Embracing the Dark Knowledge: Domain Generalization Using Regularized Knowledge Distillation

Yufei Wang
Rapid-Rich Object Search Lab, Nanyang Technological University
im.wangyufei@gmail.com

Haoliang Li
Department of Electrical Engineering, City University of Hong Kong
haoliang.li@cityu.edu.hk

Lap-pui Chau
Rapid-Rich Object Search Lab, Nanyang Technological University
elpchau@ntu.edu.sg

Alex C. Kot
Rapid-Rich Object Search Lab, Nanyang Technological University
eackot@ntu.edu.sg

ABSTRACT
Though convolutional neural networks are widely used in different tasks, lack of generalization capability in the absence of sufficient and representative data is one of the challenges that hinders their practical application. In this paper, we propose a simple, effective, and plug-and-play training strategy named Knowledge Distillation for Domain Generalization (KDDG) which is built upon a knowledge distillation framework with the gradient filter as a novel regularization term. We find that both the “richer dark knowledge” from the teacher network, as well as the gradient filter we proposed, can reduce the difficulty of learning the mapping which further improves the generalization ability of the model. We also conduct experiments extensively to show that our framework can significantly improve the generalization capability of deep neural networks in different tasks including image classification, segmentation, reinforcement learning by comparing our method with existing state-of-the-art domain generalization techniques. Last but not the least, we propose to adopt two metrics to analyze our proposed method in order to better understand how our proposed method benefits the generalization capability of deep neural networks.

CCS CONCEPTS
• Computing methodologies → Computer vision.

KEYWORDS
domain generalization, knowledge distillation, robustness

1 INTRODUCTION
Recently, convolution neural networks are widely used in different tasks and scenarios thanks to the development of machine learning and computing devices. However, one of the challenges that hinders the practical application of the convolution neural network model is the lack of generalization capability in the absence of sufficient and representative data [20, 59]. To overcome such limitation, domain adaptation (DA) have been widely studied under the assumption that a small amount of labeled target domain data or unlabeled target domain data can be obtained during the training stage.

However, in some cases, the target domain data may not always be available. As such, directly training on the source domain can lead to an overfitting problem. In other words, we can observe a significant performance drop when testing on the target domain. Domain generalization (DG) is one research direction that aims to utilize multi-source domains to train a model that is expected to be generalized better on the unseen target domains. The core idea of existing methods are either to align the features of source domains to a pre-defined simple distribution, to learn a domain invariant feature distribution [25, 35, 56], or to conduct data augmentation (e.g., data enhancement [53, 61], generative adversarial network (GAN) based methods [52, 64]) on source domains. In addition, there are also some efforts using the meta-learning based method to simulate the domain gap between the source and unseen target domain. However, the aforementioned techniques inevitably face the problem of overfitting on the source domain due to the inability to access the target domain data. It is still an open problem on how to learn a universal feature representation through the aforementioned techniques that the latent features of target lie on a similar
weak to predict the randomly initialized labels which are inconsistent with the inductive preference of the model, i.e., the recognition task becomes difficult when increasing \( \lambda \).

feature distribution compared with the latent features of source domains. As such, the generalization performance in the target domain may not be guaranteed due to the distribution mismatch. Our motivation comes from the observation that if the task is difficult for the model, i.e., the mapping from the input space to the label space is difficult to learn, the performance of the model in the target domain may have a large drop owing to the overfitting in source domains even if the performance on the source domains is still acceptable. One illustrative example to verify our observation can be seen in Fig. 2. By involving more noisy labels on the source domain, the learning task of the model also becomes difficult \([37, 59]\) and we find it further deteriorates the performance of the model on the target domain. To this end, we make the assumption that the generalization performance of the model in the unseen target domain can benefit from the easy tasks. In particular, we proposed a framework named knowledge distillation for domain generalization (KDDG) with a novel gradient regularization to decrease the difficulty of the task. The motivation is mainly in two folds. 1) we leverage the advantage of knowledge distillation to provide the student with more informative labels which make the training easier \([16, 43]\), where the rationales are illustrated in Fig. 1. We also provide a preliminary study in Fig. 3 using class activation map (CAM) \([63]\). We observe that our proposed method can help the student network to learn more general and comprehensive features, e.g., the legs and the body of the dog which are ignored by directly training the model with one-hot ground truth labels. 2) the gradient filter we proposed relaxes the objective of IRM that further makes the task easier (which is analyzed in Sec. 3.4 in details) and avoid the student model from completely imitating the teacher \([43]\), such that better generalization can be further guaranteed. We show that our proposed method can outperform existing state-of-the-art domain generalization techniques on different tasks including image classification, segmentation, reinforcement learning. We further propose to adopt two different metrics, namely cumulative weight distance and mutual information, to explain the effectiveness of our proposed method. The main idea is to measure the difficulty of the task and quantify the domain-specific information. We show that our proposed method KDDG can make the task easier to train and allow the student network to learn fewer domain-specific features, which further justify our assumption to tackle the problem of domain generalization through knowledge distillation. In summary, we make the following contributions,

1. We propose to tackle the problem of domain generalization from a perspective of task difficulty and reveal that better generalization capability can be achieved if the learning task is easier.
2. We propose a simple, effective, and plug-and-play training strategy named KDDG. Experimental results on different tasks demonstrate the effectiveness of our proposed method by comparing it with the SOTA techniques.
3. Two metrics are proposed to better understand how our proposed method benefits the generalization capability of neural networks.

2 RELATED WORK

2.1 Domain Adaptation and Generalization

How to alleviate the performance degradation caused by the domain gap between the training and test data has always been a hot research topic. One direction is to rely on unlabeled test data (a.k.a., target domain data) and apply domain adaptation (DA) or transfer learning \([18, 26, 40]\), which can be roughly divided into two categories, namely subspace learning based methods and instance re-weighting \([12, 18, 39, 60]\). In addition, deep learning based methods also demonstrate their effectiveness by feature alignment using different methods such as Maximum Mean Discrepancy (MMD) \([30]\) and adversarial based training \([5, 11, 17, 51]\).

The setting of domain generalization (DG) is similar but more challenging than the setting of domain adaptation because we cannot get any information from the target domain in the training phase. Part of the current DG methods borrow the idea from DA, e.g., domain invariant feature and feature alignment. For example, for domain invariant feature-based methods, \([56]\) aimed to use Canonical Correlation Analysis (CCA) to obtain the shareable information, \([35]\) proposed a domain invariant analysis method which also used MMD and was further extended by \([25, 27]\). Multi-task auto-encoder was also used to learn shareable feature representations \([13]\). In addition, different regularization methods were proposed, e.g., low-rank regularization was used in \([22, 55]\) to extract the invariant embedding, the puzzle task \([8]\) was used to learn a model with good generalization performance. Meta-based methods \([3, 23]\) can be also treated as a kind of regularization, e.g., \([23]\) used the second derivative as a regular term and \([3]\) directly learned a regularization network. However, we may still not be able to guarantee that the embedding from target domains share the same subspace with that of the source domains through the aforementioned domain invariant feature learning based methods since the learned classifier can be overfitted to the source domains. In addition, the regularization may make the learning task difficult which further increases the risk of overfitting in the source domains. There also exists some works using data augmentation based techniques \([52, 53, 64]\). For example, \([64]\) used an adversarial generative model.
To the best of our knowledge, knowledge distillation (KD) was first introduced by Hinton et al. [35]. In the context of domain generalization where the testing data are collected from a different distribution compared with the training data, knowledge distillation is also known as policy distillation, which can be applied to generate some samples that are out of the source domains to improve the generalization capability of the neural network. However, if the distribution of the newly generated data is not similar to that of the target domain, it may have a negative impact on the model. In addition, it may be difficult to generate samples out of the convex set of the source domains [4].

2.2 Knowledge Distillation

To the best of our knowledge, knowledge distillation (KD) was first designed for model compression [7, 48], which aims to make the output of the new model (student) similar to that of the previous model (teacher model) meanwhile decrease the size of the student one. In [16, 33], it has been shown that training the student model with “dark knowledge” by KD can lead to better performance compared with directly training with one-hot ground truth. Besides, it has been also shown that KD can benefit nature language processing [29, 62]. In the area of reinforcement learning, knowledge distillation is also known as policy distillation, which can be applied for model compression, network training acceleration, or multiple agent models merging [41, 47, 57]. Comparing with the application of KD, the theoretical analysis [9, 31, 36, 43] is relatively deficient and mainly focus on the characteristics such as the convergence [43]. While there exist some works linking KD with the generalization capability of neural network (e.g., [36, 43]), they mainly focus on analyzing the generalization capability of the model using the same dataset or data distribution, e.g., the upper bound of the transfer risk [43]. Different from the previous work, we explore and analyze KD from a new perspective by focusing on the setting of domain generalization where the testing data are collected from a different distribution compared with the training data.

3 METHODOLOGY

3.1 Knowledge Distillation

Before introducing our proposed method, we first revisit the idea of knowledge distillation [16]. Specifically, in order to obtain the soft label, temperature \( \tau \) is introduced as a hyper-parameters to soften the vanilla softmax distribution, which is given as

\[
p^i(x; \tau) = \frac{e^{s_i(x)/\tau}}{\sum_k e^{s_k(x)/\tau}} \tag{1}
\]

where \( s_i(x) \) is the score logit from the sample \( x \) of class \( i \). Subsequently, the KL-divergence is used as the distillation loss \( L_{kd} \) to measure the difference between the teacher and student:

\[
L_{kd} = -\tau^2 \sum_{x \in D_s} \sum_{i=1}^C p_i^t(x; \tau) \log(p_i^s(x; \tau)) \tag{2}
\]

where \( p_t \) and \( p_s \) denote the softened probability from the teacher and student network respectively, \( C \) is the total number of the categories, \( D_s \) indicates the distribution of all source domain data by concatenating them together. Here, the coefficient \( \tau^2 \) is used to balance the magnitude of the gradient [16] between hard and soft label. The loss function for the student network is as follows:

\[
L_{vanilla} = \lambda_1 L_{ce} + \lambda_2 L_{kd} \tag{3}
\]

where \( L_{ce} \) is the standard cross-entropy, \( \lambda_1 \) and \( \lambda_2 \) are balancing weight.

3.2 Gradient Filter

We further propose a novel gradient regularization to improve the generalization capability of the network. Without loss of generality, we use the classification task to illustrate the gradient filter \( f \). Assuming that \( p(x_i) \) denote the softmax output vector of the student network from the sample \( x_i, p^k(x_i) \) represents the probability value of the sample \( x_i \) with respect to its ground truth category \( k \) (abbreviated as \( p \)). We can define the gradient filter \( f \) as

\[
f(\omega) = \begin{cases} \frac{\omega}{1 - \eta} & 0 < \frac{p - \eta}{1 - \eta} \leq \eta \\ \frac{\omega}{1 - \eta} \left( \frac{p - \eta}{1 - \eta} \right)^2 & \eta < \frac{p - \eta}{1 - \eta} \leq 1 + \eta \\ \frac{\omega}{1 - \eta} \left( \frac{p - \eta}{1 - \eta} \right)^3 & \frac{p - \eta}{1 - \eta} > 1 + \eta \end{cases} \tag{4}
\]

where \( \omega \) denotes gradient and \( \eta \) is a hyper-parameter that controls the intensity of the filter. Noted that we can also apply the operation \( f \) on the empirical risk of a sample, as our proposed GradFilter can be treated as imposing weight on the loss function.

The significance of \( f \) comes from mainly two folds, 1) we can prevent the student network from being too similar to the teacher network [43], such that the risk of overfitting to teacher network can be reduced; 2) by filtering out the gradient corresponding to high score output, we can avoid the problem of over-confidence [14, 36] and make the optimization task easier by relaxing the objective of IRM which will be further discussed in Sec. 3.4.

3.3 Knowledge Distillation for DG

Now we introduce our proposed method built upon knowledge distillation with gradient filter regularization. Specifically, at each iteration, the gradient filter inspects the confidence of each sample and decrease the gradient weight of the sample which has a confidence score higher than a pre-defined threshold (by setting the corresponding loss to zero). Thus, the integrated loss for domain generalization can be formulated as

\[
L_{kd}^f = -\tau^2 \sum_{x \in D_s} \sum_{i=1}^C p_i^t(x; \tau) \log(p_i^s(x; \tau))) \tag{5}
\]
where $y^i$ denotes the ground truth of the sample $x$ for class $i$. $\epsilon$ is defined as follows to avoid the bad effect from the wrong predict from the teacher

$$
\epsilon(l) = \begin{cases} 
1 & v(p_t(x_i)) = y_i \\
0 & \text{otherwise}
\end{cases}
$$

(7)

where $p_t(x_i)$ and $p_t$ represent the probability value of the sample $x_i$ with the class index $k$ from the student network and probability vector from the teacher network respectively, $v$ is an operation that returns the predicted label according to the probability vector. The final objective for KDDG can be represented as

$$
L_{KDDG} = \lambda_1 L_{kd}^f + \lambda_2 L_{te}^f.
$$

(8)

For KD, we set $\tau$ to 2. We also set $\lambda_1$ and $\lambda_2$ to 0.5 so that we have roughly the same magnitude of the gradient for fair comparison and easy analysis.

3.4 Discussion

To further understand the rationale and effectiveness of gradient filter we proposed, we give a view from the perspective of invariant risk minimization (IRM) [1, 2], which is for generalization analysis of the classifier across multiple domains.

According to IRM, the task of DG can be treated as to find a representation $\Phi$ such that the optimal classifier given $\Phi$ is invariant across different domains. More specifically, for a feature extractor $\Phi$, there exists an invariant predictor $\hat{w}$ if this predictor achieves the best performance among all the domains. In [1, 2], the authors proposed to find the optimal $\Phi$ and $\hat{w}$ as

$$
\min_{\Phi \in \mathcal{H}, \hat{w} \in \mathcal{H}_w} \sum_{d \in D} L_d^d (\hat{w} \circ \Phi)
$$

(9)

where $L^d$ is the empirical risk in the domain $d$ and $D$ represents all the domains, and the solution space of $(\hat{w}, \Phi)$ can be defined as $S^{IV}$.

Unlike the previous works about IRM [1, 2], our GradFilter can be treated as to conduct solution space transformation on $S^{IV}$. By applying GradFilter operation, Eq. 9 can be reformulated as

$$
\min_{\Phi \in \mathcal{H}, \hat{w} \in \mathcal{H}_w} \sum_{d \in D} f(L^d(\hat{w} \circ \Phi))
$$

(10)

where $f$ is the GradFilter operation defined in Eq. 4 which avoid the model over-confidence in the source domains.

Our proposed GradFilter can be interpreted as to relax the objective of IRM. If directly optimizing Eq. 9, it may be hard to find a solution which satisfies both the objective and the regularization term, as there may not exist an intersection among the solution set of each source domain. By relaxing the objective, we make the optimization task easier, i.e., it is more likely that there exists an overlap region among solution space between source domains. Thus, our proposed method can be interpreted as an alternative solution for Eq. 9. More details can be found in the supplementary materials.

4 EXPERIMENTS

We evaluate our methods on different domain generalization benchmarks, including Digit-Five [42], PACS [22], mini-domainnet [42, 65], gray matter segmentation task [44] and mountain car [6]. Examples of each benchmark are shown in Fig. 4. For all the experiments, we use the same architecture for both teacher and student network. In addition, the gradient filter is only used when training the student network based on its output. Due to the limited space, more details about the experiments are in the supplementary material.

4.1 Digits-Five

Digits-Five is a benchmark used by [42], which is composed of MNIST-M, MNIST, SVHN, USPS and SYN with different backgrounds, fonts, styles, etc.

Settings: We follow the experiment settings in [42] except that we assume the target domain is unavailable during training. More specifically, we adopt a standard leave-one-domain-out manner. For the source domains, 80% of the samples are used for training and 20% for validation. All the samples in the target domain are used for testing. We follow [42] to convert all image samples with the resolution of $32 \times 32$ in RGB format. We use the same backbone network for all methods which include three convolutional blocks and two FC layers. Each convolutional block contains a $3 \times 3$ convolutional layer, a batch normalization layer, a ReLU layer, and a $2 \times 2$ max-pooling layer. The SGD is used with an initial learning rate of 0.05 and a weight decay of $5 \cdot 10^{-4}$ for 30 epochs. The cosine annealing scheduler is used to decrease the learning rate.

Results: We compare our methods with several competitive models including MLDG [23], JiGen [8], MASF [10] and RSC [19]. We report the baseline results in Table 1 by tuning the hyper-parameters in a wide range. As we can observe, our method achieves the best results which prove the effectiveness of our proposed method. In addition, using the knowledge distillation alone also has some improvement which further verifies our motivation.

| Method       | MM   | MNIST | USPS  | SVHN | SYN  | Avg. |
|--------------|------|-------|-------|------|------|------|
| DeepAll     | 66.3 | 96.7  | 94.1  | 82.1 | 88.9 | 85.6 |
| MLDG [23]   | 67.9 | 97.3  | 94.7  | 83.9 | 89.1 | 86.6 |
| JiGen [8]   | 67.8 | 97.8  | 95.9  | 84.7 | 89.6 | 87.2 |
| MASF [10]   | 69.2 | 98.6  | 96.3  | 84.4 | 90.1 | 87.7 |
| RSC [19]    | 69.8 | 98.3  | 96.1  | 84.1 | 89.9 | 87.6 |
| Distill     | 69.7 | 99.2  | 97.3  | 84.7 | 90.4 | 88.3 |
| KDDG(Ours)  | 70.5 | 99.1  | 97.6  | 85.5 | 90.6 | 88.7 |

Table 1: Evaluation of DG on the Digit-Five benchmark. The average target domain accuracy of five repeated experiments is reported. MM and SYN are abbreviations for MNIST-M and Synthetic Digits.

4.2 PACS

PACS [22] is a standard benchmark for domain generalization which includes 4 different domains: photo, sketch, cartoon, painting. There are 7 categories in the dataset and 9991 images in total.
We repeat the experiment for 5 times and report the average target lead to better performance.

on the same input between teacher and student network, which knowledge distillation in an one-one correspondence manner based train and meta-test domains. Specifically, we consider to conduct divergence of the class-specific mean feature vectors between meta-

that our method is different from MASF [10] which minimizes the are used for comparison, including CCSA [58], MLDG [23], Cross-

grayscale which are not available in the DG setting.

We do not use the augmentations, e.g., random gray scale, crop on images with a scale factor of 1.25 and random horizontal flip. We do not use the augmentations, e.g., random gray scale, considering that it can introduce the prior knowledge of specific domain information, e.g., the samples in the sketch domain are all grayscale which are not available in the DG setting.

Results: To compare the performance of different methods, we report the top 1 classification accuracy in the unseen target domain. We repeat the experiment for 5 times and report the average target domain accuracy at the last epoch. Several state of the art methods are used for comparison, including CCSA [58], MLDG [23], Cross-Grad [49], etc. The results are shown in Table 2. It is worth noting that our method is different from MASF [10] which minimizes the divergence of the class-specific mean feature vectors between meta-train and meta-test domains. Specifically, we consider to conduct knowledge distillation in an one-one correspondence manner based on the same input between teacher and student network, which may avoid negative transfer and we also empirically find that if can lead to better performance.

### 4.3 Mini-DomainNet

We then consider mini-DomainNet [65], which is a subset of DomainNet [42] for evaluation. It contains 4 domains, namely sketch, real, clipart and painting, with more than 140k images in total.

Settings: We use SGD with momentum as the optimizer. The initial learning rate is 5e-3 and is decreased using the cosine annealing rule [32]. The batch size is 128 with a random sampler from the concatenated source domains. Resnet-18 is used as the backbone network for all competitors and we train the model for 60 epochs. We use the same data augmentation for all the methods, including random flip and random crop with a scale factor of 1.25. We adjusted the hyperparameters for the competitors in a wide range and report the best results we can achieve here.

Results: We repeat the experiments for 3 times and report the average accuracy in Table 4. We can find that our method has an improvement in a clear margin compared with other methods in the mini-DomainNet benchmark which is much larger than PACS. Such observation further justifies the significance of our proposed method to handle large-scale data.

### 4.4 Gray Matter Segmentation

We further evaluate our proposed method on gray matter segmentation task [28, 44] which aims to segment the gray matter area of the spinal cord for medical diagnosis. We use the data from spinal cord gray matter segmentation challenge [44] which are collected from four different medical centers with different MRI scanner parameters, e.g., different resolution from $0.25 \times 0.25 \times 2.5mm$ to $0.5 \times 0.5 \times 5mm$, the flip angle from $7^\circ$ to $35^\circ$, different coil type. These differences lead to four different domains named `set1`, `set2`, `set3`, and `set4`.

Settings: We use the same 2D-Unet [26, 46] as the backbone for a fair comparison. A two-stage strategy in a coarse-to-fine manner is adopted following [28, 44]. More specifically, we first segment the area of the spinal cord and then extract the area of gray matter from the spinal cord.

Results: We compare our method with state of the art domain generalization methods, including MASF [10], MLDG [23], CCSA [58], LDDG [28], using different metrics including three overlapping metrics: Dice Similarity Coefficient (DSC $\uparrow$), Conformity Coefficient (CC $\uparrow$), Jaccard Index (JI $\uparrow$); one statistical based metrics: True Positive Rate (TPR $\uparrow$); one distance based metric based on 3D: Average surface distance (ASD $\downarrow$). $\uparrow$ means the higher the better, $\downarrow$ means the lower the better. The results are shown in Table 3 and Fig. 5.

| Method | Art | Cartoon | Photo | Sketch | Avg. |
|--------|-----|---------|-------|--------|------|
| DeepAll | 77.0 | 75.9 | 95.5 | 70.3 | 79.5 |
| CCSA [58] | 80.5 | 76.9 | 93.6 | 66.8 | 79.4 |
| MLDG [23] | 78.9 | 77.8 | 95.7 | 70.4 | 80.7 |
| CrossGrad [49] | 79.8 | 76.8 | **96.0** | 70.2 | 80.7 |
| JiGen [8] | 79.4 | 75.3 | **96.0** | 71.6 | 80.5 |
| MASF [10] | 80.3 | 77.2 | 93.9 | 71.7 | 81.0 |
| Epi-FCR [24] | **82.1** | 77.0 | 93.9 | **73.0** | 81.5 |
| RSC [19] | 79.5 | 77.8 | 95.6 | 72.1 | 81.2 |
| KDGG(Ours) | 81.0 | **78.7** | **96.0** | 72.3 | **82.0** |

Table 2: Evaluation of DG on the PACS benchmark. The average target domain accuracy of five repeated experiments is reported.

Figure 4: An illustration of domain generalization benchmarks used for evaluation. Each column represents a domain and the data are randomly sampled except the reinforcement learning task Mountain-Car where the domain gap is concentrated in the different gravitational constant.
According to the results, we can find that our method has the best performance overall comparing with the latest SOTA works.

4.5 Mountain Car

Besides evaluating on the benchmark datasets in computer vision, we are also interested in analyzing our proposed KDDG in the reinforcement learning task. To be specific, we use a standard reinforcement learning benchmark in OpenAI gym [6] named mountain car. The target is to let the mountain car hit the peak with the least fuel cost with the actions “push left”, “no push”, and “push right” in the action space.

**Settings:** We create different domains by change the gravitational constant $g$ in the game. More specifically, $g$ is set in a range from 0.0019 to 0.0031 with a step 0.0003, which leads to 5 different environments regarded as 5 domains. We use the setting of single domain generalization here in which only one domain is used for training, where $g = 0.0025$, and others are used as unseen target domains for the test. We use the DQNs as the baseline and use the same backbone for all methods. The details of our backbone can be found in the supplementary materials. We compare our method with MLDG [23] and Maximum Entropy Reinforcement Learning (MERL) based method which also encourages the output to be soft and increases the generalization ability.

**Results:** We repeat the experiments for 10 times and then report the average score of fuel consumption and the corresponding standard deviation in the Table 5. The smaller the value in the table, the better. It represents the amount of fuel consumption. We can find that both MLDG [23] and MERL [15] have some improvement even if we use a single domain for training. We can also observe that our method has a better performance in most of the cases and has a more stable performance than the baseline method.

4.6 Ablation Study

To further understand the contribution of each component, we conduct the ablation study by using the PACS benchmark. The results are shown in Table 6. We can find about 1.2% improvement from 79.5% to 80.7% in average when only using the gradient filter and 1.4% improvement when only using the vanilla knowledge distillation which is competitive to the results of some DG methods.
Embracing the Dark Knowledge: Domain Generalization Using Regularized Knowledge Distillation

Figure 5: Qualitative comparisons. Each row represents a sample from a specific domain. Each column denotes the input, ground truth (gt) or different methods including DeepAll, MLDG [23], Probabilistic U-Net [21] (abbreviated as PROB here), LDDG [28], CCSA [58], MASF [10] and KDDG, respectively.

Table 5: Single domain generalization results on the RL task mountain car. In the source domain, \( g \) is set to 0.0025. The amount of fuel consumption is used as the metric [6] for which the smaller the better.

| \( g \)   | Baseline | MLDG [23] | MERL [15] | KDDG   |
|----------|----------|-----------|-----------|--------|
| 0.0019   | 111.8±1.84 | 109.9±2.43 | 110.2±1.92 | 108.7±2.23 |
| 0.0022   | 109.2±1.28 | \textbf{106.8±1.49} | 108.1±1.97 | 107.9±1.54 |
| 0.0025   | 113.0±6.08  | 110.8±7.01  | 112.4±8.65  | 109.3±4.78  |
| 0.0028   | 174.6±22.9  | 121.4±21.7  | 127.9±18.3  | 117.0±20.2  |
| 0.0031   | 167.7±11.0  | 139.8±22.1  | 134.8±24.6  | 126.8±25.9  |

Table 6: Ablation study on the PACS benchmark. KD: using vanilla knowledge distillation in Eq.3. GradFilter: only use the gradient filter without a distillation manner. KDDG: our complete version.

5 FURTHER ANALYSIS

To further explore the rationale of the effectiveness of our proposed method, we propose two metrics to quantify the task difficulty and the amount of superfluous domain specific information respectively.

5.1 Difficulty Quantification

5.1.1 Metric to Quantify the Difficulty. As discussed previously, there exists a connection between the task difficulty and the generalization capability of the trained model. We have shown our preliminary experiments in Fig. 2 to explore the impact of task difficulty on generalization performance by setting different noisy label levels of the ground truth. We are also interested to quantify the task difficulty to analyze how task difficulty influences the generalization capability. While there exist some works to quantify the difficulty in computer vision tasks [38, 50], however, they mainly focus on assessing the difficulty of a single picture or of a certain area in the picture [38, 45], which may not be suitable in our setting. To this end, we propose to use cumulative weight distance (CWD) to quantify the difficulty of the task, where CWD can be defined as

\[
CWD = \sum_{k=1}^{N} \|w_k - w_{k-1}\|_2
\]  

where \( w_k \) is the parameters expanded into a vector at epoch \( k \) and \( w_0 \) denotes the initial parameters vector. \( N \) denotes the maximum number of epochs. We adopt the metric CWD to verify our assumption that the simpler the task is, the smaller CWD is.

5.1.2 Analysis of Results Based on the Cumulative Weight Distance. The results of CWD by varying noise level \( \lambda \) are shown in Fig. 6 using the same setting as Fig. 1. We can draw the conclusion that the simpler the task, the smaller CWD and the better generalization. To further explore the impact of our method KDDG on CWD, we also conduct experiments using different components of our method. The results are shown in Table 7. We can find that KD, GradFilter,
We propose to use the Monte Carlo method such that the mutual information between \( Z, Y \) can be computed by using the law of total probability \([66]\)

\[
f(z) = \sum_{i=1}^{N} f(z|x_i)p(x_i),
\]

where \( f(z|x) \) is the conditional distribution from the feature encoding sub-network of the original model, \( f(y|x) \) can be computed by using SVM with class probability \([54]\) with five cross validation in the source domains. We empirically find that it has faster speed, higher accuracy and better stability than convolutional neural networks based domain discriminators.

### 5.2.2 Analysis of Results Based on Mutual Information

Now we use the metric \( I(Z, Y_d) \) to quantify the amount of domain-specific information remaining in the network. The sketch domain in the PACS benchmark is used for evaluation on account of its good domain discrimination. We train the models for 26 epochs and show the amount of domain-specific information \( I(Z, Y_d) \) remaining in the network in Fig. 7. As we can see, domain specific information will quickly decrease at the beginning of the training phase. However, as the training progresses, the model will inevitably overfit on the training source domains. Besides, we can also draw a conclusion from Fig. 7 that the harder the task, the more domain specific information will be learned. Last, we show the results of \( I(Z, Y_d) \) in Table 8 by considering different component. We can see that by jointly considering KD and GradFilter, more domain-specific information can be removed.

![Figure 7: Trends in the amount of residual domain specific information comparing with DeepAll.](image)

Table 8: The amount of domain-specific information remaining in the network. KD: using vanilla knowledge distillation in Eq.3 from vanilla teacher.

|                | DeepAll | KD   | GradFilter | KDDG |
|----------------|---------|------|------------|------|
| \( I(Z, Y_d) \) | 0.91    | 0.82 | 0.86       | 0.73 |

To compute \( I(Z, Y_d) \), we also need to compute \( f(z) \), which is the distribution of the latent feature \( z \), as well as the likelihood of the domain discriminator \( f(y|z) \). \( f(z) \) can be computed by using the law of total probability \([66]\)

\[
f(z) = \sum_{i=1}^{N} f(z|x_i)p(x_i),
\]

where \( f(z|x) \) is the conditional distribution from the feature encoding sub-network of the original model, \( f(y|x) \) can be computed by using SVM with class probability \([54]\) with five cross validation in the source domains. We empirically find that it has faster speed, higher accuracy and better stability than convolutional neural networks based domain discriminators.

### Figure 6: The change of CWD with training using different noise level \( \lambda \). We use syn in digit-five as the target domain and train the model for 50 epochs. We can find that CWD and task difficulty are positively correlated, i.e., the easier the task, the smaller CWD.

![Figure 6: The change of CWD with training using different noise level \( \lambda \).](image)

and KDDG can reduce the CWD compared with directly training with the DeepAll baseline. Such observation further verifies our assumption. It’s worth noting that there is a lower bound of CWD to learn the domain invariant feature still, so here we emphasize the difference of CWD between baseline and our methods.

![Figure 7: Trends in the amount of residual domain specific information comparing with DeepAll.](image)

Table 7: The impact of different components on CWD.

| Component | DeepAll | KD   | GradFilter | KDDG |
|-----------|---------|------|------------|------|
| CWD       | 14.51   | 14.44| 14.30      | 14.28|

To compute \( I(Z, Y_d) \), we also need to compute \( f(z) \), which is the distribution of the latent feature \( z \), as well as the likelihood of the domain discriminator \( f(y|z) \). \( f(z) \) can be computed by using the law of total probability \([66]\)

\[
f(z) = \sum_{i=1}^{N} f(z|x_i)p(x_i),
\]

where \( f(z|x) \) is the conditional distribution from the feature encoding sub-network of the original model, \( f(y|x) \) can be computed by using SVM with class probability \([54]\) with five cross validation in the source domains. We empirically find that it has faster speed, higher accuracy and better stability than convolutional neural networks based domain discriminators.

### Table 8: The amount of domain-specific information remaining in the network. KD: using vanilla knowledge distillation in Eq.3 from vanilla teacher.

|                | DeepAll | KD   | GradFilter | KDDG |
|----------------|---------|------|------------|------|
| \( I(Z, Y_d) \) | 0.91    | 0.82 | 0.86       | 0.73 |

To compute \( I(Z, Y_d) \), we also need to compute \( f(z) \), which is the distribution of the latent feature \( z \), as well as the likelihood of the domain discriminator \( f(y|z) \). \( f(z) \) can be computed by using the law of total probability \([66]\)

\[
f(z) = \sum_{i=1}^{N} f(z|x_i)p(x_i),
\]

where \( f(z|x) \) is the conditional distribution from the feature encoding sub-network of the original model, \( f(y|x) \) can be computed by using SVM with class probability \([54]\) with five cross validation in the source domains. We empirically find that it has faster speed, higher accuracy and better stability than convolutional neural networks based domain discriminators.

### Table 7: The impact of different components on CWD.

| Component | DeepAll | KD   | GradFilter | KDDG |
|-----------|---------|------|------------|------|
| CWD       | 14.51   | 14.44| 14.30      | 14.28|

To compute \( I(Z, Y_d) \), we also need to compute \( f(z) \), which is the distribution of the latent feature \( z \), as well as the likelihood of the domain discriminator \( f(y|z) \). \( f(z) \) can be computed by using the law of total probability \([66]\)

\[
f(z) = \sum_{i=1}^{N} f(z|x_i)p(x_i),
\]

where \( f(z|x) \) is the conditional distribution from the feature encoding sub-network of the original model, \( f(y|x) \) can be computed by using SVM with class probability \([54]\) with five cross validation in the source domains. We empirically find that it has faster speed, higher accuracy and better stability than convolutional neural networks based domain discriminators.

### Table 8: The amount of domain-specific information remaining in the network. KD: using vanilla knowledge distillation in Eq.3 from vanilla teacher.

|                | DeepAll | KD   | GradFilter | KDDG |
|----------------|---------|------|------------|------|
| \( I(Z, Y_d) \) | 0.91    | 0.82 | 0.86       | 0.73 |
We address the domain generalization problem by proposing a simple, effective, and plug-and-play training strategy based on a knowledge distillation framework with a novel gradient regularization. We show that our method can achieve state-of-the-art performance on five different DG benchmarks including classification, segmentation, and reinforcement learning. We further analyze the significance and effectiveness of our method by proposing two metrics from the perspective of task difficulty and domain information.

REFERENCES

[1] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Bhuro Chandler. 2020. Invariant risk minimization games. arXiv preprint arXiv:2002.04692 (2020).
[2] Martin Arjovsky, Leon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. arXiv preprint arXiv:1907.02893 (2019).
[3] Yoshes Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. MetaReg: Towards Domain Generalization using Meta-Regularization. In NuerIPS. 988–1008.
[4] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. 2019. Understanding and improving interpolation in autoencoders via an adversarial regularizer. ICLR (2019).
[5] Konstantinos Boumal, Nathan Silberman, David Dohan, Dimitru Erhan, and Dilip Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In CVPR. Vol. 1. 7.
[6] Greg Brockman, Vicki Chen, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. arXiv preprint arXiv:1606.01540 (2016).
[7] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In SIGKDD.
[8] Fabio Maria Carlucci, Antonio D’Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. 2019. Domain Generalization by Solving Jigsaw Puzzles. arXiv preprint arXiv:1903.06864 (2019).
[9] Xi Cheng, Zhefan Rao, Yilan Chen, and Quanshi Zhang. 2020. Explaining domain generalization via invariant feature representation. In NuerIPS.
[10] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. 2018. MetaReg: Towards Domain Generalization using Regularized Knowledge Distillation.
hard detach
threshold

loss
x

Figure 8: An illustration of a cause of instability during training. Assuming that there is a sample that has a lower confidence score than the threshold $\eta$, if we use the hard detach, i.e., the loss of the sample that has a confidence score higher than the threshold is directly set to 0, after a small update to the model, the loss may rise instantly, e.g., samples that were originally under the threshold may suddenly reach the threshold, causing the abrupt changes of loss and instability of the training.

threshold $\eta$ we set which further verifies our observation in the ablation study that using GradFilter only can also improve the performance of the model by reducing the overfitting in the source domains.

Figure 9: Confidence histograms under different bin edges. We can find that most of the samples in the training set has a confidence higher than 0.999 after training using DeepAll.

A FURTHER DISCUSSION ON GRADFILTER

A.1 An Alternative Implementation of GradFilter

In this section, we explain the advantage of our proposed GradFilter with a smooth transition (shown in Eq. 15) by comparing it with a "hard detach" formulation baseline, which is shown in Eq. 16.

$$f(\omega) = \begin{cases} \omega & \eta \leq p < \frac{1+\eta}{2} \\ \frac{1-\eta}{1-\eta^2}2\omega & \eta < p \leq \frac{1+\eta}{2} \\ 0 & p \geq \frac{1+\eta}{2} \end{cases} \quad (15)$$

According to our empirical analysis, we find that a hard detach of the gradient may make the training unstable. We show one example which can cause the unstable training in Fig. 8. Comparing with the hard detach, the GradFilter we proposed avoids the abrupt changes of the loss and makes the training more stable.

A.2 The Choice of the Hyperparameter $\eta$

We set $\eta$ to 0.999 for all the experiments. At first glance, it may seem to be a relatively large value, but it is actually a relatively small value because of the miscalibration [14] of the neural network. A demonstration of the miscalibration is shown in Fig. 9 using PACS benchmark and select 'sketch' as the target domain. We can find that most of the samples have a confidence score higher than the benchmark and select 'sketch' as the target domain. We can find that most of the samples have a confidence score higher than the benchmark and select 'sketch' as the target domain.

**B ADDITIONAL EXPERIMENTS**

B.1 Time and Space Complexity Analysis

We now analyze the time and space complexity of our proposed KDDG by comparing the running efficiency during the training phase with meta-based method MLDG [23], ensemble learning based method Epi-Fcr [24], alignment based method MASF [10]. We measure the training iteration time of networks by using the same batch size and setting as the PACS benchmark on an NVIDIA GeForce RTX 2080 Ti GPU. The results are in Table 9. As we can see, the complexity of our proposed KDDG is similar to other baseline methods, as the cost of the inference stage without gradient calculation is very small. The complexity of our method is the same as DeepAll baseline during the inference stage.
Table 9: Time and space complexity analysis

|                | DeepAll | MLDG   | Epip-FCR | MASF   | KDDG (Ours) |
|----------------|---------|--------|----------|--------|-------------|
| s/iteration    | 0.0319  | 0.2824 | 0.5445   | 0.3683 | 0.0382      |
| memory used (MB)| 2115    | 3011   | 5095     | 2779   | 2227        |

B.2 More Visualization Results

In addition to the visualization results in the paper, we also show some extra visualization results in Fig. 10 using class activation map [63]. We can find that our method tends to have a larger area of the CAM which further verifies our assumption that our method can force the student network to learn more general features to reduce the overfitting in the source domains.

![Visualization Results](image)

Figure 10: The visualization of the class activation map [63] from source and target domains using PACS benchmark. The first row is the baseline method DeepAll and the second row is our method KDDG. The first three columns are source domains and the last is the unseen target domain.

C EXPERIMENT DETAILS

C.1 Digits-Five

For the data augmentation, we use the same augmentation for all baselines that normalize the image with a mean of 0.5 and a standard deviation of 0.5 and then resize to the size $32 \times 32$. The data which only have one gray channel are first converted into RGB by copying the gray channel. Following the setting in [42], we sample 25000 images from the training data and 9000 from testing data in MINST-M, MNIST, Synthetic Digits, and SVHN. For the dataset USPS which has less than 25000 images, we repeat it for 3 times to ensure that the data is roughly balanced

C.2 Gray Matter Segmentation

We adopt the same backbone network architecture 2D-Unet with [28] that the code is released in github ¹. We adopt the weighted binary cross-entropy as the classification loss, where the weight of a positive sample is set to the reciprocal of the positive sample ratio in the region of interest. The knowledge distillation loss is also weighted using the sample weight as above. We use Adam as the optimizer with the learning rate 1e-4. For data processing, the 3D MRI data are first sliced into 2D in axial slice view and then center cropped to $160 \times 160$. In the training phase, the random crop is used which leads to the size of $144 \times 144$. For evaluation, the center crop is used to resize the data to the same size. Adam optimizer is used to train the model with a learning rate of 1e-4, weight decay as 1e-8. The model is trained for 200 epochs and the learning rate is decreased 0.1 times of the original every 80 epoch. We set hyperparameter $\eta$ to 1-1e-4 for all the domains.

C.3 Mountain Car

We use the same setting for all methods. More specifically, we follow the method in [34]. Two networks are used as the action-value function and target action-value function using the same architecture which includes a single FC layer including 64 channels with a Relu activation function and two FC layers to predict the mean and the zero-center vector of the Q value separately. The state of the car is fed into the network including the position and speed. The capacity of the memory buffer is set to 500. MSE loss is used to calculate the loss in the Bellman equation. We use Adam as the optimizer with a learning rate of 1e-4 and the batch size is set to 10. The discount factor gamma is set to 0.9 and the exploration rate is set to 0.05. We train the model for 2000 episodes and merge two models every 10 episodes. As for the reward function, it is defined as the following piecewise function form

$$r(p) = \begin{cases} 
100 & p \geq 0.5 \\
10(0.4 + p)^3 & -0.4 < p < 0.5 \\
-0.1 & p \leq -0.4 
\end{cases}$$

(17)

where $p$ is the position of the car returned from the environment. With regard to our method KDDG, we first do the softmax on the output of the target action-value network and then do the same operation comparing with the operation in the object recognition task.

¹https://github.com/wyf0912/LDDG