# ML4SCI

## Proposal – Google Summer of Code (GSoC) 2023

## I. Personal Details:


- Name: Rahil Parikh
- Mobile: +91 9920828960
- Email: rahilparikh2002@gmail.com
- Country of Residence: India
- Degree: Pursuing a BTech (Bachelor of Technology) in Computer Engineering
- College: K. J. Somaiya College of Engineering (KJSCE)
- Year of Study: TY (Third Year)
- Time-zone: IST (GMT+5:30)
- Language: English


## II. Project Title:

Transformers for Dark Matter Morphology with Strong Gravitational Lensing

## III. Overview:

Dark matter is a hypothetical form of matter, which does not interact with electromagnetic radiation. It does not reflect or emit light and is not directly observable by the human eye. Its existence, however, can be inferred by observing gravitational effects on visible matter, such as stars and galaxies. Gravitational lensing causes light to bend in the presence of a strong gravitational field. Due to this bending of light, distant objects may appear to be distorted or magnified. The study of these distorted shapes can aid researchers in identifying the distribution and location of dark matter.

The main aim of this project is to train an efficient Vision Transformer model to perform image classification and distinguish between gravitational lensing images. Different transformer architectures will be explored to create a robust model for image classification. Furthermore, creating multiple ensembles of transformer models and evaluating their performance (using ROC AUC), with respect to other transformer models, and other ensembles will help in identifying the most effective neural network architecture capable of accurately classifying images of the given dataset. The total project duration will be 175 hours. The deliverables of this project are as follows:

- Python package making use of Vision Transformers (ViTs).
- Scripts for training and validation of models on the gravitational lensing dataset.
- Open-source, easily accessible, and trained Vision Transformer models, that can be applied to gravitational lensing images.
- Ready-to-use ensembles of Vision Transformers that can aid in the task of image classification.

As part of my evaluation tests for the DeepLense project, I have completed Common Test I and Specific Test V. My approach for completing Test I, consisted of training 6 deep learning models, including an ensemble of two DenseNet models (DenseNet161 and DenseNet201). The final prediction consisted of an average prediction of all 6 trained models, achieving an AUC (OVO) score of 0.99, and an AUC (OVR) score of 0.99. In order to complete Task V, data augmentation methods were applied to the training and validation sets. A total of 3 deep learning models were trained: 2 transformer models and one ensemble model. The final predictions were obtained by averaging the predictions of all 3 transformers trained. The averaged predictions resulted in an AUC (OVO) score of 1.00000, and an AUC (OVR) score of 1.00000. The code and results have been uploaded to a Github repository. The trained models for each task can be accessed and downloaded from Google Drive.

Based on the results obtained after successfully completing Specific Test V (Exploring Transformers), I strongly believe that the models trained during this project can be used to accurately classify gravitational lensing images, and expand the functionality of the DeepLense project to include Vision Transformers.

## IV. Previous Open-Source Contributions:

I have been contributing to open source for more than a year. Most of my open-source contributions have been to scikit-learn, a popular machine-learning package. My contributions range from documentation improvements and general maintenance to specific enhancements and bug fixes. Collaboration and communication, with the maintainers of scikit-learn has helped me gain a vast amount of knowledge and has given me a better insight into the fundamentals of machine learning. Fixing bugs or enhancing specific functions that make it easier for people to use machine learning algorithms, has motivated me to contribute more to open source. To date, I have created 30+ Pull Requests (PRs) to scikit-learn.

Some of my noteworthy open-source contributions to scikit-learn are as follows:

| Pull Request (PR) | Status | Description |
|---|---|---|
| PR #24033 | Merged | Includes parameter validation for Lasso, Lars and their associated variants. |
| PR #24350 | Merged | Preserves the data type for SkewedChi2Sampler. |
| PR #24714 | Merged | Preserve the data type for Isomap, based on the input given to it. |
| PR #25530 | Merged | Specifies the impact of the “tol” hyperparameter for solvers in Ridge Classifier. |
| PR #25291 | Merged | Raises an error when AdditiveChi2Sampler is not fit properly. |
| PR #25324 | Merged | Raises the required error messages for Stacking Classifier, Stacking Regressor, Voting Classifier, and Voting Regressor. |
| PR #25367 | Merged | A “NotFittedError” is raised in 3 Imputers and in Isotonic Regressor. |

| | | |
|---|---|---|
| PR #23878 | Merged | General maintenance, that validates parameters for Orthogonal Matching Pursuit and Orthogonal Matching Pursuit CV. |
| PR #25350 | Pending second approval | Includes a new example for time-series forecasting with lagged features and prediction intervals using Histogram Gradient Boosting Regressor. |
| PR #25602 | Pending second approval | Improves the EDA (Exploratory Data Analysis) and explanation for an example on feature engineering. |

## V. Milestones:

Based on the project to be created the proposed milestones are as follows:

1. **Define the basic package structure:**
    a. Create the modules and sub-modules required for the dataset, utilities, models, and results.
    b. Define the arguments that can be passed by the user, to the application through the command line. Based on these arguments, augmentations, data loading and model training will be carried out. Some examples of command line arguments that can be utilized by the user are as follows:
        i. Augmentations: Allows the user to apply augmentations to the data, if required.
        ii. Number of Workers: This value can be passed to the dataloader instance to utilize multiple sub-processes for data loading.
        iii. Ensemble: A Boolean or Integer value indicating whether the user needs to train an ensemble model or not.
        iv. Model Name(s): Based on the available models and the value of the "Ensemble" argument, the user can select the model name(s) required for training.
        v. Batch Size: This specifies the number of samples processed before the model is updated.
        vi. Dropout: The value of the dropout to be used in the model, to prevent overfitting.
        vii. Learning Rate: This is used to control the rate at which the model updates the values of its parameters.
        viii. Number of Epochs: Allows the user to control the number of times all the training data is passed through the neural network.
        ix. Device: This specifies whether the model is to be trained on the GPU or the CPU.
        x. Random Seed: This allows the user to set the seed for random number generation. Setting the same random seed across all runs ensures that the results are reproducible.
2. **Generate Dataloaders:** Apply augmentations to the given dataset (optional) and generate training and validation dataloaders to be passed to the Vision Transformer model.

3. **Single Model Training:** In order to provide easily accessible open-sourced Vision Transformers to the user, the following models can be trained and evaluated on the gravitational lensing dataset:
    a. Vision Transformers (ViTs): The following transformer models and/or their variants can be trained on the gravitation lensing dataset:
        i. Convit_Base
        ii. Convit_Small
        iii. Coatnet_1_224
        iv. Coatnext_nano_rw_224
        v. Deit3_Base_Patch16_224
        vi. Gcvit_Base
        vii. Levit_192
        viii. Maxvit_Base_224
        ix. Maxvit_rmlp_Small_rw_224
        x. Mobilevit_s
        xi. Mvitv2_Base
        xii. Vit_Base_Patch16_224
        xiii. Vit_Base_r26_s32_224
        xiv. Vit_Base_resnet26d_224
        xv. Vit_relpos_Base_Patch16_224
    b. Swin Transformers: The swin transformer is a type of vision transformer that builds hierarchical feature maps by merging image patches in deeper layers. As part of the project the following different Swin Transformers and/or their variants can be trained:
        i. Swin_Base_Patch4_Window7_224
        ii. Swin_s3_Base_224
        iii. Swin_s3_Small_224
        iv. Swinv2_cr_Base_224
        v. Swinv2_cr_Tiny_224
4. **Ensemble Model Training:** To achieve better results for the image classification task, multiple ensembles of Vision Transformers can be explored. As part of the project the following ensemble models can be trained:
    i. Swin_s3_Base_224 and Swinv2_cr_Base_224
    ii. Vit_Base_Patch_16_224 and Swin_Base_Patch4_Window7_224
    iii. Convit_Base and Mobilevit_s
    iv. Deit3_Base_Patch16_224 and Gcvit_Base
    v. Maxvit_Base_224 and Swinv2_cr_Tiny_224
5. **Model Evaluation and Results:** Evaluate the performance of the aforementioned models using ROC curves and AUC scores. The final results of all the models, including their weights, will be saved and made available to users of the package.

**Note:** The models listed above are just some of the models that can be used to classify images. The command line arguments used, and the final models trained may vary based on feedback received from the mentors and the availability of adequate computational resources (such as GPUs). Google Colab or Kaggle might need to be used to train the deep learning models.

## VI. Timeline:

Based on the deliverables and milestones of this project, the proposed timeline is as follows:

1. **5th April to 3rd May:**
    a. Exploring and understanding different architectures of Vision Transformers.
    b. Getting familiar with the process of creating Python packages.
2. **4th May to 28th May:**
    a. Community Bonding
    b. Analysing the suggestions of the mentors for the project
    c. Getting familiar with the ML4SCI organization and getting up-to-speed to begin working on the proposed project.
3. **29th May to 4th June:**
    a. Begin the coding for the project. Create the required modules and sub-modules for the Python package.
    b. Allow the user to pass arguments through the command line to execute the code.
4. **5th June to 11th June:**
    a. Pre-process the images of the training and validation dataset.
    b. Apply augmentations to the training dataset.
    c. Generate training and validation dataloaders to be passed to the model.
5. **12th June to 2nd July:**
    a. Start training the Vision Transformer (ViT) models.
    b. Evaluate the performance of each model trained.
6. **3rd July to 14th July:**
    a. Review suggestions of the mentors before the midterm evaluation deadline.
    b. Incorporate the suggestions into the project.
    c. Submit the midterm evaluation.
7. **15th July to 1st August:**
    a. Training ensembles of Vision Transformers.
    b. Evaluation of the trained models.
8. **2nd August to 14th August:**
    a. Save and display the results of the trained models.
    b. Test the Python package.
9. **15th August to 28th August:**
    a. Incorporate the changes suggested by the mentors (if any) into the project.
    b. Final testing of the functionalities of the project built.
    c. Final GSoC project submission.

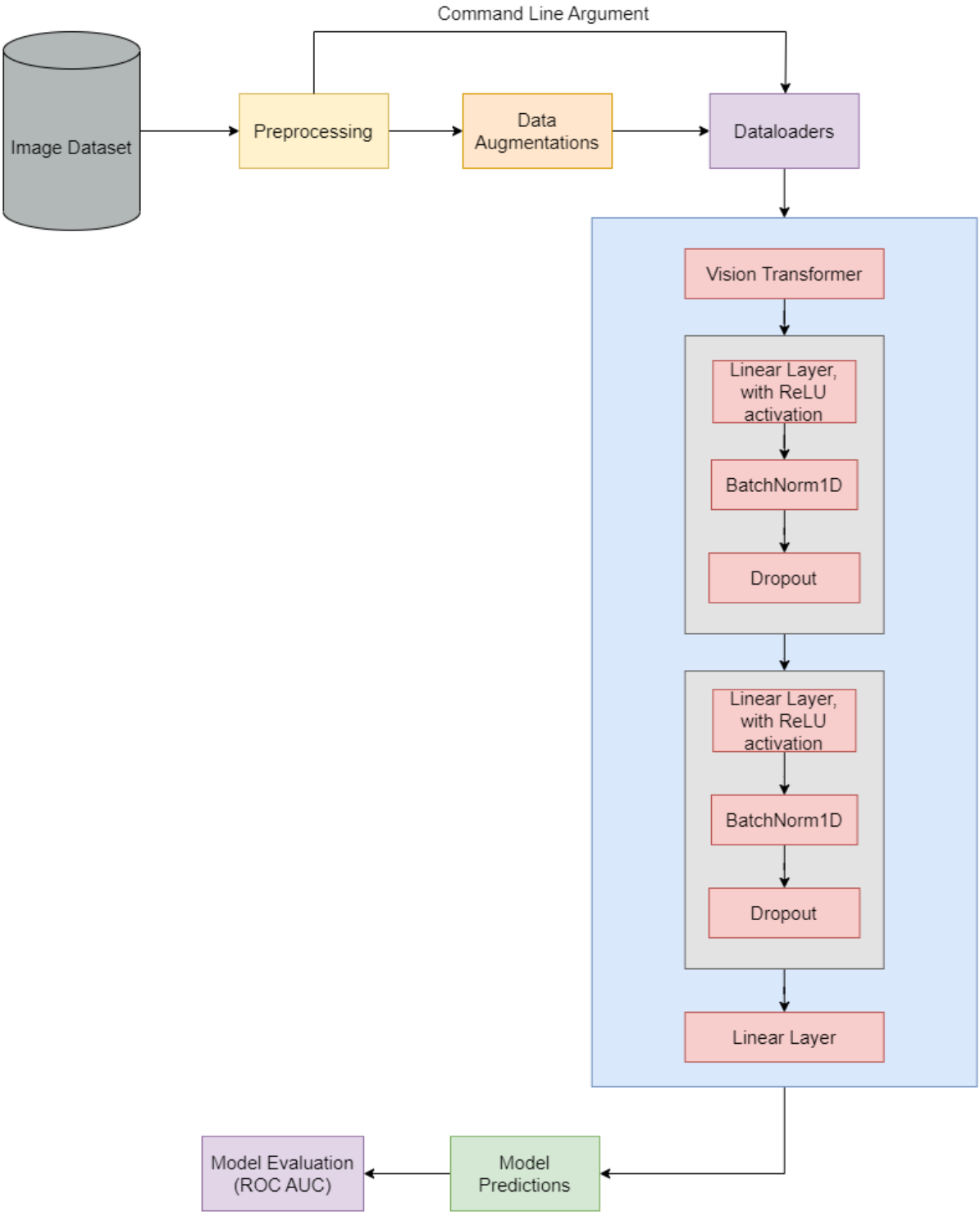
**Note:** The number of models that can be trained depends on the availability of GPU resources. Kaggle offers 30 hours of free GPU per week, which may be used to train the aforementioned models.

## VII. Engagements During the Summer:

My college exams will start on 10th May and end on 3rd June. After that, I will be working as a Data Analyst Intern at Think360.ai for 2 months during the summer (June and July). I will be able to work on my GSoC project during weekdays in the morning (6am to 8am), and in the night (9pm to 11:30pm). Apart from this, I will be free over the weekends, during which I can

dedicate the entire day to working on the GSoC project. I strongly believe that on average I can spend atleast 30 hours a week working on the project.

## VIII. Proposed System Architecture:



The system architecture is as follows:

1. The first step involves loading the image dataset consisting of gravitational lensing images.
2. The second step involves image pre-processing operations such as normalization and resizing the images of the dataset to a fixed height and width.
3. After pre-processing, if the user chooses to apply augmentations, the following operations will be applied to the images using albumentations:
    a. Train Set Transformations: The train set will include the following image transformations:
        i. HorizontalFlip
        ii. VerticalFlip
        iii. Rotation
        iv. ShiftScaleRotation
        v. RGBShift
        vi. RandomBrightnessContrast

    If the user does not want to apply augmentations to the dataset, the images will be passed directly to the dataloaders.
4. The dataloaders will be used to feed the images of the dataset into the Vision Transformer.
5. The neural network used to classify the images will use a Vision Transformer backbone along with the following additional layers:
    a. Linear Layer with ReLU activation function
    b. 1D Batch Normalization
    c. Dropout (as a form of regularization, to prevent overfitting)
    d. Linear Layer with ReLU activation function
    e. 1D Batch Normalization
    f. Dropout (as a form of regularization, to prevent overfitting)
    g. Final linear layer consisting of 2 neurons

    The neural network architecture may be altered to improve the performance of the model. For ensembles of Vision Transformers, two or more Vision Transformers will be used to create a new ensembled model.
6. Next, the model will be made to predict the classes of different images being fed into it.
7. The last step involves the evaluation of the model using ROC curves and AUC scores. To better understand the performance of the models, F1-scores and confusion-matrices can also be analyzed.

## IX. Previous Projects and Research Work:

I am a Notebooks Master on Kaggle and a Microsoft Learn Student Ambassador (MLSA). Some of my previous projects in the field of machine learning and deep learning are as follows:

1. Bird Classifier: Flutter App that can classify 400 different species of birds using MobileNetV2. The model achieves an accuracy of 93.99% on the test set. The prominent features of the app are:
    a. Take and upload a picture from your phone camera.
    b. Upload an image from Google Drive or your mobile gallery.
    c. Identify the species of the bird.

d. Get the confidence of the prediction made.
e. Look at the performance of the model and understand the architecture of the neural network.

2. Music Genre Recommender: Music Genre Recommendation website that can identify and recommend 10 different genres of music using Light Gradient Boosting Machine (LGBM). The model achieves an accuracy of 90% on the test set and an F1 score of 0.90. The training data consists of 1000 audio samples each of a duration of 30 seconds. The model is deployed using Flask. Optuna was used to perform hyperparameter tuning and improve the accuracy of the model by 8% (from 82% to 90%). Once an audio file is uploaded to the website by the user, 58 different features are extracted and passed to the model to accurately identify the genre of music. Relevant song recommendations are generated using cosine similarity and the classified genre of music.

My research experiences in the field of Deep Learning and NLP are as follows:

1. "A Transfer Learning Approach for Classification of Knee Osteoarthritis", ICEEICT 2023 (International Conference on Electrical, Electronics, Information and Communication Technologies). Paper accepted, pending publication.
2. "Identifying Instances of Cyberbullying on Twitter Using Deep Learning", ICISA 2023 (International Conference on Intelligent Systems and Applications). Paper accepted, pending publication.

## X. Leadership and Internship Experience:

My leadership and internship experiences are as follows:

1. **Team Captain**
   *The Marine Robotics Team (TMRT)*
   - Managed a team of 25+ undergraduate students focused on building an Autonomous Underwater Vehicle (AUV).
   - Achieved a mean average precision of 96.27% using YOLOv5 models for detection of underwater objects.
   - Prepared simulations for the robot using ROS Melodic, Ubuntu 18.04 and Gazebo.
   - Worked on the prototyping of a social utility robot capable of cleaning contaminated water sources.
2. **Research Intern**
   *K. J. Somaiya College of Engineering*
   - Identified Knee Osteoarthritis from X-ray images using Deep Learning. Curated a custom dataset containing 5,478 images for the binary classification task and corrected the class imbalance present in the dataset.
   - Utilized the concept of Transfer Learning, using DenseNet201 to achieve a precision of 97.62%.
3. **Machine Learning Intern**
   *The Tann Mann Gaadi*
   - Extracted 500+ frames from videos and manually labelled them using LabelImg and online annotation tools available on Roboflow. Detected 80 different classes of objects in images and videos using OpenCV and YOLOv3 models.

- Automatically saved the results of the object detection in excel spreadsheets containing the name of the class, the accuracy of the detection and the coordinates of the bounding boxes.

**4. Machine Learning Intern**

*Verzeo*

- Downloaded 600 images from the internet using web scraping, for training a deep learning model. Created a neural network to classify an image as ‘Human’ or ‘Non-Human’ with an accuracy of 94.2%.
- Compared the performance and results of 10 different architectures of Convolutional Neural Networks (CNNs).