Soham Dasgupta
soham_d@me.iitr.ac.in
Indian Institute of Technology Roorkee
Roorkee - 247557

Transformer for Dark Matter Morphology with Strong Gravitational Lensing

Personal Information

Name: Soham Dasgupta

Homepage: astonishingwolf.github.io

Email: soham_d@me.iitr.ac.in

Resume: https://drive.google.com/file/d/1fRt5aCca7peTUEeKd47GWjcpc0QYYYzM/view

Github: https://github.com/astonishingwolf

Website: https://astonishingwolf.github.io/

University: Indian Institute of Technology Roorkee

Expected Date of Graduation: July 2023

Project Abstract

Description:

Strong gravitational lensing is a promising probe of the substructure of dark matter to understand its underlying nature better. Deep learning methods have the potential to accurately identify images containing substructure and differentiate WIMP particle dark matter from other well-motivated models, including vortex substructure of dark matter condensates and superfluids.

This project will focus on further developing the DeepLense pipeline that combines state-of-the-art deep learning models with strong lensing simulations based on lenstronomy. This project focuses on using transformers (e.g. vision transformers) to augment the performance of DeepLense algorithms (e.g. classification and regression).

Our job would be to deploy current SOTA transformers models on these datasets to augment their accuracy for our datasets.

Expected Outcomes:

- Research different vision transformers available and perform benchmarking on them to determine the most appropriate one for our test cases.
- Implement and test different vision transformers on provided datasets. This might include SOTA
- Improving accuracy by trying different regularisation and data augmentation techniques.
- Necessary patches, if any, for deployment in the DEEP Lense pipeline.
- Sample examples to demonstrate the use of provided models.
- Properly documenting the code flow for further PRs and updates.

About Me [Github][Website]

Hi, I am a final-year undergraduate student in the department of Mechanical Engineering. I am interested in Computer Vision, Deep Learning and Robotics. I have completed a recent course on Computer Vision where we had extensive modelling experience with SOTA CNN and Transformer models. I have also implemented many SOTA image classification models such as VGG, ResNet, ViT, PVT, etc. That left me utterly amazed by the fast past nature of the field, and I have been looking to build my mark in that field since
I have a solid academic background with a CGPA of 8.97(top 5% of the class). I also have experience handling lidar-camera data acquisition and conducting various experiments. I have experience with clustering algorithms, state estimations, object detection, tracking and point aggregations. I worked previously with AVRL(Autonomous Vehicle Research Lab) before this during my summer internship. I am also continuing that work with them as my Bachelor's Thesis. Before this, I worked with Machine Intelligence Lab, IIT Roorkee, in controls and reinforcement learning. I was also part of the student club Team Robocon which participates in different Rover-based challenges around the globe.

PREVIOUS WORK

INDOOR PERCEPTION

Indoor perception received lots of attention in recent years. Since the establishment of Point Cloud processing using PointNet, interest surrounding point cloud detection and tracking has received a lot of traction.

We developed an end-to-end pipeline for Pose estimation and localisation of objects in a crowded and highly occluded indoor environment. We used a sensor set consisting of a Lidar and a camera. We have leveraged data from both sensors and have performed sensor fusion to fuse their data. We performed object registration from multi-frame using lidar and then successfully used RANSAC to create feature vectors. Using the feature vector, we successfully used Generalised ICP to estimate the robot's pose. This work has been done in collaboration with AVRL, University of Waterloo.

SOFT TERRAIN BIPEDAL CONTROL

In the current research field surrounding bipedal dynamics, there has been a lack of research on bipedal robots traversing in soft terrain. The reason is the need for a proper simulator for deformable ground due to high computational cost and slow processing speed.

Using multibody dynamics, we developed a simulator in PyChrono for a 9 DoF Biped traversing in soft terrain conditions. We also implemented a Reinforcement learning-based control for a biped robot travelling in deformable terrain conditions. We Initialized the gait trajectory as cycloidal and subsequently used DDPG to train the biped for gait and trajectory control. This work has been done in collaboration with Machine Intelligence Lab IITR and funded by CSIR and Nidhi-Vyas Research.

Control of ARM Manipulator in ERC 2021

In my sophomore year, I participated in the annual European Rover Challenge Online Edition 2021. As a part of the challenge, we had to control an arm manipulator over a series of tasks, including detecting Aruco tags and object pick and place.
The whole module had to be completely autonomous and required detecting Auroco tags from the environment and planning the path. We extensively used Open Source packages, including OpenCV for tag detection and MoveIT for path planning.

Required Skills and Expertise

Skill	Experience Level(Out of 5)	Expertise
Python	5	3 years working in various projects and internship
Transformers	3	Experience over few course projects over a year
Deep Learning	3	Few Applied AI courses and course projects
Computer Vision	4	2 Courses and 1 year Bachelor Thesis extensively based on computer vision

Research Work [paper1][paper2][paper3]

Background

A gravitational lens can occur when a massive amount of matter, like a cluster of galaxies, creates a gravitational field that distorts and magnifies the light from distant galaxies that are behind it but in the same line of sight. The effect is like looking through a giant magnifying glass. It allows researchers to study the details of early galaxies too far away to be seen with current technology and telescopes. Strong gravitational lensing has shown great promise in the decoding substructure of dark matter halos. The strong lensing has already proven to be a powerful probe of dark matter substructure hence it is quite logical to extend the same distinguish between different types of dark matter substructure. The deep learning method has the potential to accurately identify images containing substructure and differentiate WIMP dark matter from other well-motivated models, including vortex substructure of dark matter condensates and superfluids.

Dataset

Currently, strong lensing data is limited to a handful of images. However, the upcoming completion of the Large Synoptic Survey Telescope (LSST) will lead to thousands of lensing solid images that can be analyzed. Hopefully, in this project, we will be working with simulated lensing images using the package PyAutoLens.

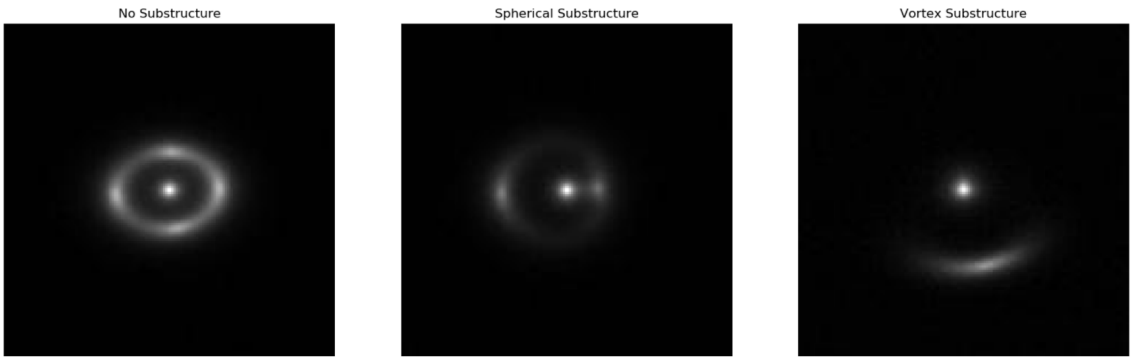


Fig: Example of simulated images for all three classes

Different types of substructure in existence

- **No Substructure**: Gravitational Lensing Images simulated with no substructure
- **Spherical Substructure**: Gravitational Lensing Images simulated with Cold dark matter
- **Vortex Substructure**: Gravitational Lensing Images simulated with the superfluid dark matter

From these images one can appreciate the difference in lensing is primarily in the morphology of the signal, making this an ideal task for classification with a vision transformer.

Previous Works

Previously, the work focused on implementing CNN models to classify different substructures as described in the paper.. Using the ResNet-50 architecture, they got an AUC score of 0.998,0.985 and 0.968 for no substructure, spherical sub-halos and vortices respectively.

As described in the repo, other work has been done in the Equivaritant Neural Network. They managed to get good accuracy while exploiting different symmetries in the datasets. The summary of the result is described below.

	C4	C8	D4	D8	ResNet-CNN
No substructure	0.9986	0.9991	0.9994	0.9988	0.9969
Sphere	0.9942	0.9976	0.9969	0.9964	0.9845
Vortex	0.9986	0.9995	0.9994	0.9991	0.9974

Finally there has been some work done in case of the transformer-based architecture for classifying the substructure as mentioned in this repository [1],[2]. They used three different models of data modelled with different methods.

The models they used:

Model_I

- Images are 150 x 150 pixels
- Modeled with a Gaussian point spread function
- Added background and noise for SNR of around 25

Model_II

- Images are 64 x 64 pixels
- Modeled after Euclid observation characteristics as done by default in lenstronomy
- Modeled with simple Sersic light profile

Model_III

- Images are 64 x 64 pixels
- Modeled after HST observation characteristics as done by default in lenstronomy.
- Modeled with a simple Sersic light profile,

They achieved decent accuracy over a few state-of-the-art methods. A summary of the results is detailed below.

	Model 1			**Model 2**			**Model 3**		
	vortex	**Cdm**	**NoSub**	**vortex**	**Cdm**	**NoSub**	**vortex**	**Cdm**	**NoSub**
Classifcation @ Archil Srivastava									
EfficientNet	0.9858	0.9811	0.9956	1	1	1	1	1	1
ViT	0.9772	0.9693	0.9988	-	-	-	1	1	1
ConViT	0.9162	0.8832	0.9694	0.9998	0.9991	0.9998	1	0.9999	1
CrossViT`	0.9326	0.8802	0.9871	1	0.9999	1	1	1	1
BottleNeck Transformer	0.9872	0.9807	0.9993	1	1	1	1	1	1
Efficient Former	0.9949	0.9899	0.9995	1	1	1	1	1	1

CoaT	0.9878	0.9804	0.9996	1	1	1	1	1	1
CoatNet	0.9952	0.9917	0.9996	1	1	1	1	1	1
Swin	0.9621	0.9420	0.9983	1	1	1	1	1	1
Classification @ Karthik Sachdeva									
CvT	0.9714	0.9657	0.9991	0.9987	0.9990	1.0000	0.9986	0.9992	1.0000
CrossFormer	0.9508	0.9511	0.9996	0.9856	0.6041	0.9998	0.9782	0.9096	0.9998
LeViT	0.9720	0.8612	0.9997	0.9936	0.9800	1.0000	0.9981	0.9973	1.0000
TwinsSVT	0.6678	0.9462	0.9988	0.9566	0.8216	0.9061	0.9999	0.9998	1.0000
CCT	0.9632	0.9462	0.9988	0.9816	0.9386	0.9959	0.5192	0.5106	0.5419
CrossViT	0.6726	0.5640	0.7497	0.9318	0.6548	0.7833	0.5160	0.4986	0.5195
T2TViT	0.9706	0.6502	0.9902	0.9883	0.9095	0.9983	0.6515	0.9737	0.8980
PiT	0.5981	0.5584	0.6580	0.5351	0.5022	0.5378	0.5176	0.5153	0.5294
Swim	0.5066	0.4886	0.4945	0.5056	0.4922	0.5022	0.4357	0.6116	0.5909

The Key takeaway from the above experiment is that our transformer models are still falling behind the Equivariant NN and CNN models developed in the past. One key reason could be that since we are training the models, the models might be underfitting. One of the possible solutions is to use instead pre-trained weights trained on natural images like ImageNet and then fine-tune our datasets. These methods have provided researchers with sufficient accuracy when used in medical images. Further, the latter models, like PiT and Swim, haven't learnt any features since the model's accuracy is close to 33.33% (the random classification). Hence we can also look into the shortcomings of implementing the above models.

Vision Transformer[paper]

Deep neural networks have become the fundamental mental infrastructure in today's artificial intelligence systems. Convolutional neural networks introduce convolutional and pooling layers and have massively succeeded in image classification problems. The DeepLense pipeline already contains existing implementations of Equivariant Neural Networks and ResNets. To add to the pipeline in this project, we would be extending the pipeline by introducing

transformer-based models. Transformer is a new type of neural network; it mainly utilises a self-attention mechanism to extract intrinsic information and has shown great potential for its extensive uses in AI. Ever since the advent of ViT, which applied a pure transformer directly to the sequence of image patches to classify the full image. In addition, image classification model transformers have been widely used in object detection, segmentation and video understanding.

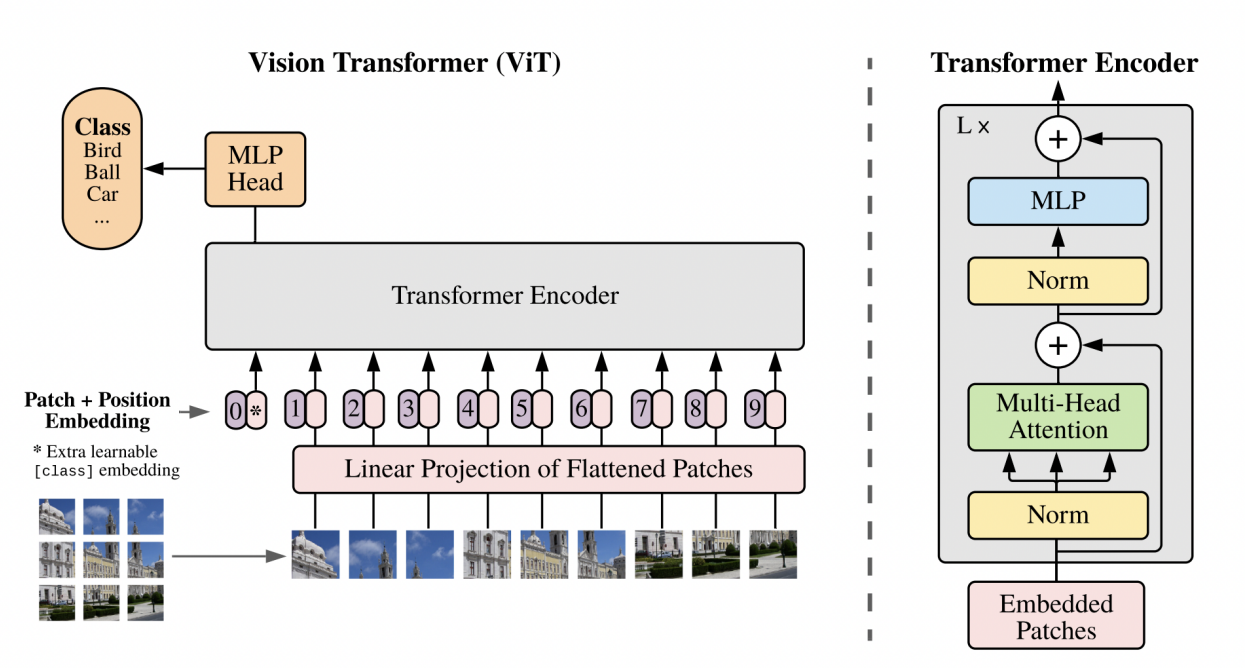
Key Concept of Transformer

The Transformer Encoder consists of

- Multi-Head Self Attention Layer(MSP) to concatenate the multiple attention outputs linearly to expected dimensions. The multiple attention heads help learn local and global dependencies in the image.
- Multi-Layer Perceptrons(MLP) contain two-layer with Gaussian Error Linear Unit(GELU)
- Layer Norm(LN) is applied before every block as it does not introduce any new dependencies between the training images. Help improve the training time and generalization performance.
- Residual connections are applied after every block, allowing the gradients to flow through the network directly without passing through non-linear activations.

An image classification head is implemented using MLP with one hidden layer at a pre-training time and a single linear layer for fine-tuning. The higher layers of ViT learn the global features, whereas the lower layers learn global and local features. This allows ViT to learn more generic patterns.

Image is first split into fixed-size patches: The 2D image of size H *W is split into N patches where $N=H*W/P^2$ .Flatten the 2D patches to 1D patch embedding and linearly embed them Each patch is flattened into a 1D patch embedding by concatenating all pixel channels in a patch and then linearly projecting it to the desired input dimension. Position embeddings are added to the patch embeddings to retain positional information. Adding the learnable position embeddings to each patch will allow the model to learn about the structure of the image.



For our use case, since the datasets we are working with are quite small, we would use a smaller variant of the following models, preferably keeping the variable count under 50M. This also ensures that training and testing the models are computationally and not too expensive.

Assignment Model(Insights from the attention maps)

Task 5 of our assignment required us to deploy a vision transformer that can classify images between the strong lensing images with and without substructure. The data we will be working with could be similar to the one we have here.

In the assignment, we used vision transformer ViT_16 with batch normalization. We received an AUC score of 0.966. However, the score could have been improved substantially if we had used augmentation techniques along with some kind of regularisation with dropout.

While drawing the attention map, it gives us a glimpse into how the models are training. We can observe from the illustrations below only during the encoder layers 10 and 11 the model was

focus on the geometric representation of the substructure while the earlier layers failed to extract any sufficient information. The final layer focuses heavily on the actual lensing image.

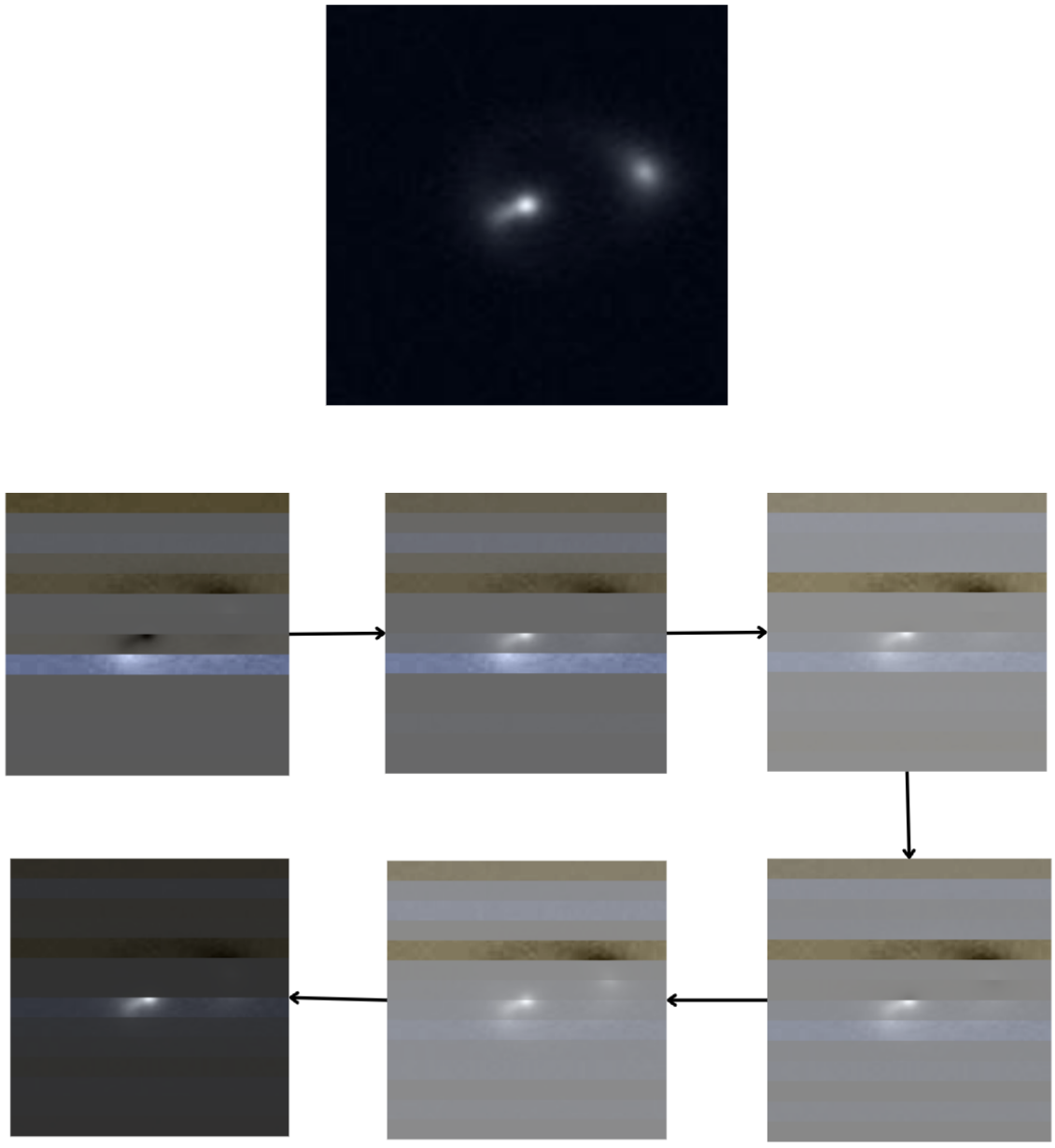


FIg: Attention maps of a VIT_16. (Top) The actual image. (middle to down) The progression of attention maps from encoder layer 6 to 11. Showing us a glimpse of the model's focus

In the common task of this project, we received similar datasets as what we are supposed to work with. In the datasets, we had a single channel and were required to duplicate it to convert it to three channels. Further, we were required to resize the image size from 150X150 to the default size for the network of 224X224. Finally, I used ResNet-18, considering its small size and computational efficiency. I received a good enough accuracy of 80%, considering the model is only trained for six epochs while the loss is still decreasing.

Finally, we received this AUC Score.

vortex	Cdm	NoSub
0.91255	0.8857	0.9522

The models were further pre-trained on ImageNet, suggesting convergence for pre-trained models. Further, the real challenge of this project is to get a comparable accuracy for the transformer with respect to its CNN counterpart.

Models:

For the transformer-based models, we propose the following 17 models. Some of the models have already been used in previous projects, although we can improve the accuracy. Finally, for our work during the initial weeks, after extensive testing, we will shorten the list into eight models we plan to implement. I have also included recent models for the recent CVPR 2022 conference. Following are the models I propose.

- **ViT Vision Transformer: [paper] [code][Apache-2.0]**
  It is a pure transformer model directly applied to a sequence of image patches.ViT is pre-trained on large datasets and then fine tunes for downstream tasks on smaller datasets. ViT is pre-trained on large datasets and then fine-tuned for downstream tasks with smaller datasets.

- **TNT: Transformer in Transformer: [paper] [code]**
  Transformer iN Transformer (TNT). represents the local patches (e.g., 16×16) as “visual sentences” and presents to further divide them into smaller patches (e.g., 4×4) as “visual words”. The attention of each word will be calculated with other words in the given visual sentence with negligible computational costs. The inner transformer block models the

relationship between sub-patches while the outer transformer block models the patch-level relationship

- **Swin Transformer:[paper] [code][MiT-Liscene]**
  The paper proposes a hierarchical Transformer whose representation is computed with shifted window. The shifted windowing scheme increases efficiency by limiting self-attention computation to non-overlapping local windows while allowing a cross-window connection. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity for image size.

- **RegionViT: [paper] [code][Apache-2.0]** The architecture adopts the pyramid structure and employs novel regional-to-local attention rather than global self-attention in vision transformers. More specifically, the model first generates regional and local tokens from images with different patch sizes. Each regional token is associated with a set of local tokens based on the spatial location. The regional-to-local attention includes two steps: first, the regional self-attention extracts global information among all regional tokens. Then, the local self-attention exchanges the information among one regional token and the associated local tokens via self-attention.

- **T2TViT:[paper] [code]**
  It stands for Tokens-To-Token Vision Transformer (T2T-ViT). It incorporates - a layer-wise Tokens-to-Token (T2T) transformation to progressively structurize the image to tokens by recursively aggregating neighbouring Tokens into one Token (Tokens-to-Token), such that local structure represented by surrounding tokens can be modelled and tokens length can be reduced. It also contains an efficient backbone with a deep-narrow structure for the vision transformer motivated by CNN architecture design.

- **PVT : [paper] [code][Apache-2.0]**
  PVT can be not only trained on dense partitions of the image to achieve high output resolution, which is important for dense predictions but also using a progressive shrinking pyramid to reduce computations of large feature maps.

- **CPVT : [paper] [code]** CPVT propose a conditional positional encoding (CPE) scheme for vision Transformers. Unlike

previous fixed or learnable positional encodings, which are pre-defined and independent of input tokens, CPE is dynamically generated and conditioned on the local neighbourhood of the input tokens.CPE can easily generalize to the input sequences longer than the model has seen during training. Besides, CPE can keep the desired translation invariance in the image classification task, resulting in improved performance. They implemented CPE with a simple Position Encoding Generator (PEG) to get seamlessly incorporated into the current Transformer framework.

- **CvT: [paper] [code][MIT-Liscene]**
  Convolutional vision Transformer (CvT) improves upon Vision Transformer (ViT) in performance and efficiency by introducing convolutions into ViT to yield the best of both designs. This is accomplished through two primary modifications: a hierarchy of Transformers containing a new convolutional token embedding and a convolutional Transformer block leveraging a convolutional projection. These changes introduce desirable convolutional neural networks (CNNs) properties to the ViT architecture while maintaining the merits of Transformers.

- **CrossFormer: [paper] [code][MIT-Liscene]**
  They propose Cross-scale Embedding Layer (CEL) and Long Short Distance Attention.. On the one hand, CEL blends each embedding with multiple patches of different scales, providing the self-attention module with cross-scale features. On the other hand, LSDA splits the self-attention module into a short-distance one and a long-distance counterpart, reducing the computational burden and keeping both small-scale and large-scale features in the embeddings. Through the above two designs, we achieve cross-scale attention. Besides, we put forward a dynamic position bias for vision transformers to make the popular relative position bias apply to variable-sized images. Hinging on the cross-scale attention module, we construct a versatile vision architecture, dubbed CrossFormer, which accommodates variable-sized inputs.

- **CaiT:[paper] [code][Apache-2.0]**
  In Class Attention in Image Transformation, they have built and optimised deeper transformer networks for image classification. In particular, they have investigated the interplay of architecture and optimization of such dedicated transformers.

- **CrossViT:[paper] [code][Apache-2.0]**
  They propose a dual-branch transformer that combines image patches (i.e., tokens in a transformer) of different sizes to produce stronger image features. The approach processes small-patch and large-patch tokens with two branches of different computational complexity. These tokens are then fused purely by attention multiple times to complement each other. They also developed a simple yet effective token fusion module based on cross-attention. It uses a single token for each branch as a query to exchange information with other branches.

- **CCT:[paper] [code][Apache-2.0]** They present an approach for small-scale learning by introducing Compact Transformers.They show with the right size, and convolutional tokenization, transformers can avoid overfitting and outperform state-of-the-art CNNs on small datasets. The models are flexible in size, and can have as little as 0.28M parameters while achieving competitive results.

- **TwinsSVT:[paper] [code][Apache-2.0]**
  They propose two vision transformer architectures, namely, Twins-PCPVT and Twins-SVT. The proposed architectures are highly-efficient only involving matrix multiplications that are highly optimized in modern deep learning frameworks. More importantly, the proposed architectures achieve excellent performance on a wide range of visual tasks, including image level classification as well as dense detection and segmentation.

- **A-ViT:[paper] [code][Apache-2.0]**
  A-ViT proposes a method that adaptively adjusts the inference cost of vision transformer ViT for images of different complexity. A-ViT achieves this by automatically reducing the number of tokens in vision transformers that are processed in the network as inference proceeds. They reformulate Adaptive Computation Time (ACT) for this task, extending halting to discard redundant spatial tokens. The appealing architectural properties of vision transformers enables our adaptive token reduction mechanism to speed up inference without modifying the network architecture or inference hardware. A-ViT requires no extra parameters or sub-network for halting, as we base the learning of

adaptive halting on the original network parameters. They further introduce distributional prior regularization that stabilizes training compared to prior ACT approaches.

- **RVT :[paper] [code]**
  They propose Robust Vision Transformer (RVT), a new vision transformer with superior performance and strong robustness. Inspired by the evaluation findings, we propose two new plug-and-play techniques called position-aware attention scaling and patch-wise augmentation to augment our RVT, which we abbreviate as RVT*.The work is based on a systematic evaluation of components of ViTs in terms of their impact on robustness to adversarial examples, common corruptions and distribution shifts.

- **CSWin Transformer:[paper] [code][MIT-Liscene]**
  A challenging issue in Transformer design is that global self-attention is very expensive to compute, whereas local self-attention often limits the field of interactions of each token. They developed the Cross-Shaped Window self-attention mechanism for computing self-attention in the horizontal and vertical stripes in parallel that form a cross-shaped window, with each stripe obtained by splitting the input feature into stripes of equal width. They also introduce Locally-enhanced Positional Encoding (LePE), which handles the local positional information better than existing encoding schemes. LePE naturally supports arbitrary input resolutions and is thus especially effective and friendly for downstream tasks. CSWin Transformer demonstrates competitive performance on common vision tasks with these designs and a hierarchical structure.

- **MobileFormer:[paper] [code][MIT-Liscene]**
  They present Mobile-Former, a parallel design of MobileNet and transformer with a two-way bridge in between. This structure leverages the advantages of MobileNet at local processing and transformer at global interaction. And the bridge enables the bidirectional fusion of local and global features. Unlike recent works on vision transformers, the transformer in Mobile-Former contains very few tokens (e.g. six or fewer tokens) that are randomly initialized to learn global priors, resulting in low computational cost.

ImageNet1k Benchmark

MODELS	PARAMS(M)	TOP-1%
ResNet-50	25.6	79.1
ViT-Base	86	85.3
TNT-Small	23.8	81.5
TNT-Base	65.6	82.9
Swin-Small	50	83.0
Swin-Base	88	83.3
RegionViT-Medium	41.2	83.1
RegionViT-Base	72.7	83.2
T2T-ViT-9	39	81.9
T2T-ViT-23	64	82.3
PVT-Medium	44.2	81.2
PVT-Large	61.4	81.7
CVPT-Small	23	81.5
CVPT-Base	88	82.3
CvT-13	20	83
CvT-21	32	83.3
CrossFormer-Small	30	82.4
CrossFormer-Base	52	83.5
CaiT-Base	68	83.2
CrossViT	43.2	82.5
CCT	22.36	80.67

Twin-SVT-Small	24	81.7
Twin-SVT-Base	56	83.2
Ada-ViT	22	80.7
RVT-Small	23.3	81.9
RVT-Base	86.2	82.5
CSWin-Tiny	23	82.8
CSWin-Small	35	83.6
CSWin-Base	78	84.2
MobileFormer-508	14.0	79.3

As evident from the above table most the models selected have parameters under 100M. Further, for our case we would be starting with smaller models are improving them to get higher accuracy. I mentioned by Karthik in his project the models were kept in the range of 4 to 7 M considering limited training capability. Further, we will be preferring models with both CNN and transformer architecture as they seem to perform better and are able to extract both local and global features.

Transfer Learning [paper]

The vision transformers, like their NLP counterparts, are characterised by very large models (tens of millions of parameters), and as a result of this model size, a prohibitive need for data and computational resources for efficient training. Therefore, training a vision transformer from scratch, even when the architecture of the network is fixed, and no architecture search is required, is essentially out of the question for most use cases and users. In our case, the dataset is small compared to the astronomical size of ImageNet 21k that normally these transformers are trained on.

Hence, Instead of fully (re-)training a vision transformer, one normally takes a pre-trained network and either use it for feature extraction or only fine-tunes the last layers based on the classes and labels of the problem at hand. When a Vision transformer such as ViT is trained on very large datasets and then fine-tuned on smaller datasets results show they generally outperform the Conventional CNN architectures in most of the computer vision tasks.

As we mentioned above, vision transformers are generally larger architectures than CNNs and are hence both slower to run and more demanding in storage at the same time the models we will be using would be trained on the ImageNet dataset which in fact according to the above paper does generalise really well when we are dealing with natural images., however since the datasets which we have is considerably differernt., it would be interesting to note how well the model is able to generalise in regards to our datasets.

License and Ownership

All the above models mentioned have compatible licenses suitable for our use case. Most of the models have either Apache2.0 or MIT Liscene, making us easier for us to implement and modify their work for our use case. Other than that, RVT and CVPT have no generic license and require us to take their permission for implementation. Hopefully, we will be able to use all the models.

Implementation Details

Regularisation and Data Augmentation

Considering we have a smaller dataset and fine-tuning it on that we can regularize our models by optimising it on basic regularization parameters for example weight decay, dropout and label smoothing. Based on the paper which shows crucial changes for us to make for fine-tuning our model for our particular datasets after pretraining it on ImageNet1K

Data augmentation is one of the most crucial steps. Data augmentation is highly dependent on the dataset we are working with. Considering these some of the common examples of resizing, flipping both horizontally and vertically would make sense on augmentation technique involving changing colours or perspective might not be best. As stated in this blog many augmentation techniques didn't yield good results. This area requires more research which should be done during the initial week of the project.

Deployment

The file structure is very important for any kind of project. It ensures future users can easily pick up the work from there with little to no hassle. Developing on this fact the file structure can be designed as follows

```
Model_CONFIG = {
    "network_type": "Model",
    "pretrained": True,
    "image_size": 224,
    "batch_size": 64,
    "num_epochs": 15,
    "optimizer_config": {
        "name": "AdamW",
        "weight_decay": 0.00-1,
        "lr": 0.001,
    },
    "lr_schedule_config": {
        "use_lr_schedule": True,
        "step_lr": {"gamma": 0.5, "step_size":
20,}},
    "channels": 1,
}
```

- **Configs:** Since configuration files are changed more frequently. It is better to keep them in a separate folder to make changes more easily. Configuration files can include lots of parameters related to the optimizer, training setting and obviously the model transformer parameters. An example model for a model is given below.
- **Models:** This would contain all the models and their implementation. Most of the implementation would be easily imported and modified from their actual repository as mentioned above.
- **Results:** This would contain all the confusion matrix and the ROC curve for comparisons.
- **Utlis:** This would contain all the codes that might be necessary for implementation including data augmentation script, training and testing scrips.
- **Miscellaneous:** Further a docker image can be created for easy deployment. Other than that I am planning to add a sample file with argument parsers to train and deploy any model with a dataset.

Timeline

Community Bonding Period

Week 1 - 3 | May 4 - May 28

I would spend the community bonding period to better know my mentors and the community. During this time, I will also focus on setting up the hardware for my project., I would set up the project by taking help from the mentors and other community members. During this period, I would be using the DEEPLENSE library extensively. Implementing existing classification algorithms that are present. Further, I would discuss my project extensively with my mentors to get a gist of the code flow. Throughout my work, I am planning to maintain a weekly log to track my progress along with extensive documentation for ease of access for future reference.

Coding Period

Week 4 | May 29 - June 4

This week I will be testing out different models in the ImageNet dataset implementing their existing repository..This week I will be also shortlisting the number of models to implement based on the performance and ease of access. Overall throughout this project, I plan to implement 8 different models.

Week 5, 6, 7, 8 | June 5 - July 9

Continuing the development from the previous week, these subsequent weeks I will be working on four different models. The four different models will be deployed and tested on the dataset provided by the organization. Further, during the trial week s we can conduct rigorous testing to figure out what regularisation and data augmentation techniques to use.Most of the parameters could be kept constant throughout the development process of all models.

Phase 1 Evaluation: July 14

Week 10, 11, 12, 13 | July 21 - August 10

Continuing the development from the previous week, these subsequent weeks I will be working on the remaining four models. The four different models will be deployed and tested on the dataset provided by the organization. Further, during these, I would clean up and organise my code base similar to the sequence mentioned above.

Week 14 | August 11 - August 17

The final week will be focused on the deployment of the final models and finishing up the documentation. Moreover, for ease of access for future users, the final models are packaged into a docker image.

Availability

I am graduating this year in the first week of May. I will be back at my home for the vacation .For the summer I have no commitments or vacations planned, and I would mostly be in my hometown. I would be working from Monday to Friday Morning 10 AM to Evening 5 PM. I would be available at any time during this hour. My timezone would be GMT+5:30. I would be devoting more than 40 hours per week to this project.

Any doubts and clarification, I can use my mail or if provided Organization discussion channel.

Post GSoC

Post GSoC I would be mostly working and contributing to the ML4SCI and its Umbrella Organization. Open Source is something that always enticed me and this would be the doorway. Hopefully, this experience would make me more skilled as a programmer and teach me the importance of open-source contributions. Also, I am hopeful the research part of this project would further my interest in research and spark a passion to contribute to more open-source developments.

References

1. https://ml4sci.org/gsoc/2023/proposal_DEEPLENSE4.html
2. https://en.wikipedia.org/wiki/Weakly_interacting_massive_particles
3. https://lenstronomy.readthedocs.io/en/latest/
4. https://arxiv.org/abs/2008.12731
5. https://arxiv.org/abs/1909.07346
6. https://arxiv.org/abs/2112.12121
7. https://arxiv.org/pdf/1909.07346
8. https://github.com/ML4SCI/DeepLense/tree/main/Equivariant_Neural_Networks_for_DeepLense_Apoorva_Singh
9. https://github.com/ML4SCI/DeepLense/tree/main/Transformers_Classification_DeepLense_Kartik_Sachdev
10. https://github.com/ML4SCI/DeepLense/tree/main/DeepLense_Classification_Transformers_Archil_Srivastava
11. https://arxiv.org/abs/2010.11929
12. https://arxiv.org/pdf/2103.00112.pdf
13. https://arxiv.org/pdf/2103.14030.pdf
14. https://arxiv.org/abs/2106.02689
15. https://arxiv.org/abs/2101.11986
16. https://arxiv.org/abs/2102.12122
17. https://arxiv.org/pdf/2102.10882.pdf
18. https://arxiv.org/pdf/2103.15808.pdf
19. https://arxiv.org/abs/2108.00154
20. https://arxiv.org/abs/2103.17239
21. https://arxiv.org/abs/2103.14899
22. https://arxiv.org/abs/2104.05704
23. https://arxiv.org/abs/2104.13840
24. https://arxiv.org/pdf/2112.07658.pdf
25. https://arxiv.org/abs/2105.07926
26. https://arxiv.org/abs/2107.00652
27. https://arxiv.org/abs/2108.0589