A Simple Framework for Contrastive Learning of Visual Representations

E6691.2024Fall.SAMH.report.SH4635
Columbia University

Abstract

In this project, I attempted to reproduce some of the results of the paper "A Simple Framework for Contrastive Learning of Visual Representations." The paper introduces the SimCLR framework, which is targeted to improve the representations created by models with contrastive prediction tasks. The authors of the paper performed various experiments and used their findings to create an optimal contrastive learning framework for visual representations called SimCLR. The main goal I pursue in this project is to reproduce the two key findings in the original paper: (1) composition of data augmentations plays a critical role in defining effective predictive tasks and (2) introducing a learnable nonlinear transformation between the representation and the contrastive loss substantially improves the quality of the learned representations. The two key findings were then combined to create a final contrastive learning framework for visual representations called SimCLR. Due to memory limitations, the outcomes of the project were not similar to the ones seen in the original paper.

1. Introduction

Conventionally, machine learning algorithms relied on labels to learn and perform various classification tasks. However, annotating data can be a really expensive and time-consuming process. As a solution to this, many researchers have been exploring a different approach to training machine learning models: self-supervised learning. In self-supervised learning, an unsupervised learning model uses different techniques to understand unlabeled data. This is usually done by performing various transformations on the raw data samples and attempting to extract meaningful features from them.

A way of achieving this is using contrastive learning. In contrastive learning, positive and negative pairs of data samples are composed and the model is trained to distance the representations of the negative pair and bring together the positive pair samples.

In the paper chosen in the project, "A Simple Framework for Contrastive Learning of Visual Representations", a new contrastive learning framework called SimCLR is introduced which is targeted at visual representations. In SimCLR, a set of random augmentation functions are selected to transform each image to two different representations. These representations are called the positive pair which the model strives to bring together. As these representations are the same thing, the model is pushed toward understanding critical information about the pictures without actually learning their labels. The other pictures in the training batch are also transformed and are then put into negative pairs with each other. The loss function enforces the model to treat these samples dissimilar to each other.

The original paper aims to introduce an enhanced contrastive learning framework based on their two findings: (1) the optimal data augmentation functions play an important role on the performance of the model, and (2) a nonlinear projection head could improve the quality of the representation produced by the original model. In this project, I aim to reproduce these two important experiments and compare my results with the ones reported in the original paper. The main issue I face is the system limitation. The dataset that the original paper uses has over 50000 high quality pictures and about 1000 classes [4]. Moreover, training a large model such as ResNet-50 could be challenging and time consuming. As a result, I will only use a subset of the original dataset and train my models for fewer epochs.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

2.1.1 Overview

This paper introduces SimCLR, a contrastive learning framework for visual representation learning. The authors achieve this goal by maximizing the agreement between various representations of the same picture (data example.) Figure 1 illustrates how this framework works in detail. As we can see, the same data sample (which is an image in this paper) is put through different augmentations (explained in part 2.1.2) and passed through a base encoder model f(.), which is a pre-trained Resnet-50. The representations (hi) are then passed through a projection head which could be a linear or non-linear model denoted as g(.). The final outputs (zi) are then fed to a loss function (elaborated in part 2.1.3) that pushes the network to maximize the agreement between them.

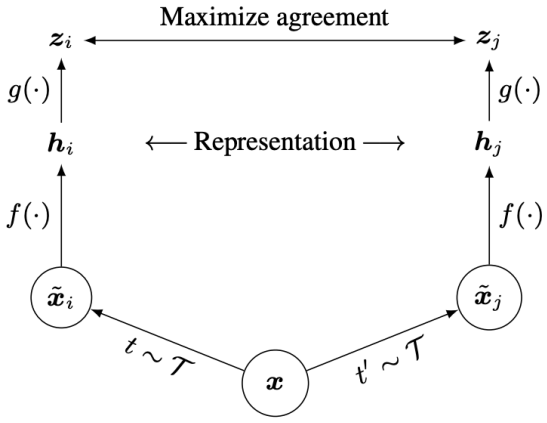

Figure 1: SimCLR framework

2.1.2 Data Augmentations

This paper uses a variety of stochastic data augmentation modules that produce a random output every time they are applied to a picture. For example, a random horizontal flipping function is applied to the images that flips each data with a probability of 50 percent each time it is called. This way, the same augmentation pipeline can produce different outputs from the same picture. These outputs are later on fed to a loss function (explained in part 2.1.3) that gives higher values if the two representations of the same picture are less similar. Many augmentation techniques were used in the paper and their effect on the performance was studied extensively. More information can be seen in part 2.2.2.

Overall the default setting that they use in the experiments (unless indicated otherwise) is as follows:

- random cropping followed by resizing to original size with random horizontal flip
- Color distortions
- Gaussian blur

As explained in the appendix of the paper, the random cropping is done with a random area size of 0.08 to 1.0 (uniform distribution is used to determine this value) and a random aspect ratio of 3/4 to 4/3 (also from a uniform distribution) of the original aspect ratio. After this step, the cropped image is restored to the original size and is always followed by a random horizontal flip with 50% probability.

```
import tensorflow as tf
def color_distortion(image, s=1.0):
    # image is a tensor with value range in [0, 1].
    # s is the strength of color distortion.

    def color_jitter(x):
        # one can also shuffle the order of following augmentations
        # each time they are applied.
        x = tf.image.random_brightness(x, max_delta=0.8*s)
        x = tf.image.random_contrast(x, lower=1-0.8*s, upper=1+0.8*s)
        x = tf.image.random_saturation(x, lower=1-0.8*s, upper=1+0.8*s)
        x = tf.image.random_hue(x, max_delta=0.2*s)
        x = tf.clip_by_value(x, 0, 1)
        return x

    def color_drop(x):
        image = tf.image.rgb_to_grayscale(image)
        image = tf.tile(image, [1, 1, 3])

    # randomly apply transformation with probability p.
    image = random_apply(color_jitter, image, p=0.8)
    image = random_apply(color_drop, image, p=0.2)
    return image
```

Figure 2: Color distortion pseudo-code

The color distortion is also described in detail in the appendix, in addition to a pseudo-code (figure 2) which was used in the project implementation.
According to the Appendix of the original paper, gaussian blurring is done using a gaussian kernel with a 50 percent chance of application. Sigma is sampled from a uniform distribution from 0.1 to 2.0 and kernel size is set to be 0.1 of the image size.

2.1.3 Contrastive Loss

The contrastive loss function used in this paper can be explained as a normalized temperature-scaled cross entropy loss. The function (can be seen in figure 3) uses cosine similarity for the similarity function sim(zi, zj) and then calculates the negative logarithm of the likelihood of zi and zj's similarity. For calculating the likelihood, the exponential of the scaled similarity is divided by the similarity of one of the pictures with other pictures in the batch. The pairs zi and zj are called a positive pair since they are the representations of the same image. All other combinations of zi with other zk values are called a negative pair. The sum of scaled similarity between all the negative pairs in the denominator ensures that the loss function pushes the network to make the samples in these pairs less similar with each other.

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Figure 3: Contrastive loss formula

2.1.4 Projection Head

Projection head is a small neural network projection head g(·) that maps representations to the space where contrastive loss is applied. This simple network is shown to be effective in improving the performance of the models and is applied after the average pooling layer in our encoder (which is ResNet-50 for the sake of simplicity.) By default, the projection head contains an

MLP with one hidden layer and an activation function of ReLU which is applied after the first layer. The outputs of this projection head are called zi and are fed to the contrastive loss function, instead of the hi values which are the representations produced by the encoder.

Throughout the experiments, a 2-layer MLP projection head is used to project the representation to a 128-dimensional latent space.

2.1.5 Evaluation Protocols

The dataset used in the paper for unsupervised pre-training is the ImageNet ILSVRC-2012 dataset which is short for **I**mageNet **L**arge-**S**cale **V**isual **R**ecognition **C**hallenge, published in 2012. The pre-trained results are then tested on a range of datasets for transfer learning. Moreover, to evaluate the learned representations, the authors follow the linear evaluation protocol.

2.1.5.1 The Linear Evaluation Protocol

This protocol measures the quality of the representations produced by pretrained models using a widely known protocol called Linear Evaluation. In this protocol, a linear classifier is trained on top of the frozen base network, and the accuracy is seen as a proxy for representation quality.

2.1.6 Other Details

There was some implementation detail regarding large batch sizes (from 256 to 8192.) Since using this number of batches is not feasible in my systems this part will not be explained in this report.

The paper trains the models in batch sizes of 4096 for 100 epochs in most cases.

Learning rate varies between the experiments, but the default learning rate is 4.8 (= 0.3 × BatchSize/256) and weight decay of 10−6 for LARS optimization. In addition, linear warmup for the 10 first epochs and cosine decay schedule without restart are applied.

2.2 Key Results of the Original Paper

The main results include three parts: (1) Exploring various data augmentation compositions and their effect on the performance of the model, (2) examining the effects of the of introducing a learnable nonlinear transformation between the representation and the contrastive loss on the quality of the learned representations, (3) Experimenting with various batch sizes and training steps and evaluating their effect on the SimCLR framework comparing to supervised learning.

2.2.1 Data Augmentations Findings

There are two main results that the paper is showing with respect to data augmentation. First one is that the composition of data augmentation operations is crucial for learning good representations. The second one is that contrastive learning needs stronger data augmentation than supervised learning.

There are two types of data augmentations: (1) The type that involves spatial/geometric transformation of data, such as cropping and resizing (with horizontal flipping), rotation and cutout. (2) The other type of augmentation involves appearance transformation, such as color distortion (including color dropping, brightness, contrast, saturation, hue), Gaussian blur, and Sobel filtering. Overall, nine augmentations are used in this study which are: (1) crop and resize (2) crop, resize, and horizontal flip (3) color distortion with dropping (4) color distortion with jittering (5) rotation (90, 180, 270) (6) cutout (7) gaussian noise (8) gaussian blur and (9) sobel filtering.

The augmentations were applied individually and in pairs to investigate their effect. Random crop and resize were always applied to the pictures, and made it hard to study the effect of other augmentations without them. Therefore, all the images were cropped and resized, but only one of the pictures in each positive pair (refer to image 1) went through the desired augmentations.

Figure 4 shows the results of the evaluation for individual transformation and pairs. The authors conclude that no single transformation suffices to learn good representations, even though the model can almost perfectly identify the positive pairs in the contrastive task. The best composition of augmentations according to the figure is random cropping and random color distortion.

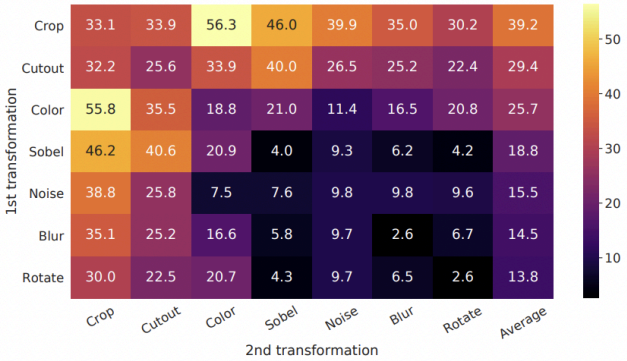

Figure 4: Linear evaluation (ImageNet top-1 accuracy) under individual or composition of data augmentations, applied only to one branch

2.2.2 Architectures for Encoder and Head

In this part, the authors evaluate the effect of the architectures of encoder and projection head models on unsupervised contrastive learning. The two main takeaways from this part are that unsupervised learning

benefits more from bigger models than its supervised counterpart and that a nonlinear projection head improves the representation quality of the layer before it.

2.2.2.1 Encoder Architecture Effect

In Figure 5 we can see that there is a trend of improving the accuracy when we have more parameters in the model. Furthermore, the performance of the supervised models and linear classifiers trained on unsupervised models get closer to each other as the number of parameters increase. In other words, it can be concluded that unsupervised learning benefits more from bigger models than its supervised counterpart.

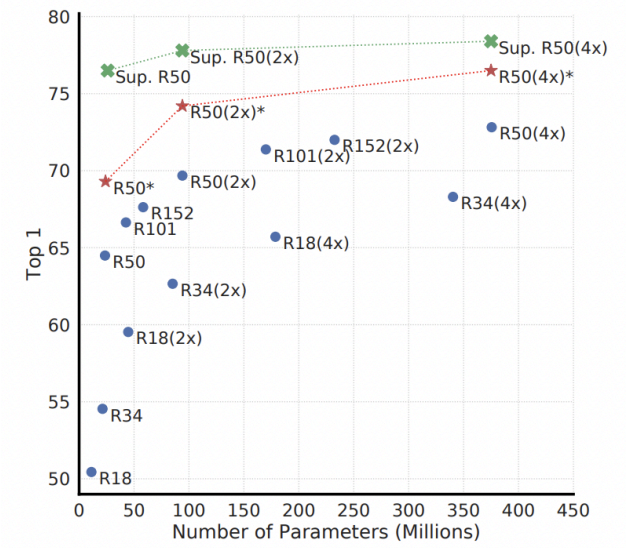

Figure 5: Linear evaluation of models with varied depth and width. Models in blue dots are ours trained for 100 epochs, models in red stars are ours trained for 1000 epochs, and models in green crosses are supervised ResNets trained for 90 epochs

2.2.2.2 Projection Head Effect

For studying the effect of including a projection head to the framework, three different settings are compared: (1) identity mapping (2) linear projection (3) the default nonlinear projection with one additional hidden layer and ReLU activation.

We can see the results of the authors experiment in figure 6, where it is clear that using a projection head significantly increases the top-1 accuracy. Also, it can be deducted from the figure that a non-linear projection head outperforms a linear projection head.

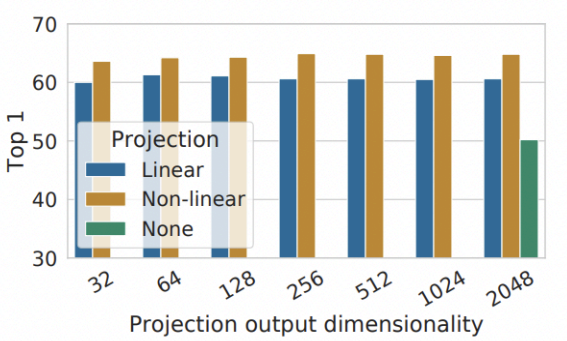

Figure 6: Linear evaluation of representations with different projection heads and various dimensions of z

Furthermore, the authors evaluated the accuracy of models when additional MLPs were trained on them and found out that using the layer before the projection head has more information than the layer after (Table 1).

Table 1: Accuracy of training additional MLPs on different representations

What to predict?	Random guess	Representation h	Representation $g(h)$
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

2.2.3 Loss Functions and Batch Size

Two results were concluded with respect to the loss function and the batch size's effect: (1) Normalized cross entropy loss with adjustable temperature works better than alternatives, (2) contrastive learning benefits (more) from larger batch sizes and longer training.

2.2.3.1 Loss Function Effect

Two conventional loss functions, namely margin loss and logistic loss were compared to the proposed loss function NT Xent. At the table 2, it can be seen that using the NT Xent loss improves top-1 accuracy significantly.

Table 2: Linear evaluation (top-1) for models trained with different loss functions. "sh" means using semi-hard negative mining

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

2.2.3.2 Batch Size Effect

Using the results from the figure 7, the authors conclude that while both supervised and unsupervised baselines benefit from higher batch sizes and training epochs, the

effect is more pronounced in the unsupervised framework. They also observed that the gap between unsupervised and supervised baselines decreases as the batch size increases, and therefore unsupervised learning can benefit more from them than supervised counterparts.

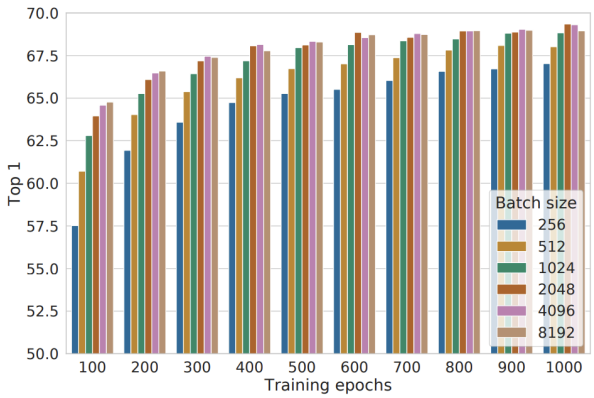

Figure 7: Linear evaluation models (ResNet-50) trained with different batch size and epochs

3. Methodology (of the Students' Project)

In this project, I attempted to reproduce the two main results: (1) how data augmentation impacts the performance and (2) what is the effect of projection head, and (3) the effect of the loss function on the output. For making the project computationally feasible, I had to use a really small subset of the dataset which is explained in part 4.1. As a result, I was not able to reach the accuracy in the original paper, but the codes are written in a way that if you run the code on a system with higher GPU power the experiments could match the original paper's work.

First,

3.1. Objectives and Technical Challenges

The objective of this project is to reproduce the two main results of the original paper which are regarding data augmentations and their effect on the top-1 accuracy and the effect of linear and non-linear projection head on top-1 accuracy.

The part about using various batch sizes was not reproduced because the initial paper starts with the batch size of 100, but the maximum batch size that my system can handle without crashing is 16. So the main challenge I faced was about the RAM size and how because of it I could not use a larger dataset to train my models. Other than that, I had a problem with the LARS optimizer that I could not resolve using my version of tensorflow, therefore I used a similar function that was introduced in the appendix as an equivalent of the LARS optimizer.

4. Implementation

For reproducing the results of the paper, first we need to define several key important functions, namely contrastive loss function, evaluation functions for top-k accuracies, training and testing functions, and a custom scheduler. Combining these functions in the right setup allows us to reproduce the results correctly and get more accurate comparisons.

The main libraries used in this project are tensorflow, keras, and numpy. Other functions were either related to visualization or were implemented from scratch. The gc library has been used to delete unwanted arguments from the GPU after each training iteration.

Two experiments were run for results reproduction all of which needed the linear evaluation protocol which was implemented in the main jupyter notebook, for the sake of simplicity and visualization of the results.

The jupyter notebook also contains the codes of the two main experiments.

All the experiments are based on a pretraining setting in which a ResNet-50 base encoder is trained for lowering the contrastive loss NT Xent alongside a projection head. The pretrained models are stored in my google drive and can be accessed using the readme file in the github repository. The main code of the pretraining part can be seen in the jupyter notebook. The primary setting for the pretraining is a nonlinear projection head (with ReLU activation and 128 output dimension), temperature of 0.5, base learning rate of 0.03 * batch_size / 256 (0.18), 10 warmup epochs, 100 training epochs, and an SGD with 0.9 momentum optimizer. These settings have been chosen to be as close as possible to the default setting of the original paper.

4.1 Data

The dataset used for this project was a part of the validation set available for the original dataset. It contains 120 classes of the original dataset, but for the sake of computational feasibility I used the 912 high-resolution pictures belonging to five classes.

4.1 Deep Learning Network

Here I discuss the architecture of the models used in the project.

4.1.1 Architectural Block Diagrams

All of the implemented models used a ResNet-50 base encoder, the block diagram of which can be seen in figure 8. For the linear evaluation protocol, the top layer of the model is not included and the parameters in the model are frozen and cannot be trained. In addition, the architecture of the non-linear projection head can be seen in figure 9. The diagram of the linear evaluation model only has one layer and therefore is not added here.

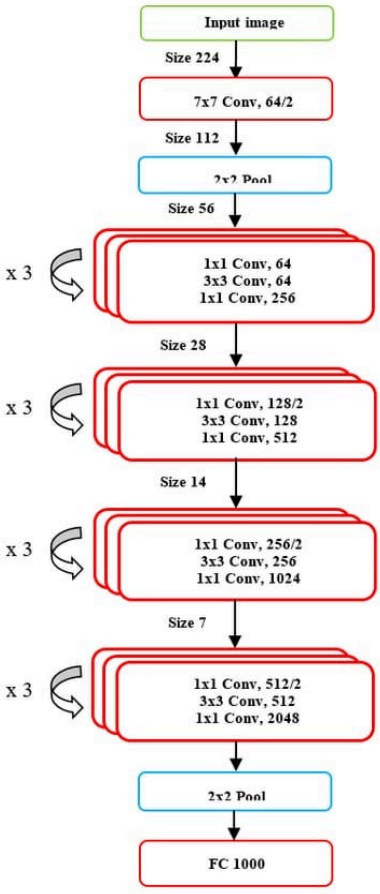

Figure 8: ResNet-50 block diagram

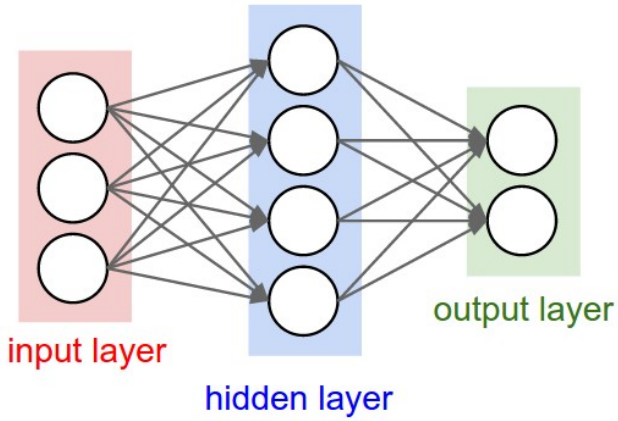

Figure 9: ResNet-50 block diagram

5. Results

5.1 Project Results

Two different experiments were performed in this project which are discussed in detail in the upcoming subsections.

5.1.1 Data Augmentation Effect

The first experiment I performed was to examine the effects of augmentations on top-1 accuracy of the model. For this part, a previously pretrained model with non-linear projection head and a hidden dimension of 128 has been used. Figure 10 shows the results of my experiment. Similar to the original paper, data augmentation functions have been analyzed in pairs and singularly. All the originally mentioned augmentations in the paper (random crop, color distortion, gaussian blur, cutout, gaussian noise, sobel filter, and rotation) have been included in this part in addition to random horizontal flipping with 50 percent accuracy. The results seen in the figure do not align with the original paper, as the most efficient pair here is sobel filter and rotation, while in the original paper it is random crop and color distortion. However, this disagreement between the results is expected since the scale of the training dataset is much smaller in this reproduction. All the figures can be seen in better detail in my github repo. [1] For reference, figure 11 shows the top-1 accuracy results from training.

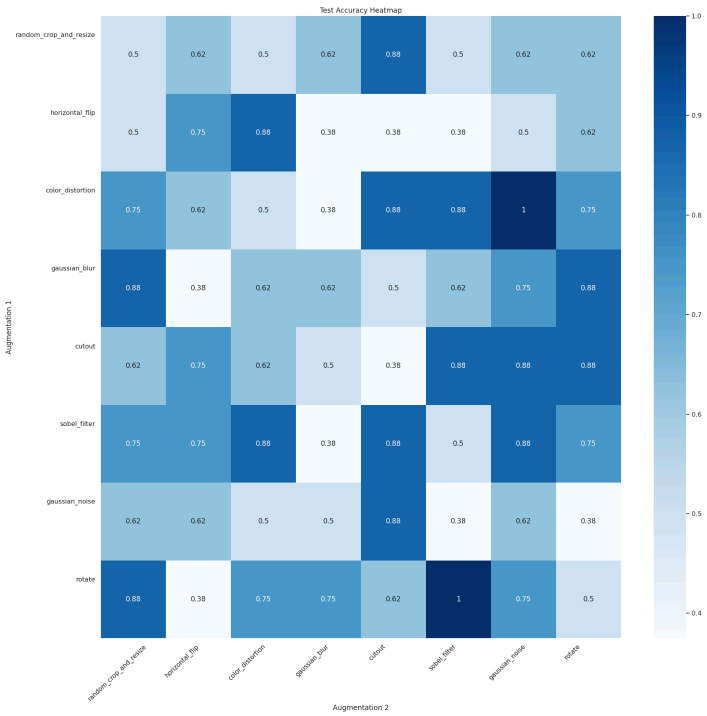

Figure 10: Test top-1 accuracy results of the effect of different pairs of data augmentation

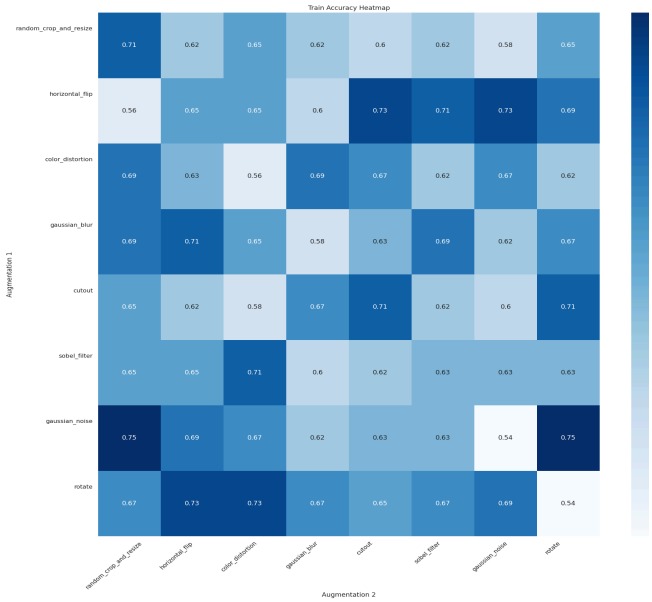

Figure 11: Train top-1 accuracy results of the effect of different pairs of data augmentation

5.1.2 Linear and Nonlinear Projection Head

The second experiment is related to the impact of the linear and nonlinear projection heads on top-1 accuracy of the output. For this part, I copied the original paper in choosing latent space dimensions from 32 to 2048. I proceed to do a pretraining and evaluation for every hidden dimension with a linear and nonlinear projection head. The linear projection head only has one layer with no activation, and the non-linear layer has one hidden layer with an activation function of ReLU. The results of this experiment can be seen in figure 12 (for test) and figure 13 (for train). As seen in the figures, the liner head generally performs better than the nonlinear one which is in contrast with the original paper's findings. Here again, I suspect that a lack of enough computational power and having a small dataset have had an impact on the output. Specifically, I assume that the information provided by the small subset of dataset I use and the number of epochs (10) are not enough for training a base encoder alongside a more complicated structure (nonlinear projection head). In other words, the more complex the structure gets, the harder it is to train it using the limited number of training epochs and data samples.

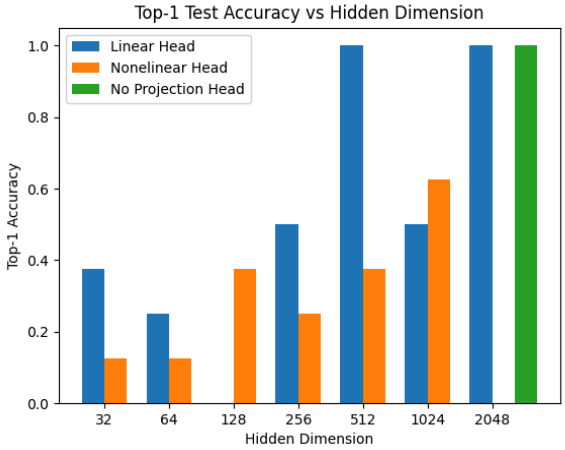

Figure 12: Top-1 test accuracies for having a non-linear and linear projection head across different latent space dimensions.

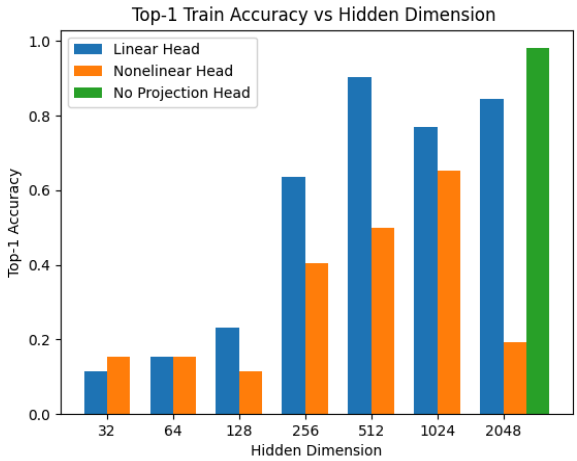

Figure 13: Top-1 train accuracies for having a non-linear and linear projection head across different latent space dimensions.

5.1.3 Pretraining with Contrastive Learning

Overall, four experiments were conducted in my projects two of which were about the key findings of the original paper. The other two are explained in this part.

5.1.3.1 Pretraining with Contrastive Learning

I used the default setting of the original paper for the SimCLR framework:

- epochs = 100
- temperature = 0.5
- base_learning_rate = 0.03 * batch_size / 256
- warmup_epochs = 10
- backbone = ResNet-50
- Optimizer = SGD with momentum

- Nonlinear projection head with hidden dimension of 128 and ReLU activation

And pretrained the model (ResNet-50 + Projection head). The results can be seen in figure 14. We can see that the loss is decreasing which means that the model is getting better at identifying the similar pictures with different representations. Another phase of pretraining the model with a linear head has been also done for the sake of comparison. We can see that the loss value of the model with linear head is lower and that the loss values follow rather similar trends.

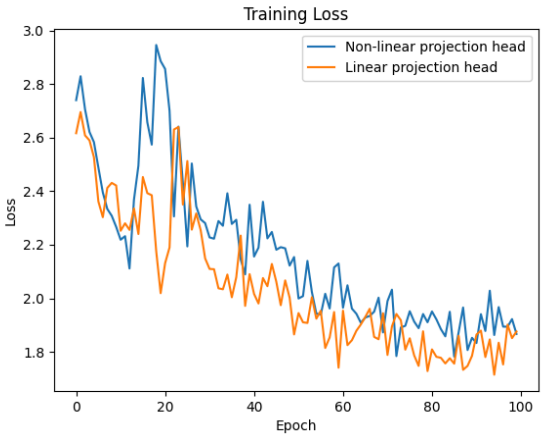

Figure 14: Pretraining of model according to SimCLR framework.

5.1.3.2 Linear Evaluation Protocol

A linear classifier has been trained on the pretrained model with SimCLR framework (nonlinear head), the results of which can be seen in figures 15 (accuracy) and 16 (loss). Fortunately, a high accuracy can be achieved (+0.8) using the pretrained model, which means that the reproduction of the SimCLR framework successfully created a high quality representation of the images in the test set.

The setting of the linear evaluation protocol are as follows:

- epoch = 15
- temperature = 0.7
- base_learning_rate = 0.1 * batch_size / 256
- warmup_epochs = 0
- Optimizer: SGD with momentum
- Loss Function: Categorical CrossEntropy

These parameters and functions were chosen to produce similar results as the ones in the original paper.

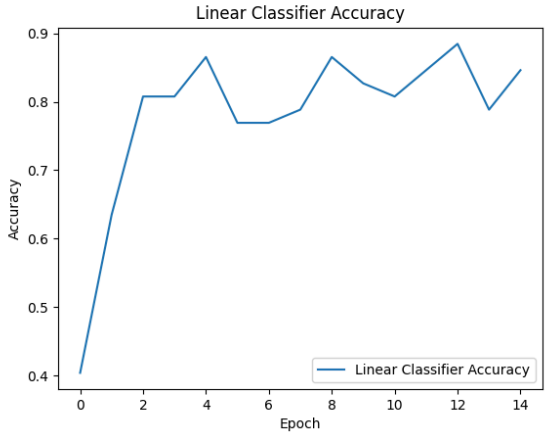

Figure 15: Linear Classifier accuracy on the pretrained model

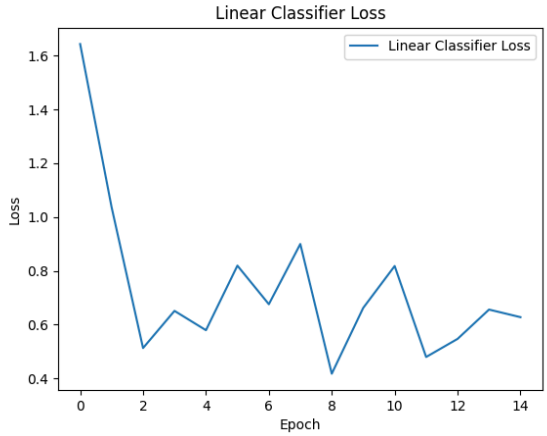

Figure 16: Linear Classifier loss on the pretrained model

5.2 Comparison of the Results

In this project, two main results were reproduced (1) the effect of data augmentation (2) the effect of projection heads. The results obtained were both different from the original paper's reports.

In the first experiment, the most important augmentations are a combination of color distortion and random crop and resize, while in my results, sobel-filtering and rotation achieved the best results. However, it is evident from both results that the chosen augmentations could have a huge effect on the performance of the model and should be chosen with care.

The second experiment also produced different results than the ones seen in the original paper. In the original paper, the model with nonlinear projection head clearly outperforms the model with no head and the model with a linear projection head. However, in the project the model with linear head mostly outperforms the model with a nonlinear head and both underperform in

comparison to the model with no projection head. This might simply be because as the models get more complex, they need more data and training iterations to perform well.

5.3 Discussion / Insights Gained

In this project, two main results were reproduced (1) the effect of data augmentation (2) the effect of projection heads. Although the results obtained were different from the ones reported in the original paper, they still provided some insight. For instance, it can be seen that the results of each combination of augmentation pairs are different from each other, meaning that finding the optimal augmentation functions for your specific contrastive learning task could be valuable and have a high impact on your final results.

In addition, it is obvious from the second experiment I reproduced that training time hugely impacts the model with nonlinear head's performance. Other models also benefit from it, but the model with nonlinear head outgrows the other two (model with linear projection head and the one without projection head) in enough training epochs (as seen in the results of the original paper).

The main reason behind the results of the original paper and my project being out of alignment was regarding the system limitations. The dataset used in the project has only about 1000 pictures (1/50 of the original paper's dataset) and 6 classes (of 1000 overall classes.) In addition, due to the long training period, the models were trained for far fewer epochs (usually about 1/10 of the original paper.) These two reasons justify the different results achieved in this project.

6. Future Work

I believe it would be of benefit to study the combination of the results of the main experiments provided by the original paper.

An example would be to study the importance and effect of augmentation pairs with respect to the batch size number and training epochs. This way, we could understand if there is a more optimal way of implementing the SimCLR framework. Another example would be to observe the effect of batch size and iteration numbers on top-1 accuracy of models pretrained with different projection heads.

7. Conclusion

In this project, I attempted to reproduce some of the results of the paper "A Simple Framework for Contrastive Learning of Visual Representations." by performing two main experiments regarding the two key findings of the paper. The experiments were designed to show the impact of data augmentation functions and projection head on a contrastive learning framework and were performed in a similar setting to the original paper. However, the results were quite different from what was reported in the original paper, both in terms of the optimal augmentation pair and the effect of nonlinear and linear projection head. This difference was mainly due to the limitations in the system speed and memory that lead to a much shorter training time and a using a far smaller dataset. Although there has been a divergence in the results, some points made by the original paper have been seen in the project as well. Such as the importance of choosing the appropriate augmentation functions for a contrastive learning environment.

6. Acknowledgement

I appreciate the support of course TAs and Professor for their guidance and responsiveness.

7. References

[1]https://github.com/ecbme4040/e4040-2024Fall-Project-SAMH-SH4635

[2]https://arxiv.org/pdf/2002.05709

[3]https://stackoverflow.com/questions/62793043/tensorflow-implementation-of-nt-xent-contrastive-loss-function

[4]https://www.image-net.org/challenges/LSVRC/2012/

8. Appendix

8.1 Individual Student Contributions in Fractions

This project was done by only one student.

8.2 Support Material

8.2.1 Comparison of the framework with SotA models

The comparisons include three parts: Linear evaluation, semi-supervised learning, and transfer learning.

For this section, ResNet-50 was used in 3 different hidden layer widths (width multipliers of 1×, 2×, and 4×). For better convergence, the models were trained for 1000 epochs.

2.2.1 Linear Classifier Results

Table 3 demonstrates the results of linear classifier training on representation results both in a setting where ResNet-50 is used and when it is not.

We can see that the SimCLR framework (based on ResNet-50, ResNet-50 (2x), and ResNet-50 (4x)) outperforms other methods by a significant amount.

Table 3: ImageNet accuracies of linear classifiers trained on representations learned with different self-supervised methods

Method	Architecture	Param (M)	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×)	86	55.4	-
BigBiGAN	RevNet-50 (4×)	86	61.3	81.9
AMDIM	Custom-ResNet	626	68.1	-
CMC	ResNet-50 (2×)	188	68.4	88.2
MoCo	ResNet-50 (4×)	375	68.6	-
CPC v2	ResNet-161 (∗)	305	71.5	90.1
SimCLR (ours)	ResNet-50 (2×)	94	74.2	92.0
SimCLR (ours)	ResNet-50 (4×)	375	**76.5**	**93.2**

Another metric they used for evaluating the performance of this part is Top-1 Accuracy, which is the conventional accuracy put in other terms. As we can see in Figure 17 SimCLR models are at the top side of the plot, meaning that they were able to obtain higher accuracies (i.e. top-1 accuracy) while competing with models of the same size (number of parameters.) These results are particularly impressive because many of the models that have inferior performance to SimCLR require specifically designed architectures. We also can see the gray cross in the left, which shows the performance of supervised pretrained ResNet-50. The performance of SimCLR (4x) is roughly the same as the supervised pretrained ResNet-50 model, but the SimCLR framework achieves this result without using/knowing the labels of the data.

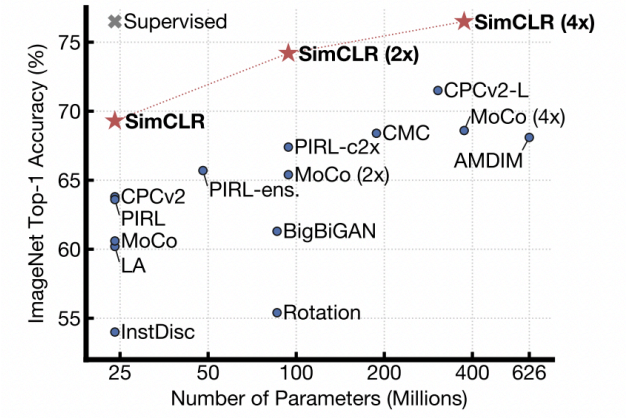

Figure 17: ImageNet Top-1 accuracy of linear classifiers trained on representations learned with different self-supervised methods (pretrained on ImageNet). Gray cross indicates supervised ResNet-50. Our method, SimCLR, is shown in bold

2.2.2 Semi-supervised Learning Results

For the semi-supervised learning experiment, the authors have sampled 1% or 10% of the labeled ILSVRC-12 training datasets in a class-balanced way (~12.8 and ~128 images per class respectively). Afterwards, they fine-tuned the whole base network on the labeled data without regularization.

Table 4 shows the comparisons of their results with other methods. We can see that both in top-1 and top-5 accuracies the SimCLR models with ResNet-50 (2x and 4x) outperform even the supervised baselines. The SimCLR ResNet-50 also shows competitive results in comparison to the State-of-the-Art, both in comparison to supervised baselines and baselines that use representation learning only.

Table 4: ImageNet accuracy of models trained with few labels

Method	Architecture	Label fraction 1%	Label fraction 10%
		Top 5	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**

2.2.3 Transfer Learning Results

Transfer learning performance has been measured across 12 natural image datasets in both linear evaluation (fixed feature extractor) and fine-tuning settings. The authors performed hyperparameter tuning for each model-dataset combination and selected the best hyperparameters on a validation set. Table 5 shows that the fine-tuned SimCLR model significantly outperformed supervised baselines on 5 datasets, and is statistically tied with the supervised baselines in 5 other ones.

Table 5: Comparison of transfer learning performance of SimCLR with supervised baselines for ResNet-50 (4×) models pre-trained on ImageNet

	Food	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
SimCLR (ours)	**76.9**	**95.3**	80.2	48.4	**65.9**	60.0	61.2	**84.2**	**78.9**	89.2	**93.9**	**95.0**
Supervised	75.2	**95.7**	**81.2**	**56.4**	64.9	**68.8**	**63.8**	83.8	**78.7**	**92.3**	**94.1**	94.2
Fine-tuned:												
SimCLR (ours)	**89.4**	**98.6**	**89.0**	**78.2**	**68.1**	**92.1**	**87.0**	**86.6**	**77.8**	92.1	**94.1**	97.6
Supervised	88.7	98.3	**88.7**	**77.8**	67.0	91.4	**88.0**	86.5	**77.8**	**93.2**	**94.2**	**98.0**
Random init	88.3	96.0	81.9	**77.0**	53.7	91.3	84.8	69.4	64.1	82.7	72.5	92.5