DL HW5

Mehdi Jamalkhah

October 2024

1 Contrastive Learning

1.1 SimCLR, Moco, and BYOL

1.1.1

Initially, we proposed the idea that we could learn meaningful features for our images by comparing an image with its augmentation. To achieve this, we envisioned a model that takes an image and its augmentation as inputs and optimizes them to be close to each other. However, we encountered a significant issue: our model could potentially cheat by mapping all images to a single point. While this optimization might appear successful, it ultimately fails to learn any useful features.

To address this, we suggested maximizing the distance between each image and other images, which is referred to as creating negative pairs.

Another approach, as introduced in the BYOL paper, involves forcing the model to map an image and its augmentation to distinct points. Specifically, this method employs two submodels with the same architecture; however, only one of them is updated using gradients, while the other is updated through a momentum average of the first. This strategy effectively prevents the collapse of the two images, because these models cannot have same weights.

1.1.2

Training with large batch size may be unstable when using standard SGD/Momentum with linear learning rate scaling. To stabilize the training, they use the LARS optimizer.

In distributed training with data parallelism, the BN mean and variance are typically aggregated locally per device. In this contrastive learning, as positive pairs are computed in the same device, the model can exploit the local information leakage to improve prediction accuracy without improving representations. This paper address this issue by aggregating BN mean and variance over all devices during the training. Other approaches include shuffling data examples across devices, or replacing BN with layer norm.

1.2 DINO

1.2.1

The DINO paper interpret the momentum teacher in DINO as a form pf Polyak-Ruppert averaging , with an exponentially decay. Polyak-Ruppert averaging is often used to simulate model ensembling to improve the performance of a network at the end of the training. This method can be interpreted as applying Polyak-Ruppert averaging during the training to constantly build a model ensembling that has superior performances. This model ensembling then guides the training of the student network.

1.2.2

DINO discovering the semantic layout of scenes through multi-crop training. The teacher network needs to accurately detect objects to recognize specific parts as belonging to those objects, effectively bringing them closer together in the process.

1.2.3

There are two forms of collapse: regardless of the input, the model output is uniform along all the dimensions or dominated by one dimension.

DINO work with a centering and sharpening of the momentum teacher outputs to avoid model collapse. The centering avoids the collapse induced by a dominant dimension, but encourages an uniform output. Sharpening induces the opposite effect. The paper shows this complementarity by decomposing the cross-entropy H into an entropy h and the Kullback-Leibler divergence ("KL") D_{KL}:

$$H(P_t, P_s) = h(P_t) + D_{KL}(P_t|P_s)$$

1.2.4

- **Fast and memory-efficient attention.** They aim to configure the hyperparameters to fully utilize the capacity of their GPU. For instance, they found out that the efficiency is best when the embedding dimension per head is a multiple of 64, and the matrix operations are even better when the full embedding dimension is a multiple of 256
- **Sequence packing.** The DINO algorithm processes both large crops (224 resolution) and small crops (98 resolution), which are represented by token sequences of different lengths. To speed up training, a technique called "sequence packing" is employed, originating from NLP. This involves concatenating the sequences into a single long sequence, which is then passed through the transformer blocks. A block-diagonal mask in the self-attention layers ensures that attention is only applied within each sequence, maintaining equivalence to separate forward passes. This approach significantly improves computational efficiency compared to previous methods. The lower-level components are available in the xFormers library.
- **Fully-Sharded Data Parallel (FSDP).** In order to reduce the memory footprint per GPU, they split the model replicas across GPUs. Consequently, the model size is not bounded by the memory of a single GPU but by the total sum of GPU memory across compute nodes.

1.2.5

DINO takes its inspiration from BYOL but operates with a different **similarity matching loss** and uses **the exact same architecture** for the student and the teacher. That way, DINO completes the interpretation initiated in BYOL of self-supervised learning as a form of Mean Teacher self-distillation with no labels.

2 GNN

2.1

2.1.1

Yes.

2.1.2

No, because each neighbor is multiplied by a different weight matrix. Therefore, if the neighbors are permuted, the result will change, indicating that it is not invariant.

2.1.3

Yes.

2.2

At the $t+1$ level, we aim to construct $h^{(t+1)}$ using $h^{(t)}$. After applying a permutation on $H^{(t)}$, the node v will be transformed into v_p. and the hidden state $h_v^{(t)}$ will transformed into $s_{v_p}^{(t)}$

$$\begin{aligned} s_{v_p}^{(t+1)} &= \sigma\left(W s_{v_p}^{(t)} + \sum_{u \in \mathcal{N}(v_p)} W' s_u^{(t)}\right) \\ &= \sigma\left(W h_v^{(t)} + \sum_{u \in \mathcal{N}(v)} W' h_u^{(t)}\right) \\ &= h_v^{(t+1)} \end{aligned}$$

By inductive proof, this process occurs for all layers. The permutation of inputs will result in a corresponding permutation of outputs, leading to the following conclusion:

$$PH^{(t+1)} = H_P^{(t+1)}$$

3 Universal Adversarial Perturbations

3.1

Let μ denote a distribution of images in $\mathcal{R}^d$, and $\hat{k}$ define a classification function that outputs for each image $x \in \mathcal{R}^d$ an estimated label $\hat{k}(x)$. The main focus of universal adversarial perturbations is to seek perturbation vectors $v \in \mathcal{R}^d$ that fool the classifier $\hat{k}$ on almost all datapoints sampled from μ. That is, it seek a vector v such that:

$$\hat{k}(x+v) \neq \hat{k}(x) \text{ for "most" } x \sim \mu$$

3.2

The existence of these perturbations is problematic when the classifier is deployed in real-world (and possibly hostile) environments, as they can be exploited by adversaries to break the classifier.

By identifying vulnerabilities in the model, we can enhance its robustness against such weaknesses.

3.3

$$\begin{aligned} &\underset{v}{\text{argmax}} \ \frac{1}{n} \sum_{i=1}^{n} g(x_i + v) \\ &\text{s.t. } ||v||_p < \epsilon \end{aligned}$$

4 VLM

4.1 CLIP vs. SimVLM vs. CoCa

Similarities:

1. All three architectures are Vision-Language Models (VLMs), aiming to establish a shared embedding space for both text and images (though not exactly in SimVLM).
2. Each model includes an image encoder.

Dissimilarities:

1. CLIP is limited to retrieval tasks, whereas SimCLR and CoCa include a text decoder, enabling them to generate captions.
2. SimCLR is capable of performing Visual Question Answering (VQA) tasks, while CLIP and CoCa do not support this functionality.

4.2 CLIP

4.2.1

- **Image Encoder.** It encodes the image into a vector of features. The original paper explores two architectures for this component: ResNet and Vision Transformer (ViT).
- **Text Encoder.** It encodes the text into a vector of features. The text encoder is a Transformer . As a base size CLIP use a 12-layer 512-wide model with 8 attention heads. The transformer operates on a lower-cased byte pair encoding (BPE) representation of the text.
- **Projection Head.** It's a linear projection to map from each encoder's representation to the multi-modal embedding space.

4.2.2

$$\mathcal{L}_1 = -\frac{1}{N}\log\frac{e^{s_{ii}}}{\sum_j e^{s_{ij}}} = -\frac{1}{N}\left[s_{ii} - \log\sum_j e^{s_{ij}}\right]$$

where,

$$s_{ij} = \frac{x_i.y_j}{\tau\|x_i\|\|y_j\|}$$

So,

$$\frac{\partial\mathcal{L}_1}{\partial x_i} = -\frac{1}{N}\left[\frac{\partial s_{ii}}{\partial x_i} - \frac{\sum_j e^{s_{ij}}(\frac{\partial s_{ij}}{\partial x_i})}{\sum_j e^{s_{ij}}}\right]$$

where,

$$\begin{aligned}\frac{\partial s_{ij}}{\partial x_i} &= \frac{y_j\tau\|x_i\|\|y_j\| - \tau\frac{x_i}{\|x_i\|}\|y_j\|(x_i.y_j)}{\tau^2\|x_i\|^2\|y_j\|^2} \\ &= \frac{1}{\tau\|x_i\|\|y_j\|}\left(y_j - \frac{(x_i.y_j)}{\|x_i\|^2}x_i\right)\end{aligned}$$