SUPPLEMENTARY MATERIAL: INFORMATION SELECTION-BASED DOMAIN ADAPTATION FROM BLACK-BOX PREDICTORS

In this supplementary material, we provide more details of the implementation and analysis of the proposed ISKD model, which are difficult to elaborate in the main paper due to the space limitation. For ease of reading, we show a quick preview of this supplementary as follows:

- We first provide a detailed description of the model implementation and the update algorithm for the parameters, and please kindly refer to the "code" folder for the specific implementation code.
- Then we additionally show the details of Gaussian Mixture Model (GMM) to obtain the predictions $\mathcal{P}^S = \{GMM(f^S(x_i^T))\}_{i=1}^{N_T}$ as well as cluster centers $\mathcal{U} = \{\mu_i\}_{i=1}^{C}$.
- We finally give more experimental analysis on different datasets to investigate (1) the effects of each parts of our ISKD method. (2) the parameter sensitivity of hyper-parameters.

1. ALGORITHM & IMPLEMENTATION DETAILS

Implementation Details. To obtain the robugstness of our ISKD method, we conducted each experiment three times and took the average as the experimental results. Generally, we randomly run our methods three times with different random seeds {2019,2020, 2021} via PyTorch and report the average accuracies. We train the source model by all the data in the source domain, and use ResNet as the backbone for the source model. We initialize the layers from the model pretrained on ImageNet and update the parameters through mini SGD, with the learning rate set to 1e-3 for the final stage and 1e-2 for the new layers. Apart from that, we also employ the rate scheduler, the momentum, weight decay and the bottleneck size are setted as 0.9, 1e-3 and 256 respectively.

Algorithm. In each round of training, all our parameters are updated only once, and we use ordinary gradient descent algorithm to update our parameters. Algorithm 1 demonstrates the distillation training process of the target model and Algorithm 2 demonstrates the fine-tuning process of the target model.

Algorithm 1 The distillation training process of our ISKD method

Input: Target set instances $\{x_i^T\}_{i=1}^{N_T}$, source model f^S, target model f^T.
Output: Model parameters Θ_{f^T}.
1: Initialize hyper-parameters α, learning rate lr.
2: **repeat**
3: Obtain the source features and target features through $f^S(x_i^T)$ and $f^T(x_i^T)$, respectively.
4: Compute the joint loss L_{jiont} through Eq. 10 in the text.
5: Update Θ_{f^T} by $\Theta_{f^T} \leftarrow \Theta_{f^T} - lr\nabla_{\Theta_{f^T}}(L_{\text{jiont}})$.
6: **until** Objective function of Eq. 10 converges or reaches maximum iterations.

Algorithm 2 The fine-tuning process of our ISKD method

Input: Target set instances $\{x_i^T\}_{i=1}^{N_T}$, target model f^T.
Output: Model parameters Θ_{f^T}.
1: Initialize hyper-parameters α, learning rate lr.
2: **repeat**
3: Obtain the target features through $f^T(x_i^T)$.
4: Compute the fine-tune loss L_{ft} through Eq. 11 in the text.
5: Update Θ_{f^T} by $\Theta_{f^T} \leftarrow \Theta_{f^T} - lr\nabla_{\Theta_{f^T}}(L_{\text{ft}})$.
6: **until** Objective function of Eq. 11 converges or reaches maximum iterations.

2. GAUSSIAN MIXTURE MODEL

The predictions of instances $\mathcal{P} = \{p_i\}_{i=1}^{N_T}$ are initialized as $p_{ij} = 1/C$, where C is the number of classes. The weights of classes $\mathcal{W} = \{w_j\}_{j=1}^{C}$ is initialized by $w_j = \sum_{i=0}^{N_T} \sigma(f(x_i))_j$, where $\sigma(\cdot)$ is the softmax function and $\sigma(f^S(x_i))_j$ is the $j-th$ value of $\sigma(f(x_i))$. And the centers of clusters $\mathcal{U} = \{\mu_j\}_{j=1}^{C}$ are initialized by $\mu_j = \frac{\sum_{i=1}^{N_T} \sigma(f(x_i))_j * f(x_i)}{w_j}$.

Then, update the predictions through:

$$\phi(x_i|\mu_j,\Sigma_j) = exp(-\frac{1}{2}(d\log 2\pi + \log|\Sigma_j| + (f(x_i)-\mu_j))^{\mathrm{T}}\Sigma_j^{-1}(f(x_i)-\mu_j)) \quad (1)$$

$$p_{ij} = \frac{\phi(x_i|\mu_j,\Sigma_j) * w_j}{\sum_{j'}\phi(x_i|\mu_{j'},\Sigma_{j'}) * w_{j'}} \quad (2)$$

where $exp(\cdot)$ is the exponential functions of natural constants, Σ is the covariance matrix of class j and j' is the class that belongs to the class set. Finally, we update the weight of each class and the center of each class through $w_j = \sum_{i=0}^{N_T} p_{ij}$ and $\mu_j = \frac{\sum_{i=1}^{N_T} p_{ij} * f(x_i)}{w_j}$ respectively.

3. FUTHER ANALYSIS

Table 1. Ablation study on **VISDA-C**.

The Variants of Our ISKD Method	Avg
No Adapt.	48.9
No Self Knowledge Distillation	81.9
No Partial Knowledge Distillation	77.4
No Confidence Score	81.0
ISKD (full)	**83.8**

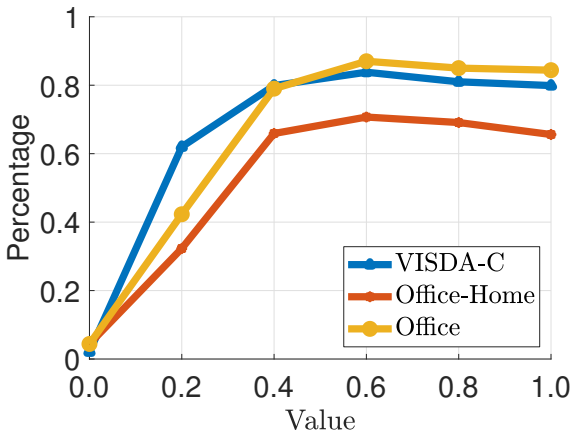

Fig. 1. Parameter sensitivity study on **VISDA-C**,**Office-Home** and **Office**.

Ablation Studies. To study the contribution of the various parts of our model, we perform ablation experiments on VISDA-C. We target three variants: No Self Knowledge Distillation: training model without self-distillation, No Partical Knowledge Distillation: no processing of the teacher network output, and No Partial Knowledge Distillation: no use of confidence scores. The experimental results show that both the confidence score and the partial distillation mechanism help a lot in model performance improvement, which indicates that our proposed contributions are practical and effective.

Analysis on Parameter Sensitivity. We analyse the sensitivity of the hyperparameter α on the three datasets in Fig. 1. The three datasets share the same trend, and when α is too small, the model performs poorly because the $\mathcal{L}_{pkd}$ is not weighted enough for the target model to gain sufficient knowledge from the source model. When α is too large, the performance of the source network constrains the performance of the target model because the effect of $\mathcal{L}_{skd}$ decreases. The best performance of the target network is reached when α is set to 0.6.