UNDER REVIEW

RESCIENCE C

Replication / ML Reproducibility Challenge 2022

[Re] Intriguing Properties of Contrastive Losses

Luca Marini[1], Mohamad Nabeel[1], and Alexandre Loiko[1]

[1]KTH Royal Institute of technology, Stockholm, Sweden

Edited by
(Editor)

Received
–

Published
–

DOI
–

Reproducibility Summary

Scope of Reproducibility – In 2021, Chen et al. [1] studied three properties of contrastive learning. One of the results from the paper shows that the instance-based objective widely used in existing contrastive learning methods can learn meaningful local features (e.g. dogs' facial components, as shown in Figure 9) despite operating on global image representation. In this paper, we validate this property, we perform experiments beyond the findings of Chen et al. [1], and we evaluate the effect of the deep projection head on the accuracy when using different batch sizes for the linear evaluation of SimCLR.

Methodology – We implemented the project with Python using PyTorch as deep learning library, while the original paper's repository[3] provides three Jupyter Notebooks using Tensorflow. In particular, the original paper's repository does not provide any code for the experiments we reproduced. Therefore, we fully re-implemented the proposed methods by following the description of the original paper. We used the pre-trained SimCLR models provided by the authors' repository.

Results – The obtained linear evaluation accuracies differ in a range between 0.19% and 2.05%, while the ones in the original paper differ from 0.20% to 0.80%. Nonetheless, we believe that our results support that the differences in top-1 accuracy among different batch sizes are minimal because of different choices of the dataset, base encoder, and batch sizes, and also because the range substantially increases when the projection head is not deep. All the other experiments support the original and the newly tested claims.

What was easy – The paper of Chen et al. [1] is well-written, which made it easy to comprehend. In addition to that, checkpoints of the models are provided and therefore it was relatively easy to reproduce the considered experiments.

What was difficult – We had issues reproducing the linear evaluation results of SimCLR due to our limited computational resources. So, we trained a smaller base encoder for fewer epochs compared to the original paper. We also had some doubts about the used version of SimCLR and some other implementation details because the original repository[3] provides checkpoints for both versions and it does not provide code of the experiments we reproduced.

Copyright © 2023 L. Marini, M. Nabeel and A. Loiko, released under a Creative Commons Attribution 4.0 International license.
Correspondence should be addressed to Luca Marini (lucamar@kth.se)
The authors have declared that no competing interests exist.
Code is available at https://github.com/mona251/Intriguing-Properties-of-Contrastive-Losses. – SWH swh:1:dir:35a398e4df2fda2f4241886f39c193b5a53a3e4c.
Open peer review is available at https://openreview.net/forum?id=gb71irTNN7.

Communication with original authors — We communicated with the first author of the original paper (Ting Chen) twice by email for doubts, and we promptly received replies.

1 Introduction

Contrastive learning has recently become a well-received component in self-supervised learning for computer vision, natural language processing (NLP), and other domains. It aims at embedding augmented versions of the same sample close to each other while trying to push away embeddings from different samples [2].
In 2020, Chen et al. [3] proposed SimCLR, a self-supervised learning method for visual representations, which is based on the contrastive loss between two views of the same image, which are transformed by a learned feature representation.
In 2021, Chen et al. [1] reported three intriguing properties of contrastive learning by using SimCLR. First, they generalized the standard loss to a broader family of losses and found that variants of contrastive losses perform similarly when a multi-layer non-linear projection head is used. Secondly, they studied whether instance-based contrastive learning can learn well on images with multiple objects. Additionally, they showed that easily learned features can inhibit the learning of more discriminative ones.

2 Scope of reproducibility

This paper reproduces some of the experiments of [1]. The first experiment of [1] compares the performance of different instantiations of generalized contrastive losses for SimCLR with varying batch sizes using the linear evaluation protocol. Their work shows that changing the batch size has a negligible impact on the linear evaluation top-1 accuracy when having a fixed number of layers in the projection head and when the base encoder is trained for a fixed number of epochs. The authors also claimed that this is especially true when the projection head has more layers.
One of the main findings of [1] is that the instance-based objective widely used in existing contrastive learning methods can learn meaningful local features despite operating on global image representation (e.g. dogs' facial components, as shown in Figure 9 in Appendix A.1). In this paper, we mainly study this property and also perform experiments beyond this finding. Specifically, we evaluate the following claims:

(i) The differences in linear evaluation top-1 accuracy among different batch sizes are minimal when having a deep projection head with a fixed number of layers.

(ii) SimCLR learns local features that exhibit hierarchical properties.

Furthermore, we test the following additional claims:

(iii) The learning of local features does not depend on K-Means, but is also obtained with other clustering algorithms.

(iv) SimCLR recognizes features with the same semantic meaning among multiple images.

3 Methodology

We re-implement the proposed experiments from the description in [1]. In particular, the original paper's repository[3] does not provide any code for the experiments that we reproduce. We use Python3 for the implementation with PyTorch [4] as the deep learning library, and OpenCV [5] for image manipulation. We use the pre-trained SimCLRv2 models of the authors by converting the Tensorflow checkpoints into PyTorch ones by

using two repositories [1] [2] linked in the original repository[3]. In particular, the provided checkpoints are of three types: for SimCLR, for SimCLRv2, and for supervised ResNet-50 2X. We decided to present the results with the latest version, i.e. SimCLRv2. However, we implemented both versions.
The additional clustering algorithms tested in experiment 3 are Ward hierarchical clustering and Bisecting K-Means.

3.1 Model descriptions

SimCLR is a simple framework for contrastive learning of visual representations introduced by Chen et al. [3] that considerably outperformed previous methods for self-supervised and semi-supervised learning on ImageNet. It consists of four major components: *(i)* a stochastic data augmentation module that transforms any given data example randomly resulting in two correlated views of the same example, *(ii)* a neural network base encoder that extracts representation vectors from augmented data examples (a ResNet [6] is used in the original paper), *(iii)* a small neural network projection head that maps representations to the space where contrastive loss is applied, *(iv)* a contrastive loss function defined for a contrastive prediction task.
We use SimCLRv2 [7], which is an improved version of SimCLR [3] that uses larger ResNet [6] models and selective kernels [8]. In Experiment 1 we use ResNet-18 as the base encoder instead of ResNet-50 due to computational and time constraints, and a logistic regression classifier to perform the linear evaluation. For Experiments 2, 3, and 4 we set up a pre-trained ResNet-50 2X model on ImageNet with its checkpoints provided by the author's repository[3].

3.2 Datasets

To reproduce Experiment 3.2 we do not need a dataset to train SimCLRv2 on since the pre-trained models on ImageNet are available. We use some input images of Imagenette[4], which is a subset of 10 ImageNet classes. In particular, we use ten English springer and five truck images. We also use ten images from the Stanford Dogs Dataset by Khosla et al. [9]. The dataset contains images of 120 dog breeds. In particular, two images of five different dog breeds are used. For experiment 1 we use CIFAR-10 [10].
Additionally, we reproduce the construction of two out of three datasets of Chen et al. [1] with explicit and controllable competing features that can be used to reproduce the other experiments of the original work that we did not replicate.

3.3 Hyperparameters

For experiment 1, we use ResNet-50 2X as the base encoder network, and an n-layer MLP projection head ($n = 1 \ldots 4$) to project the representation to a 64-dimensional latent space. We use NT-Xent as loss, optimized using LARS optimizer [11] with learning rate equal to 0.3 × BatchSize/256 and weight decay of $10e{-}6$. We train with batch sizes of 128, 256, and 512 for 100 epochs. The linear classifier is trained for 800 epochs with Adam [12] as optimizer and learning rate of 0.0003. For the last three experiments, the hyperparameters of KMeans[5] are set to their default values, except for *init*="k-means++", *n_init*=10, *max_iter*=300, *tol*=$1.0 \cdot e^{-4}$. The hyperparameters of Ward hierarchical clustering and Bisecting K-Means are set to their default values, respectively reported in [8] and [9].

[1] https://github.com/tonylins/simclr-converter
[2] https://github.com/Separius/SimCLRv2-Pytorch
[3] https://github.com/google-research/simclr/tree/master/colabs/intriguing_properties
[4] https://github.com/fastai/imagenette

3.4 Experimental setup and code

By performing inference on input images with a pre-trained ResNet-50 2X model (SimCLRv2's base encoder network), we extract the l2-normalized features from its intermediate block groups (block groups 2,3,4 of ResNet-50 2X).
We then apply K-Means [5] with 2, 4, 6, and 8 clusters on the extracted features of SimCLRv2 to paint each pixel of the feature map, which we call cluster assignment mask throughout the report. We use a pre-defined order depending on which cluster each pixel belongs to. However, K-Means is not guaranteed to return the clusters in the same order when applied more than once on the same feature map.
After that, we upsample the cluster assignment masks to the size of the input image. To upsample the mask, we provide both the bilinear and the nearest neighbors interpolation option, as the authors of [1] provided in their blog [6]. We use bilinear interpolation in the main text and defer other methods in Appendix A.2. We then overlay the upsampled cluster assignment mask with a gray-scale version of the input image, with the help of *addWeighted*[7] function from *opencv*.
We repeat the same steps for the features extracted from the supervised learning setting (i.e. from a pre-trained supervised Resnet-50 2X network).
Following Chen et al. [1], we also apply K-Means clustering on the raw pixels of input images. To this end, we downsample the input image to 14x14 pixels, the size of the feature maps extracted from Block Group 4 of the ResNet-50 2X. We then apply K-Means. After that, we upsample the image containing the clustered pixels and overlay it with the original image as we did with the extracted feature maps of SimCLRv2.
The reason for applying K-Means clustering also on the raw pixels of input images is to visually compare the cluster assignment masks of raw images against the ones obtained by SimCLRv2's learned features to evaluate if the second ones show the learning of meaningful features.
For the linear evaluation experiment, we first train SimCLRv2's base encoder for 100 epochs on CIFAR-10 without using labels. Secondly, we perform inference on CIFAR-10 with the pre-trained SimCLRv2 model and save the output representations. The projection head is removed before computing the output representations. Then, using CIFAR-10 training labels as targets, a logistic regression model is trained with the cross entropy loss function using SimCLRv2's output representations as input.

3.5 Computational requirements

For training SimCLRv2 and extracting features from it when performing inference we used NVIDIA GPU T4 x 2 provided by Kaggle Notebooks. On average, performing the linear evaluation of SimCLRv2, given a batch size and a number of layers in the projection head, took 2 hours and 34 minutes.
All the computations needed to visualize the features learned by SimCLRv2 (features and images manipulations, clustering, etc.) were executed by the 4th Generation Intel Core i7-4810MQ CPU with 4 cores and a base frequency of 2.80 GHz. On average, visualizing the features learned on a single input image took 32 seconds, while it took 10 minutes and 8 seconds to visualize the features learned on a batch of input images.

4 Results and Discussion

Our results support all the claims of Section 2.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[6] https://contrastive-learning.github.io/intriguing/
[7] https://docs.opencv.org/3.4/d2/de8/group__core__array.html#gafafb2513349db3bcff51f54ee5592a19

4.1 Results reproducing original paper

This section discusses the two experiments that test claims *(i)* and *(ii)* of Section 2, respectively.

Experiment 1: Linear Evaluation of SimCLR — Experiment 1 tests claim *(i)* of Section 2. We reproduce the original experiment on CIFAR-10 instead of ImageNet. We also use ResNet-18 instead of ResNet-50 as the base encoder and use smaller batch sizes compared to the original ones (512, 1024, 2048) due to limited computational resources. Table 1 shows the results.

Projection head	Batch size	Epoch 100
2 layers	128	51.17
	256	52.86
	512	53.22
3 layers	128	51.26
	256	52.47
	512	53.11
4 layers	128	51.69
	256	52.44
	512	52.25

Table 1. Linear eval accuracy of ResNet-18 on CIFAR-10.

The linear evaluation top-1 accuracy results obtained with different batch sizes differ in a range between 0.19% and 2.05%, while the differences in the original paper differ in a range between 0.20% and 0.80%. Although the range of our differences is broader, we believe that claim *(i)* is verified for two reasons. First, the extended range could be due to differences in the choice of the dataset, batch sizes, and base encoder. Secondly, the range of differences substantially increases when the projection head is not deep (i.e. 1 layer), as shown in Table 2.

Projection head	Batch size	Epoch 100
1 layer	128	50.38
	256	51.94
	512	53.82

Table 2. Linear eval accuracy of ResNet-18 on CIFAR-10 with a non deep projection head.

Experiment 2: Reproducing experiment 3.2 of Chen et al. [1] — The second experiment evaluates claim *(ii)* of Section 2, i.e. it evaluates the ability of SimCLR to learn meaningful local features. Some examples of cluster assignment masks of SimCLRv2 and of the supervised learning setting are respectively shown in the first and last two rows of Figure 1.

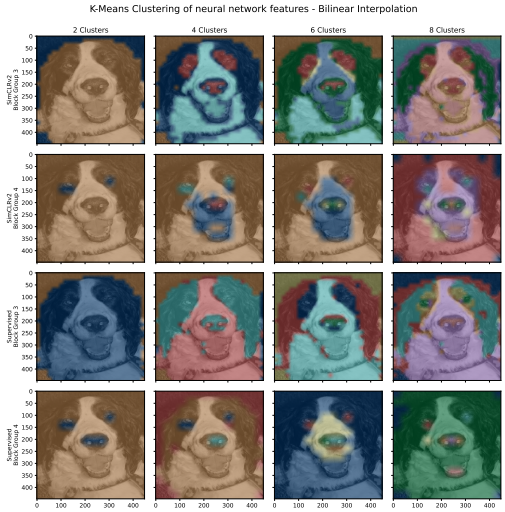

Figure 1. Visualizing features on an image with K-means clustering. Each row denotes a type of local feature. The first two are extracted from SimCLRv2, and the last two from a ResNet-50 2X trained with a supervised setting. Each column denotes the number of clusters.

Following Chen et al. [1], we also apply K-Means clustering on the raw pixels of an image. The results of an input image can be seen in Figure 2.

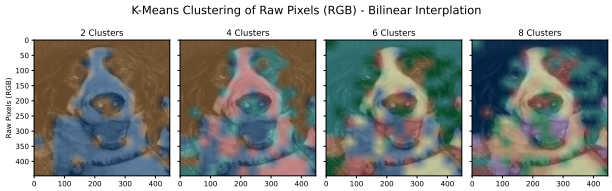

Figure 2. Visualizing clustered pixels on an image with K-means clustering. Each column denotes the number of clusters.

Figure 1 shows that as the number of clusters increases, the learned representations tend to group image regions based on parts of the object (i.e. facial components of the dog). This phenomenon appears in both SimCLRv2 and supervised learned features, but not with raw pixel features (Figure 2), confirming the claim of Chen et al. [1]. However, the colors of our cluster assignment masks (Figure 1) do not fit the dog's facial components as the ones illustrated in the original paper (Figure 9), especially in the case of 8 clusters.

4.2 Results beyond original paper

This section discusses the two experiments that test claims *(iii)* and *(iv)* of Section 2, respectively.

Experiment 3: Using different clustering methods — Experiment 3 tests claim *(iii)* of Section 2. Namely, its aim is to ensure that the learning of local features is also achieved with other clustering methods, not only with K-Means.
We apply two other clustering methods: Ward hierarchical clustering[8] and Bisecting K-Means[9]. K-means is a cluster analysis technique that groups observations by trying to

[8]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.BisectingKMeans.html

cluster samples in n groups of equal variance and uses a predetermined number of clusters. On the other hand, Hierarchical clustering is a family of clustering algorithms that creates nested clusters by progressively merging (agglomerative) or splitting (divisive) them [10]. In particular, Ward hierarchical clustering[8] is an agglomerative hierarchical clustering algorithm, in which every observation initially belongs to a separate cluster, and clusters are gradually combined. Ward hierarchical clustering[8] minimizes the sum of squared differences within all clusters. Similarly to the K-Means objective, it also minimizes the variance but it uses an agglomerative hierarchical strategy to solve the problem [10].
Bisecting K-Means [9] is an iterative version of KMeans that use divisive hierarchical clustering. Centroids are selected progressively depending on prior clustering rather than all at once. Until the desired number of clusters is obtained, a cluster is repeatedly divided into two new clusters [11].
As in the previous experiment, we apply the two additional clustering algorithms to the extracted features of SimCLRv2. The cluster assignment masks of the three methods can be seen in Figure 3. In particular, Figure 3 shows the cluster assignment masks of a feature extracted from block group 3 of SimCLRv2. The cluster assignment masks corresponding to the features of the other block groups can be seen in Appendix A.3. In Figure 3 we can notice, in general, that the cluster assignment masks seem consistent across the three clustering methods. However, in some cases, Ward hierarchical clustering seems to give a visually more distinguishable clustering assignment mask than the other two methods. For example, with 6 clusters (third column), Ward hierarchical clustering clusters the dog's eyes in a more fine-grained way.

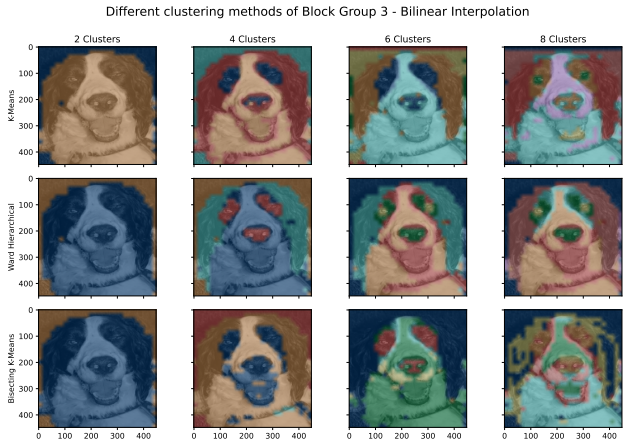

Figure 3. Visualizing features on an image with different clustering methods. The features are extracted from block group 3 of SimCLRv2. Each row denotes a clustering method, and each column denotes the number of clusters.

We continue using K-Means in the last experiment for two reasons. First, because the result of clustering is almost visually identical among the three clustering methods. Secondly, to be consistent with Chen et al. [1].

Experiment 4: Cross-image clustering — Experiment 4 evaluates claim *(iv)* of Section 2. Particularly, it goes beyond the scope of the claim of Chen et al. [1] by applying K-Means to the features extracted from batches of samples instead of single samples. We call the batch version of the cluster assignment mask cross-image cluster assignment masks. This way, the cluster centers are shared among different feature maps. This experiment

[10] https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering
[11] https://scikit-learn.org/stable/modules/clustering.html#bisecting-k-means

aims to see if features with the same semantic meaning are shared between different images or not.
The result of this experiment on a batch of dog images is shown in Figure 4. The colors are spread among the batch, but not all colors are displayed on each single batch image. In cross-image clustering, some features with the same semantic meaning have the same color. For instance, some of the dogs' faces and the dogs' ears are respectively colored light blue and red.

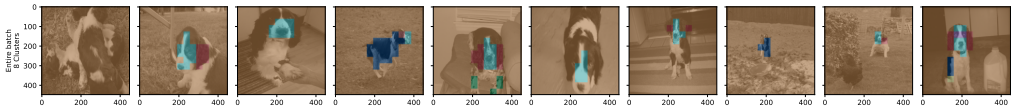

Figure 4. Visualizing features extracted from block group 4 of SimCLRv2 on a batch of images with K-means cross-image clustering with 8 clusters using nearest-neighbors interpolation. Each column represents a batch sample.

We then apply cross-image clustering on a batch of dogs images of different breeds to see if SimCLRv2 is able to learn features with the same semantic meaning but in images that are more diverse than the case of a single dog breed.
Figure 5 shows the results of cross-image clustering on a batch of images of dog faces of different breeds. SimCLRv2 seems to have learned the eyes and noses of a dog as features. This can be seen in the case with two clusters (first row, except for the second image). In the last row, the last two husky images have eyes colored blue, while most of the other dog images on the same row have eyes colored violet. This could be possibly due to the different real colors of the eyes of these husky dogs (blue) compared to the other dogs (brown or black). This difference can be seen in Figure 6.

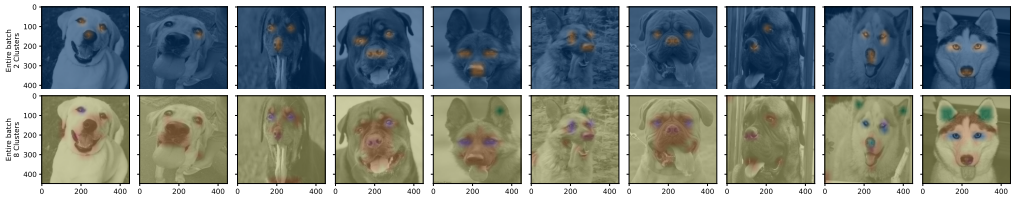

Figure 5. Visualizing features extracted from block group 4 of SimCLRv2 on a batch of images with K-means cross-image clustering. Each row denotes the number of K-means clusters, and each column represents a batch sample.

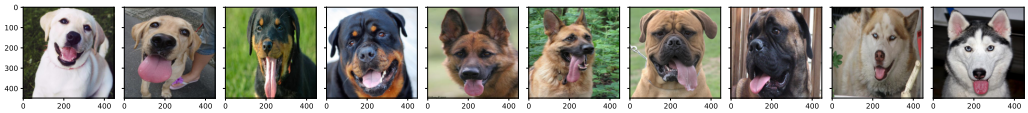

Figure 6. Batch of dog faces.

We intentionally took input images where dogs show off their tongue. We can see that block group 4 of SimCLRv2 does not seem to have learned tongue as a feature (Figure 5). However, the tongue is a feature identified by SimCLRv2 in block group 3 (Figure 7). In particular, this cross-image cluster assignment mask contains more semantic features compared to block group 4. The cross-image cluster assignment masks of block group 3 seem also to be more consistent between images. For instance, the dogs' eyes have all the same color.

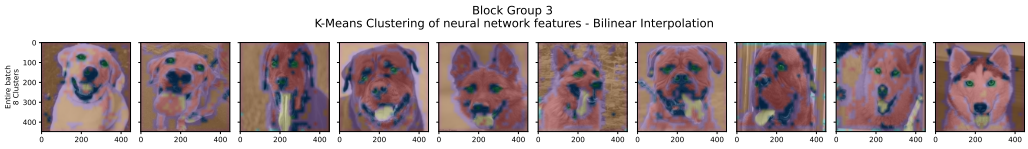

Figure 7. Visualizing features extracted from block group 3 of SimCLRv2 on a batch of images with K-means cross-image clustering with 8 clusters.

We then apply cross-image clustering on a batch containing two different classes: dogs and trucks. Figure 8 shows the cross-image cluster assignment masks of block group 4. Looking at the last row (8 clusters), green pixels characterize the dogs' noses, violet pixels characterize the dogs' eyes, and blue pixels characterize the interiors of the trucks' cabs. However, yellow pixels characterize both dogs' bodies and truck cabs. One possible explanation could be that the dogs' fur color and the trucks' cabs have similar colors (black and white), but it is difficult to determine.

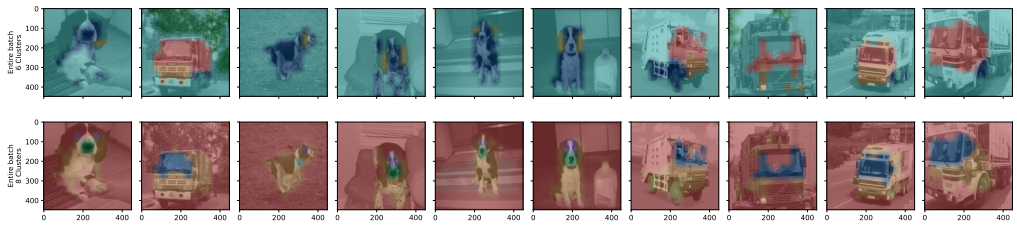

Figure 8. Visualizing features extracted from block group 4 of SimCLRv2 on a batch of images with K-means cross-image clustering. Each row denotes the number of K-means clusters, and each column represents a batch sample.

Features with the same semantic meaning were recognized as similar when having *(i)* different images of the one single dog breed, *(ii)* different images of different dog breeds *(iii)* images of two different classes (dogs and trucks). Therefore, we claim that SimCLRv2 learns features with the same semantic meaning among multiple images.

5 Conclusion

The purpose of this paper was to reproduce experiment 3.2 of [1] and provide additional insights about one of the properties claimed in [1]. We verified the claim of the original paper that states that hierarchical local features can be learned even though contrastive learning operates on global instance-level features. Due to time constraints and a lack of sufficient computation resources, this paper only reproduced the property of SimCLR learning local features despite operating on global image representation.
Next, we showed that the learning of local features does not depend on K-Means, but is also obtained with other clustering algorithms.
By applying cross-image clustering, we claim that SimCLRv2 learns features with the same semantic meaning among multiple images.

5.1 What was easy

The original paper is well-written, which facilitated the implementation. Moreover, the checkpoints are available, making inference straightforward to perform. After implementing the code, it was not difficult to test the last three claims because we just had to input images into a trained network and visually inspect the cluster assignment masks.

5.2 What was difficult

At first, we had issues finding a way to use TensorFlow checkpoints in PyTorch for the last three experiments. We also did not find information about which version was used in [1], if SimCLR [3] or SimCLRv2 [7].
While conducting Experiment 1, we were not sure if the number of epochs reported in Table 2 of [1] referred to the number of epochs that SimCLR or the linear classifier was trained for. In the first experiment, we also had to train with a smaller architecture and for fewer epochs due to our limited computational resources.
Lastly, we did not find how the authors upsampled the cluster assignment masks to the size of input images when reproducing Experiment 3.2 of [1], neither in the paper nor in the code. We assumed they used bilinear interpolation.

5.3 Communication with original authors

We contacted the authors twice by email. First, we asked if they used SimCLRv1 [3] or SimCLRv2 [7], and the first one was used. Nonetheless, we present the results of SimCLRv2 to show that it also satisfies the proposed claims. However, we tested all the claims and provide code for both versions.
Secondly, we asked if the number of epochs reported in Table 2 in the original paper referred to the number of epochs used to train SimCLR or the linear classifier. The authors replied that it refers to the number of epochs to train SimCLR.
Thirdly, we were unsure of our interpretation of the claim of the linear evaluation accuracy experiment of the original paper. From what we understood, with a fixed number of layers in the projection head, the top-1 accuracy is similar when having different batch sizes. The authors confirmed it and added that that is true especially when the projection head is deep.
Lastly, we asked from which network the “supervised learned features” were extracted in experiment 3.2 of [1]. It was ResNet-50 2X, which was the same architecture used for the self-supervised case.

References

1. T. Chen, C. Luo, and L. Li. "Intriguing Properties of Contrastive Losses." In: **Advances in Neural Information Processing Systems** 34 (2021).
2. A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. "A survey on contrastive self-supervised learning." In: **Technologies** 9.1 (2020), p. 2.
3. T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. "A simple framework for contrastive learning of visual representations." In: **International conference on machine learning**. PMLR. 2020, pp. 1597–1607.
4. A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: **Advances in Neural Information Processing Systems 32**. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.
5. G. Bradski. "The OpenCV Library." In: **Dr. Dobb's Journal of Software Tools** (2000).
6. K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition." In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016, pp. 770–778.
7. T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. "Big self-supervised models are strong semi-supervised learners." In: **Advances in neural information processing systems** 33 (2020), pp. 22243–22255.
8. X. Li, W. Wang, X. Hu, and J. Yang. "Selective kernel networks." In: **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. 2019, pp. 510–519.
9. A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. "Novel Dataset for Fine-Grained Image Categorization." In: **First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition**. Colorado Springs, CO, June 2011.
10. A. Krizhevsky, V. Nair, and G. Hinton. **Cifar-10 (canadian institute for advanced research)**. 2010.
11. Y. You, I. Gitman, and B. Ginsburg. "Large batch training of convolutional networks." In: **arXiv preprint arXiv:1708.03888** (2017).
12. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." In: **arXiv preprint arXiv:1412.6980** (2014).

A Additional Figures

A.1 Example figure from Chen et al. [1]

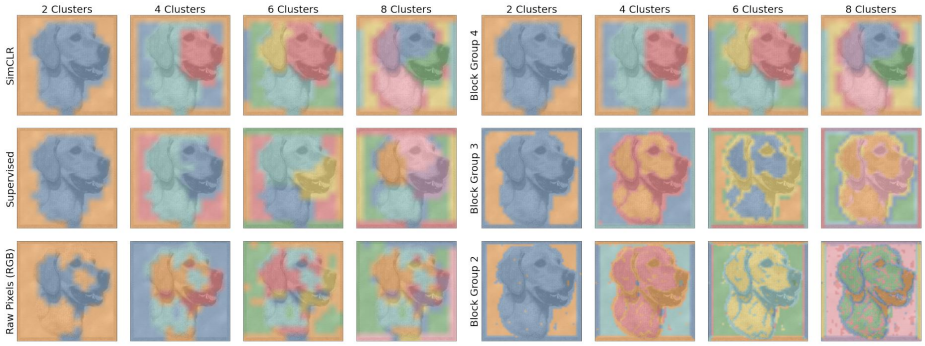

Figure 9. Result of experiment 3.2 from Chen et al. [1]. Later layers of SimCLR/supervised ResNet tend to group by object parts. A simple ResNet is used for the supervised learning case.

A.2 Experiment 2: Reproducing experiment 3.2 of Chen et al. [1]

In this appendix, we show the figures of Subsection 4.1.2 but upscaled using nearest neighbors interpolation instead of bilinear interpolation.

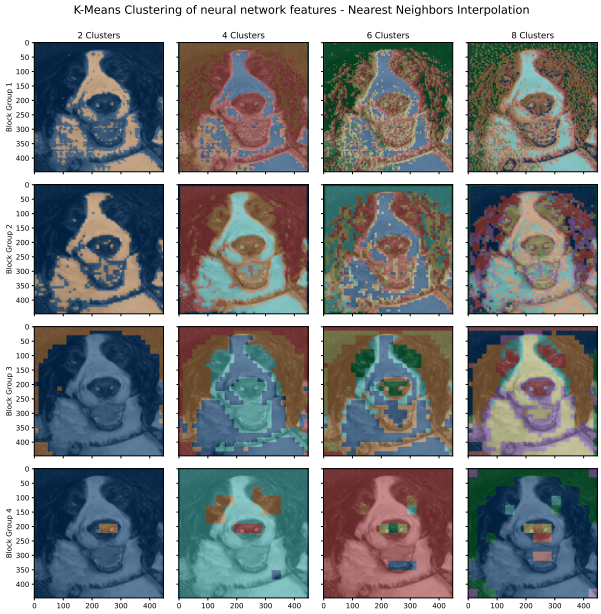

Figure 10. Visualizing features on an ImageNet validation image with K-means clustering. Each row denotes a type of local feature extracted from SimCLRv2, and each column denotes the number of K-means clusters.

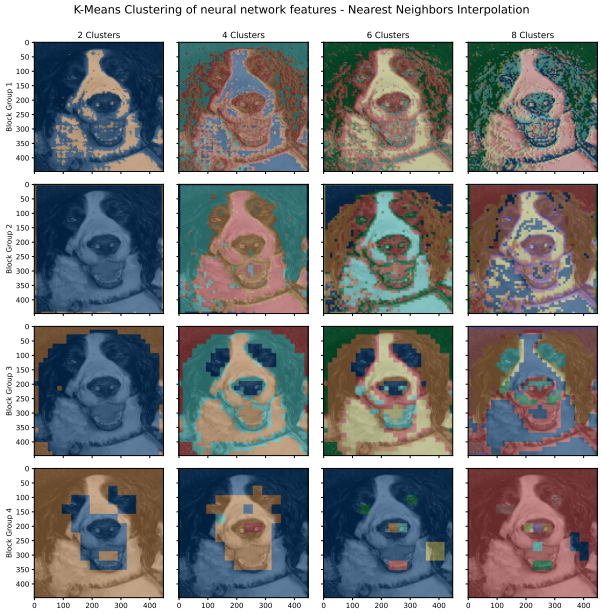

Figure 11. Visualizing features on a ImageNet validation image with K-means clustering. Each row denotes a type of local features extracted from a supervised contrastive learning setting, and each column denotes the number of K-means clusters.

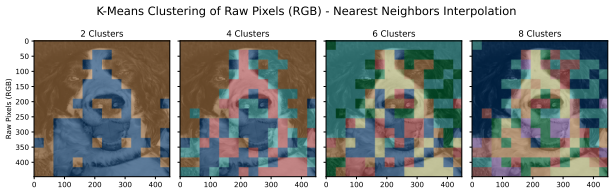

Figure 12. Visualizing clustered pixels on a ImageNet validation image with K-means clustering. Each column denotes the number of K-means clusters.

A.3 Experiment 3: Using different clustering methods

In this appendix, we show the cluster assignment masks corresponding to the features of block groups 1, 2, and 4 of various clustering methods of Experiment 3 of Subsection 4.2.1.

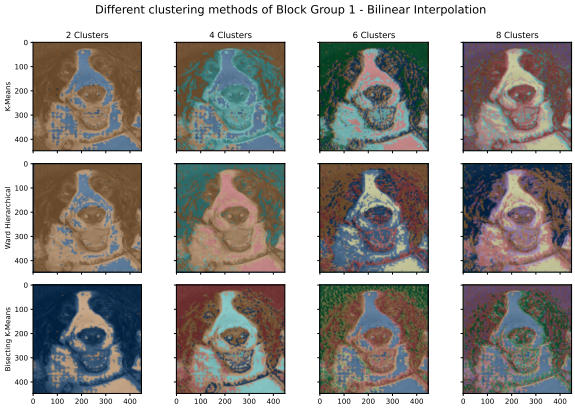

Figure 13. Visualizing features on an ImageNet validation image with different clustering methods. The features are extracted from block group 1 of SimCLRv2. Each row denotes a clustering method, and each column denotes the number of clusters.

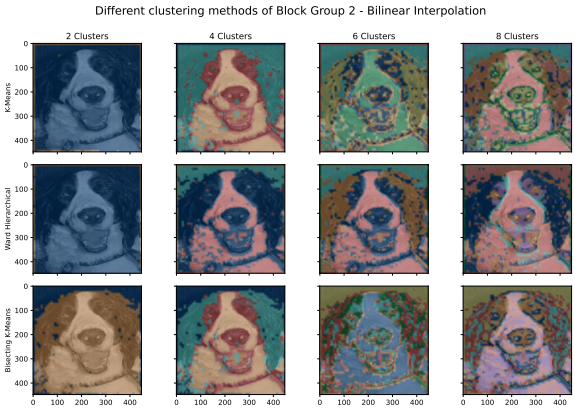

Figure 14. Visualizing features on an ImageNet validation image with different clustering methods. The features are extracted from block group 2 of SimCLRv2. Each row denotes a clustering method, and each column denotes the number of clusters.

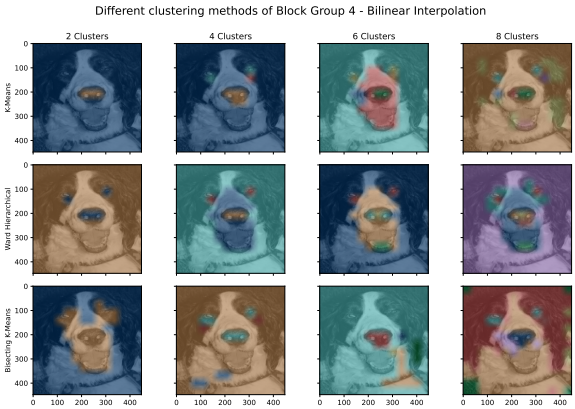

Figure 15. Visualizing features on an ImageNet validation image with different clustering methods. The features are extracted from block group 4 of SimCLRv2. Each row denotes a clustering method, and each column denotes the number of clusters.