Mid Term Project Report
Exploring the Efficacy of the Self-Supervised Learning: A Deep Dive into the DINO Algorithm

Atharv Ramesh Nair
EE20BTECH11006

Nithish S
EE20BTECH11044

Ajit Shankar
ES20BTECH11003

Theresa Karra
EE20BTECH11050

1. Introduction

In recent years, the field of deep learning has seen remarkable advancements, particularly within computer vision. One emerging approach that stands out is self-supervised learning, a paradigm that circumvents the need for labeled data, allowing models to benefit from the abundance of available unlabeled data.

Our investigation takes inspiration primarily from the work of Caron et al. [2] on the DINO (Self Distillation with No Labels) algorithm. The promising outcomes exhibited by DINO [2] and its subsequent version, DINO V2 [8], are noteworthy. It is significant that models trained with DINO V2 have demonstrated competitive performance across several downstream tasks, such as image classification, segmentation, depth estimation, and instance retrieval. The DiNo approach stands in contrast to prior contrastive learning methods, which necessitated the identification of positive and negative samples for any given image. Our primary objective is to discern the capabilities of these self-supervised methods, contrasting them against traditional supervised techniques to comprehend their strengths and potential limitations. A comprehensive grasp of Vision Transformers (ViTs) is foundational for the elucidation and application of the aforementioned methodologies. A dataset emblematic of the potential of self-supervised learning is the Optical Coherence Tomography (OCT) dataset proffered by Prabhushankar et al. [7]. This dataset encompasses approximately 78,000 images, of which a mere subset, roughly 9,400, are annotated. Our next endeavors will primarily focus on this dataset, among others with analogous characteristics, using the DiNo technique. Parallely, we also wish to gauge the efficacy of the DiNo Algorithm in diverse downstream tasks.

The rest of this report is as follows: Section 2 delves into the architecture of the Vision Transformer as proposed by Dosovitskiy et al. [4]. Section 3 provides an overview of prevailing self-supervised methods. In Section 4, the DINO algorithm is discussed comprehensively. Some very noteworthy and interesting results obtained using the DiNo model [2] are presented in Section**??**. Finally, Section 6 presents our proposed study on the application of self-supervised learning, particularly within the context of the OCT Dataset, characterized by its scarcity of labels along with some innovative ideas of downstream tasks based on models trained using DiNo approach

2. Vision Transformers

Vision Transformers (ViTs) [4] apply the transformer architecture, originally designed for Natural Language Processing tasks [10], to vision tasks. Instead of relying on convolutional layers, ViTs capitalize on the self-attention mechanism. Convolutional Neural Networks (CNNs) have been the mainstay in computer vision for some time since they exploit local spatial hierarchies through convolutional layers. On the contrary, Transformers are able to capture both local and global dependencies.

ViT Components:

- **Image Patching:** Images are divided into fixed-size patches, like 16×16 or 8×8, which once flattened become tokens, analogous to word embeddings in NLP.
- **Linear Projection and Positional Embedding:** The patches undergo linear embedding into a higher-dimensional space. A Class Token is added to the start, followed by Positional Embeddings for spatial context. Both the class token and positional embeddings are trainable.
- **Transformer Encoder:** Leveraging the transformer encoder from [10], it consists of alternate layers of multi-headed self-attention mechanisms and MLP blocks. Before and after each block, Layer normalization and residual connections are applied, respectively.

$$z_0 = [x_{\text{class}}; x_1 E; x_2 E; \dots; x_{N-1} E] + E_{\text{pos}} \tag{1}$$

Z_o is the input to the encoder

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1, \dots, L \tag{2}$$
$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \quad \ell = 1, \dots, L \tag{3}$$
$$y = \text{LN}(z_L^0) \tag{4}$$

- **Classification Head:** The output corresponding to the Class Token (y) is extracted post-transformer and passed to an MLP Layer for the final classification.

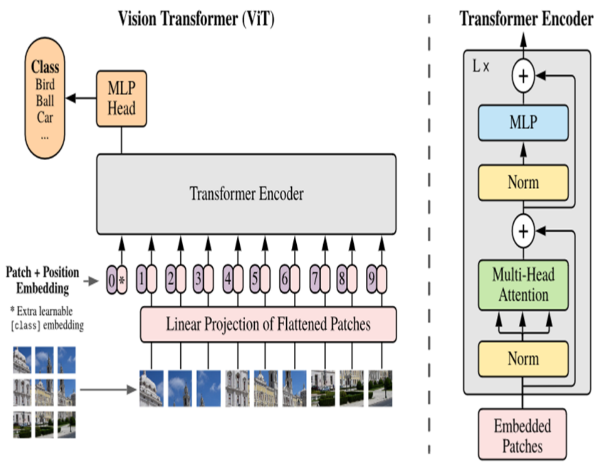

Figure 1. ViT Model Architecture

3. Self Supervised Learning

Self-supervised learning (SSL) is a machine learning approach that falls between supervised learning and unsupervised learning [1]. As mentioned earlier, self-supervised learning techniques work on unlabelled data to generate the labels, that can be used as ground truths for subsequent downstream tasks. In addition to the fact that SSL methods work on vast unlabelled data, it generates more **generic representations of the raw input data**, that serve as efficient features to solve various downstream tasks. But, converging to an efficient architecture for the self-supervised learning model is difficult. In the NLP domain, models such as BERT (Bidirectional Encoder Representations of Transformers), GPT (Generative Pre-trained Transformers), and T5 (Text-to-text Transfer Transformer) are some of the salient SSL models that are used today.

Self-supervised learning can be categorized into 4 categories, namely – the deep metric learning family, the self-distillation family, the canonical correlation analysis, and the masked image modeling family.

- The deep metric learning family of methods is based on the principle of encouraging similarity between semantically transformed versions of input. **SimCLR** learns visual representations by encouraging similarity between two augmented views of an image, instead of using sampling techniques and then checking for similarity in the sampled images, which was done before SSL was introduced. Many loss functions such as **Constrative loss** are used to measure the similarity between these augmented images. SSL makes sure that negative samples are close to each other but are far from the positive samples of the image to form a more challenging learning objective.

- Self-distillation methods such as BYOL, SimSIAM, and DINO, along with their variants rely on a simple mechanism: feeding two different views to two encoders and mapping one to the other by means of the predictor. To prevent the encoders from collapsing by predicting a constant value for any input, various techniques have been employed.

- **BYOL** (Bootstrap Your Own Latent) first introduced self-distillation to avoid collapse. It uses two networks along with a predictor to map the outputs of one network to the other. The output predicting network is often called the *student network*, while the target producing network is often called the *teacher network*. Each network is trained on different views of the image. Weights of the student network are updated using gradient descent, whereas weights of the teacher network are updated using exponential moving average (EMA), which creates **asymmetry** leading to the successful working of the algorithm.

- **SimSiam** is aimed at understanding the importance of each component of the BYOL algorithm and showed that EMA was not necessary in practice. Both approaches focus on the fact that asymmetry between the two branches, and the training dynamics used to regularize variance of the embeddings are the key features that lead to the success of the algorithm.

- **DINO** performs a centering of the output of the student network using a running mean (to avoid sensitivity to mini-batch size) and discretizes the representations by means of a softmax, and thus can be interpreted as an online clustering mechanism of DINO architecture. There are many other methods that belong to the self-distillation family such as MoCo, ISD, SSCD, MSF, and so on.

- SSL canonical correlation analysis (CCA) family exploits the cross-covariance property of two variables. In the deep learning setting, this idea was exploited to build Deep Canonically Correlated Auto Encoders (DCCAE), which is an autoencoder regularized via CCA. SSL methods that use CCA ideologies include VICReg, Barlow Twins, SWAV, and W-MSE. **VICReg**, the most recent of these methods balances three objectives based on co-variance matrix representations from two views: variance, invariance, and covariance. Regularizing variance along each dimension of the representation prevents collapse, the invariance ensures two views are encoded similarly, and the covariance encourages different dimensions of the representation to capture different features.

- Masked Image Modelling (MIM), inspired by BERT (uses a masked token as input and teaches the model

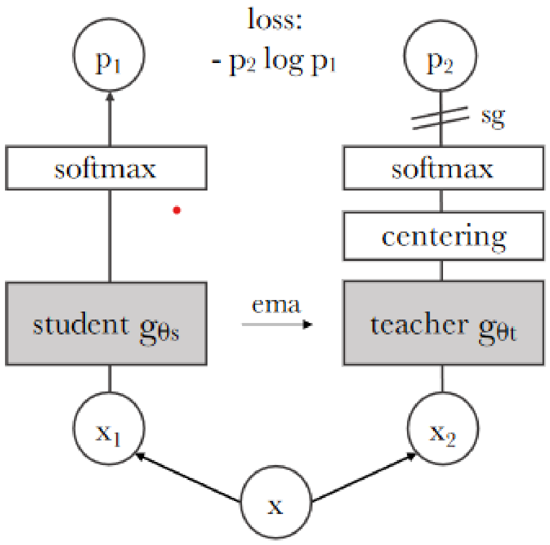

Figure 2. Self Distillation with No Labels

to recover the original text, is a masked language modeling technique), is a pre-training vision strategy that helps the model to learn to paint the masked out portions of the image. However, the performance of this technique was not comparable to the performance of supervised learning methods. This method was suitable for classification tasks in language models but was not effective for classification tasks in image models. To implement this for image classification and other downstream tasks, the authors cast MIM as a regression problem, using an autoencoder to encode image patches as discrete tokens, and then pre-training the transformers to predict the discrete token values for masked tokens. This is the BEiT method followed by the authors to improve the performance of MIM. Many modifications to this approach have been implemented to improve the performance of the algorithm and perform better than the available supervised methods.

- The most successful approaches when using a frozen encoder, iBOT, and DINOV2 employ a mix of masked image modeling and more classical approaches such as self-distillation.

4. DINO

In this section, we will briefly go through the Dino (Self **Di**stillation with **No** Labels). The approach can be clearly understood from Figure 2. The pseudocode 3 explains the Algorithm in detail. Both these have been taken directly from [2]. DiNo involves training a student network g_{θ_s} to emulate the outputs of a designated teacher network g_{θ_t}, with parameters dictated by θ_s and θ_t, respectively. This is a type of knowledge Distillation. Given augmentations (x_1,x_2) of the input image x, both networks yield probability distributions across K dimensions, denoted P_s and P_t. The probability P is obtained by normalizing the output of the network g with a softmax function. More precisely P_s is given by,

Algorithm 1 DINO PyTorch pseudocode w/o multi-crop.

```
# gs, gt: student and teacher networks
# C: center (K)
# tps, tpt: student and teacher temperatures
# l, m: network and center momentum rates
gt.params = gs.params
for x in loader: # load a minibatch x with n samples
    x1, x2 = augment(x), augment(x) # random views

    s1, s2 = gs(x1), gs(x2) # student output n-by-K
    t1, t2 = gt(x1), gt(x2) # teacher output n-by-K

    loss = H(t1, s2)/2 + H(t2, s1)/2
    loss.backward() # back-propagate

    # student, teacher and center updates
    update(gs) # SGD
    gt.params = l*gt.params + (1-l)*gs.params
    C = m*C + (1-m)*cat([t1, t2]).mean(dim=0)

def H(t, s):
    t = t.detach() # stop gradient
    s = softmax(s / tps, dim=1)
    t = softmax((t - C) / tpt, dim=1) # center + sharpen
    return - (t * log(s)).sum(dim=1).mean()
```

Figure 3. DiNo Algorithm

$$P_s(x)^i = \frac{\exp(g_s(x)^i/T_s)}{\sum_{k=1}^{K}\exp(g_s(x)^k/T_s)} \tag{5}$$

Similarly, for P_t:

$$P_t(x)^i = \frac{\exp(g_t(x)^i/T_t)}{\sum_{k=1}^{K}\exp(g_t(x)^k/T_t)} \tag{6}$$

with $T_s, T_t > 0$, temperature parameters that control the sharpness of the distribution.

Given a fixed teacher g_t, the objective of the algorithm is to optimize the parameters of the student network, θ_s, such that the distributions produced by the teacher and student networks are aligned. This is achieved by minimizing the cross-entropy loss between the output distributions of the two networks.

$$\min_{\theta_s} H(P_t(x), P_s(x)) \tag{7}$$

In the computation of the Cross Entropy Loss, as illustrated in Algorthm 1 3, the teacher probabilities are centered before applying the softmax function. Specifically, in the function $H(t, s)$, the teacher's output t has the center C subtracted before it is passed through the softmax. The center C is updated with an exponential moving average as shown in the Algorithm 1 3.

Now, let's discuss how the Cross Entropy is adapted for self-supervised learning. From a given image, a set V of different views is generated. This set consists two global views, x_1^g and x_2^g, and several local views (generally 4) of finer resolution. The student processes all crops, whereas only the global views are processed by the teacher. The Loss is computed as the aggregated cross-entropy between all pairs of augmented views, with the exception of comparisons involving identical augmentations. The loss is minimized as:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{x' \in V, x' \neq x} H(P_t(x), P_s(x')).$$

However, the standard setting for multi-crop utilizes two global views at a resolution of 224^2, covering a significant area (more than 50%) of the original image, and several local views at a resolution of 96^2, covering smaller sections (less than 50%) of the original image.

During training, the teacher network parameters are freezed. It is built using past iterations of the student network using an exponential moving average method (a.k.a momentum encoder).

$$\theta_t = \lambda\theta_t + (1 - \lambda)\theta_s \tag{8}$$

The authors claim that the teacher has better performance than the student during training due to an ensembling effect due to the EWMA method. The neural network used for both the student and the teacher is the same. The neural network consists of a backbone (ViT or ResNet [5]) and a projection head The projection head consists of a 3-layer multi-layer perceptron (MLP) with hidden layer followed by a l_2 normalization and a weightnormalized fully connected layer [9] with K dimensions. Batch-Normalisation is not used in the architecture.

As per the authors, teacher centering along with sharpening introduced through the temperature coefficients τ_t and τ_s is essential for avoiding collapse. There are two types of collapse possible. One is where the output is uniform irrespective of the input. The other is where the output is dominated by one feature. Both of these cases are obviously not desirable. The centering avoids the collapse induced by a dominant dimension, but encourages an uniform output. Sharpening induces the opposite effect. The cross entropy H can be broken into an entropy h term and a KL-Divergence D_{KL} term:

$$H(P_t, P_s) = h(P_t) + D_{KL}(P_t \| P_s). \tag{9}$$

When the KL value is zero, it signifies that the output remains constant, leading to a scenario described as a collapse. As depicted in Fig 4, [2] plot the entropy and KL throughout the training process, both with and without the

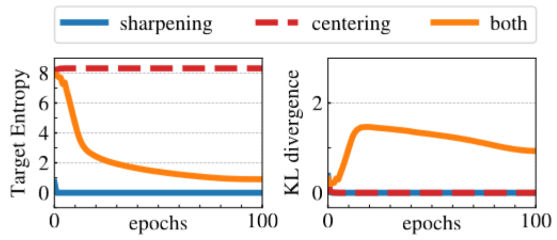

Figure 4. Analysis of Collapse

application of centering and sharpening. The absence of either operation causes the KL to approach zero, highlighting the collapse. On the other hand, the entropy, denoted as h, tends to vary in its values: it reaches 0 when devoid of centering and touches $-\log(1/K)$ without sharpening. This observation underscores the distinct collapse tendencies introduced by each operation. Combining both centering and sharpening provides a counterbalance to these tendencies.

5. Experiments with DINO SSL Algorithm

As a part of our experimentation, we first decided to test out the DiNo Algorithm on the imagenette dataset from fastai. This is a smaller dataset containing 12,954 images with a 70/30 Train-Validation split. This dataset is ideal for running preliminary tests which will help us in understanding the code properly. The code has been taken from [2]. Significant changes have been considering the limited resources at hand.

Given the nature of the dataset, we decide to go ahead with a ResNet18 model for training. In order to fairly compare the performance with the Supervised Learning method, we first train the ResNet in a supervised fashion using only 20% of the dataset. We then train the ResNet backbone attached to a projection head in a self-supervised fashion with 100% of the training dataset. We then extract the backbone and a simple linear head to it and then finetune it using the same 20% dataset used for supervised learning.

6. Future Work

Going forward, our primary objective is to delve deeper into the nuances of the DiNo approach and its applications. Specifically, we aspire to harness DiNo-based self-supervised learning techniques on the OCT Dataset for the intricate task of biomarker classification [7]. It's noteworthy to mention that Kokilepersaud et al. [6] previously employed contrastive learning as a self-supervised strategy on this identical dataset. However, our focal point will distinctly be on implementing and assessing the potential of the DiNo methodology in this context.

Apart from this, through this project, we aim to explore certain downstream tasks. One such innovative application we have thought of is to utilize the superior performance of DiNo in image segmentation and depth estimation with object removal and image filling [3] to create a new utility for DiNo in unwanted object removal. This is similar to the magic eraser feature present in the new Pixel phones. We aim to improve the performance of such a feature by utilizing DiNo.

References

[1] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari S. Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *ArXiv*, abs/2304.12210, 2023. 2

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 1, 3, 4

[3] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004. 5

[4] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. 4

[6] Kiran Kokilepersaud, Mohit Prabhushankar, and Ghassan AlRegib. Clinical contrastive learning for biomarker detection, 2022. 4

[7] Mohit Prabhushankar, Kiran Kokilepersaud, Yash-Yee Logan, Stephanie Trejo Corona, Ghassan AlRegib, and Charles Wykoff. Olives dataset: Ophthalmic labels for investigating visual eye semantics. 1, 4

[8] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 1

[9] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 4

[10] Ashish Vaswani et al. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1