MixCo: Mix-up Contrastive Learning for Visual Representation

Sungnyun Kim*, Gihun Lee*, Sangmin Bae*

Se-Young Yun
KAIST
{ksn4397, opcrisis, bsmn0223, yunseyoung}@kaist.ac.kr

Abstract

Contrastive learning has shown remarkable results in recent self-supervised approaches for visual representation. By learning to contrast positive pairs' representation from the corresponding negatives pairs, one can train good visual representations without human annotations. This paper proposes Mix-up Contrast (**MixCo**), which extends the contrastive learning concept to *semi-positives* encoded from the mix-up of positive and negative images. MixCo aims to learn the relative similarity of representations, reflecting how much the mixed images have the original positives. We validate the efficacy of MixCo when applied to the recent self-supervised learning algorithms under the standard linear evaluation protocol on TinyImageNet, CIFAR10, and CIFAR100. In the experiments, MixCo consistently improves test accuracy. Remarkably, the improvement is more significant when the learning capacity (e.g., model size) is limited, suggesting that MixCo might be more useful in real-world scenarios. The code is available at: https://github.com/Lee-Gihun/MixCo-Mixup-Contrast.

1 Introduction

Learning representations without human annotations on data is very promising since it enables us to utilize a massive amount of data without demanding effort to label them [15]. Self-supervised learning is an approach that exploits supervised signals through pretext (*or proxy*) tasks [18]. After the remarkable successes in natural language processing [2, 9, 27], recent studies are closing the gap with supervised learning in computer vision [4, 6, 7, 11]. Nowadays, the critical component of state-of-the-art achievements in self-supervised visual representation learning is contrastive learning [26], which learns to discriminate positive pairs from negative pairs. More specifically, it learns to embed the representations of differently augmented versions of the same image (*positive pair*) to be similar, while to be dissimilar if they came from different images (*negative pair*). Such representations are highly transferable to down-stream tasks

Figure 1: Performance of MixCo on TinyImageNet (linear-evaluation) of different model sizes.

[5, 12, 19], and even outperform the representations learned by the supervised manner in some tasks [7, 35]. However, the performance of learning is highly sensitive to the choice of augmentations [5] and requires a number of negatives [5, 12].
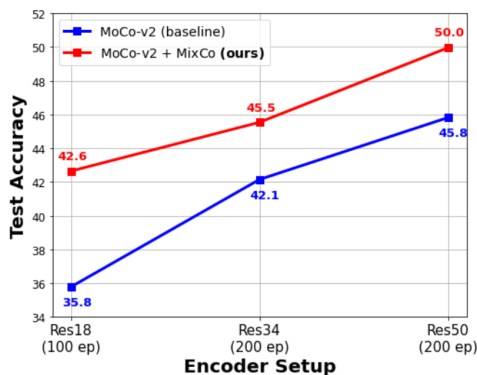
---

* Equal contribution. The names are randomly ordered.

In this paper, inspired by the success of mix-up training [34] in recent studies [29, 30, 32], we propose Mix-up Contrast (**MixCo**) as an extension of contrastive learning approach. MixCo generates the representation from a mixed image and learns how much the representation should be similar to those from the different views of each image used in the mix-up. Here, mixing images is a process of generating semi-positive pairs since the images before mixing are originally negative pairs of each other. We argue that learning through such semi-positive samples helps to learn better representations by preventing the discrimination of positive from being trivial.

Our method is simple yet effective and can be applied on-the-fly to various contrast learning based methods. To validate the efficacy of MixCo, it is applied to recent popular representation learning methods: MoCo-v2 [7] and SimCLR [5]. We show that MixCo consistently improves the linear classification results when transferred to three different datasets: CIFAR10 [21], CIFAR100 [21], and TinyImageNet [23]. Notably, the improvement is more significant when the training resource is more limited. Our contributions are: (*i*) We extended the concept of contrastive loss for visual representation learning to semi-positive samples. (*ii*) We introduced MixCo, an on-the-fly method to exploit semi-positive samples based on the popular mix-up training approach. (*iii*) We show that MixCo enables to use of the given training samples more effectively, especially when the computational budget is limited.

2   Related Work

Contrastive Learning   Although earlier self-supervised methods handcrafted pretext tasks [1, 10, 17, 20, 22, 33], most recent approaches that show remarkable performance are based on instance-level classification with contrastive loss [5, 6, 7, 12, 16, 26, 31]. The contrastive loss [26] measures the similarity of representation pairs and tries to *contrast* the positive and negative pairs. Since comparing all the possible pairs are computationally prohibitive, most methods relax the problem as discrimination to randomly sampled subsets [4]. In practice, the performance relies on careful settings on how to form negative pairs and the choice of augmentation strategies [5, 6]. To overcome the dependency of the negative sample size on batch size, MoCo [7, 12] preserved the representations of samples as a queue. SwAV [4] combined the simultaneous clustering algorithm and conducted swapped prediction to enforce consistent mapping between views.

Mix-up Training   [34] proposed mix-up training as a method to regularize the training of the network by using the samples from the vicinity of data. The vicinity samples and their labels are generated by a convex combination of samples from different classes. Mix-up training consistently improved the performance in various studies [30, 32]. Recently, [29] applied mix-up training to obtain augmented views of images, but only the input data are mixed-up instead of considering the desirable similarity. In [19], the mix-up is used to generate hard negative samples so that discriminating positive pairs become more challenging.

3   Method

Contrastive Loss   Contrastive (or InfoNCE [26]) loss tries to maximize the similarity of positive pairs and minimizes that of negative pairs:

$$\mathcal{L}_{Contrast} = -\sum_{i=1}^{n} \log \frac{\exp(\frac{v_i \cdot v_i'}{\tau})}{\sum_{j=0}^{r} \exp(\frac{v_i \cdot v_j'}{\tau})} \tag{1}$$

where n is the batch size, τ is a temperature to soften the similarity logits, and v is the normalized representation vector. Note that we denote the representation from view1 as v and from view2 as v'. Each is referred to as query and key in MoCo [7, 12]. Note that the loss function is the same with $(r+1)$-way softmax function for one positive and r number of negative classes. Although the methods based on the contrastive learning showed impressive results [6, 7, 26], learning to discriminate the positive from negatives is likely to be overfitted. Since it relies heavily on discriminating instance-level semantics, once the model can discriminate the positive from the negatives, there is not enough chance to learn from the negatives. The contrastive loss equally pushes the given representation away from that of negatives. Applying adequate augmentations helps, but as [5] suggested, such augmentations are very sensitive to the combinations.

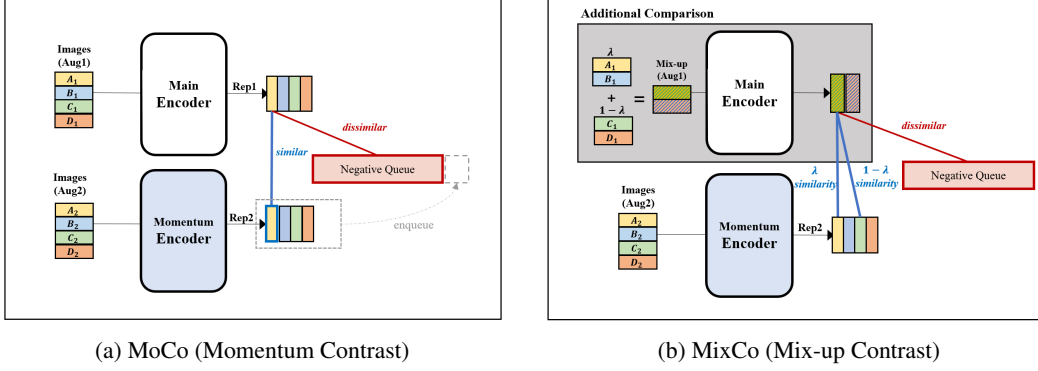(a) MoCo (Momentum Contrast) (b) MixCo (Mix-up Contrast)

Figure 2: An overview of MoCo and MixCo. (a) The representations are compared with their counterparts from the different augmentation strategy. (b) The representation from mixed-up images are *semi-positive* with the representations of the original images (different augmentation). The negatives are shared with the original MoCo queue. See Appendix A for the detailed algorithm.

Mix-up Contrast   The motivation of MixCo is exploiting softened data and targets for the contrastive learning to relieve the instance discrimination problem by letting the model learn the implicit relationship between the positives and the negatives. Similar to mix-up training, MixCo generates mixed data $x_{mix_{i,k}}$ using convex combination between data x_i and data x_k, and its representation $v_i^{mix_{i,k}}$ is defined as follows:

$$x_{mix_{i,k}} = \lambda_i \cdot x_i + (1 - \lambda_i) \cdot x_k, \qquad v_i^{mix_{i,k}} = f_{\text{encoder}}(x_{mix_{i,k}}) \tag{2}$$

where k represents any arbitrary data index in the same batch with x_i. Now we have 2 semi-positive pairs of which each corresponds to data x_i and data x_k. By letting the desired similarity to v_i' and v_k' as λ_i and $1 - \lambda_i$, respectively, we have:

$$\mathcal{L}_{MixCo} = -\sum_{i=1}^{n_{mix}} \left[\lambda_i \cdot \log \left(\underbrace{\frac{\exp\left(\frac{v_i^{mix_{i,k}} \cdot v_i'}{\tau_{mix}}\right)}{\sum_{j=0}^{r} \exp\left(\frac{v_i^{mix_{i,k}} \cdot v_j'}{\tau_{mix}}\right)}}_{\text{Similarity to } x_i} \right) + (1 - \lambda_i) \cdot \log \left(\underbrace{\frac{\exp\left(\frac{v_i^{mix_{i,k}} \cdot v_k'}{\tau_{mix}}\right)}{\sum_{j=0}^{r} \exp\left(\frac{v_i^{mix_{i,k}} \cdot v_j'}{\tau_{mix}}\right)}}_{\text{Similarity to } x_k} \right) \right] \tag{3}$$

where n_{mixco} is the number of mixed samples to be used at MixCo and τ_{mix} is softening temperature for similarity logits. Finally, the total loss is defined as a combination of $\mathcal{L}_{Contrast}$ and $\mathcal{L}_{MixCo}$ using a hyperparameter β:

$$\mathcal{L}_{total} = \mathcal{L}_{Contrast} + \beta \mathcal{L}_{MixCo} \tag{4}$$

Note that x_i and x_k are negatives (i.e., does not share the semantics) for each other. Therefore, the mixed image representation $v_i^{mix_{i,k}}$ has *semi-positive* relation to the representations v_i and v_k, which are from the original images. Learning to capture such a relation is much harder than merely discriminating against a single positive from the negatives, which leads to more efficient use of negatives. An example of MixCo application to MoCo [7, 12] is illustrated in Figure 2.

4   Experiment

4.1   Experimental setup

In unsupervised learning, we train ResNet [13] encoder with different depths on TinyImageNet. ResNet-34 and ResNet-50 encoder are trained for 200 epochs with batch size 128, and ResNet-18 is trained with 100 epochs. For MixCo hyperparameters, $\beta = 1$ and $\tau_{mix} = 0.05$. For linear evaluation, we follow the standard protocol, training a classifier for 100 epochs with initial learning rate 3.0 while freezing the pretrained encoder. Both in pretraining and linear evaluation, other unspecified hyperparameters are the same as MoCo-v2 [7], including augmentations, optimizer, and learning rate scheduler. SimCLR results are yet to be provided with thorough hyperparameter search.

4.2 Linear evaluation results

Table 1 shows the overall linear evaluation results on different pretrain setups. All the encoders are pretrained on TinyImageNet [23], and the linear classifier is trained to be transferred on different datasets. In the experiment, MixCo consistently improves on different encoder sizes. Note that the encoders do not see the CIFAR10 [21] and CIFAR100 [21] datasets in the pretraining phase. MixCo also improves the results on such datasets, suggesting that the obtained embedding encoded by MixCo is not limited to a specific dataset used at the pretraining phase. The improvement by applying MixCo is most significant ($+6.84\%$) at ResNet-18 with 100 epochs, where the smallest training resource is used. The comparison of the required training resources for pretraining is in Appendix B. Additional experiments, including ImageNet dataset and pretraining with longer epochs, are found in Appendix C.

Table 1: Linear evaluation results on TinyImageNet [23] pretrained encoders.

Architecture (epochs)	Method	MixCo (ours)	Top-1 Accuracy (%)		
			TinyImageNet	CIFAR10	CIFAR100
ResNet-18 (100)	Supervised	-	63.96	-	-
	MoCo-v2 [7]	X	35.79	71.02	48.81
		O	**42.65**	**74.25**	**53.23**
	SimCLR [5]	X	35.30	70.64	47.48
		O	**36.44**	**71.44**	**48.32**
ResNet-34 (200)	Supervised	-	64.99	-	-
	MoCo-v2 [7]	X	42.15	72.97	48.56
		O	**45.54**	**75.06**	**52.67**
ResNet-50 (200)	Supervised	-	67.03	-	-
	MoCo-v2 [7]	X	45.82	78.57	58.21
		O	**49.96**	**80.60**	**59.36**

4.3 Visualization of learned representations

The t-SNE[25] visualization of our learned representation is in Figure 3. As shown in Figure 3b, MixCo captures the semantics of data better than the baseline. We evaluate each result using Davies-Bouldin index [8] and Calinski-Harabasz index [3]. Note that in the Davies-Bouldin index, the lower value indicates the better result, whereas the higher value is better in the Calinski-Harabasz index.

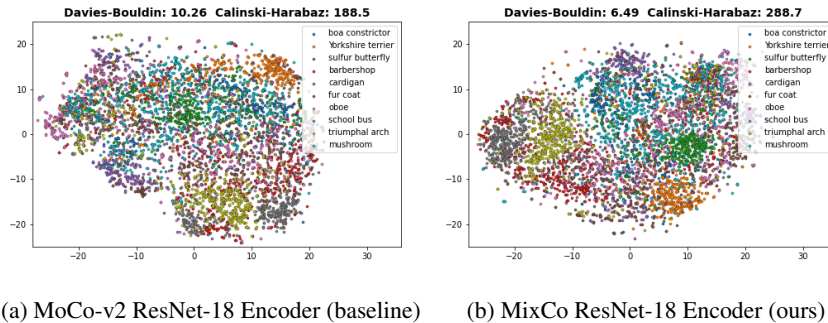(a) MoCo-v2 ResNet-18 Encoder (baseline)   (b) MixCo ResNet-18 Encoder (ours)

Figure 3: Visualization of learned train set representations. Each encoders are pretrained on TinyImageNet. 10000 training samples from 10 different classes are used. The classes are randomly chosen to have different categories (ex. terrier, coat, bus). The evaluation metrics are at the title of each figure.

5 Conclusion

We introduce MixCo, a novel method applicable to recent self-supervised learning algorithms as on-the-fly, where contrastive learning is the key component. MixCo extends the concept of contrastive learning, which discriminates positive pairs from their negatives, to semi-positive pairs generated by the mix-up. By learning to capture the relative similarities for semi-positive pairs, the representations learned by MixCo show improved performance when transferred to downstream tasks. The improvement is more significant when training resource is limited, which implies the potential usefulness in real-world situations.

Acknowledgments

References

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE international conference on computer vision*, pages 37–45, 2015.

[2] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

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

[8] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[15] Geoffrey E Hinton, Terrence Joseph Sejnowski, Tomaso A Poggio, et al. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[16] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[17] Simon Jenni and Paolo Favaro. Self-supervised feature learning by learning to spot artifacts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2733–2742, 2018.

[18] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[19] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning, 2020.

[20] Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. Learning image representations by completing damaged jigsaw puzzles. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 793–802. IEEE, 2018.

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European conference on computer vision*, pages 577–593. Springer, 2016.

[23] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7, 2015.

[24] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020.

[25] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[26] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[29] Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, and Trevor Darrell. Rethinking image mixture for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020.

[30] Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 13888–13899, 2019.

[31] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[32] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019.

[33] Xiaolong Wang, Kaiming He, and Abhinav Gupta. Transitive invariance for self-supervised visual representation learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1329–1338, 2017.

[34] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[35] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*, 2020.

[36] Benjin Zhu, Jun qiang Huang, Zeming Li, Xiangyu Zhang, and J. Sun. Eqco: Equivalent rules for self-supervised contrastive learning. *ArXiv*, abs/2010.01929, 2020.

A Algorithm

Algorithm 1 Psuedocode of MixCo on MoCo [12] in PyTorch-like style.

Input: encoder f_q, f_k, training dataset X, batch size B, feature dimension C, queue length K, momentum m, temperature τ and τ_{mix}

f_k.params = f_q.params   // initialize key encoder networks
queue = torch.randn(C, K)   // initialize a queue of K keys: [CxK]

for x in Dataloader(X):   // load a minibatch x with B samples
    x_q, x_k = $aug(x)$, $aug(x)$ // randomly augmented version for query and key

    $\lambda \sim Unif(0, 1)$
    x_q^{mix} = λ * x_q[:B/2] + (1 - λ) * x_q[B/2:]

    q, q_{mix} = f_q.forward(x_q), f_q.forward(x_q^{mix})   // queries: BxC, mixup queries: [(B/2)xC]
    k = f_k.forward(x_k).detach()   // keys: BxC, no back-propagation on f_k

    // MoCo update
    l_{pos} = torch.bmm(q.view(B,1,C), k.view(B,C,1))   // positive logits: [Bx1]
    l_{neg} = torch.mm(q.view(B,C), queue.view(C,K))   // negative logits: [BxK]

    logits = torch.cat([l_{pos}, l_{neg}], dim=1)   // logits: [Bx(1+K)]
    labels = torch.zeros(B)   // all positive logits are the 0-th, labels: [B]
    loss = torch.nn.CrossEntropyLoss(logits / τ, labels)

    // MixCo update
    l_{semi_pos} = torch.mm(q_{mix}.view(B/2,C), k.view(C,B))   // semi-positive logits: [(B/2)xB]
    l_{neg} = torch.mm(q_{mix}.view(B/2,C), queue.view(C,K))   // negative logits: [(B/2)xK]

    logits = torch.cat([l_{semi_pos}, l_{neg}], dim=1)   // logits: [(B/2)x(B+K)]
    // λ and (1-λ) for i-th and (i+B/2)-th dim, mixco labels: [(B/2)x(B+K)]
    labels = torch.cat([torch.diag(λ, 0) + torch.diag((1-λ, B/2)), torch.zeros(K)])
    loss += torch.nn.KLDivLoss(logits / τ_{mix}, labels)

    loss.backward()
    f_q.params = torch.optim.SGD(f_q.grads)   // SGD update for query network
    f_k.params = m * f_k.params + (1-m) * f_q.params   // momentum update for key network

    queue = enqueue_and_dequeue(queue, k)   // maintain the queue size by K

B Memory and GPU time usage

Architecture (epochs)	MixCo	Memory (GB/GPU)	Time (s/epoch)	Top-1 Accuracy (%)
ResNet-18 (100)	X	3.5	132.6	35.79
ResNet-18 (100)	O	4.5	153.2	42.65
ResNet-34 (200)	X	4.3	246.3	42.15
ResNet-50 (200)	X	8.1	306.8	45.82

Table 2: Required resources to train MoCo-v2 [7] encoder. Every experiment is conducted on TinyImageNet with 2 RTX 2080Ti GPUs.

In Table 2, we compare the required resources to pretrain MoCo-v2 encoder with MixCo pretraining. Using MixCo is more efficient in memory, computation, and time than vanilla MoCo-v2. MixCo requires more (mixed) inputs than MoCo, and the number of additional inputs is half the batch size according to our implementation. The increased number of inputs implies the increment of memory and the computation of logit vectors. As with the ResNet-18 encoder, this causes around $\times 1.29$ memory and $\times 1.16$ time cost. Compared with ResNet-34 encoder, ResNet-18 MixCo encoder produces similar test accuracy. However, ResNet-34 encoder requires more computation due to the

larger number of parameters. Therefore, training time is ×3.22 larger than ResNet-18 MixCo (note that ResNet-34 is trained for 200 epochs to be comparable with ResNet-18 MixCo performance).

C Additional experiments

C.1 ImageNet experiment

We compare MixCo to other methods in common ImageNet [28] experiment protocol. MixCo achieved 68.4% of linear classification accuracy, which is comparable to the state-of-the-art model. We set $\tau_{mix} = 1$.

Table 3: ImageNet linear evaluation top-1 accuracy (1-crop, 224×224), trained on features from the unsupervised pretraining. We compared several existing methods that share the similar settings: architecture, number of epochs, and batch size, since the performance highly depends on these factors.

Method	architecture	epochs	batch size	ImageNet acc. (%)
MoCo-v1 [12]	R50	200	256	60.6
PCL [24]	R50	200	256	61.5
CPC-v2 [14]	R50	200	512	63.8
SimCLR [5]	R50-MLP	200	256	61.9
MoCo-v2 [7]	R50-MLP	200	256	67.5
PCL-v2 [24]	R50-MLP	200	256	67.6
SiMo [36]	R50-MLP	200	256	**68.5**
MixCo (ours)	R50-MLP	200	256	68.4

C.2 Pretraining with longer epochs

Table 4: Linear evaluation results (%) while pretraining with different number of epochs.

Architecture	Epochs	MoCo-v2 [7]		MixCo (ours)	
		Acc@1	Acc@5	Acc@1	Acc@5
ResNet-18	100	35.79	61.16	**42.65**	**67.77**
ResNet-18	200	42.12	68.33	**45.15**	**70.25**
ResNet-18	400	45.73	72.17	**48.27**	**72.72**