiv:1912.01805*, 2019.
- Yan, S., Song, H., Li, N., Zou, L., and Ren, L. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020.
- Zhao, H., Combes, R. T. d., Zhang, K., and Gordon, G. J. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*, 2019.
- Zhao, S., Gong, M., Liu, T., Fu, H., and Tao, D. Domain generalization via entropy regularization. *Advances in Neural Information Processing Systems*, 33, 2020.
- Zhou, K., Yang, Y., Hospedales, T. M., and Xiang, T. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, pp. 13025–13032, 2020.