Supplementary of "A Unified Mixture-View Framework for Unsupervised Representation Learning"

Xiangxiang Chu[1]
chuxiangxiang@meituan.com
Xiaohang Zhan[2]
xiaohangzhan@outlook.com
Bo Zhang[1]
zhangbo97@meituan.com

[1] Meituan
Beijing, China
[2] The Chinese University of Hong Kong
HongKong, China

1 Index of Symbols

To facilitate readability, we give a complete list of notations in Table 1.

Symbol	Definition
x_i, x_j	input sample
x'_i, x''_i	augmented sample (view)
$\mathcal{T}'$, $\mathcal{T}''$	augmentation distribution
h'_i, h''_i	feature representation
z'_i, z''_i	representation mapped to z-space
$\beta(\alpha,\alpha)$	beta distribution
λ	sampled variable from β
τ	softmax temperature
q	query in the MoCo framework
$k_+^{\lambda}, k_+^{1-\lambda}$	key of the mixed samples
k_i	key in the current queue
$f(\cdot)$, f_θ	encoder network
$g(\cdot)$, g_θ	projection head
q_θ	predictor
ℓ'_i	contrastive loss with $x_{i,j}$ as an anchor
$\mathcal{L}_{NT-Xent}(\lambda)$	loss of SimCLR-BSIM
$\mathcal{L}_q$	loss of MoCo-BSIM
$\mathcal{L}'_{\theta,\xi}$	loss of BYOL-BSIM
$\mathcal{L}_{WBSIM}$	loss of weighted BSIM

Table 1: List of symbols used throughout the paper.

2 BSIM as a General Adds-on Approach

Apart from the discussed integration with SimCLR, MoCo and BYOL, we can also simply treat BSIM as an adds-on to SIM-based methods by a weighted summation of loss functions,

$$\mathcal{L}_{WBSIM} = w_1 * \mathcal{L}_{BSIM} + w_2 * \mathcal{L}_{SIM}, \tag{1}$$

where $w_1, w_2 \in (0,1)$. We refer this approach as weighted-BSIM (**WBSIM**). When $w_1 = 0, w_2 = 1$, it is the conventional single instance multi-view approach. When $w_1 = 1, w_2 = 0$, it is BSIM. We set $w_1 = w_2 = 0.5$ throughout the paper to benefit from both SIM and BSIM.

Feature-level mixture is utilized as a regularization to perform hard example mining [14], which can boost discrimination. Other than using it as an extra augmentation, we focus on image-level mixture to define the spurious-positive examples and quantify how close two images are.

3 Experiment Details

3.1 Self-supervised Pre-training

In self-supervised pre-training, we generally follow the default settings of the competing methods for fair comparisons. We freeze the weights of ResNet50. Unless otherwise specified, all methods are trained for 200 epochs on the ImageNet dataset.

SimCLR-BSIM. We use the same set of data augmentations as [3], i.e., random cropping, resizing, flipping, color distortions, and Gaussian blur. The projection head is a 2-layer MLP that projects features into 128-dimensional latent space. We use the modified NT-Xent loss as in Equation 2 and optimize with LARS [22] with the weight decay 1e-6 and the momentum 0.9. We reduce the batch size to 256, and learning rate to 0.3 with linear warmup for first 10 epochs and a cosine decay schedule without restart.

MoCo-BSIM. We follow [12] for MoCo experiments. We first train ResNet-50 with an initial learning rate 0.03 for 200 epochs on ImageNet (about 53 hours on 8 GPUs) with a batch size of 256 using SGD with weight decay 1e-4 and momentum 0.9. For downstream tasks, the model is finetuned with BNs enabled and synchronized across GPUs. For MoCoV1, we utilize a linear neck with 128 output channels and a τ of 0.07. As for MoCoV2, we use two FC layers (2048, 2048, 128) to perform projections and a temperature coefficient of 0.2.

BYOL-BSIM. Data augmentation is the same as [10]. We follow [10] for the default hyper-parameters. Note that [10] states they prefer 300 epochs to make comparisons, we also conform to it to be consistent. Since many methods report their performance on 200 epochs, we also add an extra setting of 200 epochs to make fair comparison. To differentiate these two versions, we use 200 by default and name the BYOL300 for the former. Specifically, we also optimize LARS [22] with weight decay $1.5 \cdot 10^6$. We set the initial learning rate 3.2 and use a batch size of 4096. The target network has an exponential moving average parameter $\tau = 0.996$ and increased to 1.

SimSiam-BSIM. We use the same setting as [4], which is similar to [12]. Note that the weight decay of 0.0001 is applied for all parameter layers, including batch normalization and bias.

3.2 Downstream classification

Linear classification on ImageNet for BYOL. BYOL [10] adopts a quite different setting. To reproduce the baseline results, we train it for 90 epochs using the SGD optimizer with 0.9 momentum. Besides, we use L2 regularization with 0.0001 and a batch size of 256. The initial learning rate is 0.01 and scheduled by the 0.1 $\times$ at epoch 30 and 60.

Linear SVM classification on VOC2007. Following [9, 17, 26], we use the `res4` block (notation from [8]) of ResNet-50 as the fixed feature representations and train SVMs [1] for classification using LIBLINEAR package [7]. We train on the `trainval` split of the VOC2007 dataset [6] and report the mean Average Precision (mAP) on the `test` split by 3 independent experiments. All methods are evaluated using the same hyper-parameters as in [9].

Linear classification on ImageNet. We follow the linear classification protocol in [12] where a linear classifier is appended to frozen features for supervised training. As for MoCo, we train 100 epochs using SGD with a batch size of 256. The learning rate is initialized as 30 and scheduled by 0.1 $\times$ at epoch 30 and 60.

Low-shot classification on VOC2007. We evaluate the low-shot performance when each category contains much fewer images. Following [9, 20, 26], we use seven settings (N=1, 2, 4, 8, 16, 32, 64 and 96 positive samples), train linear SVMs on the low-shot splits, and report the test results across 3 independent experiments.

Evaluation on Semi-supervised Classification For semi-supervised training, we use the same split 1% and 10% amount of labeled ImageNet images as done in [3, 24]. We follow [3, 13, 15, 25] to finetune ResNet50's backbone on the labeled data. We train 20 epochs using SGD optimizer (0.9 momentum) with a batch size of 256. The learning rate is initialized as 0.01 and decayed by 0.2$\times$ at epoch 12 and 16.

3.3 Object Detection and Instance Segmentation

Evaluation on PASCAL VOC Object Detection Following the evaluation protocol by [12] where ResNet50-C4 (i.e., using extracted features of the 4-th stage) is used as the backbone and Faster-RCNN [18] as the detector, we benchmark our method for the object detection task on the VOC07 [6] `test` set. All models are finetuned on the `trainval` of VOC07+12 dataset for 24k iterations. We use Detectron2 [21] like MoCo did. Results are reported in Table 2, which are mean scores across five trials as [5] using the COCO suite of metrics [16]. Combined with BSIM, BYOL achieves 1.4% higher AP and 0.8% higher AP$_{50}$.

Method	Epoch	AP_{50}	AP_{75}	AP
supervised	-	81.3	58.8	53.5
SimCLR (2020)	200	79.4	55.6	51.5
SimCLR-BSIM	200	79.8	56.0	51.8
SimCLR-WBSIM	200	80.0	56.2	51.9
MoCo (2020)	200	81.5	62.6	55.9
MoCoV2 (2020)	200	82.4	63.6	57.0
MoCoV2 (2020)	200	82.5	64.0	57.4
MoCoV2-BSIM	200	82.7	64.0	57.3
MoCoV2-WBSIM	200	**83.0**	**64.2**	**57.5**
SimSiam, base (2021)	200	82.0	62.8	56.4
SimSiam, optimal (2021)	200	82.4	63.7	57.0
SimSiam-BSIM	200	82.8	64.0	57.3
SimSiam-WBSIM	200	**83.0**	**64.2**	57.4
BYOL (2020)	200	81.0	56.5	51.9
BYOL-BSIM	200	81.8	58.4	53.3
BYOL-WBSIM	200	82.0	58.5	53.5
SwAV (2020)	800	82.6	62.7	56.1

Table 2: Detection results on PASCAL VOC `trainval07+12`, which are reported across 5 trials. To make fair comparisons, the backbone is pre-trained for 200 epochs. MoCoV2-WBSIM trained 200 epochs surpasses MoCoV2 and SwAV trained for 800 epochs.

Method	Epoch	AP^b_{50}	AP^b_{75}	AP^b	AP^m_{50}	AP^m_{75}	AP^m
Supervised	-	59.9	43.1	40.0	56.5	36.9	34.7
SimCLR (2020)	200	59.1	42.9	39.6	55.9	37.1	34.6
SimCLR-BSIM	200	59.3	43.1	39.8	56.2	37.4	34.8
SimCLR-WBSIM	200	59.5	43.2	40.0	56.4	37.5	34.9
MoCo (2020)	200	60.5	44.1	40.7	57.3	37.6	35.4
MoCoV2 (2020)	200	60.1	44.0	40.6	56.9	38.0	35.3
MoCoV2-BSIM	200	60.3	44.2	40.9	57.0	38.2	35.4
MoCoV2-WBSIM	200	60.4	**44.4**	**41.1**	**57.2**	**38.3**	**35.5**
BYOL (2020)	200	60.5	43.9	40.3	56.8	37.3	35.1
BYOL-BSIM	200	60.8	44.2	40.7	57.0	37.5	35.3
BYOL-WBSIM	200	**61.0**	44.3	40.9	**57.2**	37.6	**35.5**
SwAV (2020)	800	59.8	42.0	39.1	56.2	36.1	34.2

Table 3: Object detection and instance segmentation fine-tuned results on COCO2017 dataset using $2\times$ schedule.

Evaluation on COCO Objection Detection and Instance Segmentation We also follow the evaluation protocol by [12] for the object detection and instance segmentation task on COCO2017 [16]. Specifically, we use the ResNet50-C4 Mask R-CNN framework [11] and follow $2\times$ schedule [8] as [12] since this setting can make fairer evaluations. All models are fine-tuned on the `train2017` set and evaluated on `val2017`. We report the bounding box AP and mask AP on COCO in Table 3.

3.4 Training and Memory Cost

All the experiments are done on Tesla V100 with 8 GPUs. We use a batch size of 2048 for the BYOL experiment and accumulate gradients to simulate a batch size of 4096. MoCo-BSIM adds no extra memory cost to MoCo where we simply replace query samples with mixed ones. Whereas the WBSIM version has to maintain the originally augmented query samples to compute MoCo's default loss. See details in Table 5.

	MoCo	MoCo-BSIM	MoCo-WBSIM
Memory(G)	5.5	5.5	8.2
Cost (Hour)	53	53	65

Table 4: Memory cost and training cost tested using a batch size of 256 across 8 GPUS. The training cost is calculated based on 200 epochs.

Method	Batch Size	Memory (G)	GPU Days
MoCo	256	44	17.7
MoCo-BSIM	256	44	17.7
MoCo-WBSIM	256	65.6	21.7
BYOL	4096	216	16
BYOL-BSIM	4096	216	16
BYOL-WBSIM	1024	200	28
SwAV	4096	819	33.3

Table 5: GPU resources cost. SwAV is tested on 64 V100-16G GPUs, others on 8 V100-32G GPUs. The training cost is calculated based on 200 epochs.

The sampling process. For a mini-batch of samples with size N, theoretically, we can sample λ for N times to enrich the information. Consequently, we can construct the loss by using N different weighted items. However, this process can hardly be implemented efficiently in the PyTorch framework. Instead, we make use of the 8 GPU workers and set different seeds at the beginning of training. This approach is quite efficient and possesses rich mixtures.

We also compare the performance of various methods given longer training epochs. The results are listed in Table 6. Compared with MoCoV2, MocoV2-WBSIM can further improve 0.3% top-1 accuracy on ImageNet validation dateset. Moreover, it can boost about 1 *AP* on VOC detection task.

Method	Epoch	ImageNet Acc		VOC Detection		
		Top-1	Top-5	AP_{50}	AP_{75}	AP
MoCoV2 (2020)	800	71.1	-	82.5	64.0	57.4
MoCoV2-WBSIM	800	71.4	90.4	83.4	65.0	58.3

Table 6: Linear evaluation on ImageNet and object detection on VOC using ResNet50.

4 More Discussions

Comparison with mixture-based approaches. Shen et al. [19] propose a somewhat complicated iterative mixture strategy exploiting Mixup [27] and CutMix [23] to generate a weighted mixture of samples. The mixture can be considered a weakened version of the original images which is harder to recognize, hence rendering flattened predictions. As suggested from the label-smoothing perspective, it is meant to suppress incorrect response on hard negative samples. However, image mixtures are used as-is, i.e., it learns the mixture-to-mixture similarity when combined with MoCo [12], while we learn the similarity between the mixture and its parents (forming spurious-positive pairs). This poses a fundamental difference as the loss has to be redesigned accordingly. Notice [19] also designs a too complex approach to strive for semantical harmony by decaying the 'context' image while not necessary in our case.

TSNE Visualization to showcase the working mechanism of BSIM We extend the discussions about the working mechanism of BSIM. As mentioned Sec 5 (main text), BSIM's latent space is less crowded to facilitate discrimination. To better illustrate this benefit, we pick 32 images and construct their mixtures and map their latent representations via TSNE, see Figure 2. It turns out that BSIM works like a ruler that measures how far a mixed instance should be from its parents. For instance, 6_25 (Figure 1) is distant from both 6 and 25 in SIM, while it is closer to 6 in BSIM. It is also evident that the latent space is evenly spaced in BSIM than SIM. Recall that TSNE is a dimension reducing tool, whose visualized distance is a relative measure in the latent space, not necessarily proportional to its crop size. Hence 6_25 shall not be accurately centered in between 6 and 25 although only each half of 6 and 25 are used for the mixture. For SIM, 25_6 is close to 25 but 6_25 is far from both. For BSIM, 6_25 is close to 6, but so is 25_6 to 25. This subtle difference exactly manifests the difference of the two. That is, we stretch out the latent space in terms of the relative distance of every two instances to their spurious pairs, while SIM can not.

Nevertheless, it is easier for decision making when instances are well-organized other than cluttered. The mixture near decision boundary can serve as a pivot for quantitively separating instances, which also helps scattering the representations evenly in the latent space.

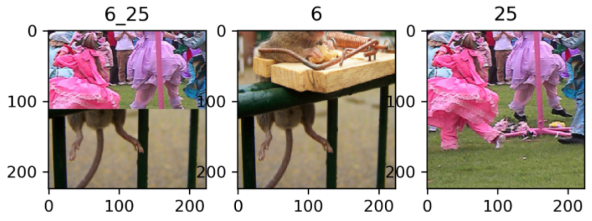

Figure 1: CutMix mixture for image 6 and 25 of Fig. 2. SIM is more sensitive to large color change.

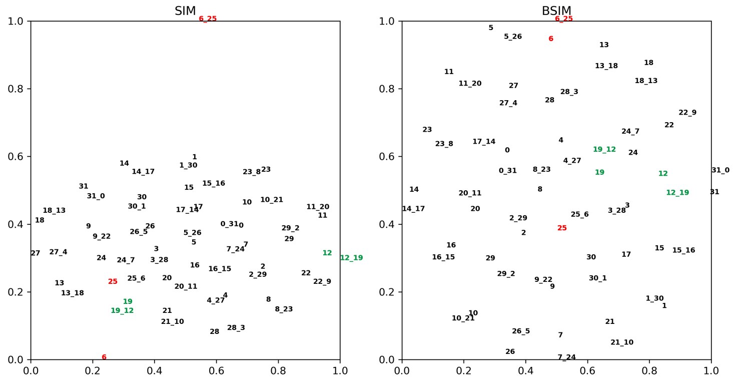

Figure 2: The representation of 2D embeddings using TSNE. We use 32 images and construct the mixture using i-th and $31-i$-th images. The mixed embedding is marked by i_31-i. Without BSIM, the embedding cannot make good use of the space (i.e. more crowded). **Left**: SIM, **Right**: BSIM. Notice SIM pushes a simple mixture 6_25 (red) distant from others, leaving a narrow space and a large amount of unused space. Besides, 12_19 and 19_12 (green) should be close but are pushed too far. In contrast, in BSIM all instances are scattered quite evenly, while mixed instances are also better placed near its parents.

5 List of Additional Figures

Figure 3 demonstrates that BSIM has better intra-class discrimination that SIM.

Figure 4 shows the difference of mixed images between Mixup and CutMix, where the latter is perceptually more natural.

Figure 5 illustrates a schematic view of latent sphere where the mixed representation is normalized on the surface.

Figure 6 manifests the schematics of of SimCLR-BSIM.

Figure 8 gives the second implementation of BYOL-BSIM.

Figure 9 depicts the beta distribution given different α, where we choose α to have a uniform distribution.

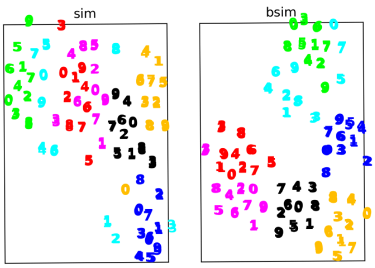

Figure 3: The representation of 2D embeddings using TSNE (sampled 3 times). We draw 10 instances (denoted by number) from each class and make 5 times of data augmentation per instance. The same class are denoted by the same color. BSIM has a better inter-class discrimination than SIM, where instances of the same class are mostly distant from those of other classes.

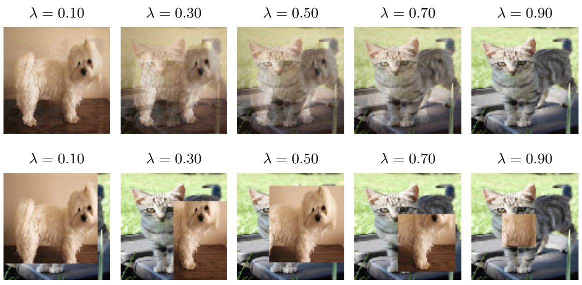

Figure 4: Comparison of Mixup and CutMix with λ as the interpolation ratio for Mixup, and cutout ratio for CutMix

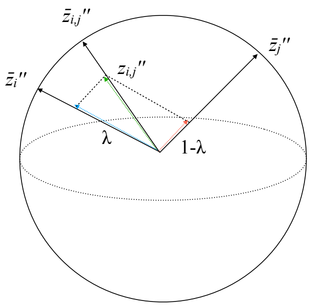

Figure 5: Schematic view of the unit ball in high-dimensional space. The target embedding of the mixed image is $\bar{z}''_{i,j}$, which is formed by normalizing $z''_{i,j}$ (dashed in green) to the sphere of the unit ball. $z''_{i,j}$ is obtained by the λ controlled linear interpolation between z''_i and z''_j.

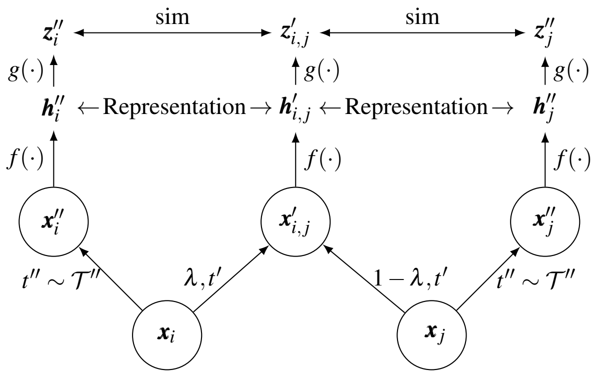

Figure 6: Adapting BSIM into SimCLR for contrastive learning. Given two images x_i and x_j (we use $j = N - i$ for speed-up within a batch of N samples), we generate a spurious-positive sample by mixing separately augmented features ($t' \sim \mathcal{T}'$) into $x'_{i,j}$, which pairs with x''_i and x''_j. SimCLR is a special case of SimCLR-BSIM when λ is 0 or 1. Note $f(\cdot)$ is an encoder network and $g(\cdot)$ refers to a projection head, both are trained with the contrastive loss in Equation 1. The representation $\boldsymbol{h}$ is later used for downstream tasks.

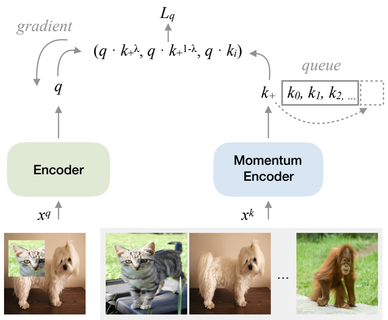

Figure 7: Applying BSIM to MoCo. Given a batch of images, we mix them in a pair-wise manner to produce x_q (only one mixture is shown for simplicity). We encode the mixed images as query q and the whole current batch as keys k. We thus generate spurious-positive pairs from q and k, and the current queue is used as negative samples.

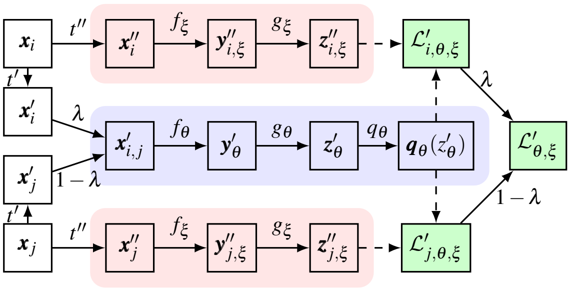

Figure 8: Applying BSIM to BYOL as in Equation 4. The above blue region is the online network, the below red one is the same target network.

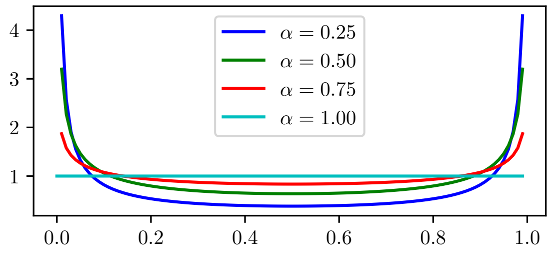

Figure 9: Probability density function of Beta distribution $\beta(\alpha, \alpha)$ under different settings. Notice when $\alpha = 1$ we have a uniform distribution.

References

[1] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33, 2020.

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

[4] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

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

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[8] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[9] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, pages 6391–6400, 2019.

[10] 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.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

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

[13] 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. *International Conference on Machine Learning*, 2019.

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

[15] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2661–2671, 2019.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[17] Andrew Owens, Jiajun Wu, Josh H McDermott, William T Freeman, and Antonio Torralba. Ambient sound provides supervision for visual learning. In *European conference on computer vision*, pages 801–816. Springer, 2016.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[19] 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.

[20] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, pages 616–634. Springer, 2016.

[21] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[22] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 6, 2017.

[23] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019.

[24] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485, 2019.

[25] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[26] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6688–6697, 2020.

[27] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.