Project 2

João Victor da Silva Guerra, Leonardo Alves de Melo, and Marcos Felipe de Menezes Mota [*]

Abstract

In this project, we attempt to reproduce SimCLR, which is an self-supervised framework for contrastive learning of visual representations. Contrastive learning is the learning paradigm that leverage unlabeled data to teach an algorithm to distinguish between similar and dissimilar data. The composition of data transformations, the architecture composed by a base encoder and a projection head, and the contrastive loss using the nonlinear transformation of the representation to learn the parameters improves the quality of the learned representations. Following the approach of the original paper, we evaluate the effects of batch size, optimizing algorithm, similarity function and data transformations. Finally, we trained an model for 100 epochs with the best design choices from our experiments and evaluate the final model accuracy on CIFAR-10 dataset.

1 Introduction

The creation of a predictive algorithm to classify images is a great challenge in machine learning and computer vision areas. Labeling every image in a dataset could be an expensive and slow task, being impossible sometimes. A way to solve that is using self-supervised algorithms, which can predict a type of class even without someone explicitly tells what are the types, they basically compares each example to another to get the similarity between them, using this comparisons to learn useful representations of the data. With such representations, a simple linear classifier on top of the self-supervised framework can predict class labels with great accuracy. Self-supervised is flexible because the learned representations can be used for other machine learning tasks like semi-supervised learning and transfer learning [1].

In this project we will explore and reproduce a state-of-the-art self-supervised machine learning algorithm called SimCLR [1], which reached great results compared with supervised algorithms. The SimCLR accomplished one of the main goals of the self-supervision community which is to achieve supervised performance using much less data. In the paper the authors reported 85.8% top-5 accuracy in ImageNet using 100x fewer labels than AlexNet. Our implementation will be trained over CIFAR-10 dataset and the experiments will evaluate the accuracy and best design choices.

2 Dataset

In our experiments, we used the CIFAR-10 dataset, which consists of 60000 colour images with 32x32 RGB pixels, totalizing 3072 features. The dataset contains 10 classes (i.e. airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck) with 5000 images in the training set and 1000 images in the test set per class. We show some examples of each class in Figure 1.

3 Implementation

Our architecture code is divided in three main components: a data augmentation, a base encoder $f(\cdot)$ and a projection head $g(\cdot)$ modules, as can be seen in Figure 2.

The data augmentation module randomly draws examples from the CIFAR-10 dataset, transforming each example twice using a combination of simple augmentations described in Section § 3.1, creating two sets (x_i and x_j) of corresponding views.

The base encoder $f(\cdot)$ computes the image representations ($h = f(x)$) using a convolutional neural network variant based on the ResNet architecture [2], which we use a resnet-50 model from `torchvision` package. The

[*]117410, 156188 and 211893. j117410@dac.unicamp.br, leonardo.alves.melo.1995@gmail.com., and marcos.mota@ic.unicamp.br

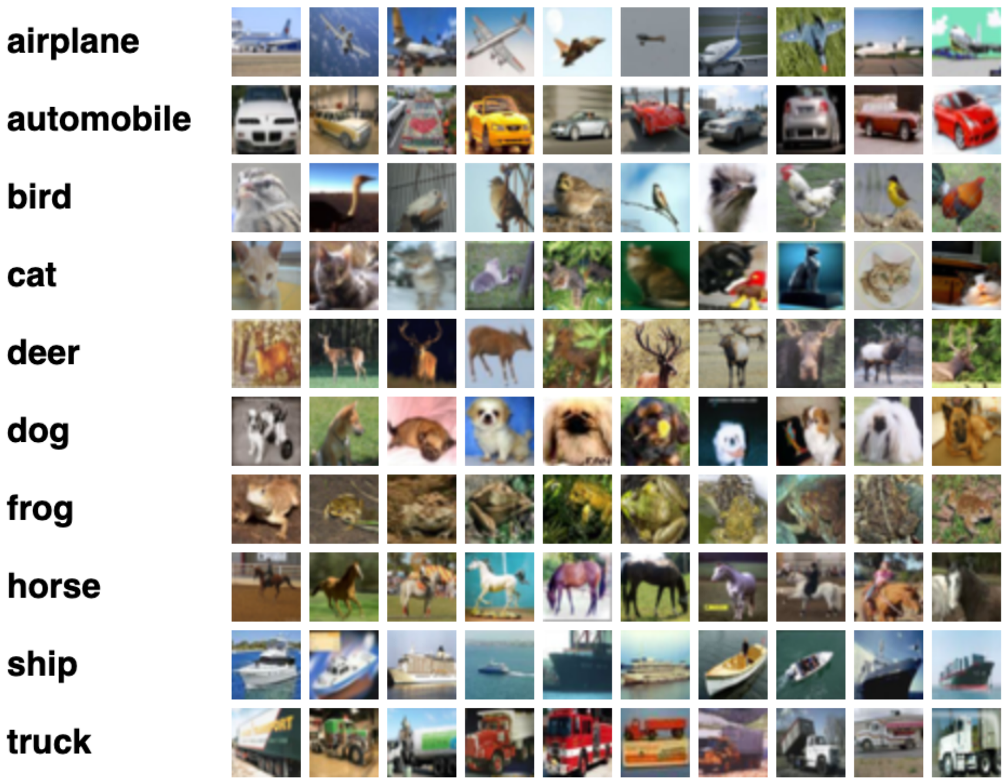


Figure 1. Examples of CIFAR-10 classes.

ResNet receives an augmented image x of shape (32, 32, 3) and outputs a 2048-dimensional embedding vector h, which is our latent representation.

The projection head $g(\cdot)$ computes a non-linear projection of the image representation ($z = g(h)$) using a MultiLayer Perceptron (MLP), which has two dense layers (the first has 2048 units and the second 512 units) and the hidden layer has a non-linearity (ReLU) activation function. The MLP receives a latent representation vector h of size 2048 and outputs a 128-dimensional final representation z. This module amplifies the invariant features and maximizes the ability of the network to identify different transformations of the same image.

3.1 Data augmentation

Our implementation learns representations by maximizing agreement between differently augmented image of the same data example via a contrastive loss in the latent space. The data augmentation operations employed in this project include spatial/geometric transformations of data, such as cropping, resizing (with horizontal flipping) and rotation, and appearance transformation, such as color distortion and Gaussian blur. In Figure 3, we show the augmentations, that we use in this project, applied to an example of our dataset.

3.2 Training procedure

The parameters of both base encoder (ResNet) and projection head (MLP) are updated through a stochastic gradient descent (SGD) to minimize the loss function of the contrastive objective.

For the similarity function, we employed a cosine similarity function (s; Eq. 1), which measures the cosine of the angle between two non-zero d-dimensional vectors. Then, our method uses a contrastive loss function, called NT-Xent (the normalized temperature-scaled cross entropy) loss, shown in Eq. 2, which final value is computed across all positive pairs, (i, j) and (j, i), in a mini-batch.

$$s_{i,j} = \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|} \tag{1}$$

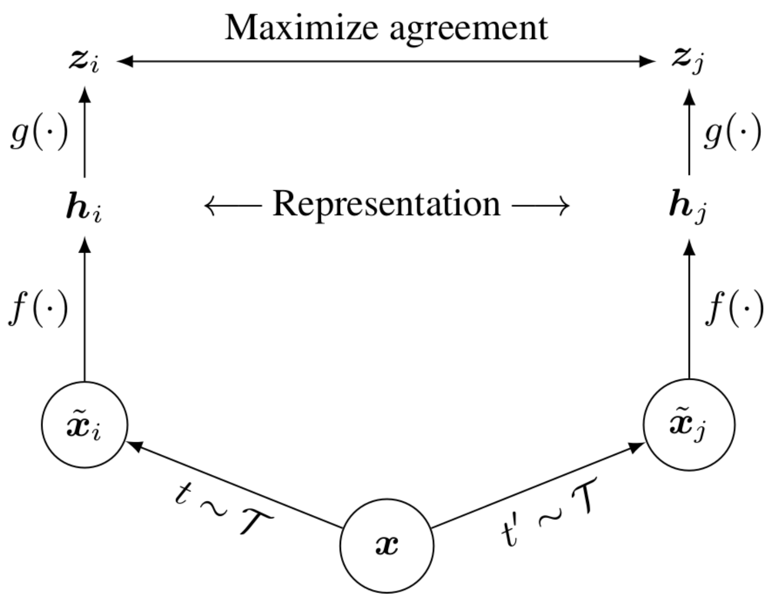
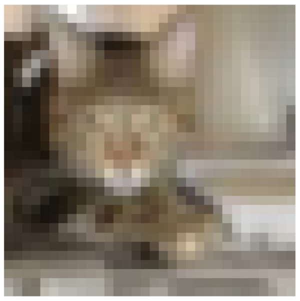
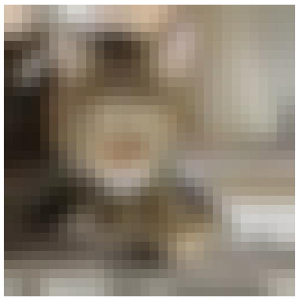


Figure 2. SimCLR Framework.



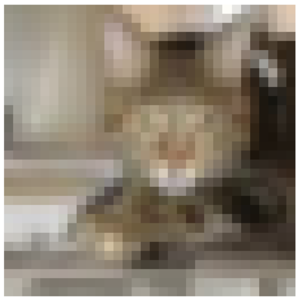
(**a**) Original



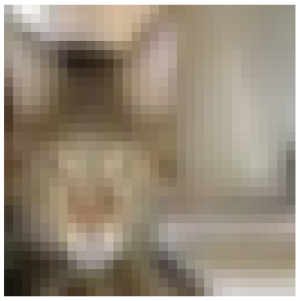
(**b**) Gaussian blur

(**c**) Color distortion (jitter)

(**d**) Color distortion (drop)

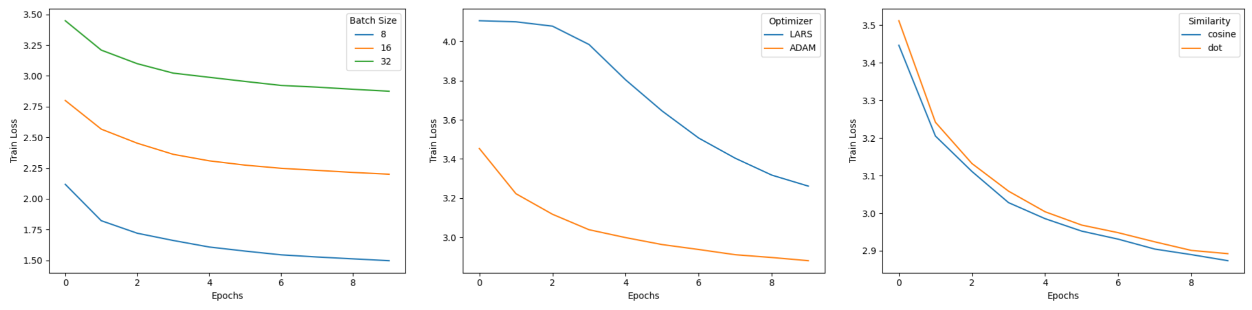


(**e**) Horizontal flip



(**f**) Crop and resize

Figure 3. Illustration of the data augmentation operations. Each augmentation transform data stochastically with some internal parameter.



(a) Batch size **(b)** Optimizer **(c)** Similarity function

Figure 4. NT-Xent loss per epoch.

$$\ell_{i,j} = -\log \frac{\exp(s(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(s(z_i, z_k)/\tau)} \tag{2}$$

where $\mathbb{1}_{[k\neq i]} \in 0, 1$ is an indicator function evaluating to one if and only if $k \neq i$ and τ is a temperature parameters.

Once our system is trained, we can either directly use the output of the base encoder (ResNet) as the representation h of an given image x, or we can use the representations h to learn downstream tasks.

4 Experiments

First, we compared some design choices: batch size (8, 16 and 32), optimizing algorithm (Adam [3] and LARS [4]) and similarity function (cosine similarity and dot similarity). The optimizing algorithm and similarity function experiments used a batch size of 32 samples. For each of these design choices, we trained our model for 10 epochs. After, we trained our final model with the best design choices for 100 epochs.

5 Results

5.1 Loss

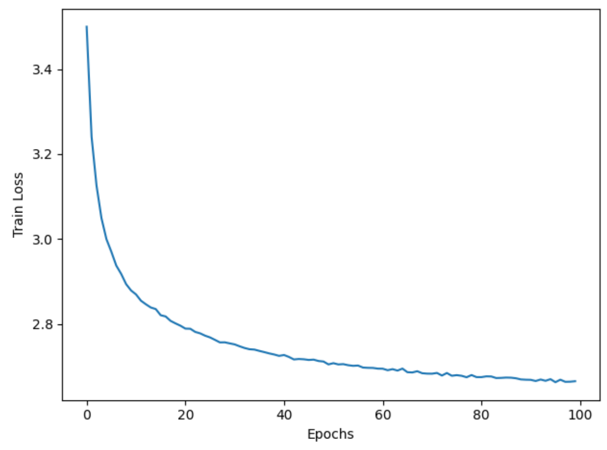
The NT-Xent loss function per epoch of the batch size, optimizing algorithm and similarity function experiments are shown in Figures 4a, 4b and 4c, respectively. We observed that the loss function decreases and smoothly stabilizes in all scenarios, which is the expected behaviour for our loss function. These minimization of the loss in the latent space indicates that our models are learning representations that maximizes the agreement between our differently augmented views of the same data example.

The loss for larger batches is greater than the ones reported in the original paper. Batch size has a great impact when the number of epochs are small and stabilizes as the number of epochs used increased. The Adam optimizer showed a more stable behaviour when compared to the LARS optimizer, which was used in [1]. Probably, the LARS would achieve a more stable behaviour if we finely tuned its parameters. Finally, the cosine similarity presented a smaller loss when compared to the dot product similarity. The difference between cosine similarity and dot product similarity is small. Thus we can see that learned representations have a normalizing factor because the angle between vectors in the representation space better define the separation of data classes.

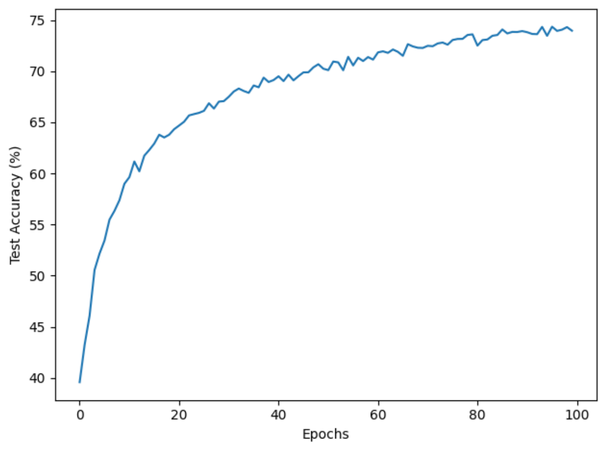
For our final model, we chose a batch size of 32 samples, the Adam optimizer and the cosine similarity function. The NT-Xent loss function per epoch is shown in Figure 5a. We noted that the loss function decreases and smoothly stabilizes. However, we believe that it still does not converge completely. Hence, with more epochs and larger batch sizes, we would achieve a smaller loss and a better accuracy score.

5.2 Test accuracy

The test accuracy compare the representations of our training set with the ones of our test set, both without any augmentation. We calculate the similarity between them and evaluate if the target label for each test sample agree

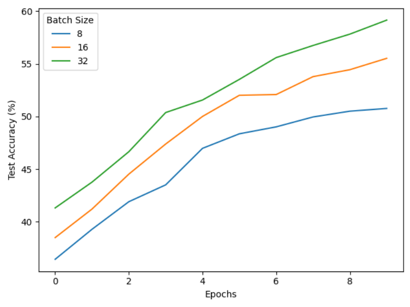


(a) NT-Xent loss per epoch

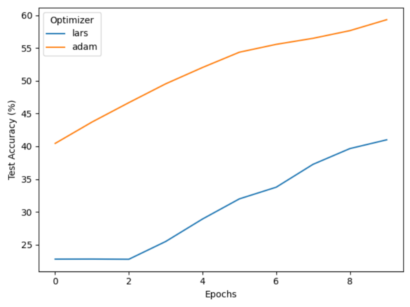


(b) Test accuracy per epoch

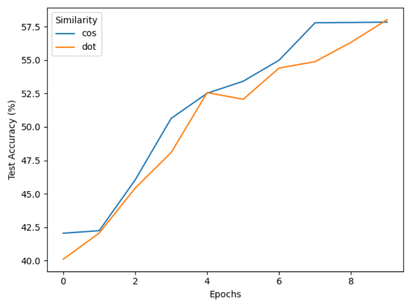
Figure 5. Metrics for our final model.



(a) Batch size



(b) Optimizer



(c) Similarity function

Figure 6. Test accuracy per epoch.

with the label of the most similar training sample; hence, we define the test accuracy as the number of correct labels in the test set divided by the number of test samples.

The test accuracy per epoch of the batch size, optimizing algorithm and similarity function experiments are shown in Figures 6a, 6b and 6c, respectively. We noted that the accuracy increases in all scenarios, which is expected behaviour since our loss is decreasing. The accuracy for larger batches is greater as expected. The Adam optimizer showed a greater accuracy when compared to the LARS optimizer. Finally, the cosine similarity presented a greater accuracy loss when compared to the dot similarity.

For our final model, the test accuracy per epoch is shown in Figure 5b. We observed that the accuracy increases; however, it could be higher if we trained our network with a larger batch size, more epochs and finely tuned our LARS optimizer (which achieves better results with larger batches as well).

5.3 Data transformations

To address the effects of different data augmentations, we trained our network for 10 epochs with different combinations of data transformations. We applied only two data transformations in the training step for each experiment and evaluated the test accuracy. The test accuracy for each combination is shown in Figure 7.

We realized that no single transformation was enough to learn good representations. In addition, we observed that the crop and resize operation is the most effective first transformation applied and the gaussian blur operation is the least effective. When comparing data augmentation pairs, the combination of crop resize and color jitter showed the best results. Therefore, gathering this information, we understand that some compositions of data augmentations are capable of improving the quality of the representations. Further, if we compose more than two transformations, we would achieve even better results, as happened in our final model.

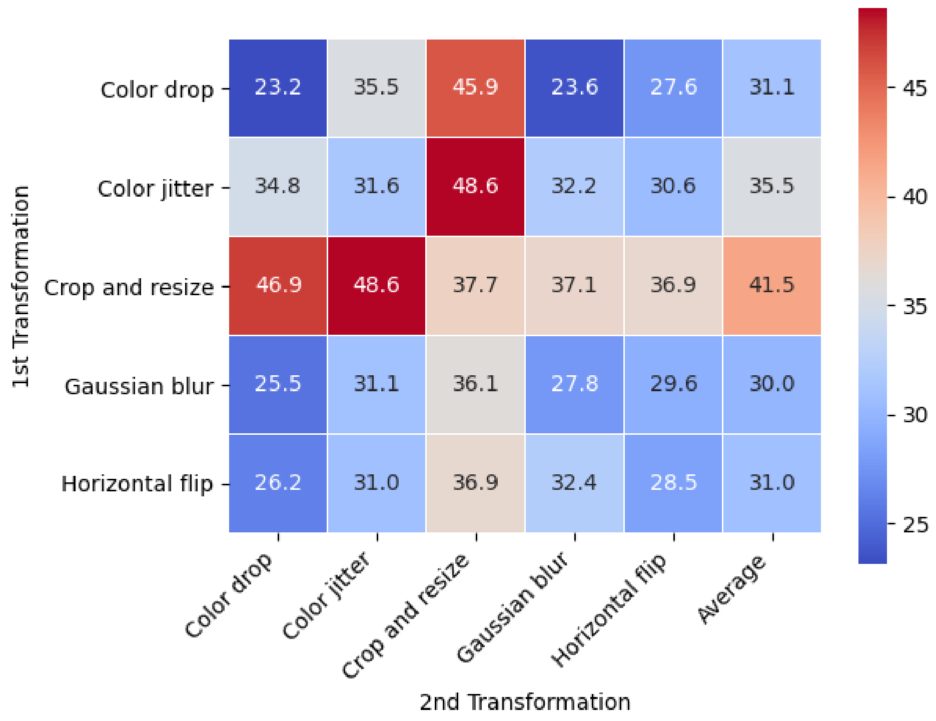


Figure 7. Top-1 test accuracy for different combination of data transformation. The diagonal corresponds to single transformations, and off-diagonals correspond to compositions of two transformations applied sequentially.

5.4 Linear Evaluation Protocol

We also tested our final model using a process called Linear Evaluation Protocol, which consists in using the output of the encoder to train a linear classifier using the image labels as output, evaluating the quality of the representations learned by SimCLR. We used a multi-class Logistic Regression algorithm, and the accuracy of the model over the data is used to measure the feature quality.

For this project, we created encoded representations of each image in dataset, and matched to its respective labels in dataset. The group of all combinations of representation and labels was separated in 75% for training the Logistic Regression and 25% for testing. We got the value of 77.19% of accuracy in training and 77.18% in test, showing that our SimCLR has a good feature quality, considering that we trained for 10% of the number of epochs of the original paper and the accuracy value of the paper is 90,6%.

6 Conclusion

The attempt to reproduce a state-of-the-art framework for contrastive learning of visual representations was an important step in our learning process in the field of unsupervised machine learning. The combination of data transformations used to produce the corresponding views are critical, which prevent trivial forms of agreement. The MLP-based non-linear projection helps to identify the invariant features of each input image and improves the representation quality. Processing more examples in the same batch lead to better performance. Hence, the improvement achieved by SimCLR framework over previous methods is not due to any single design choice, but to the combination of them. Our final model with the best design choices explored achieved an Top-1 accuracy in test of 74%. Finally, our reproduction of the SimCLR framework and showed that it is a strong and simple framework for doing further research in this direction and improve the state of self-supervised learning.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, 2020. eprint: `arXiv:2002.05709`.

[2] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: `1512.03385 [cs.CV]`.

[3] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. eprint: `arXiv:1412.6980`.

[4] Y. You, I. Gitman, and B. Ginsburg, *Large batch training of convolutional networks*, 2017. eprint: `arXiv:1708.03888`.