We sincerely thank all reviewers for their valuable comments. Below we address concerns raised by all reviewers. We will carefully revise our paper, and release the code provided in the submission for reproducibility.

***Q1 (R1) The number of parameters. A1:** As stated in Section 3 in the submission, we follow the existing works and evaluate our method using AlexNet and ResNet. Therefore, our model has **the same number of parameters as the other methods**, *i.e.,* ~61M (AlexNet) and ~11M (ResNet-18) during inference. Moreover, at the training stage, most methods exploit auxiliary modules to model the constraints, which increase the number of parameters. For example, MASF [6] proposes a metric-learning component with two fully-connected layers for local sample clustering. Apart from the main network, Epi-FCR [20] trains one additional feature extractor and classifier for each domain. We provide the performance and the number of parameters (AlexNet) of several methods in Table 1 for better comparison.

Table 1: Performance on PACS, additional parameters at training, and parameters at inference. Left to right: Epi-FCR [20], CIDDG [22], MASF [6], and Ours.

Accuracy/%				Additional Param/M (Train)				Param/M (Infer)			
72.03	72.20	75.21	75.67	183	83.89	4.46	73.4	61	61	61	61

***Q2 (R1) Hyperparameters and dataset splits. A2:** At the training stage, we train the model on **the training set of the source domains**, and choose the hyperparameters and the best model on **the validation set** of the source domains. To evaluate the generalization capabilities, we **test the best model selected by source domain validation data** on the target dataset (whole set for VLCS, official test set for PACS). We will clarify this in the final version.

***Q3 (R1) Missing some existing works. A3:** Deshmukh *et al.* [3] proposed the kernel-based method for multi-class domain generalization, and proved the first known generalization error bound . Blanchard *et al.* [1] analyzed the problem of domain generalization from a different perspective by augmenting the original feature space. Then, they developed a kernel-based method that predicts classifiers from augmented feature space. As stated in Section 1 of the submission, Muandet *et al.* [2] proposed a kernel-based optimization algorithm, called DICA, which can not only minimize the difference between marginal distributions of the domains but also preserve the functional relationship between input and output variables. **In contrast, our work focuses on learning conditional-invariant deep representations across all source domains. We will add the discussion in the revision.**

***Q4 (R1) Dataset in Figure 1. A4:** Here, we compare our method with two methods, *i.e.,* the basic solution through adversarial learning (Basic-Adv) and CIDDG [22]. The latter one aims to learn domain-invariant features by introducing one domain discriminator for each class. To create the target dataset, we slightly adjust the two marginal distributions of Domain_0. The average accuracy over 5 repeated experiments is 78.2% (Ours), 78.0% (CIDDG), and 77.6% (Basic-Adv), respectively. Ours and CIDDG have close performance on this simple simulation dataset, since they both focus on learning the domain-invariant features. Additionally, both of them perform better than the baseline adversarial training method.

***Q5 (R3&R4) Claims about the improvements. A5:** In comparison to existing methods and the provided strong baseline, our model overall performs better. For example, our method yields higher average accuracy on both VLCS and PACS. For the Cartoon and Photo datasets, we hypothesize that the distinctive shape on the former and background on the latter make our method less effective on the two datasets. We appreciate the reviewer's constructive comments on the claim, and we will carefully improve the presentation in the revision.

***Q6 (R4) Class imbalance. A6:** We address the class imbalance issue by using the weighted cross-entropy loss according to the number of each class in each batch, which can be found in the provided source code (func *_compute_cls_loss* in train.py). **If not using the weighted loss**, *i.e.,* setting the weight to 1 for each class, the model yields a lower average accuracy of 75.58% (weighted loss used: 75.67%) on PACS.

***Q7 (R4) Additional classifiers. A7:** We exploit **the additional classifiers only at the training stage and remove them during inference,** *i.e.,* **only the feature extractor F and the main classifier T are preserved**, which can be found in the submitted supplementary (S.Section 3 and S.Figure 1). Therefore, our model has the same capacity as other methods in the inference stage, please refer to ***Q1** and Table 1 for details of the model parameters . Moreover, we have analyzed the loss function in the ablation study (Section 3.3). Specifically, when removing the last term of the loss function (*i.e.,* removing the extra classifiers), we obtain an average accuracy of 75.37% ($\alpha_3 = 0$ in Table 5), **which is lower than using all terms but higher than no entropy regularization** ($\alpha_2 = 0$). As we stated in L133-136 of the submission, we use the extra classifiers to make the training stage more stable. Additionally, as shown in Table 5, the model trained with the extra classifiers but without the proposed entropy regularization does not perform well in most cases (*e.g.,* the accuracy is less than 75%), while the entropy regularization performs better (>75.2%). This also shows the significance of the entropy regularization. The number of domains would affect the consumed memory during training, but has no impact in inference. **In a nutshell, the extra classifiers are only adopted at the training stage for improving the stability, and do not increase the number of parameters and model complexity. Additionally, the effectiveness of the proposed regularization term is verified in the ablation study.**