solo-learn: A Library of Self-supervised Methods for Visual Representation Learning

Victor G. Turrisi da Costa
University of Trento - Trento, Italy
VG.TURRISIDACOSTA@UNITN.IT

Enrico Fini
University of Trento - Trento, Italy
ENRICO.FINI@UNITN.IT

Moin Nabi
SAP AI Research - Berlin, Germany
M.NABI@SAP.COM

Nicu Sebe
University of Trento - Trento, Italy
NICULAE.SEBE@UNITN.IT

Elisa Ricci
University of Trento and Fondazione Bruno Kessler - Trento, Italy
E.RICCI@UNITN.IT

Editor: Alexandre Gramfort

Abstract

This paper presents solo-learn, a library of self-supervised methods for visual representation learning. Implemented in Python, using Pytorch and Pytorch lightning, the library fits both research and industry needs by featuring distributed training pipelines with mixed-precision, faster data loading via Nvidia DALI, online linear evaluation for better prototyping, and many additional training tricks. Our goal is to provide an easy-to-use library comprising a large amount of Self-supervised Learning (SSL) methods, that can be easily extended and fine-tuned by the community. solo-learn opens up avenues for exploiting large-budget SSL solutions on inexpensive smaller infrastructures and seeks to democratize SSL by making it accessible to all. The source code is available at https://github.com/vturrisi/solo-learn

Keywords: Self-supervised methods, contrastive learning

1. Introduction

Deep networks trained with large annotated datasets have shown stunning capabilities in the context of computer vision. However, the need for human supervision is a strong limiting factor. Unsupervised learning aims to mitigate this issue by training models from unlabeled datasets. The most prominent paradigm for unsupervised visual representation learning is Self-supervised Learning (SSL), where the intrinsic structure of the data provides supervision for the model. Recently, the scientific community devised increasingly effective SSL methods that match or surpass the performance of supervised methods. Nonetheless, implementing and reproducing such works turns out to be complicated. Official repositories of state-of-the-art SSL methods have very heterogeneous implementations or no implementation at all. Although a few SSL libraries (Goyal et al., 2021; Susmelj et al., 2020) are available, they assume that larger-scale infrastructures are available or they lack some recent methods. When approaching SSL, it is hard to find a platform for experiments that

* Victor G. Turrisi da Costa and Enrico Fini contributed equally.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/ Attribution requirements are provided at http://jmlr.org/papers/v23/21-1155.html
allows running all current state of the art methods with low engineering effort and at the
same time is effective and straightforward to train. This is especially problematic because,
while the SSL methods seem simple on paper, replication of published results can involve a
huge time and effort from researchers. Sometimes official implementations of SSL methods
are available, however, releasing standalone packages (often incompatible with each other)
is not sufficient for the fast-paced progress in research and emerging real-world applications.
There is no toolbox offering a genuine off-the-shelf catalog of state-of-the-art SSL techniques
that is computationally efficient, which is essential for in-the-wild experimentation.

To address these problems, we present solo-learn, an open-source framework that
provides standardized implementations for a large number of state-of-the-art SSL methods.
We believe solo-learn will enable a trustworthy and reproducible comparison between
the state of the art methods. The code that powers the library is written in Python,
using Pytorch (Paszke et al., 2019) and Pytorch Lightning(PL) (Team, 2019) as back-ends
and Nvidia DALI\footnote{https://github.com/NVIDIA/DALI} for fast data loading, and supports more modern methods than related
libraries. The library is highly modular and can be used as a complete pipeline, from
training to evaluation, or as standalone modules.

2. The solo-learn Library: An Overview

Currently, we are witnessing an explosion of works on SSL methods for computer vision.
Their underlying idea is to unsupervisedly learn feature representations by enforcing similar
feature representations across multiple views from the same image while enforcing diverse
representations for other images. To help researchers have a common testbed for repro-
ducing different results, we present solo-learn, which is a library of self-supervised meth-
ods for visual representation learning. The library is implemented in Pytorch, providing
state-of-the-art self-supervised methods, distributed training pipelines with mixed-precision,
faster data loading, online linear evaluation for better prototyping, and many other training
strategies and tricks presented in recent papers. We also provide an easy way to use the
pre-trained models for object detection, via DetectronV2 (Wu et al., 2019). Our goal is to
provide an easy-to-use library that can be easily extended by the community, while also
including additional features that make it easier for researchers and practitioners to train
on smaller infrastructures.

2.1 Self-supervised Learning Methods

We implemented 13 state-of-the-art methods, namely, Barlow Twins (Zbontar et al., 2021),
BYOL (Grill et al., 2020), DeepCluster V2 (Caron et al. 2020), DINO (Caron et al., 2021),
MoCo V2+ (Chen et al., 2020b), NNCLR (Dwibedi et al., 2021), ReSSL (Zheng et al., 2021),
SimCLR (Chen et al., 2020a), Supervised Contrastive Learning (Khosla et al., 2020),
SimSiam (Chen and He, 2021), SwAV (Caron et al., 2020), VICReg (Bardes et al., 2021)
and W-MSE (Ermolov et al., 2021).

2.2 Architecture

In Figure 1, we present an overview of how a training pipeline with solo-learn is car-
rried out. In the bottom, we show the packages and external data at each step, while at
the top, we show all the defined variables on the left and an example of the newest defined
variable on the right. First, the user interacts with *solo.args*, a subpackage that is responsible for handling all the parameters selected by the user and providing automatic setup. Then, *solo.methods* interacts with *solo.losses* to produce the selected self-supervised method. While *solo.methods* contains all implemented methods, *solo.losses* contains the loss functions for each method. Afterwards, *solo.utils* handles external data to produce the pretrain dataloader, which contains all the transformation pipelines, model checkpoint, automatic UMAP visualization of the features, other backbone networks, such as ViT (Dosovitskiy et al., 2021) and Swin (Liu et al., 2021), and many other utility functionalities. Lastly, this is given to a PL trainer, which provides hardware support and extra functionality, such as, distributed training, automatic logging results, mixed precision and much more. We note that although we show all subpackages working together, they can be used in a standalone fashion with minor modifications. Apart from that, we have documentations in the folder *docs*, downstream tasks in *downstream*, unit tests in *tests* and pretrained models in *zoo*.

### 2.3 Comparison to Related Libraries

The most related libraries to ours are VISSL (Goyal et al., 2021) and Lightly (Susmelj et al., 2020), which lack some of our key features. First, we support more modern SSL methods, such as BYOL, NNCLR, SimSiam, VICReg, W-MSE and others. Second, we target researchers with fewer resources, namely from 1 to 8 GPUs, allowing much faster data loading via DALI. Lastly, we provide additional utilities, such as automatic linear evaluation, support to custom datasets and automatically generating UMAP (McInnes et al., 2020) visualizations of the features during training.

### 3. Experiments

**Benchmarks.** We benchmarked the available SSL methods on CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 Krizhevsky et al. (2009) and ImageNet-100 (Deng et al., 2009) and made public the pretrained checkpoints. For Barlow Twins, BYOL, MoCo V2+, NNCLR, SimCLR and VICReg, hyperparameters were heavily tuned, reaching higher performance than reported on original papers or third-party results. Tab. 1 presents the top-1 and top-5 accuracy values for the online linear evaluation. For ImageNet-100, traditional offline linear evaluation is also reported. We also compare with the results reported by Lightly in Tab. 3.
Nvidia DALI vs traditional data loading. We compared the training speeds and memory usage of using traditional data loading via Pytorch Vision\(^2\) against data loading with DALI. For consistency, we ran three different methods (Barlow Twins, BYOL and NNCLR) for 20 epochs on ImageNet-100. Tab. 2 presents these results.

Table 1: Online linear evaluation accuracy on CIFAR-10, CIFAR-100 and ImageNet-100. In brackets, offline linear evaluation accuracy is also reported for ImageNet-100.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>ImageNet-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc@1</td>
<td>Acc@5</td>
<td>Acc@1</td>
</tr>
<tr>
<td>Barlow Twins</td>
<td>92.10</td>
<td>99.73</td>
<td>70.90</td>
</tr>
<tr>
<td>BYOL</td>
<td>92.58</td>
<td>99.79</td>
<td>70.46</td>
</tr>
<tr>
<td>DeepCluster V2</td>
<td>88.85</td>
<td>99.58</td>
<td>63.61</td>
</tr>
<tr>
<td>DINO</td>
<td>89.52</td>
<td>99.71</td>
<td>66.76</td>
</tr>
<tr>
<td>MoCo V2+</td>
<td>92.94</td>
<td>99.79</td>
<td>69.89</td>
</tr>
<tr>
<td>NNCLR</td>
<td>91.88</td>
<td>99.78</td>
<td>69.62</td>
</tr>
<tr>
<td>ReSSL</td>
<td>90.63</td>
<td>99.62</td>
<td>65.92</td>
</tr>
<tr>
<td>SimCLR</td>
<td>90.74</td>
<td>99.75</td>
<td>65.78</td>
</tr>
<tr>
<td>Simsiam</td>
<td>90.51</td>
<td>99.72</td>
<td>66.04</td>
</tr>
<tr>
<td>SwAV</td>
<td>89.17</td>
<td>99.68</td>
<td>64.88</td>
</tr>
<tr>
<td>VICReg</td>
<td>92.07</td>
<td>99.74</td>
<td>68.54</td>
</tr>
<tr>
<td>W-MSE</td>
<td>88.67</td>
<td>99.68</td>
<td>61.33</td>
</tr>
</tbody>
</table>

Table 2: Speed and memory comparison with and without DALI on ImageNet-100.

<table>
<thead>
<tr>
<th>Method</th>
<th>DALI</th>
<th>20 epochs</th>
<th>1 epoch</th>
<th>Speedup</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barlow Twins</td>
<td>✓</td>
<td>1h 38m 27s</td>
<td>4m 55s</td>
<td>-</td>
<td>5097 MB</td>
</tr>
<tr>
<td>BYOL</td>
<td>✓</td>
<td>1h 38m 46s</td>
<td>4m 56s</td>
<td>-</td>
<td>5409 MB</td>
</tr>
<tr>
<td>NNCLR</td>
<td>✓</td>
<td>1h 38m 30s</td>
<td>4m 55s</td>
<td>-</td>
<td>5060 MB</td>
</tr>
</tbody>
</table>

Table 3: Comparison with Lightly on CIFAR10.

<table>
<thead>
<tr>
<th>Method</th>
<th>Ours</th>
<th>Lightly</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimCLR</td>
<td>90.74</td>
<td>89.0</td>
</tr>
<tr>
<td>MoCoV2+</td>
<td>92.94</td>
<td>90.0</td>
</tr>
<tr>
<td>Simsiam</td>
<td>90.51</td>
<td>91.0</td>
</tr>
</tbody>
</table>

4. Conclusion

Here, we presented solo-learn, a library of self-supervised methods for visual representation learning, providing state-of-the-art self-supervised methods in Pytorch. The library supports distributed training, fast data loading and provides many utilities for the end-user, such as online linear evaluation for better prototyping and faster development, many training tricks, and visualization techniques. We are continuously adding new SSL methods, improving usability, documents, and tutorials. Finally, we welcome contributors to help us at https://github.com/vturrisi/solo-learn

Acknowledgments. This work was supported by a joint project under Grant No. JQ18012, by the EU H2020 AI4Media No. 951911 project and the European Institute of Innovation & Technology (EIT).

2. https://github.com/pytorch/vision
References


Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 2009.


