A Survey on Contrastive Self-supervised Learning

Seri Lee
Computer Science and Engineering
Seoul National University
Seoul, Republic of Korea
sally20921@snu.ac.kr

***Abstract*—Self-supervised learning has gained popularity because of its ability to avoid the cost of annotating large-scale datasets. It is capable of adopting self-defined pseudo labels as supervision and use the learned representations for several downstream tasks. Specifically, contrastive learning has recently become a dominant component in self-supervised learning methods for computer vision, natural language processing, and other domains. It aims at emgedding augmented versions of the same sample close to each other while trying to push away embeddings from different samples. This paper provides an extensive review of self-supervised methods that follow the contrastive approach. The work explains commonly used pretext tasks in a contrastive learning setup, followed by different architectures that have been proposed so far. Next, we have a performance comparison of different methods for multiple downstream tasks such as image classification, object detection, and action recognition. Finally, we conclude with the limitations of the current methods and the need for further techniques and future directions to amke substantial progress.**

***Index Terms*—contrastive learning, self-supervised learning, discriminative learning, image/video classification, object detection, unsupervised learning, transfer learning**

I. Introduction

The advancements in deep learning [1] have elevated it to become one of the core components in most intelligent systems in existence. The ability to learn rich patterns from the abundance of data available today has made deep neural networks (DNNs) a compelling appraoch in the majority of computer vision (CV) tasks such as image classification, object detection, image segmentation, activity recognition as well as natural language processing (NLP) tasks such as sentence classification, language models, machine translation, etc. However, the supervised approach to learning features from labeled data has almost reached its saturation due to the intense labor required in manually annotating millions of data samples. This is because most of the modern computer vision systems (that are supervised) try to learn some form of image representations by finding a pattern between the data points and their respective annotations in large datasets. Works such as GRAD-CAM have proposed techniques that provide visual explanations for decisions made by a model to make them more transparent and explainable.

Traditional supervised learning approaches heavily rely on the amount of annotated training data available. Even though there's a plethora of data available out there, the lack of annotations has pushed researchers to find alternative approaches that can leverage them. This is where self-supervised methods play a vital role in fueling the progress of deep learning without the need for expensive annotations and learn feature representations where data itself provides supervision.

Supervised learning not only depends on expensive annotations but also suffers from issues such as generalization error, spurious correlations, and adversarial attacks. Recently, self-supervised learning methods have integrated both generative and contrastive approaches that have been able to utilize unlabeled data to learn the underlying representations. A popular approach has been to propose various pretext tasks that help in learning features using pseudo-labels. Tasks such as image inpainting, colorizing greyscale images, jigsaw puzzles, super-resolution, video frame prediction, audio-visual correspondence, etc have proven to be effective for learning good representations.

Generative models have gained its popularity after the introduction of Generative Adversarial Networks (GANs) in 2014. The work later became the foundation for many successful architectures such as CycleGAN, StyleGAN, PixelRNN, Text2Image, DiscoGAN, etc. These methods inspired more researchers to switch to training deep learning models with unlabeled data in a self-supervised setup. Despite their success, researchers started realizing some of the complications in GAN-based approaches. They are harder to train because of two main reasons: (a)non-convergence: the model parameters oscillate a lot and rarely converge, and (b) the discriminator gets too successful that the generator network fails to create real-like fakes due to which the learning cannot be continued. Also, proper synchronization is required between the generator and the discriminator that prevents the discriminator to converge and the generator to diverge.

Unlike generative models, contrastive learning is a discriminative approach that aims at grouping similar samples closer and diverse samples far from each other. To achieve this, a similarity metric is used to measure how close two embeddings are. Especially, for computer vision tasks, a contrastive loss is evaluated based on the feature representations of the image extracted from an encoder network. For instance, one sample from the training dataset is taken and a transformed version of the sample is retrieved by applying appropriate data augmentation techniques. During training, the augmented version of the original sample is considered as a positive sample, and the rest of the samples in the batch/dataset are considered negative samples. Next, the model is trained in a way that it learns to differentiate positive samples from the negative ones.

The differentiation is achieved with the help of some pretext task. In doing so, the model learns quality representations of the samples and is used later for transferring knowledge to downstream tasks. This idea is advocated by an interesting experiment conducted by Epstein in 2016, where he asked his students to draw a dollar bill with and without looking at the bill. The results from the experimentation show that the brain does not require complete information of a visual piece to differentiate one object from another. Instead, only a rough representation of an image is enough to do so.

Most of the earlier works in this area combined some form of instance-level classification approach with contrastive learning and were successful to some extent. However, recent methods such as SwAV, MoCo, and SimCLR with modified approaches has produced results comparable to the state-of-the-art supervised method on ImageNet dataset. Similarly, PIRL, Selfie are some papers that reflect the effectiveness of the pretext tasks being used and how they boost the performance of their models.

II. Pretext Tasks

Pretext tasks are self-supervision tasks that act as an important strategy to learn representations of the data using pseudo labels. These pseudo labels are generated automatically based on the attributes found in the data. The learned model from the pretext task can be used for downstream tasks such as classification, segmentation, detection etc. Furthermore, these tasks can be applied to any kind of data such as image, video, speech, signals, and so on. For a pretext task in contrastive learning, the original image acts as an anchor, its augmented version acts as a positive sample, and the rest of the images in the batch or in the training data act as negative samples.

Most of the commonly used pretext tasks are divided into four main categories: color transformation, geometric transformation, context-based tasks, and cross-modal based tasks. These pretext tasks have been used in various scenarios based on the problem intended to be solved.

A. Color Transformation

Color transformation involves basic adjustments of color levels in an image such as blurring, color distortions, converting to grayscale, etc. During this pretext task, the network learns to recognize similar images invariant to their colors.

B. Geometric Transformation

A geometric transformation is a spatial transformation where the geometry of the image is modified without altering its actual pixel information. The transformations include scaling, random cropping, flipping, etc. Here the original image is considered as the global view and the transformed version is considered as local view. Chen et al. performed such transformations to learn features during pretext task.

C. Identifying the right pre-text task

The choice of pretext task relies on the type of problem being solved. Although numerous methods have been proposed in contrastive learning, a separate track of research is still going on to identify the right pre-text task. Work has identified and proved that it is important to determine the right kind of pre-text task for a model to perform well with contrastive learning. The main aim of a pre-text task is to compel the model to be invariant to these transformations while remaining discriminative to other data points. But the bias introduced through such augmentations could be a double-edged sword, as each augmentation encourages invariances to a transformation which can be beneficial in some cases and harmful in others. For instance, applying rotation may help with view-independent aerial image recognition but might significantly downgrade the performance while trying to solve downstream tasks such as detecting which way is up in a photograph for a display application. Similarly, colorization-based pretext tasks might not work out in a fine-grain classification.

III. Architectures

Contrastive learning methods rely on the number of negative samples for generating good quality representations. It can be seen as a dictionary-lookup task where the dictionary is sometimes the whole training set and the rest of the times some subset of the dataset. An interesting way to categorize these methods would be based on the technique used to collect negative samples against a positive data-point during training. Based on the approach taken, we categorized the methods into four major architectures. Each architecture is explained separately along with examples of successful methods that follow similar principles.

A. End-to-End Learning

End-to-End is a complex learning system that uses gradient-based learning and is designed in such a way that all modules are differentiable. This architecture prefers large batch sizes to accumulate a greater number of negative samples. Except for the original image and its augmented version, the rest of the images in the batch are considered negative. The pipeline employs two encoders: a Query encoder (Q) and a Key encoder (K). The two encoders can be different and are updated end-to-end by backpropagation during training. The main idea behind training these encoders separately is to generate distinct representations of the same sample. Using a contrastive loss, it converges to make positive samples closer and negative samples far from the original sample. Here, the query encoder Q is trained on the original samples and the key encoder k is trained on their augmented versions (positive samples) along with the negative samples in the batch. The features q and k generated from these encoders are used to calculate the similarity between the respective inputs using a similarity metric. Most of the time, the similarity metric used is cosine similarity which is simply the inner product of two vectors normalized to have length 1.

Recently, a successful end-to-end model was proposed in SimCLR where they used a batch size of 4096 for 100 epochs. It has been verified that end-to-end architectures are simple in complexity but perform better with large batch sizes and higher number of epochs.

The number of negative samples available in this approach is coupled with the batch size as it accumulates negative samples from the current batch. Since the batch size is limited by the GPU memory size, the scalability factor with these methods remains an issue. Furthermore, for larger batch sizes, the methods suffer from a large mini-batch optimization problem and require effective optimization strategies.

B. Using a Memory Bank

With potential issues from having large batch sizes that could inversely impact the optimization during training, a possible solution is tot maintain a separate dictionary known as memory bank. The aim of maintaining a memory bank is to accumulate a large number of feature representations of samples that are used as negative samples during training. For this purpose, a dictionary is created that stores and updates the embeddings of samples with the most recent ones at regular intervals. The memory bank (M) contains a feature representation m_I for each sample I in dataset D. The representation m_I is an exponential moving average of feature representations that were computed in prior epochs. It enables replacing negative samples $m_{I'}$, by their memory bank representations without increasing the training batch size.

The representation of a sample in the memory bank gets updated when it is last seen, so the sampled keys are essentially about the encoders at multiple different steps all over the past epoch. PIRL is one of the recent successful methods that learns good visual representations of images trained using a memory bank. It requires the learner to construct representations of images that are covariant to any of the pretext task being used, though they focus mainly on the Jigsaw pretext task. Another popular work that uses a memory bank under contrastive setting was proposed by Wu et al. where they implemented a non-parametric variant of softmax classifier that is more scalable for big data applications.

However, maintaining a memory bank during training can be a complicated task. One of the potential drawbacks of this approach is that it can be computationally expensive to update the representations in the memory bank as the representations get outdated quickly in a few passes.

C. Using a Momentum Encoder

To address the issues with a memory bank, the memory bank gets replaced by a separate module called Momentum Encoder. The momentum encoder generates a dictionary as a queue of encoded keys with the current mini-batch enqueued and the oldest mini-batch dequeued. The dictionary keys are defined on-the-fly by a set of data samples in the batch during training. The momentum encoder shares the same parameters as the encoder Q. It is not backpropagated after every pass, instead, it gets updated based on the parameters of the query encoder.

Though the keys in the queue are encoded by different encoders (in different mini-batches), the difference among these encoders can be made small.

The advantage of using this architecture over the first two is that it does not require training two separate models. Furthermore, there is no need to maintain a memory bank that is computationally and memory inefficient.

D. Clustering Feature Representations

All these architectures explained above focus on comparing samples using a similarity metric and try to keep similar items closer and dissimilar items far from each other allowing the model to learn better representations. On the contrary, this architecture follows an end-to-end approach with two encoders that share parameters, but instead of using instance-based contrastive approach, they utilize a clustering algorithm to group simiilar features together.

One of the most recent works that employ clustering methods is SwaV. The diagram points out the differences between other instance-based contrastive learning architectures and the clustering-based methods. Here, the goal is not only to make a pair of samples close to each other, but also make sure that all other features that are similar to each other form clusters together. For example, in an embedded space of images, the features of cats should be closer to features of dogs but should be far from the featurse of houses.

In instance-based learning, every sample is treated as a discrete class in the dataset. This makes it unreliable in conditions where it compares an input sample against other samples from the same class that the original sample belongs to. To explain it clearly, imagine we have an image of a cat in the training batch that is the current input to the model. During this pass, all other images in the batch are considered as negative. The issue arises when there are images of other cats in the negative samples. This condition forces the model to learn two images of cats as not similar during training despite both being from the same class. This problem is implicitly addressed by a clustering-based approach.

IV. Encoders

Encoders play an integral role in any self-supervised learning pipeline as they are responsible for mapping the input samples to the latent space. Without effective feature representations, a classification model might have difficulty in learning to distinguish among different classes. Most of the works in contrastive learning utilitze some variant of ResNet model. Among its variants, ResNet-50 has been the most widely used because of its balance between size and learning capability.

In an encoder, the output from a specific layer is pooled to get a single-dimensional feature vector for every sample. Depending on the approach, they are either upsampled or downsampled. For example, in the work proposed by Misra et al., a ResNet-50 architecture is used where the output of the res5 (residual block) features are average-pooled to get a 2048-dimensional vector for a given sample. They further apply a single linear projection to get a 128-dimensional feature vector. Also, as part of their ablation test, they investigated features from various stages such as res2, res3, and res4 to evaluate the performance. As expected, features extracted from the later stages of the encoder proved to be a better representation of the input than the features extracted from the output of the average pooling layer. Further, a shallow MLP (1 hidden layer) maps representations to a latent space where a contrastive loss is applied. For training a model for action recognition, the most common approach is to extract features from a sequence of image frames is to use use a 3D-ResNet as encoder.

V. Training

To train an encoder, a pretext task is used that utilizes contrastive loss for backpropagation. The central idea in contrastive learning is to bring similar instances closer and push away dissimiilar instances far from each other. One way to achieve this is to use a similarity metric that measures the closeness between the embeddings of two samples. In a contrastive setup, the most common similarity metric used is cosine similarity that acts as a basis for different contrastive loss functions.

References

[1] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*, 1(2), 2020.