Personal Information

Name: Prajwal A
Email: prajwalanagani@gmail.com
University: PES University
Github: https://github.com/LawJarp-A
LinkedIN: https://www.linkedin.com/in/prajwal-anagani-a8401a191/
Location: Bengaluru, Karnataka, India
Timezone: IST - GMT+5:30

Project: Self-Supervised Learning for Strong Gravitational Lensing from the DEEPLENSE Project

Mentors

- Michael Toomey (Brown University)
- Pranath Reddy (BITS Pilani Hyderabad)
- Sergei Gleyzer (University of Alabama)
- Emanuele Usai (University of Alabama)
- Saranga Mahanta (Institut Polytechnique de Paris)
- Karthik Sachdev (RWTH Aachen)

Self-Supervised Learning Vision transformers for Strong Gravitational Lensing using EsViT: An efficient vision transformer based framework for self-supervised learning

Literature Survey

During the literature survey, I referred to several papers to prepare a plan of action for self-supervised learning with vision transformers. These papers include:

1. A Simple Framework for Contrastive Learning of Visual Representations: This paper introduces SimCLR, which is one of the earliest and best-performing architectures based on contrastive loss functions. SimCLR achieved better results than Pretext-Invariant Representation Learning (PIRL) by a good margin.
2. Self-Supervised Learning of Pretext-Invariant Representations: This paper proposes a new objective function for self-supervised learning, which is based on the concept of pretext tasks. This approach trains the model to predict some pre-defined pretext task that is not related to the final downstream task.

3. An Empirical Study of Training Self-Supervised Vision Transformers: This paper investigates several techniques for self-supervised learning with vision transformers, including contrastive loss, InfoNCE, and clustering. The authors also analyze the effect of different augmentation strategies on the performance of self-supervised learning.
4. Bootstrap Your Own Latent A New Approach to Self-Supervised Learning: This paper introduces BYOL, which is a new approach to self-supervised learning based on the idea of predicting a target network representation of an image under a different augmented view. The authors also propose a new way of computing the target network, which is more efficient than the traditional way.
5. Emerging Properties in Self-Supervised Vision Transformers: This paper introduces DINO, which is a self-supervised learning approach that uses vision transformers. The authors analyze the features that ViTs learn in self-supervised learning compared to supervised learning and show that SSL ViTs have a much richer semantic understanding of the images.
6. EFFICIENT SELF-SUPERVISED VISION TRANSFORMERS FOR REPRESENTATION LEARNING: This paper proposes an efficient self-supervised learning approach called EsViT, which achieves state-of-the-art results while maintaining the efficiency of the network. EsViT largely uses the same fundamental principles as DINO but is much more efficient to train and use, and beats the DINO scores on the benchmark ImageNet dataset.

Overall, the literature survey revealed that self-supervised learning with vision transformers has advanced significantly in recent years, with the emergence of several high-performing architectures. These architectures are based on various objective functions and techniques, and they have different strengths and weaknesses. The EsViT approach, which achieves state-of-the-art results with high efficiency, is a promising choice for strong gravitational lensing images.

Above is the image highlighting the difference between the self-attention maps with Supervised and Self-Supervised learning with ViT’s

Method	#Parameters ↓	Throughput ↑	Linear ↑	k-NN ↑
SoTA SSL methods with Big ConvNets				
SwAV, RN50w5 (Caron et al., 2020)	586	76	78.5	67.1
BYOL, RN200w2 (Grill et al., 2020)	250	123	79.6	73.9
SimCLR-v2, RN152w3+SK (Chen et al., 2020d)	794	46	79.8	73.1
Skyline methods with excessively long sequences for self-attentions				
DINO, DeiT-S/8 (Caron et al., 2021)	21	180	79.7	78.3
DINO, ViT-B/8 (Caron et al., 2021)	85	63	80.1	77.4
MoCo-v3, ViT-B-BN/7 (Chen et al., 2021)	85	∼63	79.5	-
MoCo-v3, ViT-L-BN/7 (Chen et al., 2021)	304	∼17	81.0	-
iGPT, iGPT-XL (Chen et al., 2020b)	6801	-	72.0	-
EsViT, Swin-T/$W=14$	28	660	78.7 (77.9)	77.0 (75.5)
EsViT, Swin-S/$W=14$	49	383	80.8 (79.4)	79.1 (77.3)
EsViT, Swin-B/$W=14$	87	254	**81.3** (80.5)	**79.3** (78.3)
Transformer-based SSL, with moderate sequence length for self-attentions				
Masked Patch Pred., ViT-B/16 (Dosovitskiy et al., 2021)	85	312	79.9†	-
DINO, DeiT-S/16 (Caron et al., 2021)	21	1007	77.0	74.5
DINO, ViT-B/16 (Caron et al., 2021)	85	312	78.2	76.1
MoCo-v3, ViT-B/16 (Chen et al., 2021)	85	312	76.7	-
MoCo-v3, ViT-H-BN/16 (Chen et al., 2021)	632	∼32	79.1	-
MoBY, Swin-T (Xie et al., 2021b)	28	808	75.1	-
EsViT, Swin-T	28	808	78.1 (77.0)	75.7 (74.2)
EsViT, Swin-S	49	467	79.5 (79.2)	77.7 (76.8)
EsViT, Swin-B	87	297	**80.4** (79.6)	**78.9** (77.7)

Table 1: Comparison with SoTA across different architectures on ImageNet linear probing. EsViT with $\mathcal{L}_L + \mathcal{L}_R$ is reported, while EsViT with only $\mathcal{L}_R$ is shown in parentheses. $W = 14$ is the window size, otherwise the default $W = 7$. ViT-BN is ViT that has BatchNorm (Frankle et al., 2020), and "$/P$" denotes a patch size of $P \times P$. "∼" indicates through-puts estimated by comparing different papers, detailed in Appendix. † The mask patch prediction in (Dosovitskiy et al., 2021) is pre-trained on JFT-300M and end-to-end fine-tuned in ImageNet, which we append as a reference.

Above is the image highlighting the `params` and `metrics` of discussed architectures.

As we can see that the EsViT is achieving SoTA results whilst maintaining the efficiency of the network (denoted by num of params)

EsViT for self supervised vision transformers for representation

After conducting an extensive literature survey, I have decided to implement the EsViT architecture (from EFFICIENT SELF-SUPERVISED VISION TRANSFORMERS FOR REPRESENTATION LEARNING) for self-supervised learning (SSL) on strong gravitational lensing images. The decision was based on several factors, including the fact that EsViT has demonstrated state-of-the-art performance compared to other ViT-based architectures.

One of the main advantages of EsViT over other SSL methods is **its efficiency**. While ViT-based architectures are generally considered large and computationally expensive, EsViT manages to achieve impressive results while reducing the number of parameters by half compared to DINO, ViT-B/16. Additionally, EsViT provides better throughput and performance metrics, as shown in Table 1 of the paper.

Our proposed tasks include

- Training the EsViT architecture in a self-supervised manner to learn robust representations of strong gravitational lensing images.

- Leverage the backbone weights obtained from the self-supervised training phase to fine-tune the model on baseline tasks, such as classification and regression.

By utilizing EsViT for SSL on strong gravitational lensing images, we hope to achieve higher performance and efficiency, and ultimately advance the field of computer vision in astrophysics.

Background on EsViT

- EsViT (Efficient Self-Supervised Vision Transformer) is an architecture proposed by Microsoft Research that builds upon the success of previous self-supervised learning (SSL) methods, particularly DINO, and aims to improve efficiency while maintaining or surpassing performance.
- EsViT uses a Vision Transformer (ViT) as its base architecture, which is a type of neural network that replaces the convolutional layers with attention layers, allowing the model to attend to different parts of the input image at different scales.
- EsViT adds a few novel components to the ViT architecture, such as a temporal average pooling layer and a stochastic depth layer, to improve training efficiency and stability.
- The temporal average pooling layer aggregates the feature maps over time, which allows the model to learn more robust representations by taking into account multiple views of the same image.
- The stochastic depth layer helps prevent overfitting by randomly dropping out layers during training.
- EsViT also uses a contrastive loss function and a momentum encoder, similar to DINO, to learn representations that are invariant to the augmentations applied to the input images.

Homework

In the pre-requisites tests, I used a simple SimCLR architecture (with ViT) with minimum augmentations and barely 75 epochs to learn representations, that reached an AUC score of 95% within 25 epochs on downstream tasks like classification.
This shows that the SSL approaches will definitely work great with our datasets, and the EsViT would produce greater results with the added benefits of exploring the self attention maps to produce explainability (why choose this class? and where is the model looking in the picture while choosing that class?)

Implementation details

Frameworks

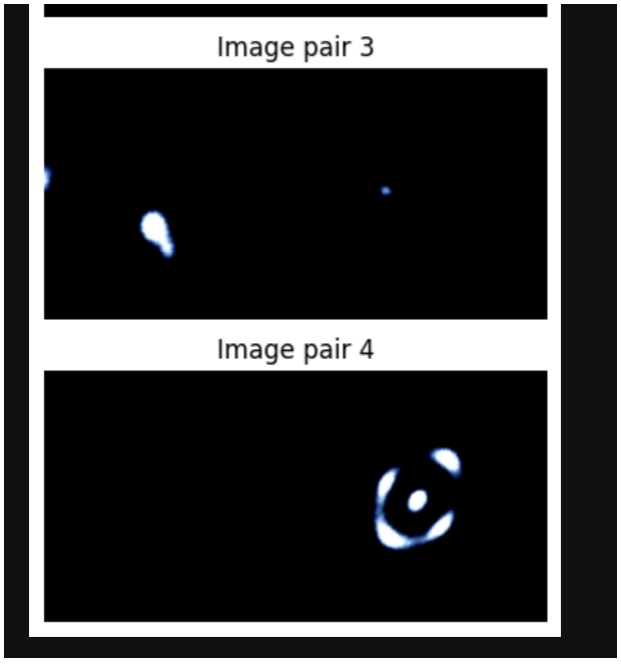
- PyTorch lightning: To enable quick use of efficient training methods like half precision, logging, gradient accumulation, etc quickly while maintaining a good control on the structure of code
    - Will be mainly used for SSL pre training
- Fastai: To enable quick training of pretrained weights (from SSL) onto downstream well structured tasks (classification, regression)
- WandB: Used to track the experiments and save the models.
    - Helps track every aspect of model training, ranging from loss per step to GPU utilization
    - Stores the entire tracking history of all the experiments run along with model weights

The use of these frameworks does not change the underlying fundamental code of the models,but helps facilitate quick experimentation and research.

Augmentations and preprocessing (Unique addition)

Similar to SimCLR, EsViT also uses augmented versions of image and encourages models to tell them as similar, though EsViT introduces a few more tasks.

One of the differences between mainstream datasets like ImageNet and Strong gravitational lensing is the negative space, the images have a lot of black space around them when not cropped, which results in pairs like pair 4



(Note that a left part is image 1, a augmented version of the original image, and the right part of each pair is image 2, yet another augmented version of the original image)

While creating positive pairs, we use random resize crop among other augmentation (color jitter, flipping, rotation, grayscale, etc) as it is said to work best (there is a study in SimCLR paper that outlines this)

- We create positive pairs by augmenting image twice
- Sometimes we get images like pair 3, which are good, as both the images have some part of the original image
- But sometimes we get images like Image pair 4, where one image is just the black background that conatins no infdormation about that particular structure
    - This is bad because the model will try to learn them as similar and it will confuse the model in long term leading to unstable training
    - This happens as there is a good amount of black space around the main structure, and while doing random resize crop, we sometimes crop that part and resize it

I propose to use strict pre processing techniques and augmentation techniques that take into account this issue.

How to tackle this

- While augmentation, we set a check to confirm if either of the augmented images are just black space, if so, redo the augmentation process again. This will ensure that no pairs will have useless black background.
- Use cropping to eliminate as much of the black background as possible while keeping the important parts of the image intact

EsViT

The Microsoft Research team has made it easier for us to implement EsViT by providing a PyTorch implementation of the architecture on their GitHub repository (https://github.com/microsoft/esvit). We can leverage the code snippets from the repository and use them to train our models on our own data.

Thanks to this pre-existing implementation, I can avoid starting from scratch, saving time and effort in the development process. While I will follow the architecture as outlined in the original paper, I plan to make changes to the preprocessing and augmentations to suit our specific needs.

This simplified approach to implementing EsViT will allow us to focus on tailoring the model to our data and applications, enabling us to achieve optimal results with minimal overhead.

Explainability with attention maps

Though overlooked, explainability is a important part of AI, we need to know why the model chose a patricular class as the end result.

In this regard, I plan to conduct a study to explore the self-attention maps of ViT (Vision Transformer) models after training. By analyzing the attention maps, we can gain insights into where the model is focusing its attention while making classifications.

My study will involve sampling random images and examining the corresponding attention maps to identify which regions of the image the model is attending to. Furthermore, to verify if the ViT is learning to attend to unique features or patterns, we will consult domain experts.

By understanding the attention patterns of ViT models, we can gain valuable insights into how they make decisions, which can be used to improve their performance and ensure their trustworthiness in critical applications.

Deliverables

- A self-supervised learning framework for strong gravitational lensing using EsViT
    - A novel augmentation technique for creating image pairs tailored for images from the strong gravitational lensing dataset
- Fine-tuned models of EsViT architecture on baseline tasks such as classification and regression.
- Conduct a study to examine the self-attention maps and the ViT's ability to learn the underlying structure.

Benefits to the community

- The proposed self-supervised learning framework would help in improving the performance of computer vision models on strong gravitational lensing images.
- The pre-trained models of EsViT architecture can be used for transfer learning and as a starting point for other computer vision tasks.
- The fine-tuned models of EsViT architecture on baseline tasks can be used in real-world applications of computer vision in astrophysics.
- The proposed approach is efficient and reduces the number of parameters by half compared to DINO, ViT-B/16, making it more accessible to researchers with limited computational resources.
- The proposed framework and models can contribute to advancing the field of computer vision in astrophysics and have potential applications in other fields such as medical imaging and satellite imagery analysis.

About me

I have a keen interest and expertise in the field of Deep Learning and Computer Vision. My proficiency in these domains is demonstrated through multiple projects and published papers. In 2022, I worked as the principal developer for a startup where I built a prototype for medical image analysis using deep

learning. This prototype was a game-changer for the startup, leading to its incubation by UC Berkeley Skydeck.

My unique approach to achieving great results with less data involved utilizing state-of-the-art classification models and contrastive learning with Siamese Networks. Additionally, I have a strong grasp of various PyTorch frameworks, which can significantly expedite the development process. Having worked with contrastive learning methods, such as Siamese Networks before, I am confident that I can deliver exceptional outcomes for this project.

Timeline

Willing to put in 3 hours for 5 days a week. From May 29th to August 28th (roughly 13 weeks). Total of 195 hours

May 4 - May 28: Community Bonding

- Set up communication channels with mentors and community
- Familiarize myself with domain knowledge
- Discuss project requirements and goals with mentors
- Plan project milestones and deliverables
- Discuss and start setting up the development environment

May 29 - June 11: Setting up the Development Environment and start of implementation

- Install the necessary software and dependencies
- Verify that the development environment is set up correctly
- Familiarize yourself with the dataset
- Plan on the preprocessing techniques and code out the augmentation modules to be used

June 12 - July 10: Implementing the SSL Model

- Use the code from the repository to build the SSL model
- Run the model on a small subset of data to verify
- Train the model on the full dataset
- Perform hyperparameter tuning to improve the model's performance
- Get the attention masks from the trained networks on random sample of images

July 11 - July 23: Training on Baseline Tasks

- Use the SSL model to train on a set of baseline tasks
- Test the model's performance on the baseline tasks
- Refine the model based on the results

July 24 - August 13: Gathering Results and Writing Paper

- Collect and analyze the results of the model on the baseline tasks
- Write a paper summarizing the project, methodology, and results
- Document the code and ensure it is easy to understand and use

August 14 - August 20: Final Touches and Submission

- Make final tweaks and improvements to the code and paper
- Review the code and documentation to ensure it is complete
- Submit the project and paper to the mentor for evaluation

August 21 - August 28: Final Working Week

- Respond to feedback from the mentor and make any necessary changes to the project or paper
- Wrap up the project and prepare for the final evaluation.

For each task, timeline is given while taking a buffer period into account (2 days for each tasks)