ML4SCI

Google Summer of Code Project Proposal

Super Resolution for Strong Gravitational Lensing

Contact Information :

Full Name : Om Anil Doiphode

University: Veermata Jijabai Technological Institute (VJTI) , Mumbai

Course: Bachelor of Technology (BTECH)

Branch: Computer Engineering

Email: omdoiphode161@gmail.com

oadoiphode_b21@ce.vjti.ac.in

Physical Location: Mumbai, Maharashtra ,India

Time Zone: Indian Standard Time (UTC+5.30)

GitHub

LinkedIn

Resume

To give feedback, please contact the email address.

Content

1. Overview

1.1 Project Abstract

- Strong gravitational lensing can reveal information about dark matter and its substructure.
- Deep learning can accurately identify substructure in lensing images.
- Image super-resolution techniques can enhance low-resolution lensing images, improving our understanding of mass distribution and source properties.
- The project focuses on developing deep learning-based image super-resolution techniques for gravitational lensing data.
- The models may also be used for other strong lensing tasks such as regression and lens finding.
- **RankSRGAN will be developed for the best visual quality.**
- **Various super-resolution algorithms like SRResNet, SRGAN and ESRGAN will be developed for expanding deeplense.**

1.2 Benefits to Community

1. **Improved understanding of dark matter**: By using deep learning techniques to identify and analyze substructures in strong gravitational lensing data, this project can help us better understand the nature of dark matter and its distribution in the universe.
2. **Improved accuracy of measurements**: The use of machine learning-based image super-resolution techniques can improve the resolution of low-quality gravitational lensing data, enabling more accurate measurements of the lensing effects and a better understanding of the mass distribution of the lensing galaxy and its environment.
3. **Advancement of machine learning techniques**: This project can contribute to the development of new machine learning techniques for image super-resolution, which can have broader applications in fields beyond astrophysics.
4. **Public education and outreach**: The project can help educate the general public on the importance of strong gravitational lensing and its potential to improve our understanding of the universe. Additionally, the project's results and techniques can be shared with the wider scientific community to promote collaboration and advance research in the field.

2. Goals

- Expand the DeepLense functionality with superresolution algorithms suitable for computer vision tasks applicable to strong gravitational lensing data.
- Develop a **superresolution** model for DeepLense training and inference.

3. In Depth Description

- Image super-resolution (SR) recovers high-resolution (HR) images from low-resolution (LR) images.
- SR has many real-world applications, such as medical imaging, satellite imaging, surveillance, and astronomical imaging.
- Deep learning-based SR models have achieved state-of-the-art performance on various benchmarks.
- Methods for SR include Convolutional Neural Networks (CNN) and Generative Adversarial Nets (GAN) based approaches.
- **Single image super-resolution (SISR) aims to reconstruct a high-resolution image from a single low-resolution image.**
- The relationship between LR and HR images can vary depending on the situation, including bicubic downsampling, blur, decimation, or noise.

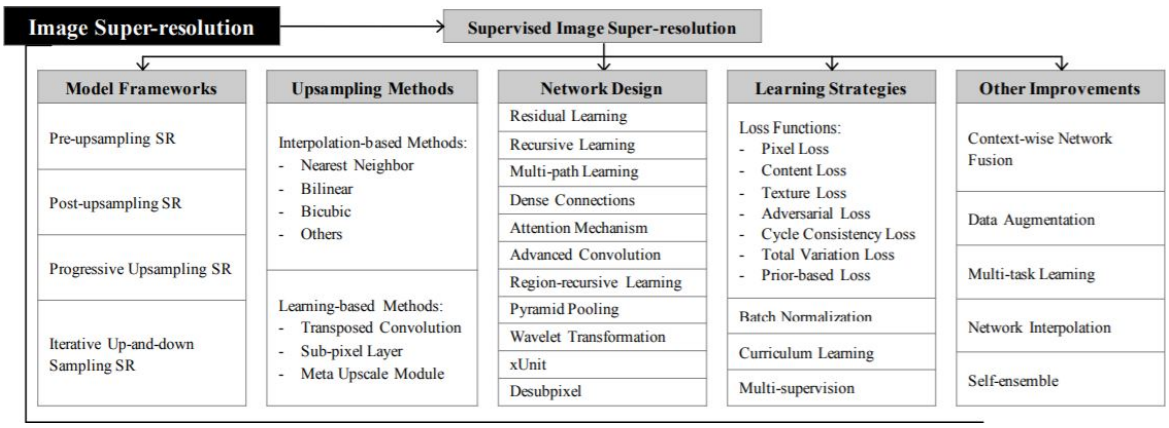


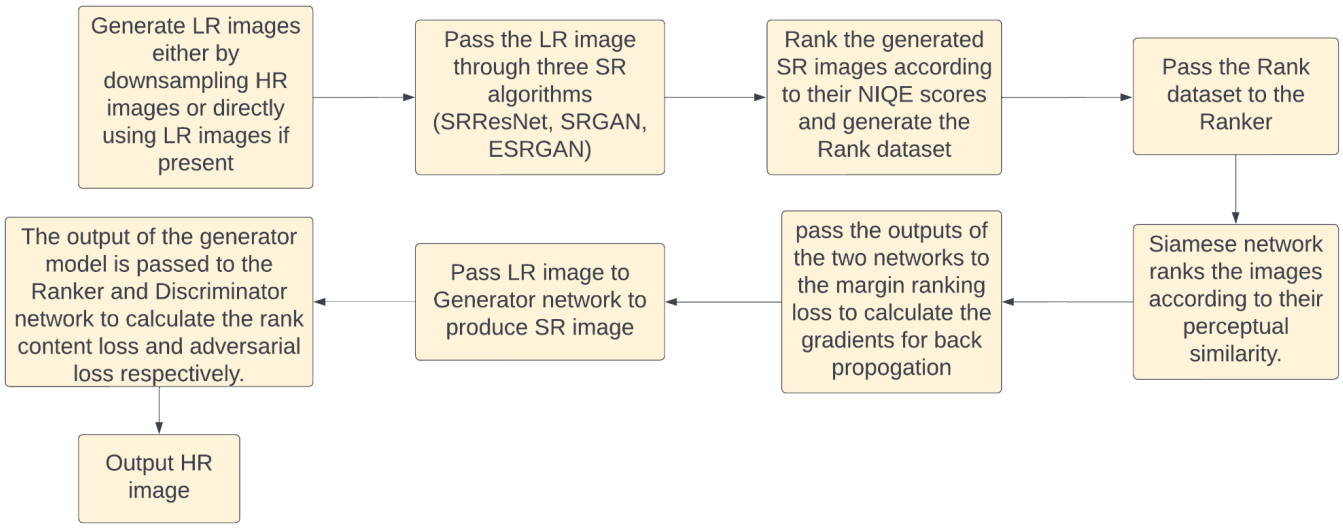
Fig. 1 Exhaustive table of topics in Supervised Image Super-Resolution

3.1 Approach followed in the specific task(Specific Test VI. Image Super-resolution)

For this task I have used **SRGAN (Super resolution Generative Adversarial Network)**. SRGAN was chosen since it produces high quality images in contrast to

prediction based methods and also it's a SOTA model. The algorithm uses a deep residual network (ResNet) with skip connections and diverges from the mean squared error (MSE) as the sole optimization target. More details can be found here.

3.2 RankSRGAN (Super-Resolution Generative Adversarial Networks with Ranker)



Working of the proposed approach

- SRGAN can provide high resolution images with good visual quality, but it may not be consistent with human ratings.
- Metrics like NIQE, PI, and Ma are used to improve super resolution quality, but they cannot be optimized by existing algorithms.
- **RankSRGAN is a new algorithm that can be used to optimize perceptual metrics.**
- **Ranker is trained to learn the perceptual metrics first, and then rank content loss is used to optimize perceptual quality.**
- **RankSRGAN improves consistency with human ratings and provides better super resolution quality.**

RankSRGAN is built upon the GAN model. There are three stages in the pipeline as shown in Fig.1:

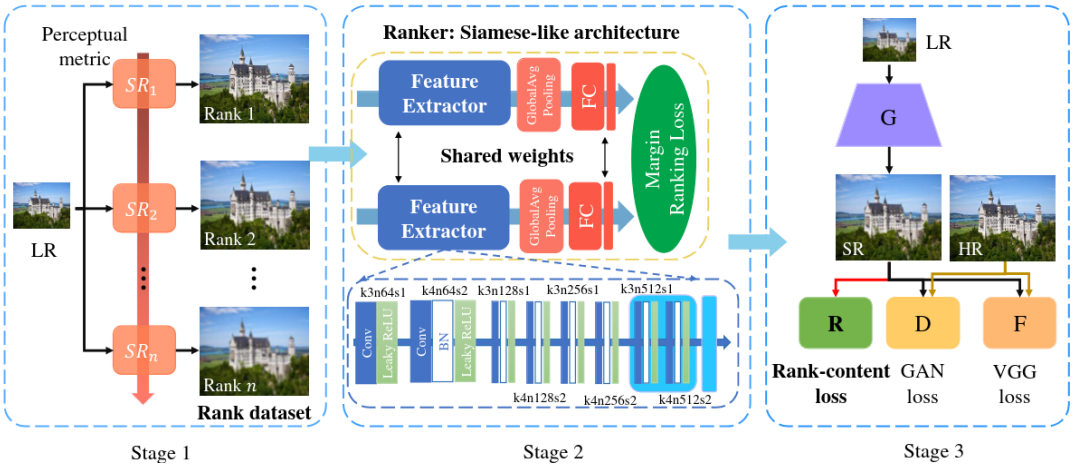


Fig.1 Overview of RankSRGAN model

Stage 1: Generate pairwise rank images (Rank set):

- Pairwise rank images are generated to train the Ranker by using different SR algorithms to generate super-resolved images on public SR datasets.
- A chosen perceptual metric, such as NIQE, is applied to the generated images to calculate quality scores.
- Pairs of patches from the same original image are ranked based on their perceived quality using the chosen perceptual metric.
- The image with the best quality score is labeled 1, the second-best is labeled 2, and the worst is labeled 3.
- These pairwise ranking labels are used to train the Ranker to learn the ranking orders of the quality of the images.
- **Three SR algorithms (SRResNet, SRGAN, ESRGAN) are used to generate super-resolved images at different perceptual levels.**
- **Patches are extracted from super-resolved images with a stride of 200 pixels and a patch size of 296 x 296 pixels.**
- The patch size can be modified based on the number of resultant patches and its effect on the model's efficiency.

Stage 2: Train Ranker

- **The Ranker uses a Siamese architecture that takes in two input images and produces a single output indicating the ranking order between the two images.**
- The architecture contains two identical branches with convolutional, pooling, and fully connected layers.

- **The outputs of the two branches are then passed to a margin-ranking loss module.**
- The margin-ranking loss is a common loss function used in "learning to rank" problems, which involves ranking a set of items based on their features.
- The Ranker is trained to rank image pairs based on their perceptual scores using a chosen perceptual metric such as NIQE.
- The goal of the Ranker is to learn to produce ranking scores that are similar to the true perceptual scores of the images.
- The Ranker architecture is based on a Siamese-like model with two identical network branches.
- The branches contain convolutional layers, LeakyReLU activation functions, pooling layers, and fully connected layers.
- A Global Average Pooling layer is used after the Feature Extractor.
- A fully connected layer is used as a regressor to quantify the rank results.
- **The Ranker does not predict the real values of the perceptual metric but ranks images based on their quality scores.**
- **The margin-ranking loss module computes the gradients and applies back-propagation to update network parameters.**

The optimization process for training the Ranker involves minimizing the margin-ranking loss function, which is commonly used in sorting problems. The margin-ranking loss function is given by:

$$L(s_1, s_2; \gamma) = \max(0, (s_1 - s_2) * \gamma + \varepsilon)$$
$$\begin{cases} \gamma = -1 & if \quad m_{y_1} < m_{y_2} \\ \gamma = 1 & if \quad m_{y_1} > m_{y_2} \end{cases},$$

During the optimization process, the aim is to minimize the margin-ranking loss function for all the N pairwise training images. Therefore, the network parameters can be updated by computing the gradients and applying backpropagation using the following formula:

$$\hat{\Theta} = \arg\min_{\Theta_R} \frac{1}{N} \sum_{i=1}^{N} L(s_1^{(i)}, s_2^{(i)}; \gamma^{(i)})$$
$$= \arg\min_{\Theta_R} \frac{1}{N} \sum_{i=1}^{N} L(R(y_1^{(i)}; \Theta_R), R(y_2^{(i)}; \Theta_R); \gamma^{(i)})$$

Here, Θ_R represents the network weights and biases, and N represents the total number of pairwise training images. The function R(.) represents the mapping function of the Ranker, and $y_1^{(i)}$ and $y_2^{(i)}$ represent the pairwise training images.

Stage 3: Rank Content Loss

After the Ranker is trained, it can be used to define a rank-content loss for a standard SRGAN. The idea behind this is to use the learned ranking ability of the Ranker to guide the training of the SRGAN in a way that maximizes perceptual quality.

$$\min_{\theta} \max_{\eta} E_{y \backsim p_{HR}} log D_{\eta}(y) + E_{y \backsim p_{LR}} log(1 - D_{\eta}(G_{\theta}(x))),$$

- The SRGAN consists of a generator Gθ and a discriminator Dη, which play a min-max game during training.
- The generator takes an LR image x and generates an HR image Gθ(x), while the discriminator takes both Gθ(x) and the ground truth HR image y and tries to differentiate between them.
- **The objective consists of two terms: the log-likelihood of the discriminator output for the real HR images y, and the log-likelihood of the discriminator output for the generated HR images Gθ(x).**
- **The generator tries to minimize this objective by generating HR images that have high probabilities of being classified as real by the discriminator, while the discriminator tries to maximize this objective by correctly classifying the real and generated HR images.**
- **The generator and discriminator are alternately updated during training until convergence.**

Perceptual Loss

It measures the similarity between the synthesized image and the ground truth image in terms of their high-level features.

$$L_P = \sum_i ||\phi(\hat{y}_i) - \phi(y_i)||_2^2,$$

- **Perceptual loss is computed using a pre-trained deep neural network, such as VGG19.**
- Feature maps are obtained by passing both the synthesized image and the ground truth image through the network.

- Euclidean distance is computed between the feature maps of the synthesized image and the ground truth image.
- Perceptual loss is the sum of the squared Euclidean distance between feature maps at each layer.
- **Using perceptual loss has been shown to improve the visual quality of synthesized images compared to using pixel-wise loss functions.**

Adversarial Loss

The generator G is trained to minimize L_G, while the discriminator D is trained to maximize L_G by distinguishing real images from generated images. The discriminator loss L_D is defined as:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

where x_i represents the HR image. The first term encourages the discriminator to correctly classify real HR images, and the second term encourages the discriminator to correctly classify generated SR images as fake.

The generator loss in adversarial training is typically defined based on the output of the discriminator, as shown in the equation below. The goal of the generator is to produce images that can fool the discriminator into thinking they are real, and minimizing this loss helps to achieve that goal.

$$L_G = -logD(G(x_i)),$$

Ranker Loss

$$L_R = sigmoid(R(G(x_i))),$$

- The rank-content loss is used to maintain the content of the original image in the generated image.
- The Ranker is used as a content evaluator to obtain the ranking score of the generated image.
- A lower ranking score indicates better perceptual quality, and hence a lower loss value is desired.
- The sigmoid function is applied to map the ranking scores to a range of 0 to 1.

- The rank-content loss is defined as the sigmoid of the ranking score of the generated image.
- By minimizing this loss, the generator is encouraged to generate images that are visually pleasing and maintain the content of the original image.

4. Timeline and Research Plan

Period	Work Expected
4.1 Application Review Period (20th March- 4th May)(On my mark)	• Getting familiar with different Super resolution algorithms like Richardson Lucy algorithm, prediction based algorithms, SRGAN, ESRGAN. • Learning about the pros and cons of the above algorithms and which algorithm can be best used for gravitational lensing, • Researching more about the concepts discussed above to get a better understanding of the project implementation.
4.2 Community Bonding Period (4th May - 28th May)(Get Set)	
Week 1 (4th May - 10th May) : Planning	• Discussing with mentors the high level details of the project (goals, priority and deliverables, meeting schedule). • After this, create a general structure of workflow of the project and adjust timeline and research plan accordingly. • Along with all this, continue learning and researching more about the things mentioned.
Week 2 (10th May - 17th May): Getting	• Prepare the working environment and

things ready	repository. ● Obtain and clean the dataset. ● Do all necessary preprocessing (normalization) in the Jupyter Notebook.
Week 3(17th May- 28th May): Bonding	● I would like to hear other students' as well as mentors' perspectives on the use of deep learning in gravitational lensing and how super resolution algorithms can be used for better understanding of dark matter.
4.3 Coding (29th May- 28th August)(GO)	
Week 1-2 (29th May- 11th June)	● Code and train SRResNet and SRGAN in Jupyter Notebook. ● Do the training several times, save the performance statistics and the models. ● These performance statistics will be used as a benchmark when developing the Rank SRGAN model. ● Make required changes to the first approach report and notebook as per feedback.
Week 3-8(12th June - 24th July)	● In week 3, we will code and train ESRGAN model. ● During this period, we'll begin building the RankSRGAN model. ● Week 4: Build Stage I of the model with inputs from the mentors. ● Week 5 and 6: Build 2nd stage of the model. This is a crucial stage as this forms the heart of the proposed model and hence mentors' input will be a valuable asset. ● Week 7 and 8: Build the 3rd stage of the model. ● A short evaluation will be submitted to the

	mentors in the second last week of this part of the project. **(Week 7)**
Week 9-10(25th July- 8th August)	• Once we have built the RankSRGAN model we will benchmark it against existing super resolution models. • Visualize the comparison, e.g., the training and testing curve for the generator and discriminator network, Perceptual loss, MSE loss, etc. • Note down the performance comparison analysis from the visualizations.This analysis will be reused as the final report's material.
Week 11(9th August - 15th August)	• Clean up the Jupyter Notebook for training. • Prepare the documentation
Week 12(16th August - 23rd August)	• Draft the final report. • Make any changes to the tutorial-like docs made in week 9 as per feedback.
4.4 Students Submit Code and Final Evaluations (24th August-4th September)	• Create the final evaluation and submit to mentors

5. About Me

I am currently a sophomore at Veermata Jijabai Institute of Technology,Mumbai pursuing Bachelor of Technology in Computer Engineering and will graduate in June 2025. During my high school days, I was highly interested in astronomical events, used to read blog posts, articles related to advancements in research in this enigmatic field. The concept of dark matter has been fascinating to me because it challenges our understanding of the universe and the laws of physics. It suggests that there is much more to the universe than what we can directly observe with our current technology and that there is still much to be discovered.

I have experience in Python, C, C++, React, Node, Express. I have also made projects in Computer Vision and Deep Learning.

I don't know if it's relevant to say here but one thing about me is that if some error or problem occurs, I try to find the solution then and there even if it takes hours to get resolved or may not get resolved at all. I am highly passionate about what I do and always try to give my 100% to the job in hand.

6. Open Source Contributions

1. **DeepForest**
   Fixed deprecation warning when running main.deepforest()
   https://github.com/weecology/DeepForest/pull/375 (merged)

2. **Ivy**
   - **Add logical_and method to PyTorch Frontend**
     https://github.com/unifyai/ivy/pull/13114 (merged)
   - **Add bitwise_left_shift method to PyTorch Frontend**
     https://github.com/unifyai/ivy/pull/13120 (merged)
   - **Add Linear Algebra operations to PyTorch Frontend**
     https://github.com/unifyai/ivy/pull/13162 (open)
   - **Add bitwise_right_shift method to PyTorch Frontend**
     https://github.com/unifyai/ivy/pull/13116 (open)

After contributing to open source for quite a long time I have a good understanding of **git**, **github workflows** and the version control system.

7. Other Commitments

I have my summer holidays during June to July and won't have any major commitments. I will devote **6** hours daily during **June to July** which I would divide into two parts: 7-8 hours for the actual project and 1-2 hours(learning time) and **3-4** hours daily during **August to September** considering my college hours.

However, I will do my best not limiting myself to work only for a few hours per day but continue working as long as I can and get the best out of this opportunity.

Considering the small amount of time allotted to this project, I would like to continue working on this project even after the end of the program(if the organization allows it) and keep contributing to this field.

Note: I have considered extended time durations on a daily basis because there can be unprecedented errors, system issues which may slow down the project work. The learning time mentioned is considered because I want this project to be a learning experience rather than just completing it as a task.

8. References

[1] RankSRGAN: Generative Adversarial Networks with Ranker for Image Super-Resolution

[2] Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

[3] Siamese Network

[4] Image Quality Assessment

[5] Quality Metrics