Project 2

Ivan Lima do Espirito Santo
Rosa Yuliana Gabriela Paccotacya Yanque
Thiago Gomes Marçal Pereira

I. Project Description

Learning effective visual representations without human supervision is a long-standing problem. Most mainstream approaches fall into one of two classes:

- generative that learn to model the input distribution
- discriminative that learn splitting hyperplanes to distinguish the classes.

Inspired by recent contrastive learning algorithms, SimCLR [1] learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space.

II. Goal

The goal of this project is to reproduce the results of SimCLR [1] for CIFAR10. In other words, implementing SimCLR's method and execute the experiments to compare:

- different augmentation functions T
- different similarity functions

We intent to experiment with different architecture's configurations (beyond the one presented in the paper) and perform further comparison. We might evaluate different transformation functions T to augment the data (section 3, [1]) and then compare their performance. Furthermore, we need to evaluate other similarities functions and explore how they change the method's performance. Regarding their architecture for the encoder $f()$ and the projection head $g()$. See Fig. 1 for a graphical representation. We explain the architecture in more details in the results session.

Section 3 to Section 6 presents a review of SimCLR [1] and subsequent sections shows our work.

III. SimCLR

SimCLR presents a simple framework for contrastive learning of visual representations. They simplify recently proposed contrastive selfsupervised learning algorithms without requiring specialized architectures or a memory bank.

In this work, they introduced a simple framework for contrastive learning of visual representations, which they called SimCLR. Not only did SimCLR outperform previous work, but it is also simpler, requiring neither specialized architectures nor a memory bank. In order to understand what enables good contrastive representation learning, they systematically study the major components of their framework and concluded that:

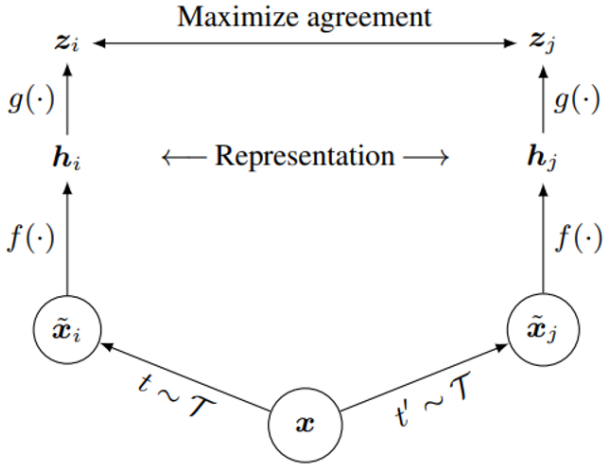


Fig. 1: SimCLR [[1], Fig. 2] framework. Two separate data augmentation operators are sampled from the same family of augmentations and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation h for downstream tasks

- Composition of multiple data augmentation operations is crucial in defining the contrastive prediction tasks.
- Introducing a learnable nonlinear transformation between the representation and the contrastive loss substantially improves the quality of the learned representations.
- Representation learning with contrastive cross entropy loss benefits from normalized embeddings and an appropriately adjusted temperature parameter.
- Contrastive learning benefits from larger batch sizes and longer training compared to its supervised counterpart. Like supervised learning, contrastive learning benefits from deeper and wider networks.

They combined these findings to achieve a new state-of-the-art in self-supervised and semi-supervised learning on ImageNet ILSVRC-2012 [4] as can be shown in Figure 2. Under the linear evaluation protocol, SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over previous state-of-the-art [2]. When fine-tuned with only 1% of the ImageNet labels, SimCLR achieves 85.8% top-5 accuracy, a relative improvement of 10% ([2]). When fine-tuned on other natural image classification datasets, SimCLR performs on par with or better than a strong supervised baseline [3] on 10 out of 12 datasets.

RA: 956694 - i956694@g.unicamp.br
RA: 263068 - r263068@dac.unicamp.br
RA: 189691 - t189691@g.unicamp.br

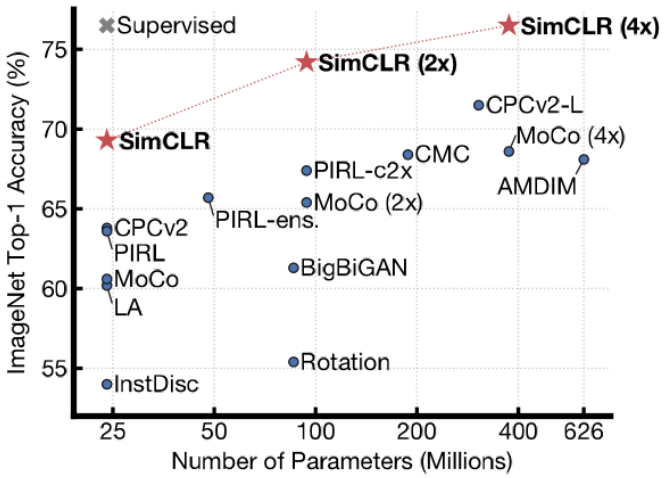


Fig. 2: ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold.

IV. SimCLR Method

A. The Contrastive Learning Framework

Inspired by recent contrastive learning algorithms, SimCLR learns representations by maximizing agreement between differently augmented views of the same data example via a contrastive loss in the latent space. As illustrated in Figure 1 this framework comprises the following four major components.

- A stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same example, denoted $\tilde{x}_i$ and $\tilde{x}_j$ which we consider as a positive pair. In this approach, they sequentially apply three simple augmentations: random cropping followed by resize back to the original size, random color distortions, and random Gaussian blur. The combination of random crop and color distortion is crucial to achieve a good performance.
- A neural network base encoder $f(\cdot)$ that extracts representation vectors from augmented data examples. Their framework allows various choices of the network architecture without any constraints. They opt for simplicity and adopt the commonly used ResNet to obtain $\boldsymbol{h}_i = f(\tilde{\boldsymbol{x}}_i) = \mathrm{ResNet}(\tilde{\boldsymbol{x}}_i)$ where $\boldsymbol{h}_i \in \mathbb{R}^d$ is the output after the average pooling layer.
- A small neural network projection head $g(\cdot)$ that maps representations to the space where contrastive loss is applied. They use a MLP with one hidden layer to obtain $z_i = g(\boldsymbol{h}_i) = W^{(2)}\sigma\left(W^{(1)}\boldsymbol{h}_i\right)$ where σ is a ReLU nonlinearity. They find it beneficial to define the contrastive loss on z_i 's rather than h_i 's.
- A contrastive loss function defined for a contrastive prediction task. Given a set $\{\tilde{\boldsymbol{x}}_k\}$ including a positive pair of examples $\tilde{x}_i$ and $\tilde{x}_j$, the contrastive prediction task aims to identify $\tilde{x}_j$ in $\{\tilde{x}_k\}_{k\neq i}$ for a given $\tilde{x}_i$

They randomly sample a minibatch of N examples and define the contrastive prediction task on pairs of augmented examples derived from the minibatch, resulting in $2N$ data points. They do not sample negative examples explicitly. Instead, given a positive pair, they treat the other $2(N-1)$ augmented examples within a minibatch as negative examples. Let $\mathrm{sim}(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}^\top\boldsymbol{v}/\|\boldsymbol{u}\|\|\boldsymbol{v}\|$ denote the cosine similarity between two vectors u and v Then the loss function for a positive pair of examples (i,j) is defined as:

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

where $\mathbb{1}_{[k\neq i]} \in \{0,1\}$ is an indicator function evaluating to 1 iff $k \neq i$ and τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i,j) and (j,i), in a mini-batch. This loss has been used in previous work [5], [7], [6]. The following Algorithm summarizes the proposed 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** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N}\sum_{k=1}^{N}\left[\ell(2k-1,2k) + \ell(2k,2k-1)\right]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

B. Training with Large Batch Size

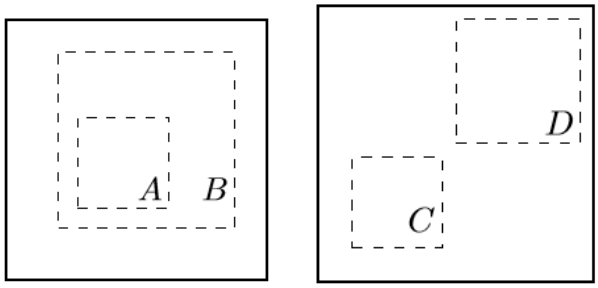
They do not train the model with a memory bank ([7]). Instead, they vary the training batch size N from 256 to 8192. A batch size of 8192 gives them 16382 negative examples per positive pair from both augmentation views. Training with large batch size may be unstable when using standard SGD/Momentum with linear learning rate scaling. To stabilize the training, they use the LARS optimizer for all batch sizes. They train their model with Cloud TPUs, using 32 to 128 cores depending on the batch size. [2]

Global BN. Standard ResNets use batch normalization. In distributed training with data parallelism, the BN mean and

variance are typically aggregated locally per device. In their contrastive learning, as positive pairs are computed in the same device, the model can exploit the local information leakage to improve prediction accuracy without improving representations. They address this issue by aggregating BN mean and variance over all devices during the training.

C. Evaluation Protocol

The protocol for their empirical studies aim to understand different design choices in their framework.



(a) Global and Local Views (b) Adjacent Views

Fig. 3: Solid rectangles are images, dashed rectangles are random crops. By randomly cropping images, they sample contrastive prediction tasks that include global to local view $(B \rightarrow A)$ or adjacent view $(D \rightarrow C)$ prediction.

Dataset and Metrics. Most of their study for unsupervised pretraining (learning encoder network f without labels) is done using the ImageNet ILSVRC-2012 dataset. They also test the pretrained results on a wide range of datasets for transfer learning. To evaluate the learned representations, they follow the widely used linear evaluation protocol, where a linear classifier is trained on top of the frozen base network, and test accuracy is used as a proxy for representation quality. Default setting. For data augmentation they use random crop and resize (with random flip), color distortions, and Gaussian blur. They use ResNet-50 as the base encoder network, and a 2 -layer MLP projection head to project the representation to a 128-dimensional latent space. As the loss, They use NT-Xent, optimized using LARS with linear learning rate scaling (i.e. LearningRate $= 0.3\times$ BatchSize $/256$) and weight decay of 10^{-6}. They train at batch size 4096 for 100 epochs. [3] Furthermore, they use linear warmup for the first 10 epochs, and decay the learning rate with the cosine decay schedule without restarts.

V. Data Augmentation for Contrastive Representation Learning

Data augmentation defines predictive tasks. While data augmentation has been widely used in both supervised and unsupervised representation learning, it has not been considered as a systematic way to define the contrastive prediction task. Many existing approaches define contrastive prediction tasks by changing the architecture.

Some researchers achieve global-to-local view prediction via constraining the receptive field in the network architecture, whereas others achieve neighboring view prediction via a fixed image splitting procedure and a context aggregation network. This complexity can be avoided by performing simple random cropping (with resizing of target images, which creates a family of predictive tasks subsuming the above mentioned two, as shown in Figure 3. This simple design choice conveniently decouples the predictive task from other components such as the neural network architecture. Broader contrastive prediction tasks can be defined by extending the family of augmentations and composing them stochastically.

A. Composition of data augmentation operations is crucial for learning good representations

To systematically study the impact of data augmentation, they consider several common augmentations. One type of augmentation involves spatial/geometric transformation of data, such as cropping and resizing (with horizontal flipping), rotation and cutout. The other type of augmentation involves appearance transformation, such as color distortion (including color dropping, brightness, contrast, saturation, hue), Gaussian blur, and Sobel filtering. Figure 11 visualizes the augmentations studied in SimCLR work.

To understand the effects of individual data augmentations and the importance of augmentation composition, they investigate the performance of our framework when applying augmentations individually or in pairs. Since ImageNet images are of different sizes, they apply crop and resize images, which makes it difficult to study other augmentations in the absence of cropping. To eliminate this confound, they consider an asymmetric data transformation setting for this ablation. Specifically, they always first randomly crop images and resize them to the same resolution, and then apply the targeted transformation(s) only to one branch of the framework in Figure 1 while leaving the other branch as the identity (i.e. $t\left(x_i\right)=x_i$). This asymmetric data augmentation hurts the performance. Nonetheless, this setup should not substantively change the impact of individual data augmentations or their compositions.

Figure 4 shows linear evaluation results under individual and composition of transformations. They observe that no single transformation suffices to learn good representations, even though the model can almost perfectly identify the positive pairs in the contrastive task. When composing augmentations, the contrastive prediction task becomes harder, but the quality of representation improves dramatically. One composition of augmentations stands out: random cropping and random color distortion. They conjecture that one serious issue when using only random cropping as data augmentation is that most patches from an image share a similar color distribution. Figure 5 shows that color histograms alone suffice to distinguish images. Neural nets may exploit this shortcut to solve the predictive task. Therefore, it is critical to compose cropping with color distortion in order to learn generalizable features.

B. Contrastive learning needs stronger data augmentation than supervised learning

To further demonstrate the importance of the color augmentation, they adjust the strength of color augmentation as

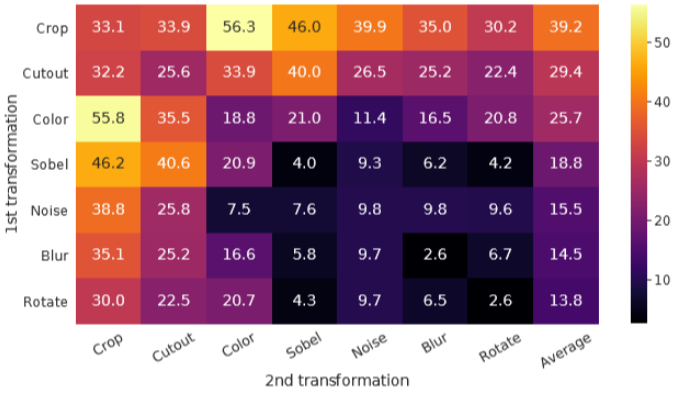
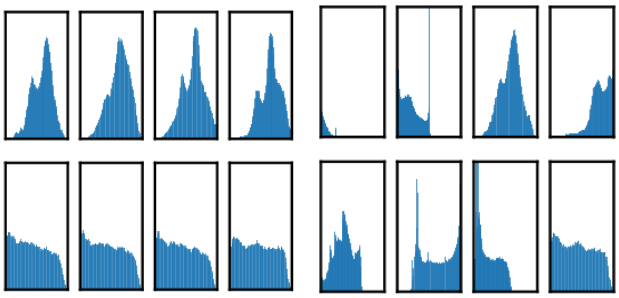


Fig. 4: Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch. For all columns but the last, diagonal entries correspond to single transformation, and off-diagonals correspond to composition of two transformations (applied sequentially). The last column reflects the average over the row.



(a) Without color distortion (b) With color distortion

Fig. 5: Histograms of pixel intensities (over all channels) for different crops of two different images (i.e. two rows). The image for the first row is from figure 11. All axes have the same range.

shown in Table I. Stronger color augmentation substantially improves the linear evaluation of the learned unsupervised models. In this context, a sophisticated augmentation policy found using supervised learning, does not work better than simple cropping + (strong) color distortion. When training supervised models with the same set of augmentations, they observe that stronger color augmentation does not improve or even hurts their performance. Thus, their experiments show that unsupervised contrastive learning benefits from stronger (color) data augmentation than supervised learning. Although previous work has reported that data augmentation is useful for self-supervised learning, they show that data augmentation that does not yield accuracy benefits for supervised learning can still help considerably with contrastive learning.

VI. Architectures for Encoder and Head

A. Unsupervised contrastive learning benefits(more) from bigger models

Figure 6 shows that increasing depth and width both improve performance. While similar findings hold for supervised learning, they find the gap between supervised models and linear classifiers trained on unsupervised models shrinks as the model size increases, suggesting that unsupervised learning benefits more from bigger models than its supervised counterpart.

Methods	Color distortion strength 1/8	1/4	1/2	1	1(+ Blur)	AutoAug
SimCLR	59.6	61.0	62.6	63.2	64.5	61.1
Supervised	77.0	76.7	76.5	75.7	75.4	77.1

TABLE I: Top-1 accuracy of unsupervised ResNet-50 using linear evaluation and supervised ResNet- 50^5, under varied color distortion strength and other data transformations. Strength 1 (+ Blur) is their default data augmentation policy.

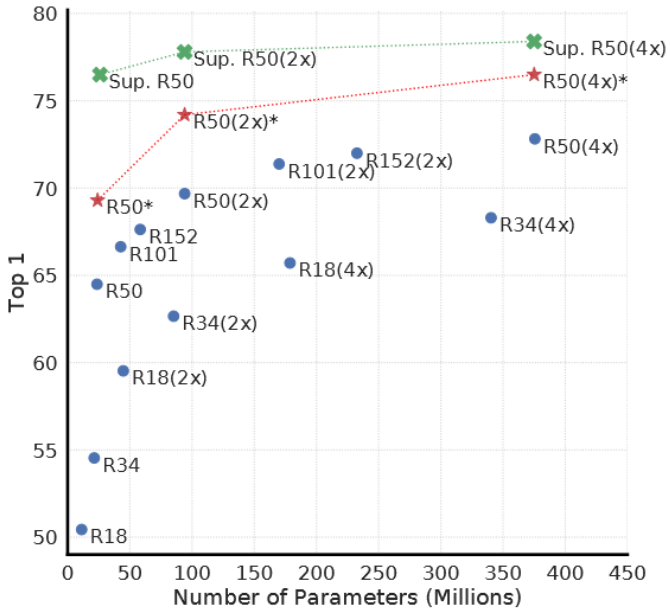


Fig. 6: Linear evaluation of models with varied depth and width. Models in blue dots are their trained for 100 epochs, models in red stars are their trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs.

B. A nonlinear projection head improves the representation quality of the layer before it

They also study the importance of including a projection head, i.e. $g(h)$. Figure 7 shows linear evaluation results

Table II shows h contains much more information about the transformation applied, while $g(h)$ loses information.

C. Loss Functions and Batch Size

Normalized cross entropy loss with adjustable temperature works better than alternatives. Table IV shows the objective function as well as the gradient to the input of the loss function. Table V shows that, while (semi-hard) negative mining helps, the best result is still much worse than our default NT-Xent loss. Table VI shows that without normalization and proper temperature scaling, performance is significantly worse.

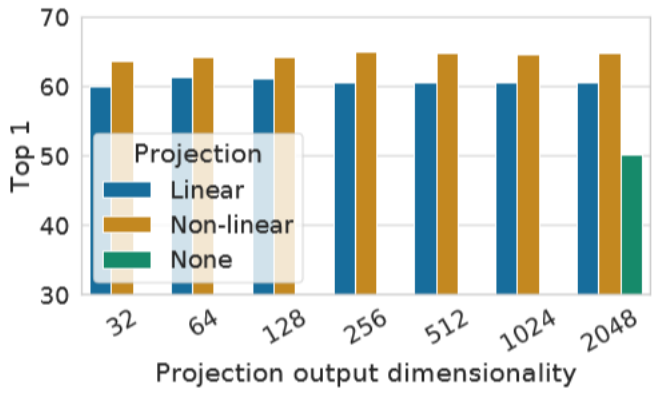


Fig. 7: Linear evaluation of representations with different projection heads go and various dimensions of $z = g(h)$. The representation h (before projection) is 2048-dimensional here.

What to predict?	Random guess	Representation $\boldsymbol{h}$	$g(\boldsymbol{h})$
Color vs grayscale	80	99.3	97.4
Rotation	25	67.6	25.6
Orig. vs corrupted	50	99.5	59.6
Orig. vs Sobel filtered	50	96.6	56.3

TABLE II: Accuracy of training additional MLPs on different representations to predict the transformation applied. Other than crop and color augmentation, we additionally and independently add rotation (one of $\{0°, 90°, 180°, 270°\}$), Gaussian noise, and So bel filtering transformation during the pretraining for the last three rows. Both h and $g(h)$ are of the same dimensionality, i.e. 2048

Contrastive learning benefits(more) from larger batch sizes and longer training (Figure 8).

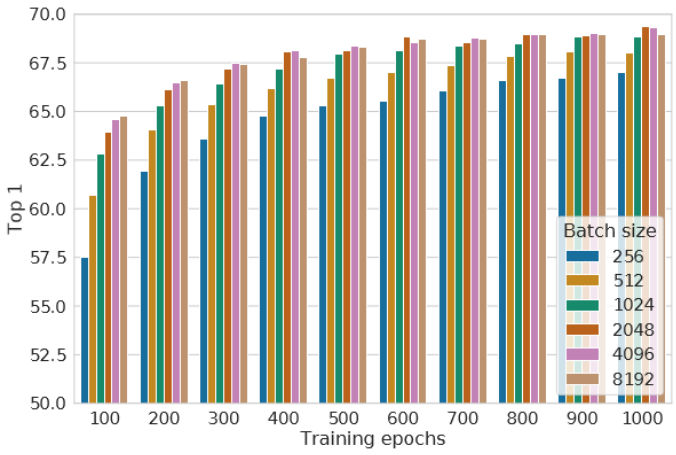


Fig. 8: Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.

D. Comparison with State-of-the-art and SimCLR Conclusion

Table III shows the comparisons of SimCLR results against recent methods. The approach shows significantly improves over state-of-the-art with both 1% and 10% of the labels.

Method	Architecture	Label fraction 1%	10%
		Top 5	
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	–	88.5
FixMatch (w. RandAug)	ResNet-50	–	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	–	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161 (*)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet- 50(4×)	**85.8**	**92.6**

TABLE III: ImageNet accuracy of models trained with few label

They evaluate transfer learning performance across 12 natural image datasets in both linear evaluation (fixed feature extractor) and fine-tuning settings. Following [3], they perform hyperparameter tuning for each model-dataset combination and select the best hyperparameters on a validation set.

VII. Development Stages

To reproduce SimCLR, we worked trough different stages:

- Dataset Acquisition
- ResNet50 Modification
- Loss Implementation
- SimCLR Implementation

VIII. Difficulties

The main difficulty in this project was to modify the Model. At a first approach, we opted to use Keras, because we have more familiarity with this framework. But to modify the model layers, to be equivalent with the one proposed by [1], we would have to almostly recreate the model. So, we opted to use PyTorch, and this becomes a new challenge as we are not so familiar.

IX. Adopted Approach

After defining the framework to be used, we divided the implementation in simple parts.

A. Dataset Acquisition

Even though Cifar10 can be easily retrieved from PyTorch datasets, for this project, we needed the dataset to be compounded by two different transformations of the original image.
So, we created a class that inherits from CIFAR10 model that returns both transformed images as the items.

B. ResNet50 Modification

The paper proposed changes to ResNet50 to be applied to Cifar10 dataset, like:

- Change stride and kernel from first convolutional layer
- Remove the first MaxPooling layer

To remove the MaxPooling layer, we replaced it with a Conv1D layer, with kernel size 1, to act just like a ”bypass” layer. Output will be the same as the input.
We also added linear projection layers at the end, to help with the loss calculation.

C. Loss Implementation

The paper suggests the usage of NT-Xent loss, to provide better results. As it's not normal in the framework, we had to implement it by hand using equation1.

D. SimCLR Implementation

In this final implementation phase, we join the parts together, following the framework shown in figure 1.
For this purpose, we split our data in training set and validation set and define a number of epochs and batch size as well.
Subsequently, we pass both transformed images into the model, and calculate the losses from the output. We use this information to optimize our model, and after each epoch, we calculate the loss in the validation dataset.

X. Results

The loss decays throughout the epochs. It means that the model is learning in a self-supervised manner.
We could not run experiments with different datasets or in the same dataset with different batch sizes and parameters. These attempts might improve the learning performance.
Running in CPUs, one viable comparison we make is reducing the number of images in the dataset. This experiment shows a lower loss and a higher decay. However, it might be caused by some overriding in the model.

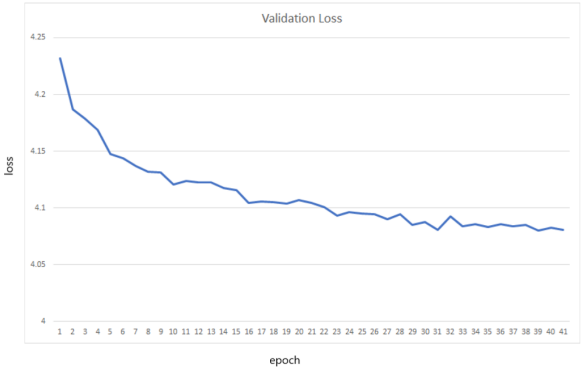


Fig. 9: Loss versus number of epoch

As one can see, due to the size of the images in the dataset, after the transformations, some images might loose all the information in it, affecting the final results. The loss decay is shown in figure 9

XI. Conclusion

Semi-supervised models are getting attention of researchers and state-of-the-art models are even showing results that overcome fully supervised learning models.
This project presented us with an easy and also very interesting solution and we could get good results, yet not better than the ones in the paper.
It also opens space for many approaches to try to get better results that could be part of future work and research, like:

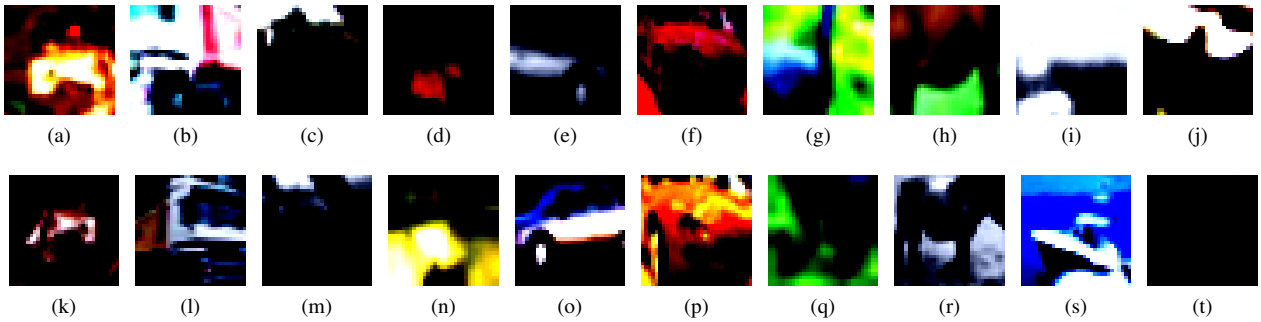
- trying different modifications in ResNet50
- try different base networks
- Explore Datasets and variations of parameters

We could also even try to use this approach as an input for labeling unlabeled data.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.
[2] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019.
[3] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? *CoRR*, abs/1805.08974, 2018.
[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
[5] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016.
[6] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.
[7] Z. Wu, Y. Xiong, S. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance-level discrimination. *CoRR*, abs/1805.01978, 2018.

APPENDIX A
PROJECT RESULTS



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

(k) (l) (m) (n) (o) (p) (q) (r) (s) (t)

Fig. 10: results

APPENDIX B
SIMCLR PICTURES

(a) Original (b) Crop and Resize (c) Crop, resize and flip (d) Color dist drop (e) Color dist jitter

(f) Rotate (g) Cutout (h) Gaussian noise (i) i (j) Sobel Filtering

Fig. 11: Illustrations of the studied data augmentation operators. Each augmentation can transform data stochastically with some internal parameters (e.g. rotation degree, noise level). Note that we only test these operators in ablation, the augmentation policy used to train our models only includes random crop (with flip and resize), color distortion, and Gaussian blur. (Original image cc-by: Von.grzanka)

APPENDIX C
SIMCLR TABLES ANDBENCHMARK

Name	Negative loss function	Gradient w.r.t. $\boldsymbol{u}$
NT-Xent	$\boldsymbol{u}^T\boldsymbol{v}^+/\tau - \log \sum_{\boldsymbol{v}\in\{\boldsymbol{v}^+,\boldsymbol{v}^-\}} \exp\left(\boldsymbol{u}^T\boldsymbol{v}/\tau\right)$	$\left(1 - \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^+/\tau)}{Z(\boldsymbol{u})}\right)/\tau\boldsymbol{v}^+ - \sum_{\boldsymbol{v}^-} \frac{\exp(\boldsymbol{u}^T\boldsymbol{v}^-/\tau)}{Z(\boldsymbol{u})}/\tau\boldsymbol{v}^-$
NT-Logistic	$\log\sigma\left(\boldsymbol{u}^T\boldsymbol{v}^+/\tau\right) + \log\sigma\left(-\boldsymbol{u}^T\boldsymbol{v}^-/\tau\right)$	$\left(\sigma\left(-\boldsymbol{u}^T\boldsymbol{v}^+/\tau\right)\right)/\tau\boldsymbol{v}^+ - \sigma\left(\boldsymbol{u}^T\boldsymbol{v}^-/\tau\right)/\tau\boldsymbol{v}^-$
Margin Triplet	$-\max\left(\boldsymbol{u}^T\boldsymbol{v}^- - \boldsymbol{u}^T\boldsymbol{v}^+ + m, 0\right)$	$\boldsymbol{v}^+ - \boldsymbol{v}^-$ if $\boldsymbol{u}^T\boldsymbol{v}^+ - \boldsymbol{u}^T\boldsymbol{v}^- < m$ else $\mathbf{0}$

TABLE IV: Negative loss functions and their gradients. All input vectors, i.e. u, v^+, v^-, are ℓ_2 normalized. NT-Xent is an abbreviation for "Normalized Temperature-scaled Cross Entropy". Different loss functions impose different weightings of positive and negative examples

Margin	NT-Logi.	Margin (sh)	NT-Logi.(sh)	NT-Xent
50.9	51.6	57.5	57.9	63.9

TABLE V: Linear evaluation (top-1) for models trained with different loss functions. "sh" means using semi-hard negative mining

ℓ_2 norm?	τ	Entropy	Contrastive acc.	Top 1
Yes	0.05	1.0	90.5	59.7
	0.1	4.5	87.8	64.4
	0.5	8.2	68.2	60.7
	1	8.3	59.1	58.0
No	10	0.5	91.7	57.2
	100	0.5	92.1	57.0

TABLE VI: Linear evaluation for models trained with different choices of ℓ_2 norm and temperature τ for NT -Xent loss. The contrastive distribution is over 4096 examples

Method	Architecture	Param.	Top 1	Top 5
Methods using ResNet-50:				
Local Agg.	ResNet-50	24	60.2	–
MoCo	ResNet-50	24	60.6	–
PIRL	ResNet-50	24	63.6	–
CPC v2	ResNet-50	24	63.8	85.3
SimCLR (ours)	ResNet-50	24	**69.3**	**89.0**
Methods using other architectures:				
Rotation	RevNet-50 $(4\times)$	86	55.4	–
BigBiGAN	RevNet-50 $(4\times)$	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	–
CMC	ResNet-50 $(2\times)$	188	68.4	88.2
MoCo	ResNet-50 $(4\times)$	375	68.6	–
CPC v2	ResNet-161 $(*)$	305	71.5	90.1
SimCLR (ours)	ResNet-50 $(2\times)$	94	74.2	92.0
SimCLR (ours)	ResNet-50 $(4\times)$	375	**76.5**	**93.2**

TABLE VII: ImageNet accuracies of linear classifiers trained on representations learned with different self-supervised methods