Reproducing[*] and Analyzing Results of SimCLR:[1] A Simple Framework for Contrastive Learning of Visual Representations

Avital Rose

August 2024

1 Brief Summary of the Paper

SimCLR stands for Simple Framework for Contrastive Learning of Visual Representations. SimCLR is an unsupervised approach that trains via contrastive loss, model weights for future use in separate downstream computer vision tasks, such as classification or detection. It does this by focusing on the task of identifying augmented views of the same image among a batch of unrelated images. The paper further identifies key features of the framework that enhance the performance. These features include (1) composition of data augmentations (2) a learnable nonlinear transformation between the representation and the contrastive loss and (3) contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning.
The framework includes the following steps illustrated in Figure 1.

1. **A Data Augmentation Module:** which transforms a given image randomly in two ways, yielding two correlated views of the same example. Augmented images from the same source image are counted to be positive pairs, and if the augmented images are derived from different source images they are considered negative. The augmentations include - Crop, Gaussian blur, Color distortion, Gaussian noise, Rotation, Sobel filtering, and Cutout.

2. **A Base Encoder:** such as the ResNet-50 architecture (although you can use almost any encoder) to extract representation vectors from augmented data examples. This is the model whose weights you train for the eventual downstream task.

3. **A Projection Head:** which is an MLP (with one hidden layer) that just makes the representation vectors smaller before the loss function.

4. **A Contrastive Loss Function:** which is low if the two correlated ”positive” vectors (from the same original image) are similar to each other AND dissimilar to the vectors from a set of “negative” examples (all the other examples in the mini-batch; not from the same original example), and high otherwise. I.e. the contrastive loss aims to create similar representations of positive pairs.

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

where

- N: Number of image pairs.
- $\text{sim}(z_i, z_j)$: Similarity score between embeddings z_i and z_j.
- τ: Temperature parameter.
- z_i, z_k: Embedding vectors for images.

[*]Code used is from: https://github.com/Spijkervet/SimCLR

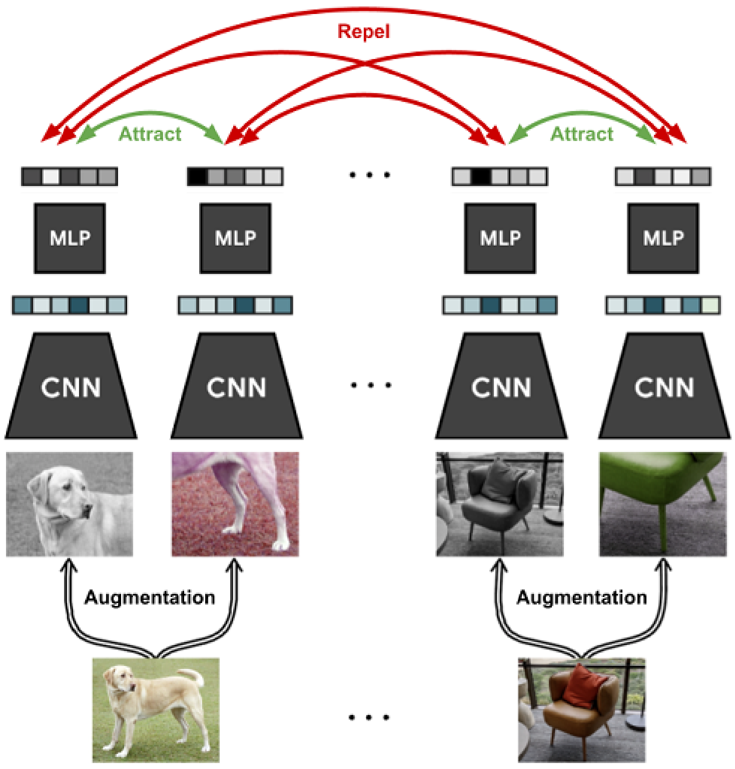


Figure 1: SimCLR framework illustration. Bottom to Top: Data Augmentation (e.g. cropping, blurring, etc.), Base Encoder (e.g. Resnet-50), MLP Projection head, and Contasrtive Loss Function.

The paper used the CIFAR10 dataset[1]. The framework can be trained from start to end or can utilize pre-trained base encoders and fine-tune above them. The paper utilizes several ResNet models as a base model (e.g. ResNet 18, ResNet 50). The evaluation is done by adding a linear layer above the representation and then classifying and computing the accuracy[2]. The paper claims that "A linear classifier trained on self-supervised representations learned by SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over previous state-of-the-art, matching the performance of a supervised ResNet-50. When fine-tuned on only 1% of the labels, we achieve 85.8% top-5 accuracy, outperforming AlexNet with 100× fewer labels".

2 Description of the Conceptes Attempted for Replication

Since reproducing SimCLR requires heavy computation resources that are not available (With 128 TPU v3 cores, it takes ~1.5 hours to train the ResNet-50), in this analysis we focus on the concepts SimCLR introduces rather than the final results. Furthermore, a pre-trained ResNet18 was used to reproduce results rather than training start-to-end. SimCLR identifies three main components that enhance the performance.

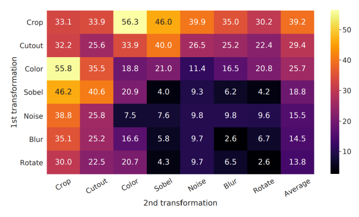
1. Composition of data augmentations matters, for example, cropping alone does not suffice for learning a good representation.
2. A learnable nonlinear transformation between the representation and the contrastive loss performs much better than a linear transformation.
3. Contrastive learning benefits from larger batch sizes and more training steps compared to supervised learning.

[1]https://www.cs.toronto.edu/ kriz/cifar.html

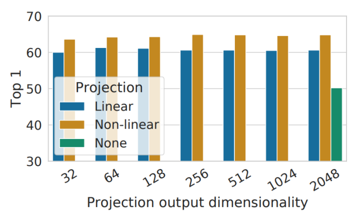
[2]Performance is measured as Top-1 accuracy which is the conventional accuracy, model prediction (the one with the highest probability) must be exactly the expected answer. It measures the proportion of examples for which the predicted label matches the single target label.

3 Original Results from Paper

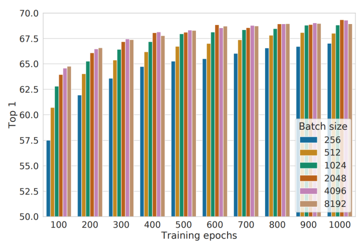
The results from the paper (Figure 2) support the concepts stated in the previous section.



(a)



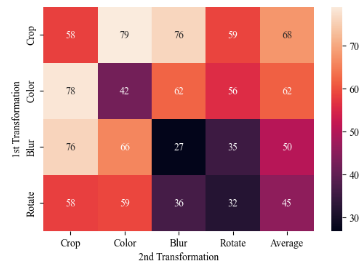
(b)



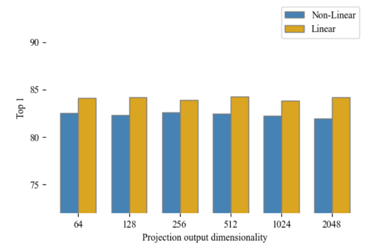
(c)

Figure 2: **Original Graphs From Paper**. **(a) Data Augemtations** Linear evaluation 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. **(b) Projection Head** Linear evaluation of representations with different projection heads and various dimensions. **(c) Batch Size** Linear evaluation models (ResNet-18) trained with different batch size and epochs. Each bar is a single run from scratch.

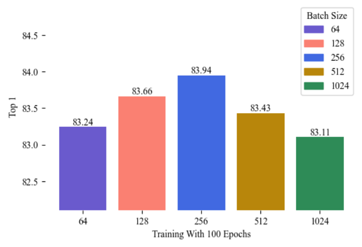
4 Reproduced Results



(a)



(b)



(c)

Figure 3: **Regenearted Results. (a) Data Augemtations** Same as original, but for a subset of augmentations. Trends match the original paper trends. **(b) Projection Head** Same as original, trends match the original results. **(c) Batch Size** Different batch size range and only for 100 epochs, trends partially match the original.

5 Discussion

Some of the reproduced results (Figure 3) match the original while others differ. For data augmentations (Figure 3 (a)), we focused on 4/9 augmentations tested in the paper - ”Crop”, ”Color”, ”Blur” and ”Rotate”, which are the augmentations provided in the code. The trends match the paper, yet the reproduced accuracy is significantly higher (probably due to using the pre-trained network). A few notable trends - the order of the average accuracy for every single augmentation matched the original - ”Crop”, ”Color”, ”Blur” and ”Flip” in descending order. It is also worth noting that the highest performance is achieved when ”Crop” and ”Color” are combined, aligning with the findings in the paper. While the paper provides an in-depth analysis of this phenomenon, it is beyond the scope of our discussion.
Second, the projection head results (Figure 3 (b)) trend matches the trend from the paper - a non-linear head performs better than a linear one (although the ranges are different, the new accuracy is much higher than the original one). Lastly, for the batch size analysis (Figure 3 (c)), we tested only the 100 epochs option, and in a different batch range, due to memory and computation restrictions. The recreated batch size analysis partially matched the original, and the accuracies are once again higher

than the original and also much closer to each other. In the smaller range not in the paper (64, 128), the trend matched - larger batch size leads to better performance. In the range from the paper (256, 512, 1024), the results are the opposite, the bigger the batch size, the lower the performance. This is typical behavior for a supervised task, which may explain why the results differ.
To explain the variation between the regenerated results and the original results, we can first consider the option of a faulty code or faulty execution. Although this possibility cannot entirely be ruled out, it is improbable due to the consistency of the results, and a completely faulty recreation would end up with random results, not concise ones. Also, to further rule out a flawed code implementation, a second git repository[3] was tested, and the batch size trend remained opposite to the paper. Another possibility is that the specific hyperparameters for running the code were incorrect (We could not find them properly in the paper). An alternate explanation is that since we used pre-trained models and the majority of the training was done before, the fine-tuning is not changing much, and the setup is closer to the supervised setting where smaller batch sizes are superior. Lastly, the restricted computation resources can decrease the results, and larger batch sizes require more resources.

6 Challenges in Replication: How Authors Could Have Made It Easier?

There were several challenges encountered while regenerating the paper.

1. **Computation Resources**: The heavy resources required to run the models described in the paper were not available. Therefore, we could not run the models end-to-end, and can only run on pre-trained models (and only small pre-trained models, ResNet-18 and not Resnet-50). Furthermore, the number of experiments was limited, and the scope was limited (i.e., can compute up to batch size 2048, only 100 epochs, etc). This is not something the authors can make available to us.

2. **Correct Code**: The original code[4] provided by the paper uses TensorFlow, and is outdated. There is a newer code also provided, which is not fully compatible with the original code or with the Cuda version in our servers, which we were not able to change. We decided to use an external PyTorch implementation. The authors of the paper are from Google, therefore it is highly unlikely that they will use PyTorch which is easier to handle.

3. **Correct Parameters**: Some of the exact parameters for the experiments were unclear- for example the correct projection head when studying the effect of the batch sizes. Also, as stated in section 1, to evaluate the representation, a linear head is attached above, and then it was not clear in the code implementation what should be the batch size at this point. We tried several approaches which all ended up with the same results. A more clear description under the graph would have been optimal (or some other place).

7 Summary

SimCLR offers a contrastive self-supervised learning approach and highlights important aspects to improve the performance - correct selection of data augmentations, adding a non-linear head, and larger batch sizes. The regeneration of the results was pretty straightforward, but a recurring theme of running into resource limitations highlights the problem of ideas proposed by tech giants- other than them, no entity can run the models. This seriously limits the ability to independently verify the concepts and poses real restrictions on the innovation race. In the scope of our reproduction, despite the computation restrictions, we were able to see and understand how the concepts work, and then hopefully apply them to our work in the future. While SimCLR does not introduce a novel algorithm, it does emphasize how meticulous architecture planning including parameter choices can greatly impact performance.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[3]https://github.com/sthalles/SimCLR
[4]https://github.com/google-research/simclr