Appendix: Cross-Domain Cross-Set Few-Shot Learning via Learning Compact and Aligned Representations

Wentao Chen[1,2], Zhang Zhang[2,3], Wei Wang[2,3], Liang Wang[2,3], Zilei Wang[1], and Tieniu Tan[1,2,3]

[1] University of Science and Technology of China, Hefei, China
[2] Center for Research on Intelligent Perception and Computing, NLPR, CASIA, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
wentao.chen@cripac.ia.ac.cn, zlwang@ustc.edu.cn
{zzhang, wangwei, wangliang, tnt}@nlpr.ia.ac.cn

1 Details of our approach

Hyper-parameters. In our implementation, ResNet-18 [2] is adopted as the backbone, which outputs a 512-d feature vector. Before feeding the vector for prototypical alignment, we apply ℓ_2 normalization for the feature vector and prototypes. The temperature τ for ℓ_{s-t} and ℓ_{t-s} is 0.25 and 0.1, respectively. The max training steps T_{max} is set as 50,000 for the DomainNet and 1,000 for the Office-Home, which are roughly equal to *training epochs* $\times$ *dataset size/batch size*. The confidence threshold β for ℓ_{t-s} is set as 0.5. λ is equal to 0.2 to balance the pseudo labels generated by the initial classifier and the online updated classifier. The momentum term m is set as 0.1. These hyper-parameters are tuned based on performance on the validation set.

Training. We train our approach for 50 epochs on the DomainNet dataset. On the smaller Office-Home dataset, we train the model for 100 epochs. Adam [4] is adopted as the default optimizer with the learning rate as 1e-3. The batch size is set as 256, where source data and target data have the same number in a batch (128).

Evaluation. During evaluation, we fix the feature extractor and apply ℓ_2 normalization to the output feature vector. The linear classification head for each few-shot task (episode) is randomly initialized, and trained on the support features for 1000 steps with logistic regression. 15 query samples per class are used to evaluate the performance of the learned classifier. We finally report the average 5-way 1-shot and 5-way 5-shot accuracies over 600 episodes with 95% confidence intervals.

Table 1. The impact of interpolation coefficient λ.

	painting-real		real-painting	
λ	1-shot	5-shot	1-shot	5-shot
0.0	$53.91_{\pm 1.03}$	$72.64_{\pm 0.85}$	$53.73_{\pm 0.90}$	$64.95_{\pm 0.79}$
0.2	$54.44_{\pm 1.00}$	$\mathbf{73.63_{\pm 0.82}}$	$53.86_{\pm 0.89}$	$\mathbf{65.65_{\pm 0.74}}$
0.4	$\mathbf{54.55_{\pm 1.03}}$	$73.50_{\pm 0.83}$	$\mathbf{53.99_{\pm 0.90}}$	$64.87_{\pm 0.78}$
0.6	$50.50_{\pm 1.03}$	$69.11_{\pm 0.89}$	$50.47_{\pm 0.87}$	$61.26_{\pm 0.81}$
0.8	$50.07_{\pm 1.00}$	$68.50_{\pm 0.90}$	$50.40_{\pm 0.87}$	$60.52_{\pm 0.80}$
1.0	$49.79_{\pm 1.03}$	$68.42_{\pm 0.90}$	$50.28_{\pm 0.89}$	$60.60_{\pm 0.79}$

2 Updating pseudo label

Since we resort to pseudo labels for prototype estimation and feature alignment, ensuring the pseudo label accuracy is very important to the effectiveness of our bi-directional prototypical alignment strategy. Pseudo labels can be predicted with a fixed classifier pre-trained on the source base dataset, as in [8], or a classifier that is online updated along the representation learning. In our implementation, we combine them together by linearly interpolating their pseudo labels. We assess the effectiveness of this combination strategy by changing the interpolation coefficient λ from zero to one. When the interpolation coefficient $\lambda = 0\,(\text{or}1)$, our approach degenerates to only using the fixed (or online updated) classifier. The results on the DomainNet are shown in Table 1.

It can be noticed that the performance grows as we increase λ from zero and the best performance can be achieved when $\lambda \in [0.2, 0.4]$. The improvement demonstrates that updating the fixed pseudo labels with an online classifier is useful to get better pseudo labels. However, when λ gets too large, the performance drops very quickly, which means we can not only depend on the online classifier. The possible reason is that the pseudo labels predicted by the online classifier change rapidly, and thus impose adverse impacts on the training stability.

3 Hyper-parameter sensitivity

To analyse the sensitivity of a hyper-parameter, we change its value from the minimum to the maximum and keep other hyper-parameters unchanged. We test the performance of each value on the DomainNet *real-painting* and *painting-real*. The experimental results are shown in Figures 1 and 2. For the momentum coefficient m, a small m is usually better than a large one. The gap between the best performance ($m = 0.1$) and the worst performance ($m = 0.99$) is 2.2 points in 1-shot and 1.6 points in 5-shot. For the confidence threshold β, the performance grows in the range of $[0, 0.3]$ and decreases rapidly in the rage of $[0.5, 0.99]$. The difference between the best and the worst results are 2.4 points in 1-shot and 2.3 points in 5-shot, which are a little bit larger than the differences

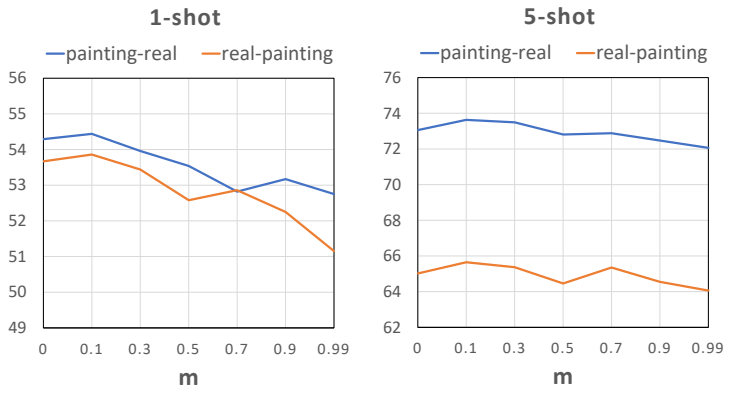

Fig. 1. The sensitivity of momentum coefficient m.

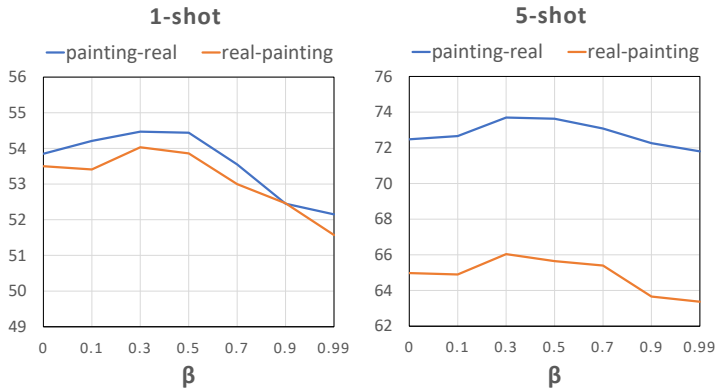

Fig. 2. The sensitivity of confidence threshold β.

of m. However, the performance of the proposed approach is still competitive even with the worst hyper-parameters, indicating that our approach is not very sensitive to hyper-parameters.

4 Prototype Distance (PD) and Average Distance Ratio (ADR)

To measure domain distance, we first calculate prototypes p_k^s and p_k^t for each novel class in the source and target domains. Then we obtain the Euclidean distance between the two prototypes per class and compute the average distance over all novel classes. We refer to this metric as Prototype Distance (PD), which

can be formulated as:

$$PD = \frac{1}{|\mathcal{Y}_\mathcal{N}|} \sum_{k \in \mathcal{Y}_\mathcal{N}} ||p_k^s - p_k^t||, \tag{1}$$

where $\mathcal{Y}_\mathcal{N}$ is the set of novel classes. A small PD value means the two domains are well aligned to each other.

To represent class separability, for each sample (x_i, y_i), we calculate the ratio between its distance to the prototype p_{y_i}, and the distance to the closest neighbouring prototype. Then an average is computed over all samples in novel classes, which is termed Average Distance Ratio (ADR). Formally,

$$ADR = \frac{1}{|\mathcal{X}_\mathcal{N}|} \sum_{x_i \in \mathcal{X}_\mathcal{N}} \frac{||f(x_i) - p_{y_i}||}{\min_{k \neq y_i} ||f(x_i) - p_k||}, \tag{2}$$

where $\mathcal{X}_\mathcal{N}$ is the set of samples of novel classes. When ADR is less than 1, most samples are classified into their ground-truth classes. We calculate ADR for two domains separately to validate whether the learned features can generalize in each domain.

5 Baselines

For a fair comparison, we implement all the baseline methods with the same ResNet-18 backbone adopted in our approach. But the augmentation strategies may be different for different methods, as some methods [18, 7, 10, 8, 3] have specified particular augmentation in their papers, where FixMatch[10] adopt the same augmentation techniques as ours. When no augmentation is specified, we simply apply CenterCrop and Normalization to the input images.

ProtoNet and RelationNet. ProtoNet[9] and RelationNet[11] are two representative meta-learning methods, which are trained on a series of few-shot tasks (episodes). We implement these two methods based on publicly-available codes [4]. During training, we randomly sample episodes from the base set, each of which contains $N = 5$ classes and $K = 5$ samples per class serving as the support set, and another 15 samples per class as the query set. We also train ProtoNet and RelationNet for 50 epochs on the DomainNet dataset and 100 epochs on the Office-Home dataset. The number of training episodes of each epoch is particularly defined to make sure the number of seen samples (both the support and query samples) in an epoch is roughly equal to the size of the dataset.

MetaOptNet. MetaOptNet[5] aims to learn an embedding function that generalizes well to novel categories with closed-form linear classifiers (e.g., SVMs). We implement this method based on the official code [5] but replace the backbone network and optimizer to be the same as our approach. Similar to ProtoNet and RelationNet, the training process of MetaOptNet is also episodic.

[4] https://github.com/wyharveychen/CloserLookFewShot
[5] https://github.com/kjunelee/MetaOptNet

Tian et al. Tian et al.[14] follows the transfer learning paradigm, which trains a base model by classifying base classes, and then leverages the learned representations to classify novel classes by learning a new classification head. We train this baseline with the same optimization method as our approach except that the batch size is set as 128 as only source data are used for training.

DeepEMD. DeepEMD[18] is also a meta-learning method, which aims to compute the query-support similarity based on Earth Mover's Distance (EMD). It contains two training phases: (i) pre-training the feature extractor by classifying base classes (similar to Tian et al.) and (ii) meta-training the whole model on training episodes. We use the output model of Tian et al. as the pre-trained model and then follow the official implementation [6] to finetune the model via meta-training.

FWT and ATA. FWT[15] and ATA[17] are two CD-FSL methods, which aims to learn generalized representations during meta-training so that the model can generalize to a new domain. To this end, FWT proposes a feature-wise transformation layer, of which the parameters can be manually set, or learned from multiple data sources. In our experiments, we choose to manually set the parameters as only data from one domain (the source domain) are labeled. ATA proposes to augment the task distributions by maximizing the training loss and meanwhile learn robust inductive bias from augmented task distributions. It does not need to access extra data sources, and thus can be trained on the base set. We implement these two methods based on their official codes[7], except that we train them from scratch as we find that additional pre-training will reduce performance.

S2M2. S2M2[7] follows the transfer learning paradigm, which leverages the data augmentation technique, MixUp[16], and self-supervised learning tasks (e.g., rotation) to learn generalized representation for few-shot learning. We follow the same augmentation and implementation as the official codes[8].

DANN. We use a three-layer fully connected network as the domain discriminator to implement DANN, following the Pytorch implementation [9] released by [6]. The gradient reverse layer [1] is adopted to train the feature vector and domain discriminator in an adversarial manner. To stabilize training, the weight of the adversarial loss starts from zero, and gradually grows to one.

PCT. PCT[12] is a generic domain adaptation method that can deal with single-source, multi-source, class-imbalance and source-private domain adaptation problems. Similar to our approach, PCT also aligns features via prototypes.

[6] https://github.com/icoz69/DeepEMD
[7] https://github.com/hytseng0509/CrossDomainFewShot, https://github.com/Haoqing-Wang/ CDFSL-ATA
[8] https://github.com/nupurkmr9/S2M2_fewshot
[9] https://github.com/thuml/CDAN

However, it only aligns features from the target domain to the prototypes trained with labeled source domain data. We implement this baseline according to the official codes[10].

Mean Teacher, Fixmatch and STARTUP. All of these approaches use pseudo-labeled samples to train the model. Differently, Mean Teacher[13] predicts pseudo labels with a teacher network that is the ensemble of historical models by aggregating their model weights with exponential moving average (EMA). In our implementation, the smoothing coefficient for EMA is set as 0.99. Fixmatch[10] trains the model with a consistency loss, i.e., enforcing the network prediction for a strongly augmented sample to be consistent with the prediction of its weakly augmented counterpart. We implement Fixmatch based on a publicly available implementation[11]. STARTUP[8] adopts fixed pseudo labels that are predicted by a classifier pre-trained on the base set, and imposes a self-supervised loss on the target data. In our re-implementation, we do not utilize the self-supervised loss item since we find that it does not improve performance.

6 Dataset partition details

DomainNet. DomainNet contains 345 classes in total. We discard 19 classes with too few images and randomly split the rest 326 classes into three sets: 228 classes for the base set, 33 classes for the validation set, and 65 classes for the novel set. The detailed classes of each set are listed below:

$$\mathcal{Y}_{base} =$$

{aircraft carrier, airplane, alarm clock, ambulance, animal migration, ant, asparagus, axe, backpack, bat, bathtub, beach, bear, beard, bee, belt, bench, bicycle, binoculars, bird, book, boomerang, bottlecap, bowtie, bracelet, brain, bread, bridge, broccoli, broom, bus, butterfly, cactus, cake, calculator, camera, candle, cannon, canoe, car, cat, ceiling fan, cell phone, cello, chair, church, circle, clock, cloud, coffee cup, computer, couch, cow, crab, crayon, crocodile, cruise ship, diamond, dishwasher, diving board, donut, dragon, dresser, drill, drums, duck, ear, elbow, elephant, envelope, eraser, eye, fan, feather, fence, finger, fire hydrant, fireplace, firetruck, flamingo, flashlight, flip flops, flower, flying saucer, foot, fork, frog, frying pan, giraffe, goatee, grapes, grass, guitar, hamburger, hammer, hand, harp, headphones, hedgehog, helicopter, helmet, hockey puck, hockey stick, horse, hot air balloon, hot tub, hourglass, hurricane, jacket, key, keyboard, knee, ladder, lantern, laptop, leaf, leg, light bulb, lighter, lightning, lion, lobster, lollipop, mailbox, marker, matches, megaphone, mermaid, microphone, microwave, moon, motorbike, moustache, nail, necklace, nose, octagon, oven, paint can, paintbrush, palm tree, panda, pants, paper clip, parachute, parrot, passport, peanut, pear, peas, pencil, penguin, pickup truck, picture frame, pizza, pliers, police car, pond, popsicle, postcard, potato, power outlet, purse, rabbit, radio, rain, rainbow, rake, remote control, rhinoceros, rifle, sailboat, school bus, scorpion, screwdriver, see saw, shoe, shorts, skateboard, skyscraper, smiley

[10] https://github.com/korawat-tanwisuth/Proto_DA
[11] https://github.com/kekmodel/FixMatch-pytorch

face, snail, snake, snorkel, soccer ball, sock, stairs, stereo, stethoscope, stitches, stove, strawberry, submarine, sweater, swing set, sword, t-shirt, table, teapot, teddy-bear, television, tent, the Eiffel Tower, the Mona Lisa, toaster, toe, toilet, tooth, toothbrush, tornado, tractor, train, tree, triangle, trombone, truck, underwear, van, vase, violin, washing machine, watermelon, waterslide, whale, wheel, windmill, wine bottle, zigzag}

$$\mathcal{Y}_{validation} =$$

{arm, birthday cake, blackberry, bulldozer, campfire, chandelier, cooler, cup, dumbbell, hexagon, hospital, house plant, ice cream, jail, lighthouse, lipstick, mushroom, octopus, raccoon, roller coaster, sandwich, saxophone, scissors, skull, speedboat, spreadsheet, suitcase, swan, telephone, traffic light, trumpet, wine glass, wristwatch}

$$\mathcal{Y}_{novel} =$$

{anvil, banana, bandage, barn, basket, basketball, bed, blueberry, bucket, camel, carrot, castle, clarinet, compass, cookie, dog, dolphin, door, eyeglasses, face, fish, floor lamp, garden, garden hose, golf club, hat, hot dog, house, kangaroo, knife, map, monkey, mosquito, mountain, mouth, mug, ocean, onion, owl, piano, pig, pillow, pineapple, pool, river, rollerskates, sea turtle, sheep, shovel, sink, sleeping bag, spider, spoon, squirrel, steak, streetlight, string bean, syringe, tennis racquet, the Great Wall of China, tiger, toothpaste, umbrella, yoga, zebra}

Office-Home. There are 65 classes in the Office-Home dataset. We select 40 classes as the base set, 10 classes as the validation set, and 15 classes as the novel set, which are listed below:

$$\mathcal{Y}_{base} =$$

{alarm clock, bike, bottle, bucket, calculator, calendar, chair, clipboards, curtains, desk lamp, eraser, exit sign, fan, file cabinet, folder, glasses, hammer, kettle, keyboard, lamp shade, laptop, monitor, mouse, mug, paper clip, pen, pencil, postit notes, printer, radio, refrigerator, scissors, sneakers, speaker, spoon, table, telephone, toothbrush, toys, tv}

$$\mathcal{Y}_{validation} =$$

{bed, computer, couch, flowers, marker, mop, notebook, pan, shelf, soda}

$$\mathcal{Y}_{novel} =$$

{backpack, batteries, candles, drill, flipflops, fork, helmet, knives, oven, push pin, ruler, screwdriver, sink, trash can, webcam}

References

1. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)

2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
3. Islam, A., Chen, C.F.R., Panda, R., Karlinsky, L., Feris, R., Radke, R.J.: Dynamic distillation network for cross-domain few-shot recognition with unlabeled data. Advances in Neural Information Processing Systems **34**, 3584–3595 (2021)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
5. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
6. Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. In: Advances in Neural Information Processing Systems. pp. 1645–1655 (2018)
7. Mangla, P., Kumari, N., Sinha, A., Singh, M., Krishnamurthy, B., Balasubramanian, V.N.: Charting the right manifold: Manifold mixup for few-shot learning. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2218–2227 (2020)
8. Phoo, C.P., Hariharan, B.: Self-training for few-shot transfer across extreme task differences. In: International Conference on Learning Representations (2021)
9. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems. vol. 30 (2017)
10. Sohn, K., Berthelot, D., Li, C.L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. arXiv preprint arXiv:2001.07685 (2020)
11. Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P.H., Hospedales, T.M.: Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1199–1208 (2018)
12. Tanwisuth, K., FAN, X., Zheng, H., Zhang, S., Zhang, H., Chen, B., Zhou, M.: A prototype-oriented framework for unsupervised domain adaptation. In: NeurIPS (2021)
13. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Advances in Neural Information Processing Systems. vol. 30 (2017)
14. Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J.B., Isola, P.: Rethinking few-shot image classification: a good embedding is all you need? In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)
15. Tseng, H.Y., Lee, H.Y., Huang, J.B., Yang, M.H.: Cross-domain few-shot classification via learned feature-wise transformation. In: ICLR (2020)
16. Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., Bengio, Y.: Manifold mixup: Better representations by interpolating hidden states. In: International Conference on Machine Learning. pp. 6438–6447. PMLR (2019)
17. Wang, H., Deng, Z.H.: Cross-domain few-shot classification via adversarial task augmentation. In: IJCAI (2021)
18. Zhang, C., Cai, Y., Lin, G., Shen, C.: Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (June 2020)