LSD-C: Linearly Separable Deep Clusters –Supplementary Material–

Sylvestre-Alvise Rebuffi* Sebastien Ehrhardt* Kai Han*
Andrea Vedaldi Andrew Zisserman
Visual Geometry Group, University of Oxford

In this supplementary material, we provide our implementation details, the confusion matrices on CIFAR 10 using our method with kNN labeling and some additional ablation studies. We also include the code to run our method on CIFAR 10 together with the network pretrained with RotNet [3].

1. Implementation details

Self-supervised pretraining. We train the RotNet [3] (i.e. predicting the rotation applied to the image among four possibilities: 0°, 90°, 180°, and 270°) on all datasets with the same configuration. Following the authors' released code, we train for 200 epochs using a step-wise learning rate starting at 0.1 which is then divided by 5 at epochs 60, 120, and 160.

Main LSD-C models. After the self-supervised pretraining step, following [4] we freeze the first three macro-blocks of the ResNet-18 [5] as the RotNet training provides robust early filters. We then train the last macro-block and the linear classifier using our clustering method. For all the experiments, we use a batch size of 256. We summarize in table 1 all the hyperparameters for the different datasets and labeling methods.

Table 1: **Hyperparameters.** Optimizer, ramp-up function and parameters of different labeling methods on different datasets.

	Optimizer				Ramp-up		Cosine	SNE		kNN
	Type	Epochs	LR steps	LR init	λ	T	τ	τ	Temp	k
CIFAR 10	SGD	220	[140, 180]	0.1	5	100	0.9	0.01	1.0	20
CIFAR 100-20	SGD	200	170	0.1	25	150	0.95	0.01	0.5	10
STL 10	SGD	200	[140, 180]	0.1	5	50	-	0.01	0.5	-
MNIST	SGD	15	-	0.1	5	50	-	-	-	10
Reuters 10K	Adam	75	-	0.001	25	100	-	-	-	5

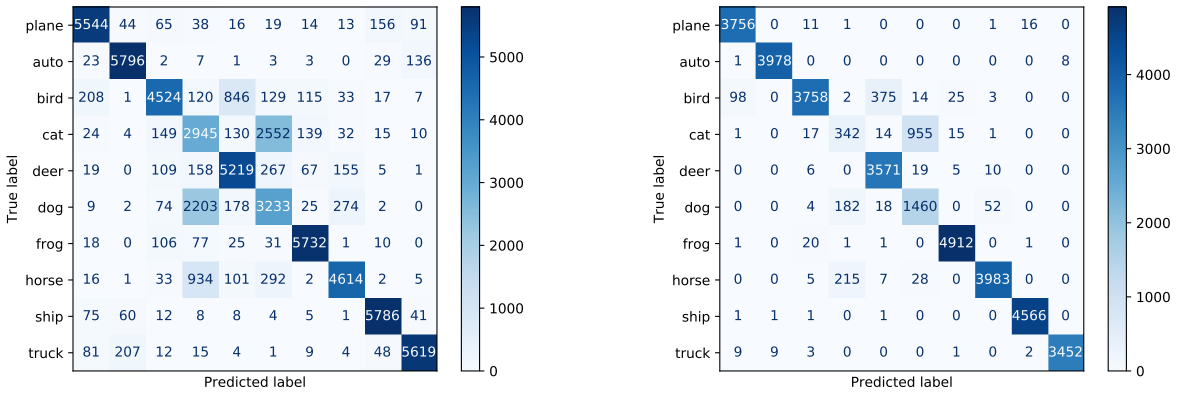
Data augmentation techniques. We showed in the main paper that data composition techniques like RICAP [8] and MixUp [9] are highly beneficial to our method. For RICAP, we follow the authors' instructions to sample the width and height of crops for each minibatch permutation by using a Beta(0.3, 0.3) distribution. Regarding MixUp, we note that using a Beta(0.3, 0.3) distribution for the mixing weight works better in our case than the Beta(1.0, 1.0) advised for CIFAR 10 in the MixUp paper. Furthermore, we have to decrease the weight decay to 10^{-4} to make MixUp work.

Miscellaneous. Our method is implemented with PyTorch 1.2.0 [7]. Our experiments were run on NVIDIA Tesla M40 GPUs and can run on a single GPU with 12 GB of RAM. Inference time for CIFAR10/CIFAR100-20/STL10 are 10.647s/10.453s/5.517s on a single GeForce GTX 1080 Ti GPU.

2. Confusion matrices on CIFAR 10

In fig. 1, we show some confusion matrices on CIFAR 10 to analyse how our clustering method performs on the different classes. We notice that there are 8 confident clusters with a very high clustering accuracy of 94.0% for confident samples. The "dog" and "cat" clusters are not well identified possibly due to a huge intra-class variation of the samples.

*Authors contributed equally



(a) All samples

(b) Confident samples

Figure 1: **Confusion matrices on CIFAR 10 using our method with kNN labeling.** Figure 1a shows that most of the errors are due to the "cat" and "dog" classes. When taking the samples with prediction above 0.9 (60% of the samples) in Figure 1b, there are less than 2000 predictions on classes "cat" and "dog" whereas there are more than 3500 for each of the other classes. Our method manages to ignore the problematic classes when taking the confident samples. Indeed, the accuracy for confident samples is 94.0%.

3. Additional experiments

In this section we report results of additional experiments we carried out. In table 2 we evaluate the impact of more components of our method. For example, we apply K-means [6] on the feature space of the pretrained RotNet model and we note very poor performance on CIFAR 10 and CIFAR 100-20. We can conclude that before training with our clustering loss, the desired clusters are not yet separated in the feature space. After training with our clustering loss, the clusters can be successfully separated. Moreover, if we only use the clustering loss and drop the consistency MSE loss, the performance decreases on both CIFAR 10 and CIFAR 100-20 by 1.5 and 1.3 points respectively, showing that the MSE provides a moderate but clear gain to our method. Finally, if we replace the linear classifier by a 2-layer classifier (i.e. this corresponds to a non-linear separation of clusters in the feature space), it results in a small improvement on CIFAR 10 but a clear decrease of 1.9 points on CIFAR 100-20. Hence using a linear classifier provides more consistent results across datasets.

Table 2: **Impact of the different losses.** From the first column, we observe that the desired clusters are not yet separated in the feature space after the RotNet pretraining. The second column shows that the MSE consistency loss provides a boost of more than 1 point to our method. Finally, we see that using a non-linear classifier harms the performance on CIFAR 100-20. All methods were trained for one seed only.

	K-means + RotNet	Ours (kNN)	Ours (kNN) w/o MSE	Ours (kNN) w/ non-lin.
CIFAR 10	14.3	81.7	80.2	82.0
CIFAR 100-20	9.1	40.5	39.2	38.6

Since we only trained the last macro block and the linear layer of a ResNet-18[5] we test the impact of increasing or decreasing the number of trainable weights in table 3. Unsurprisingly, training only the last linear layer reaches about the accuracy of k-means. However we still get reasonable performances by training from the second macro block.

We also study the impact of the self-supervised pretraining method in table 4. We note that our method has less impact when combined with self-supervised training method that are based on contrastive loss. However we note that the data augmentation we used was not the one proposed in the original paper. Hence there might be opportunities for improvements by tweaking hyper parameters or changing the data augmentation.

Finally since our method uses similarity between elements of a batch, we measure the impact of varying batch size in table 5. We observe different trends in CIFAR-10 and CIFAR100-20. For the former bigger batch sizes are beneficial while this is the opposite for CIFAR100-20. For batch size of 256 the results are stable however. The divergence is probably caused by a higher diversity of data in CIFAR100-20 which results in an unstable training signal.

Table 3: **Impact of number of trainable weights.** We test the impact of training the network with our clustering method (SNE) starting from different macro block of ResNet. We report results over two seeds (except #3 which uses 10 seeds). For fairness, we use the same hyper-parameters for each row.

From macro block	#2	#3	#4 (linear layer)
CIFAR 10	53.6±1.4	81.7±0.9	14.7±0.0
CIFAR 100-20	30.6±0.4	42.3±1.0	10.8±0.1

Table 4: **Impact of self supervised method.** We test the impact of using different self-supervised techniques on CIFAR-10 for one seed (except RotNet which used 10 seeds). All self-supervised method are trained with the same data augmentation. For fairness, all clustering methods use the same hyper-parameters for each row. We trained parameters starting from the third macro block of ResNet-18.

	RotNet[3]	SimCLR[1]	MOCO v2[2]
K-means[6]	14.3	59.45	45.48
Ours (SNE)	81.7	43.5	23.7

Table 5: **Impact of batch size.** We test the impact of training with different batch size on our clustering method (SNE). We report results over two seeds (except 256 which used 10 seeds). For fairness, we use the same hyper-parameters for each row.

Batch-size	64	128	256	512
CIFAR 10	72.4±1.3	76.9±1.5	81.7±0.9	80.9±1.2
CIFAR 100-20	46.2±0.2	42.2±0.0	42.3±1.0	31.9±1.0

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, 2020. 3

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3

[3] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018. 1, 3

[4] Kai Han, Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *Proc. ICLR*, 2020. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 1, 2

[6] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 2, 3

[7] 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 *Proc. NIPS*, 2019. 1

[8] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, 2018. 1

[9] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv*, 2017. 1