GSoC Proposal

ML4SCI: Transformers for Dark Matter Morphology with Strong Gravitational Lensing

Name:	Gautham Krishnan P
Github:	https://github.com/gauthamk02
Test Solution repo link:	https://github.com/gauthamk02/deeplense-test-2023
Country (Current City):	India (Kollam, Kerala)
Preferred Time Zone:	Indian Standard Time, IST (UTC +5:30)
Expected working hours:	30 hours/week
Preferred method of contact:	Email: gauthamkrishnanpriya@gmail.com Phone: +91 9526782285

Introduction

By utilising deep learning methods and strong gravitational lensing simulations, DeepLense enables us to probe the substructure of dark matter, distinguish between different dark matter models, and potentially revolutionize our current understanding of the fundamental nature of the universe[1].

Existing DeepLense algorithms — many of which were developed during previous GSoC projects — are capable of accurately identifying the presence of substructures and classifying the type of substructure model[6], using regression models to predict axion mass density[7] and anomaly detection from the gravitational lensing images[8]. These projects have explored and benchmarked various deep learning models such as ResNet, Vision Transformers, Auto-Encoders, and Equivariant models and also tried out various techniques like Domain Adaptations for the above-mentioned tasks.

This project aims to incorporate Transformer models to augment the capabilities of the DeepLense algorithms on tasks such as classification and regression. Some of the reasons why vision transformers will outperform traditional CNN-based models in working with gravitational lensing images are given below:

- Vision-Transformers[2] are capable of capturing the global context and handling long-range dependencies on the given image. This feature makes vision transformers more suitable for lensing images as the effect of substructures can be subtle and spread out over a large area creating wide distortions on the lens.

- Hybrid-transformer models that use both convolutional and transformer layers are becoming popular for their capability to capture both local and global contexts as they use convolutional layers as a backbone for extracting local information and feeding it into the transformer layers. Convolutional layers are exceptionally good at capturing local context and will help in identifying the precise location and shapes of local substructures.

Few of these hybrid models were explored for the specific test and their advantages. Their workings are listed in the coming pages.

The ultimate goal of this project is for the further development of the existing DeepLense pipeline and to better understand the underlying nature of dark matter, particularly WIMP particle dark matter, as well as other models such as vortex substructure of dark matter condensates and superfluids, through the analysis of strong gravitational lensing data.

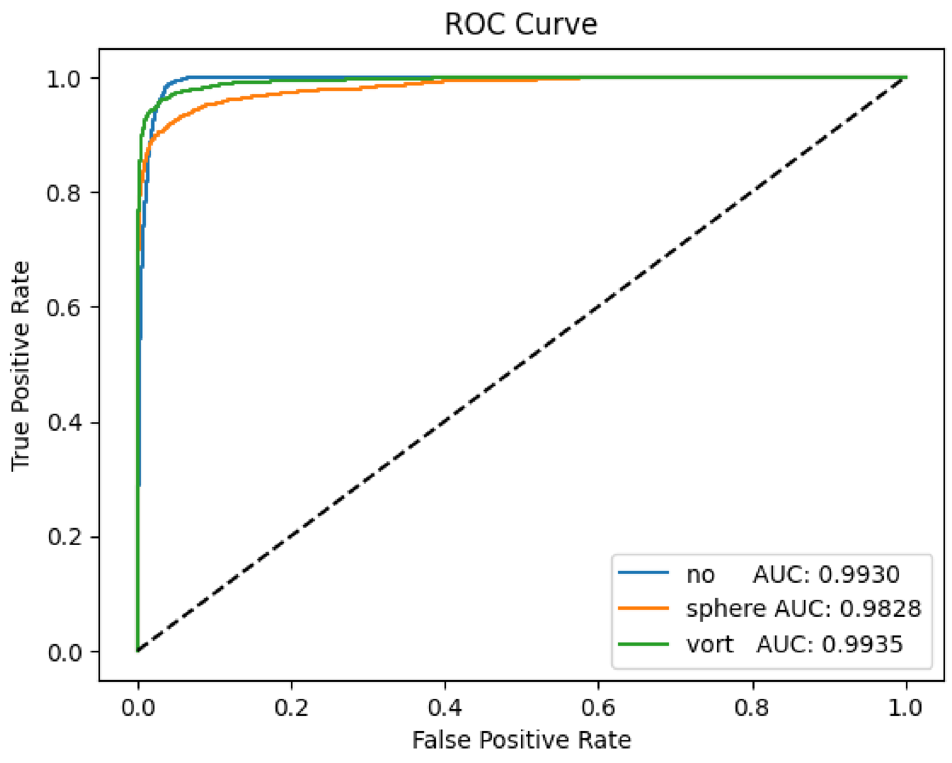
Common Test

For the common test, we were supposed to build a classification model to classify the lensing images into no substructure, subhalo substructure and vortex substructure. Initially, common classification models such as ResNet34, ResNet101, EfficientNet etc., were tried but they all gave very poor accuracies. Later, Vision Transformers and Equivariant Neural Networks were tried which gave good accuracies of which ENN was used for the final model as it performed better.

Considering that the lensing images are invariant in rotation and reflection — an Equivariant Network — C8SteerableCNN was built using the e2cnn library. The library provides a set of neural network modules that are equivariant under all isometries E(2) of the image plane $\mathbb{R}^2$, including translations, rotations, and reflections. The C8SteerableCNN model is designed to be equivariant under all rotations by 45 degrees (2*pi/8). It comprises six R2 Convolutional blocks, with a pooling operation after every two blocks, and fully connected layers at the end.

Results (test-set):

Accuracy: 95%
ROC-AUC Score (Overall): 0.9926442933333334



Specific Test

As part of the specific test for this project, we had to use Vision Transformer models for detecting the presence of substructure in lensing images. In this test, multiple transformer models were explored with different hyperparameter configurations. The three best-performing models were uploaded to the GitHub repository — and their hyperparameters, results and analysis are shown below.

	CrossFormer	**RegionViT**	**CvT**
Model Definition	`CrossFormer( num_classes = 2, dim = (64, 128, 256, 512), depth = (2, 2, 8, 2), global_window_size = (8, 4, 2, 1), local_window_size = 7)`	`RegionViT( dim = (64, 128, 256, 512), depth = (2, 2, 8, 2), window_size = 7, num_classes = 2, tokenize_local_3_conv = False, use_peg = False)`	`CvT(num_classes= 2, dropout= 0.1)`
Epochs	30	30	30
Initial learning rate	0.0001	0.0001	0.0001
Optimiser	Adam	Adam	Adam
Criterion	Cross Entropy Loss	Cross Entropy Loss	Cross Entropy Loss
Learning rate scheduler and its parameters	Cosine Annealing LR T_min = 0.01 * initialLR	Step LR Step size = 10 gamma = 0.1	Cosine Annealing LR T_min = 0.01 * initialLR
Results (test set)			
Accuracy	0.99	0.94	0.99
ROC-AUC Score (overall)	0.99986	0.98913	0.99992
No Substructure	0.9998	0.9902	0.9999
Substructure	0.9999	0.9891	0.9999

It can be seen that all three models result in good accuracy and ROC-AUC scores. Initially, the traditional ViT model was tried but the accuracy was below 85% even with extensive data augmentation, thus it was discarded in place of more specific Vision Transformer models.

The basic difference between each of the models used for this test is given below

1. CvT (Convolutional Vision Transformer)[3]:

   Advantages:
   - Combines the strengths of CNNs and Transformers, allowing it to capture both local and global information more efficiently which lets it accurately identify the complex features of lensing images.
   - Can handle large images well.

   Disadvantages:
   - Requires more training time than traditional CNNs due to the hybrid architecture.

2. CrossFormer[4]:

   Advantages:
   - Can capture both local and global features at multiple scales using cross-scale attention mechanisms.
   - Achieves state-of-the-art performance on various image classification tasks.

   Disadvantages:
   - May require more computational resources than other Vision Transformer models due to the cross-scale attention mechanisms.

3. RegionViT[5]:

   Advantages:
   - Can efficiently capture both local and global information by dividing the image into regions and using a region-based attention mechanism.

   Disadvantages:
   - May not be suitable for large images due to the region-based approach.

It is evident from these tests that hybrid transformer models that make use of both convolution and transformer layers give the best results for the given test cases. The substructures present in galaxies can be small and their effect on the lens can be subtle. Therefore, convolutional layers will help in extracting such fine-grained local features and the transformer layers will allow capturing the global context as the effects of the substructures can be spread over different regions. Moreover, taking a hybrid architecture allows us to get the advantages of both traditional convolutional networks and transformer networks.

Implementation Plan

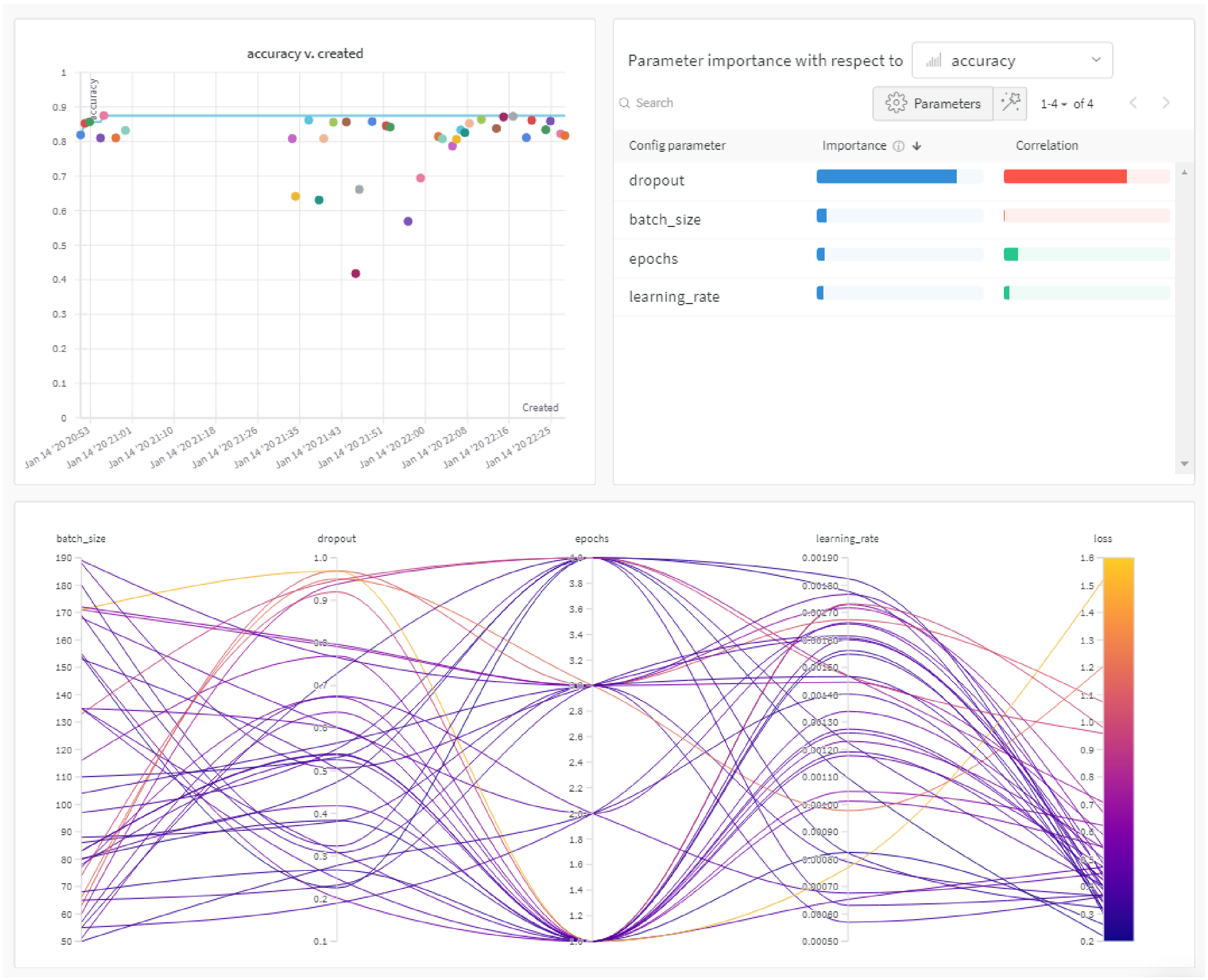
For this project, we have to expand the existing DeepLense pipeline by integrating transformer models for computer vision tasks applicable to DeepLense Data. Completing the common and specific test required working with the simulated lensing images extensively which gave many useful insights into the approaches that would yield better results. Keeping these insights in mind, the below implementation plan is proposed.

1. **Model Training**

   The most important part of the project is on implementing transformer models for tasks applicable to DeepLense data which will require trying out various vision transformer models, benchmarking and optimising them to find the best-performing models for different tasks. So the two major parts of model training are.

   a. Model Exploration: Multiple vision transformer models were proposed in the last few years, each having its own advantages and use cases such as the ones used for the specific test. Implementations of these different models can be found in libraries like vit-pytorch or their corresponding papers. We have to explore these different hybrid vision-transformer models available and train and benchmark them on the DeepLense Data.

   b. Hyperparameter Searching: After finding the better-performing models we can do hyperparameter searches on them. Hyperparameter searching allows us to find the optimal configuration of hyperparameters for a given model. Functions like wandb.sweep() by Weights & Biases provides different methods for hyperparameter searching like Bayesian, Grid and random search to search the hyperparameter space. By using it on the models for DeepLense Data, we can get the best performance out of them.



“Example image of using hyperparameter sweep provided by Weights & Biases to automate hyperparameter search and explore the space of possible models“

2. Python Package & Repository

Once the models are trained, a repository has to be made with all the necessary documentation about the models, configurations and results.

a. Document all the design principles that went into the final models and results in Jupyter Notebooks and README files.

b. Create a repository containing Python scripts for custom training or inference of the built models based on arguments.

c. Additionally, make a Python package from which models can be imported with the option for custom training or inference using the pre-trained models if time permits.

3. Deployment

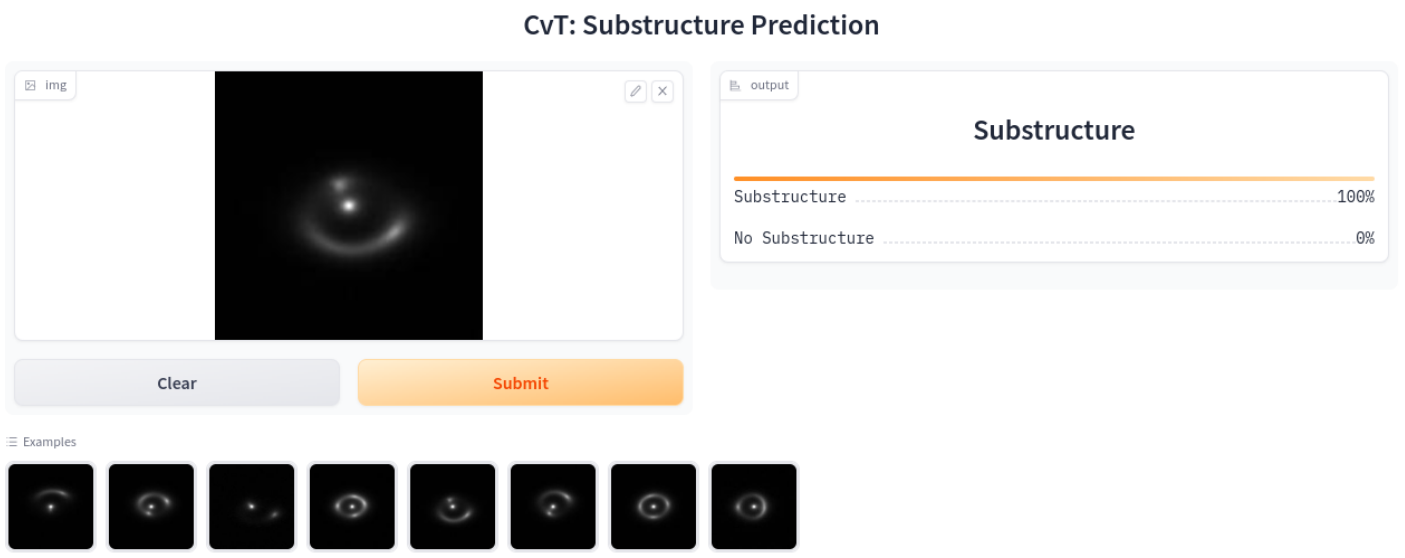
Deploying the trained models will make them more accessible and easy to test. By converting the model into ONNX format, we can make the model easily deployable using free services like Hugging Face Spaces for easy access and quick inference.

a. Convert the trained Pytorch models into ONNX format using torch.onnx.format() and ensure its similarity with the PyTorch model.

b. Deploy all the trained models to Hugging Face Spaces. Libraries like Gradio can be used for building a clean and intuitive frontend for deployment.

Below is an example of a working deployment of the CvT model which I made for test-5 in Hugging Face Spaces.

Deployment Link: https://huggingface.co/spaces/gauthamk/deeplense_cvt_classiffier



Substructure

```
Substructure ........................ 100%
No Substructure ..................... 0%
```

Timeline

Project Duration: 175 hours, 3 months

Week	Dates	Objective(s)
-	May 4	Accepted GSoC contributor projects announced.
-	May 4 - May 28	● Community bonding period. ● Get to know the mentors and interact with other contributors. ● Research more about the problem statement and different Vision Transformers on the side.
1	May 28 - June 3	● Explore existing DeepLense projects. ● Get familiar with the datasets and try out different preprocessing operations on them.
2 - 3	June 4 - June 17	● Start working on the problem statement. Begin with Classification models. ● Try out different hybrid Vision Transformer models on the dataset and benchmark them.
4	June 18 - June 24	● Present the progress and work to mentors. ● Take their feedback and implement the necessary changes in the project.
5	June 25 - July 1	● Start working on the Regression models for various real/simulated lensing images to find the mass densities of the substructures. ● Use the insights gained from previously working with classification models and implement the transformers models for different regression problems.
6	July 2 - July 8	● Take feedback from mentors on the progress of regression models and make the necessary changes.
7	July 9 - July 15	● Buffer Week to complete the work in case any of the previous work is remaining. ● Start writing the Midterm Blog.
	July 14	Midterm Evaluation
8 - 9	July 16 - July 29	● Start making the repository containing all the

		models with the code for inference and training. ● Add options like argument parsing for hyperparameters and data augmentation in the python scripts. ● Complete making the script and add the necessary documentation to the repository.
10	July 30 - August 5	● Start working on deploying the models. ● Convert all trained models to ONNX format, write the inference code and start making the frontend in Gradio. ● Deploy all the trained models to Hugging Face.
11 - 12	August 6 - August 19	● Take review from mentors on the work till now. ● Make the necessary changes and complete the codebase.
13	August 20 - August 27	● Finalize the project. ● Start writing the final blog
	August 28	Final Date of submission

Further Work

I plan to build upon the work I completed during my GSoC project by utilizing the insights I gain from the project. Some of the future works I have in my mind are:

- Implementing a Python library to integrate the data generation (using packages such as PyAutoLens and Lenstronomy) and commonly used DeepLense models into one modular package that is convenient and easy to extend for different applications. Some of the features that could be implemented in the Library include:
    - Different DeepLense models for tasks such as classification, regression and anomaly detection in a separate module that can be imported easily with the option for pre-trained weights or custom training.
    - Generating data based on user/pre-defined parameters for training and testing of models.
    - Utility functions that will automatically test the compatible models and benchmark them on a dedicated DeepLense dataset using a given metric. This would allow researchers to easily compare their models against the existing ones and gain valuable insights.

Such a library would take a considerable amount of work and time to implement but it will significantly reduce the effort in exploring new methods for the problems that are relevant to the DeepLense projects and increase the development speed.

- Currently, all the DeepLense projects are present in their respective GitHub repositories. Trying them out will require cloning the repositories and following the documentation to get an inference. However, the DeepLense models can be deployed using free services like Hugging Face Spaces, as I have mentioned in my implementation plan. This would make the DeepLense models more accessible and easier to try out.

- Follow up with the work I did for GSoC and continue working with the mentors on extending my project further. Continue being part of the community and be a part of future projects.

About Me

I am a sophomore CSE student studying at Amrita Vishwa Vidyapeetham, Amritapuri. I am interested in Machine Learning and primarily work with Computer Vision Based models and have completed multiple projects in it.

I have previously been part of a small research internship meant to introduce students to research under a faculty from my university. As part of the internship, I explored different methods for Sample Size Determination for training machine learning models to achieve a certain accuracy and explored various algorithms and evaluated their effectiveness in different scenarios. During the internship, I also explored McNemar's and other statistical tests for determining the better-performing model. Link to the final report.

I also read research papers on the field frequently and have implemented some of the common Machine Learning architectures from scratch. Most of the work I do is on PyTorch and have hosted some of my below-mentioned work in Hugging Face. Other than Machine Learning, I also know frameworks like React, Django and Flutter and have built many projects using these frameworks (DS-Visualiser, Flutter Chat App).

Some of my recent projects include:

- Making UNET and ResNet34 models from scratch using PyTorch and hosting them on Hugging Face after training on water-body segmentation from satellite images and bird-classification datasets respectively. You can find them deployed here and here.

- JUNO-Editor, an open-source desktop app built as part of the NASA SpaceApps Challenge for processing and editing the raw multichannel grey-scale images captured by JUNO Spacecraft and combining them to make an RGB photo with additional functionalities for image processing and auto-enhancement in which I was working on implementing the Image processing and enhancement algorithms for our specific use case.

- PocketTracker, a mobile app built using Flutter and Firebase for expense tracking that can automatically parse the details from the bill photo and update the database.

Moreover, working on these diverse projects has helped me get a good understanding of the software development process and I have developed the skill for learning new things quickly and implementing them.

I also participate routinely in hackathons, workshops and other technical events and I am an active member of India's leading student-run open-source club amfoss.in and have made minor contributions to many open-source projects such as DevoWorm, DeepChem, Wikimedia projects etc.

For more information, you can check out my CV.

References

[1] Alexander, S., Gleyzer, S., McDonough, E., Toomey, M. W., & Usai, E. (2019). Deep Learning the Morphology of Dark Matter Substructure. *ArXiv*. https://doi.org/10.3847/1538-4357/ab7925

[2] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv*. https://doi.org/10.48550/arXiv.2010.11929

[3] Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2021). CvT: Introducing Convolutions to Vision Transformers. *ArXiv*. https://doi.org/10.48550/arXiv.2103.15808

[4] Wang, W., Chen, W., Qiu, Q., Chen, L., Wu, B., Lin, B., He, X., & Liu, W. (2023). CrossFormer++: A Versatile Vision Transformer Hinging on Cross-scale Attention. *ArXiv*. https://doi.org/10.48550/arXiv.2303.06908

[5] Chen, C., Panda, R., & Fan, Q. (2021). RegionViT: Regional-to-Local Attention for Vision Transformers. *ArXiv*. https://doi.org/10.48550/arXiv.2106.02689

[6] Archil Srivastava. DeepLense Classification Transformers https://github.com/ML4SCI/DeepLense/tree/main/DeepLense_Classification_Transformers_Archil_Srivastava

[7] Yurii Halychanskyi. DeepLense Regression https://github.com/ML4SCI/DeepLense/tree/main/DeepLense_Regression_Yurii_Halychanskyi

[8] Saranga K Mahanta. Updating the DeepLense Pipeline https://github.com/ML4SCI/DeepLense/tree/main/Updating_the_DeepLense_Pipeline__Saranga_K_Mahanta