Curriculum Contrastive Learning of Image Representations via Sample Reweighting

Joel Neeser, Jérôme Kuner, Jonas Mehr and Lukas Mouton

Abstract

How should we choose negative examples for contrastive learning? Early approaches simply sample uniformly at random from the training data, while more sophisticated methods attempt to pick suitable negative examples based on their hardness. We present a novel sample reweighting scheme that allows for varying the hardness of the considered negative examples as training progresses. It assesses the hardness of the possible negative examples and assigns weights based on how close they are to the currently desired hardness. Empirically, we find that varying the difficulty of negative examples during training does not improve performance in general, while focusing exclusively on semi-hard examples does.

1 Introduction

The choice of features is crucial when building a machine learning model [1]. As such, a lot of work has focused on learning good representations/features for use on downstream tasks. In particular, the quality of the obtained representations should allow for good performance on a possibly wide range of downstream tasks, even when only using simple models.

Lately, much of the focus in representation learning has shifted towards unsupervised methods. As labelled data is often unavailable, or expensive to obtain, such methods are in high demand. Moreover, successful unsupervised representation methods also promise to be useful when labels are available. In particular, they may be employed in a pre-training phase before the actual supervised training of a model. As seen in other areas, coupling unsupervised and supervised methods in such a way often leads to improvements [2]. However, learning effective visual representations in an unsupervised manner is still a largely open problem [3].

Recently, there has been a lot of progress in so-called contrastive learning methods [4]. Using an appropriate training objective, an embedding $f(\cdot)$ is learned that should map similar (positive) pairs of data points to representations close together in latent space, and dissimilar (negative) pairs to representations further apart. Since we do not have access to labels, the choice of similarity function, loss function and contrastive pairs is crucial for successfully learning the embedding.

A successful approach in contrastive learning of image representations, SimCLR [3], uses a loss term of the form

$$\log\left(1+\sum_{x^-\in \mathrm{N}(x)}\frac{\exp(\mathrm{sim}(f(x),f(x^-))/\tau)}{\exp(\mathrm{sim}(f(x),f(x^+))/\tau)}\right) \tag{1}$$

for an anchor point x with corresponding positive pair (x, x^+). The set $\mathrm{N}(x)$ is assumed to consist of points x^- s.t. (x, x^-) is a negative pair, while τ is a temperature parameter. Moreover, $\mathrm{sim}(u, v) = u^\top v/\|u\|\|v\| \in [-1, 1]$ is the cosine similarity, which quantifies how similar two representations are based on the angle between them.

In SimCLR, a positive pair is formed by a random augmentation (e.g. crop or colour distortion) of an anchor point in a drawn batch. Another approach [5] proposes to perform “positive sample mining” instead, where one picks the nearest neighbour in the latent space to form a positive pair.

In SimCLR, given an anchor point, the set of negative images is taken to be the remainder of the drawn batch. As such, the negative pairs are chosen without considering how informative they are for the current embedding function. Thus, they may contain uninformative pairs that are very dissimilar already, or false negative pairs, where both images actually belong to the same class. SimCLR tries to combat these effects with a large batch size, such that hopefully sufficiently many informative negative pairs are included.

A more recent approach [6] proposes a sampling method for selecting hard negative samples, i.e., samples with a very similar current representation to that of the anchor. They claim that hard negative samples lead to improved representations. However, they also state that aiming for very hard negative samples could be dangerous, as this may result in lots of false negative pairs. Finally, in the contrastive learning approach [7], the model is trained using semi-hard negative samples. The authors do so claiming that too hard or too easy negative samples may lead to worse representations.

The idea of curriculum learning [8] is to gradually increase the difficulty of examples used during training from low to high. While this approach is natural for human learning, it may be generalised further by also considering other difficulty trajectories during training, e.g., less natural curricula such as starting with hard examples and gradually decreasing the difficulty, or randomly choosing the difficulty at each step.

In this work, we present an approach to combine contrastive learning with generalised curriculum learning by adapting the approach in SimCLR. The main two challenges are the following: (a) given an anchor x and a possible negative example y, we need to determine the unknown hardness of y; (b) having determined the hardness of all possible negative examples for the anchor x, we need to adapt the contrastive loss accordingly based on the currently desired hardness in the curriculum.

Our main contributions are our approaches to these two challenges. We tackle (a) by using the similarity between the current representations of x and y as a proxy for the hardness: the more similar the two representations are, the harder the negative example. We overcome (b) by introducing a weight for each term in the sum in eq. (1) that depends on the hardness value determined in (a): the closer the value to the current hardness value in the curriculum, the larger the weight.

Moreover, we empirically evaluate our approach on the dataset CIFAR-10, and comment on its effectiveness, advantages and disadvantages.

2 Methods

2.1 Curriculum Contrastive Learning

Our method extends the contrastive learning approach of SimCLR by introducing a reweighting scheme. We first explain the whole model training process and then focus in more detail on the additional components needed for curriculum learning.

Given a training image x_k from a drawn batch $\{x_k\}_{k=1}^N$ of size N, two correlated views $\tilde{x}_{2k-1}$ and $\tilde{x}_{2k}$ are obtained by augmenting it twice and then standardising using the mean and standard deviation of the training dataset. As augmentations we use random cropping, resizing, horizontal flipping, colour jitter, grey scaling and Gaussian blur. This procedure yields an augmented batch $\{\tilde{x}_k\}_{k=1}^{2N}$ of size $2N$, where $(\tilde{x}_{2k-1}, \tilde{x}_{2k})$ is a positive pair.

A neural network is used as the embedding function $f(\cdot)$ to map an augmented image $\tilde{x}_k$ to its representation $h_k = f(\tilde{x}_k)$. The obtained representation h_k is then mapped to a lower dimensional space using a linear transformation $g(\cdot)$, yielding the "projection" $z_k = g(h_k)$ [3].

For an anchor point $\tilde{x}_i$ with corresponding positive pair $(\tilde{x}_i, \tilde{x}_j)$ we use the adapted loss term

$$l_{i,j}^{(t)} = \log\left(1 + \rho_{i,j}^{(t)} \sum_{k=1}^{2N} \mathbb{1}_{[k \neq i,j]} \alpha_{i,k}^{(t)} \frac{\exp(s_{i,k}/\tau)}{\exp(s_{i,j}/\tau)}\right) \quad (2)$$

at training epoch t, where $s_{i,k} = \text{sim}(z_i, z_k)$, τ is a temperature parameter, $\alpha_{i,k}^{(t)}$ is a reweighting term and $\rho_{i,j}^{(t)}$ is a weight normalisation term. The latter two terms are used to implement the reweighting scheme and are discussed further below. In standard SimCLR they are both set to 1. The final objective to minimise is the average of the loss terms over all anchor points in the current batch, i.e.,

$$\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} (l_{2k-1,2k}^{(t)} + l_{2k,2k-1}^{(t)}). \quad (3)$$

At the end of training, the projection head $g(\cdot)$ is discarded and only the embedding $f(\cdot)$ is used to compute representations.

Note that the contrastive loss is applied not to the representations, but to their projections. This is done, as in [3] it was demonstrated empirically that applying the contrastive loss to the projections instead of the representations improves performance.

We define a curriculum via a sequence $\mu^{(t)} \in [-1, 1]$. The value at epoch t represents the current difficulty of negative pairs that the algorithm should focus on during training, with -1 being the easiest and 1 being the hardest.

A negative pair $(\tilde{x}_i, \tilde{x}_k)$ is considered to be hard, or difficult to tell apart, if the two projections z_i and z_k are similar, i.e., $\text{sim}(z_i, z_k)$ is close to 1; it is considered to be easy if the two projections are dissimilar, i.e., $\text{sim}(z_i, z_k)$ is close to -1. As such, the similarity value of a negative pair can be compared to the current curriculum value to decide whether it should be considered in the current loss.

To focus on the negative pairs with the currently desired hardness given by $\mu^{(t)}$, we use the reweighting term

$$\alpha_{i,k}^{(t)} = \exp\left(-\frac{(s_{i,k} - \mu^{(t)})^2}{\sigma^2}\right) \quad (4)$$

in eq. (2): a term corresponding to a negative pair is multiplied by a positive value between 0 and 1, depending on how close the similarity value and the current curriculum value $\mu^{(t)}$ are. The parameter σ^2 determines the range of hardness values around $\mu^{(t)}$ that should still be considered: the smaller σ^2, the faster the term decays as the similarity moves away from $\mu^{(t)}$, leading to a stricter curriculum learning approach.

Allowing $\mu^{(t)}$ to be very close to -1 may lead to a strong focus on negative pairs that are too easy and hence uninformative, while allowing it to be very close to 1 may lead to a strong focus on false negative pairs. Reducing the allowed range of $\mu^{(t)}$ to a smaller subset of $[-1, 1]$ could thus be desirable. This effect is investigated in the empirical study (Section 3).

The weight normalisation term

$$\rho_{i,j}^{(t)} = \frac{2N-2}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i,j]} \alpha_{i,k}^{(t)}} \tag{5}$$

introduced in eq. (2) attempts to keep all separate loss terms in eq. (3) on a similar scale. This guarantees that all terms have an equal participation in the total loss: Intuitively, the weighting should act as a selection mechanism for negative pairs in individual loss terms, and not as a weighting of different anchor terms in the total loss.

Note that during optimisation, the reweighting terms and the weight normalisation terms should be regarded as constant w.r.t. the weights of the model being learned. The weights of the model should only be adapted according to the weighted loss determined using the above terms; they should not be influenced by the specific form of the function used to calculate these factors, however. Therefore, we do not backpropagate through these terms in the loss.

2.2 Model Evaluation

To evaluate the learned embeddings we use the standard linear evaluation protocol [9], where a linear classifier is trained on top of the frozen trained feature extractor. In a first step, using a train/val split of the training data, a suitable weight decay value is determined. The linear classifier is then trained using the determined parameter and all of the training data until convergence. We refer to the accuracy of this final linear classifier on the unseen test set as the model score.

3 Empirical Study

In this section we describe the most relevant experiments we carried out and offer our interpretations of the results. In all experiments we use a ResNet50 network with output dimension 2048 as the embedding function, and a linear projection layer that maps the representations to a 128-dimensional space where the contrastive loss is applied. To optimise the loss, we use Adam [10] with an initial learning rate of 0.001. All our models are trained for 500 epochs in total. When determining the model score as described in section 2.2, each linear classifier is trained using Adam and an initial learning rate of 0.001 for 60 epochs. We train and test our models using the dataset CIFAR-10. All models are implemented using PyTorch Lightning.

Batch size does not have a large impact
Preliminary experiments on CIFAR-10 using the batch sizes 256, 512 and 1024 for both the standard SimCLR loss and our weighted loss showed that the batch size had no significant impact on the model score. Therefore, the remaining experiments presented in this section were carried out using a batch size of 256. Table 1 showcases some of these preliminary results.

BS	Model Score
256	0.818
512	0.817
1024	0.811

Table 1: Model scores of our SimCLR implementation on CIFAR-10 with temperature 0.1 using different batch sizes.

Temperature has a measurable effect
In order to get a baseline to compare the performance of our model with, we first tuned our implementation of SimCLR. We tested the temperature values used in [3], namely 0.05, 0.1, 0.5 and 1. The results can be found in Table 2. The best-performing hyperparameter was $\tau = 0.1$. Since the temperature is not of primary interest in our work, we also use the same temperature value for the following experiments with our own model.

τ	Model Score
0.05	0.784
0.1	0.818
0.5	0.811
1	0.756

Table 2: Model scores of our SimCLR implementation on CIFAR-10 with different temperature values using a batch size of 256.

Narrow focus on a difficulty level leads to instabilities
We tested two different σ values: 0.25 and 0.5. The models using $\sigma = 0.25$ performed significantly worse, with occasional numerical instabilities during training, manifested in a strongly non-monotone training loss. The small σ led to a very narrow focus on the current difficulty value in the curriculum, resulting in mostly very small reweighting terms. We chose to use $\sigma = 0.5$ for the remaining experiments, as a larger value would have weakened the effect of our model and made it more difficult to observe its influence. Indeed, it is easy to see that our weighted loss approaches the standard SimCLR loss as σ grows large.

Hard examples boost performance
In a series of experiments we investigated the effect of *positive curriculum learning*, where the curriculum value $\mu^{(t)}$ increases monotonically from -1 to 1 as we train. More precisely, $\mu^{(t)}$ increases from -1 to 1 in *steps* many epochs and then stays constant for the remaining epochs. Thus, by varying the *steps* parameter we can choose how quickly we want to increase the difficulty level of the negative examples that our model focuses on during training. The results obtained using different values for *steps* are displayed in Figure 1.

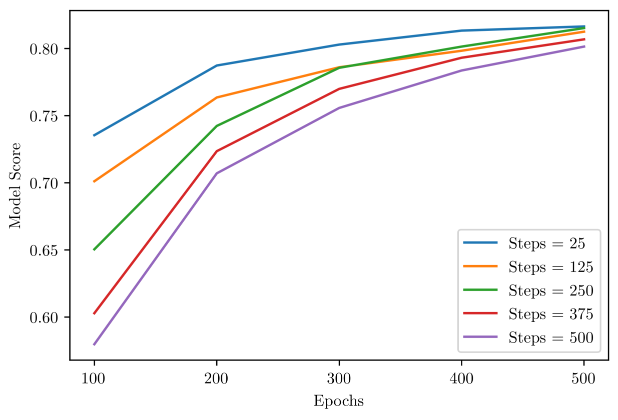

Figure 1: Model score trajectories of our model on CIFAR-10 using *positive curriculum learning* with different *steps* parameters.

Empirically, we observe that smaller *steps* parameters lead to better performance. The difference is particularly apparent when the model has been trained only for a small number of epochs. This implies that training on hard examples as soon as possible leads to better performance.

Weight normalisation is important

To gauge whether the weight normalisation presented in Section 2 has a desirable effect on the model score (MS), we compared the performance of our model with and without weight normalisation (WN) using different *step* parameters. The results are displayed in Table 3.

Steps	MS with WN	MS without WN
125	0.813	0.776
250	0.815	0.782
375	0.807	0.765

Table 3: Model scores of our model on CIFAR-10 using *positive curriculum learning* with/without weight normalisation and different *steps* parameters.

The models using weight normalisation consistently performed much better than the ones not using it. This result was expected due to the reasons given in Section 2.1. As a consequence, we chose to make use of weight normalisation in all of our subsequent experiments.

Avoiding very easy or very hard examples appears beneficial

In another series of experiments we tested the performance of other possible curricula. In particular, the strategies we implemented were the following: *negative curriculum learning*, where $\mu^{(t)}$ decreases from 1 to -1 in *steps* many epochs and then stays constant for the remaining epochs; *random curriculum learning*, where $\mu^{(t)}$ is drawn uniformly at random from $[-1, 1]$ at each epoch; and *constant curriculum learning*, where $\mu^{(t)}$ equals some desired constant difficulty value at all epochs. The comparison of the model scores for those strategies, as well as those for our implementation of SimCLR, is displayed in Figure 2.

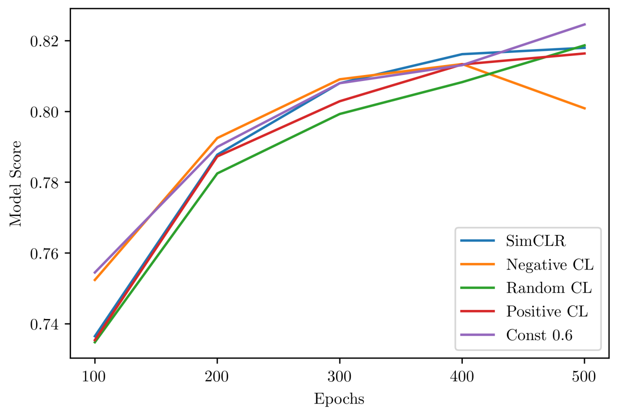

Figure 2: Model score trajectories of our model and our implementation of SimCLR on CIFAR-10 using different curricula. Note that Positive CL was trained using *steps*=25 and negative CL using *steps*=500.

While the different models behave qualitatively similarly up to training epoch 400, there are noteworthy differences in the last 100 training epochs. The scores of the constant and random curriculum model increase by a large margin, while the score of the positive curriculum model hardly increases. The score of the negative curriculum model even decreases by a lot! We note that during these last 100 epochs, the negative curriculum focuses on examples with difficulty value close to -1, i.e., very easy examples. Combining these observations with the results in Figure 1, we reach the following conclusion: Training using very easy examples for a long time at the beginning of training leads to a slower learning curve, while training using very easy examples towards the end of training can even have a detrimental effect on performance.

We note that the two best performing models (constant curriculum model and random curriculum model) either never consider very easy and very hard examples, or only with relatively small probability. This leads us to believe that difficulty values close to -1 and 1 should not be considered altogether when training: very easy examples with difficulty value close to -1 may simply not be informative for the model; very hard examples with difficulty value close to 1 may either be too difficult for the model to learn, or may actually be false negative examples, i.e., examples that belong to the same class as the anchor currently under consideration.

We also observe that after only 100 epochs of training, the constant curriculum model and the negative curriculum model perform much better than the standard SimCLR model. This advantage however diminishes as the models converge. Hence, in cases where only limited computational power is available, these models may be a better choice.

Focusing on semi-hard examples leads to best performance

In a final series of experiments, we restricted the range of $\mu^{(t)}$ to check our assumption that the exclusion of very easy and very hard examples is beneficial. In particular, we tested models using *positive curriculum learning* where the start point and end point are closer together, e.g., $\mu^{(t)}$ increases from -0.6 to 0.6 in *steps* many epochs and then stays constant. The model scores of several such experiments are shown in Table 4.

Begin	End	Model Score
-1	0.6	0.810
-0.6	0.6	0.821
-0.6	1	0.822
Const 1		0.816
Const 0.8		0.819
Const 0.6		0.825
Const 0.4		0.822

Table 4: Model scores of our model using several curricula where the range of $\mu^{(t)}$ is restricted. *Steps* is set to 250 where applicable.

The model going from -1 to 0.6 has the worst performance in this series and is the only model that considers very easy examples at the beginning, confirming the earlier results that excluding very easy examples is beneficial. The results do not paint such a clear picture for the very hard examples, though. While a constant semi-hard difficulty value of 0.6 does lead to better performance than a constant very hard difficulty value of 1, we observe almost identical performance of the model going from -0.6 to 0.6 and from the model going from -0.6 to 1. Hence, the removal of very hard examples may potentially lead to better performance, but the improvement is not so clear.

4 Discussion

Our best performing model, using a constant difficulty of 0.6, achieves a model score of 0.825 - thereby slightly improving on the best model score of 0.818 achieved by the standard SimCLR model. Based on the experiments we reach the conclusion that excluding very easy examples increases performance, while excluding very hard examples may potentially lead to improvements. However, it is important not to overinterpret these results, as we only considered a single dataset and a single type of downstream task. The obtained results would need to be validated further to obtain a clearer picture.

Furthermore, we observe that a “standard” curriculum where the difficulty starts low and increases only very slowly does not lead to improved performance. Only when the difficulty increases very quickly, or starts at an intermediate value, do we get performance scores that compete with the ones obtained by SimCLR. Overall, well-performing variants of our model achieve similar model scores to SimCLR. However, to achieve good performance with our model additional hyperparameters need to be tuned, whereas for SimCLR tuning a single hyperparameter, the temperature, suffices. Thus, for applications the use of SimCLR seems to be the better choice.

In future work one could also investigate the influence of a curriculum for the positive pairs: as positive pairs are obtained by applying two distinct augmentations to an image, one could vary the difficulty of this pair by adapting the augmentation strength over time, for instance by changing the allowed degree of rotation.

5 Summary

In this work we present a novel reweighting scheme that allows us to combine curriculum learning with contrastive learning. We use the similarity between two representations as a proxy for the hardness of a contrastive pair. The loss terms inside the classical contrastive loss are weighted appropriately to focus on training examples with difficulty levels close to the one specified by the current curriculum. Contrary to “standard” curriculum learning, our scheme also allows for more general curricula, in which the desired difficulty value to train on can be an arbitrary function of the training epoch.

Based on experiments on CIFAR-10 we argue that ignoring very easy examples always leads to better performance, while the exclusion of very hard examples usually leads to improvements. Our model performs best when trained using a constant curriculum only focusing on examples of difficulty around 80% of the maximum difficulty, leading to the conclusion that training using semi-hard examples is most beneficial. Moreover, this model slightly surpasses the performance of our baseline given by SimCLR.

Overall, our model performs similarly to SimCLR. Nevertheless, the introduced weighting scheme for curriculum learning could potentially be used in other models and with other loss functions, and is therefore of value on its own.

References

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: `10.1109/TPAMI.2013.50`.

[2] Dumitru Erhan et al. "Why Does Unsupervised Pre-training Help Deep Learning?" In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 201–208.

[3] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607. URL: `https://proceedings.mlr.press/v119/chen20j.html`.

[4] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: `10.1109/CVPR.2006.100`.

[5] Debidatta Dwibedi et al. "With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations". In: *arXiv:2104.14548 [cs]* (Oct. 2021). arXiv: 2104.14548. URL: `http://arxiv.org/abs/2104.14548`.

[6] Joshua Robinson et al. *Contrastive Learning with Hard Negative Samples.* 2021. arXiv: `2010.04592 [cs.LG]`.

[7] Sangryul Jeon et al. "Mining Better Samples for Contrastive Learning of Temporal Correspondence". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2021, pp. 1034–1044.

[8] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: `10.1145/1553374.1553380`. URL: `https://doi.org/10.1145/1553374.1553380`.

[9] Richard Zhang, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization". In: *Computer Vision – ECCV 2016.* Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 649–666. ISBN: 978-3-319-46487-9.

[10] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015).