A. Detailed Proof

In the main paper, we propose to use a project gradient algorithm to efficiently optimize the hard IRM objective. In this section, we provide a formal proof composed of two main steps: (1) We show in Section A.1 that the original IRM objective is equivalent to the PG-IRM objective shown in Theorem 1. (2) In Section A.2, we show that the PG-IRM objective can be efficiently optimized by the project gradient descent algorithm illustrated in Alg. 2.

A.1. PG-IRM objective is equivalent to IRM

As a recap of our learning setting, a learner is given access to a set of training data from E environments $\mathcal{E} = \{e^{(1)}, e^{(2)}, .., e^{(E)}\}$ and the IRM objective is the following constrained optimization problem:

$$\min_{\phi,\beta^*} \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \mathcal{R}^e(\phi,\beta^*) \tag{11}$$

$$\text{s.t.} \quad \beta^* \in \arg\min_{\beta} \mathcal{R}^e(\phi,\beta) \quad \forall e\in\mathcal{E}, \tag{12}$$

where the risk function for a given domain/distribution e is:

$$\mathcal{R}^e(\phi,\beta) \doteq \mathbb{E}_{(\mathbf{x}_i,y_i,e_i=e)\sim\mathcal{D}} \ell\left(f(\mathbf{x}_i;\phi,\beta),y_i\right).$$

Theorem. *(Recap of Theorem 1) For all $\alpha \in (0,1)$, the IRM objective is equivalent to the following objective:*

$$\min_{\phi,\beta_{e^{(1)}},\ldots,\beta_{e^{(E)}}} \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \mathcal{R}^e(\phi,\beta_e) \tag{13}$$

$$s.t.\ \forall e\in\mathcal{E}, \exists \beta_e \in \Omega_e(\phi), \beta_e \in \Upsilon_\alpha(\beta_e), \tag{14}$$

where the parametric constrained set for each environment is simplified as

$$\Omega_e(\phi) = \arg\min_{\beta} \mathcal{R}^e(\phi,\beta),$$

and we define

$$\begin{aligned}\Upsilon_\alpha(\beta_e) = \{\upsilon |_{\min_{\forall e'\in\mathcal{E}\setminus e, \beta_{e'}\in\Omega_{e'}(\phi)}} \|\upsilon-\beta_{e'}\|_2 \\ \leq \alpha \min_{\forall e'\in\mathcal{E}\setminus e, \beta_{e'}\in\Omega_{e'}(\phi)} \|\beta_e-\beta_{e'}\|_2\}\end{aligned} \tag{15}$$

Proof. The constraint (12) means that the β^* is the optimal linear classifier at all environments, which is equivalent to saying that β^* lies in the joint of the optimal solution set in each environment. Equivalently, we can formularize the optimization target (11) as a parametric constrained optimization problem with constrain:

$$\beta^* \in \bigcap_{e\in\mathcal{E}} \Omega_e(\phi), \tag{16}$$

(arg min $\mathcal{R}^e(\phi,\beta)$ over β)

where the parametric constrained set for each environment is $\Omega_e(\phi) = \arg\min_{\beta} \mathcal{R}^e(\phi,\beta)$ (Note that $\Omega_e(\phi)$ can be a set with cardinality bigger than 1, since the optimal linear classifier may not be unique). The constraint (16) implies that β^* lies in the joint set of $\Omega_e(\phi)$, which also means that there is an element in each $\Omega_e(\phi)$ equal to β^*. We refer to such element to be $\beta_e \in \Omega_e(\phi)$, and we have the alternative form:

$$\forall e\in\mathcal{E}, \exists \beta_e\in\Omega_e(\phi), \beta^* = \beta_e \tag{17}$$

Equivalently,

$$\forall e\in\mathcal{E}, \exists \beta_e\in\Omega_e(\phi), \beta_e \in \bigcap_{e'\in\mathcal{E}\setminus e} \Omega_{e'}(\phi) \tag{18}$$

(by (16) and (17))

The interpretation of constraint (18) is that — for all environments, there is a hyperplane in the optimal set $\Omega_e(\phi)$ that also lies in the intersection of other environments' optimal set ($\bigcap_{e'\in\mathcal{E}\setminus e} \Omega_{e'}(\phi)$). Now we rewrite the optimization target (3) as:

$$\min_{\phi,\beta_{e^{(1)}},\ldots,\beta_{e^E}} \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \mathcal{R}^e(\phi,\beta_e) \tag{19}$$

$$\text{s.t. } \forall e\in\mathcal{E}, \exists \beta_e\in\Omega_e(\phi), \beta_e \in \bigcap_{e'\in\mathcal{E}\setminus e} \Omega_{e'}(\phi) \tag{20}$$

In this way, we can get rid of finding a unique β^*, but instead optimizing multiple linear classifiers $\beta_{e^{(1)}}, \ldots, \beta_{e^E}$, which is easier to optimize in a relaxed form as we will show next.

One key challenge for this optimization problem is that there is no guarantee that $\bigcap_{e'\in\mathcal{E}\setminus e} \Omega_{e'}(\phi)$ is non-empty for a feature extractor ϕ and β_e. We therefore relax the optimization target as:

$$\min_{\phi,\epsilon,\beta_{e^{(1)}},\ldots,\beta_{e^E}} \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \mathcal{R}^e(\phi,\beta_e) \tag{21}$$

$$\text{s.t. } \forall e\in\mathcal{E}, \exists \beta_e\in\Omega_e(\phi), \max_{e'\in\mathcal{E}\setminus e} \|\beta_e - \Omega_{e'}(\phi)\|_2 \leq \epsilon, \tag{22}$$

(relax $\beta_e \in \bigcap_{e'\in\mathcal{E}\setminus e} \Omega_{e'}(\phi)$)

where we define the l_2 distance between a vector β and a set Ω as : $\|\beta-\Omega\|_2 = \min_{\upsilon\in\Omega}\|\beta-\upsilon\|_2$.

Practically, ϵ can be set to be any variable converging to 0 during the optimization stage. Without losing the generality, we change the constraint (22) to the following form:

$$\begin{aligned}&\forall e\in\mathcal{E}, \exists \beta_e\in\Omega_e(\phi), \\ &\max_{e'\in\mathcal{E}\setminus e} \min_{\beta_{e'}\in\Omega_{e'}(\phi)} \|\beta_e-\beta_{e'}\|_2 \leq \\ &\qquad \alpha \max_{e'\in\mathcal{E}\setminus e} \min_{\beta_{e'}\in\Omega_{e'}(\phi)} \|\beta_e-\beta_{e'}\|_2,\end{aligned} \tag{23}$$

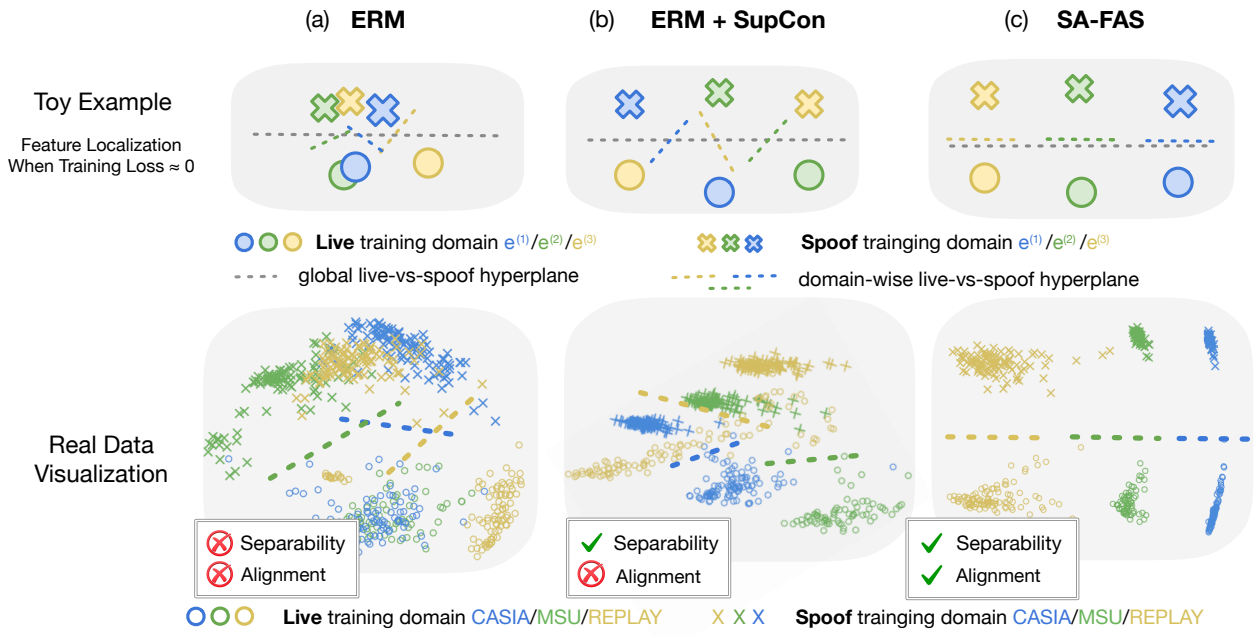


Figure 8. Illustration of feature space with three optimization objectives (ERM/ERM+SupCon/SA-FAS). For each objective, the first row shows the feasible solution in toy examples where each domain with a live/spoof label is represented by one circle/cross. The second role shows the visualization of real data via linear projection. The visualization is conducted by inserting and scattering the features from a 2-dimensional hidden layer between the penultimate layer and the final output layer.

where $\alpha \in (0, 1)$. Note that constraint (23) will be satisfied only when $\max_{e' \in \mathcal{E} \setminus e} \min_{\beta_{e'} \in \Omega_{e'}(\phi)} \|\beta_e - \beta_{e'}\|_2 = 0$. Therefore, constraint (23) is equivalent to constraint (18), and thus equivalent to the original constraint (12).

If we let the set

$$\begin{aligned} \Upsilon_\alpha(\beta_e) = \{\upsilon | & \max_{e' \in \mathcal{E} \setminus e} \min_{\beta_{e'} \in \Omega_{e'}(\phi)} \|\upsilon - \beta_{e'}\|_2 \\ & \leq \alpha \max_{e' \in \mathcal{E} \setminus e} \min_{\beta_{e'} \in \Omega_{e'}(\phi)} \|\beta_e - \beta_{e'}\|_2\} \end{aligned} \tag{24}$$

Then the constraint (23) can be simplified to

$$\forall e \in \mathcal{E}, \exists \beta_e \in \Omega_e(\phi), \beta_e \in \Upsilon_\alpha(\beta_e) \tag{25}$$

□

A.2. Projected Gradient Optimization for PG-IRM objective

We proceed with introducing how the Projected Gradient Descent can effectively optimize the PG-IRM objective. We start by introducing the background of the Projected Gradient Descent algorithm.

Projected Gradient Descent is commonly applied in constrained optimization, which aims to find a point θ achieving the smallest value of some loss function $\mathcal{L}$ subject to the requirement that θ is contained in the feasible set Ω. Formally, the objective can be written as:

$$\min_{\theta \in \Omega} \mathcal{L}(\theta)$$

If we minimize the objective $\mathcal{L}(\theta)$ by gradient descent, we have

$$\text{(GD)} \quad \theta := \theta - \gamma \nabla \mathcal{L}(\theta),$$

where γ is the step size. However, it is not guaranteed that the updated θ still falls into the set Ω. The projected gradient descent (PGD) algorithm is designed to project the solution back in the feasible set. Formally,

$$\text{(PGD)} \quad \theta := P_\Omega(\theta - \gamma \nabla \mathcal{L}(\theta)),$$

where the $P_\Omega(\cdot)$ is defined as the Euclidean Projection:

$$P_\Omega(u) = \arg\min_{v \in \Omega} \|u - v\|_2$$

In the PG-IRM objective, we have the constraint set $\Omega = \Upsilon_\alpha(\beta_e)$, we show in the next Lemma 2 that the Euclidean Projection from β_e to $\Upsilon_\alpha(\beta_e)$ is equivalent to the linear interpolation between β_e and the farthest hyperplane $\beta_{\bar{e}}$ for environment $\bar{e}$.

Lemma 2. *Given that*

Algorithm 2 PG-IRM

Initialize $\phi, \beta_{e^{(1)}}, ..., \beta_{e^{(E)}}$, learning rate γ, alignment parameter α, alignment starting epoch T_a.
for t in 0, 1, ..., **do**
  Run forward pass and calculate the gradient.
  for $e \in \mathcal{E}$ **do**
    $\tilde{\beta}_e^{t+1} = \beta_e^t - \gamma \nabla_{\beta_e^t} \mathcal{L}_{\textit{PG-IRM}}$
    $\alpha' := 1 - \mathbf{1}_{t>T_a}(1-\alpha)$
    select $\beta_{\bar{e}}^t$ with $\bar{e} = \underset{e' \in \mathcal{E} \setminus e}{\text{argmax}} \|\tilde{\beta}_e^{t+1} - \beta_{e'}^t\|_2$
    $\beta_e^{t+1} = \alpha' \tilde{\beta}_e^{t+1} + (1-\alpha')\beta_{\bar{e}}^t$
  end for
  Update $\phi^{t+1} = \phi^t - \gamma \nabla_{\phi^t} \mathcal{L}_{\textit{PG-IRM}}$.
end for

$$\Upsilon_\alpha(\beta_e) = \{v | \max_{e' \in \mathcal{E} \setminus e} \min_{\beta_{e'} \in \Omega_{e'}(\phi)} \|v - \beta_{e'}\|_2 \leq \alpha \max_{e' \in \mathcal{E} \setminus e} \min_{\beta_{e'} \in \Omega_{e'}(\phi)} \|\beta_e - \beta_{e'}\|_2\}$$

We have:

$$P_{\Upsilon_\alpha(\beta_e)}(\beta_e) = \alpha\beta_e + (1-\alpha)\beta_{\bar{e}},$$

where $\beta_{\bar{e}}$ *is selected with* $\bar{e} = \underset{e' \in \mathcal{E} \setminus e}{argmax} \|\beta_e - \beta_{e'}\|_2$.

Proof. We give the proof in an intuitive way shown in Figure 9. Specifically, the feasible region $\Upsilon_\alpha(\beta_e)$ can be regarded as an intersection of several hyper-spheres centered with all domain-wise live-vs-spoof hyperplanes $\beta_{e'}$. The radius is given by the α multiplying the distance to the farthest hyperplane $\beta_{\bar{e}}$. Therefore the Euclidean projection of β_e to the feasible set simultaneously lies on the surface of the hypersphere and the line segments between β_e and $\beta_{\bar{e}}$. It can be easily verified that

$$P_{\Upsilon_\alpha(\beta_e)}(\beta_e) = \alpha\beta_e + (1-\alpha)\beta_{\bar{e}},$$

satisfies the given criteria. □

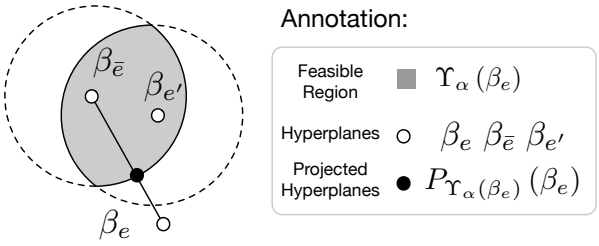


Figure 9. Illustration of the Euclidean projection results (solid black dot) to the feasible set $\Upsilon_\alpha(\beta_e)$.

Main results. When we have the projected form on the constraint set, deriving the optimization strategy is thus straightforward. As shown in Alg. 2, we first calculate the gradient of hyperplanes for all domains

$$\tilde{\beta}_e^{t+1} = \beta_e^t - \gamma \nabla_{\beta_e^t} \mathcal{L}_{\textit{PG-IRM}}.$$

We then select the farthest domain-wise hyperplanes $\beta_{\bar{e}}$ from other environments. The final projection results are thus given by

$$\beta_e^{t+1} = \alpha' \beta_e^{t+1} + (1-\alpha')\,\beta_{\bar{e}},$$

as we demonstrated in Lemma 2.

Remark on the T_a. In the first T_a epochs, we let the feature encoder ϕ and domain-wise hyperplanes β_e trained in a standard way. The goal is to ensure that the hyperplanes β_e will reach or be close to the minimum of the domain-wise empirical risk, and we have:

$$\beta_e \in \Omega_e(\phi).$$

In Alg. 2, we use an additional parameter α' to manifest this procedure:

$$\alpha' := 1 - \mathbf{1}_{t>T_a}(1-\alpha)$$

Specifically, when $t < T_a$, $\alpha' = 1$, which means the original gradient descent algorithm is applied. When $t < T_a$, $alpha' = \alpha$, the projected gradient descent takes charge.

B. Why do we need a fair setting?

By visualizing the line plot of the HTER performance over 100 training epochs in Fig. 10, we realize the test performance on the unseen domain is highly testset-dependent and unstable especially in the early epochs. Therefore, the best number reported commonly adopted in existing literature [30, 66, 72] usually happens in an unpredictable earlier epoch. Such "best" snapshot is also hard to be selected by validation strategy because we have zero information regarding the test domain. As an alternative, we noticed that the test performance is more stable in the last 10 epochs upon convergence, which motivates us to propose using a fairer comparison strategy introduced in Section 4.

C. Convergence of PG-IRM

Recall that in PG-IRM, we optimize multiple linear classifiers simultaneously $\beta_{e^{(1)}}, \beta_{e^{(2)}}, \beta_{e^{(3)}}$ and gradually align them during training. In this section, we would like to verify if PG-IRM indeed regularizes domain classifiers to be close to each other and finally converges to the same one $\beta^* = \beta_{e^{(1)}} = \beta_{e^{(2)}} = \beta_{e^{(3)}}$. Empirically, we use the averaged cosine distance between domain classifiers to measure the distance between them:

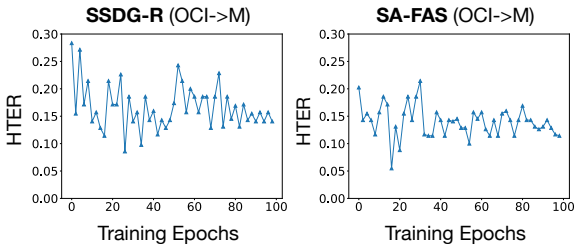


Figure 10. The line plot of the HTER performance tested on MSU dataset when trained on CASIA, Replay and OULU with SSDG-R [30] and SA-FAS over 100 training epochs.

$$S_{\cos} = \mathbb{E}_{e,e' \in \mathcal{E}, e \neq e'}[cos(\beta_e, \beta_{e'})]$$

As shown in Fig. 11, the averaged cosine value between domain classifiers diminishes gradually and finally converges to 1, which suggests that they converge to a β^* that is aligned for all domains.

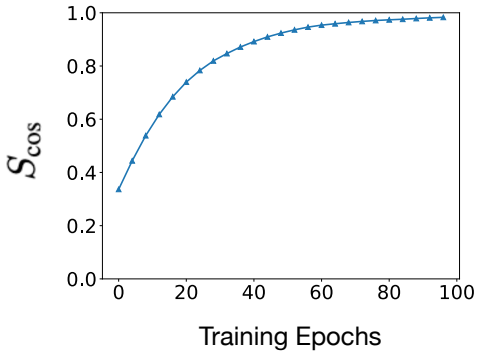


Figure 11. The line plot of the $S_{\cos}$ when trained on CASIA, Replay and OULU with PG-IRM over 100 training epochs.

D. Sensitivity Analysis

In this section, we perform the sensitivity analysis of hyper-parameter settings for SA-FAS in Fig. 12. The performance comparison in the bar plot for each hyper-parameter is reported by fixing other hyper-parameters. In the figure, we observe that the performance of SA-FAS is less sensitive to the learning rate and the alignment starting epoch compared with the maximum gap of 1.2% in the given range. We also notice that choosing the right alignment parameter α is more important, since a proper α ensures the domain-wise decision boundaries are aligned not too fast and not too slow. In the extreme case, if $\alpha = 0$, it degenerates to the ERM after epoch T_a and if $\alpha = 1$, the domain-wise boundaries will never get aligned with each other. In summary, our algorithm does not require heavy hyper-parameter tuning as long as it falls into a reasonable range.

E. Limitation

Our work has two limitations. Firstly, our framework assumes the dataset collected from each domain contains both live and spoof data. For example, SA-FAS can not handle the training data with live samples only from domain A and spoof samples only from domain B. Secondly, SA-FAS may cause extra computation costs when the domain amount is very large since we set up one hyperplane for each domain.

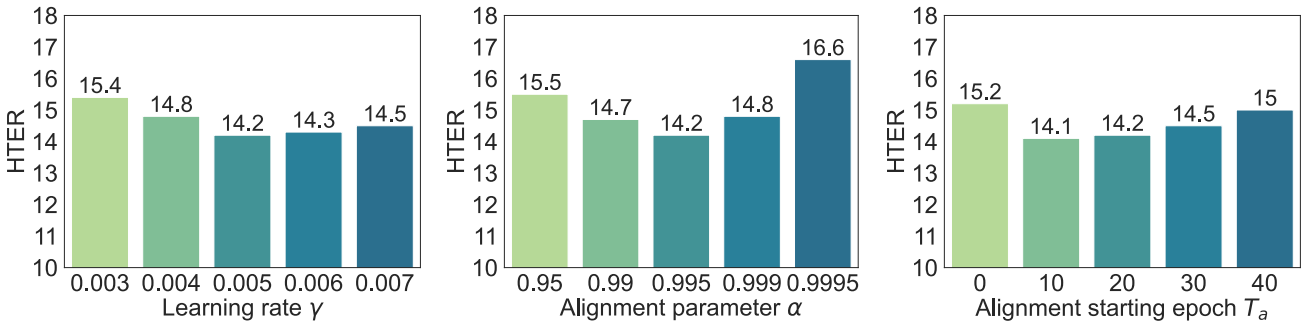
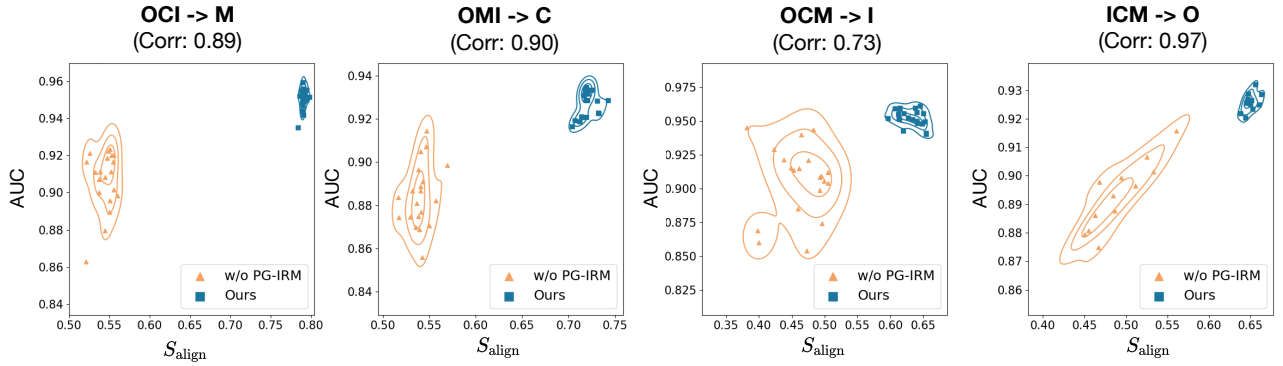
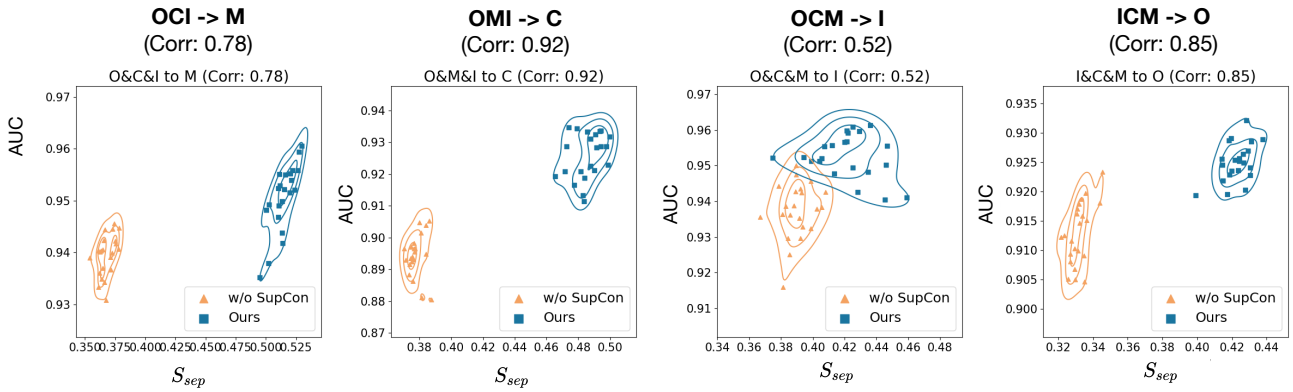


Figure 12. Sensitivity analysis of hyper-parameters: learning rate γ, alignment parameter α, alignment starting epoch T_a. The HTER is reported on the mean performance based on the last 10 epochs. The middle bar in each plot corresponds to the hyperparameter value used in our main experiments.



(a) Correlation Between AUC and Alignment



(b) Correlation Between AUC and Separability

Figure 13. Correlation between the test performance AUC and two properties measure. Each dot represents one snap-shot during the training stage in four cross-domain settings.