IMPROVING MEMORY BANKS FOR UNSUPERVISED LEARNING WITH LARGE MINI-BATCH, CONSISTENCY AND HARD NEGATIVE MINING

Adrian Bulat, Enrique Sánchez-Lozano, Georgios Tzimiropoulos

Samsung AI Cambridge, Cambridge, UK

ABSTRACT

An important component of unsupervised learning by instance-based discrimination is a memory bank for storing a feature representation for each training sample in the dataset. In this paper, we introduce 3 improvements to the vanilla memory bank-based formulation which brings massive accuracy gains: (a) *Large mini-batch*: we pull multiple augmentations for each sample *within the same batch* and show that this helps the leads to better models and enhanced memory bank updates. (b) *Consistency*: we enforce the logits obtained by different augmentations of the same sample to be close without trying to enforce discrimination with respect to negative samples as proposed by previous approaches. (c) *Hard negative mining*: since instance discrimination is not meaningful for samples that are too visually similar we devise a novel nearest neighbour approach for improving the memory bank that gradually merges extremely similar data samples that were previously forced to be apart by the instance level classification loss. Overall, our approach greatly improves the vanilla memory-bank based instance discrimination and outperforms all existing methods for both seen and unseen testing categories with cosine similarity.

Index Terms— unsupervised learning, constrastive loss, memory banks

1. INTRODUCTION

Supervised learning with Deep Neural Networks has been the de facto approach for feature learning in Computer Vision over the last decade. Recently, there is a surge of interest in learning features in an unsupervised manner. This has the advantage of learning from massive amounts of unlabelled/uncurated data for feature extraction and network pre-training and is envisaged to surpass the standard approach of transfer learning from ImageNet or other large labelled datasets.

The approach we describe in this paper builds upon the widely-used framework of contrastive learning [1, 2, 3, 4, 5, 6, 7] which utilizes a contrastive loss to maximize the similarity between the representations of two different instances of the same training sample while simultaneously minimizing the similarity with the representations computed from different samples. A key point for contrastive learning is the availability of a large number of negative samples for computing the contrastive loss that are stored in a memory bank. Since the memory bank is updated rarely, this is believed to hamper training stability, hence recent methods, like [1, 3, 8] advocate online learning without a memory bank.

For this reason, the method of [1] advocates an online approach by defining the positive pair from two differently augmented versions of the same training sample and considers as negatives all other pairs from the same batch, eliminating the memory bank. As opposed to [1], we show how to train a powerful network in an unsupervised manner relying on a memory bank-based training approach. Momentum Contrast [3] maintains and updates a separate encoder for the negative samples rather than storing a memory bank in a fashion similar to the "mean teacher" [9]. More recently, SimCLR [8] emphasized the importance of composite augmentations, large batch sizes, bigger models and the use of a nonlinear projection head. They suggested that a large minibatch can replace a memory bank. In contrast, our approach employs a memory bank for contrastive learning.

Our main contribution is to show how to massively improve the vanilla memory bank approach of [2] by introducing minimal changes. We explore 2 key ideas: **(1)** What is the effect of larger batch sizes on contrastive learning with a memory bank? Concurrent work [8] has advocated the use of a large batch size for online training, i.e. *without a memory bank* as it increases the number of negative pairs. We show that a large batch size is also effective for contrastive learning *with a memory bank* (hence decoupling its positive effect from the number of negative pairs) which identifies a connection with gradient smoothing and improved memory bank updates. Furthermore, we show that if a larger mini-batch is constructed so that a set of K augmentations for each instance are used, additional consistency between the instance augmentations can be enforced to further enhance training. **(2)** Is contrastive learning effective when instances are too visually similar? Intuitively, instance discrimination is not meaningful for such cases. We show that if these samples are "merged" into the memory bank, a much more powerful network can be trained.

When reproducing the evaluation protocol of [1], we report improvements over [2] of up to $\sim 9\%$ on CIFAR-10 and of up to $\sim 10\%$ on STL-10. Furthermore, with these improvements, our method surpasses [1] and [10] by $\sim 6\%$ on CIFAR-10 and by $\sim 3\%$ on STL-10, setting for these datasets a new state-of-

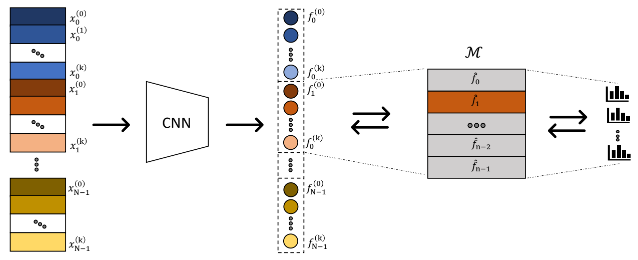


Fig. 1: Overall training process. Each instance within the batch is augmented K times and passed as input to the network, producing $N \times K$ embeddings. The final scores are produced by taking the inner product between the feature embedding $\mathbf{f}_i$ and the representations stored in the memory bank $\mathcal{M}$.

the-art. Overall, we make the following **3 contributions**:

1. We propose a **large mini-batch** for memory-bank based contrastive learning by pulling, for each sample, a set of K **augmentations within the same batch**. We show that this approach leads to stronger networks and improves the memory bank representation. (**Section 2.2**).
2. By having a set of K augmentations in our disposal, we also propose a simple **consistency** loss which enforces the logits obtained by different augmentations of the same sample to be close enough. Notably, this is achieved without trying to enforce discrimination with respect to the negative samples as proposed by previous approaches (**Section 2.3**).
3. We observe that instance discrimination is not meaningful for samples that are too visually similar. Hence, we propose a **hard negative mining** approach for improving the memory bank that gradually merges extremely visually similar data samples that were previously forced to be apart by the instance level classification loss (**Section 2.4**).

2. METHOD

2.1. Background

Given a set of n unlabelled images $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, our goal is to learn a mapping $\Phi(\mathbf{x}, \theta)$ from the data to a d-dimensional feature embedding $f(\mathbf{x}_i) = \Phi(\mathbf{x}_i, \theta) \in \mathbb{R}^d$. Typically Φ is a neural network and θ its parameters. Throughout the paper we will simply refer to the feature embedding of the i-th sample as $\mathbf{f}_i$ and assume that $\|\mathbf{f}_i\|_2 = 1$. Following [11, 2] our pretext task will consist in distinguishing the i-th instance from the rest of the samples present in the dataset (*i.e.* each data sample will be treated as a separate class). The training objective is thus formulated as minimizing the negative log-likelihood over all instances of the training set:

$$\mathcal{L}_{CE} = \log \prod_{i=1}^{n} P(i|\mathbf{x}_i) = \sum_{i=1}^{n} \log \frac{e^{\hat{\mathbf{f}}_i^T \mathbf{f}_i / \tau}}{\sum_{j=1}^{n} e^{\hat{\mathbf{f}}_j^T \mathbf{f}_i / \tau}} \qquad (1)$$

where $\hat{\mathbf{f}}_j$ is a negative sample coming from within the batch [1] or from a memory bank [2], and τ is a temperature parameter

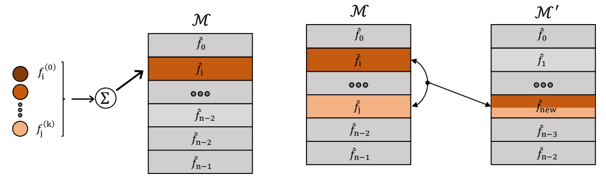
Expansion method	K		
	1	2	4
Standard	80.6	84.7	86.6
Multi-augm.		**85.0**	**87.1**

Table 1: Top-1 (%) accuracy on CIFAR-10 for different ways of increasing the batch size.

Num. feat.	K	
	2	4
1	83.9	**85.0**
K	85.3	**87.1**

Table 2: Top-1 (%) accuracy on CIFAR-10 vs. number of features used for updating the memory bank.

that controls the concentration of the parameters [2].



(a) The proposed memory bank update rule shown for a given instance i.

(b) Offline hard mining strategy. Samples with large cosine similarity are merged together.

Fig. 2: Proposed memory bank update mechanisms.

2.2. Large mini-batch with multiple augmentations

In contrastive learning, a large mini-batch can be motivated for the case of online learning (no memory bank is used) for increasing the number of negative samples. However, for the case of contrastive learning with a memory bank, the number of negative samples is fixed and independent of the batch size.

We make the observation that for the memory-bank case a large mini-batch is useful because it results in more frequent updates for a given feature $\hat{\mathbf{f}}_i$ inside the memory bank. For example if the batch-size is doubled then $\hat{\mathbf{f}}_i$ will be updated twice more frequently. As already mentioned in [2], a memory-bank approach comes at the cost of a large oscillation during training due to inconsistencies caused by updating the feature representations for different samples at very different time instances. Hence, more frequent updates of the memory bank – offered by a larger mini-batch – can help stabilize training. We consider increasing the batch size by an expansion factor of K. There are two ways to achieve this. The *standard* way is to just increase the number of samples at each iteration. All samples, in this case, are different to each other. Table 1 shows the results obtained by training a network with contrastive learning for $K = 1, 2, 4$ on CIFAR-10 using the kNN evaluation protocol. Clearly, a large batch-size results in much higher accuracy showcasing its benefit in contrastive learning. The second way to increase the batch size we explore in this work is by using *multiple* – in particular K – augmentations per sample *within the same batch*. Specifically, for every input sample $\mathbf{x}_i$ from the batch we propose to construct a series of K perturbed copies $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, \cdots, \mathbf{x}_i^{(k)}, \cdots, \mathbf{x}_i^{(K-1)}$ using a randomly composed set of augmentations $\mathcal{T}_k$. As such, the

None	ℓ_2	KL (Eq. 3)
85.0	85.6	**86.0**

Table 3: Top-1 (%) accuracy on CIFAR-10 obtained using kNN for different methods of consistency regularization for $K = 2$ augmentations.

Training	Stage	
	1	2
scratch	86.5	86.7
resume	88.4	**89.5**

Table 4: Top-1 (%) accuracy on CIFAR-10 using kNN for different stages and training strategies.

loss from one batch $\mathcal{B}$ (with size $|\mathcal{B}|$) becomes:

$$\mathcal{L}_{CE} = \sum_{i=1}^{|\mathcal{B}|}\sum_{k=1}^{K} \log \frac{e^{\hat{\mathbf{f}}_i^T \mathbf{f}_i^{(k)}/\tau}}{\sum_{j=1}^{n} e^{\hat{\mathbf{f}}_j^T \mathbf{f}_i^{(k)}/\tau}}, \qquad (2)$$

where $\mathbf{x}_i^{(k)} = \mathcal{T}_k(\mathbf{x}_i)$ is the k-th augmented copy of image $\mathbf{x}_i$ transformed using a randomly selected set of chained augmentation operators $\mathcal{T}$ (*i.e.* flipping, color jittering *etc.*) and $f_i^{(k)}$ the corresponding embedding produced by passing the sample $\mathbf{x}_i^{(k)}$ through the network. This is illustrated in Fig. 1 where different shades of the same color represent different augmentations for the same instance. This second way is primarily motivated by being able to enforce the consistency loss described in the next section. The results, shown in Table 1, confirm that by applying the proposed way even higher accuracies can be achieved.

We note that by increasing the batch size in the proposed way (*i.e.* using multiple augmentations) by K, the feature $\hat{\mathbf{f}}_i$ is actually updated after the same number of iterations, regardless the value of K, which corresponds to the same number of iterations than that of not increasing the batch size. To overcome this issue, we propose the feature $\hat{\mathbf{f}}_i$ to be updated by aggregating the features produced by the K augmented versions of $\mathbf{x}_i$: $\hat{\mathbf{f}}_i = m\hat{\mathbf{f}}_i + (1-m)\sum_{k=1}^{K}\frac{1}{K}\mathbf{f}_i^{(k)}$ (see Fig. 2a).

The latter observation allows us to further study where the accuracy improvement in Table 1 comes from. To this end, we further study the case of using K augmentations to calculate the loss of Eq. (2) but updating the memory bank *only once* (equivalent to using $K = 1$). The results for this case for $K = 2, 4$ are shown in Table 1. Interestingly, we observe a significant accuracy improvement over the baseline (no augmentation). Since the memory bank is updated in the same way as for the case $K = 1$, we conjecture that this accuracy improvement is coming from the *smoothed gradients* due to the use of the large batch size. When measured, the (average) cosine distance between the memory bank representations at adjacent epochs becomes smaller as K increases. Overall, we conclude that a large batch size helps improving both network training and updating the memory bank.

2.3. Instance consistency

With the introduction of multiple instantiations $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(k)}$ of the same sample within the batch in the previous subsection, generated by applying a different set of randomly selected transformations $\mathcal{T}_k$, herein we propose to explicitly enforce a consistent representation between the augmented representations of the same image. A similar idea has been explored for the case of semi-supervised learning [12, 13, 14], however, to our knowledge, in the context of contrastive learning, this has not explored before. Notably, this consistency is enforced without trying to enforce discrimination with respect to the negative samples as proposed by recent contrastive approaches [1, 2, 3, 4, 5, 6, 7]. More specifically, given a set of logits produced by each of the K augmented copies of i, we define our consistency loss as follows:

$$\mathcal{L}_{cons} = \sum_{k=1}^{K}\sum_{j\neq k} \mathrm{KL}\left(P(i|\mathbf{x}_i^{(k)})||P(i|\mathbf{x}_i^{(j)})\right) \qquad (3)$$

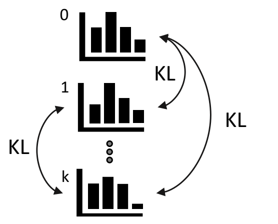


Fig. 3: KL consistency loss applied between the logits produced by the K augmented samples of a given sample i.

Note that the proposed loss term performs a dense corresponding matching (*i.e.* every possible pair formed using the K augmented samples is considered). This is illustrated in Fig. 3. For completeness, we also evaluated an ℓ_2 loss for enforcing consistency. As the results from Table 3 show, the proposed consistency loss offers noticeable improvements over the *vanilla* training process and the ℓ_2 form of regularization directly on the feature embeddings.

2.4. Hard negative mining

Unsupervised learning with instance discrimination assumes that a sample within the dataset forms a unique class. An obvious limitation of this approach is that near-identical or very similar samples are artificially forced to be apart in the embedding space. To alleviate this, we propose an offline kNN-based strategy that merges similar instances into a single class. As opposed to the deep clustering approach from [15], we do not seek to construct large clusters in an online manner via K-means nor replacing the instance-level discrimination task; instead, during an offline grouping stage, for each memory bank feature representation, we compute its nearest neighbours, and then group the ones located in its immediate σ vicinity (see Fig. 2b). This process is reminiscent of hard negative mining with the difference being that after the hard negative samples are identified they are treated as positives. Once the selected instances are merged together they will have a common representation and share the same location inside the memory bank. Similarly, during training, for the grouped instances instead of using K augmentations of the same image, we uniformly sample and augment images located within the same group. By using a small σ, the large majority of the

Table 5: Top-1 (%) acc. on CIFAR-10 obtained using kNN.

Method	kNN
Random CNN	32.1
DeepCluster (1000) [15]	67.6
Exemplar [11]	74.5
NPSoftmax [2]	80.8
NCE [2]	80.4
Triplet [1]	57.5
Triplet (Hard) [1]	78.4
Invariant Instance [1]	83.6
Ours	**89.5**

samples after grouping stage remain ungrouped (only 5-10% of samples are grouped). As such the effect of the proposed approach is to remove very similar samples from being forced to produce different features. Our proposed conservative hard mining strategy is run in an offline manner near the end of the training, each time grouping the most similar samples by means of measuring their cosine distance. Firstly, we notice that the gains flatten out after the algorithm is run for 2 times (*i.e.* denoted as **stages** in the tables). Secondly, while the method offers improvements even when the model is retrained from scratch using the computed assignments, we find the gains are significantly larger if we continue training from the current checkpoint. Table 4 summarizes results showcasing the large impact of our hard negative mining approach.

3. EXPERIMENTS

We report results for two popular settings: on *seen testing categories* (testing and training is performed on images that contain mutual categories) and *unseen testing categories* (training and testing categories are disjoint). All methods were implemented using PyTorch [16].

Seen Testing Categories. Following [2, 1] the experiments are performed on the CIFAR-10 [17] and STL-10 [18] datasets under the same settings. In particular, we use a ResNet18 [19] as a feature extractor setting the output embedding size to 128. As per [1], the network is trained for 300 epochs using a starting learning rate of 0.03, which is then dropped by 0.1 at epochs 80, 140 and 200. The network is optimized using SGD with momentum ($= 0.9$) and a weight decay of $5e-4$. During training each input sample is randomly augmented using a combination of the following transformations: Random resize and crop, random grayscale, random mirroring and color jittering. The temperature τ is set to 0.1, the memory bank momentum to 0.5 and the consistency regularization factor to $\beta = 10^5$. Following [2], we adhere to the *linear* and *kNN* protocols. As Tables 5 and 6 show, our method surpasses other methods, including our direct baseline, the method of [2] by a significant margin.

Unseen Testing Categories. Following Song *et al.* [20], we report results by training a ResNet-18 model on unseen categories on the Standford Online Product [20] dataset. The images corresponding to the first half of categories are used for training, in an unsupervised manner, without using their labels, while the testing is done on images belonging to unseen categories. We closely align our setting and training details with [1, 21]: we report results in terms of the clustering quality and NN retrieval performance. We denote with $R@k$ the probability of any correct matching to occur in the top-k retrieved [20]. NMI, the second reported metric, measures the quality of the clustering. As Table 7 shows, our method improves in terms of R@1 on top of the state-of-the-art by almost 4% and on top of our baseline from [2] by 9%.

Table 6: Top-1 (%) acc. on STL-10 using a linear and kNN classifier.

Method	# img.	Linear	kNN
Random CNN	None	-	22.4
HMP* [22]	105K	64.5	-
Satck* [23]	105K	74.3	-
Exemplar* [11]	105K	75.4	-
Invariant [1]	105K	77.9	81.6
NPSoftmax [2]	5K	62.3	66.8
NCE [2]	5K	61.9	66.3
DeepCluster (100) [15]	5K	56.6	61.2
Invariant [1]	5K	69.5	74.1
Ours	5K	71.9	77.6
Ours	105K	80.0	**84.7**

Table 7: Results (%) on Product dataset.

Method	R@1	R@10	R@100	NMI
Exemplar [11]	31.5	46.7	64.2	82.9
NCE [2]	34.4	49.0	65.2	84.1
MOM [24]	16.3	27.6	44.5	80.6
Invariant Instance [1]	39.7	54.9	71.0	84.7
Ours	**43.6**	**57.5**	**71.8**	**85.3**

4. CONCLUSION

We described three simple yet powerful ways to improve unsupervised contrastive learning with a memory bank. Firstly, we proposed a large mini-batch with multiple instance augmentations for providing smoother gradients for improving network training and increasing the quality of the features stored in the memory bank. Secondly, we introduced a simple, yet effective, intra-instance consistency loss that encourages the distribution of each augmented sample to match that of the remaining augmentations. Finally, we presented our very hard mining strategy that attempts to overcome one of the problems of unsupervised instance discrimination: that of trying to push apart near-identical images. We exhaustively evaluated the proposed improvements reporting large accuracy improvements.

5. REFERENCES

[1] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang, "Unsupervised embedding learning via invariant and spreading instance feature," in *CVPR*, 2019.

[2] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *CVPR*, 2018.

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

[4] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv*, 2018.

[5] Olivier J Hénaff, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord, "Data-efficient image recognition with contrastive predictive coding," *arXiv*, 2019.

[6] Yonglong Tian, Dilip Krishnan, and Phillip Isola, "Contrastive multiview coding," *arXiv*, 2019.

[7] Philip Bachman, R Devon Hjelm, and William Buchwalter, "Learning representations by maximizing mutual information across views," in *NeurIPS*, 2019.

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

[9] Antti Tarvainen and Harri Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.

[10] Ishan Misra and Laurens van der Maaten, "Self-supervised learning of pretext-invariant representations," *arXiv*, 2019.

[11] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *TPAMI*, 2015.

[12] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *NeurIPS*, 2016.

[13] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *TPAMI*, 2018.

[14] 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," *arXiv*, 2019.

[15] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018.

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

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

[18] Adam Coates, Andrew Ng, and Honglak Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AIStat*, 2011.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[20] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016.

[21] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh, "No fuss distance metric learning using proxies," in *ICCV*, 2017.

[22] Liefeng Bo, Xiaofeng Ren, and Dieter Fox, "Unsupervised feature learning for rgb-d based object recognition," in *Experimental robotics*, 2013.

[23] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun, "Stacked what-where auto-encoders," *arXiv*, 2015.

[24] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum, "Mining on manifolds: Metric learning without labels," in *CVPR*, 2018.