Supplementary for CustomKD: Customizing Large Vision Foundation for Edge Model Improvement via Knowledge Distillation

Jungsoo Lee Debasmit Das Munawar Hayat
Sungha Choi† Kyuwoong Hwang Fatih Porikli

Qualcomm AI Research‡

{jungsool, debadas, hayat, sunghac, kyuwoong, fporikli}@qti.qualcomm.com

A. Further Details of Experimental Setup

Datasets For the datasets in unsupervised domain adaptation (UDA), we use OfficeHome and DomainNet. OfficeHome contains four domains (images of art, clipart, product, real world) with 65 categories and 16,107 of images in total. While DomainNet originally includes six domains, we conduct experiments on four domains, images of clipart, real world, sketch, and painting, following the previous studies [17, 20, 23], with 345 classes and 373,061 of images in total. For the ImageNet in semi-supervised learning (SSL), we use 13000 and 128000 images during training for the evaluation settings of 1% and 10% of labeled samples, respectively, following SimMatch [33].

Implementation details Eq. 5 in the main paper indicates the entropy minimization loss, $\mathcal{L}_U$, which we apply to all feature-level KD baseline methods [18, 19, 21] (including ours) using a consistent λ_U value of 0.1 across all methods. Since prediction-level KD baseline methods [1, 8, 32] have target predictions for x_U obtained from the teacher models, we replace the entropy minimization loss with KD loss functions using those target predictions for them.

While using the default values of hyper-parameters of each baseline method, we find the lambda value for each KD loss function through a grid search of $\{0.1, 1, 10, 100\}$. For UDA, we use the lambda value of 0.1 for Soft Targets [8], 1.0 for Logits [1], 0.1 for DKD [32], 10.0 for RKD [18], 10.0 for CC [19], and 10.0 for FitNet [21]. For CustomKD, while fixing 10.0 for λ_{f_t}, we set $\lambda_{\tilde{f}_t}$ as 10.0 and 1.0 for OfficeHome and DomainNet, respectively. For SSL, we use the lambda value of 0.1 for Soft Targets [8], 0.1 for Logits [1], 0.1 for DKD [32], 0.1 for RKD [18], 1.0 for CC [19], and 100.0 for FitNet [21]. For CustomKD, we set $\lambda_{f_t} = \lambda_{\tilde{f}_t} = 100.0$ for CIFAR-100 and $\lambda_{f_t} = \lambda_{\tilde{f}_t} = 10.0$ for ImageNet. For training KD baselines in UDA, we use the learning rate of 0.1 for both datasets, and we use 60 epochs and 20 epochs for OfficeHome and DomainNet, respectively. For SSL, we use the learning rate of 0.0001 for CIFAR-100 and 0.00005 for ImageNet with 50 epochs of training for both datasets. For linear probing, we train θ_t^c for 20 epochs for OfficeHome in UDA while using 10 epochs for DomainNet and CIFAR-100. All experiments were conducted with a single GPU of RTX A5000 using less than 24GB GPU memory usage.

We fix the resolution size of the input image for teacher models to 224×224 while using the image size of edge models by following the protocol we used. This leads us to use 224×224 for UDA and ImageNet in SSL, while using 32×32 for CIFAR-100 in SSL. Regarding the computations for the number of parameters and Multiply-Accumulate Operations (MACs) in Table 5 of the main paper, we use the repository named pytorch-OpCounter [1]. Again, we compute the number of parameters and MACs by using the input size of 224×224 and 32×32 for the teacher and the student, respectively.

Regarding θ_t^h, we use a non-linear layer that consists of 1) a linear layer that transforms the embedding dimension of f_t to that of f_s, 2) a batch normalization layer, and 3) ReLU function. For θ_s^h, the input dimension and the output dimension for the first linear layer correspond to the embedding dimension of f_s and f_t, respectively, while also using the batch normalization layer and ReLU function. We remove the batch normalization layers in both θ_t^h and θ_s^h for ImageNet in SSL.

B. Pseudocodes for CustomKD

Along with the description in Sec. 3.3 of the main paper, we also explain our proposed method CustomKD through pseudocodes. Stop gradient denotes that we do not allow gradient backpropagation for the process in the correspond-

† Corresponding author. ‡ Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

[1] https://github.com/Lyken17/pytorch-OpCounter

ing line. As mentioned in the main paper, we do not modify the forward process of the edge models during the inference stage.

Algorithm 1 Pseudocodes for CustomKD

Require: Pre-trained edge model θ_s, pre-trained LVFM θ_t, labeled dataset D_L, unlabeled dataset D_U
for $epoch = 1, 2, \ldots, N$ **do**
  for all $(x, y) \in D_L$ **do** ▷ Feature customization
    Obtain $f_t = \theta_t^e(x)$ ▷ Stop gradient
    Obtain $\tilde{f}_t = \theta_t^h(f_t)$
    Compute $\mathcal{L}_t = CE(\theta_s^c(\tilde{f}_t), y)$
    Update only θ_t^h via $\nabla\mathcal{L}_t$
  end for
  for all $(x, y) \in (D_L \cup D_U)$ **do** ▷ KD
    Obtain $f_t = \theta_t^e(x)$, $\tilde{f}_t = \theta_t^h(f_t)$ ▷ Stop gradient
    Obtain $f_s = \theta_s^e(x)$, $\tilde{f}_s = \theta_s^h(f_s)$
    Compute $\mathcal{L}_{\tilde{f}_t} = ||f_s - \tilde{f}_t||^2$, $\mathcal{L}_{f_t} = ||\tilde{f}_s - f_t||^2$
    Compute $\mathcal{L}_L = CE(\theta_s(x_L), y_L)$, $\mathcal{L}_U = H(\hat{\theta}_s(x_U))$
    Compute $\mathcal{L}_s = \mathcal{L}_L + \lambda_U\mathcal{L}_U + \lambda_{f_t}\mathcal{L}_{f_t} + \lambda_{\tilde{f}_t}\mathcal{L}_{\tilde{f}_t}$
    Update θ_s^h, θ_s^c, θ_s^e via $\nabla\mathcal{L}_s$
  end for
end for

C. Further Experiments and Results

In the main paper, we could not include the experiments of SSL task using ImageNet due to the page limit. Table 1 shows that CustomKD again improves the SSL performance on ImageNet, demonstrating the scalability of CustomKD on large-scale datasets. Again, we want to emphasize again that CustomKD does not require complicated techniques such as artificial label augmentation or prediction augmentations, which are widely used in SSL studies [4, 13, 25, 31, 33]. Therefore, we believe that the superiority of CustomKD is well established considering its consistent performance gains on various datasets and the simplicity of its training process.

In Table 2, we include the results of other SSL baseline methods on CIFAR-100 that we could not report due to the space limit in Table 4 of our main paper. We also compare our method with other KD baseline methods by using the pretrained model trained without any SSL module, denoted as Source. Again, we observe that our method consistently improves the SSL performance on both edge models of AdaMatch and Source, demonstrating the wide scalability of our method regardless of the pretrained edge models. Additionally, applying our method on Source model achieves a comparable performance with our reproduced AdaMatch.

Table 3 shows that CustomKD also improves the UDA performances of other student models regardless of the backbone scale, following Table 5 of the main paper that reports the results on SSL. For the experiments, we use MobileNetV3 [9] and ShuffleNet [14] for the edge models and OpenCLIP [10] for the teacher model. As shown, additionally using $\mathcal{L}_{\tilde{f}_t}$ consistently improves the UDA performance compared to using only $\mathcal{L}_{f_t}$. Also, the performance gap between our method and using only $\mathcal{L}_{f_t}$ increases as we change the backbone of the teacher model from a small one (*i.e.*, ViT-B) to a big one (*i.e.*, ViT-L). Such a result demonstrates that it is challenging to improve the UDA performance with large backbones, while CustomKD can consistently improve it through customizing the well-generalized knowledge of teachers to a given edge model.

Due to the space limit in the main paper, only the average results are reported in Section 5. The results in Table 4, Table 5, Table 6 include the results of each domain in Table 6, Table 7(a), Table 7(b) of the main paper, respectively.

Methods	Top-1		Top-5	
	1%	10%	1%	10%
Vat + EntMin [7, 15]	-	68.8	-	88.5
S4L-Rotation [30]	-	53.4	-	83.8
UDA [27]	-	68.8	-	88.5
FixMatch [22]	-	71.5	-	89.1
CoMatch [13]	67.1	73.7	87.1	91.4
SimMatch [33]	67.2	74.4	87.1	91.6
CustomKD	**67.5**	**74.7**	**87.6**	**91.9**

Table 1. Image classification accuracy of semi-supervised learning on Imagenet.

D. Limitations

One limitation, as previously mentioned in the conclusion of the main paper, is that this paper mainly focuses on addressing the image classification task. While computer vision encompasses a plethora of tasks such as object detection or semantic segmentation, we leave them as the future work of our study. However, we believe that our approach can be modified in a task-specific manner and applied to these tasks, potentially inspiring future researchers.

Category	Methods	Labels 400	2500	10000
	DINO-V2 [16]	17.92	11.49	9.13
	Supervised* [29]	28.20	28.20	28.20
SSL	Pseudolabel [12]	87.15	59.09	38.86
	Meanteacher [24]	90.34	61.13	39.05
	Vat [15]	83.11	53.17	36.58
	MixMatch [3]	79.95	49.58	32.10
	RemixMatch [2]	57.10	34.77	26.18
	AdaMatch [4]	47.82	33.26	27.53
	FixMatch [22]	53.37	34.29	28.28
	FlexMatch [31]	50.15	33.35	27.12
	Dash [28]	53.98	34.47	27.72
	Crmatch [6]	49.39	31.35	26.24
	CoMatch [13]	60.98	37.24	28.15
	SimMatch [33]	48.82	32.54	26.42
	FreeMatch [26]	49.24	32.79	27.17
	SoftMatch [5]	49.64	33.05	27.26
Pretrained	AdaMatch* [4]	52.07	37.92	32.5
KD	Soft Target [8]	48.71	31.73	27.66
	Logits [1]	49.71	33.42	28.16
	DKD [32]	45.18	30.43	26.19
	RKD [18]	50.11	34.24	29.11
	CC [19]	49.85	33.72	28.75
	FitNet [21]	48.58	30.87	29.41
	Ours	**32.51**	**25.52**	**24.66**
Pretrained	Source* [4]	90.36	69.73	49.60
KD	Soft Target [8]	66.67	61.77	41.81
	Logits [1]	75.03	62.75	42.43
	DKD [32]	54.51	57.19	39.19
	RKD [18]	92.20	64.00	42.73
	CC [19]	91.90	63.62	41.76
	FitNet [21]	68.00	49.40	37.72
	Ours	**52.02**	**38.59**	**31.06**

Table 2. Error rates of semi-supervised learning on CIFAR-100. For the results of SSL methods, we report the mean results in Unified SSL Benchmark (USB) [25]. * indicates reproduced results using codes of USB. We compare our method with other KD baselines using edge models pretrained 1) with AdaMatch and 2) without any SSL module, denoted as Source. Supervised indicates using all labels for the entire data samples.

Teacher Type	Teacher Backbone	Teacher Accuracy*	Methods	MobileNetV3 [9] (2.54M, 0.06G)	ShuffleNet [14] (7.39M, 0.60G)
-	-	-	Source	52.70	62.16
OpenCLIP [10]	ViT-B (57.26M, 11.27G)	77.24	$\mathcal{L}_{f_t}$	56.37	66.84
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**62.76 (+6.39)**	**69.56 (+2.72)**
	ViT-L (202.05M, 51.89G)	85.55	$\mathcal{L}_{f_t}$	57.66	68.06
			$\mathcal{L}_{f_t} + \mathcal{L}_{\tilde{f}_t}$	**68.02 (+10.36)**	**72.50 (+4.44)**

Table 3. Averaged image classification accuracy on art, clipart, and product using the student model pretrained on the images of real world as the source domain in OfficeHome. The first and second numbers in the bracket of each model indicate the number of parameters and Multiply-Accumulate Operations (MACs), respectively. * indicates that we performed linear probing using only labeled samples (*i.e.*, images of source domain) for the teacher.

Metric	θ_t^c Init.	A2CA	A2P	A2RW	CA2A	CA2P	CA2RW	P2A	P2CA	P2RW	RW2A	RW2CA	RW2P	Avg.
Teacher Acc.	Random	63.96	80.92	85.17	72.23	81.12	80.26	68.65	66.87	85.31	77.75	68.82	89.3	76.70
	θ_s^c	65.59	79.95	84.51	70.29	80.15	79.66	65.68	64.72	82.76	76.02	66.76	89.64	75.48
Student Acc.↑	Random	32.71	43.59	58.30	35.15	52.89	50.91	30.00	33.33	59.65	53.36	44.81	71.07	47.15
	θ_s^c	44.77	61.03	66.79	46.77	63.35	59.97	38.77	40.99	66.90	59.66	51.48	77.16	**56.47**
CKA(f_s, f_t)↑	Random	0.29	0.50	0.47	0.36	0.55	0.47	0.34	0.42	0.49	0.42	0.50	0.59	0.45
	θ_s^c	0.53	0.60	0.56	0.43	0.63	0.53	0.41	0.50	0.57	0.49	0.55	0.69	**0.54**
CKA(f_s, $\tilde{f}_t$)↑	Random	0.31	0.50	0.49	0.34	0.51	0.42	0.33	0.41	0.46	0.44	0.49	0.58	0.44
	θ_s^c	0.65	0.72	0.69	0.47	0.65	0.57	0.48	0.57	0.64	0.61	0.65	0.76	**0.62**

Table 4. Comparisons on the initialization of the head classifier of teacher models during the feature customization stage. We report the results of each domain of Table 6 in the main paper. CKA refers to the centered kernel alignment [11].

$\mathcal{L}_L, \mathcal{L}_U$	$\mathcal{L}_{f_t}$	$\mathcal{L}_{\tilde{f}_t}$	RW2A	RW2CA	RW2P	Avg.
✓	–	–	52.62	43.96	71.91	56.16
✓	✓	–	52.49	44.93	71.53	56.32
✓	–	✓	59.99	49.67	76.75	62.14
✓	✓	✓	59.66	51.50	77.13	**62.76**

Table 5. Ablation study on our loss functions. Given that the real world images are used for the source domain, we report the results of three target domains for Table 7(a) of the main paper.

Alternating Epochs	RW2A	RW2CA	RW2P	Avg.
30:1	59.25	50.74	76.68	62.22
10:1	59.13	50.90	76.57	62.20
5:1	59.29	51.02	76.53	62.28
1:1	59.66	51.50	77.13	**62.76**

Table 6. Analysis on the frequency of the feature customization stage. Given that the real world images are used for the source domain, we report the results of three target domains for Table 7(b) of the main paper.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014. 1, 3

[2] David Berthelot, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. 2019. 3

[3] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning, 2019. 3

[4] David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alexey Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. In *ICLR*, 2022. 2, 3

[5] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. 2023. 3

[6] Yue Fan, Anna Kukleva, Dengxin Dai, and Bernt Schiele. Revisiting consistency regularization for semi-supervised learning. *International Journal of Computer Vision*, 2022. 3

[7] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2004. 2

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 1, 3

[9] Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *ICCV*, 2019. 2, 4

[10] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 2, 4

[11] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *ICML*, 2019. 4

[12] Dong-Hyun Lee. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 3

[13] Junnan Li, Caiming Xiong, and Steven C.H. Hoi. Semi-supervised learning with contrastive graph regularization. In *ICCV*, 2021. 2, 3

[14] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 2, 4

[15] Takeru Miyato, Shin ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning, 2018. 2, 3

[16] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 3

[17] Sunghyun Park, Seunghan Yang, Jaegul Choo, and Sungrack Yun. Label shift adapter for test-time adaptation under covariate and label shifts. In *ICCV*, 2023. 1

[18] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019. 1, 3

[19] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *ICCV*, 2019. 1, 3

[20] Viraj Prabhu, Shivam Khare, Deeksha Kartik, and Judy Hoffman. Sentry: Selective entropy optimization via committee consistency for unsupervised domain adaptation. In *ICCV*, 2021. 1

[21] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yann LeCun Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 1, 3

[22] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020. 2, 3

[23] Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey, 2020. 1

[24] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. 2018. 3

[25] Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark for classification. In *NeurIPS*, 2022. 2, 3

[26] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, , Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. 2023. 3

[27] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. 2019. 2

[28] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *ICML*, 2021. 3

[29] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016. 3

[30] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. 2019. 2

[31] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. 2021. 2, 3

[32] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *CVPR*, 2022. 1, 3

[33] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. In *CVPR*, 2022. 1, 2, 3