Summary Report : A Simple Framework for Contrastive Learning of Visual Representations

Théo Michel, Widad Zizouan

Summary Report : A Simple Framework for Contrastive Learning of Visual Representations

Widad ZIZOUAN
wzizouan@enseirb-matmeca.fr
Théo MICHEL
tmichel003@enseirb-matmeca.fr
Supervisor : Michael Clément

Index Terms—Self-Supervised, Machine Learning, ResNet, Contrastive Learning

1 Abstract

This is a short article trying to reproduce the some of the findings of the SimCLR paper [1]. The SimCLR paper was published at the ICML conference in 2020 by the Google AI team. The following experiment has been done with different hyper-parameters and with sometimes different data sets than the original paper, but still managed to observe around the same relative effectiveness to other supervised and self-supervised methods trained in the same conditions. The results found here are in no way an affirmation of the original paper, just a hint at the possibility of using those same methods with smaller training requirements. This was written in the context of a second year introduction to doctoral research at Enseirb-Matmeca by computer science students.

2 Introduction

Recently the explosion in computing power has lead to a deep learning revolution going ever deeper and training on ever more data. We have managed to have incredible results doing this. But the number of huge data sets is limited to some tasks, that is why we need to find more efficient methods to train our algorithms using for example self-supervised and transfer learning.
The SimCLR method for training neural networks (here a ResNet 50) has two particularities, a projection head, and a contrastive learning with a precise, empirically chosen set of transformations on the images. Using these techniques self-supervised learning is trying to be achieved, and the effectiveness of this training in improving other tasks is being measured.
In this article we chose to use the SimCLR algorithm on a ResNet 18 architecture, comparing it to different auto-encoders, and to supervised learning (ResNet 18) algorithms to try to reproduce parts of the findings of the aforementioned paper.

2.1 Presentation of the algorithm

The SimCLR method's aim is to teach a model to find good representations of images from unlabeled data. It does so by using contrastive learning and strong data augmentation, comparing the augmented images from the same class with the augmented images from different images, and trying to increase the difference in representation between the augmented images obtained from different classes and at the same time maximising the agreement between different view of the same image. This is done using a contrastive loss. The loss function for a positive pair (i,j) is defined as below:

$$l_{i,j} = -log(\frac{exp(\frac{sim(z_i,z_j)}{\tau})}{\sum_{k=1}^{2N} \mathbf{1}_{k\neq i} exp(\frac{sim(z_i,z_k)}{\tau})})$$

with $\mathbf{1}_{k\neq i}$ being the indicator function, it allows us to eliminate the computation of the similarity of z_i with itself. It's worth mentioning that the goal of the function `sim` is to measure the similarity between two vectors, the higher it is the higher the similarity should be between these two vectors. Here, It is defined by :
$sim(u,v) = \frac{u^T v}{||u||v||}$ which is the `cosine` of the "angle" between the two vectors

2.2 The different modules of the algorithm

- A data augmentation module applying random transformations on the images$x$ given as input. These transformations have been chosen empirically and it would be interesting to see if better augmentations could be found (using adversarial attacks for example). The best transformations have been found to be : a random cropping followed by a resize back to the original size, random color distortions, and random Gaussian blur.[1]
  In the paper the images after transformation have been noted : $\tilde{x}_i$ and$\tilde{x}_j$.
- The ResNet based encoder is noted $f(.)$ and its output is noted h in the paper, it is important to note that this net could be replaced by any other neural net architecture.
  h is the representation being learned, this vector is the main focus of the method. We are aiming for this vector to be a"good" representation of x, which means a vector that extracts usefull information.

- A projection head which is a small two layer deep and 128 neurons wide neural network. It takes as input h and is noted $g(.)$ and its output z. This neural network has also been empirically shown to improve performance of the h representation. $z = g(h) = W^{(2)}\sigma(W^{(1)}h)$ with W the weights and σ a Relu function.
- The contrastive loss function $NT - Xent$ aiming to be **low** for $z_k, k \in \{i, j\}$and$z_k, k \in \{i, j\}$ as inputs, and **high** for $z_k, k \in \{i, j\}$and $z_l k \neq \{i, j\}$ as inputs.

2.3 The training method

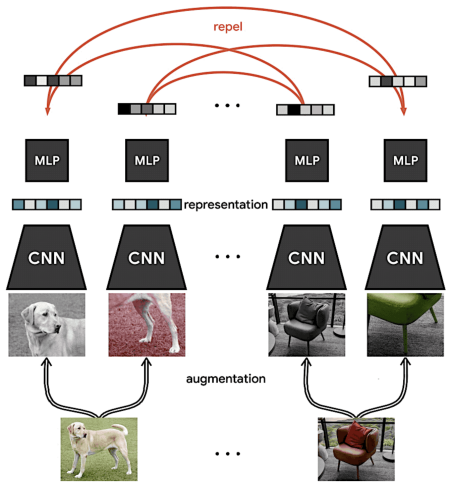

Fig. 1. Visual representation of the algorithm [2]

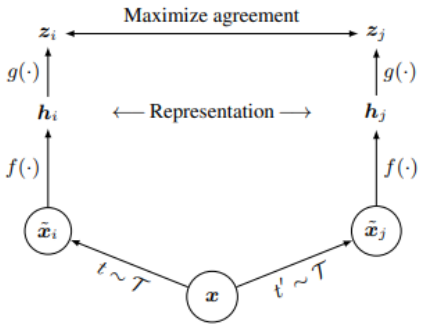

Fig. 2. Visual representation of the algorithm taken from the original paper [1]

3 Methodology

In the original paper the training algorithm is being computed by a cluster of Tesla V100 graphic cards, enabling the use of large and High-Res data Sets such as Image Net and large batch sizes (4096) which are reported to have an improvement on the performance. In this article everything has been scaled down proportionally: data set size, image size and batch size in the hope of seeing the same patterns emerge as in the SimCLR paper.
3 neural nets and trained in varying ways were compared :

- A Resnet 18 neural net trained with the SimCLR method

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^{\top} \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

Fig. 3. Algorithm shown in the paper [1]

- A Resnet 18 algorithm trained in a supervised manner
- A 6 layer deep autencoder 3 layers deep on encode and decode, with a 128 encoded size.

The specific example being reproduced is **the evaluation of the representation obtained on the layer h of the SimCLR algorithm with a linear classifier.**
The idea behind using a linear classifier to judge the quality of a representation stems from the fact that linear classifiers learn best from data that is well structured in which patterns can be easily identified. The exact procedure is as follows :

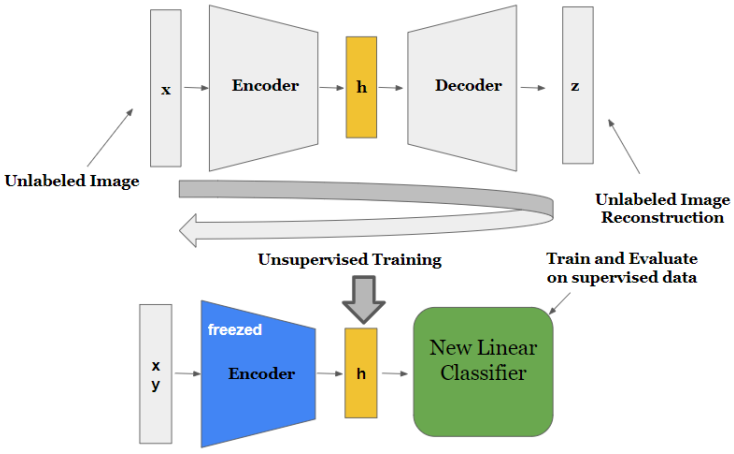

Fig. 4. Example of the training and evaluation pipeline of the auto Encoder. On top we can see the auto encoder being trained on unlabeled data reconstructing image from it's own representation. On the bottom the subsequent extraction and freezing of the encoder and the evaluation of a new linear classifier on the output of this one.

an algorithm is being trained on data, often non labeled data. Then you take away the end layers of the algorithm to access the interesting representation we call it h. Finally

you freeze the weights of the remaining encoder, which means they won't update their weights and biases with subsequent gradient descents. And you add at the end a Linear Classifier with randomly initialized weights and train the model on labeled data. You then test the prediction accuracy on the output of the linear classifier.

3.1 The points of comparison

To evaluate the quality of the representation the Authors of the SimCLR paper have decided to evaluate the quality of the representation by putting a linear classifier with as input the representation being evaluated. All the weights of the rest of the model where the frozen and only the last layer was trained then evaluated on labeled images. Here the Resnet18 model supervised algorithm is used as the supervised learning baseline.
The Auto-Encoder is used as a comparison to an other self-Supervised algorithm, even though it has to be noted that this is not a fair comparison, the Auto-Encoder being of a shallower model. A better comparison would be to an Auto-Encoder using a ResNet18.
The representation created on the last layer of the Resnet18 supervised layer has also been evaluated to have a glimpse at the quality and transferability of inner layers of a neural net.

3.2 Experimental Conditions

The experiments have been carried out on PyTorch instead of TensorFlow for ease of use, the SimCLR method algorithm and the augmentations are based on a PyTorch implementation of the algorithm by Thalles Silva [3]. The training has been carried out on a GTX 1050 with 4GB of Vram, the batch size and the size of the images where as high as we could go. Even though the effectiveness of the method is greatly reduced on such hardware because of the fact that for contrastive learning bigger mini batches means you can compare your augmented data between more views, it is interesting to see the effectiveness of self supervised learning in smaller hardware as maybe it could be used to one day learn more effectively for IOT devices. This is speculation but we could for example imagine a smart home assistant starting to recognize habits in a self supervised manner, and then users would just need to do a little bit of labeling to teach it to be effective, making the task of home automation less tedious.

Training Parameters	
Projection head Dimensionality	128, 128
Data Set	Cifar10
Epochs	100
Optimizer	Adam
Batch size	256
Learning rate	1e-3
Weight decay	1e-4

4 Experimental results

Let's first look the supervised learning baseline, using a ResNet18 on Cifar 10.

4.1 Supervised Baseline

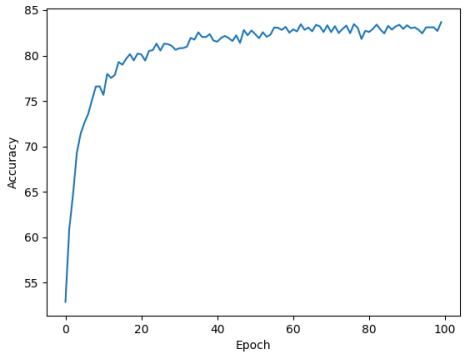

Fig. 5. Accuracy on the validation set of the Supervised Baseline

We see the learning plateau of at around 83% .

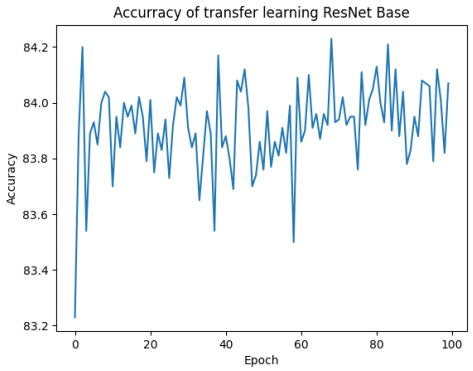

Fig. 6. Linear classifier learning on the last layer of the ResNet18 trained in a supervised manner

The ResNet's last layer is a linear classifier, so what we have done is essentially take out the last layer and replace it with the same kind of layer. So it is not surprising that by training it again on labeled data you achieve the same accuracy as in the original ResNet. We can conclude that the representation on the before last layer of a ResNet18 model trained in a supervised manner is of quality.

4.2 Auto-Encoder

After 100 epochs figure 7 are the decoded images of the auto encoder.

These images Fig 7 have been decoded from a 128 wide layer of neurons, we can visualise these images to try to understand what information is present in the representation at the heart of the auto-encoder.

On the Fig 10 the linear Classifier has pretty quickly achieved a learning of 54% on the representation obtained at the exit of the encoder, but doesn't go beyond that. The figure 9 can help you visualise the precision of the classifier on top of the encoder.

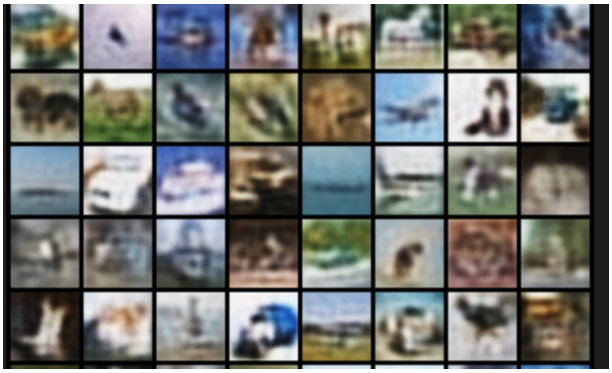

Fig. 7. Output of the Auto-Encoder trained for 100 epochs

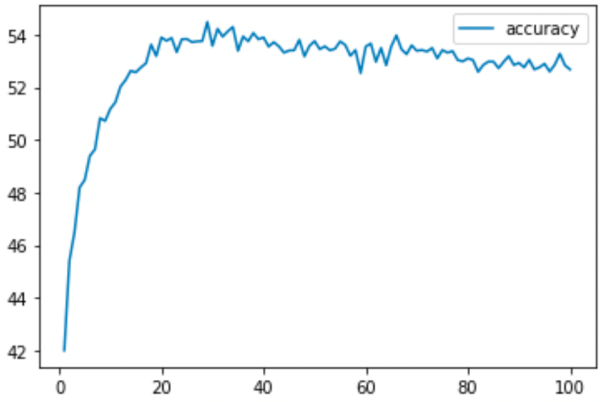

Fig. 8. Transfer learning on the supervised learning algorithm

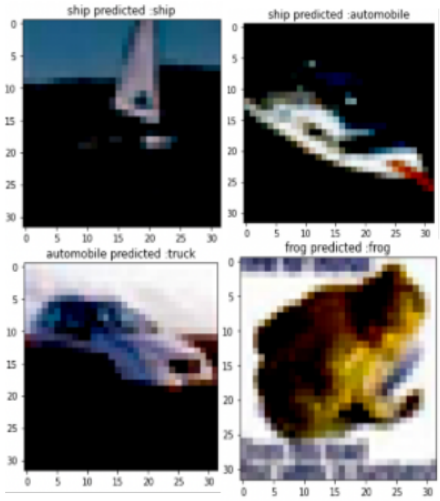

Fig. 9. Transfer learning on the supervised learning algorithm

4.3 SimCLR trained algorithme

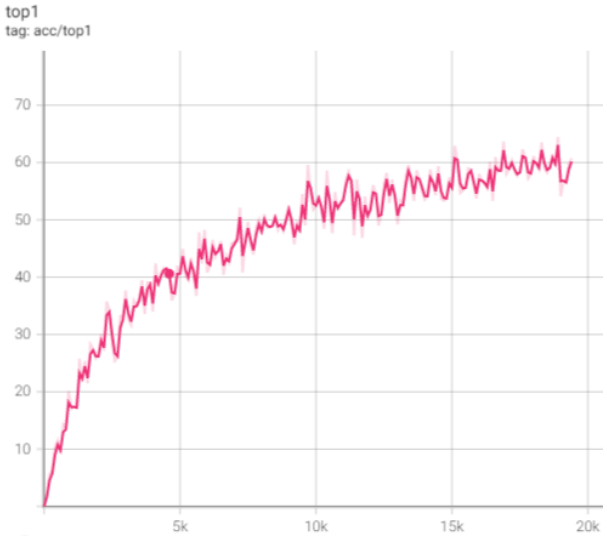

Fig. 10. Accurracy of the SimCLR algorithm on the mini Batch recognition. Exceptionnaly here, the accuracy is a measure of the distance between the vectors of similarity matrix and the labels

The ResNet18 has correctly learned the task of differentiating images among themselves doing contrastive learning Fig.11. But this result is lower than the one observed in the paper on the same data set, this can be explained by the lower Batch Size which is really important see Figure 9 of the original paper. It is also explained by the different architecture which is a ResNet18 in our case, instead of a ResNet50. [1].

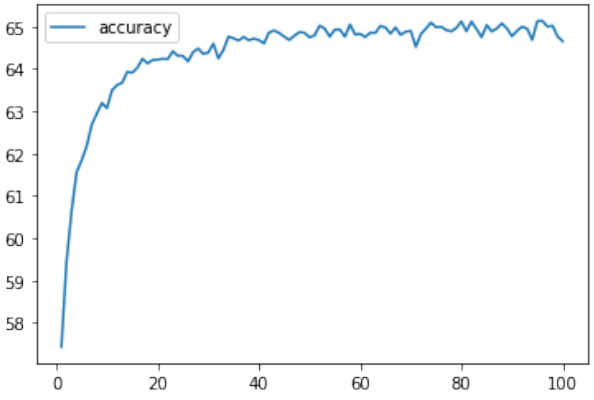

Fig. 11. Accuracy of the ResNet 18 model trained with SimCLR, with a classification layer on top on the validation set

The accuracy of the top1 predictions on Cifar10 from a linear classier on top of the SimCLR representation is around 65% after 100 epochs. This result is quit a bit lower what was found on the original paper of 84% and is maybe due to varying hyper-parameters and the different ResNet50 architecture see Figure B.7 of the original paper [1].

5 Conclusion

To sum up, we have measured the effectiveness of SimCLR on smaller data Sets, with smaller epochs and smaller models. We have seen results coherent with the original paper's findings even though significantly lower which could indicate, in the same way the SimCLR paper pointed out, that for this method scale is important.

On a more personal note, this experience has been an incredible learning opportunity on the manipulation and understanding of Machine Learning algorithms. It also was a really humbling experience to understand the rigorousness needed to be a good researcher, and the difficulty of contributing effectively to the scientific community.

6 Acknowledgements

We would like to thank Frédéric HERBRETEAU have given us the opportunity to work on such an interesting subject.
We also would like to thank Michael CLEMENT for having patiently guided us threw the understanding of this paper, and for the help in the scientific method needed to carry out such experiments.

References

[1] Original Paper, https://arxiv.org/abs/2002.05709, *Ting Chen, Simon Kornblith, Mohammad Norouzi, Geoffrey Hinton.*
[2] Google AI Blog, https://ai.googleblog.com/2020/04/advancing-self-supervised-and-semi.html,*Ting Chen Geoffrey Hinton*
[3] The Pytorch repository used to implement the SimCLR training protocol https://github.com/sthalles/SimCLR,*Thalles Silva,*