7-2021

Domain Generalization and Adaptation with Generative Modeling and Representation Learning

Jaideep Vitthal Murkute
jvm6526@rit.edu

Domain Generalization and Adaptation with Generative Modeling and Representation Learning

by

Jaideep Vitthal Murkute

A Thesis Submitted in Partial Fulfillment of the Requirements for the *Master of Science* *in* *Computer Science*

Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences,
Rochester Institute of Technology.

July 2021

Domain Generalization and Adaptation with Generative Modeling and Representation Learning

by
Jaideep Vitthal Murkute

Approved By:

Dr. Linwei Wang, Advisor (Committee Chair)

Dr. Ifeoma Nwogu, Reader (Committee Member)

Dr. Yu Kong, Observer (Committee Member)

Contents

1 Abstract

Despite the success of deep learning methods on object recognition tasks, one of the challenges deep learning systems face in the real world is the ability to perform well on the visually different data samples i.e. under a distribution shift caused by the samples of the same object category but from the significantly different visual domain. Many approaches have been proposed in both of these settings, however, not many other works focus on the generative modeling in this context, neither focus studying the structure of hidden representations learned by the deep learning models. We hypothesize that learning the generative factors and studying the structures of features learned by the models can allow us to develop new methodology for domain generalization and domain adaption setting. In this work, we propose a new methodology by designing a Variational Autoencoder (VAE) based model with structured three-part latent code representing specific aspects of data. We also make use of adversarial approaches to make model robust towards changes in visual domains, to improve the domain generalization performance. For the domain adaptation, we make use of semi-supervised learning as a primary tool to adapt model parameters to learn the new data distribution of the target domain. We propose a novel variation of data augmentation used in semi-supervised methods based on the latent code sampling. We also propose a new adversarial constraint for domain adaptation which does not require explicit information about the 'domain' of the new data sample. From empirical evaluation, our method performs on par with the other state of the art methods in domain generalization setting, while improving state of the art for multiple datasets in the domain adaptation setting.

2 Motivation and Objectives

2.1 Motivation

Robustness of machine learning systems to perform well the inputs on the previously unseen characteristics is an important factor that decides effectiveness of such systems in the real world. This problem is not limited to any data modality liker vision or natural language processing. Neither is limited to any application domains. one of the example of application area is the healthcare sector. Machine learning based medical diagnosis assistants need to be able to generalize well on the set of patients it has not seen before. New set of patients may exhibit anatomical or pathological characteristics ML model has not seen before. Another change can be in the sensor quality, artifacts involved because of the data collection process. Often-times, datasets used contain the information about these changes or 'groups'. This auxiliary data or meta-data often gets ignored. We hypothesize there can be a great value in utilizing this additional information present in the datasets. This additional data can be as simple as patient ID or more detailed data about the patient's condition. Changes like these can be considered under the research area of domain generalization and adaptation, by considering the 'group' information as the domain related information. For this study, we will limit to the computer vision application and the task of object classification; however, we proposed methods and learning can be extended to other application areas.

2.2 Objectives

The core objective of this work is to build a domain generalization and adaptation framework by leveraging generative modeling, and in particular VAEs, which is the area less explored. Not many in depth studies exist on this topic that visually analyze the representations learned by the networks in detail. We also plan to perform in-depth ablation study of the effectiveness of components of the framework developed. For the domain adaptation method, we aim to build a method that can adapt to any number of (≥ 1) of source and target domains. Although in this work, we'll be considering the computer vision application and the image classification task, we aim to develop a generic framework, which can be adapted to other applications and tasks.

3 Background

In this section, we will briefly introduce the terminologies and concepts that are part of the proposed methodology.

3.1 Domain Generalization

Generalization in machine learning refers to the models ability to perform well or generalize to the previously unseen, new data samples. While in typical setting of machine learning models, no special assumption is made about the data samples on which generalization ability of the model is tested. Sometimes, it's ensured that data distribution of the data samples on which generalization power is being measured is represented sufficiently in the seen or the training samples.

The goal of the domain generalization methods is to perform well across the data domains and evaluated by the performance on the data samples from an unseen domain(s). The 'domain' refers to the particular setting in which images or data samples are recorded, and domain assumed to have significant effect on the characteristics of the input. Data within a domain assumed to reflect commonalities belonging to that domain while data samples from different domains differ significantly for the aspects influenced by the domain. A simple example of such domain can be images of cats taken in an common indoor setting vs images taken in the forest.

3.2 Domain Adaptation

Domain adaptation(DA) approaches consider a scenario where models can learn from or adapt to the target domain data, but without the ground truth labels. Domain adaptation is considered as a sub-category of more popular transfer learning. Domain adaptation assumes that the source and the target domains share a same or similar feature space but they differ in the data distribution. Transfer learning includes all scenarios where the target dataset can have significantly different feature space or even task description. The shift in source and target feature distributions, mentioned above, is popularly termed as 'domain shift' in the DA setting. One of the essential desired characteristic of the domain adaptation methods is the capability to update the learned distribution to accommodate and perform well on this new distribution.

From learning methodology perspective, domain adaptation methods can be broadly classified as unsupervised, semi-supervised and weakly-supervised methods, while unsupervised setting being the most common setting studied.

3.3 Representation learning

Representation learning refers to the task of learning representations of the data that makes it easier to extract useful information about the relationships between categories, and which in turn can make the easier to analyze and improve downstream tasks like classification [1]. Our proposed approach is centered around the representation learning and disentangled representation learning. We explore representation learning based approach that makes use of generative modeling to learn the representations. In the context of probabilistic generative modeling, which is central to our approach, representation learning is the ability to capture the posterior distributions of the generative factors for the observed data.

3.4 Generative Modeling

In generative modeling, data under consideration is assumed to have generated from the underlying distribution defined by the generative factors which we're interested in learning. Popular types of generative models include Gaussian mixture models, Bayesian Network (ex. Naive Bayes), Hidden Markov Models, Boltzmann Machines (ex. Restrictive Bolzmann Machines, Belief Networks), Autoencoders etc. With the success of deep learning, generative methods with deep learning have been proposed, termed as deep generative models (DGMs). Also, many of the classical generative modeling methods have been modified to incorporate deep learning as the learning process. For example, deep variational autoencoders, Gaussian Mixture Models with Deep neural networks, along with methods like Deep belief networks, generative adversarial networks (GANs), energy based models etc.. Generative models and methods aim to learn the generative factors of the data and can be used to generate data samples from the learned distributions by manipulating them. Discriminative models solely focus on identifying the differences between the categories of interest. Generative models are sometimes also employed for discriminative task, to discriminate based on the generative features learned for different categories or classes of interest. Discriminative task is either treated as as a downstream task or are trained for both generative and discriminative tasks simultaneously.

Out of the different generative models mentioned above, deep VAEs and GANs have gained popularity in recent years, due to their success of learning complex factors and generating new high resolution, realistic looking images. Particularly, GANs are current state of the art for generating high resolution imagery. However, for the representation learning and disentanglement learning setting, VAEs are a more common choice. With VAEs we can observe the latent features extracted by the model for a given input. We can also visualize the relationships between the features of inputs from different categories, both of which would be difficult to do with the GANs.

3.5 Disentanglement learning

Disentangled representation learning in the generative modeling context is the ability of the machine learning system to learn the independent variables, each of which is defining separate salient factors of variation, guiding the data generation process. In a typical setting of variational autoencoders(VAEs), it has been observed in the that, each of the units in the stochastic bottleneck layer can capture the change in different data generative factors [20], [11]. This behavior arises from the typical design choice that each latent unit in the VAE is assumed to have independent Gaussian distribution. In the context of images, such factors can be color, object size, rotation angle etc..

3.6 Variational Autoencoder

Autoencoders are a popular family of models used for the generative modeling purpose, to learn a rich and meaningful set of features from the dataset. Analogy can be made to the dimensionality reduction approaches in classical machine learning. A common hypothesis is that learning this small set of features from high dimensional data can give us meaningful set of features that are actually useful for the downstream tasks or to analyze the generative factors of the data etc.. In case of autoencoders, this mapping from high dimensional data to low dimensional space is learned by the neural network termed as encoder. A Decoder network is also trained simultaneously to reconstruct the original data sample from this low dimensional representation, to avoid trivial solutions and ensure that meaningful features are being captured. Fig. 1 shows a typical autoencoder architecture. Each or at least some of the units in the 'bottleneck' layer in the autoencoder is assumed to learn some generative factor of the data.

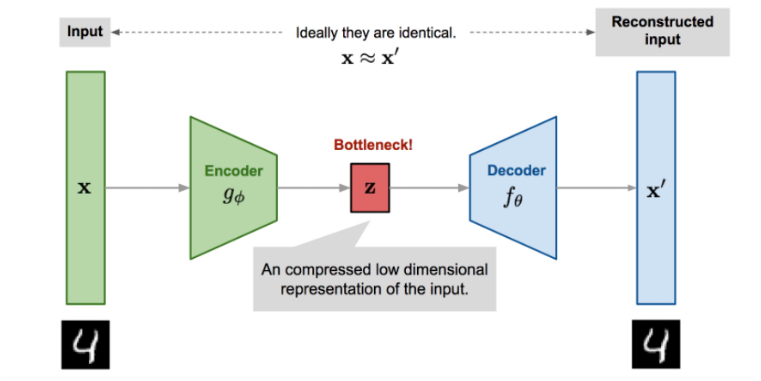
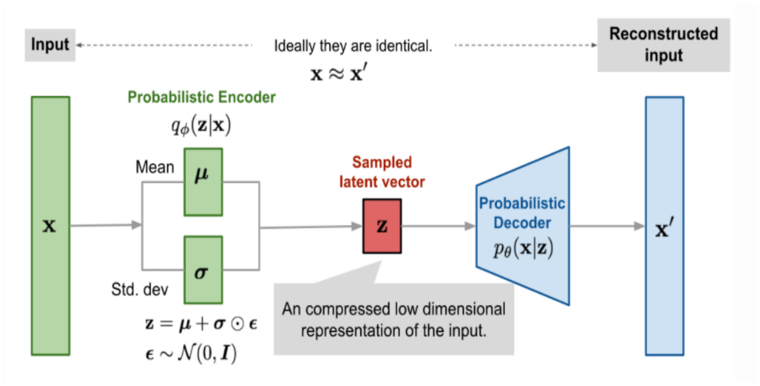

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 1: A typical autoencoder schematic. Image reference: [54]

The variational autoencoders (VAEs) are based on the similar idea as the autoencoders with one distinction. In autoencders, the 'bottleneck' layer is a deterministic layer i.e. each input gets mapped to a exact vector in the latent space. VAEs instead learn a stochastic bottleneck layer i.e. they learn a distribution for each of the latent unit in the bottleneck layer. Assumption is that the generative factors of the data exhibit a distribution, so learning a model that can directly express the distribution can be better suited. Also, VAEs create a much more continuous latent space representations than autoencoders, which can be useful to observe the effect of change in generative factor values. To learn the latent distributions, each unit in the bottleneck is assumed to have a Gaussian prior(Gaussian is a common choice, and is a hyper-parameter) and the posterior distributions are parameterized as Gaussians with a diagonal co-variance matrix. Fig. 2 shows the schematics of the typical autoencoder with a Gaussian prior.



Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
Mean
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
Std. dev
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Figure 2: A typical variational autoencoder schematic. Image reference: [54]

Formally, let's consider that we have some data distribution repренseted by x and consider latent generative feature set to be learned is z. If we want to learn the generative model for x, we can aim to learn the model parameters that maximize the likelihood of the training data p_θ and can be given as:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz \tag{1}$$

We can assume assume $p_\theta(z)$ to be a gaussian prior, for the term $p_\theta(x|z)$ we can employ a decoder neural network to learn the mapping, same as in autoencoders. However, the integral term over z would be intractable since z is a continuous vector. The posterior can be given as:

$$p_\theta(z|x) = \int p_\theta(x|z) p_\theta(z) / p_{\theta(x)} \tag{2}$$

This posterior would also be intractable because of the continuous $p_{\theta(x)}$ term. This problem can be solved by employing an encoder network, $q_\theta(z|x)$, to estimate $p_\theta(z|x)$. With this data likelihood can be written as:

$$log(p_\theta(x^{(i)})) = E_{z \sim q_\theta(z|x^{(i)})}[log p_\theta(x^{(i)})] \tag{3}$$

From the Bayes' rule, we can write Eq. 3 as:

$$log(p_\theta(x^{(i)})) = E_z[log\frac{p_\theta(x^i|z)p_\theta(z)}{p_\theta(z|x^i)}] \tag{4}$$

Multiplying numerator and denominator by constant $q_\theta(z|x^i)$ and then expanding the logarithm, we arrive at:

$$log(p_\theta(x^{(i)})) = \underbrace{E_z[logp_\theta(x^i|z)] - D_{KL}(q_\theta(z|x^i)||p_\theta(z))}_{\text{ELBO}} + D_{KL}(q_\theta(z|x^i)||p_\theta(z|x^i)) \tag{5}$$

; where D_{KL} refers to the KL-divergence. As we saw earlier, $p_\theta(z|x^i)$ term is intractable. But since KL is always $>= 0$; we can treat the first two terms as the lower bound, commonly termed as Evidence Lower Bound (ELBO) and optimize the neural network to maximize just this lower bound. The first term in ELBO maximizes the likelihood of the original input being reconstructed and is enforced with reconstruction loss between reconstruction and the input data, for example, a mean squared error. The second term in ELBO, the KL term, encourages posterior to be close the to the prior and thus acts as a regularizer that can restrict which features can get encoded. We aim to learn a distribution in a latent space and sample from it, but the sampling operation being stochastic, we can't backpropagate through the sampling function. Instead, we can train VAE by using the 're-parameterization trick' [29]. With this, we learn the deterministic mean(μ) and the standard deviation(σ). The noise, $varepsilon$, will be sampled from the fixed standard Gaussian $N(\mu = 0, \sigma = 1)$. Thus the sampling operation becomes $z = \mu + \sigma * \varepsilon$ and the gradients can now flow through to allow training.

Write about the Re-parameterization trick

3.7 Learning better representations with Mixup

In typical supervised learning setting, the goal is to find a mapping function of interest $f \in F$, between data distribution X and the target variables Y which follows a joint distribution $P(X, Y)$.

In deep learning, this is achieved by minimizing a loss or an objective function l, also known as an Empirical Risk Minimization (ERM) [50]. Since the true data is distribution is unknown in many real world scenarios, P is learned to minimize an empirical distribution. Many approaches like Vicinal Risk Minimization (VRM) have been proposed to approximate the true distribution. Mixup [59] is a generic form of vicinal distribution which proposes to create convex combinations of data samples and their labels to train neural networks with. These combinations can be created by a simple linear interpolation method, as shows by Eq. 6 and Eq. 7:

$$x^{'} = \lambda^{'} x_1 + (1 - \lambda^{'}) x_2 \tag{6}$$

$$y^{'} = \lambda^{'} y_1 + (1 - \lambda^{'}) y_2 \tag{7}$$

; where (x_1, y_1) and (x_2, y_2) are the pairs from original data and labels available, while $(x^{'}, y^{'})$ is a new interpolated instance on which we can train the neural network on. The λ values are sampled typically, from a symmetric Beta distribution given by; (α, α). Higher value of α typically acts as a higher regularization as it presents network to classify 'harder' examples. Such mixup can be performed in the input space or any latent feature space of the neural network layers.Mixup training encourages models to learn the clusters of the classes that are well separated from each other and are tight. Mixup training also helps to create a smoother decision boundaries. Both of these qualities are assumed to potentially help with the better generalization. Many variants of the Mixup techniques like input mixup [59], manifold mixup [51], CutMix [58], SnapMix [22] etc. have been proposed. For this work, we employ the generic form as presented in [59] and [51]. The mixup training is assumed to create the representations better suited for generalization purpose. This stems from the observation that mixup training helps to create tight and well separated clusters for the classes.

3.8 Semi-Supervised Learning

Semi-supervised learning (SSL) is an approach to build machine learning models by utilizing a small amount of labeled data and a large amount of unlabeled data to learn the desired task. This problem setting focuses on reducing number of labeled data requirements to achieve high performance. Reducing label complexity can results in cost savings since data labeling can be a very expensive process in many applications. In some of the applications, data labeling may require an expert knowledge where labeling process becomes even more costly and difficult. Semi-supervised approaches assume that even though we do not have labels for the unlabeled data, under certain assumptions, and utilizing the small amount of labeled data, we can learn from the unlabeled data and gain significant improvements in performance. In a typical supervised learning setting, we have a dataset X and the associated target variables Y which follows a joint distribution $P(X,Y)$. And the goal is to learn the function $f \in F$ that maximizes the $E_{x,y \sim p(x,y)}[logp(y|x)]$. Whereas, in semi-supervised learning, we have a labeled dataset $D_L : (x,y) \sim p(x,y)$ and an unlabeled dataset $D_U : x \sim p(x)$. A common assumption is made about the unlabeled dataset that unlabeled data comes from the same distribution as represented by $P(X,Y)$. The goal of the SSL methodologies is to figure out how to utilize the unlabeled data to learn better from or impose constraints on learning from the labeled data. Alternatively, some methods also focus on estimating the label associations or other information about the the unlabeled data based on the labeled data to learn from. Some other methods impose unsupervised constraints by on the labeled and the unlabeled data, based on the cluster assumptions.

4 Related Work

4.1 Generative Modeling with VAEs

Variational autoencoders have been quite successful in learning meaningful set of genereative features of the data, which provide an ease to analyze the learned features for any given input sample, unlike GANs. Since the 'vanilla' version of variational autoencoders that can be trained with backpropagation algorithm as presented in [29], many other variations have been proposed which either claim to achieve higher disentanglement performance or better reconstruction quality, some variants aim to learn the richer latent space by imposing priors other than the Gaussian, while some other alleviate the problem of 'posterior collapse' with VAEs, etc.. [20] proposes a β penalty over the KL-divergence term in the VAE objective. Higher β values results in higher disentanglement performance, with the trade-off of the reconstruction quality. Wasserstein VAE [48] introduce the Wasserstein distance as a regularizer over latent space and report improved reconstruction quality. [31] propose to use GAN objective by stacking a discriminator after the decoder of VAE, promoting VAE to increase the reconstruction quality. Works such as [10], [57] propose use of modeling the latent space with the mixture of Gaussian to perform the clustering and to capture the richer semantic features. For this work, we are only considering the vanilla VAE and the β-VAE versions - since we are not focused on achieving good reconstructions but on achieving better domain generalization/adaptation performance. Better reconstruction quality is assumed to not necessarily strongly correlate with the classification performance in DA/DG setting. Also, experimentally, we find for our case, using a Gaussian prior is good enough.

The work presented in [36] claimed with empirical evidence that purely unsupervised disentanglement learning may not be possible without the inductive biases, introduced from the model architecture, objective function and/or the data. It also shows that well-disentangled models can-not be distinguished without supervision. In this work we impose explicit inductive biases in the model architecture and the supervised objectives, based on our hypothesis about the methodology that can yield better DG/DA performance. Another recent work presented in [39] proposes that higher disentanglement performance does not strongly correlate with the combinatorial generalization i.e generalization on combinations of generative factors unseen during training. However, experiments in that work primarily focussed on vanilla VAE models trained in an unsupervised manner; and they evaluate generalization based on the reconstruction quality disentanglement score; and on limited synthetic datasets; - all of which differ from our approach and the problem setting.

4.2 Domain Generalization

Many of approaches that have been proposed for the domain generalization can be broadly categorized into three types: *(a) Model-level methods*, *(b) Feature-level methods*, *(c) Data-level methods*. *Model-level methods* modify model architecture to have separate modules to encode information about each of the domains. [17] leveraged multi-task learning with separate decoders for each of the domains, [34] proposes low rank decomposition based method for separate domains, while [12] proposes adding domain specific aggregation modules for each domain. One of the limitation of such approaches lies in the design principal that network parameter complexity increases with the number of source domains. These methods do not allow network to learn the invariance to domains but guide the flow of information through network based on the domain label. *Feature-level methods* like [16], [35] introduce adversarial constraints to remove domain information from latent features, [40] and [44] explore approach to project samples from the same class close to each other. [24], and [3] propose VAE based method that rely on the KL-divergence term to restrict the information being encoded about the domain identities. *Data-level methods* like [46] propose domain guided perturbations across multiple domains while [52] presents approach of adversarial perturbations that is possible even with a single source domain. These approaches aim to increase diversity of the original data samples by injecting cross-domain knowledge into instances.

Recent success of self-supervised learning has inspired many DG methods utilizing self-supervision tasks as a pre-training or a joint onjective. [6] and [4] use the task of solving Jigsaw puzzle to improve the generalization performance. While, [14] makes use of rotation angle prediction task as an auxiliary objective and reports improvement in the domain generalization performance.

4.3 Domain Adaptation

Recent advancements in semi-supervised learning and weak-supervised learning have inspired a new set of domain adaptation frameworks. [28] focuses on reducing the intra-cluster discrepancy between the source and target domain clusters. For this purpose, authors propose a DA specific domain adaptive adversarial perturbation method and a scheme to align the local features in class dependent manner. [25] proposes a general bidirectional adversarial training scheme that instead of adding noise in arbitrary direction, guides the network to close the domain gap (aka domain shift) in more structured manner. Amongst other methods, [18] proposed a conditional entropy minimization approach to domain adaptation. [33] proposed a pseudo-label based semi-supervised learning. Many of recent approaches like [38], [9], [47] make the cluster assumption - decision boundaries should not cross high-density regions [7]. [38] proposes adversarial perturbations to create robust decision boundaries, [9] analyzes use of combined training of GAN and the SSL classification objectives while [47] proposes iterative adversarial training and combines objectives from the other works like entropy minimization and mixup training.

[37] is a model-level method that proposes to use separate layers for encoding information about each of the domains. And then uses adversarial training to reduce the joint maximum mean discrepancy(JMMD) between the domains to reduce the shift between the domains. Some of the other methods that focus on the feature alignment include [44] which uses difference in moments of the source and the target distributions to measure the domain gap and close it while [35] uses MMD measure to measure and minimize the gap. [16] proposed one of the very first domain adversarial training strategies which uses adversarial trained neural networks instead of measure like moments or MMD to measure the discrepancy. An adversarial branch is trained to classify the embeddings

of the source and the target domains. The backbone network is encouraged to maximize the loss of this branch which can be done by making source and the target representations indistinguishable. [55] extends this idea by incorporating mixup objective into the adversarial training.

Domain adaptation approaches presented in the literature are evaluated in one of the following manner based on number of source and target domains: (a) one source-one target [16], [37], (b) multi source-multi target, (c) multi source-one target [44], [56] (d) one source-multi target [8] settings. Among these, (a) and (c) are the most common choices in the literature.

An another way domain adaptation approaches are classified in the literature is open-set vs closed-set setting. Closed set [16], [24] is the setting where assumption is made that the source and the target domains have same classes or categories of interest present. Open set [5] setting makes no such assumption and target domain may or may not have categories from source domain present. Closed set is the popular setting studied while open-set methods have good overlap with the area of study of meta-learning. Closed-set methods are the area of focus of this work.

5 Methodology

5.1 Generative modeling with the variational autoencoder framework

For building generative model, we utilize deep variational autoencoder(VAE) framework. We aim to the learn the latent generative factors z from the training data distribution x by learning the likelihood $p_\theta(x|z)$. VAE models the generative distribution of data x, $p_\theta(x|z)$; with a decoding neural network parameterized by θ. It then models the approximated posterior distribution of the latent variable z, $q_\theta(z|x_n)$, by another encoding neural network parameterized by ϕ. This gives rise to a encoding-decoding architecture that can be optimized by maximizing the *evidence lower bound* (ELBO) of the marginal likelihood of the observed data $x_{n}{}_{n=1}^{N}$. β-VAE [20] is a variant of the VAE that increases the penalty on the KL divergence term in the ELBO objective and encourages disentanglement. Thus ELBO objective becomes:

$$\mathcal{ELBO} = \frac{1}{N}\sum_{n=1}^{N} {}_{q_\phi(z|x_n)}[\log p_\theta(x_n|z)] - \beta KL[q_\phi(z|x_n)||p(z)] \tag{8}$$

where $p(z)$ defines the prior distribution of z and $KL(||)$ refers to the non-negative Kullback-Leibler divergence between the prior and the approximate posterior. As a common practice, the prior is often assumed to be an isotropic Gaussian $\mathcal{N}(0,1)$, and the posterior $q(z|x)$ distributions are parameterized as Gaussians with a diagonal co-variance matrix. The value of the β penalty indicates the amount of regularization on the latent space and so also has the effect on the classification performance. The first term of the Eq. 8 can be approximated with the reconstruction error between the reconstruction generated by the decoder and the input; and can be approximated by the mean squared error or binary cross entropy formulation.

5.2 Domain generalization via disentanglement and adversarial learning

5.2.1 Systematic disentangled representation learning

For our domain generalization and adaptation purpose, we extend the concept of vanilla VAE with single latent code z further by splitting the bottleneck layer into three separate latent factors or codes. We assume that image generation process is guided by a three high level sets of factors - class identity related factors, visual domain identity related factors, and the factors that are common across classes domains or 'noise' factors. In class identity related code, we can force network to explicitly encode the factors related to the class identity of the input samples, and encourage invariance to the which domain the sample came from and to the noise factors; via adversarial constraints. The second code can be used to explicitly encode the specific domain related factors, while being invariant of the class identity noise factors. And an additional noise code can be used to encode common factors or noise while being invariant to both class and the domain identity.

Since we aim to model three separate codes clas(C), domain(D) and noise(Noi) in the VAE latent space; Eq. 9 now will have three separate KL terms:

$$\begin{aligned}\mathcal{ELBO} = \frac{1}{N}\sum_{n=1}^{N} {}_{q_\phi(z|x_n)}[\log p_\theta(x_n|z)] - (\beta_C KL[q_{\phi'}(z_C|x_n)||p(z_C)] + \\ \beta_D KL[q_{\phi''}(z_D|x_n)||p(z_D] + \beta_{Noi} KL[q_{\phi'''}(z_{Noi}|x_n)||p(z_{Noi}])\end{aligned} \tag{9}$$

; where $z = z_C \oplus z_D \oplus z_{Noi}$ and $\oplus$ denotes the concatenation operation over samples obtained from class code, domain code and the noise code. To encode class and domain related information in specific codes by making use of class and domain labels that are available, we can minimize the objective in Eq. 10 with the negative log likelihood with the binary cross-entropy loss formulation.

$$\mathcal{L}_{\text{sup}} = -\frac{1}{N}\sum_{n=1}^{N} log(p_\theta(\hat{y}_n|x_n)) \tag{10}$$

; where $\hat{y}$ is the estimated output probability of sample belonging to class C. Since we aim to encode class as well as domain related information, overall supervised loss would be aggregate of object classification loss (sup-C) and domain classification loss (sup-D):

$$\mathcal{L}_{\text{sup-aggr}} = \mathcal{L}_{\text{sup-C}} + \mathcal{L}_{\text{sup-D}} \tag{11}$$

Since model can learn to disentangle the domain related changes from the object category related changes and the noise, we hypothesize that such ability can help to generalize better. We will discuss the aforementioned adversarial constraints in the next section.

5.2.2 Domain adversarial learning

As mentioned earlier, we aim to model class identity, domain identity and other noise factors in independent manner in separate and invariant codes. Previous approaches [21], [3] that use factorized VAE framework for generalization only use supervised classification losses and rely on KL-divergence term to limit the information being encoded by each of the factors. However, from our experiments, we observe empirically, that KL-Divergence penalty is not often enough constraint and we find that application of adversarial losses can further improve the invariance. This observation is in accordance with the success of approaches presented in [30], [42] which utilize explicit adversarial losses to 'remove' the information about the known factors of variations from the representations for image manipulation and classification tasks respectively.

In order to impose adversarial losses, we would split each iteration of the the training process into two steps - (a) steps to train the adversarial branches with the classification loss; (b) steps to train the VAE with the adversarial loss as calculated by the adversarial branch. We can compute the classification loss for the adversarial branch with Eq. 12. For the VAE to minimize the adversarial loss being propagated it'd need to maximize the classification loss for the adversarial branch. Thus, we can treat negative of the loss as computed by Eq. 12 as the adversarial loss. Thus,

$$\mathcal{L}_{\text{sup-disc}} = -\frac{1}{N}\sum_{n=1}^{N} log(p_\theta(\hat{y}_n|x_n)) \tag{12}$$

$$\mathcal{L}_{\text{adv}} = -\mathcal{L}_{\text{sup-disc}} \tag{13}$$

For clarity, with this framework, we will have 4 adversarial losses: (a) Loss to remove domain information from the class code (adv-DC); (b) Loss to remove class information from the domain code (adv-CD); (c) and (d): Loss to remove class and domain information from the noise code (adv-NC adv-ND). Thus, aggregate adversarial loss becomes:

$$\mathcal{L}_{\text{adv-aggr}} = \mathcal{L}_{\text{adv-DC}} + \mathcal{L}_{\text{adv-CD}} + \mathcal{L}_{\text{adv-NC}} + \mathcal{L}_{\text{adv-ND}} \tag{14}$$

Thus, aggregate loss function for domain generalization becomes:

$$\mathcal{L} = \mathcal{ELBO} + \mathcal{L}_{\text{sup-aggr}} + \mathcal{L}_{\text{adv-aggr}} \tag{15}$$

5.3 Domain Adaptation

The domain adaptation framework extends DG setting further by allowing models to transfer or adapt to the new domain, without labels, and tests performance after the adaptation process is complete.

5.3.1 Domain Adaptation via semi-supervised learning:

For our methodology, we consider, label-guessing approaches presented in the semi-supervised learning, specifically the Mix-Match [2] algorithm. Mix-Match approach performs label guessing by multiple augmentations of unlabeled data samples, does entropy reduction with temperature-scaling and mixes labeled and unlabeled data samples with Mixup [51]. In particular, MixMatch performs K random augmentations of the input image and generates k output probability predictions. These k outputs are then averaged over to generate a single probability estimates over C classes as shown in Eq. 16. Inspired by success of entropy minimization in SSL, this average prediction is sharpened with Eq. 17.

$$p = \frac{1}{K}\sum_{k=1}^{K} P_{model}(y|u_k;\theta) \tag{16}$$

$$Sharpen(p,T)_i = p_i^{\frac{1}{T}} / \sum_{j=1}^{L} p_j^{\frac{1}{T}} \tag{17}$$

; where *u* is an unlabeled data sample, θ are the parameter of the neural network, *p* is the average probability outcome from k augmentations, *T* is the Temperature parameter(a hyper-parameter) for entropy minimization. As $T \to 0$; output approaches Dirac(one-hot) distribution. We can employ the same version of label guessing approach to calculate supervised loss for unlabeled data from the guessed labels. Samples with guessed output probabilities are treated as labeled samples and are 'mixed' with the labeled samples that have the known ground truth. MixMatch approach assigns higher weight to the sample with known ground truth while mixing, which allows to minimize the noise from the incorrectly guessed labels. This mixup typically done in a linear manner. Consider two data samples x_1 and x_2 with associated class probabilities, either from the ground truth or guessed, as y_1 and y_2 then linear mixup can be given by Eq. 18 to 21.

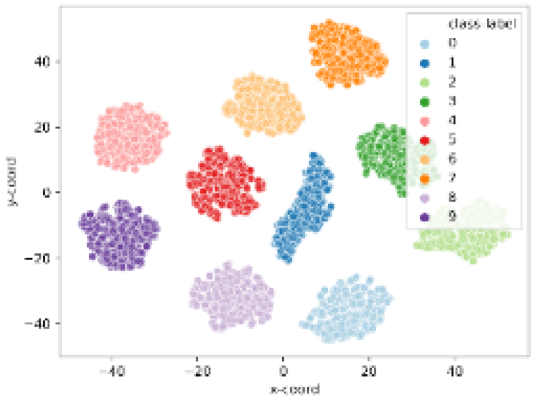
$$\lambda \sim Beta(\alpha,\alpha) \tag{18}$$

$$\lambda^{'} = max(\lambda, 1-\lambda) \tag{19}$$

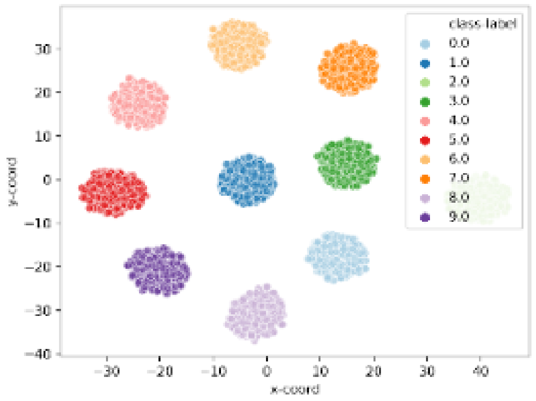
$$x^{'} = \lambda^{'} x_1 + (1-\lambda^{'})x_2 \tag{20}$$

$$y^{'} = \lambda^{'} y_1 + (1-\lambda^{'})y_2 \tag{21}$$

; where $\alpha > 0$ and is a hyper-parameter. α acts as a shape-parameter that defines the Beta distribution from which mixing proportion λ is sampled. x can be input sample or latent embedding and y is the class probability distribution for each class - and is either from the ground truth or is guessed. Model will be trained on mixed (data, label) pairs $(x^{'}, y^{'})$. As we discussed in **??**, mixup training leads to better representations in general. Thus, we employ mixup not just for the domain adaptation but also in the domain generalization framework. Effect of using mixup is shown in Fig. 3.



(a) Without Mixup training



(b) With Mixup training

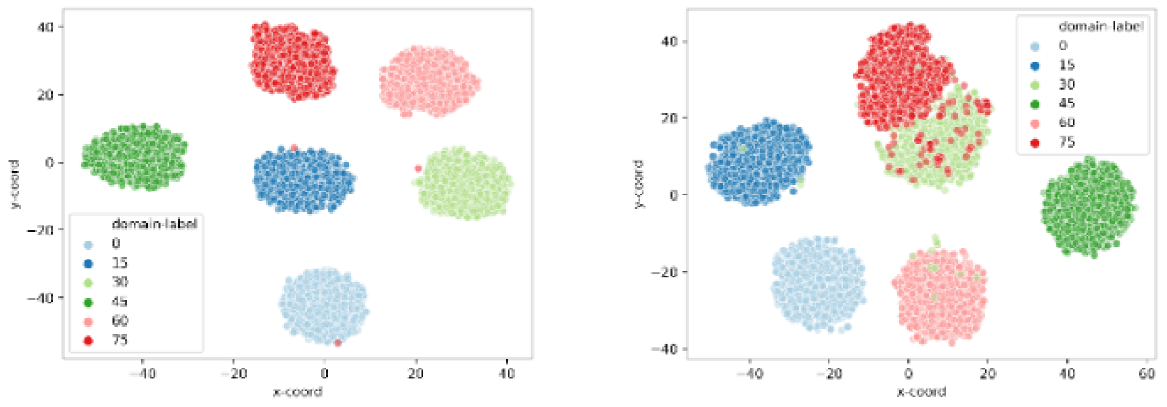
Figure 3: Effect of using mixup training on embeddings learned at the class code. Each cluster represents and is color coded by the digit identity for the training samples of the MNIST dataset. Mixup training creates well-separated, tight clusters which may help to generalize better.

5.3.2 Augmentation via latent domain code sampling:

We also propose a new technique that can be used in place of or alongside with the input augmentations. Since we are learning separate code to encode the domain related information, in the VAE latent space, we can pair the generate embeddings in the class code with K samples from any of the clusters in the domain code. classifier that estimates the class labels for unlabeled data can predict output distributions for the stacked representations from the class code and the domain code. We call this approach ’domain augmentation’. Performing domain level augmentations in the latent space is cheap compared to augmentations in the input space since only single forward pass is required and can perform on-par with the input augmentations method as we show in the evaluation section.

5.3.3 Domain Adaptation via regression adversarial constraints over latent space:

Further, we propose a new adversarial loss based on our observations about the latent space of VAE. We find that when we train network with K source domains, network, as expected learns K clusters in the domain code. But when we run inference with samples from new unseen domain, instead of confusing this new domain with one of the seen domains, network extrapolates and projects samples to a new cluster in the domain code. If we run inference on two unseen domains, network projects samples onto two well separated clusters. We can make use this behavior for domain adaptation, in scenarios where we may get samples from new domain but without the domain label. Since our previous classification based adversarial loss would not work in this case, we can use still achieve invariance by computing regression based adversarial loss. To achieve this, from the embeddings of the class code, we try to estimate the domain code. Since samples from new domain have distinct cluster in the domain code, network would need to make class code samples invariant to this new cluster, and we can achieve invariance without needing domain label information.



(a) Target domain 75°　　(b) Target domains: 30° and 75°

Figure 4: Samples from unseen target domain get projected to a separate cluster in the domain code.

Since we consider the case of not having domain label for new samples, we can't use binary cross entropy loss as we did for classification based losses and instead, we can use 'mean squared error' to estimate the loss to train the adversarial branch as shown in Eq. 22. Loss being back-propagated can be measured with respect to the prior of $\mathcal{N}(0,1)$ and given with 23. the We also find that combining regression and classification based adversarial losses help achieve better domain adaptation performance, as we show in the experiments section.

$$\mathcal{L}_{\text{regr}} = \frac{1}{N}\sum_{n=1}^{N} ||z_d - \hat{z}_d||^2 \tag{22}$$

$$\mathcal{L}_{\text{adv-regr}} = \frac{1}{N}\sum_{n=1}^{N} ||\hat{z}_d - z_{N(0,1)}||^2 \tag{23}$$

Thus aggregate loss function for domain adaptation framework becomes:

$$\mathcal{L} = \mathcal{ELBO} + \lambda_1\mathcal{L}_{\text{sup-aggr}} + \lambda_2\mathcal{L}_{\text{adv-aggr}} + \lambda_3\mathcal{L}_{\text{adv-regr}} \tag{24}$$

; where λ_1, λ_2, and λ_3 are the scalers that determine the influence of each of the constraints.

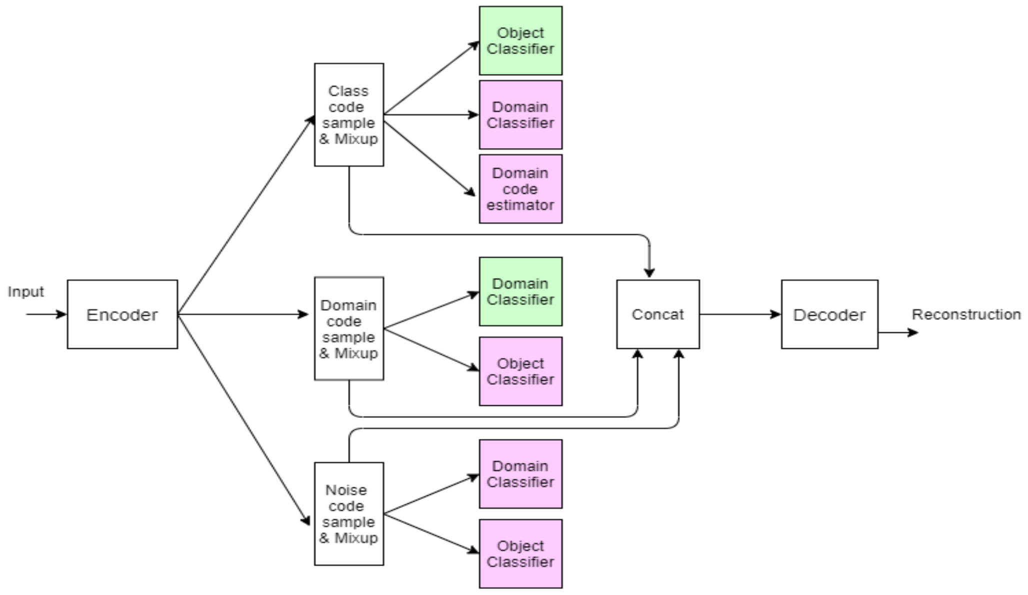


Figure 5: High-level schematics of the proposed model architecture. The green boxes indicate the fully connected branches that backpropagate the classification loss to the backbone model while the red boxes indicate the adversarial branches that backpropagate the adversarial loss to the backbone.

6 Evaluation

6.1 Datasets

We will evaluate our approach on commonly used benchmark datasets in the DG and DA literature. The most popular domain generalization benchmark datasets are: "VLCS" [49], [27] and the MNIST-rotation dataset. 'VLCS' contains 5 image categories from four domains: Pascal-**V**OC 2007, **L**abelMe, **C**altech[13], **S**UN09 datasets. VLCS dataset has 5 classes: 'car', 'bird', 'chair', 'dog', and 'person'.

For the VLCS dataset, to provide fair comparison with the past methods, we'll evaluate based on the models trained on pre-extracted DECAF features[15]

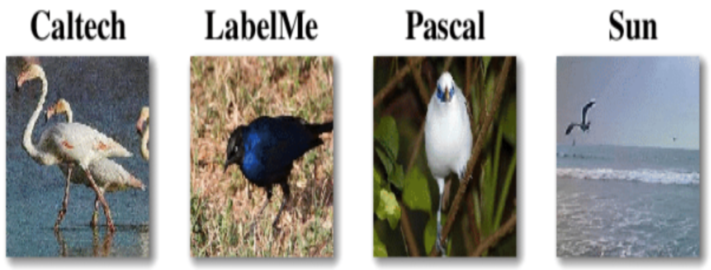


Figure 6: Data samples from the VLCS dataset. Images in the 'V', 'L', and 'S' domains are scene-centric whereas images in the 'C' domain are object-centric.

The MNIST-rotation dataset is created by taking a random subset of the original MNIST digits dataset, has 10 classes of digits from 0 to 9. Images in this subset are rotated at 6 rotation angles - $0°$, $15°$, $30°$, $45°$, $60°$, $75°$ and each of the rotation angles is treated as a domain.

To evaluate domain adaptation performance, we will use 'Digit-Five' and 'Office-31' [45] datasets. 'Digit-Five' dataset is a collection of data samples from five popular digit datasets which treated as the domains: MNIST [32], MNIST-M [15], Synthetic Digits [15], SVHN [41] and USPS [23]. It naturally has 10 classes, digits from 0 to 9, and and has varying degrees of similarities with one another.

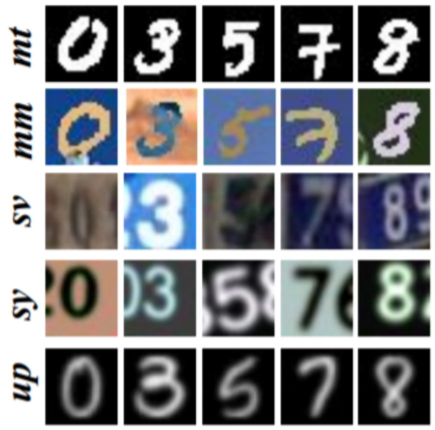


Figure 7: Data samples from the Five Digits dataset. . Abbreviations: mt: MNIST, mm: MNIST-M, sv: SVHN, sy: Synthetic-Digits,up: USPS

An another commonly used dataset to benchmark the domain adaptation methods is the ‘Office-31’ dataset consists of images from three visual domains - Amazon, Webcam, and DSLR. [13]. The dataset contains 4110 images from 31 classes. Images in the ‘Amazon’ domain are the images of the products on the amazon.com website and are high resolution images with clean backgrounds. ‘DSLR’ domain images are high resolution and low-noise images. Each class has 5 images and each image has images from, on average, 3 viewing angles. The ‘Webcam’ domain has a lower resolution images, with much higher amount of noise, color range and other artifacts.

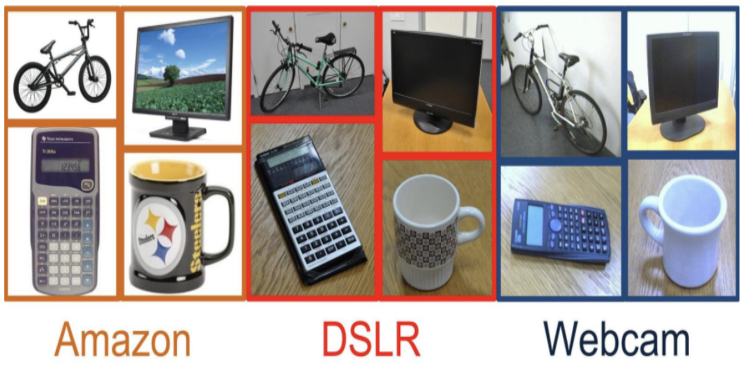


Figure 8: Data samples from the three domains of the Office-31 dataset.

7 Experiments

For the VLCS dataset, the models are trained with the DECAF[49] features, a set of pre-extracted 1-dimentional features, we use fully connected VAE. Encoder is composed of a single fully connected hidden layer with 4096 units followed by a bottleneck layer of size 1024.

For the MNIST-Rotation dataset, we used a simple CNN with 2 blocks of Convolution-Batch Normalization-Max Pooling layer sequence, followed by 2 fully connected layers, to get to the embedding of size 1024. For the model that we used for visual analysis of latent units via traversals (Fig. 11), we used a simpler model with the class code having a capacity of 5 units, domain code with 5 units and the noise code with the capacity of 15 units. This choice was made, because with large number of latent units, it gets harder to observe the effect of traversals over units. Also, disentanglement quality may suffer with large bottleneck size, effect of which we were interested in observing in the visual analysis.

For the experiments with the Five-Digits dataset, we used a 3 layer deep convolutional neural network(CNN), as referred from the other works. We use the symmetric decoder network. The size of the bottleneck layer kept at 1024 units, with separate encoders i.e 3 CNN encoders trained in parallel, extracting three codes, each of size 1024. A single decoder aggregates all three codes to reconstruct the input. We use binary-cross-entropy loss as the reconstruction loss.

For Office-31 dataset, we use ResNet-50[19] architecture as an encoder. Because of resource constraints, we use single shared encoder to extract the information about all three codes. As we obtain 512 sized feature vector from the ResNet-101, we project these as three codes with three separate branches. The size of each code is kept as 512 for fair comparison. The decoder architecture we used in this is a simple 12 layers of Convolution, Batch Normalization, followed by ReLU activation. The Decoder is kept shallow because of 1] resource constraints, 2] to avoid possible posterior collapse like problems that are usually observed with very deep decoder architecture with

VAEs.

For the data augmentation needed for the semi-supervised method, use used below set of transformations, each of which is applied with the probability of 50%. random horizontal or vertical Flips, random rotations within $45°$ on either side of the axis, random translation upto -10% to +10% range, random scaling within 80% to 120% range.

Training schedule for backbone network and adversarial branches: As discussed, we have adversarial components in our networks. We train adversarial branches in similar alternating fashion as many GAN related works i.e. training the adversarial branch on the classification objective for a few iterations and then training the backbone network for few iterations. We use separate optimizers for training the adversarial branches, thus, only the weights in the adversarial branches are updated to learn the classification objective. While the backbone network is trained, adversarial loss is computed and backpropagated through the adversarial branches for backbone network to create invariant representations. However, this adversarial loss does not update the weights of the layers in the adversarial branches. All the trainable parameters of the adversarial branches are 'frozen' while backpropagating the adversarial loss. For MNIST-Rotation dataset, we found it useful to train backbone for 2 epochs and adversarial branches for 2 epochs. For VLCS, Five-Digits office-31 datasets, we train backbone for 6 epochs and adversarial branches for 3 epochs, in an alternating manner.

All classification-based loss are computed with binary cross entropy loss function. The regression adversarial losses are computed as the mean squared loss. For the reconstruction loss, we use binary cross entropy loss, since we are using images scaled between 0 - 1 range. For the experiments with the VLCS dataset, as DECAF features are not scaled, we use mean squared error loss for the reconstruction objective.

For the β parameter as seen 9, we used value 1.0, except for the Office-31 dataset. For the Office-31 dataset, we used 0.5 as the β-penalty for computing the KL-divergence for the class code, and used the value 1.0 for the domain and the noise code. For all the datasets, we use α, the shape-parameter of the beta distribution for mixup, as 1.0. Semi-supervised domain adaptation methods use multiple data augmentations to guess the labels of the unlabeled data sample. For this purpose, we experimented with 2 and 4 augmentations for initial experiments and using four augmentations was found to be marginally better than two augmentations. We used sharpening parameter in 17 with the value of 1.

Rest of the hyper-parameters, such as learning rate, l2-penalty etc. were kept to the values commonly found in other DG/DA methods and we did not tune them for a fair comparison.

8 Results

8.1 Qualitative Results / Analysis

We perform visual analysis of the bottleneck layer of the VAE framework to validate our goal of invariant and disentangled feature learning. We aim to this in two ways: 1] by analyzing image reconstruction output of VAEs by manipulating the latent code and 2] by performing t-SNE(t-stochastic neighbor embeddings) analysis over representations at the VAE bottleneck.

8.1.1 Reconstruction based analysis

We consider two ways to perform reconstruction based analysis: (a) Conditional generation, and (b) Traversals over latent units.

Conditional Generation

In conditional generation, we switch the conditions or the latent code that is input the decoder and observe the reconstruction. With disentangled representations between codes, we expect to see domain related changes(for example, the rotation angle in case of MNIST-Rotation dataset) and no object identity related changes (i.e. the digit identity). Similarly, when class code is changed, we expect to see the change only in the class identity and not in the rotation angle. To do this, we will fix two of the codes, class or domain or noise code, and sample from the different clusters learned by the remaining third factor.

Figures 9 and 10 show initial results of successful disentanglement between class and the domain codes. In fig. 9, every three columns show reconstructions by a fixed sample from the class code from three input images of that class and sample is taken from each of the cluster learned by the domain(rotation angle in this case) related code. Noise code kept unchanged. While in fig.

10, every three columns show reconstructions with a fixed sample from the domain related code from three input images of that domain and sample is taken from each of the cluster learned by the class(digit identity in this case) related code while Noise code kept unchanged.

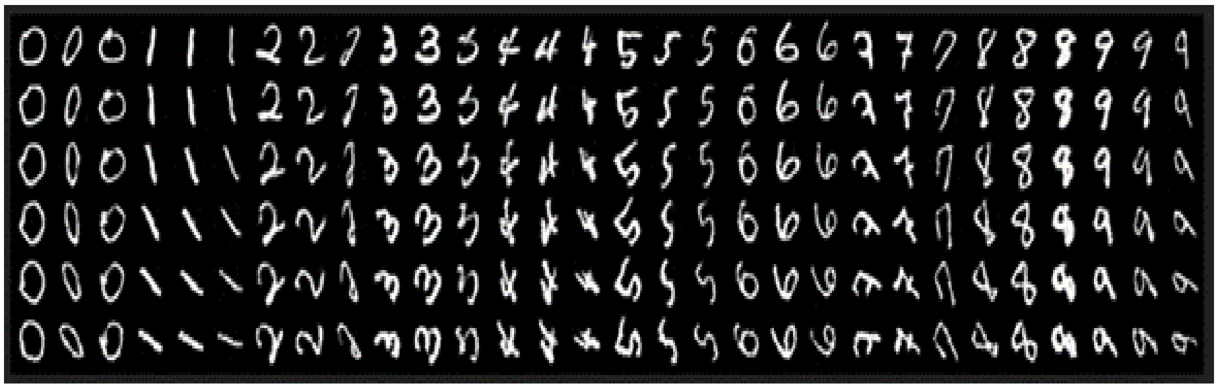


Figure 9: Reconstructions with fixed sample from the class code and noise code while sampling from switching domain code samples to represent rotation angles.

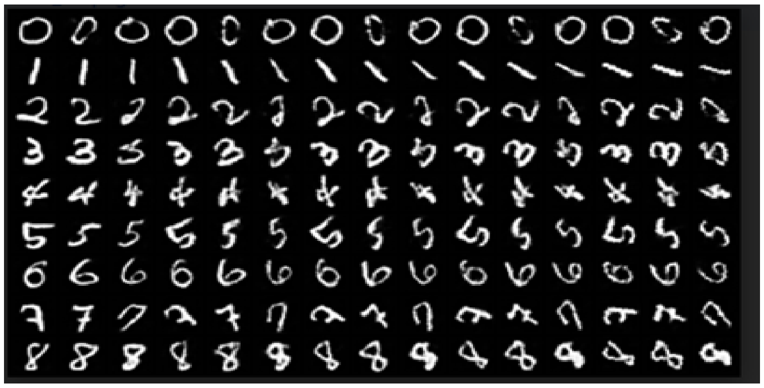
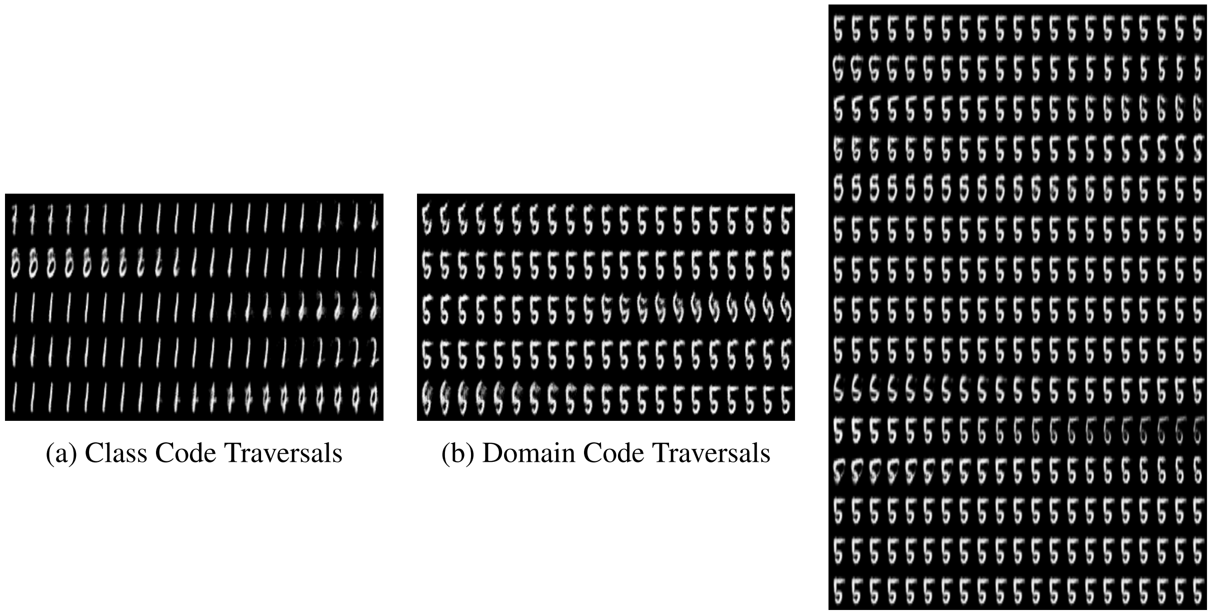


Figure 10: Reconstructions with fixed sample from the domain code and sampling from distributions of the class code.

Traversals over latent units

As discussed earlier, within each of the three codes, we can expect to see disentangled behavior amongst different units. For example each unit in the class code may correspond to the change in the specific digit identity. Since we are learning distributions for each of the latent units in the VAE bottleneck layer, we can sample from the learned distributions. For more systematic analysis, we can traverse from lower to higher sample values over the learned distributions for each unit. We fix samples from rest of the units and analyze reconstruction by traversing one unit at a time. Figure 11 indicates example of traversals over each of the three codes. We can see indication of disentanglement that class code traversals primarily encode change in the digit identity, domain code traversals reflect change in rotation angles in the reconstruction and noise code traversals result in change in minor factors like thickness of strokes, writing styles for digits etc. and is insensitive to explicit rotation angles(domain) and digit identity(class). It's worth noting that, some rotation behavior seen in the noise code traversal is assumed to be caused by different writing styles in the original images and not caused by the explicit rotations of images.

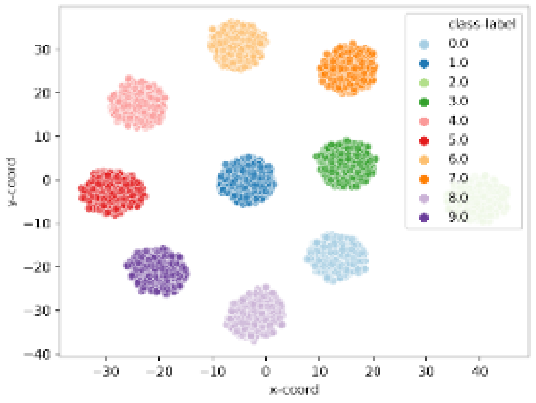


(a) Class Code Traversals (b) Domain Code Traversals (c) Noise Code Traversals

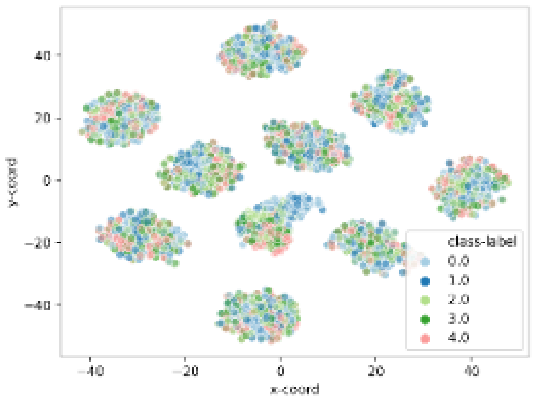
Figure 11: Traversals over latent units in class code, domain code and the noise code. Each row represents one unit in the bottleneck layer and each column is a sample. Sample values increase from left to right. In this model, class code has 5 units, domain code has 5 units and noise code has 15 units capacity.

8.1.2 Embedding t-SNE analysis

We will perform in-depth t-SNE analysis by plotting embeddings at the bottleneck layer of the network. We can analyze these plots for class code, domain code and the noise code separately. Color coding based on class identity and the domain identity of each of the data points would give us two point of views of the embeddings.

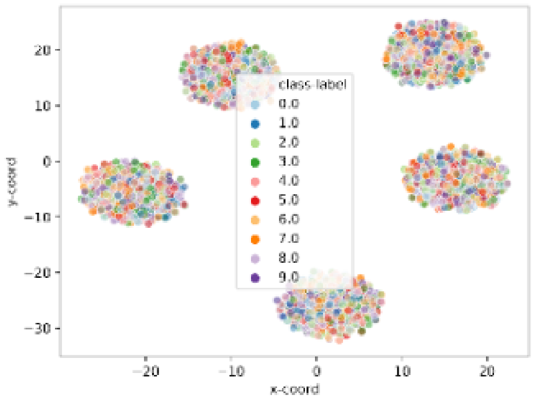


(a) Color Code: class-id

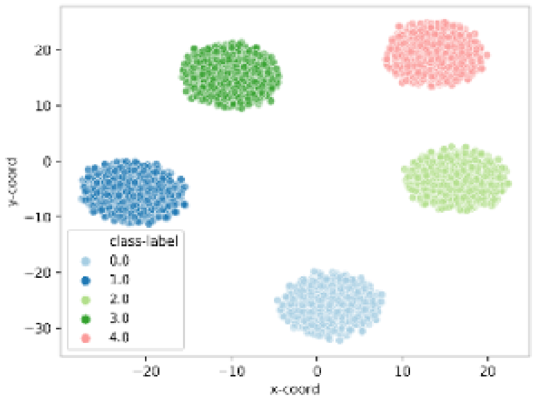


(b) Color Code: domain-id

Figure 12: The embeddings generated at the class code. Well-separated clusters representing digit identities are formed while being invariant to the domain identity. The first image is color coded by the class-id of each of the inputs and the second image has each data point representing the domain-id of the input data.



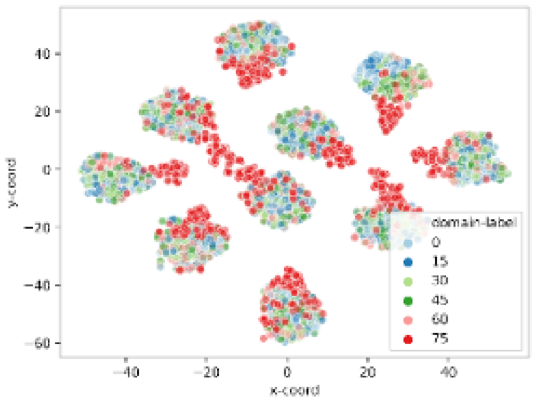
(a) Color Code: class-id



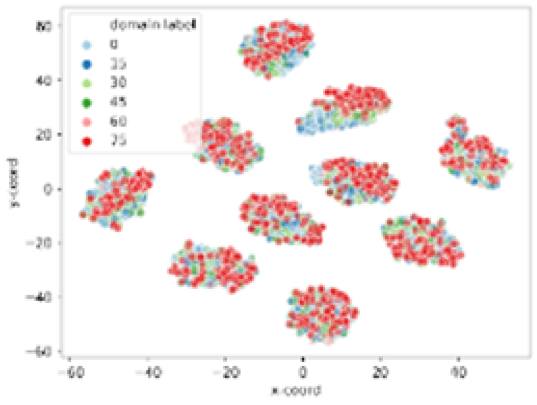
(b) Color Code: domain-id

Figure 13: The embeddings generated at the domain code. Well-separated clusters representing the domain identities are formed while being invariant to the domain identity. The first image is color coded by the class-id of each of the inputs and the second image has each data point representing the domain-id of the input data.

Figure 14 shows t-SNE plots before and domain adaptation. Network before adaptation is uncertain about the new data samples and samples lie on the boundary of each clusters. As the network adapts to the target domain, network becomes more and more confident and correct about the samples from the target domain and target domain samples eventually merge almost completely into the class specific clusters. We also plan on presenting in depth analysis for both domain generalization and adaptation scenarios.



(a) Before domain Adaptation



(b) After domain Adaptation

Figure 14: t-SNE plots generated with the embeddings from the class-code. Domain label ’75’(red) is the target domain in this case. Rest are the source domains that the model is trained with.

8.2 Quantitative Evaluation

As is the common practice in the literature, both generalization and the adaptation methods will be evaluated and compared with other methods based on the top-1 classification accuracy metric. Neural network architectural choices will be made to allow fair comparison with other methods. We also plan to perform a detailed ablation study demonstrating contribution of architectural choices, data augmentation strategies employed for label guessing, and different components of the objective function. For the baseline of domain generalization approaches, we will use a neural network trained with an i.i.d. assumption. For Domain adaptation, we will use our best domain generalization approach as a baseline, also referred to as ’Source-Only’. Apart from comparison with the baseline and comparison of configurations in the ablation experiments, we will also compare our approach with popular and state-of-the-art methods in the literature. For domain generalization we can compare works presented in [34] , [17], [35], [6] etc.. For domain adaptation, some of the comparison points can be the approaches presented in [16], [37], [44], [56], [8].

Domain Generalization - MNIST Rotation						
Target Domain	DA[16]	LG[46]	HEX[53]	ADV[53]	DIVA[24]	**Ours**
0°	86.7	89.7	90.1	89.9	93.5	89.7
15°	98.0	97.8	98.9	98.6	99.3	**99.7**
30°	97.8	98.0	98.9	98.8	99.1	**99.2**
45°	97.4	97.1	98.8	98.7	99.2	98.9
60°	96.9	96.6	98.3	98.6	**99.3**	**99.3**
75°	89.1	92.1	90.0	90.4	93.0	**93.5**

Table 1: Comparison of our approach with other state-of-the-art domain generalization methods for the MNIST-rotation dataset where rotation angle is treated as a domain. Source domains are all the domains, except the mentioned target domain, from the set $(0°, 15°, 30°, 45°, 60°, 75°)$.

Domain Generalization - VLCS Dataset							
Target Domain	D-MTAE[17]	MMD-AAE[35]	DANN[16]	AGG	JiGen[6]	MASF	**Ours**
V	63.9	67.7	66.4	65.4	70.62	69.14	**70.87**
L	60.1	62.6	64.0	60.6	60.9	**64.90**	63.84
C	89.1	94.4	92.6	93.1	**96.93**	94.78	93.1
S	61.3	64.4	63.6	65.8	64.30	**67.64**	65.28

Table 2: Comparison of our approach with other state-of-the-art domain generalization methods for VLCS dataset.

Domain Adaptation - Five-Digits					
Method	mt,up,sv,sy $\rightarrow$ mm	mm,up,sv,sy $\rightarrow$ mt	mm,mt,sv,sy $\rightarrow$ up	mm,mt,up,sy $\rightarrow$ sv	mm,mt,up,sv $\rightarrow$ sy
Source-Only	63.37	90.50	88.71	63.54	82.44
DANN [16]	71.30	97.60	92.33	63.48	85.34
JAN [37]	65.88	97.21	95.42	75.27	86.55
ADDA [52]	71.57	97.89	92.83	75.48	86.45
DCTN	70.53	96.23	92.81	77.61	86.77
MEDA	71.31	96.47	97.01	78.45	84.62
M^3SDA[44]	**72.82**	98.58	96.14	81.32	89.58
Ours - Input Aug.	70.28	98.65	98.11	80.12	95.57
Ours - Domain Aug.	72.34	98.86	99.05	81.15	**96.1**
Ours - Input + Domain Aug.	71.43	**98.63**	**99.24**	**81.85**	95.98

Table 3: Comparison of our approach with other state-of-the-art domain adaptation methods. 'Input Aug.': 4 augmentations per sample in the input space. 'Domain Aug.': 4 augmentations with domain cluster sampling in the latent space. 'Latent Domain Aug.': 2 input augmentations followed by 2 domain augmentations per sample. Abbreviations: mt: MNIST, mm: MNIST-M, sv: SVHN, sy: Synthetic-Digits, up: USPS.

Domain Adaptation - Office-31						
Method	A $\rightarrow$ W	D $\rightarrow$ W	W $\rightarrow$ D	A $\rightarrow$ D	D $\rightarrow$ A	W $\rightarrow$ A
Source-Only	68.4	96.7	99.3	68.9	62.5	60.7
RevGrad [15]	82.0	96.9	99.1	79.7	68.2	67.4
DANN[16]	80.5	97.1	99.6	78.6	63.6	62.8
JAN [37]	85.4	97.4	99.8	84.7	68.6	70.0
MADA [43]	90.0	97.4	99.6	87.8	70.3	66.4
CAN [26]	**94.5**	99.1	99.8	**95.0**	78.0	77.0
Ours - Input Aug.	93.7	98.87	99.83	93.48	**80.38**	77.51
Ours - Domain Aug.	92.88	98.96	99.73	93.57	80.11	77.48
Ours - Input + Domain Aug.	93.17	**99.18**	**99.87**	93.51	80.32	**77.78**

Table 4: Comparison of the proposed approach with other domain adaptation methods. 'Input Aug.': 4 augmentations per sample in the input space. 'Domain Aug.': 4 augmentations with domain cluster sampling in the latent space. 'Latent Domain Aug.': 2 input augmentations followed by 2 domain augmentations per sample. Abbreviations: A: Amazon, D: DSLR, W: Webcam

We perform ablation analysis of some of the components of the objective function deigned, for the domain adaptation setting. We use the MNIST-Rotation for this purpose, to perform analysis with lower compute requirements.

Ablation Analysis	
Description	Top-1 Test Accuracy (%) Test Domain: 75°
DG w/o Adv. loss and mixup	91.1
DG w/ Adv. loss	92.73
DG w/ Adv. loss and mixup ($\alpha = 1.0$)	93.5
DA w/o Adv. loss	95.26
DA w/ labeled Adv. loss	98.34
DA w/ regress. Adv. loss	97.93
DA w/ labeled and regress. Adv. loss	99.37

Table 5: Ablation Analysis of the components of the loss function for the DG and DA settings. The 'labeled Adv. loss' refers to the adversarial loss assuming we know the domain label of the new samples. The 'regress. Adv. loss' is the adversarial loss based on estimating the domain code embeddings as explained in Section. 5.3.3. SSL Adaptation is done with with just standard input augmentations method.

9 Conclusion and Future Work

9.1 Conclusion

In this work, we presented a new approach for domain generalization and domain adaptation framework, based on generative modeling and representation learning. The domain generalization framework performs on par with the other state of the art methods while the domain adaptation approach improves state of the art on multiple tasks. We also conducted in depth analysis of the representations clusters learned by the neural network model validating the effectiveness of components like mixup training and adversarial invariance learning objectives.

We presented an alternative to the input data augmentations needed for the semi-supervised methods to perform label-guessing, called 'domain augmentation'; which results in improved computational efficiency and performance of the semi-supervised domain adaptation methods. In addition, we presented a novel adversarial objective that works on the VAE latent space and without explicitly needing the domain identity related information for the new samples to achieve domain invariance.

9.2 Future Work

Although invariance based approaches seem to work well; it could be interesting to explore if we can use the information about the class or cluster identities identities in richer manner. With objective functions that aim to achieve invariance, loose some of the information about the similarity of inputs, that otherwise gets naturally encoded by the generative modeling. Such similarity information that may not be available in the labels, can possibly be useful to develop a new set of generalization and adaptation methods.

One of the limitations of the proposed method is large number individual objectives, each of which has varying degrees of effect on performance. We performed a very limited hyper-parameter tuning in this work, because of its expensive nature. Most of the tuning was limited to 1] scaling of the classification and adversarial loss values for the class code and, 2] the training schedule. Further improvements might be possible with more thorough hyper-parameter search.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] D. Berthelot, N. Carlini, I. Goodfellow, A. Oliver, N. Papernot, and Raffel C. Mixmatch: A holistic approach to semi-supervised learning. *33rd Conference on Neural Information Processing Systems.*, 2019.

[3] D. Bouchacourt, R. Tomioka, and S. Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *AAAI Conference on Artificial Intelligence*, 2018.

[4] S. Bucci, A. D'Innocente, Y. Liao, F. Carlucci, B. Caputo, and T. Tommasi. Self-supervised learning across domains. *IEEE - TPAMI*, 2021.

[5] P. Busto and J. Gall. Open set domain adaptation. *International Conference on Computer Vision and Pattern Recognition*, 2017.

[6] F. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. *Conference on Computer Vision and Pattern Recognition*, 2019.

[7] O. Chapelle and A. Zien. Good semi-supervised learning that requires a bad gan. *AISTATS*, page 57–64, 2005.

[8] Z. Chen, J. Zhuang, X. Liang, and L. Lin. Blending-target domain adaptation by adversarial meta-adaptation networks. *Computer Vision and Pattern Recognition (CVPR)*, 29, 2020.

[9] Z. Dai, Z. Yang, Yang F., W. Cohen, and R. Salakhutdinov. Good semi-supervised learning that requires a bad gan. *Neural Information Processing Systems (NIPS)*, 2017.

[10] N. Dilokthanakul, P. Mediano, M. Garnelo, M. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoder. *CoRR*, abs/1611.02648, 2016.

[11] Z. Ding, Y. Xu, G. Parmar, Y. Yang, Max Welling, and Z. Tu. Guided variational autoencoder for disentanglement learning. *International Conference on Computer Vision and Pattern Recognition*, 2020.

[12] A. D'Innocente and B. Caputo. Domain generalization with domain-specific aggregation modules. *German Conference on Pattern Recognition*, pages 187–198, 2018.

[13] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *EEE Trans. Pattern Recognition and Machine Intelligence.*, 2006.

[14] Z. Feng, C. Xu, and D. Tao. Self-supervised representation learning from multi-domain data. *International Conference on Computer Vision*, 2019.

[15] Y. Ganin, V. Lempitsky, Y. Bengio, and P. Haffner. Unsupervised domain adaptation by backpropagation. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1180–1189, 2015.

[16] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, Hugo Larochelle, F. Laviolette, M. Marchand, and Lempitsky V. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

[17] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. *Conference on Computer Vision and Pattern Recognition*, 2015.

[18] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. *Neural Information Processing Systems (NIPS)*, 2004.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *International Conference on Computer Vision and Pattern Recognition*, 2016.

[20] I. Higgins, L. Matthey, A. Pal, C. Burgess, Hugo Larochelle, F. Laviolette, M. Marchand, X. Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. -vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017.

[21] H. Hosoya. Group-based learning of disentangled representations with generalizability for novel contents. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[22] S. Huang, x. Wang, and D. Tao. Snapmix: Semantically proportional mixing for augmenting fine-grained data. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

[23] J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

[24] M. Ilse, J. Tomczak, C. Louizos, and M. Welling. Diva: Domain invariant variational autoencoder. *ICLR 2019 Workshop on Deep Generative Models for Highly Structured Data*, 2019.

[25] P. Jiang, A. Wu, Y. Han, Y. Shao, M. Qi, and B. Li. Bidirectional adversarial training for semi-supervised domain adaptation. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[26] G. Kang, L. Jiang, Y. Yang, and A. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. *European Conference on Computer Vision, 2012. Proceedings. ECCV '12.*, 2012.

[28] T. Kim and C. Kim. Attract, perturb, and explore: Learning a feature alignment network for semi-supervised domain adaptation. *European Conference on Computer Vision*, 2020.

[29] D.P. Kingma and M. Welling. Auto-encoding variational bayes. *International Conference on Learning Representations(ICLR)*, 2013.

[30] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and Ranzato M. Fader networks: Manipulating images by sliding attributes. *Neural Information Processing Systems*, 2017.

[31] A. Larsen, S. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. *e International Conference on Machine Learning (ICML)*, 2021.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE '98.*, pages 2278–2324, 1998.

[33] D Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. *Workshop on Challenges in Representation Learning, ICML*, 3, 2013.

[34] D. Li, Y. Yang, Y-Z.. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. *IEEE International Conference on Computer Vision.*, pages 5542–5550, 2017.

[35] H. Li, S. Pan, S. Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. *Conference on Computer Vision and Pattern Recognition*, 2018.

[36] F. Locatello, S. Bauer, M. Lucic., G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *Proceedings of the 36 th International Conference on Machine Learning*, 2019.

[37] M. Long, H. Zhu, J. Wang., and M. Jordan. Deep transfer learning with joint adaptation network. *International Conference on Machine Learning*, 2017.

[38] T. Miyato, S. Maeda, Koyama M., and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *International Conference on Learning Representations (ICLR)*, 2016.

[39] M. Montero, C. Ludwig, R. Costa, G. Malhotra, and J. Bowers. The role of dientanglement in generalization. *International Conference on Learning Representations (ICLR)*, 2021.

[40] S. Motiian, M. Piccirilli, D. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. *Conference on Computer Vision and Pattern Recognition*, pages 187–198, 2017.

[41] Y. Netzer, T. Wang, A. Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.*, 2011.

[42] O. Ozdenizci, Ye. Wang, T. Koike-Akino, and D. Erdogmus. Transfer learning in brain-computer interfaces with adversarial variational autoencoders. *9th International IEEE EMBS Conference on Neural Engineering (NER'19)*, 2018.

[43] Z. Pei, Z. Cao, M. Long, and J. Wang. Multi-adversarial domain adaptation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[44] X. Peng, Q. Bai, X. Xia., Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. *International Conference of Computer Vision*, 2019.

[45] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *European Conference on Computer Vision, 2010. Proceedings. ECCV '10.*, 2000.

[46] S. Shankar, Vihari. Piratla, S. Chakrabarti, and Sarawagi S Jyothi, P. Generalizing across domains via cross-gradient training. *International Conference on Learning Representations*, 2018.

[47] R. Shu, H. Bui, Narui H., and S. Ermon. A dirt-t approach to unsupervised domain adaptation. *Workshop on Challenges in Representation Learning, ICML*, 3, 2013.

[48] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Scholkopf. Wasserstein auto-encoders. *International Conference on Learning Representations*, 2021.

[49] A. Torralba and A. Efros. Unbiased look at dataset bias. *International Conference on Computer Vision and Pattern Recognition.*, 2011.

[50] V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 1971.

[51] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, Courville A., D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. *International Conference on Machine Learning.*, 2019.

[52] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Duchi, and S. Savarese. Unified deep supervised domain adaptation and generalization. *Neural Information Processing Systems (NIPS)*, 2018.

[53] H. Wang, Z. He, Z. Lipton, and E. Xing. Learning robust representations by projecting superficial statistics out. *International Conference on Learning Representations*, 2019.

[54] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io/lil-log*, 2018.

[55] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang. Adversarial domain adaptation with domain mixup. *Association for the Advancement of Artificial Intelligence*, 2020.

[56] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[57] L. Yang, N. M. Cheung, J. Li, and J. Fang. Deep clustering by gaussian mixture variational autoencoders with graph embedding. *International Conference on Computer Vision (ICCV)*, 2019.

[58] S. Yun, D. Han, S. Oh, S. Chun, J Choe, and Y. Yoo. *International Conference on Computer Vision.*

[59] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations.*, 2018.