Learning Disentangled Features for Domain Generalization

Linfeng Du[*]
Department of Computer Science
University of Toronto
`linfeng.du@mail.utoronto.ca`

Kejia Yin[*]
Department of Computer Science
University of Toronto
`kejia.yin@mail.utoronto.ca`

Huakun Shen[*]
Department of Computer Science
University of Toronto
`huakun.shen@mail.utoronto.ca`

Abstract

Despite their outstanding performance on i.i.d. training and testing set, existing machine learning models may suffer undesired performance drop on out-of-domain data. In this paper, we propose novel approaches addressing this challenge by disentangling the latent features to be domain-specific and domain-invariant by swapping. Experiments on the Digits-DG dataset demonstrate our methods' superiority over all the baseline methods, including non-neural network methods. This work has practical applications in real world situations where data from unseen domains is commonly encountered. Our code is available at `https://github.com/CapFreddy/CSC2515-Final-Project`

1 Introduction

Over the past few years, Deep Neural Networks (DNNs) achieved extraordinary results on various tasks. Their success always rely on the i.i.d. (independent and identically distributed) assumption, which requires training and testing data to be drawn from the same distribution. However, such assumption can be easily violated when dealing with new domain data, and severe performance degradation have been observed in many studies [1], which is a critical issue to apply DNNs in real-world scenarios. Suppose we have an autonomous driving system which is trained with data in sunny days, would it be acceptable if our system cannot drive safely when weather is foggy or snowy? To tackle this problem, researches have been done in Domain Generalization (DG) [2].

In DG tasks, we divide our data to source domains and target domains. During training, we can only use the data in source domains and leave target domains unseen. This is a more challenging setting compared with Domain Adaptation, which can leverage unlabeled target domain data as part of the training set [3]. Such restrict setting helps us better evaluate how our trained model will behave when dealing with unseen domains in real-world environment.

[*]Equal contribution. Listing order is random. Linfeng raise the general idea of disentangling via swapping latent variables in the VAE framework. After consultation with the TAs, we decided to limit our scope to more simple network architectures. Kejia formalized our idea into our three proposed methods and implemented the initial version of the models. Linfeng refined based on which for faster experimentation. Huakun conducted thorough experimentation for the baselines. Every team member participated in the report writing and correction, as well as experimentation w.r.t. our proposed methods.

Since the key of DG is to learn a robust model against all possible domain shifts, we should let our model only pay attention to domain-invariant features and make prediction based on them. One may suggest an intuitive solution to DG by enlarging the source domains. For example, not only use sunny images, but also collect foggy and snowy images to train our autonomous driving system. This is effective, because it will be easier for the model to learn such generalizable features, but not very practical. On the one hand, it's very costly to collect new data; on the other hand, we can never collect all the data and enlarge the source data to cover all possible domains. Thus, this cannot be a general approach to our problem.

In this paper, we proposed 3 methods focusing on disentangling the latent domain-specific and domain-invariant features within existing source domains. Rather than implicitly make the model focusing on domain-invariant features with enlarged source data, we explicitly explore the original source domains and try to learn such features directly. We evaluated our proposed methods with the simplest DNNs architecture, namely multi-layer perceptrons, and compared our methods with different baseline machine learning models. Experiments demonstrate the effectiveness of our methods.

2 Related Work

Domain Generalization (DG) aims to learn a model with enough ability to generalize to arbitrary unseen target domains from existing source domains. In this section, we'll first discuss the related work using disentangled representations which are closely related with our proposed methods. We will also briefly review other related works which can be roughly categorized to methods based on domain alignment and methods based on data augmentation.

Disentangle Representations: The idea of disentangling representations is to separate features into different components and relies on the domain-invariant parts to help improve model's generalization ability. One way to do this is to use generative models. Ilse *et al.* utilized VAE to formulate domain, class and other feature respectively[4]. Nam *et al.* extracted content and style information separately and applied AdaIN to perform both style randomization and content randomization between training data[5]. The style randomization encouraged the classifier to predict the class based on content information, while adversarial training is applied with content randomization to force style information to be class-agnostic.

The main difference between our approach and previous work is that we use a simple swap operation to separate our desired features, while existing methods always require reconstruction loss or adversarial training.

Domain Alignment: The motivation of domain alignment is straight forward - since we need to improve our model's generalization ability, why not let our model learn from the invariant feature among source domains? In order to obtain such domain-invariant features, methods based on domain alignment try to minimize the distance between source domains with different metrics. Wang *et al.* proposed to align domain-agnostic posteriors within each class via the KL divergence[6]. Li *et al.* adopt autoencoder architecture and minimized Maximum Mean Discrepancy distance on latent features between source domains[7]. Instead of explicitly minimizing distance metrics, one can also use adversarial training to make latent features to be indistinguishable among all source domains[3]. Though this is originally proposed to deal with Domain Adaptation tasks, we adopt this method in our experiment setting and reported its results for comparison.

Data Augmentation: As a general approach to reduce over-fitting, data augmentation has been widely used in DG tasks. One can simply apply some traditional image transformation algorithms, like rotation, random flip and color changing, to improve model's generalization ability[8]. Inspired by adversarial attack, Shiv Shankar *et al.* perturbed the training sample with the gradient from a domain classifier in order to make its domain being different while preserving its class label[9]. Other methods try to augment source domain data with different styles based on the observation that domain shifts always results in different styles of image rather than content. Xu *et al.* proposed to augment source domain data in Fourier domain by swapping or mixing up amplitude between images as the amplitude contains rich style information[10].

Note that one of our proposed methods swaps domain features in latent space and can be considered as implicit data augmentation.

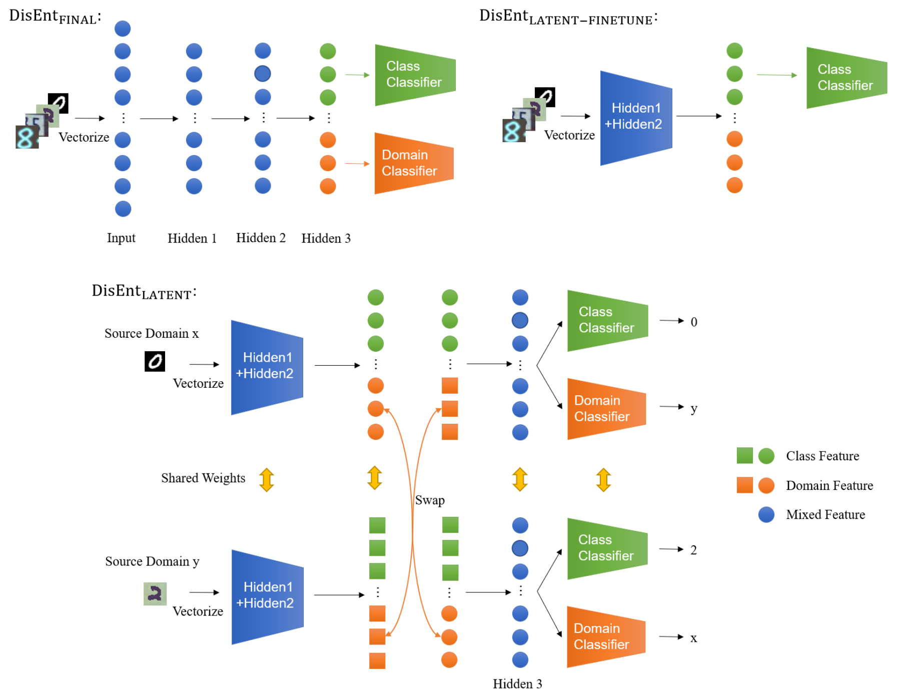


Figure 1: Overview of our proposed methods. The upper left, bottom, and upper right part illustrates DisEnt$_{\text{FINAL}}$, DisEnt$_{\text{LATENT}}$, and DisEnt$_{\text{LATENT-FINETUNE}}$ respectively. The items with green color denote the domain-invariant part, and the ones with orange color denote the domain-specific part. In DisEnt$_{\text{FINAL}}$, we simply use two classifiers which takes different part of the last latent feature as input. In DisEnt$_{\text{LATENT}}$, we swap the latent feature between training samples to better disentangle feature representations. In DisEnt$_{\text{LATENT-FINETUNE}}$, we build another class classifier based on pre-trained DisEnt$_{\text{LATENT}}$ model.

3 Method

Problem Definition: Given a set of source domains $\mathcal{D}_s = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_S\}$ with the k-th domain consists of N_K labeled data $\{x_i^k, y_i^k, d_i^k\}_{i=1}^{N_K}$, where x_i^k, y_i^k and d_i^k denote input data, class label and domain label respectively. The goal of DG is to learn a model f_θ on the source domains which can generalize well to arbitrary unseen target domain $\mathcal{D}_t$.

Model Architecture: In this work, we adopt a 3-hidden-layer fully-connected neural network. If we ignore the domain features in the network, the number of hidden units in each layer are fixed to be 512, 128, and 64 respectively. With this backbone network, we can perform fair comparison between each proposed methods and search for best hyperparameter combination.

In the following subsections, we will illustrate how our methods work and why we use them.

3.1 DisEnt$_{\text{FINAL}}$: Disentangling the Final Representations

The motivation of our work is to disentangle latent representations into domain-specific and domain-invariant features. Suppose we can somehow separate these features with our neural network, it's obvious that we can make good prediction of class label only based on domain-invariant features and domain label based on domain-specific features. Following this idea, we formulate our first method DisEnt$_{\text{FINAL}}$ as shown in the upper left in Fig. 1.

To be more specific, we use two classifiers here with different inputs. The class classifier only takes domain-invariant features as input and its dimension is fixed to be 64, while the domain-classifier only takes domain-specific features as input and its dimension is a hyperparameter. The loss function is formulated as standard cross-entropy:

$$L_{cls} = -y_i^k \log(\sigma(f_{\theta,cls}(x_i^k))), \tag{1}$$
$$L_{dom} = -d_i^k \log(\sigma(f_{\theta,dom}(x_i^k))), \tag{2}$$
$$L = L_{cls} + \alpha L_{dom}, \tag{3}$$

Where σ denotes softmax function and α is hyperparameter. This loss function can separate the latent representation to become class-related and domain-related, but we cannot guarantee the class-related feature to be domain-invariant. Though this method doesn't fully fulfill our motivation, it's still worth evaluating it as we can think the domain classifier here to be a regularizer and may help our model to generalize to other unseen domains.

3.2 DisEnt$_{\text{LATENT}}$: Disentangling Latent Representations via Feature Swapping

To better disentangle domain-specific and domain-invariant features, we need to add more constraints and this leads to our second method. As illustrated at the bottom of Fig. 1, we propose to implement such constraint by swapping part of the latent features between training data.

To be more specific, suppose we have two training samples denoted by $\{x_1, y_1, d_1\}$ and $\{x_2, y_2, d_2\}$. By passing x_1, x_2 through the first and second hidden layer, we can get the corresponding latent representations f_1, f_2. We can now generate two other representations $f_1^{'}, f_2^{'}$ by swapping part of the features between f_1 and f_2. Both generated features and original features are then passed into the third hidden layer and two classifiers. As for the generated features, we also swap the ground truth domain label of them, which means we need to eventually predict $\{y_1, d_2\}$ from $f_1^{'}$ and $\{y_2, d_1\}$ from $f_2^{'}$. We use the same loss function with DisEnt$_{\text{FINAL}}$ as defined in Eq. 3.

After training, we believe the swapped part of the feature will become domain-specific and the unswapped part will be domain-invariant. To better understand why this will work, let's think about the following situation: Suppose the swapped part still contains class-related information and the unswapped part contains domain-related information, then we know $f_1^{'}$ contains information about both y_1 and y_2, d_1 and d_2. In this case, it will be a hard problem for our model to predict only $\{y_1, d_2\}$ from $f_1^{'}$. In order to make better prediction, our model will learn to reduce the information related to y_2 and d_1. Eventually, the swapped part can only contain very less class-related information and the unswapped part can only contain very less domain-related information.

Note that it's also very important to pass the swapped feature through another hidden layer to mix the disentangled features. If we don't do so, there could be a shortcut for the two classifiers to make prediction only based on unswapped and swapped part respectively. In this case, DisEnt$_{\text{LATENT}}$ will become similar to DisEnt$_{\text{FINAL}}$, and we cannot force our model to separate the features.

3.3 DisEnt$_{\text{LATENT-FINETUNE}}$: Finetuning Disentangled Latents

As demonstrated in the former section, we can obtain domain-specific and domain-invariant features with DisEnt$_{\text{LATENT}}$. However, the class classifier in DisEnt$_{\text{LATENT}}$ doesn't make good use of this fact and makes prediction based on the mixed features. Therefore, we propose our third method DisEnt$_{\text{LATENT-FINETUNE}}$.

As shown in the upper right in Fig. 1, we build another class classifier which only takes domain-invariant features as input. In order to obtain such feature, we use the pre-trained DisEnt$_{\text{LATENT}}$ model. This method best aligns with our motivation, as we successfully disentangle the latent features and build the classifier only based on domain-invariant ones. In this method, it doesn't involve any training with domain labels and the loss function is just the standard cross-entropy defined in Eq. 1.

4 Experiments

4.1 Dataset and Evaluation Metrics

We evaluate our proposed methods on Digits-DG [11], a collection of four digit datasets including MNIST [12], MNIST-M [13], SVHN [14] and SYN [13]. Each of the four datasets is treated as a

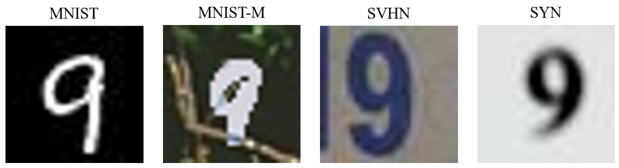


Figure 2: Example images from Digits-DG.

separate domain (see Fig. 2) which contains 600 images for each class. We use the same data splits as suggested by [11] where each dataset is split into training and validation sets by a ratio of 4:1. At each time, we combine the training and validation sets of three domains for training and model selection, respectively, and leave one domain out as the target domain for testing, where we test on all examples in the two splits. We report the classification accuracy on each target domain as well as the one averaged across four domains. For all of our methods as well as baselines, we train them using three different seeds for model initialization and data-shuffling, and report the mean and standard deviation of the independent runs.

4.2 Baseline Methods

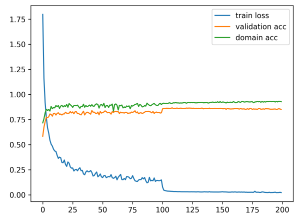
We compare our proposed methods with three groups of baselines. The first group consists of non-neural methods including Support Vector Machine (SVM), Random Forest (RF) and AdaBoost, which are representative in the family of kernel machines, bagging, and boosting, respectively. To the best of our knowledge, these methods were never compared to in existing literature of domain generalization. We fill this gap by benchmarking their performance under this typical DG setting. To this end, we conduct thorough hyperparameter tuning for each of them by grid-searching over a large hyperparameter space. The second group is the MLP with three hidden layers which is used as the backbone of our proposed methods. For a fair comparison, we train this vanilla MLP under the same hyperparameter settings except that no DG-targeting techniques is applied. For the third group, we adapt the domain-adversarial training framework [3] (DANN) to our backbone architecture, which applies a domain classifier to the full features produced by the MLP. The domain classification head generates reversed gradients to update the feature extractor in order to remove domain-relevant information from the extracted features. While our methods share the same ultimate goal of classifying based on domain-removed features, our approach is from a different perspective.

Since we limit our scope to this rather simple fully-connected network, we do not compare with methods that rely on more sophisticated network architectures as well as complex training schemes.

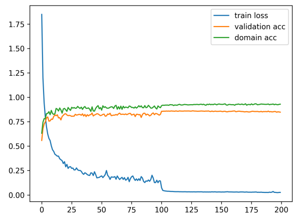
4.3 Implementations and Hyperparameter Settings

Images are resized to 32×32 and flattened to one-dimensional. The backbone feature extractor is a MLP with three hidden layers of size 512, 128 and 64. We use single fully-connected layers for both object and domain classification. For the three of our methods, we increase the units in the layer where disentanglement happens and treat the activation of the added units as domain features, while the activation of the original units are treated as object features. For method 2 and 3, domain feature swapping is implemented batch-wise by randomly permutating domain features across the batch and concatenating them with the original object features. The networks are trained from scratch using Adam [15] with an initial learning rate of 0.01, batch size of 256 and weight decay of 1e-3 for 200 epochs. The learning rate is decreased by 0.1 for every 50 steps. For the fine-tuning phase of method 3, we apply a small learning rate of 1e-5 to update the feature extractor, which performs slightly better than freezing the pretrained feature extractor. We adopt an early stopping scheme that the checkpoint with the best validation accuracy is used for testing.

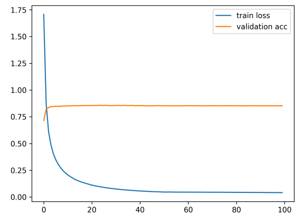
The only two hyperparameters that we search for are the dimension of domain features and the α in Eq. 3 that trades off the two losses. To limit our search space, we first fix α to be 0.5 and search for the optimum domain dimensionality. Then we fix the domain dimensionality and search for the best α. All of three methods reach the best performance when domain dimensionality is 64 and α is 0.1. As shown in Fig. 3, all methods converge well under this hyperparameter setting.



(a) Method 1



(b) Method 2



(c) Method 3 - Finetune

Figure 3: Training curves of the proposed methods (MNIST-M as target domain).

Method	MINST	MINIST_M	SVHN	SYN	Avg.
AdaBoost	67.10 ± 1.2	25.09 ± .3	22.81 ± 1.1	34.17 ± 1.9	37.29
RF	79.06 ± .7	25.74 ± .3	32.08 ± .5	42.09 ± .3	44.74
SVM	73.92 ± .0	29.90 ± .0	32.17 ± .0	55.37 ± .0	47.84
Vanilla MLP	89.58 ± .6	52.62 ± .5	52.71 ± .4	68.36 ± .7	65.81
DANN	88.79 ± .3	51.33 ± .5	52.07 ± .5	**69.61** ± .6	65.45
DisEnt$_{\text{FINAL}}$	90.12 ± .3	53.23 ± .3	**53.80** ± .1	68.32 ± 1.1	66.37
DisEnt$_{\text{LATENT}}$	**90.62** ± .4	53.67 ± .9	52.74 ± .7	69.06 ± .6	66.52
DisEnt$_{\text{LATENT-FINETUNE}}$	89.73 ± .4	**53.73** ± .1	53.50 ± .3	69.35 ± .2	**66.58**

Table 1: Classification Accuracy on Digits-DG

4.4 Main Results

Experiment results are shown in Table 1. We identified that a large gap exists between neural and non-neural methods. We find that the "traditional" method suffer severe underfitting (SVM) or overfitting (RF and AdaBoost) which may be due to their limited model capacity. It is unexpected that DANN performs worse than vanilla MLP. We suppose the reasons are two-fold: First, the DANN was proposed to deal with DA tasks and may not work well under DG settings. Second, due the adversarial training, the results with different hyperparameters are quite unstable and we may not have found the best hyperparameters. However, both of these reasons suggest that applying DANN to DG problem can be difficult.

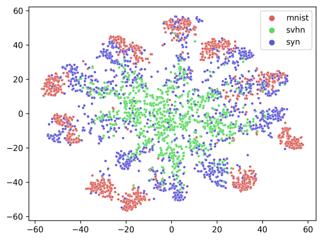
All of our proposed methods perform better than baseline methods. DisEnt$_{\text{FINAL}}$ has the least constraint separating domain-specific and domain-invariant features and has the lowest classification accuracy within three proposed methods. DisEnt$_{\text{LATENT}}$ performs better than DisEnt$_{\text{FINAL}}$ as we force the network to disentangle the latent representations by swapping them. In addition, DisEnt$_{\text{LATENT-FINETUNE}}$ achieves even better results while only rely on domain-invariant features to make predictions. This improvement demonstrates the effectiveness of our methods.

4.5 Visualization of Learned Features

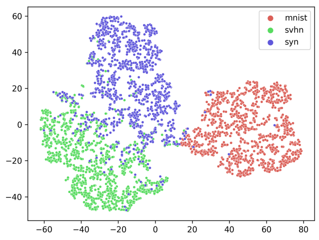
To better understand the effectiveness of our proposed methods, we visualize the disentangled representations learned by DisEnt$_{\text{FINAL}}$ and DisEnt$_{\text{LATENT}}$ via t-SNE [16]. As shown in Fig. 4, in terms of the domain label, it can be identified that some clusters of object representations are mixed together while domain representations are well-separated, which indicates that our methods are able to separate domain-relevant features, whether by directly disentangling the final layer or by swapping latent features.

5 Conclusions and Future Work

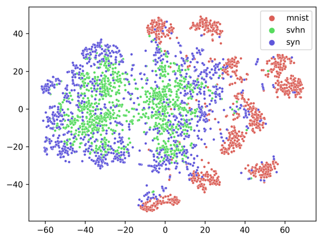
In this paper, we propose three methods to disentangle object and domain features for better domain generalization. Disentanglement is achieved via directly learning separate object and domain representations via different classification heads, as well as enforcing latent disentanglement via feature



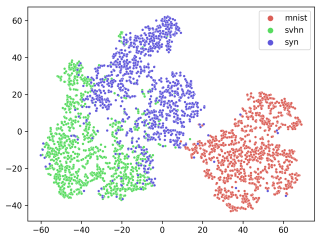
(a) DisEnt$_{\text{FINAL}}$ - Object



(b) DisEnt$_{\text{FINAL}}$ - Domain



(c) DisEnt$_{\text{LATENT}}$ - Object



(d) DisEnt$_{\text{LATENT}}$ - Domain

Figure 4: t-SNE visualization of learned features (MNIST-M as target domain).

swapping. We demonstrate the effectiveness of our proposed methods via quantitative analysis based on the Digits-DG dataset and showcase the disentangle effect via visualization techniques.

Although mixed clusters can be identified from Fig. 4, we still find the disentanglement somehow unsatisfactory since there still exists some clear boundaries between different domains. This could be due to the lack of domain-invariant constraints as in domain adversarial training [3]. Future work could look at incorporating such constraints into the framework as well as examining the performance based on more complex neural architectures with certain domain-specific priors.

References

[1] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[2] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

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

[4] Maximilian Ilse, Jakub M Tomczak, Christos Louizos, and Max Welling. Diva: Domain invariant variational autoencoders. In *Medical Imaging with Deep Learning*, pages 322–348. PMLR, 2020.

[5] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8690–8699, 2021.

[6] Ziqi Wang, Marco Loog, and Jan van Gemert. Respecting domain relations: Hypothesis invariance for domain generalization. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9756–9763. IEEE, 2021.

[7] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018.

[8] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[9] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[10] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yanfeng Wang, and Qi Tian. A fourier-based framework for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14383–14392, 2021.

[11] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13025–13032, 2020.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[14] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

[16] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL `http://jmlr.org/papers/v9/vandermaaten08a.html`.