Cross-Domain Feature Augmentation for Domain Generalization

Yingnan Liu\textsuperscript{1,2}, Yingtian Zou\textsuperscript{1,2}, Rui Qiao\textsuperscript{1}, Fusheng Liu\textsuperscript{2}, Mong Li Lee\textsuperscript{1,2} and Wynne Hsu\textsuperscript{1,2}

\textsuperscript{1}School of Computing, National University of Singapore
\textsuperscript{2}Institute of Data Science, National University of Singapore

\{liu.yingnan, yingtian, rui.qiao, fusheng\}@u.nus.edu, \{leeml, whsu\}@comp.nus.edu.sg

Abstract

Domain generalization aims to develop models that are robust to distribution shifts. Existing methods focus on learning invariance across domains to enhance model robustness, and data augmentation has been widely used to learn invariant predictors, with most methods performing augmentation in the input space. However, augmentation in the input space has limited diversity whereas in the feature space is more versatile and has shown promising results. Nonetheless, feature semantics is seldom considered and existing feature augmentation methods suffer from a limited variety of augmented features. We decompose features into class-generic, class-specific, domain-generic, and domain-specific components. We propose a cross-domain feature augmentation method named XDomainMix that enables us to increase sample diversity while emphasizing the learning of invariant representations to achieve domain generalization. Experiments on widely used benchmark datasets demonstrate that our proposed method is able to achieve state-of-the-art performance. Quantitative analysis indicates that our feature augmentation approach facilitates the learning of effective models that are invariant across different domains.

1 Introduction

Deep learning methods typically assume that training and testing data are independent and identically distributed. However, this assumption is often violated in the real world, leading to a decrease in model performance when faced with a different distribution [Torralba and Efros, 2011]. The field of domain generalization aims to mitigate this issue by learning a model from one or more distinct yet related training domains, with the goal of generalizing effectively to domains that have not been previously encountered. Studies suggest that the poor generalization on unseen distributions can be attributed to the failure of learning the invariance across different domains during the training phase [Muandet \textit{et al.}, 2013; Li \textit{et al.}, 2018]. To tackle this, research has focused on representation learning and data augmentation as key to learning invariance.

Invariant representation learning aims to align representation across domains [Shi \textit{et al.}, 2022], and learn invariant causal predictors [Arjovsky \textit{et al.}, 2019]. They usually impose regularization, which may result in a hard optimization problem [Yao \textit{et al.}, 2022a]. In contrast, data augmentation techniques propose to generate additional samples for the learning of invariance, and avoid the complexities in the regularization approach [Mancini \textit{et al.}, 2020]. Data augmentation can be generally classified into two types: input space and feature space augmentation. The former often encounters limitations due to a lack of diversity in the augmented data [Li \textit{et al.}, 2021b], while the latter offers more versatility and has yielded promising outcomes [Zhou \textit{et al.}, 2021].

Despite the versatility of feature space augmentation, existing methods such as MixStyle [Zhou \textit{et al.}, 2021] and DSU [Li \textit{et al.}, 2022] do not consider feature semantics during the
augmentation process. Instead, they alter feature statistics which often leads to a limited range of diversity. This lack of diversity in the generated features motivates us to decompose the features according to feature semantics. We build on prior research which suggests that the features learned for each class can be viewed as a combination of class-specific and class-generic components [Chu et al., 2020]. The class-specific component carries information unique to a class, while the class-generic component carries information that is shared across classes. We observe that, even within the same class, features of samples from different domains can be distinguished, indicating that these features may contain domain-specific information. As such, we broaden our understanding of features to include domain-specific and domain-generic components.

We introduce a method called XDomainMix that changes domain-specific components of a feature while preserving class-specific components. With this, the model is able to learn features that are not tied to specific domains, allowing it to make predictions based on features that are invariant across domains. Figure 1 shows samples of original images and reconstructed images based on existing feature augmentation technique (DSU) and the proposed XDomainMix. We visualize the augmented features using a pre-trained autoencoder [Huang and Belongie, 2017]. From the reconstructed images, we see that DSU’s augmented features remain largely the same as that of the original image feature. On the other hand, the images reconstructed from the features obtained using XDomainMix have richer variety while preserving the salient features of the class.

Results of experiments on benchmark datasets show the superiority of XDomainMix for domain generalization. We quantitatively measure the invariance of learned representation and prediction to show that the models trained with XDomainMix’s features are more invariant across domains compared to state-of-the-art feature augmentation methods. Our measurement of the divergence between original features and augmented features shows that XDomainMix results in more diverse augmentation.

2 Related Work

To learn invariance, existing domain generalization approaches can be categorized into representation learning methods and data augmentation methods. Works on learning invariant representation employ regularizers to align representations or gradients [Sun and Saenko, 2016; Li et al., 2020; Kim et al., 2021; Mahajan et al., 2021; Shi et al., 2022; Rame et al., 2022; Yao et al., 2022b] across different domains, enforce the optimal classifier on top of the representation space to be the same across all domains [Arjovsky et al., 2019; Ahuja et al., 2021], or uses distributionally robust optimization [Sagawa et al., 2020]. However, the use of regularization terms during learning of invariant representation could make the learning process more complex and potentially limit the model’s expressive power.

Another approach is to employ data augmentation to learn invariance. Existing work that operates in the input space includes network-learned transformation [Zhou et al., 2020; Li et al., 2021a], adversarial data augmentation [Volpi et al., 2018; Shankar et al., 2018], mixup [Mancini et al., 2020; Yao et al., 2022a], and Fourier-based transformation [Xu et al., 2021]. Each of these techniques manipulates the input data in different ways to create variations that help the model learn invariant features. However, the range of transformations that can be applied in the input space is often limited.

On the other hand, feature augmentation can offer more flexibility and potential for learning more effective invariant representations. Prior work has generated diverse distributions in the feature space by changing feature statistics [Zhou et al., 2021; Jeon et al., 2021; Wang et al., 2022; Li et al., 2022; Fan et al., 2023], adding noise [Li et al., 2021b], or mixing up features from different domains [Mancini et al., 2020; Qiao and Peng, 2021]. For example, MixStyle [Zhou et al., 2021] synthesizes new domains by mixing the feature statistics of two features. DSU [Li et al., 2022] extends the idea by modeling feature statistics as a probability distribution and using new feature statistics drawn from the distribution to augment features. In addition to generating diverse distributions, RSC [Huang et al., 2020] adopts a different approach by discarding the most activated features instead of generating diverse data. This encourages the model to use less-activated features that might be associated with labels relevant to data outside the domain.

In contrast to existing methods, our work carefully considers features semantics by leveraging class-label information and domain information to augment features. This increases intra-class variability and helps the model to learn a broader understanding of each class, thus improving its ability to handle new, unseen data.

3 Proposed Method

We consider the problem where we have a set of source domains $D_S = \{S_1, \ldots, S_N\}$, $N > 1$ and a set of unseen domains $D_U$. Each source domain $S_i = \{(x^{(i)}_j, y^{(i)}_j)\}_{j=1}^n$ has a joint distribution on the input $x$ and the label $y$. Domains in $D_U$ have distinct joint distributions from those of the domains in $D_S$. We assume that all domains in $D_S$ and $D_U$ share the same label space but the class distribution across domains need not be the same. The goal is to learn a mapping $g : x \rightarrow y$ using the source domains in $D_S$ such that the error is minimized when $g$ is applied to samples in $D_U$.

In deep learning, $g$ is typically realized as a composition of two functions: a feature extractor $f : x \rightarrow Z$ that maps input $x$ to $Z$ in the latent feature space, followed by a classifier $c : Z \rightarrow y$ that maps $Z$ to the output label $y$. Ideally, $f$ should extract features that are domain invariant yet retain class-specific information. The features of a given input can be decomposed into two components: class-specific and class-generic. The class-specific component consists of feature semantics that are strongly correlated with class labels, making them more important in discriminating a target class from other classes.

Furthermore, features can also be decomposed into domain-specific and domain-generic components. This is because samples from different domains, even if they belong to the same class, possess unique feature characteristics to their
Let \( z \) be the derivative of class classifier \( v \). We determine whether \( z \) is class-specific or class-generic via the class importance score, which is computed as the product of feature value and the derivative of class classifier \( c \)'s predicted logit \( v_c \) of \( x \)'s ground truth class \([Selvaraju et al., 2017; Chu et al., 2020]\) as they show how much \( z \) contributes to \( v_c \):

\[
\text{class\_score}(z^k) = \frac{\partial v_c}{\partial z^k} z^k
\]

To determine if \( z^k \) is domain-specific, we employ a domain classifier \( d \) that has an identical architecture as the class classifier \( c \). \( d \) is trained to predict domain labels of features extracted by the feature extractor. Similar to Equation 1, domain importance score is computed using the derivative of \( d \)'s predicted logit \( v_d \) of \( x \)'s ground truth domain:

\[
\text{domain\_score}(z^k) = \frac{\partial v_d}{\partial z^k} z^k
\]

Let \( \tau_c \) and \( \tau_d \) be predefined thresholds for filtering feature dimensions that are class-specific and domain-specific respectively. We obtain a class-specific mask \( M_c \in \mathbb{R}^K \) and a domain-specific mask \( M_d \in \mathbb{R}^K \) on \( Z \) for \( \{z^k\}_{k=1}^K \) where their respective \( k^{th} \) entries are given as follows:

\[
M_c[k] = \begin{cases} 
1 & \text{if } \text{class\_score}(z^k) > \tau_c \\
0 & \text{otherwise}
\end{cases}
\]

\[
M_d[k] = \begin{cases} 
1 & \text{if } \text{domain\_score}(z^k) > \tau_d \\
0 & \text{otherwise}
\end{cases}
\]

Complementary class-generic mask and domain-generic mask are obtained by \( 1 - M_c \) and \( 1 - M_d \) where \( 1 \) is the tensor of values 1 and of the same size as \( Z \). Class-specific domain-specific \( (Z_{c,d}) \), class-specific domain-generic \( (Z_{c,-d}) \), class-generic domain-specific \( (Z_{-c,d}) \), and class-generic domain-generic \( (Z_{-c,-d}) \) feature components are obtained by

\[
Z_{c,d} = M_c \odot M_d \odot Z \\
Z_{c,-d} = M_c \odot (1 - M_d) \odot Z \\
Z_{-c,d} = (1 - M_c) \odot M_d \odot Z \\
Z_{-c,-d} = (1 - M_c) \odot (1 - M_d) \odot Z
\]

where \( \odot \) is element-wise multiplication. Note that \( Z_{c,d} + Z_{c,-d} + Z_{-c,d} + Z_{-c,-d} = Z \).

3.2 Cross Domain Feature Augmentation

To achieve domain invariance and reduce reliance on domain-specific information presented in training domains during
prediction, we manipulate domain-specific feature components to enhance diversity from a domain perspective. Further, the augmentation should increase feature diversity while preserving class semantics using existing data. This is achieved by mixing the class-specific domain-specific feature component of a sample with the class-specific domain-specific feature component of a same-class sample from other domains. For class-generic domain-specific feature component, it is mixed with the class-generic domain-specific feature component of a different-class sample from other domains, introducing further diversity.

Specifically, for the feature $Z$ extracted from input $x$, we randomly sample two inputs $x_i$ and $x_j$ whose domains are different from $x$. Further, $x_i$ has the same class label as $x$ while $x_j$ is from a different class. Let $Z_i$ be the feature extracted from input $x_i$ and $Z_j$ be the feature extracted from $x_j$. Then we have

$$
\begin{align*}
\tilde{Z}_{c,d} &= \lambda_1 Z_{c,d} + (1 - \lambda_1) Z_{ic,d}, \\
\tilde{Z}_{\neg c,d} &= \lambda_2 Z_{\neg c,d} + (1 - \lambda_2) Z_{j\neg c,d}
\end{align*}
$$

(5)

where $\lambda_1$ and $\lambda_2$ are the mixup ratios independently sampled from a uniform distribution $U(0,1)$.

A new feature $\tilde{Z}$ with the same class label as $Z$ is generated by replacing the domain-specific component in $Z$ as follows:

$$\tilde{Z} = \tilde{Z}_{c,d} + \tilde{Z}_{\neg c,d} + Z_{\neg c,d} + Z_{\neg c,d}
$$

(6)

To further encourage the model to focus on domain-invariant features and exploit the less activated feature during class prediction, we discard the class-specific domain-specific feature component with some probability $p_{\text{discard}}$ as follows:

$$\tilde{Z} = \begin{cases} 
\tilde{Z}_{c,d} + \tilde{Z}_{\neg c,d} + Z_{\neg c,d} + Z_{\neg c,d} & \text{if } p \leq p_{\text{discard}} \\
\tilde{Z}_{c,d} + \tilde{Z}_{\neg c,d} + Z_{\neg c,d} + Z_{\neg c,d} & \text{otherwise}
\end{cases}
$$

(7)

where $p$ is randomly sampled from a uniform distribution $U(0,1)$.

### 3.3 Training Procedure

Prior research has shown that empirical risk minimization (ERM) [Vapnik, 1999] is a competitive baseline [Gulrajani et al., 2021; Wiles et al., 2022]. The objective function of ERM is given by

$$\mathcal{L}_{\text{erm}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(y_j^{(i)}, \hat{y}_j^{(i)})
$$

(8)

where $\ell$ is the loss function to measure the error between the predicted class $\hat{y}_j^{(i)}$ and the ground truth $y_j^{(i)}$. $N$ is the number of training examples and $n_i$ is the number of training samples in domain $i$.

We train the model in two phases. During the warm-up phase, the feature extractor $f$ and class classifier $c$ are trained on the original dataset for class label prediction following $\mathcal{L}_{\text{erm}}$. The domain classifier $d$ is trained using $Z$, the features extracted by $f$, to predict domain labels. The objective function of $d$ is given by

$$\mathcal{L}_d = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(d(Z_j^{(i)}), i)
$$

(9)

where $\ell$ is the loss function to measure the error between the predicted domain $d(Z_j^{(i)})$ and the ground truth $i$.

When the warm-up phase is completed, we use Equation 4 to decompose the features obtained from $f$. Augmented features are then derived using Equation 7. Both feature extractor $f$ and class classifier $c$ are trained using the original and augmented features with the following objective function:

$$\mathcal{L}_{\text{aug}} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2n_i} \sum_{j=1}^{n_i} \left[ \ell(c(Z_j^{(i)}), y_j^{(i)}) + \ell(c(\tilde{Z}_j^{(i)}), y_j^{(i)}) \right]
$$

(10)

where $\tilde{Z}_j^{(i)}$ is the augmented feature derived from $Z_j^{(i)}$, $c(Z_j^{(i)})$ is the predicted class given $Z_j^{(i)}$, and $c(\tilde{Z}_j^{(i)})$ is the predicted class given $\tilde{Z}_j^{(i)}$. $y_j^{(i)}$ is the ground truth class.

We also train the domain classifier $d$ using $\mathcal{L}_d$. Note that the domain classifier $d$ is not trained using this set of augmented features, as the augmented features do not have assigned domain labels since they need not follow the distribution of existing domains.

### 4 Performance Study

We implement our proposed solution using PyTorch 1.12.0 and perform a series of experiments on NVIDIA Tesla V100 GPU to evaluate the effectiveness of the proposed XDomain-Mix. The following benchmark datasets are used:

- Camelyon17 [Bandi et al., 2018] from Wilds [Koh et al., 2021]. This dataset contains 455,954 tumor and normal tissue slide images from 5 hospitals (domains). Distribution shift arises from variations in patient population, slide staining, and image acquisition.
- FMoW [Christie et al., 2018] from Wilds. This dataset contains 141,696 satellite images from 62 land use categories across 16 years from 5 regions (domains).
- PACS [Li et al., 2017]. This dataset contains 9,991 images of 7 objects in 4 visual styles (domains): art painting, cartoon, photo, and sketch.
- Terralncognita [Beery et al., 2018]. The dataset contains 24,788 images from 10 categories of wild animals taken from 4 different locations (domains).
- DomainNet [Peng et al., 2019]. This dataset contains 586,575 images from 365 classes in 6 visual styles (domains): clipart, infographic, painting, quickdraw, real, and sketch.

The class importance thresholds $\tau_c$ and domain importance thresholds $\tau_d$ in Equation 3 are set as follows: $\tau_c$ is set to be the 50%-quantile of the class importance scores of $\{z_k\}$ in a feature $Z$ so that 50% dimensions are considered class-specific, while the remaining 50% are class-generic. $\tau_d$ controls the strength of the augmentation as it determines the identification of domain-specific feature components. We employ a cyclic changing scheme for $\tau_d$ to let the model learn gradually from weak augmentation to strong augmentation and give the domain classifier more time to adapt to a more domain-invariant feature extractor. The value is initially set
to be $90\%$-quantile of domain importance scores of $\{z^k\}$ in a feature $Z$. As the training proceeds, $\tau_d$ is decreased by 10% quantile for every $n$ step until it reaches the $50\%$-quantile, where it also remains for $n$ steps. The same cycle is repeated where $\tau_d$ is set to be $90\%$-quantile again. Input $x_i$, $x_j$ used in augmentation (Equation 5) are samples from the same training batch. $p_{\text{discard}}$ is set to 0.2.

For Camelyon17 and FMoW datasets, we follow the setup in LISA [Yao et al., 2022a]. Non-pretrained DenseNet-121 is used for Camelyon17 and pretrained DenseNet-121 is used for FMoW. We use the same partitioning in Wilds [Koh et al., 2021] to obtain the training, validation, and test domains. The batch size is set to 32, and the model is trained for 5000 steps. The learning rate and weight decay are set to 1e-4 and 0. The warm-up phase is set to 4000 steps. We tune the step $n$ in $\{100, 500\}$ for changing $\tau_d$. The best model is selected based on its performance in the validation domain.

For PACS, TerraIncognita and DomainNet datasets, we follow the setup in DomainBed [Gulrajani and Lopez-Paz, 2021, Cha et al., 2021], and use a pre-trained ResNet-50. Each domain in the dataset is used as a test domain in turn, with the remaining domains serving as training domains. The batch size is set to 32 (24 for DomainNet), and the model is trained for 5000 steps (15000 steps for DomainNet). We tune the learning rate in $\{2e^{-5}, 3e^{-5}, 4e^{-5}, 5e^{-5}, 6e^{-5}\}$ and weight decay in $\{1e^{-6}, 1e^{-2}\}$ using the DomainBed framework. The warm-up phase is set to 3000 steps and $n$ is set to 100 steps. The best model is selected based on its performance on the validations split of the training domains.

### 4.1 Domain Generalization Performance

We compare our proposed XDomainMix with ERM [Vapnik, 1999] and the following state-of-the-art methods:

- **GroupDRO** [Sagawa et al., 2020] minimizes worst-case loss for distributionally robust optimization.
- **RSC** [Huang et al., 2020] discards features that have higher activation to activate the remaining features appear to be applicable to out-of-domain data.
- **MixStyle** [Zhou et al., 2021] synthesizes new domains by mixing feature statistics of two features.
- **DSU** [Li et al., 2022] synthesizes new domains by re-normalizing feature statistics of features with the ones drawn from a probability distribution.
- **LISA** [Yao et al., 2022a] selectively mixes up samples to learn an invariant predictor.
- **Fish** [Shi et al., 2022] aligns gradients across domains by maximizing the gradient inner product.

Classification accuracy, which is the ratio of the number of correct predictions to the total number of samples is reported. Following the instruction of datasets, average accuracy on the test domain over 10 runs is reported for the Camelyon17 dataset; worst-group accuracy on the test domain over 3 runs is reported for the FMoW dataset.

Table 1 shows the results. Our method consistently achieves the highest average accuracy across all the datasets, outperforming SOTA methods. This result suggests that XDomainMix is able to train models with good domain generalization ability.

### 4.2 Model Invariance

One advantage of XDomainMix is that is able to learn invariance across training domains. We quantify the invariance in terms of representation invariance and predictions invariance. Representation invariance refers to the disparity between representations of the same class across different domains. The distance between second-order statistics (covariances) [Sun and Saenko, 2016] can be used to measure representation invariance. Prediction invariance considers the variation in predictions across different domains. We employ risk variance [Yao et al., 2022a] which measures how similar the model performs across domains.

Let $\{Z_j^i | y_j^i = y_c\}$ be the set of representations of class label $y_c$ from domain $i$. We use $C_{y_c}^{(i)}$ to denote the covariance matrix of the representations. Given the set of class labels $Y$ and the set of training domains $D_S$, the measurement result is given by $\frac{1}{|Y||D_S|} \sum_{y_c \in Y} \sum_{i,i' \in D_S} ||C_{y_c}^{(i)} - C_{y_c}^{(i')}||_F^2$. $|.|_F$ denotes the squared matrix Frobenius norm. A smaller distance suggests that same-class representations across domains are more similar.

Let $R^i$ be the loss in predicting the class labels of inputs from domain $i$. The risk variance is given by the variance among training domains, $\text{Var}\{R^1, R^2, ..., R^{|D_S|}\}$. Lower risk variance suggests a more consistent model performance across domains.
(a) Representation invariance measured by distance of covariance matrix of same class representations across domains.

(b) Prediction invariance measured by variance of risk across domains. The results are reported in the unit of 1e-3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Camelyon17</th>
<th>FMoW</th>
<th>PACS</th>
<th>TerraIncognita</th>
<th>DomainNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERM</td>
<td>0.47±0.18</td>
<td>0.35±0.09</td>
<td>0.69±0.12</td>
<td>0.50±0.09</td>
<td>3.60±0.38</td>
</tr>
<tr>
<td>GroupDRO</td>
<td>0.21±0.02</td>
<td>0.40±0.04</td>
<td>0.77±0.16</td>
<td>0.43±0.05</td>
<td>2.13±0.09</td>
</tr>
<tr>
<td>RSC</td>
<td>0.32±0.17</td>
<td>0.55±0.14</td>
<td>42.3±12.8</td>
<td>29.9±3.02</td>
<td>17.3±1.43</td>
</tr>
<tr>
<td>MixStyle</td>
<td>0.28±0.26</td>
<td>0.36±0.05</td>
<td>0.69±0.03</td>
<td>0.38±0.09</td>
<td>3.39±0.17</td>
</tr>
<tr>
<td>DSU</td>
<td>0.07±0.02</td>
<td>0.32±0.02</td>
<td>0.33±0.04</td>
<td>9.18±1.84</td>
<td>4.61±0.24</td>
</tr>
<tr>
<td>LISA</td>
<td>0.19±0.05</td>
<td>0.29±0.05</td>
<td>0.04±0.00</td>
<td>0.13±0.01</td>
<td>0.72±0.06</td>
</tr>
<tr>
<td>Fish</td>
<td>3.95±3.28</td>
<td>0.47±0.02</td>
<td>0.64±0.34</td>
<td>0.34±0.04</td>
<td>2.60±0.26</td>
</tr>
<tr>
<td>XDomainMix</td>
<td>0.19±0.07</td>
<td>0.28±0.01</td>
<td>0.02±0.00</td>
<td>0.04±0.00</td>
<td>0.11±0.02</td>
</tr>
</tbody>
</table>

Table 2: Results of model invariance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Camelyon17</th>
<th>FMoW</th>
<th>PACS</th>
<th>TerraIncognita</th>
<th>DomainNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>MixStyle</td>
<td>6.65±0.18</td>
<td>8.58±0.50</td>
<td>3.1±0.34</td>
<td>3.20±0.26</td>
<td>2.81±0.11</td>
</tr>
<tr>
<td>DSU</td>
<td>26.77±2.53</td>
<td>20.93±1.87</td>
<td>6.97±0.03</td>
<td>10.85±0.32</td>
<td>5.75±0.01</td>
</tr>
<tr>
<td>XDomainMix</td>
<td>38.82±0.28</td>
<td>34.06±0.11</td>
<td>14.82±0.11</td>
<td>14.36±0.12</td>
<td>10.27±0.02</td>
</tr>
</tbody>
</table>

Table 3: Divergence of the augmented feature and original feature measured by MMD in the unit of 1e-2.

Table 2 shows the results. We see that our method has the smallest covariance distance in the FMoW, PACS, TerraIncognita, and DomainNet dataset, and the second-smallest in the Camelyon17 dataset. The results indicate that the representations of the same class learned by our method have the least divergence across domains. Additionally, XDomainMix has the lowest risk variance, suggesting that it is able to maintain consistent performance in predictions across domains. Overall, the results demonstrate that our approach is able to learn invariance at both the representation level and the prediction level.

4.3 Diversity of Augmented Features

To show that XDomainMix can generate more diverse features, we measure the distance between the original and augmented features using maximum mean discrepancy (MMD). A higher MMD suggests that the distance between the original and augmented features is further. The same set of original features is used to ensure fairness and comparability of the measurement result. Average and standard deviation over three runs are reported. Table 3 shows the results. Features augmented by XDomainMix consistently have the highest MMD compared to MixStyle and DSU which are two SOTA feature augmentation methods. This suggests that the features augmented by XDomainMix exhibit the most deviation from the original features, leading to a more varied augmentation. Visualization of sample images reconstructed from augmented features are given in Supplementary.

4.4 Experiments on the Identified Features

In this set of experiments, we demonstrate that XDomainMix is able to identify features that are important for class and domain prediction. We evaluate the model performance for class or domain prediction after eliminating those features with the highest importance score computed in Equations 1 and 2. A decrease in accuracy suggests that the features that have been removed are important for the predictions.

For comparison, we implement two alternative selection strategies: a random method that arbitrarily selects features to remove, and a gradient norm approach, where features are chosen for removal based on the magnitude of the gradient in the importance score computation. Samples in the validation set of PACS dataset are used in this experiment.

Figure 3 shows the results. Our method shows the largest drop in both class prediction and domain prediction accuracies compared to random removal and gradient norm methods. This indicates that XDomainMix is able to identify features that are specific to the domain and class effectively.

To visualize the extracted features, we map the high dimensional feature vectors obtained by $f$ to a lower dimensional space. This transformation is carried out using two linear layers, as described in [Zhang et al., 2021]. Figure 4(a) provides a visualization for the model that is trained on the PACS dataset, with Art as the unseen domain. We see that
Figure 3: Prediction accuracy after removing x% of features with the highest importance scores.

Figure 4: Visualization of features from different classes/domains, indicated by the different colors.

the features identified as class-specific are well separated by class. This is in contrast to the features that are generic across classes, which are not as clearly delineated.

Similarly, we also visualize the extracted domain-specific and domain-invariant features. As shown in Figure 4(b), the domain-specific features are noticeably better separated compared to the domain-invariant features.

4.5 Ablation Study

To understand the contribution of each component in the augmentation, we perform ablation studies on Camelyon17 and FMoW datasets. Table 4 shows the result. Compared to the baseline, mixing class-specific domain-specific feature components \( Z_{c,d} \) only, or mixing class-generic domain-specific feature components \( Z_{¬c,d} \) only in the augmentation can improve the performance. This suggests that by manipulating domain-specific feature components, models that are better at domain generalization can be learned. Mixing \( Z_{¬c,d} \) leads to greater improvement, indicating that enriching diversity by content from other classes is more helpful than simply intra-class augmentation.

Augmenting both \( Z_{c,d} \) and \( Z_{¬c,d} \) does not consistently lead to performance improvement, possibly due to dataset-specific characteristics. Probabilistically discarding \( Z_{c,d} \) seems to encourage the model to use less domain-specific information and exploit less activated features in prediction, which improves the domain generalization performance.

<table>
<thead>
<tr>
<th>mix</th>
<th>mix</th>
<th>discard</th>
<th>Camelyon17</th>
<th>FMoW</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>70.3±6.4</td>
<td>32.3±1.3</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>78.3±5.5</td>
<td>32.9±2.2</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>79.1±6.0</td>
<td>33.6±1.1</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>79.6±7.0</td>
<td>31.9±0.4</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>80.9±3.2</td>
<td>35.9±0.8</td>
</tr>
</tbody>
</table>

Table 4: Ablation study.

5 Conclusion and Future Work

In this work, we have developed a feature augmentation method to address the domain generalization problem. Our approach aims to enhance data diversity within the feature space for learning models that are invariant across domains by mixing domain-specific components of features from different domains while retaining class-related information. We have also probabilistically discarded domain-specific features to discourage the model from using such features for their predictions, thereby achieving good domain generalization performance. Our experiments on multiple datasets demonstrate the effectiveness of the proposed method.

While our method presents a promising approach to solving the domain generalization problem, there are several limitations. Our method needs more than one training domain to perform cross-domain feature augmentation. Our method assumes that the datasets across different domains share the same label space and similar class distributions, and its performance may be affected if this is not the case. Our method is mainly empirically validated, and a theoretical analysis or guarantee of its performance is still lacking. Further research is needed to provide a deeper theoretical understanding of the proposed method and its performance bounds.

Acknowledgements

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-GC-2019-001-2B). We thank Dr Wenjie Feng for the helpful discussions.
References


