Mid Term Project Report
Exploring the Efficacy of 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 eliminates 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 [1] on the DINO (Self Distillation with No Labels) algorithm. The promising outcomes exhibited by DINO [1] and its subsequent version, DINO V2 [2], are noteworthy. Significantly, models trained using DINO and 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 explore the capabilities of these self-supervised methods, contrasting them against traditional supervised techniques to comprehend their strengths and potential limitations. We aim to compare the differences between these self-supervised techniques on partially labeled datasets. One such dataset learning is the Optical Coherence Tomography (OCT) Images [3]. This dataset has approximately 78,000 B-scan images, of which a mere subset, roughly 9,400, are labeled. Taking into account the computational constraints, we decided to first test out this aproach on the smaller imagenette dataset from FastAI [4] [1]

The rest of this report is as follows: Section 2 briefly explains the architecture of the Vision Transformer [5] which has been extensively used by the DiNo [1] to produce competitive results. Section 3 provides an overview of some previously used self-supervised methods and Section 4 takes a deeper dive into Boot Stap Your Latent (BYOL) [6] which is the predecessor of DiNo. In Section 5, the DINO algorithm is discussed comprehensively.In section 6, we present results produced using the DiNo algorithm on the imagenette dataset. In section 7, we show some results on downstream tasks using the trained model provided by [1]. Finally, Section 8 discusses what we plan to do in the final report. Code used to produce the results shown in this report is made available her [2].

2. Vision Transformers

Vision Transformers (ViTs) [5] apply the transformer architecture, originally designed for Natural Language Processing tasks [7], 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 [7], 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.

$$z_0 = [x_{\text{class}}; x_1E; x_2E; \ldots; x_{N-1}E] + E_{\text{pos}} \quad (1)$$

Z_o is the input to the encoder

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \quad \ell = 1, \ldots, L \quad (2)$$

$$z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell, \quad \ell = 1, \ldots, L \quad (3)$$

$$y = \text{LN}(z^0_L) \quad (4)$$

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

[1] https://github.com/fastai/imagenette/

[2] https://github.com/AtharvRN/EE6380_Project

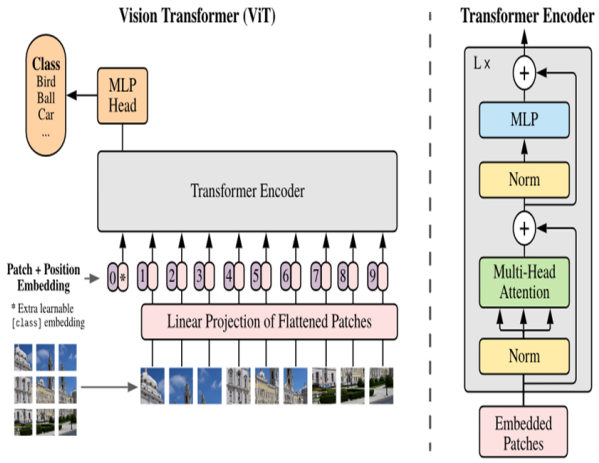

Figure 1. ViT Model Architecture

3. Self Supervised Learning

As stated earlier, self-supervised learning (SSL) [8] is an emerging field in machine learning that learns representations of data from unlabelled data, which gives significant advantage over the supervised learning approaches that require labelled data for training, which isn't available handful these days. In the field of computer vision, SSL has led to the development of various methods like Bootstrap Your Own Latent (BYOL) and DINO, that learn all-purpose features of images that help in performing various downstream tasks efficiently. While constructing these all purpose features, many methods have been developed, and can be categorized under four categories respectively – the deep metric learning family, the self-distillation family, the canonical correlation analysis, and the masked image modeling family. One such popular method **SimCLR** [9] 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. **VICReg** [10], 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 co-variance encourages different dimensions of the representation to capture different features. **MoCo** (Momentum Contrast) [11] uses a constrastive loss function, where the model is trained to maximize the similarity between positive pairs, i.e., different views of the same image and minimize the similarity between negative pairs, i.e., different images, where the negative pairs are sampled from a queue of negative examples, which is updated using a momentum encoder.

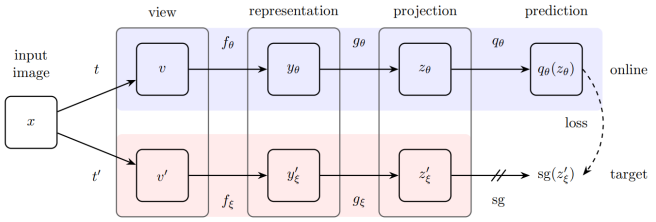

Figure 2: BYOL's architecture. BYOL minimizes a similarity loss between $q_\theta(z_\theta)$ and $\mathrm{sg}(z'_\xi)$, where θ are the trained weights, ξ are an exponential moving average of θ and sg means stop-gradient. At the end of training, everything but f_θ is discarded, and y_θ is used as the image representation.

Figure 2. BYOL Architecture

4. Bootstrap Your Own Latent

Unlike the state-of-art contrastive methods that are primarily built on reducing the distance between 'positive pairs' and increasing the distance between the 'negative pairs', a new self-supervised learning algorithm, called BYOL (Bootstrap Your Own Latent) introduced by Grill et al., [6] relies on two neural networks, referred to as online and target networks, that interact and learn from each other, to produce results comparable to the state-of-art performance on learning image representations, without using negative pairs. The online network is used to predict the target network representation of the same image under different augmented views. At the same time, the target network is updated with a slow-moving average of the online network. Using a combination of online network and target network helps the BYOL algorithm to avoid collapsing to trivial solutions, i.e., outputting the same vector for all images. Also, BYOL suffers much smaller performance drop than SimCLR, when only using random crops for image segmentation.

The authors claim that "The core motivation for BYOL: from a given representation, referred to as target, we can train a new potentially enhanced sequence of representation, referred to as online, by predicting the target representation. From there, we can expect to build a sequence of representations of increasing quality by iterating this procedure, using subsequent online networks as new targets for further pretraining." By using a slow-moving average over the outputs of the online networks as the target network, the representations are refined.

The online network is defined by a set of weights θ consisting of three stages: an encoder f_θ, a projector g_θ, and a predictor q_θ. The target network, whose parameters are ξ are an exponential moving average of the online parameters θ and are updated as follows, given a decay rate $\tau \in [0, 1]$,

$$\xi \leftarrow \tau\xi + (1 - \tau)\theta.$$

Given a set of images $\mathcal{D}$, an image $x \sim \mathcal{D}$ sampled uniformly from $\mathcal{D}$ and two distributions of image augmentations $\mathcal{T}$ and $\mathcal{T}'$, two augmented views of the image $v \triangleq t(x)$

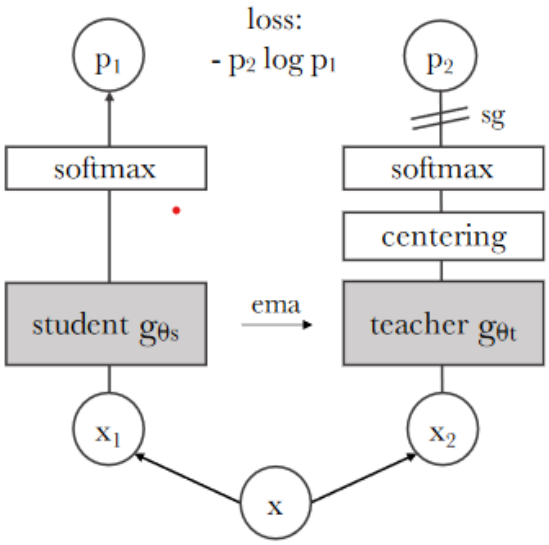

Figure 3. Self Distillation with No Labels

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 4. DiNo Algorithm

and $v' \triangleq t'(x)$ are produced from x by applying respective image augmentations $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$. From the first augmented view v, the online network outputs a representation $y_\theta \triangleq f_\theta(v)$ and a projection $z_\theta \triangleq g_\theta(y)$. From the second augmented view v', the target network outputs $y'_\xi \triangleq f_\xi(v')$ and the target projection $z'_\xi \triangleq g_\xi(y')$. We then output a prediction $q_\theta(z_\theta)$ of z'_ξ and l_2-normalize both $q_\theta(z_\theta)$ to $\overline{q_\theta}(z_\theta) \triangleq \frac{q_\theta(z_\theta)}{\|q_\theta(z_\theta)\|_2}$ and z'_ξ to $\overline{z'}_\xi \triangleq \frac{z'_\xi}{\|z'_\xi\|_2}$. Also, note that predictor is the only stage that is present in the online network and absent in the target network. Now, we calculate the mean squared error between normalized predictions and target projections,

$$\mathcal{L}_{\theta,\xi} \triangleq \|\overline{q_\theta}(z_\theta) - \overline{z'}_\xi\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi \rangle}{\|q_\theta(z_\theta)\|_2 \cdot \|z'_\xi\|_2}.$$

The loss function is then symmetrized by feeding v' to the online network and v to the target network to compute $\tilde{\mathcal{L}}_{\theta,\xi}$. Only θ is updated using stochastic optimization of minimization of $\mathcal{L}^{\text{BYOL}}_{\theta,\xi} = \mathcal{L}_{\theta,\xi} + \tilde{\mathcal{L}}_{\theta,\xi}$. So the parameter updation of online network and target network are as follows:

$$\theta \leftarrow \text{optimizer}(\theta, \nabla_\theta \mathcal{L}^{\text{BYOL}}_{\theta,\xi}, \eta),$$

$$\xi \leftarrow \tau\xi + (1-\tau)\theta.w$$

5. Self Distillation with No labels (DiNo)

In this section, we will go through the Dino (Self **Di**stillation with **No** Labels) Algorithm. The approach can be clearly understood from Figure 3. The pseudocode 4 explains the Algorithm in detail. Both these have been taken directly from [1]. 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 [12]. 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,

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

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 Algorithm 1 4, 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 4.

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 of 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 frozen. 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 [13]) and a projection head The projection head consists of a 3-layer multi-layer perceptron (MLP) with a hidden layer followed by a l_2 normalization and a weightnormalized fully connected layer [14] with K dimensions. Batch-normalization 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 a 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 5, [1] plot the entropy and KL

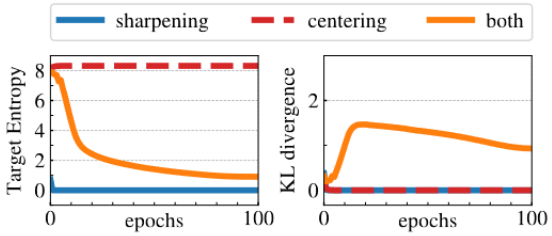

Figure 5. Analysis of Collapse

Model	Validation Set Accuracy (%)
Supervised (from scratch)	70.06
Supervised (pre-trained weights)	85.35
DiNo (fine-tuned)	82.17
DiNo (transfer learning)	82.29

Table 1. Validation set accuracy for different models.

throughout the training process, both with and without the 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.

6. 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 [4]. This is a smaller dataset containing 12,954 images of 10 classes with a 70/30 Train-Validation split. This dataset is ideal for running preliminary tests to help us understand the algorithm properly. The code has been taken from [15][3]. Significant changes have been made considering the limited resources at hand. Given the nature of the dataset, we decided to go ahead with a ResNet18 model for training. The DiNo hyperparameters used are listed as follows:

- total number of crops: 6 (4 local,2 global)
- momentum teacher (EWMA factor) = 0.995
- teacher temp = 0.04
- student temp = 0.1
- output dimension of the head (K) : 1024
- centre momentum (EWMA factor) 0.9 (fixed)
- DiNo MLP Head with 3 layers (1000,512,1024) with GeLU [16] non-linearities

[3] https://github.com/facebookresearch/dino

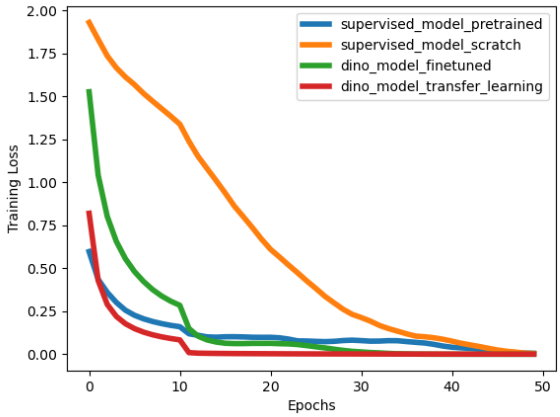

Figure 6. Training Loss (Moving Average)

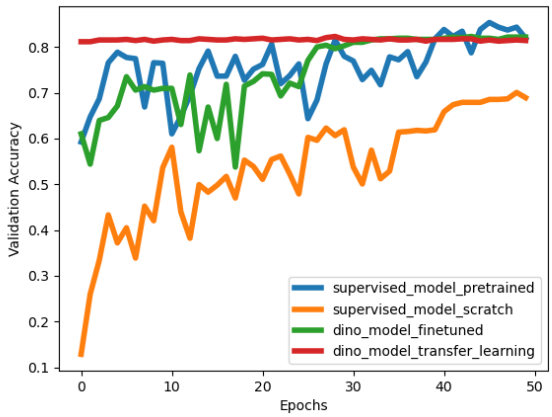

Figure 7. Validation Accuracy

We changed some of these hyperparameters due to computational constraints. Originally, [1] used 8 crops in total (2 global,6 local). The value of K used by them was 65,536. They also used more nodes higher for the middle layer in the DiNo head (4096).

Self Supervised Training using DiNo with no labels was done for 100 epochs using the Adam optimizer with a learning rate of 2.5×10^{-4}, momentum of 0.9 (default), and weight decay of 0.4. K-Nearest Neighbours Accuracy was used as a validation metric. With 100 epochs, the best model had a KNN accuracy of 78.8 %.

To fairly compare the performance DiNo with the Supervised Learning method, we first train the ResNet18 [13] by attaching a linear classifier to it in a supervised fashion using only 20% of the dataset. For the sake of comparison, we also fine-tune the ImageNet [17] pretrained ResNet18. For the DiNo model, we extract the backbone, add a simple linear classifier head to it, and train(both by finetuning and by freezing the backbone) using the same 20% dataset used for supervised learning. All supervised training was done for 50 epochs using the Adam- optimizer using a learning rate of 10^{-3}, momentum of 0.9, and weight decay of 10^{-4}. The Validation Scores for each technique used have been shown in Table 1. It can clearly seen that both the DiNo models outperform the supervised model by a significant margin (10%). The performance is, in fact, quite comparable with the model fine-tuned using ImageNet pretrained weights.

Figure 6 and 7 depict the training loss and the validation set accuracy, respectively. Interestingly, the frozen DiNo model performed better during the initial epochs than the supervised model fine-tuned using ImageNet pre-trained weights. However, over time, there is hardly any significant improvement. Eventually, the fine-tuning model catches up and achieves a similar score. Another interesting thing is despite the supervised model (trained from scratch) having a poor validation score; the training loss does converge to a decent extent. This could be because of the overfitting of the model on the small dataset (1900 images)

It's important to note that each model can probably be further optimized by training for more epochs and choosing more appropriate hyper-parameters. What we have presented here is some preliminary analysis of the DiNo self-supervised learning model

7. Downstream Tasks using DiNo

In this section, we will be discussing the downstream tasks that we tried to implement. The code used has been primarily obtained from [15]. The authors of [1] report that one can effectively generate segmentation masks from the attention maps. The segmentation masks are generated by retaining only a certain fraction of the highest attention values in an attention map, that is the values in the attention maps are sorted, and only a fraction of the attention values from the top are retained. As shown in 8, we can see that the segmentation masks generated from attention maps are quite accurate.

There are six heads in the attention modules and we get the overall attention map by taking the pixel-wise maximum across the attention maps to find out all the regions in the image, to which the transformer pays attention. We have displayed the overall attention maps for the same four images in 9.

We also used a vision transformer pre-trained using DiNo on ImageNet as a backbone and implemented a projection head on top of it to perform image classification. We freeze the weights of the backbone and fine-tuned the weights of the projection head(MLP). The authors of [1] report that they could achieve 99% accuracy while finetun-

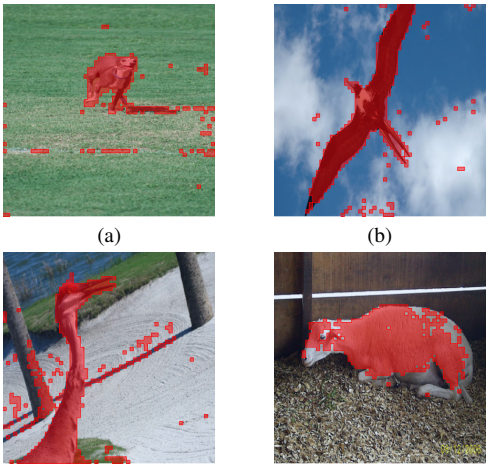

Figure 8. Segmentation Masks

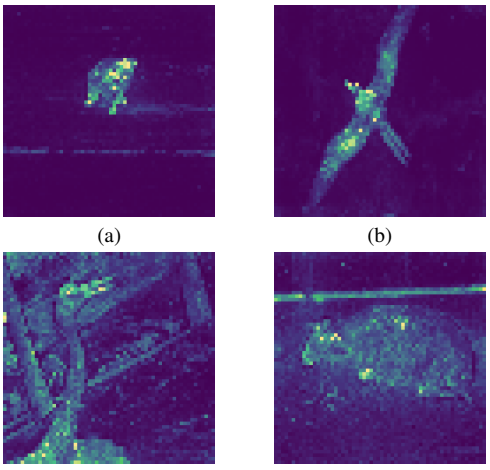

Figure 9. Overall Attention Map

ing on the CIFAR10 Dataset. However, when we tried to reproduce the result we could only achieve 93% accuracy. This could be due to different hyperparameters and a lower number of epochs.

8. Future Work

So far we have experimented with the DiNo by training a ResNet model with the DiNo approach on the Imagenette dataset. We would like to train ViTs with the DINO approach on the same dataset and see how these two architectures compare. Further, 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 [3]. It's noteworthy to mention that Kokilepersaud et al. [18] 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. Primarily we would like to quantify the accuracy of the segmentation masks that are generated from the attention maps. We will accomplish this by computing the Jaccard index between the segmentation mask and ground truth on the PASCAL VOC 12 Dataset. Since the current segmentation masks have some spurious noise, we would like to improve the accuracy of these segmentation masks by running the Minor Blob removal algorithm. Further, we would like to demonstrate how the different heads of the attention module focus on different regions of the image, by visualizing the different regions to which the attention module pays attention.

References

[1] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 1, 3, 4, 5

[2] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023. 1

[3] Mohit Prabhushankar, K. Kokilepersaud, Yash-Yee Logan, S. T. Corona, G. AlRegib, and C. Wykoff, "Olives dataset: Ophthalmic labels for investigating visual eye semantics." [Online]. Available: https://zenodo.org/record/7105232 1, 6

[4] J. Howard, "Imagewang." [Online]. Available: https://github.com/fastai/imagenette/ 1, 4

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

[6] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020. 1, 2

[7] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 1

[8] R. Balestriero, M. Ibrahim, V. Sobal, A. S. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, A. Schwarzschild, A. G. Wilson, J. Geiping, Q. Garrido, P. Fernandez, A. Bar, H. Pirsiavash, Y. LeCun, and M. Goldblum, "A cookbook of self-supervised learning,"

ArXiv, vol. abs/2304.12210, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258298825 2

[9] 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. 2

[10] A. Bardes, J. Ponce, and Y. LeCun, "Vicreg: Variance-invariance-covariance regularization for self-supervised learning," *arXiv preprint arXiv:2105.04906*, 2021. 2

[11] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738. 2

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531 3

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '16. IEEE, Jun. 2016, pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459 4, 5

[14] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/ed265bc903a5a097f61d3ec064d96d2e-Paper.pdf 4

[15] P. Bojanowski, "Dino github." [Online]. Available: https://github.com/facebookresearch/dino 4, 5

[16] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2023. 4

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. 5

[18] K. Kokilepersaud, M. Prabhushankar, and G. AlRegib, "Clinical contrastive learning for biomarker detection," 2022. 6