Preventing Dimensional Collapse in Contrastive Local Learning with Subsampling
Louis Fournier, Adeetya Patel, Michael Eickenberg, Edouard Oyallon, Eugene Belilovsky

To cite this version:
Louis Fournier, Adeetya Patel, Michael Eickenberg, Edouard Oyallon, Eugene Belilovsky. Preventing Dimensional Collapse in Contrastive Local Learning with Subsampling. ICML 2023 Workshop on Localized Learning (LLW), Jul 2023, Honolulu (Hawaii), USA, United States. hal-04156218

HAL Id: hal-04156218
https://hal.science/hal-04156218
Submitted on 20 Jul 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Preventing Dimensional Collapse in Contrastive Local Learning with Subsampling

Louis Fournier 1  Adeetya Patel 2  Michael Eickenberg 3  Edouard Oyallon 1  Eugene Belilovsky 2

Abstract
This paper presents an investigation of the challenges of training Deep Neural Networks (DNNs) via self-supervised objectives, using local learning as a parallelizable alternative to traditional backpropagation. In our approach, DNN are segmented into distinct blocks, each updated independently via gradients provided by small local auxiliary Neural Networks (NNs). Despite the evident computational benefits, extensive splits often result in performance degradation. Through analysis of a synthetic example, we identify a layer-wise dimensional collapse as a major factor behind such performance losses. To counter this, we propose a novel and straightforward sampling strategy based on blockwise feature-similarity, explicitly designed to evade such dimensional collapse.

1. Introduction
Training a DNN via backpropagation is a computationally expensive process, necessitating the sequential and synchronous processing of layers while storing intermediary computations in memory (Jaderberg et al., 2017). A promising alternative is local learning, where a NN is divided into smaller blocks updated in parallel via local gradient estimates from small auxiliary NNs, allowing efficient parallelization of compute and limited memory use. Nonetheless, larger splits in supervised settings often result in a more significant accuracy gap with End-to-End training (Belilovsky et al., 2021) which is often attributed to information loss, where the auxiliary NN may concentrate only on features relevant to its specific task, inadvertently allowing other potentially useful features for subsequent layers to dissipate.

This phenomenon seems also present in unsupervised settings as very deep NNs are usually split in a limited number of blocks (only 4 in Löwe et al. (2019); Siddiqui et al. (2023)), limiting their parallelization potential. In this paper, we consider the challenge of dividing a NN trained via self-supervision into a larger split while maintaining competitive final accuracy. We focus on approaching the contrastive learning SimCLR framework (Chen et al., 2020), a leading contrastive learning method, through the lens of local learning, owing to its simplicity and widespread use.

Our contributions are as follows: (1) We identify a dimensional collapse phenomenon caused by local self-supervised learning resulting in an undesirable information loss. (2) Motivated by a synthetic experiment, we propose a simple feature-similarity-based sampling method that prevents this collapse in local learning settings, reducing information degradation. (3) Our experiments conducted on the CIFAR-10, Fashion-MNIST and STL-10 datasets validate the effectiveness of our method. (4) We carry out ablation experiments on the CIFAR-10 dataset to emphasize the improvement achieved through our method, notably over the dimensional collapse.

Our code is available at: https://github.com/fournierlouis/subsampled_local_simclr.

2. Related Work
Contrastive self-supervised learning  Contrastive methods for self-supervised learning have made significant strides in recent years. They learn representations by contrasting positive (similar) and negative (dissimilar) examples. Notably, MoCo (He et al., 2019) uses a dynamic dictionary to store features, while the SimCLR (Chen et al., 2020) framework on which we focus this work leverages data augmentation strategies to produce positive examples. However, Jing et al. (2022) noted that contrastive training leads to dimensional collapse without proper projectors, a similar phenomenon to the one we find in local learning.

Supervised local learning  Local learning methods have been extensively explored (Nøkland & Eidnes, 2019; Belilovsky et al., 2021; 2019; Gomez et al., 2022; Ren et al., 2022) to eliminate computational locks inherent in
Various data sampling strategies have been proposed to im-
with few splits. Xiong et al. (2020) expanded on this idea but
violated the decoupling principle. Illing et al. (2021) also
expanded on the idea with a Hebbian learning method, and
Siddiqui et al. (2023) proposed the Barlow Twins loss for
local decoupled learning on the ImageNet dataset. However,
both showed a similar accuracy loss due to decoupling.

**Self-supervised local learning** Löhke et al. (2019) pro-
pose a self-supervised method based on mutual information
criteria for predicting patches from the same image, albeit
with few splits. Xiong et al. (2020) expanded on this idea but
violated the decoupling principle. Illing et al. (2021) also
expanded on the idea with a Hebbian learning method, and
Siddiqui et al. (2023) proposed the Barlow Twins loss for
local decoupled learning on the ImageNet dataset. However,
both showed a similar accuracy loss due to decoupling.

**Data subsampling methods in self-supervised learning**
Various data sampling strategies have been proposed to im-
prove training in the SimCLR framework. In particular,
hard negative sampling prioritizes challenging negative ex-
amples (Robinson et al., 2021), but is mainly designed for
metric learning. Some like Tabassum et al. (2022) refine
the method with more intricate reweighting schemes. Our
method, considering only high feature similarity examples,
is related to these approaches; but does not require addi-
tional computation for importance scores, and improves
specifically local learning.

3. Method

3.1. Framework: Decoupled SimCLR

We adapt the typical SimCLR pipeline to a decoupled NN
composed of \( K \) blocks, following Belilovsky et al. (2021).
Data augmentation is applied to a batch of size \( N \), gen-
erating \( 2N \) augmented data examples. Each pair of data-
augmented examples is considered a positive example; and
all other possible pairs are considered negative examples.
This results in \( N \) positive and \( 2N(N - 1) \) negative pairs.

Each block of the decoupled network, a small network \( f^k \),
forward-propagates features to the next block \( f^{k+1} \) without
receiving gradients from it; which gives the intermediate
representations \( h_i^k = f^k(h_i^{k-1}) \) (with the sample \( x_i = h_i^0 \)).
Then, the \( K \) small projector head networks \( g^k \) project the
representations to \( z_i^k = g^k(h_i^k) \). For unlocking backward
and update steps (see Jaderberg et al. (2017)), each pair
\( \{f^k, g^k\} \) is updated locally via backpropagation follow-
ing the local loss of the block. With the similarity score
\( \text{sim}(x_i, x_j) \) (here the cosine similarity) associated with
each pair of samples and \( \tau \) a temperature parameter, this loss is
the decoupled SimCLR loss:

\[
\ell_{ij}^k = -\log \frac{\exp(\text{sim}(z_i^k, z_j^k)/\tau)}{\sum_{l=1}^{2N} \delta_{l\neq i} \exp(\text{sim}(z_i^l, z_j^k)/\tau)}.
\] (1)

This decoupled training procedure, as it stands, is unable to
prevent collapse of the features in the intermediate levels, which we motivate with the following
linear model.

3.2. Preventing dimensional collapse in a linear model
with subsampling

**Linear model framework** We consider the framework
presented by Jing et al. (2022) which simplifies the con-
trastive framework by implementing a linear model \( W \). We
consider a dataset composed of two views of \( N \) data points \( (x_i)_1, (x_i)_2 \), of dimension \( D \), \( x \) an augmented dataset vector and \( z = Wx \) the representation vector. Then,
the dynamic of the weight matrix under gradient descent
following the contrastive InfoNCE loss is, with similarity
terms \( s_{ij} = \frac{1}{\tau^2} e^{-\frac{1}{2} |z_i - z_j|^2} \), \( s_{ii} = e^{-\frac{1}{2} |z_i - z_i|^2} \) and
\( Z_i = \sum_{j \neq i} e^{-\frac{1}{2} |z_i - z_j|^2} + e^{-\frac{1}{2} |z_i - z_i|^2} \):

\[
\frac{d}{dt} W = \sum_{i,j \neq i} s_{ij} (z_i - z_j)(x_i - x_j)^T - \sum_t \left( 1 - s_{ii} \right) (z_t^* - z_i)(x_t^* - x_i)^T
\]

(2)

For fixed values \( s_{ij} \), Jing et al. (2022) finds that strong
data augmentation results in collapsed dimensions in \( W \)
due to negative eigenvalues in \( \frac{d}{dt} W \) caused by the data
augmentation distribution term. Indeed, at convergence,
\( \frac{d}{dt} W = 0 \). For the sake of simplicity, if two data-augmented
samples should lead to the same exact representation, one
has that \( s_{ii} = 1 \) and thus \( s_{ij} = 0 \).

Following Jing et al. (2022), if data augmentation is only
applied at certain features \([d + 1 : D]\), then the highest
rank matrix \( W \) which allows convergence is proportional to
\( \begin{pmatrix} \mathbb{I}_d & 0 \\ 0 & 0 \end{pmatrix} \), and the augmented feature dimensions need to
be collapsed.

**Subsampling to prevent collapse** Now, consider that we restrict both sums to only consider examples with high simi-
with our subsampling when increasing the threshold $T$ we test it on the toy model of the linear framework of Jing et al. (2022), with standard normal noise data of dimension $D = 16$, augmented with standard normal noise with amplitude $\sigma$ on the 8 last dimensions. We train our method following Eq. (5) with gradient descent. Compared to the gradient dynamic in Eq. (2), we use normalized representations. We report in Fig. 1a the singular values of the covariance matrix of the representations $z$. Standard SimCLR training produces collapsed dimensions, which disappear with our subsampling when increasing the threshold $T$ as predicted.

### 4. Numerical experiments

**Architecture** We consider a ResNet-50 (He et al., 2016) base encoder, which we divide into 4, 8 or 16 decoupled blocks. Each local projector $g_k$ is composed of a convolution and a multilayer perceptron. More details are given in the appendix.

**Hyper-parameters and datasets** We train the network on three image classification datasets: CIFAR-10 (Krizhevsky et al.), Fashion-MNIST (Xiao et al., 2017) and STL-10 (Coates et al., 2011). Since Fashion-MNIST is an easier dataset than the others, we use a ResNet-18 model rather than a ResNet-50, and thus decouple only up to 8 blocks. Each block is trained using either the training objective Eq. (1) or (5). Following standard practice (Chakraborty et al., 2020), our model is trained with the Adam optimizer (Kingma & Ba, 2014) with a learning rate of $10^{-3}$ and weight decay of $10^{-6}$ for 1000 epochs, with batch size 256, and temperature is kept at the default $\tau = 0.5$. We choose the threshold $T$ following the training accuracy in linear evaluation: $T = 0$ for CIFAR, $T = 0.3$ for Fashion-MNIST and $T = -0.4$ for STL-10.

#### 4.1. Accuracy on image recognition tasks

**Linear evaluation** We evaluate the model train with a standard linear evaluation on the supervised image classification task. The model is frozen, and the representations at the end of the model are used to train a linear classifier. The layer is trained for 200 epochs with the cross-entropy loss, with the same optimizer and batch size as the self-supervised training. We use the STL-10 5k labeled training samples for training, like Löwe et al. (2019).

**Classification results** Tab. 1 reports testing accuracies on standard image datasets, comparing our method with the standard SimCLR loss, for splits in $K = 1, 4, 8$ and 16 blocks. As expected and previously noticed for supervised settings, the testing accuracy degrades as the number of splits increases. Introducing our data-sampling technique of Eq. (5) does not affect significantly our model if split in $K = 1$ or 4 blocks, since information loss is limited. The accuracy on STL-10 improves meaningfully only for $K = 16$. In the case of CIFAR-10, our sampling strategy allows us to improve the accuracy of $K = 16$ close to that of $K = 8$ with no subsampling, a significant improvement. We also show the high parallelization potential of our method, by providing the memory cost of our method as implemented, as well as the MACs value of the slowest block among the $K$, which is a lower bound MACs for a fully parallel model. We also confirm our findings for the easier Fashion-MNIST dataset in Tab. 2 for $T = 0.3$ on a ResNet-18. We find a similar improvement due to our subsampling method for $K = 8$, and a decrease for the end-to-end model.

#### 4.2. Analysis of the internal representations.

**Dimensional collapse** We now investigate the extent of dimensional collapse in the representations generated by different methods. To do so, we consider the (average pooled) intermediate representations $(h^k_l)_l$ after training on CIFAR-10, from which we compute the covariance matrix and its...
sensitive value decomposition. We obtain the intrinsic dimensionality of the features, a linear proxy for their information content. Fig. 1b displays the spectrum for a ResNet-50 trained in various settings. The internal representations of the decoupled model trained on SimCLR suffers progressively from a dimensionality collapse, which is consistent with the findings of Wang et al. (2021) that found a drop in mutual information between the representations and both the labels and the input with depth.

However, we argue that our method prevents this dimensionality loss. We observe in the same figure the spectrum with and without subsampling. Despite no differences in the first layers, there is a significant increase in later layers both in the decoupled network and surprisingly in the End-to-End network. Notably, in the last layer, the dimensionality of the decoupled model matches the SimCLR-trained End-to-End model. Since the End-to-End model does not suffer from information loss, the dimensionality does not however improve its accuracy, compared to the decoupled model.

**Linear probes** To further study the effect of our subsampling, particularly relating to depth, we compute linear evaluation on the intermediate representations $h^k$ with $K$ linear probes. We report in Fig. 2 the accuracy of the linear probes after training for a model trained End-to-End with SimCLR, and a decoupled model ($K = 16$) trained with and without subsampling. We observe similar linear probes accuracy curves to those obtained in local supervised learning despite using the SimCLR loss, with progressive accuracy for the End-to-End model, and high accuracy in early blocks before plateuing for the decoupled network. The model trained with subsampling shows a similar curve, with a slightly increasing improvement through depth. Yet, we do not get a model closer to an End-to-End one; as would observe a decrease of early layer accuracy.

**Impact of the sampling during training** We now study how sampling is affected during training. We report in Fig. 5 the ratio of positive and negative examples kept at each block during training, for a model trained on CIFAR-10 with $K = 16$ and $T = 0$. Surprisingly, almost all positive examples are kept during training despite the improvement provided by their removal. In comparison, as few as a
Table 1. Linear evaluation test accuracy results on CIFAR-10 and STL-10, after training on both SimCLR and our subsampling method (with mean and standard deviation over 5 runs for CIFAR-10). We also report the memory cost of local self-supervised learning and the maximum block MACs required among the $K$.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Method</th>
<th>CIFAR-10</th>
<th>STL-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Max MACs</td>
<td>Memory</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>1 (E2E)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SimCLR</td>
<td>92.1 ± 0.2</td>
<td>1.31 G</td>
</tr>
<tr>
<td></td>
<td>+ ours</td>
<td>91.4 ± 0.4</td>
<td></td>
</tr>
<tr>
<td>STL-10</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SimCLR</td>
<td>90.0 ± 0.1</td>
<td>349 M</td>
</tr>
<tr>
<td></td>
<td>+ ours</td>
<td>89.9 ± 0.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SimCLR</td>
<td>87.5 ± 0.4</td>
<td>196 M</td>
</tr>
<tr>
<td></td>
<td>+ ours</td>
<td>88.4 ± 0.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SimCLR</td>
<td>85.9 ± 0.3</td>
<td>123 M</td>
</tr>
<tr>
<td></td>
<td>+ ours</td>
<td>87.1 ± 0.4</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Linear evaluation test accuracy results on Fashion-MNIST of our method for $K = 1$ and 8 for $T = 0.3$ on a ResNet-18. We observe a similar improvement for this dataset. Results are shown in average for 5 runs.

<table>
<thead>
<tr>
<th>K</th>
<th>SimCLR</th>
<th>+ ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (E2E)</td>
<td>91.3 ± 0.1</td>
<td>90.3 ± 0.2</td>
</tr>
<tr>
<td>8</td>
<td>87.2 ± 0.3</td>
<td>88.2 ± 0.3</td>
</tr>
</tbody>
</table>

Figure 3. Test accuracy on CIFAR10 after training on SimCLR with $K = 16$ for varying threshold $T$. Accuracy peaks around $T = 0$, showing an equilibrium between the dimensionality increase and the removal of examples.

Figure 4. Test accuracy on CIFAR-10 depending on the number of decoupled blocks $K$ after training on SimCLR with our subsampled method, by subsampling either positive or negative examples or both. Subsampling negative examples give the main improvement, but both are needed to reach the best accuracy.

Impact of the threshold value $T$ We also study the accuracy of our model depending on the value of the threshold $T$, and find it to follow three piecewise linear curves. Below $T = -0.2$, accuracy is not improved as few examples are removed. There is a stark increase until approximately $T = 0$ before a decrease. This indicates an equilibrium between increasing dimensionality and removing too many examples. It is unclear why the accuracy peak is as sudden as it is.

Impact of the positive and negative examples To ensure that the removal of both positive and negative examples improves accuracy, we propose to train our method by subsampling only positive or negative examples. More precisely, in Eq. (5) we only add the term $\alpha^k_{i,j}$ to consider positive examples thresholding, or the denominator term $\alpha^k_{i,j}$ to consider negative examples thresholding. We report in Fig. 4 the accuracy of our method for varying $K$ by subsampling through depth.

quarter of negative examples are kept in the last block at convergence, contradicting popular contrastive learning belief that more negative examples improve training. The ratio of negative examples kept decreases during training, which is not surprising as the model converges to have a negative alignment for negative examples. In both cases, the ratio kept decreases also with depth. With a difference of almost 10% between the first and last block, there are fewer negative examples to contrast with the positive ones in later blocks, which indicates a form of curriculum learning.
either positive or negative examples or both. We report an improvement in both cases. Yet, thresholding all examples provides the best accuracy as $K$ increases.

5. Conclusion

This paper investigated the training of DNNs with self-supervised local learning methods. We find that a dimensional collapse partially causes the drop in accuracy known in local learning with larger splits. By studying a linear model, we motivate a simple local feature similarity sampling method which improves on the original SimCLR loss. This method remedies this dimensional collapse, reducing the accuracy loss due to decoupling for models. However, decoupling still causes a significant accuracy gap, indicating other issues. We leave the generalization of our findings to non-contrastive self-supervised objectives for future work.

References


The training was done on A100 GPUs, requiring 12 hours as for the training.

for consistency, on an A100 GPU with the same batch sizes allocated by CUDA, with cuDNN benchmarking turned off for STL10. The memory cost is the maximum memory for a CIFAR-10 run on 1 GPU and 4 hours on 16 GPUs.

Appendix: Implementation details

The training was done on A100 GPUs, requiring 12 hours for a CIFAR-10 run on 1 GPU and 4 hours on 16 GPUs for STL10. The memory cost is the maximum memory allocated by CUDA, with cuDNN benchmarking turned off for consistency, on an A100 GPU with the same batch sizes as for the training.

Architecture details

The model we study is a ResNet-50, composed of a convolutional layer, a BatchNorm layer followed by a ReLU activation and a max pooling layer, then 16 Bottleneck blocks, a final global average pooling and a fully connected layer; following Pytorch official implementation at https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py.

Since the datasets we consider have lower image sizes than the ImageNet dataset for which the ResNet model was initially designed for, we propose different first layers before the bottleneck layers, as standard. For STL-10, CIFAR-10 and Fashion-MNIST datasets, we remove the max pooling layers. For STL-10, the convolution layer is the same as standard, except for a slight reduction of kernel size from 7 to 5. For the other smaller datasets, the convolution layer has a kernel size of 3, and stride and padding of 1.

Table 3. Decoupling points of our ResNet-50 architectures depending on the number of blocks $K$. The network is composed of 4 main layers, each composed of 3, 4, 6 and 2 Bottleneck blocks respectively. At each decoupling point, the local training loss is computed and backpropagated, and the following representations pass through a StopGrad operator to prevent gradients between blocks.

<table>
<thead>
<tr>
<th>K</th>
<th>Decoupling points (after Layer i and Bottleneck j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>(2, 1), ((3, 1), (3, 5))</td>
</tr>
<tr>
<td>8</td>
<td>(1, 2), (2, 1), (2, 3)), ((3, 1), (3, 3), (3, 5)), (4, 1)</td>
</tr>
<tr>
<td>16</td>
<td>After each Bottleneck</td>
</tr>
</tbody>
</table>

Split details

Our model is decoupled at several ‘decoupling points’ where gradient information is stopped. For $K = 4.8$ the split is adjusted to maintain an equal number of bottlenecks in each block, leading for $K = 16$ to split after the bottleneck of each block. To be more precise on the location of these decoupling points, we refer to Table 3, where we localize them with the layer (out of 4) and bottleneck numbers starting from 1.

Each local projector $g^k$ is composed of: a $3 \times 3$ convolutional layer (with stride 2 and the same number of channels as the representation), a batch normalization and ReLU layer, a global average pooling, then a fully connected layer, a ReLU, and a final fully connected layer with output dimension 128. Note also that the auxiliary projector networks used in our methods are used at every decoupling point before the local loss, but not for the final layer, which uses the classical 2-layer MLP projection head.

Augmentation details

Images are augmented following the simple augmentation procedure proposed by SimCLR without Gaussian Blur: a Random Resized Crop to the necessary image size, a random horizontal flip (with probability 0.5), a random color jitter (with probability 0.8 and bright-
ness contrast and saturation parameters equal set to 0.4 and hue to 0.1 and random color dropping (setting to grayscale, with probability 0.2). Test images are not augmented.