Self-supervised visual representation learning: SimCLR & Deep Clustering

Dzodzoenyenye Adjowa Senanou, Prince Mensah, Najlaa Mohamed, Leonard Sanya
African Masters for Machine Intelligence (AMMI)
dsenanou@aimsammi.org, pmensah@aimsammi.org
nmohamed@aimsammi.org, lsanya@aimsammi.org

Abstract

Manual annotation of large collections of unlabeled data is extremely expensive. These large-scale datasets come with a variety of complexity and higher dimensionality. Several self-supervised methods have been developed to face challenges in analyzing unlabelled data. In this work, we provided a comprehensive review of two methods; Simple Contrastive Learning Representations (SimCLR) and Deep Clustering.

SimCLR learns robust data representations by simultaneously optimizing for contrastive similarity among augmented views of the same instance and dissimilarity between views of different instances.

Deep Clustering is a combination of Deep Learning and clustering approaches. So it generally retrieve information about features and perform clustering simultaneously basing on them. With the efficiency of deep learning, the accuracy of the clusters outperformed the standard clustering appproch.

1 Introduction

Building machine learning models typically requires large-scale annotated datasets, which demand significant manual annotation effort, time, and expense. To alleviate the burden of manual data annotation and reduce memory costs, self-supervised learning techniques have emerged as a viable solution [1]. These techniques leverage unsupervised learning capabilities to perform tasks traditionally requiring supervised learning. Unlike supervised learning, where ground truth labels are provided by humans, self-supervised learning generates these labels during the learning process from unstructured or unlabeled data.

Self-supervised learning is a notable form of unsupervised learning where "pseudo-labels" are derived directly from raw input data and used in place of human-annotated labels through pretext tasks [2]. This approach reduces the dependency on labeled data, making it scalable to large datasets. The technique is easily generalized across various tasks and domains, and the learned representations can be fine-tuned for specific tasks, showcasing its transfer learning capabilities.

Self-supervised learning typically involves two main stages: the pretext task (representation learning) and the downstream task [3]. The pretext task focuses on learning useful representations from data without needing labeled examples, while the downstream task applies these learned representations to specific applications, often incorporating labeled data. Multiple techniques exist for training self-supervised models, primarily divided into discriminative and generative methods. Contrastive learning, a discriminative approach, trains models to differentiate between pairs of data points, enabling the discovery of valuable features by comparing data pairs. In contrast, self-predictive techniques, a generative approach, train models to generate labels from the data itself, predicting certain aspects of the input data based on other elements.

Submitted to the Workshop on Instruction Tuning and Instruction Following at NeurIPS 2023. Do not distribute.

In this work , we focus on **SimCLR** and **Deep Clustering.**

2 SimCLR: A simple framework for constrastive learning of visual representations

SimCLR is a constrastive learning approach that maximises agreement across various augmented views of a given data example through a contrastive loss in a latent space, thereby learning representations.

2.1 SimCLR Architecture

There are four main parts to the SimCLR architecture.

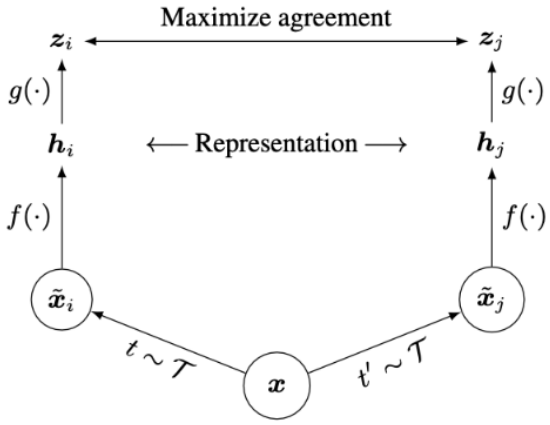


Figure 1: SimCLR Architecture [4]

1. **Data augmentation:** This module converts an example of data into two views of the same example that are related to one another. Operations for enhancing data include rotation, Gaussian blur, random colour distortion, random cropping, and more. After augmentation with operations t and $t^{'}$, x is shown in Figure 1 in two views: $\hat{x}_i$ and $\hat{x}_j$. Once augmentation is applied, we will have $2N$ training examples.
2. **Neural networks base encoder $f(\cdot)$:** Features that are representative of the input images are extracted. There are no restrictions when selecting a neural network architecture based on an encoder module thanks to the SimCLR framework. ResNet is the neural network architecture that is most frequently utilised. The results of applying the encoder are h_i and h_j, where $h = f(\hat{x}_i)$ and $h = f(\hat{x}_j)$ respectively, and are d-dimensional outputs following average pooling.
3. **Small neural networks projection head $g(\cdot)$:** Maps the representations to a new space where the contrastive loss function is applied. They used a one hidden layer MLP to obtain $z_i = g(h_i) = W^{(2)}\sigma(W^{(1)}h_i)$ where σ is a ReLU nonlinearity.
4. **Contrastive loss function:** The contrastive prediction task tries to find $\tilde{x}_j$ in $\{\tilde{x}_k\}_{k\neq i}$ for a given $\tilde{x}_i$, the collection $\{\tilde{x}_k\}$ includes a positive pair of samples $\tilde{x}_i$ and $\tilde{x}_j$. The other $2(N-1)$ augmented examples in a minibatch are treated as negative examples given a positive pair.
   Denote the dot product between l_2 normalised u and v (i.e., cosine similarity) as $\text{sim}(u, v) = u^T \cdot v/\|u\|\|v\|$. Next, we define the loss function for a positive pair of samples (i, j) as follows:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1[k \neq i] \exp(\text{sim}(z_i, z_k)/\tau)},$$

where $1[k \neq i] \in \{0, 1\}$ is an indicator function evaluating to 1 if $k \neq i$, 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.

Algorithm 1: SimCLR's main learning algorithm [4]

input: batch size N, constant τ, structure of f, g, T
for *sampled minibatch* $\{x_k\}_{k=1}^{N}$ **do**
  for *all* $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim T$, $t_0 \sim T$; // the first augmentation
    $\tilde{x}_{2k-1} = t(x_k)$; // augmented example
    $h_{2k-1} = f(\tilde{x}_{2k-1})$; // representation
    $z_{2k-1} = g(h_{2k-1})$; // projection
    $\tilde{x}_{2k} = t_0(x_k)$; // the second augmentation
    $h_{2k} = f(\tilde{x}_{2k})$; // representation
    $z_{2k} = g(h_{2k})$; // projection
  end
  for *all* $i \in \{1, \ldots, 2N\}$ *and* $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \frac{z_i^\top z_j}{k\|z_i\|\|z_j\|}$; // pairwise similarity
  end
  Define $\mathcal{L}(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} 1[k \neq i] \exp(s_{i,k}/\tau)}$; // positive pair loss
  $L = \frac{1}{2N} \sum_{k=1}^{N} [\mathcal{L}(2k-1, 2k) + \mathcal{L}(2k, 2k-1)]$; // total loss
  update networks f and g to minimize L
end
return encoder network $f(\cdot)$, and discard $g(\cdot)$

3 Deep Clustering

This approach is a clustering based approach of unsupervised learning that is being applied and studied in computer vision. Deep clustering technique simultaneously learns a neural network's parameters and clusters according to the generated features. Using the traditional clustering approach k-means, DeepCluster groups the features iteratively and updates the network's weights using the succeeding assignments as supervision.

3.1 DeepCluster Process Illustration

We used CIFAR-10 dataset which consists of 60,000 32x32 color images in 10 different classes, in the input as shown in Figure 2 passed through Convnet model (AlexNet Backbone) feature extraction. Principal Component Analysis (PCA) is then used to reduce the dimension of these characteristics in order to improve the efficiency of the clustering process and lower noise levels in the data. K-Means is used to cluster the reduced features, allocating a picture to a cluster.

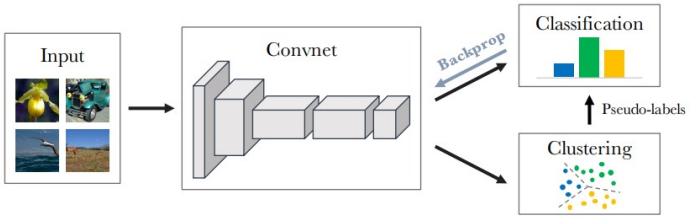


Figure 2: DeepCluster Process Illustration [2]

The convnet's characteristics $f_\theta(x_n)$ are sent into the k-means algorithm, which sorts them into k different groups according to a geometric criterion. In other words, it solves the following problem to jointly learn the cluster assignments y_n of each picture n and a $d \times k$ centroid matrix C.

$$\min_{C \in \mathbb{R}^{d \times k}} \frac{1}{N} \sum_{n=1}^{N} \min_{y_n \in \{0,1\}^k} \|f_\theta(x_n) - C y_n\|_2^2 \quad \text{such that} \quad y_n^\top 1_k = 1$$

For each $n \leq N$, solving this problem yields a centroid matrix C^* and a set of optimal assignments $\mathbf{y}_n^*$ [2]. As pseudo-labels, these cluster assignments are simply labels that are inferred from the data itself without any human annotation.These pseudo-labels are used to refine the Convet model during training. A loss function, cross-entropy is used to test the model's predictions against the pseudo-labels, and the model weights are adjusted to minimise this loss.

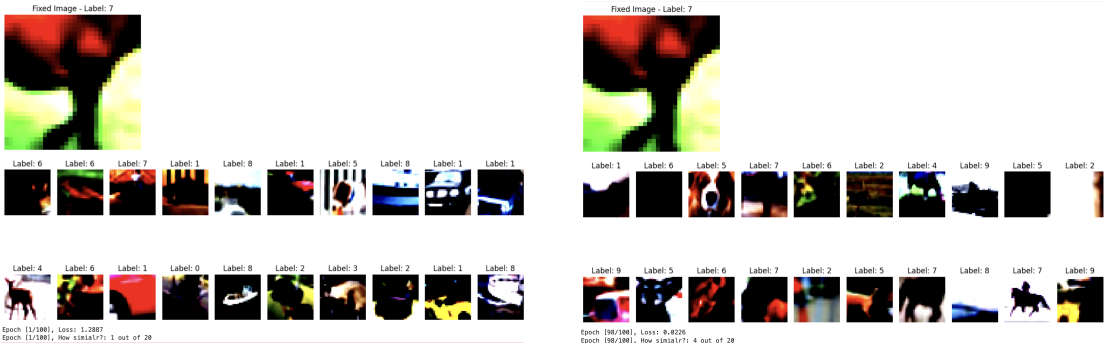
$$\min_{\theta, W} \frac{1}{N} \sum_{n=1}^{N} \ell(g_W(f_\theta(x_n)), y_n),$$

where ℓ is the multinomial logistic loss, also known as the negative log-softmax function.

4 Experiments

4.1 SimCLR Experiment 1:

In our experiments, we first trained the SimCLR model for 50 epochs and then for 100 epochs using this configuration: ResNet18 as the feature extractor, a projection head with two linear layers and a ReLU activation, a contrastive loss function, and the LARS optimizer. Despite extending training to 100 epochs, the results remained similar, suggesting that our batch size of 128 was insufficient specially that LARS optimizer performs the best with large batch size. This limitation due to memoryi limitations likely hindered the model from generating distinctive embeddings due to the lack of diverse negative samples.

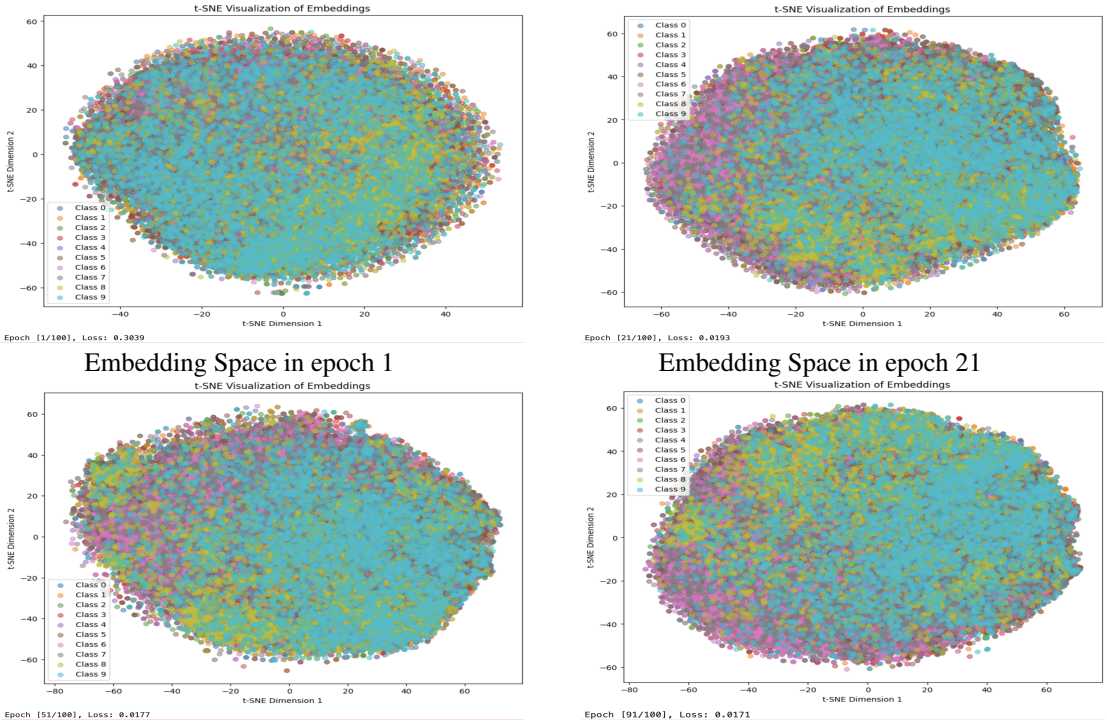


Closest Embeddings in epoch 1 Closest Embeddings in epoch 98

Figure 3: Closest 20 embeddings during training

4.2 SimCLR Experiment 2:

In the second experiment, T-SNE was used to visualize the embedding space throughout the 100 epochs. The embeddings showed minimal change, indicating that the small batch size constrained the model's ability to learn effectively.



Embedding Space in epoch 1 | Embedding Space in epoch 21

Embedding Space in epoch 51 | Embedding Space in epoch 91

Figure 4: Closest 20 embeddings during 100 epochs of training.

4.3 Deep Clustering

Our deep clustering experiment findings are compiled in Table 1. We found that the performance of our model was not up to par after training it on the CIFAR-10 dataset. Despite not being overly complex, CIFAR-10 nevertheless needs a more advanced model architecture in order to achieve greater generalisation. As such, we choose to train our model using the MNIST dataset, which is a simpler dataset. Our model performed noticeably better on MNIST, according to the results, demonstrating its usefulness on simpler datasets.

Method	Accuracy	
	MNIST	CFAR10
Supervised	0.982	0.7409
Self-supervised	0.9193	0.3536

Table 1: Performance of the

5 Applications of Self-Supervised Learning

- Computer Vision: Enhances image and video understanding tasks such as object detection, image classification, and segmentation [5].
- Medical Imaging: Improves analysis of medical images where labeled data is scarce [6].
- Robotics: Enhances visual perception in autonomous systems [7].

6 Conclusion

In this project we provided a solid review of SimCLR and a scalable clustering approach for the unsupervised.

The SimCLR paper introduces a solution to the challenge of limited labeled data, enabling improved performance on tasks like image classification, object detection, and image retrieval, through the use of contrastive learning. In our SimCLR implementation, the use of a batch size of 128 compromised performance, leading to poor representation learning. A T-SNE analysis over 100 epochs showed minimal changes in the embedding space with this batch size, indicating the need for longer training periods.

The DeepCluster alternates between updating the weights of the convolutional network by predicting the cluster assignments as pseudo-labels and clustering the features generated by the network using k-means. This method achieves better performance than the previous state-of-the-art on every standard transfer learning task.

Find here the link of the repository for SimCLR and DeepCluster implementation.

References

[1] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):857–876, 2023.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[3] Andrew Ng. Cs229 lecture notes. cs229 lecture notes, 1(1):1–3. 2000.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Zibei Wang. Self-supervised learning in computer vision: a review. In *International Conference on Computer Engineering and Networks*, pages 1112–1121. Springer, 2022.

[6] Saeed Shurrab and Rehab Duwairi. Self-supervised learning methods and applications in medical imaging analysis: A survey. *PeerJ Computer Science*, 8:e1045, 2022.

[7] Yongqiang Huang, Juan Wilches, and Yu Sun. Robot gaining accurate pouring skills through self-supervised learning and generalization. *Robotics and Autonomous Systems*, 136:103692, 2021.