Supplementary Material of Rethinking Prototypical Contrastive Learning through Alignment, Uniformity and Correlation

Shentong Mo[1]
shentonm@andrew.cmu.edu
Zhun Sun†[2]
zhunsun@gmail.com
Chao Li[3]
chao.li@riken.jp

[1] Carnegie Mellon University
Pittsburgh, PA 15213, United States
[2] Tohoku University
Sendai, Miyagi, Japan
[3] Center for Advanced Intelligence Project (AIP), RIKEN
Tokyo, Japan

In this supplementary material, we will provide additional materials involving

1. PyTorch implementation of *Normalized Earth Mover's Distance* (NEMD);
2. PyTorch implementation of prototypical alignment, uniformity, and correlation losses;
3. detailed training configurations about hyper-parameters, datasets, and network architecture;
4. more experiments on object detection and instance segmentation;
5. more visualizations of our PAUC pre-trained representations;
6. more experimental analysis on the property of each loss.

1 Pseudo code for NEMD

Figure 1 shows the Pytorch [11] implementation for adopting Sinkhorn [5] algorithm to calculate the *Normalized Earth Mover's Distance* (NEMD), that is, $\mathcal{L}_{\text{NEMD}}$.

2 Pseudo code for each loss

We provide the PyTorch [11] implementation of our proposed alignment, uniformity, and correlation loss in Figure 2.

†Corresponding author

```
# bsz : batch size
# d   : latent dimensionality
# x   : Tensor, shape=[bsz, d]
#     latents for one side of prototypical embeddings
# y.  : Tensor, shape=[bsz, d]
#       latents for the other side of prototypical embeddings
# iter : maximum number of Sinkhorn iterations
# eps  : regularization coefficient
# redc : specifies the reduction to apply to the output

def M(self, C, u, v, eps):
    return (-C + u.unsqueeze(-1) + v.unsqueeze(-2)) / eps

def cost_matrix(x, y, p=2):
    x_col = x.unsqueeze(-2)
    y_lin = y.unsqueeze(-3)
    C = torch.sum((torch.abs(x_col - y_lin)) ** p, -1)
    return C

def nemd_sinkhorn(x, y, iter, eps, redc='sum'):
    # The Sinkhorn algorithm takes as input three variables :
    C = cost_matrix(x, y)  # Wasserstein cost function
    x_points = x.shape[-2]
    y_points = y.shape[-2]
    if x.dim() == 2:
        batch_size = 1
    else:
        batch_size = x.shape[0]

    # both marginals are fixed with equal weights
    mu = torch.empty(batch_size, x_points, dtype=torch.float,
                     requires_grad=False).fill_(1.0 / x_points).squeeze()
    nu = torch.empty(batch_size, y_points, dtype=torch.float,
                     requires_grad=False).fill_(1.0 / y_points).squeeze()

    u = torch.zeros_like(mu)
    v = torch.zeros_like(nu)
    # To check if algorithm terminates because of threshold
    # or max iterations reached
    actual_nits = 0
    # Stopping criterion
    thresh = 1e-1

    # Sinkhorn iterations
    for i in range(iter):
        u1 = u  # useful to check the update
        u = eps * (torch.log(mu+1e-8) -
            torch.logsumexp(M(C, u, v, eps), dim=-1)) + u
        v = eps * (torch.log(nu+1e-8) -
            torch.logsumexp(M(C, u, v).transpose(-2, -1), dim=-1)) + v
        err = (u - u1).abs().sum(-1).mean()

        actual_nits += 1
        if err.item() < thresh:
            break

    U, V = u, v
    # Transport plan pi = diag(a)*K*diag(b)
    pi = torch.exp(self.M(C, U, V))
    # Sinkhorn distance
    cost = torch.sum(pi * C, dim=(-2, -1))

    if redc == 'mean':
        cost = cost.mean()
    elif redc == 'sum':
        cost = cost.sum()

    return cost
```

Figure 1: PyTorch implementation of $\mathcal{L}_{\text{NEMD}}$.

```
# bsz : batch size
# d   : latent dimensionality
# x   : Tensor, shape=[bsz, d]
#       latents for one side of prototypes
# y   : Tensor, shape=[bsz, d]
#       latents for the other side of prototypes
# s   : alignment factor s
# t.  : uniformity factor t

def p_align(x, y, s=2):
    return (x - y).norm(dim=1).pow(s).mean()

def p_uniform(x, t=2):
    sq_pdist = torch.pdist(x, p=2).pow(2)
    return sq_pdist.mul(-t).exp().mean().log()

def p_corr(x, y):
    corr_dist = x * torch.log2(torch.divide(x,y))
    return corr_dist[~torch.isnan(corr_dist)].sum().mean()
```

Figure 2: PyTorch implementation of $\mathcal{L}_{\text{p-align}}$, $\mathcal{L}_{\text{p-uniform}}$ and $\mathcal{L}_{\text{p-corr}}$.

3 Detailed training configurations

ImageNet-100. Following previous methods [8, 10, 12], we perform the linear evaluation on the pre-trained representations from the global average pooling features (2048-D) of ResNet-50, where we apply a fully connected layer followed by softmax as a logistic regression classifier. We train it for 100 epochs with an initial learning rate of 10 and a weight decay of 0, using SGD with a momentum of 0.9. We use a batch size of 256.

ImageNet-1K. For ImageNet-1K, we also implement a fully connected layer followed by softmax as a linear classifier to evaluate the pre-trained representations from the global average pooling features (2048-D) of ResNet-50. Similar to state-of-the-art methods [1, 10, 14], we use SGD with a momentum of 0.9 to optimize the linear classifier for 100 epochs with a batch size of 256, where we apply an initial learning rate of 10 and a weight decay of 0.

PASCAL VOC2007. For low-shot classification, we follow PCL [10] and train a linear SVM for k-shot object classification on the pre-trained representations from the global average pooling features (2048-D) of ResNet-50. We vary the number of samples per-class $k = 1, 2, 4, 8, 16$.

MS COCO. In terms of object detection and instance segmentation, we follow the same hyper-parameters in MoCo [8], and finetune a Mask R-CNN [7] with C4 backbone on the MS COCO [15] train2017 set with $2\times$ schedule and evaluate on val2017 set.

4 More experiments

KNN Classification. Following PCL [10], we evaluate the k-nearest neighbor (kNN) classification on ImageNet. The comparison results with previous methods in terms of top-1 accuracy and the $\mathcal{L}_{\text{NEMD}}$ scores are reported in Table 2. We can observe that our PAUC performs the best compared to previous prototypical contrastive learning frameworks, which implies the high quality of the PAUC pre-trained embeddings by the lowest $\mathcal{L}_{\text{NEMD}}$ score.

Table 1: Comparison results (%) of object detection and instance segmentation fine-tuned on MS COCO, where the pre-training models are trained on ImageNet-1K. AP^b and AP^m denote the metrics for the bounding box and the mask, respectively. Bold and underline denote the first and second place.

Method	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Random Initialization	32.80	50.90	35.30	29.90	47.90	32.00
Supervised	39.70	59.50	43.30	35.90	56.60	38.60
SwAV [1]	37.60	57.60	40.30	33.10	54.20	35.10
SimSiam [3]	39.20	59.30	42.10	34.40	56.00	36.70
MoCo [8]	40.70	60.50	44.10	35.40	57.30	37.60
MoCHi [9]	39.40	59.00	42.70	34.50	55.70	36.70
MoCo v2 [4]	39.80	59.80	43.60	36.10	56.90	38.70
DenseCL [13]	40.30	59.90	<u>44.30</u>	**36.40**	57.00	**39.20**
PCL [10]	<u>41.00</u>	**60.80**	44.20	35.60	**57.40**	37.80
PAUC (ours)	**41.12**	<u>60.75</u>	**44.32**	<u>36.23</u>	<u>57.36</u>	<u>38.85</u>

Table 2: KNN classification results on ImageNet-1K.

Method	NPID	MoCo	LA	PCL	CLD	SwAV	**PAUC** (ours)
Accuracy (↑)	46.5	47.1	49.4	54.5	63.6	65.7	**67.3**
$\mathcal{L}_{\text{NEMD}}$ (↓)	0.553	0.511	0.463	0.396	0.332	0.314	**0.291**

Table 3: AMI score for k-means clustering (k = 25000) on ImageNet-1K representation.

Method	DeepCluster	MoCo	PCL	CLD	SwAV	**PAUC** (ours)
AMI (↑)	0.281	0.285	0.410	0.462	0.481	**0.522**
$\mathcal{L}_{\text{NEMD}}$ (↓)	0.501	0.487	0.289	0.253	0.234	**0.195**

Clustering Evaluation. In Table 3 we evaluate the adjusted mutual information (AMI) score between the clusterings generated by various methods and the ground-truth labels for ImageNet training data. As can be seen, our PAUC achieves a higher AMI score than previous methods, suggesting the high quality of our PAUC pre-trained representations. In the meantime, the representations with a high AMI score have a lower $\mathcal{L}_{\text{NEMD}}$ score. This further validates the rationality of the $\mathcal{L}_{\text{NEMD}}$ score in evaluating the level of the collapsing problem.

Low-shot classification. Following the setup in PCL [10], we apply fixed our PAUC pre-trained representations to train linear SVMs on the PASCAL VOC2007 [6] dataset for k-shot object classification. Specifically, we vary k, the number of samples per class, from $1, 2, 4, 8, 16$ in terms of 5 random seeds. The comparison results of mAP and standard deviation across 5 runs are reported in Table 4. As can be seen, our PAUC outperforms previous instance-wise and prototypical contrastive learning methods with larger average classification accuracy and smaller standard deviation across 5 runs for low-shot classification. This demonstrates the higher quality of prototypical representations pre-trained by our PAUC.

Object detection. Table 1 reports the comparison results for object detection on the MS COCO dataset, where the pre-trained models are optimized on the ImageNet-1K dataset. We can observe that our PAUC achieves the state-of-the-art performance in terms of AP^b and AP^b_{75}. Compared to SwAV [1], our PAUC outperforms it by a large margin in terms of three metrics, *i.e.*, $3.52, 3.15$, and 4.02. We also achieve comparable and even better performance over the general prototypical contrastive learning framework. This further implies the advantage of

Table 4: Comparison results (%) across 5 runs for low-shot classification on VOC2007 dataset.

Method	Arch.	$k=1$	$k=2$	$k=4$	$k=8$	$k=16$
MoCo [8]	ResNet-50	31.40 ± 5.12	42.00 ± 4.02	49.50 ± 3.25	60.00 ± 2.13	65.90 ± 1.05
MoCo v2 [8]	ResNet-50+MLP	32.70 ± 4.86	43.10 ± 3.78	52.50 ± 2.76	61.00 ± 1.55	67.10 ± 0.87
SimCLR [8]	ResNet-50+MLP	46.30 ± 4.35	58.30 ± 3.06	64.90 ± 2.52	72.50 ± 1.03	76.10 ± 0.58
PCL-v1 [10]	ResNet-50	46.90 ± 4.06	56.40 ± 2.65	62.80 ± 2.21	70.20 ± 0.49	74.30 ± 0.39
PCL-v2 [10]	ResNet-50+MLP	47.90 ± 4.12	59.60 ± 2.70	66.20 ± 2.17	74.50 ± 0.54	78.30 ± 0.38
PAUC (ours)	ResNet-50	$\mathbf{50.25 \pm 2.18}$	$\mathbf{61.68 \pm 2.63}$	$\mathbf{68.23 \pm 1.66}$	$\mathbf{75.72 \pm 0.32}$	$\mathbf{79.86 \pm 0.21}$

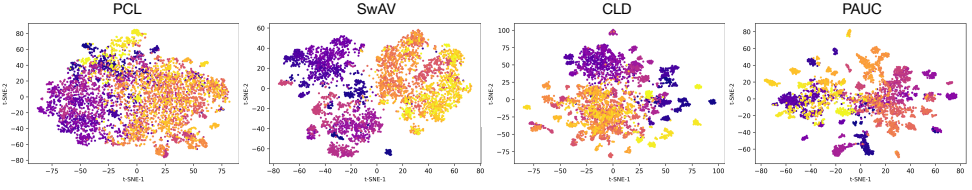

Figure 3: Visualization of pre-trained representations generated by PCL, SwAV, CLD, and PAUC (ours) from random 100 classes in the ImageNet validation set.

our PAUC pre-trained visual representations in transferring to object detection.

Instance segmentation. We also apply our PAUC pre-trained models to evaluate the instance segmentation performance on the COCO dataset, as shown in Table 1. In terms of AP^m and AP^m_{75}, we achieve better results than PCL [10], which validates the effectiveness of our PAUC in learning prototypical representations. Our PAUC also performs comparably with DenseCL [13], where a pixel-level contrastive learning framework is applied.

5 More visualizations of pre-trained representations

To evaluate the quality of prototypical representations pre-trained by our PAUC, we visualize our PAUC pre-trained embeddings on the ImageNet-100 dataset in Figure 3 by using different batch sizes. As can be seen, our PAUC pre-trained representations form more separated clusters that are distributed more uniformly on the space, compared to the pre-trained representations generated by previous prototype-based contrastive learning methods [1, 10, 14]. This further implies the advantage of our PAUC in learning prototypical representations with high quality.

6 More experimental analysis on the property of each loss

We report the experiments with different K and r on ImageNet-100 using the same configuration as in Table 5, the results are reported in the table below. It can be seen that the framework is more sensitive to the number of negative prototypes , which is important in the uniformity loss and correlation loess. On the other hand, the numbers of prototypes should not be too large in order to keep prototypes have sufficient images within them.

We also perform an extensive ablation study on how the property of each loss affects the quality of prototypical contrastive representations, as shown in Table 6. Specifically, we set the weights of alignment, uniformity, and correlation loss (α, β, γ) to $0.1, 1.0, 5.0$, the alignment factor s to $1.0, 2.0, 3.0$, and the uniformity factor t to $2.0, 3.0, 4.0$. Then we compare our

Table 5: Comparison of performance of top-1, top-5 accuracy, and $\mathcal{L}_{\text{NEMD}}$ by ablating the K and r, on ImageNet-100.

K	r	top-1 (%)	top-5 (%)	$\mathcal{L}_{\text{NEMD}}(\downarrow)$
10000, 20000, 40000	1024	83.22±0.11	96.79±0.07	0.095±0.008
5000, 10000, 20000	1024	83.91±0.08	96.83±0.06	0.088±0.006
2500, 5000, 10000	1024	**84.46±0.05**	**97.15±0.03**	**0.056±0.004**
1250, 2500, 5000	1024	84.03±0.06	96.86±0.04	0.076±0.005
2500, 5000, 10000	2048	84.12±0.05	96.93±0.03	0.069±0.005
2500, 5000, 10000	512	83.48±0.07	96.77±0.03	0.087±0.005
2500, 5000, 10000	256	82.73±0.08	96.42±0.04	0.093±0.006

Table 6: Comparison of performance of top-1, top-5 accuracy, and $\mathcal{L}_{\text{NEMD}}$ by manipulating the weights and factor of the proposed alignment, uniformity and correlation loss/factor, on ImageNet-100.

alignment($\boldsymbol{\alpha}$)	a-factor($\boldsymbol{s}$)	uniformity($\boldsymbol{\beta}$)	u-factor($\boldsymbol{t}$)	correlation($\boldsymbol{\gamma}$)	top-1 (%)	top-5 (%)	$\mathcal{L}_{\text{NEMD}}(\downarrow)$
1.0	2.0	1.0	3.0	1.0	84.46 ± 0.05	97.15 ± 0.03	0.056 ± 0.004
0.1	2.0	1.0	3.0	1.0	84.15 ± 0.08	96.97 ± 0.04	0.069 ± 0.006
5.0	2.0	1.0	3.0	1.0	84.32 ± 0.07	97.04 ± 0.03	0.063 ± 0.005
1.0	**1.0**	1.0	3.0	1.0	83.23 ± 0.07	96.75 ± 0.04	0.078 ± 0.005
1.0	**3.0**	1.0	3.0	1.0	83.61 ± 0.08	96.82 ± 0.05	0.073 ± 0.005
1.0	2.0	**0.1**	3.0	1.0	83.53 ± 0.05	96.74 ± 0.03	0.076 ± 0.005
1.0	2.0	**5.0**	3.0	1.0	83.75 ± 0.06	96.88 ± 0.03	0.071 ± 0.004
1.0	2.0	1.0	**2.0**	1.0	82.63 ± 0.04	96.22 ± 0.03	0.083 ± 0.004
1.0	2.0	1.0	**4.0**	1.0	82.55 ± 0.06	96.11 ± 0.04	0.085 ± 0.005
1.0	2.0	1.0	3.0	**0.1**	84.21 ± 0.06	97.02 ± 0.03	0.066 ± 0.005
1.0	2.0	1.0	3.0	**5.0**	84.37 ± 0.05	97.11 ± 0.03	0.059 ± 0.004

PAUC pre-trained representations in terms of top-1, top-5 accuracy of the linear classification and the $\mathcal{L}_{\text{NEMD}}$ scores to evaluate the quality of self-supervised learned prototypes.

Alignment. When varying the weight (α) of alignment loss from 0.1 to 5.0, we do not observe a large change in the results of our PAUC, which shows the robustness of our prototypical alignment loss. In the meanwhile, the variance of performance between different alignment factors s is much larger, suggesting the importance of choosing the right distance metric between positive prototypes to solve collapsing issues.

Uniformity. Our PAUC with different weights of the uniformity loss achieves a larger variance of performance than those of the alignment loss, which validates the effectiveness of the uniformity loss in learning a uniform distribution of prototypical representations on the normalized hypersphere to avoid collapsing solutions. With the increase of the uniformity factor t, the ℓ_2 distance between prototypes plays a less crucial role in the uniformity loss. As a result, the quality of our PAUC pre-trained embeddings becomes worse without much weight on the uniformity loss.

Correlation. Changing the weight of the correlation loss proposed in our PAUC has a slight impact on the top-1 and top-5 accuracy of linear classification. Moreover, the variance of the $\mathcal{L}_{\text{NEMD}}$ scores is smaller than that of the alignment loss. This further shows the stability of our PAUC to the correlation loss and the efficiency of our PAUC in increasing the discriminability of differences between prototypical representations on the hyper-sphere.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In

Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2020.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020.

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[6] Mark Everingham, Luc Van Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, pages 303–338, 2010.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[8] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020.

[9] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[10] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi. Prototypical contrastive learning of unsupervised representations. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019.

[12] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[13] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[14] Xudong Wang, Ziwei Liu, and Stella X Yu. CLD: unsupervised feature learning by cross-level instance-group discrimination. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[15] Tsung yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.