Summary of DG Papers (2)

1. JiGen

【**Paper**】Domain Generalization by Solving Jigsaw Puzzles

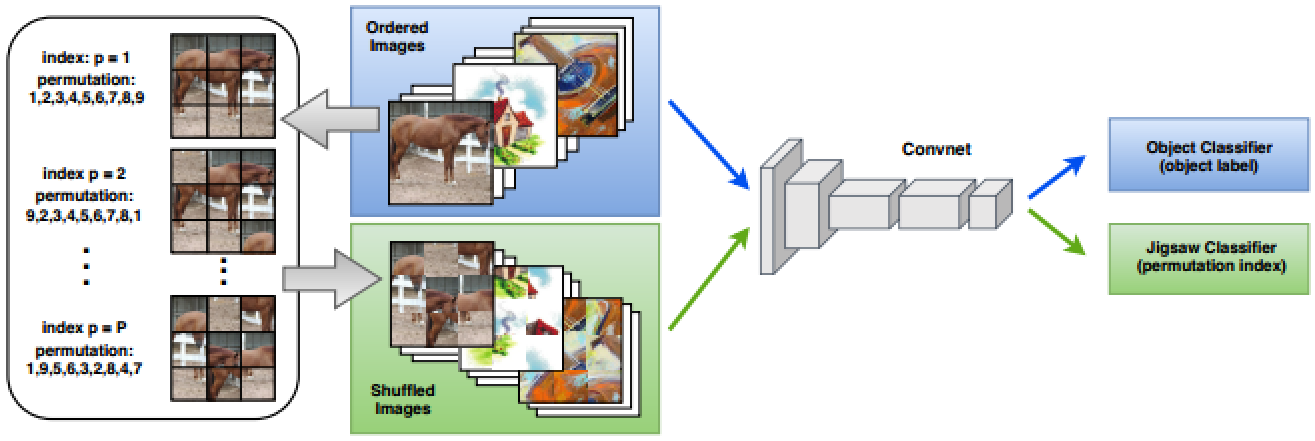
【**Date**】CVPR 2019

【**Tag**】Multi-task

This paper adopts a multi-task pipeline to guide the network to do a classification task as well as a "Jigsaw Puzzle" task at the same time.

The Jigsaw task is described as recovering the original image from shuffled orders of image patches. The motivation is that doing a Jigsaw task can help the network learn spatial correlations of different image parts, which may be useful for generalization. As for infants, doing a puzzle game can help them better recognize different objects.

The overall architecture of JiGen is as follows


index: p = 1
permutation: 1,2,3,4,5,6,7,8,9
index p = 2
permutation: 9,2,3,4,5,6,7,8,1
index p = P
permutation: 1,9,5,6,3,2,8,4,7
Ordered Images
Shuffled Images
Convnet
Object Classifier (object label)
Jigsaw Classifier (permutation index)

For each image, P different orders are chosen by the *maximal Hamming distance algorithm*. The Jigsaw task is defined as correctly classifying the permutation index *p* (i.e., the correct shuffled order). The shuffled images are not used for object classification because this would make object recognition tougher.

During the test stage, only the object classification stage is used.

Note that this method **do not need domain labels**.

【**Hyperparameters**】

There are four important hyperparameters in JiGen for DG settings:

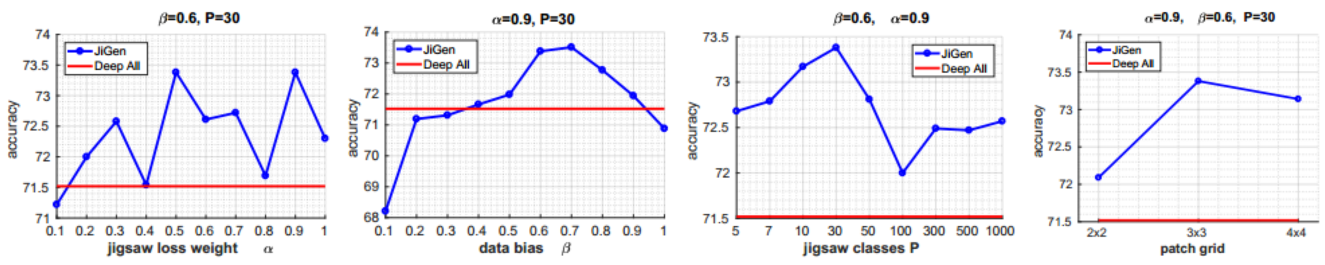
- the grid size $n \times n$
- the number of different patch permutation orders P

(This two hyperparameters are related to the jigsaw task.)

- weights α of the jigsaw loss
- the data bias parameter β, denoting that the ratio of the ordered images in a mini-batch

(This two hyperparameters are related to the learning process.)

The authors study the influences of different hyperparameters on PACS:



The impact of parameter β is saliency. For $\beta < 0.5$, the shuffled images are more than the ordered images, which leads the object classification task to be a less important task. Therefore the performance of JigGen is worse than baseline. For $\beta = 1$, the model does not utilize any information from shuffled images, therefore the performance is not good.

2. Feature-Critic

【Paper】Feature-Critic Networks for Heterogeneous Domain Generalization

【Date】2019

【Tag】Meta-learning

This paper proposed a meta-learning based method which aims at learning an auxiliary loss to guide the network learn to generalize. The method in this paper, called feature-critic, is very similar to *MetaReg*. Because learning an auxiliary loss can also be seen as learning a regularizer.

This paper also studied a new problem called **heterogeneous domain generalization**, which is defined as follows:

【Definition】 heterogeneous domain generalization refers to the DG problems where the label spaces between source and target domains are disjoint.

In heterogeneous DG problem, the labelled training and testing split of target domain are available. The authors use a fixed feature extractor to extract features, and then only train a simple classifier (e.g., SVM or KNN) with the training split of target domain.

Few-shot cases where only several training examples are available on target domain are also studied.

In the case of homogeneous DG (i.e., the common DG problems), both source domains and target domain use a same classifier, as their are handling a same task.

【Method】

The main difference between original Meta-Reg and feature critic is that Meta-Reg only regularizes on the classifier, while feature critic only regularizes on the feature extractor. But Meta-Reg can be adjusted to heterogeneous DG by applying the regularizer on the feature extractor instead of the classifier.

In addition, the regularizer of adjusted Meta-Reg operates on all the parameters of feature extractor. But the regularizer (or auxiliary loss) of feature-critic operates on the output features of feature extractor, in order to be scalable of parameters.

Two choices of auxiliary loss are available:

- the set embedding:

$$h_\omega(F) = \frac{1}{M}\sum_{i=1}^{M} \mathrm{MLP}_\omega(F_i)$$

- the flattened covariance matrix:

$$h_\omega(F) = \mathrm{MLP}_\omega\big(\text{ Flatten }\left(F^T F\right)\big)$$

where F is the $M \times H$ sized matrix, consisting of the H-dimensional features from M examples in a mini-batch. ω is the parameter of the learned auxiliary loss.

The source domains are split into virtual source domains (training domain) and virtual target domains (testing domains) in order to support meta-learning.

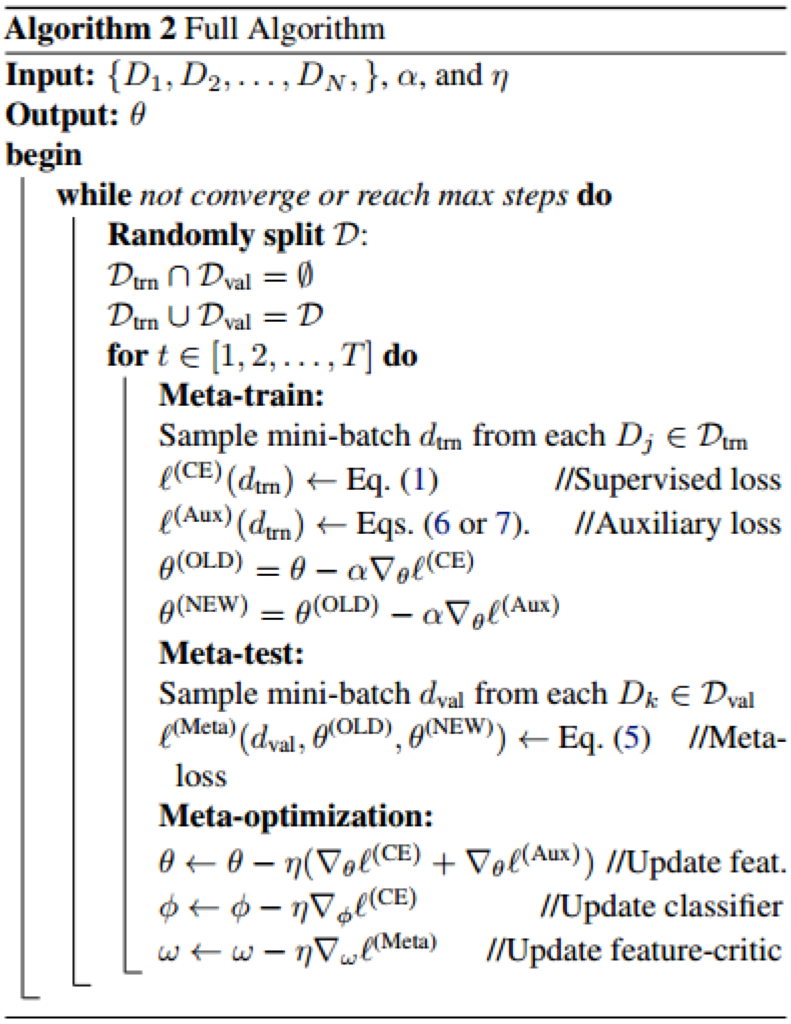
The objective used to optimize the auxiliary loss is:

$$\begin{aligned} \max\nolimits_{D_j \in \mathcal{D}_{\text{val}}, d_j \in D_j} \tanh\Big(&\gamma\left(\theta^{(\text{NEW})}, \phi_j, x^{(j)}, y^{(j)}\right) \\ &-\gamma\left(\theta^{(\text{OLD})}, \phi_j, x^{(j)}, y^{(j)}\right)\Big) \end{aligned}$$

where γ is a function that measures the validation domain performance (larger is better). θ^{New} and θ^{Old} are parameters updated with or without the auxiliary loss respectively. This objective ensures that after updating the auxiliary loss h_ω, the performance on the validation domain should be better than not using the auxiliary loss.

The authors use classification cross entropy for γ.

The full algorithm is:



Algorithm 2 Full Algorithm

```
Input: $\{D_1, D_2, \ldots, D_N,\}$, $\alpha$, and $\eta$
Output: $\theta$
begin
    while not converge or reach max steps do
        Randomly split $\mathcal{D}$:
        $\mathcal{D}_{\text{trn}} \cap \mathcal{D}_{\text{val}} = \emptyset$
        $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{val}} = \mathcal{D}$
        for $t \in [1, 2, \ldots, T]$ do
            Meta-train:
            Sample mini-batch $d_{\text{trn}}$ from each $D_j \in \mathcal{D}_{\text{trn}}$
            $\ell^{(\text{CE})}(d_{\text{trn}}) \leftarrow$ Eq. (1)          //Supervised loss
            $\ell^{(\text{Aux})}(d_{\text{trn}}) \leftarrow$ Eqs. (6 or 7).    //Auxiliary loss
            $\theta^{(\text{OLD})} = \theta - \alpha \nabla_\theta \ell^{(\text{CE})}$
            $\theta^{(\text{NEW})} = \theta^{(\text{OLD})} - \alpha \nabla_\theta \ell^{(\text{Aux})}$
            Meta-test:
            Sample mini-batch $d_{\text{val}}$ from each $D_k \in \mathcal{D}_{\text{val}}$
            $\ell^{(\text{Meta})}(d_{\text{val}}, \theta^{(\text{OLD})}, \theta^{(\text{NEW})}) \leftarrow$ Eq. (5)    //Meta-loss
            Meta-optimization:
            $\theta \leftarrow \theta - \eta(\nabla_\theta \ell^{(\text{CE})} + \nabla_\theta \ell^{(\text{Aux})})$ //Update feat.
            $\phi \leftarrow \phi - \eta \nabla_\phi \ell^{(\text{CE})}$          //Update classifier
            $\omega \leftarrow \omega - \eta \nabla_\omega \ell^{(\text{Meta})}$       //Update feature-critic
```

Step 1: update the feature extractor with the auxiliary loss on training domains

$$\theta \leftarrow \theta - \eta \left(\nabla_\theta \ell^{(\text{CE})} + \nabla_\theta \ell^{(\text{Aux})} \right)$$

Step 2: according to the last step, decomposing the update process to compute θ^{OLD} and θ^{NEW}

$$\theta^{(\text{OLD})} = \theta - \alpha \nabla_\theta \ell^{(\text{CE})}$$

$$\theta^{(\text{NEW})} = \theta^{(\text{OLD})} - \alpha \nabla_\theta \ell^{(\text{Aux})}$$

(*There remains a question here: why using different learning rate α from the last step η.*)

Step 3: fix the feature extractor, update the classifiers on training and validation domains respectively

$$\phi \leftarrow \phi - \eta \nabla_\phi \ell^{(\text{CE})}$$

Step 4: update the auxiliary loss using θ^{OLD}, θ^{NEW}, ϕ_{val} on the validation domain

$$\omega \leftarrow \omega - \eta \nabla_\omega \ell^{(\text{Meta})}$$

Note that the parameter θ of feature extractor is only updated on training domains; the parameter ω of the auxiliary loss is only updated on validation; the parameter ϕ of the classifier is domain-specific for heterogeneous DG but shared for homogenous DG.

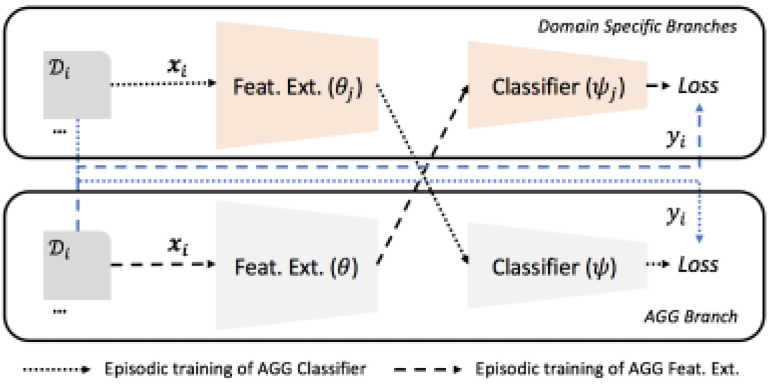
3. Epi-FCR

【**Paper**】Episodic Training for Domain Generalization

【**Date**】2019

【**Tag**】Training strategy

This paper proposed a simple training strategy to make the baseline AGG model more robust to domain shifts.



The method includes a domain-agnostic module and domain-specific modules. Each individual module consists of a feature extractor and a classifier. The domain-agnostic module is the final model shared across all domains. The domain specific modules are used to help the training of the domain-agnostic module.

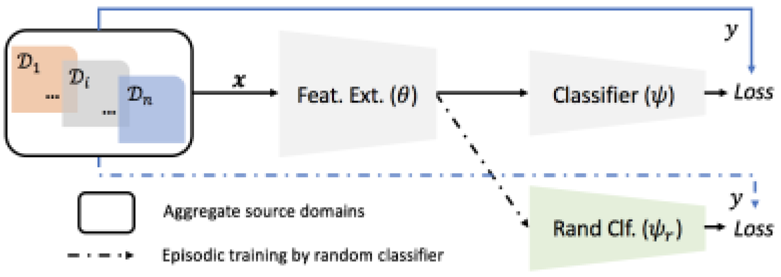
The main idea is to combine the universal feature extractor with domain-specific classifiers which are mismatched to the current input domain, or combine the universal classifier with mismatched domain-specific feature extractors.

By doing so, the universal feature extractor or classifier should try its best to help an "inexperienced" partner on current domain, which can improve the generalization ability of corresponding parts.

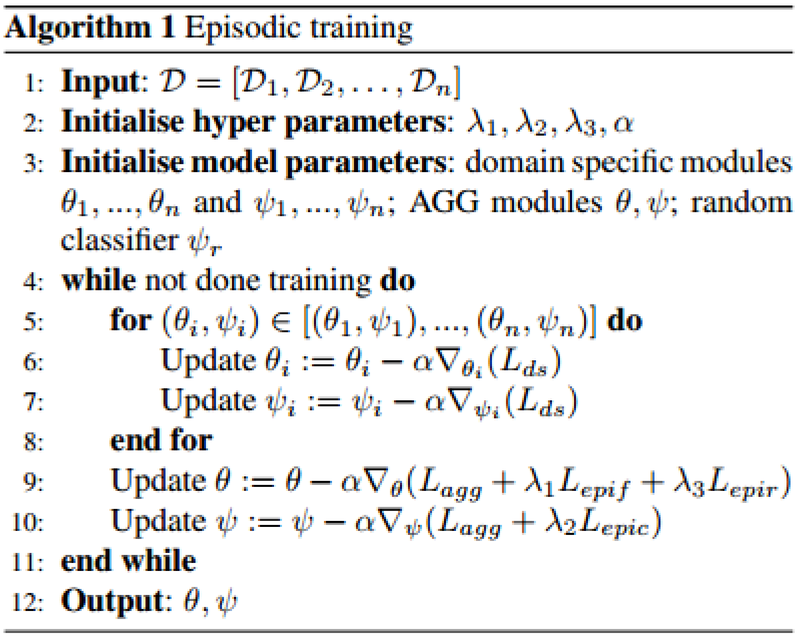
Note that when training the universal feature extractor, its paired domain-specific classifier is fixed. The settings for training the universal classifier is the likewise.

The authors also studied heterogenous DG problems. The goal is to find a robust off-the-shelf feature extractor.

In heterogenous DG, the label spaces are disjoint, hence mismatched domain-specific classifiers cannot be simply combined with the universal feature extractor. Therefore, the authors use a classifier with random weights to combine with the universal feature extractor. This configuration can also be used for homogenous DG.



The full algorithm is



Algorithm 1 Episodic training

```
1:  Input: $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n]$
2:  Initialise hyper parameters: $\lambda_1, \lambda_2, \lambda_3, \alpha$
3:  Initialise model parameters: domain specific modules $\theta_1, ..., \theta_n$ and $\psi_1, ..., \psi_n$; AGG modules $\theta, \psi$; random classifier $\psi_r$
4:  while not done training do
5:      for $(\theta_i, \psi_i) \in [(\theta_1, \psi_1), ..., (\theta_n, \psi_n)]$ do
6:          Update $\theta_i := \theta_i - \alpha\nabla_{\theta_i}(L_{ds})$
7:          Update $\psi_i := \psi_i - \alpha\nabla_{\psi_i}(L_{ds})$
8:      end for
9:      Update $\theta := \theta - \alpha\nabla_{\theta}(L_{agg} + \lambda_1 L_{epif} + \lambda_3 L_{epir})$
10:     Update $\psi := \psi - \alpha\nabla_{\psi}(L_{agg} + \lambda_2 L_{epic})$
11: end while
12: Output: $\theta, \psi$
```

【Issue】

Batch norm should be carefully used when applying modern CNN as base model, such as ResNet. Because the statistics between source and target domains are different. Using ImageNet pretrain models and freezing the statistics of BN will be a good choice.

4. AFLAC

【Paper】 Adversarial Invariant Feature Learning with Accuracy Constraint for Domain Generalization

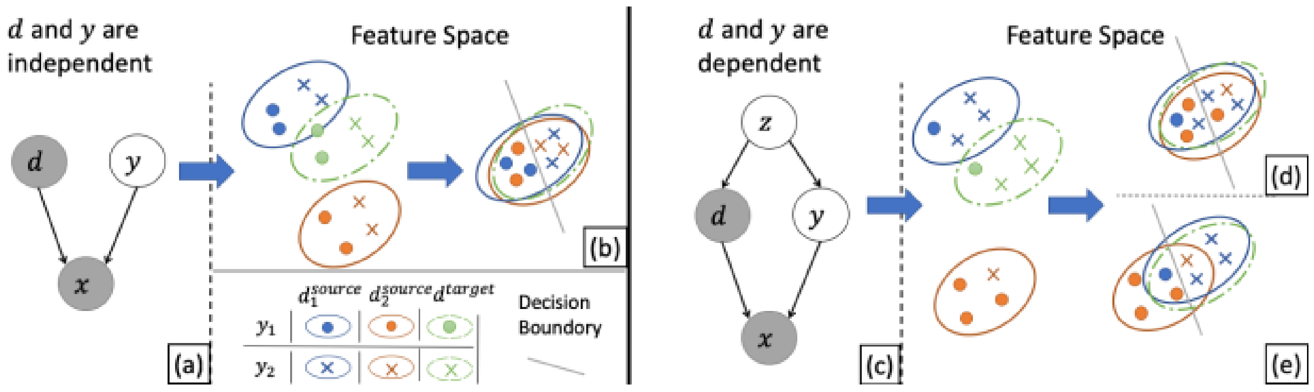
【Date】 2019

【Tag】 Adversarial

【Introduction】

This paper provides an insight for domain-invariant feature learning. Merely improving domain-invariance does not guarantee the prefect classification accuracy on target domain.

The paper assumes that domain d and class labels y are dependent. Therefore, there is a trade-off between domain-invariance and classification accuracy. The improvement of the invariance can negatively affect DG performance under this trade-off.


d and y are independent
Feature Space
d
y
x
(a)
(b)
d_1^{source} d_2^{source} d^{target}
y_1
y_2
Decision Boundory
d and y are dependent
Feature Space
z
d
y
x
(c)
(d)
(e)

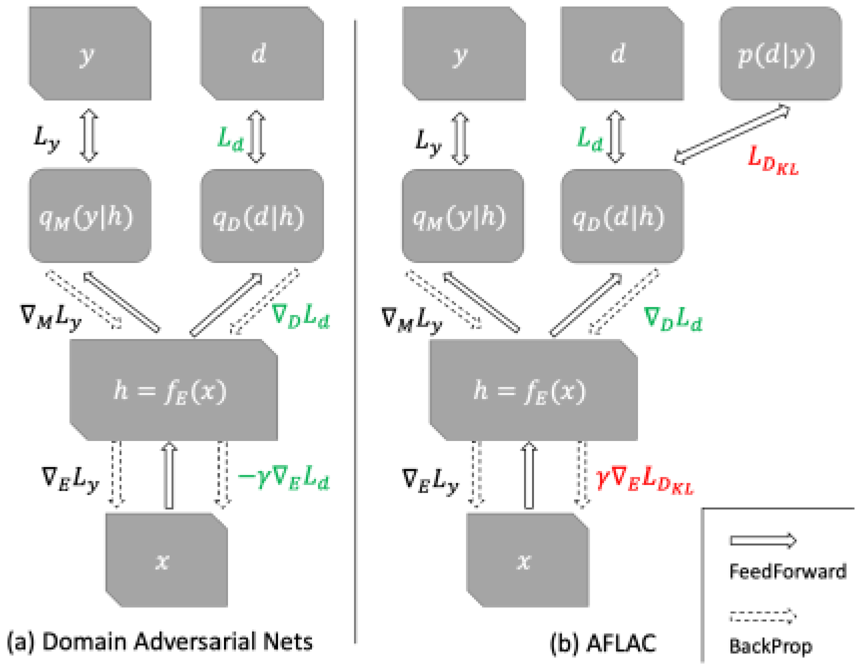
Since y contains information about d under the dependency, encoding information about d into y. However, previous domain-invariant feature learning methods attempt to remove all domain information from the learned latent feature h.

In contrast, the authors propose to maximize domain invariance within a range that does not interfere with the classification accuracy.

【Methodology】

The authors prove that when $H(d|h) = H(d|y)$, i.e., the learned representation h contains as much domain information as the class labels, it does not affect the classification performance.

The whole framework is based on DANN.

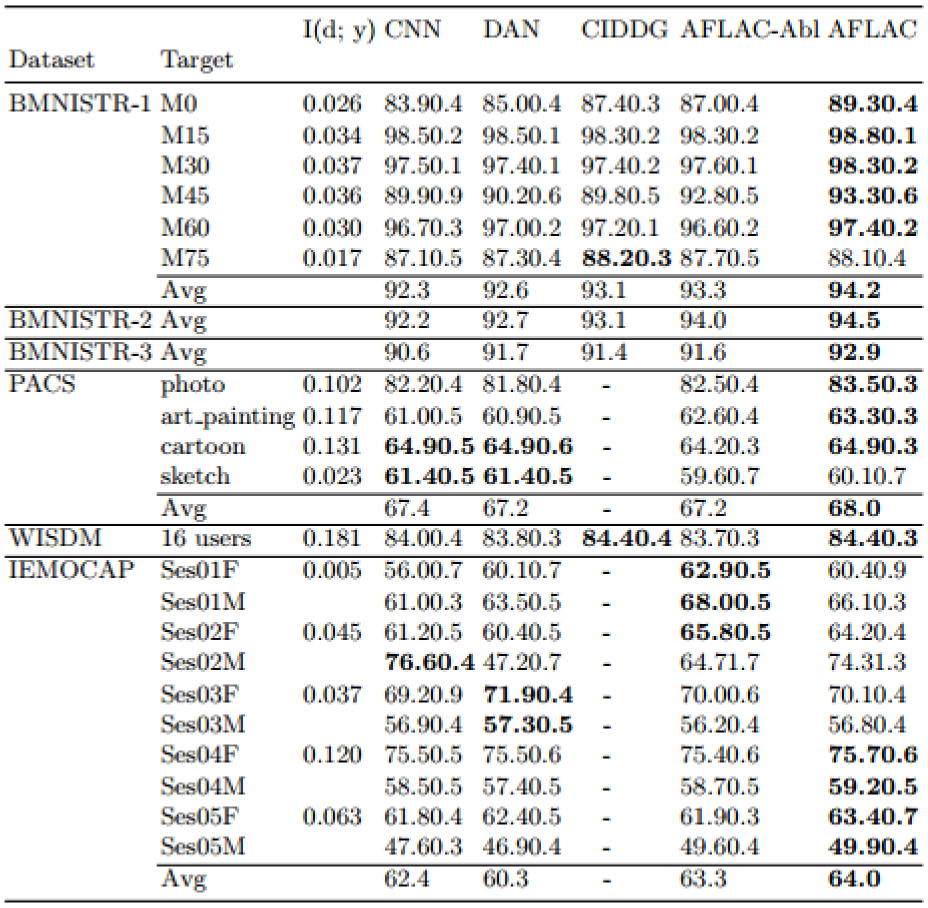


The authors impose regularization that ensures $H(d|h) = H(d|y)$, to achieve "accuracy -constrained domain invariance". In the framework of AFLAC, the encoder tries to minimize the label classification loss, as well as the KL divergence between $p(d|y)$ and $q_D(d|h)$ (i.e., the output of the discriminator). The discriminator tries to minimize the domain classification loss while the classifier tires to minimize the label classification loss.

$$\begin{aligned} \min_{D} W(E, D) =& \mathbb{E}_{p(x,d)}\left[L_d\right] \\ \min_{E,M} V(E, M) =& \mathbb{E}_{p(x,d,y)}\left[\gamma L_{D_{KL}} + L_y\right] \\ & \text{where} L_{D_{KL}} := D_{KL}\left[p(d|y) | q_D\left(d|h = f_E(x)\right)\right] \end{aligned}$$

【Experiment】

The experiment results of AFLAC are not very good. The average accuracy on PACS is 68%.



Dataset	Target	I(d; y)	CNN	DAN	CIDDG	AFLAC-Abl	AFLAC
BMNISTR-1	M0	0.026	83.90.4	85.00.4	87.40.3	87.00.4	**89.30.4**
	M15	0.034	98.50.2	98.50.1	98.30.2	98.30.2	**98.80.1**
	M30	0.037	97.50.1	97.40.1	97.40.2	97.60.1	**98.30.2**
	M45	0.036	89.90.9	90.20.6	89.80.5	92.80.5	**93.30.6**
	M60	0.030	96.70.3	97.00.2	97.20.1	96.60.2	**97.40.2**
	M75	0.017	87.10.5	87.30.4	**88.20.3**	87.70.5	88.10.4
	Avg		92.3	92.6	93.1	93.3	**94.2**
BMNISTR-2	Avg		92.2	92.7	93.1	94.0	**94.5**
BMNISTR-3	Avg		90.6	91.7	91.4	91.6	**92.9**
PACS	photo	0.102	82.20.4	81.80.4	-	82.50.4	**83.50.3**
	art_painting	0.117	61.00.5	60.90.5	-	62.60.4	**63.30.3**
	cartoon	0.131	**64.90.5**	**64.90.6**	-	64.20.3	**64.90.3**
	sketch	0.023	**61.40.5**	**61.40.5**	-	59.60.7	60.10.7
	Avg		67.4	67.2	-	67.2	**68.0**
WISDM	16 users	0.181	84.00.4	83.80.3	**84.40.4**	83.70.3	**84.40.3**
IEMOCAP	Ses01F	0.005	56.00.7	60.10.7	-	**62.90.5**	60.40.9
	Ses01M		61.00.3	63.50.5	-	**68.00.5**	66.10.3
	Ses02F	0.045	61.20.5	60.40.5	-	**65.80.5**	64.20.4
	Ses02M		**76.60.4**	47.20.7	-	64.71.7	74.31.3
	Ses03F	0.037	69.20.9	**71.90.4**	-	70.00.6	70.10.4
	Ses03M		56.90.4	**57.30.5**	-	56.20.4	56.80.4
	Ses04F	0.120	75.50.5	75.50.6	-	75.40.6	**75.70.6**
	Ses04M		58.50.5	57.40.5	-	58.70.5	**59.20.5**
	Ses05F	0.063	61.80.4	62.40.5	-	61.90.3	**63.40.7**
	Ses05M		47.60.3	46.90.4	-	49.60.4	**49.90.4**
	Avg		62.4	60.3	-	63.3	**64.0**