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:

Deep Regression Techniques for Decoding Dark Matter with Strong Gravitational Lensing

III. Overview:

Dark matter is a hypothetical form of matter composed of particles that do not absorb, reflect or emit light. Since dark matter does not interact with the electromagnetic field, it can be difficult to detect. The presence of dark matter can be inferred from phenomena such as gravitational lensing. Massive objects such as clusters of galaxies, lying between a distant source and an observer behave as a lens, bending light from the source. By analysing a variety of different images, it is possible to deduce the distribution of dark matter. Furthermore, by measuring the distortion geometry, the mass of the surrounding cluster of dark matter can be determined.

This project aims to train effective neural networks (CNNs and ConvNeXt variants) to learn the mapping between gravitational lensing images and the dark matter halo mass. Multiple architectures of CNNs and ConvNeXt variants will be explored to predict the mass of dark matter. The ability of the model to predict the mass of dark matter will be evaluated using MSE (Mean Squared Error) as a performance metric. Furthermore, hyperparameter sweeps using WandB can help in tuning the hyperparameters of the neural network, thus resulting in an improved performance. The duration of the entire project will be 175 hours. The deliverables of the proposed project are as follows:

i. Creation of a Python package that makes use of CNNs and ConvNeXt models to predict the mass of dark matter.
ii. Scripts for training and validation of models on the provided dataset.
iii. Hyperparameter tuning of the neural network to improve its performance and generalizability for estimating the mass of dark matter from similar unseen images.
iv. Open-source and easily accessible deep learning models that can expand the functionality of the DeepLense project and aid users working on a similar task.

As part of my evaluation for the DeepLense project, I have successfully completed Common Test I and Specific Test III. 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. For Task III the images were split into training and testing datasets, from which the corresponding dataloaders were created. EfficientNetB4, ConvNeXtBase, and InceptionResNetV2 were trained to estimate the mass of dark matter. The aforementioned models achieved an MSE of 2.007×10^{-4} (0.0002007), 2.763×10^{-4} (0.0002763), and 2.618×10^{-4} (0.0002618) respectively. The code to complete the above tasks has been uploaded to a Github Repository. The models trained to complete the tasks have been uploaded and save to Google Drive.

On the basis of the results obtained, I firmly believe that it is possible to estimate the mass of dark matter from images of gravitational lensing. This project can be used to provide ready-to-use deep learning models to users working on similar tasks and can also be utilized to extend the functionality of the DeepLense project with regards to estimating the mass of dark matter.

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. **Creating the required modules and sub-modules:**
   a. Defining and creating the modules or sub-modules required for the dataset, utilities, dataloaders, models and results.
   b. Allowing the user to pass in arguments through the command line, that can be used to control pre-processing, data loading, model training and hyperparameter optimization of the required neural network. Some of the commands that can be passed by the user are as follows:
      i. Random Seed: This sets the same seed throughout the code and assists in providing the same reproducible results for different runs.
      ii. Number of Workers: This command line argument can be used to improve the efficiency of data loading by utilizing multiple sub-process.
      iii. Model Name: This allows the user to utilize specific models that are available in timm (a Python package containing state-of-the-art computer vision models).
      iv. Batch Size: This permits the user to control the number of images in each batch for the model. A high batch size for large models, might lead to Out of Memory errors. The batch size depends on the model being used and the amount of memory available to the user.
      v. Number of Epochs: The controls the amount of time the training data is passed through the model.
      vi. Device: This specifies whether the model should use the CPU or the GPU for training and testing purposes.
      vii. Hyperparameter Tuning: This gives the user the ability to specify whether the model needs hyperparameter tuning or not. Based on this value the user can tune the following hyperparameters of the model using hyperparameter sweeps with Weights and Biases (WandB):
         I. Optimizer: This gives the user control over the optimizer used to train the model. Example: Adam, SGD, etc.
         II. Fully Connected Layer Sizes: The user can alter the size of the fully connected layers used for training the specified neural network.
         III. Dropout: This specifies a random dropout of neurons in a particular layer of the neural network. Dropout acts as a form of regularization and prevents overfitting of the neural network.

IV. Learning Rate: This allows the user to control the rate at which the neural network updates its internal parameters. A very high learning rate would cause the model to jump over the global minima, whereas a very low learning rate would result in extremely slow convergence or an undesirable local minimum.

2. **Dataloader Generation:** This step involves the creation of training and validation/testing dataloaders that can be used by the neural network for training and performance evaluation.
3. **Model Training:** In order to find the most efficient model capable of predicting the mass of dark matter from gravitational lensing images, different neural network architectures need to be explored. The different models that will be trained are as follows:
    - i. DenseNet121
    - ii. DenseNet161
    - iii. DenseNet169
    - iv. DensetNet201
    - v. DenseNet264
    - vi. Inception_V3
    - vii. Inception_V4
    - viii. Inception_Resnet_V2
    - ix. VGG16
    - x. VGG19
    - xi. EfficientNet_b0
    - xii. EfficientNet_b2
    - xiii. EfficientNet_b5
    - xiv. EfficientNet_b6
    - xv. EfficientNet_b8
    - xvi. EfficientNet_lite0
    - xvii. EfficientNet_v2_m
    - xviii. CSPResNet50
    - xix. ECAResNet200d
    - xx. ResNet152
    - xxi. ResNet200
    - xxii. ResNetV2_101
    - xxiii. ResNetV2_152
    - xxiv. MobileNetV2_100
    - xxv. MobileNetV3_small_100
    - xxvi. ConvNeXtBase
    - xxvii. ConvNeXt_Atto
    - xxviii. ConvNeXt_pico
    - xxix. ResNeXt_101_32x4d
    - xxx. ResNeXt_101_64x4d
4. **Hyperparameter Tuning:** Based on the arguments provided by the user, the hyperparameters of the aforementioned models can be optimized to improve performance. Hyperparameter sweeps can be performed using Weights and Biases (WandB).
5. **Model Evaluation and Results:** The performance of the models will be evaluated using MSE (Mean Squared Error). The trained models will subsequently be open-sourced and made readily available to the user.

Note: The hyperparameters and models listed above are a few out of many that can be used to predict the mass of dark matter. The final hyperparameters used and models trained may vary based on feedback received from the mentors and the availability of adequate computational resources (such as GPUs).

VI. Timeline:

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

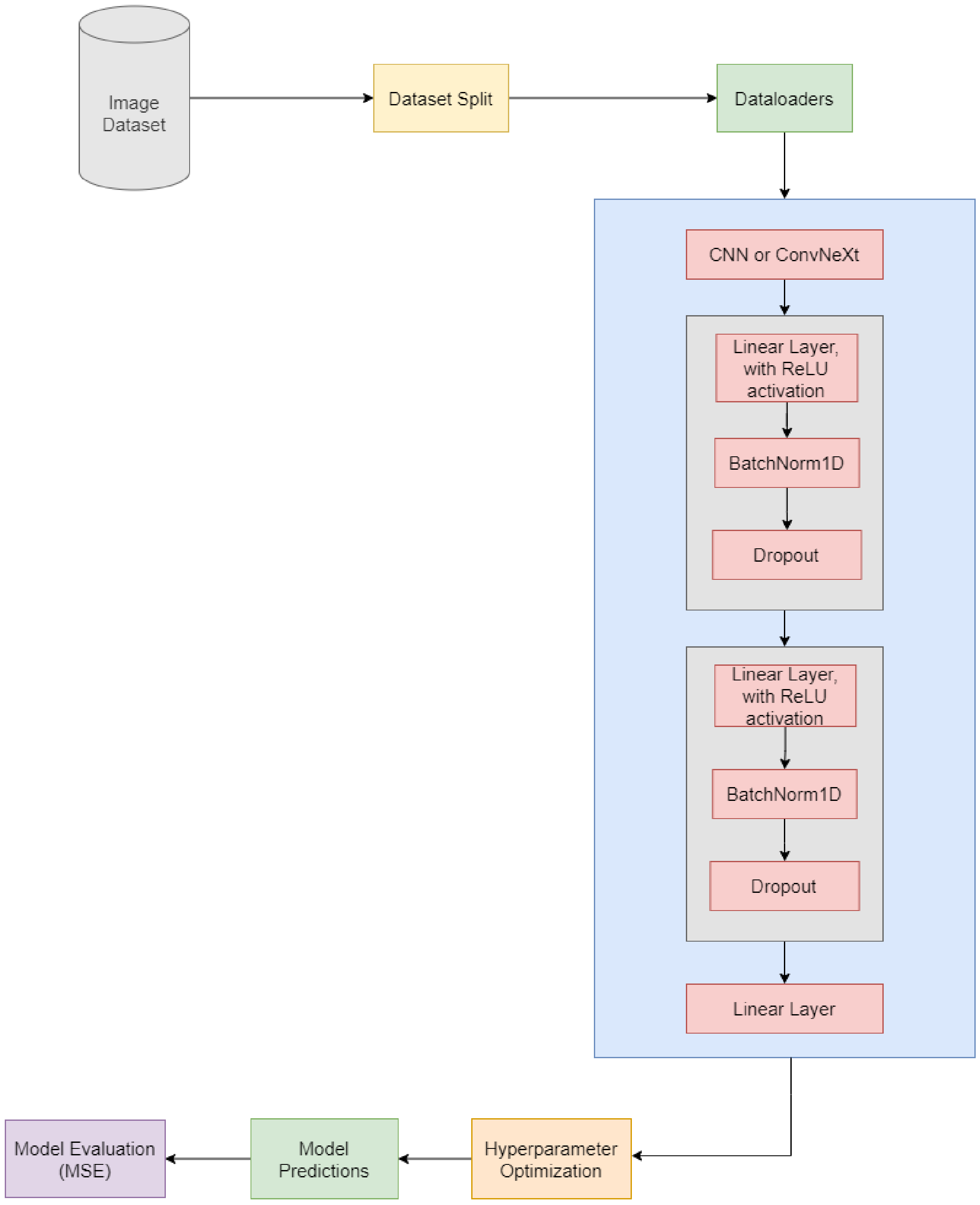
1. **5th April to 3rd May:**
    a. Understand the different model architectures being used to predict the mass of dark matter.
    b. Familiarization with the process of creating a Python package.
2. **4th May to 28th May:**
    a. Community Bonding.
    b. Analysing the suggestions of the mentors.
    c. Familiarization with the ML4SCI organization.
3. **29th May to 4th June:**
    a. Begin coding for the project and implement the required modules and sub-modules.
    b. Allow the users to pass arguments through the command line to execute the code.
4. **5th June to 11th June:**
    a. Load the gravitational lensing dataset and split it into train and valid/test datasets.
    b. Generate the training and validation/testing dataloaders to be passed to the model.
5. **12th June to 30th June:**
    a. Train the first 20 different architectures of the models.
    b. Perform hyperparameter sweeps of the above models trained, based on the input provided by the user.
6. **1st July to 14th July:**
    a. Incorporate the suggestions of the mentors into the project.
    b. Submit the midterm evaluation.
7. **15th July to 1st August:**
    a. Train the remaining models and evaluate their performance.
    b. Perform hyperparameter optimization of the neural networks trained above using WandB.
8. **2nd August to 14th August:**
    a. Save and display the results of all the trained models.
    b. Test the functionalities of the Python package.
9. **15th August to 28th August:**
    a. Address and include the changes suggested by the mentors, if any.
    b. Perform final testing of the Python package.
    c. Submit the final GSoC project.

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 to 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 architecture of the system is as follows:

1. Firstly, the dataset containing the gravitational lensing images needs to be loaded.
2. Next, the dataset is split into training and validation/testing sets. The training set is used to train the neural network, whereas the validation or testing set is used to evaluate the performance of the neural network using MSE (Mean Squared Error) as the evaluation metric.
3. The dataloaders are used to feed the gravitational lensing images into the neural network, which estimates the mass of dark matter present.
4. The neural network used to predict the mass of dark matter will make use of a CNN or ConvNeXt backbone. The network will have 7 further layers which are:
    a. Linear layer, with a ReLU activation function.
    b. 1D batch normalization layer.
    c. Dropout layer, to prevent overfitting of the neural network.
    d. Linear layer, with a ReLU activation function.
    e. 1D batch normalization layer.
    f. Dropout layer, as a form of regularization.
    g. Final linear layer consisting of 1 neuron.
5. If the performance of the neural network is to be improved different hyperparameters can be optimized. Some of the hyperparameters that can be tuned using WandB are as follows:
    a. Optimizer
    b. Fully connected layer sizes
    c. Dropout
    d. Learning Rate
6. The penultimate stage consists of estimating the mass of dark matter using the trained and optimized neural network.
7. The final step consists of evaluating the predictions made by the neural network using MSE (Mean Squared Error) as a performance metric.

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 experience in the field of deep learning and NLP is 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).