Machine Learning for Science - DeepLens

Superresolution for Strong Gravitational Lensing

(and a website for its use in the open-source pipeline)

Project Proposal for Google Summer of Code 2023

Personal Information

Display Name: yaashwardhan
Name: Yashwardhan Deshmukh
Email: yaashwardhan@gmail.com
Github: github.com/yaashwardhan
Country of Residence: India (UTC + 5:30)
Resume: CV_Yashwardhan_Deshmukh.pdf

Project Title

Title: Superresolution for Strong Gravitational Lensing (and a tool for easy access)
Mentors: Anna Parul, Michael Toomey, Sergei Gleyzer, Saranga Mahanta, Karthik Sachdev
Number of Tests Completed: **7** (GitHub)

Project Summary

Past research[1], confidently points out that deep learning methods successfully leverage latent spaces to learn meaningful representations from lensing data. However, due to the lack of sufficient high-resolution data, we are forced to rely on simulations for initial model training, followed by methods such as domain adaptation as more data is available, to then improve the accuracy (59% to 97% as proved in[2]). Hence, it is vital to learn an embedding space that can upscale low-resolution images to better understand the structures present in the lensing images.

For this, I will be creating and implementing a novel super-resolution architecture and using it for self-supervised learning using denoising autoencoders, following which it will be passed to a generative adversarial network as a generator for refining the perceptual quality.

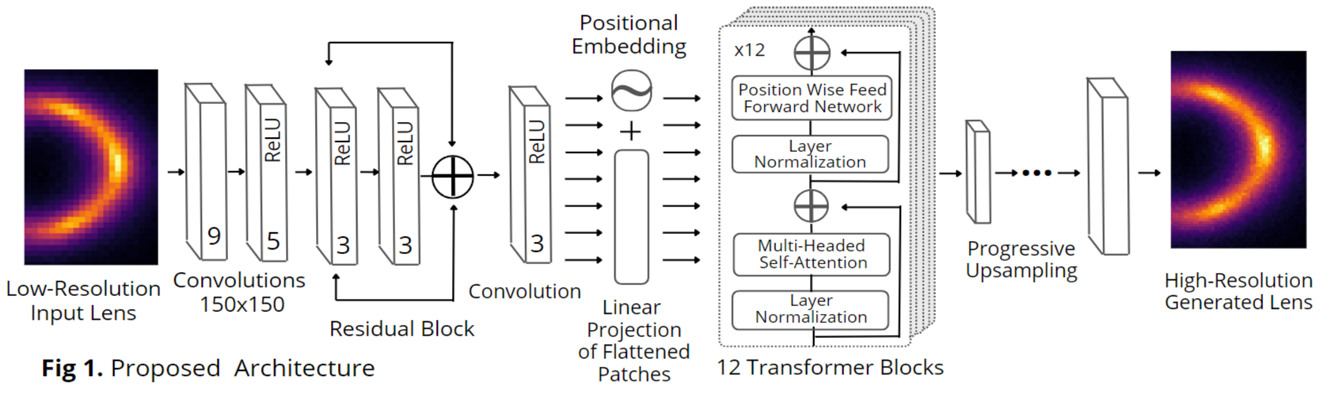


Fig 1. Proposed Architecture

[1] Alexander, S., Gleyzer, S., McDonough, E., Toomey, M. W., & Usai, E. (2020). Deep Learning the Morphology of Dark Matter Substructure. The Astrophysical Journal, 893(1), 15. doi:10.3847/1538-4357/ab7925

[2] Alexander, S., Gleyzer, S., Reddy, P., Tidball, M., & Toomey, M. W. (2021). Domain Adaptation for Simulation-Based Dark Matter Searches Using Strong Gravitational Lensing. ArXiv. /abs/2112.12121

This novel fine-tuned architecture mentioned in Fig 1. is inspired by the current state-of-the-art models, by amalgamating the strengths of the following two approaches:

I. Conventional Deep Learning methods: e.g SuperResCNN, Enhanced Deep Residual Networks, Laplacian Pyramid Super-Resolution Network
II. Generative Adversarial Networks and Revolutionary methods: e.g Enhanced Super-Resolution Generative Adversarial Networks, Vision Transformers

For e.g For our model, I will be implementing residual blocks from EDSR to extract and feed features to the vision transformer model. This will help the vision transformer model capture even more complex features and overcome the vanishing gradient problem.

This hybrid network will then be made to learn representations in an unsupervised manner using denoising autoencoders as a pretext task. The pre-trained network will be used as a generator for an adversarial network with PatchGAN and SRGAN as the discriminators. Once we have our model defined, we will tune the hyperparameters using Bayesian Optimization.

If the mentors approve, I would like to also develop a tool/website in Javascript for the model. This tool will let the user upload a .zip file of the low-resolution lensing dataset, and apply model inference to generate a corresponding dataset with high-resolution images. This will make our work even more towards the open source side as this way users without any Deep learning knowledge will be able to access our work.

Table of Contents

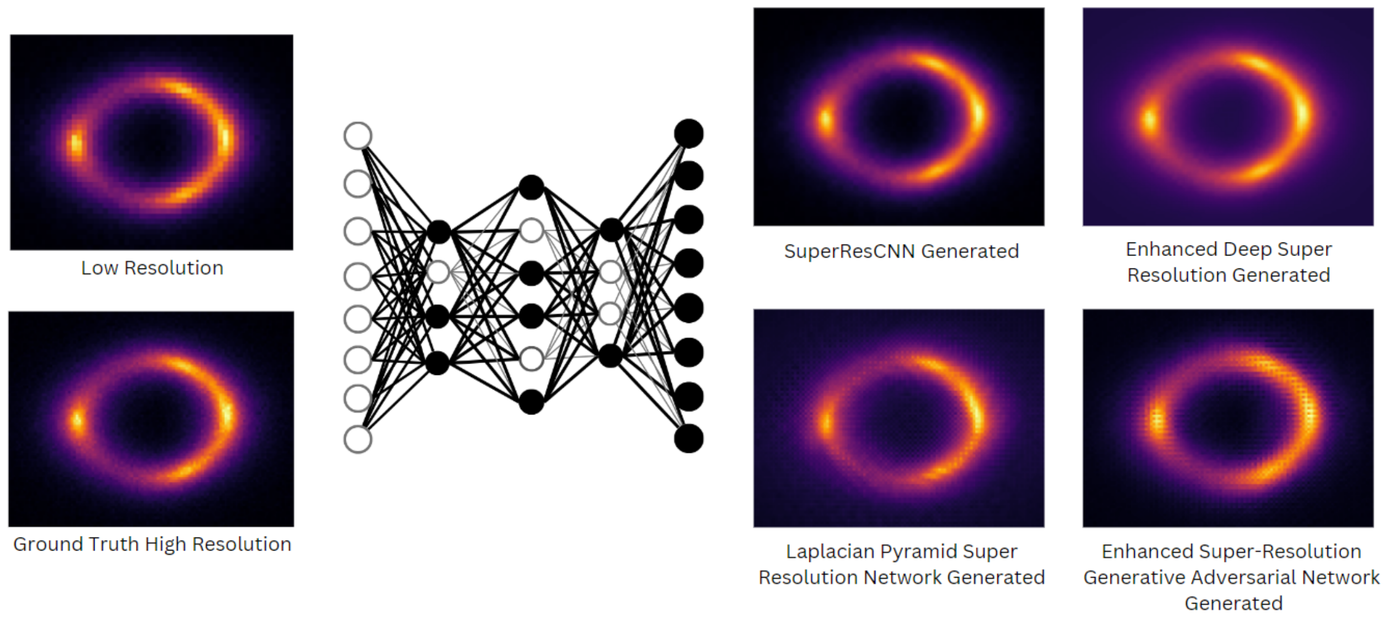
Insights Gained from Tests

I have effectively **completed 7 out of the 8 tests** (2 mandatory and 5 additional ones) for the DeepLens project with desired results. For some tests, I implemented/tested multiple approaches.

The notebooks and well-documented results (in ReadMe.md) can be found in my GitHub repo at: github.com/yaashwardhan/Evaluation-Test-DeepLense

The hosted models can be found at:
https://drive.google.com/drive/folders/1x5gm4ywOQ8brxMn11KjLBxsehQuxtx19?usp=sharing

The specific Test VI on Image SuperResolution introduced a baseline foundation to build upon during the project phase, which helped me get familiar with the project and its requirements.



Find a higher-quality image here: Results_Of_Task_VI.png

From Tests I and II on multi-label classification and lens finding, I was able to infer to channel-wise attention mechanisms that can be employed in our project for our network to selectively attend to the relevant features that contribute to high-frequency details recovered from a low-resolution input image, such as edges, textures, and fine structures, while suppressing noise or irrelevant features that might degrade the quality of the super-resolved image.

Tests III and IV introduced equivariance, a property that will lead to better generalization of our super-resolution model since the structural symmetries of input data can be preserved, such that generated high-resolution images are consistent with the structure of the input images, being less likely to cause artifacts or distortions.

Moving on, Test V demonstrates the benefits of using vision transformers which treat images as sequences of patches and learning long-range dependencies using multi-headed self-attention mechanisms. One approach we could implement from this for our case is by dividing the low-resolution image into smaller patches which are then linearly embedded into a flat vector and used as input tokens for the transformer model, which generates output tokens, that can be then reshaped and combined to form a higher-resolution image.

Another approach could be to use these vision transformers to extract features from low-resolution images, which can then be fed into a separate upsampling network. This can allow for the combination of the strengths of Vision Transformers and other architectures like CNNs.

Finally, Test VIII covers one of the most crucial aspects of deep learning, i.e using pretext tasks for self-supervised learning to learn hierarchical representations of data.

Since we already have low-resolution images and their corresponding ground truth high-resolution images, we can leverage this to create an optimal pretext task to train a denoising autoencoder that adds noise to the low-resolution images and then compresses them into a lower-dimensional representation to reconstruct cleaner low-resolution images. This helps the model learn useful features and representations that we can finetune using corresponding ground truth high-resolution images.

More methods will be discussed in the Methods and Project Deliverables section. All tests are written in TensorFlow and Keras, but I can do the same and start the project in PyTorch if needed.

About Me

I am a final-year computer science undergraduate student from NMIMS university, Mumbai, who is fueled with a passion for research and development in neural networks and computer vision.

Amidst my first year, I spent most of my time developing full-stack deep learning applications using Flutter and Tensorflow Lite and freelanced mobile phone apps for around 17 clients.

During my second year, I was accepted to visit and work as a deep-learning research intern at Max-Planck-Institut für Intelligente Systeme in Stuttgart, Germany.

At Max Planck, I built over 42 physics-informed self-attention transformer neural networks to work as a surrogate model for a finite element mesh simulation to plot human fingertip deformations for static and frequency response situations. Following this, I built a python library to create an interactive 3-D mesh plot for the neural network-generated data and embedded it into a website for public use. A paper for this is in progress and will be published soon along with the tool.

In the course of my third year, I was awarded the Mitacs Globalink Research Internship Grant, for which I spent my summer in Quebec, Canada, building neural networks for temporal sequence data collected from gyroscopic sensors installed into helmets, and implementing feedback loops for improving the safety of users.

Presently, this is the summer of my final year of university and I am looking forward with great positivity to investing my time into working on this project for DeepLens, which has fascinated me and piqued my interest to learn and improve in this domain in many ways. I have thoroughly read the past 3 papers from DeepLens and familiarized myself with the concepts involved in them.

Apart from academics, I invest some of my time into my hobbies (jamming my guitar and representing in karate tournaments) and giving back to the community through social work.

More of my past internships and projects can be found in CV_Yashwardhan_Deshmukh.pdf

Methods and Project Deliverables - A Detailed Report

In this section, we will discuss the methods in detail that I will implement during the period.

Our main goal is to expand the DeepLense functionality with single-image super-resolution algorithms suitable for computer vision tasks applicable to strong gravitational lensing data. This can be achieved by developing a superresolution model for DeepLense training and inference and then (optional part if permitted by the mentors) deploying the model on a website or a tool using TensorFlow.js or PyTorch where the user can drop the dataset with low-resolution images and receive the corresponding high-resolution dataset. We will also refer to and cite previous research in this domain and improve upon it for our custom approach. The following steps will be taken:

I. Preliminary Overview of Architectures

To create a custom architecture that is optimized for our task, we must analyze and reflect on the insights gained from architectures (SuperResCNN, EDSR, LapSRN, and ESRGAN) used in Specific Test VI. From this, we can understand the strengths and weaknesses of all architectures such that we can unify them to achieve the best from each of them.

Architecture	Strengths	Weakness
SuperResCNN [3] Super-Resolution Convolutional Neural Network	- Simple and Efficient. - Delivers satisfactory results for end-to-end mapping between low and high-resolution images.	- Struggles handling large upscaling factors to recover finer details. - Assumes a single, fixed scaling factor, making it less adaptable to varying input resolutions.
EDSR [4] Enhanced Deep Super Resolution Network	- Introduces residual connections, leading to better performance. - Recovers detailed, high-quality images with improved textures.	- Computationally expensive. - Also assumes a single, fixed scaling factor, restricting itself to varying input resolutions.
LapSRN [5] Laplacian Pyramid Super-Resolution Network	- Utilizes Laplacian pyramid for adaptable multi-scale handling. - Produces high-quality images with sharp edges and fine details.	- Computationally expensive. - Can sometimes result in unexpected artifacts or distortions.
ESRGAN [6] Enhanced Super Resolution Generative Adversarial Networks	- Produces high-quality images with impressive perceptual quality. - Recovers textures and details that are not present in low-res images. - Residual-in-Residual Dense Block for effective feature extraction.	- Training can be unstable due to the adversarial nature of GANs. - Could result in mode collapse. - Sensitive to hyperparameters. - Computationally expensive.

[3] Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Image Super-Resolution Using Deep Convolutional Networks. ArXiv. /abs/1501.00092

[4] Lim, B., Son, S., Kim, H., Nah, S., & Lee, K. M. (2017). Enhanced Deep Residual Networks for Single Image Super-Resolution. ArXiv. /abs/1707.02921

[5] Lai, W., Huang, J., Ahuja, N., & Yang, M. (2017). Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution. ArXiv. /abs/1704.03915

[6] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., & Tang, X. (2018). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. ArXiv. /abs/1809.00219

II. Developing the Custom Hybrid Super-Resolution Architecture

Steps I will implement during GSoC to develop our super-resolution architecture:

a) Define convolutional layers from SuperResCNN as a baseline to extract low-level features from the low-resolution image.

b) Implement aspects from vision transformers[7], such that the model can capture long-range dependencies in the images to preserve the visual quality by understanding the underlying image structure, patterns, and objects. In our case of gravitational lensing images, these objects can be large galaxies or lensing features such as arcs and Einstein rings. The model may even infer the lensing features and learn representations to reveal the presence of dark matter. To use this approach in our case, I will program classes/functions for:

- Diving low-resolution images into patches and embedding them into a flat vector as tokens.
- Adding positional embeddings to the patch embeddings to encode the location of each patch in the low-resolution image to maintain spatial information of these structures.
- A transformer architecture[8] class which will be programmed using the below steps:

$$Attention([Q],[K],[V]) = \frac{Softmax(([Q]*[K]^{Transpose}))}{\sqrt{key\_dimention}} * [V]$$

  - Using learned weight matrices for each attention head, we project the input token embeddings into three vectors, query (Q), key (K), and value (V).
  - We calculate the importance of tokens relative to each other in the sequence by first measuring the similarity between query (Q) and key (K) by taking their dot product. Then we will divide the dot product by the square root of the key dimension to assist gradient stabilization while the model learns. Finally, we apply softmax to get the attention scores of tokens with respect to each other.
  - Once we get the attention scores, we can calculate a weighted sum of the value (V), which will highlight the relevant tokens for our task. This could be the lens in our case.
  - Then we can concatenate the weighted value (V) vectors from all attention heads and pass it through a linear layer, to generate our multi-headed self-attention result.

- Passing the embeddings with the dummy input class token portraying the entire image into the above transformer model class for processing. After which the information-rich token embedding will be passed to the classification head to select the highest probability one.

Transfer learning using pre-trained ImageNet weights will be implemented if a considerable jump in accuracy is depicted with its use.

c) Since our network will be very deep, we might face the problem of vanishing gradients. To overcome this problem, I will employ residual blocks similar to the ones in EDSR. These blocks will contain a few convolutional layers followed by a ReLU activation function. The input to the blocks will be added to the output of the blocks. This skip connection helps learn the residual/difference between the input and the output of the blocks such that the model will identify mappings only when needed and learn complex non-linear mappings from between low and high res images.

[7] 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. /abs/2010.11929

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. ArXiv. /abs/1706.03762

d) From the results of the specific test VI, we can see that although the SuperResCNN and EDSR can generate high-resolution images, they are not able to capture the grain and the essence of the texture that is defined in the corresponding ground truth high-resolution image.

Fig. 2 depicts that LapSRN can overcome this problem at the expense of lower structure similarity and peak-to-noise ratio value. In our custom hybrid model, we will further improve this by adding progressive upsampling by implementing a series of layers of upscaling and image reconstruction operating at multiple scales and learning the difference between the upsampled image and the ground truth using the residual blocks mentioned in step (c).

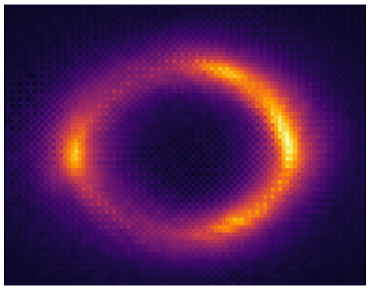


Fig. 2 LapSRN Generated

d) Generated images may lack perceptual quality. The best way to tackle this roadblock can be through generative adversarial networks used to capture high-frequency details and textures. An explanation of this approach for our model will be discussed more in detail in a later part of the proposal, especially after the self-supervised learning section.

III. Incorporating Self-Supervised Representational Learning (Denoising Autoencoders)

Self-supervised representational learning is a method in which a deep learning model learns meaningful representations from the data without any labels. It does this by solving a pretext task such as contrastive loss, image colorization, solving a jigsaw puzzle, etc. which are designed to expose the model to important patterns present in the data.

In our case of lensing images, we will use a denoising autoencoder[9] as a pretext task due to its ability to learn useful features by reconstructing noisy data into higher-resolution images. Steps I will take to implement self-supervised learning using denoising autoencoders:

- Add random Gaussian noise or other types of noises to the low-resolution data, to create noisy inputs for our autoencoder which requires an encoder and a decoder.
- For the encoder block, we can borrow the convolution layers along with a stack of the residual blocks from our hybrid architecture to capture both high and low-level features.
- Then include the developed transformer block to capture long-range dependencies.
- For the decoder block, we can borrow the hybrid architecture's progressive upsampling layers that will upsample the extracted feature maps to match the target resolution. A combination of transposed/sub-pixel convolution layers having an increasing stride will be used for refining the feature maps. Finally, a sigmoid or tanh layer to produce the outputs of the autoencoder.
- Implement a loss function that will optimize the denoising autoencoder to minimize the difference between its output and the ground truth. For this, I will implement a custom loss function which will be a combination of Mean Squared Error and Structural Similarity Index[10].

After pretraining on the pretext task, there is no need to remove the final layer for further finetuning since the final layers are already designed for the superresolution task.

[9] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning (ICML '08). Association for Computing Machinery, New York, NY, USA, 1096–1103. https://doi.org/10.1145/1390156.1390294

[10] Johnson, J., Alahi, A., Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9906. Springer, Cham. https://doi.org/10.1007/978-3-319-46475-6_43

IV. Integrating Generative Adversarial Networks

Implementing GANs after pretraining can be beneficial for our task in ways such as employing adversarial loss to produce images with high perceptual quality, that are indistinguishable from ground truth high-resolution images. GANs can recover high-frequency details and fine textures and substructures present in the lensing image that conventional methods generally fail to achieve. The generator-discriminator battle, causes GANs to be more robust to noise and artifacts.

Firstly, I will implement the two main components of the custom GAN:

- Generator and its loss function
    - For the generator, I will use our hybrid model (from section II) that has been pre-trained using self-supervised representational learning with the denoising autoencoder pretext task. The model should have learned useful features to reconstruct the high-res images.
    - Major issues GANs face are mode collapse, vanishing gradients, or unstable training dynamics. To tackle this issue, along with a content loss (perceptual loss or MSE), an adversarial loss method will be implemented and tested. Wasserstein loss**(11)** is said to be one such method, which has shown remarkable results in the past.
    - The content loss which will be computed using a pre-trained VGG-19 network, will help ensure the generated images are perceptually similar to the ground truth, while the adversarial loss will make the generated images hard to distinguish from the real ones.
- Discriminator and its loss function
    - The role of the discriminator is to classify whether the images generated by the generator are real or not. For creating our discriminator I will test two approaches.
    - First approach I will implement will be the PatchGAN**(12)** architecture, which classifies patches of images rather than the entire image. In this, I will first implement
        1. Convolutional layers with a stride for downsampling during feature extraction.
        2. I will then introduce non-linearity using LeakyReLU and increase the filters in the convolutional layers to capture complex patterns.
        3. Finally, I will employ a single neuron output layer with a sigmoid to classify a patch as real or fake.
    - For the second approach to create the discriminator, I will implement the SRGAN discriminator**(13)** architecture. In this, I will first program
        1. A series of convolution layers that have a LeakyReLU activation with the slope being 0.2 and then double the number of filters in each later after each downsampling operation.
        2. Then I will apply global average pooling followed by a dense layer with the convolutional layer's configuration.
        3. Finally, one dense output layer with sigmoid classifies the image as real or fake.
    - Loss function in both discriminator approaches will be the binary cross-entropy loss.

[11] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. ArXiv. /abs/1701.07875

[12] P. Isola, J. -Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 5967-5976, doi: 10.1109/CVPR.2017.632.

[13] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2016). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. ArXiv. /abs/1609.04802

V. Bayesian Optimization for Hyperparameter Tuning

Hyperparameters play an extremely vital role in the development of any deep learning model. When it comes to generative adversarial networks, it is extremely important to find a way to get the most optimal hyperparameters for our task such that we don't end up with problems such as mode collapse or vanishing gradients.

Bayesian Optimization is a superior method for tuning hyperparameters. It first constructs a probabilistic model such as a Gaussian Process, to approximate the objective function of the model.

It captures the relationships between the objective function and the model's hyperparameters by modeling their uncertainties. It consists of acquisition functions that search unexplored areas and search near the current best solution.

This makes it use less number of steps as compared to other methods such as grid search and random search.

For our network, I will tune the following parameters:

- In the hybrid architecture:
  - For our first CNNs, we will tune the kernel size and the number of neurons.
  - Then we will tune the number of CNNs in the residual block and their respective kernel size and the number of neurons.
  - For the vision transformer, we can tune the patch size, number of heads, and the dropout rate of each encoder block.
  - Number of layers, kernel size, and the number of filters will be tuned in the progressive upsampling block.
- In the denoising autoencoder pretext:
  - We will tune the number of layers, units, and activation functions of the encoder-decoder networks
  - We will tune the noise type by testing different types of noises such as Gaussian, uniform, or salt and pepper noise. The intensity will also be tuned for the noises.
- In the PatchGAN and SRGAN discriminator:
  - The number of convolution layers, their neurons, and kernel size will be tuned.
  - I will also test the network with and without batch normalization

VI. Developing a Website for Model Inference (If Permitted by Mentors)

Since these datasets contain a lot of images, loading deep learning models and getting inferences from them to predict corresponding high-resolution images for a set of low-resolution samples can be time-consuming and inefficient.

Hence as an additional step to the project, I want to propose building a tool/website after the model has been built such that the input dataset can be dragged and dropped into the website to automatically infer the model and create the zip file containing corresponding generated high-resolution images.

The Fig 3. Displays the website I will be implementing for the same. For this, I used Javascript along with HTML and CSS. Once the model is trained we can convert it into a JSON model using Tensorflow.js and easily implement it on the website.

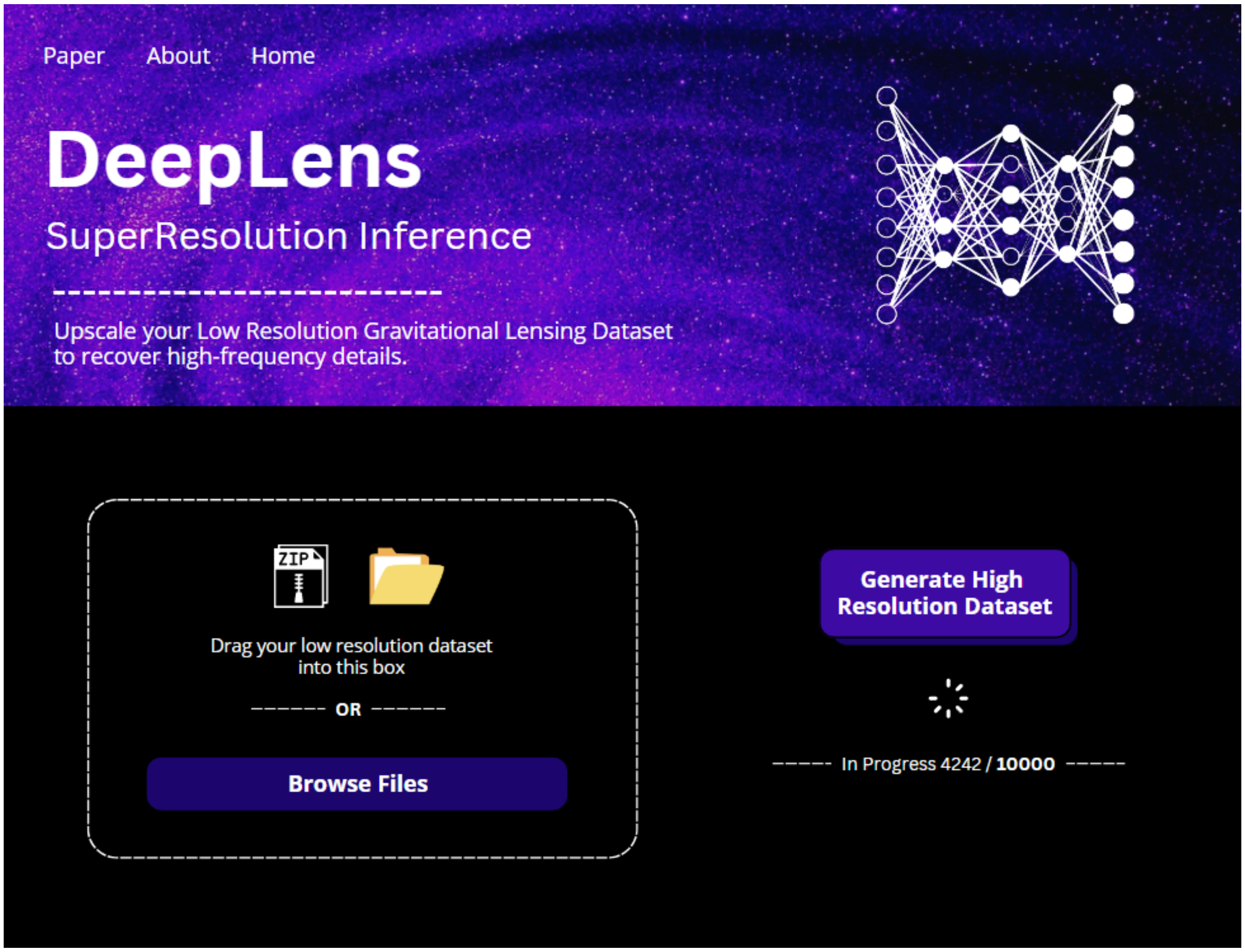
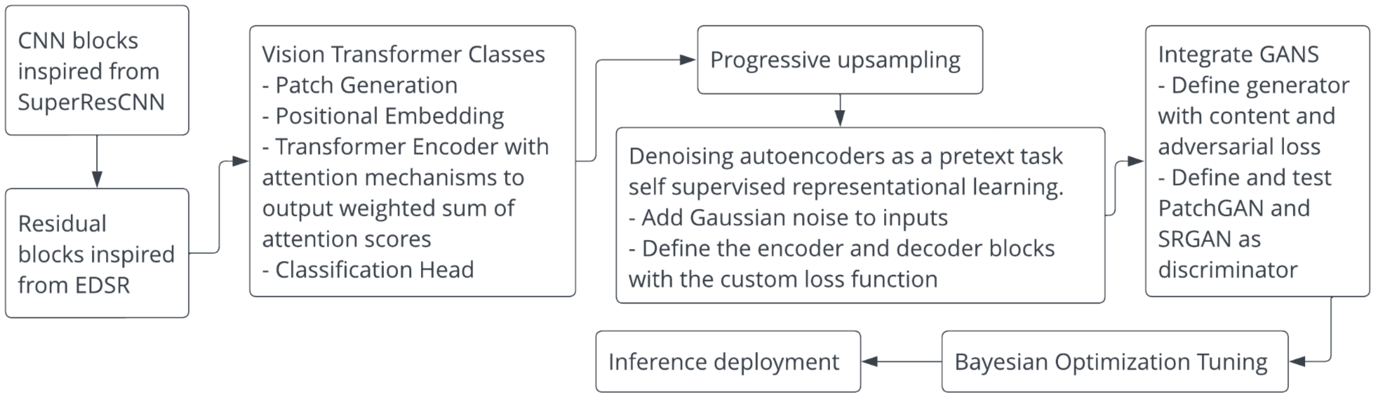

Paper
About
Home
DeepLens
SuperResolution Inference
Upscale your Low Resolution Gravitational Lensing Dataset to recover high-frequency details.
ZIP
Drag your low resolution dataset into this box
OR
Browse Files
Generate High Resolution Dataset
In Progress 4242 / 10000

Fig 3. Website UI Designed and Developed by Yashwardhan Deshmukh

Timeline with the Goals

I have divided the workflow into a period of 12 weeks followed by a final submission week, according to the GSoC schedule. I am assuming the work can be done in TensorFlow and Keras. However, if required by the mentors and the project, I can do it in PyTorch as well.
Summarized flowchart of the required work that is to be done:


CNN blocks inspired from SuperResCNN
Residual blocks inspired from EDSR
Vision Transformer Classes
- Patch Generation
- Positional Embedding
- Transformer Encoder with attention mechanisms to output weighted sum of attention scores
- Classification Head
Progressive upsampling
Denoising autoencoders as a pretext task self supervised representational learning.
- Add Gaussian noise to inputs
- Define the encoder and decoder blocks with the custom loss function
Integrate GANS
- Define generator with content and adversarial loss
- Define and test PatchGAN and SRGAN as discriminator
Bayesian Optimization Tuning
Inference deployment

★ **Community Bonding Period (May 4 - 28)**

- Get to know the project mentors and discuss the project more in-depth with them.
- Discuss with the mentors if building the inference website would be okay and accordingly adjust the schedule.
- Get familiar with the work done by previous GSoC contributors and the deep lens pipeline. Find a spot for our superresolution project in the pipeline.
- Explore all the previous research papers in super-resolution algorithms for lensing and papers that may have implemented a strategy similar to ours.
- Partially finalize the architecture we will build such that it can be adjusted later.

★ **Week 1 (May 29 - June 4)** [Convolutions and Residuals]

- Discuss if any changes are necessary for the current week's work with the mentors.
- Start implementing the unified architecture by working on convolutional blocks from SuperResCNN blocks
- Implement residual blocks and test convergence for different variations. Experiment with the Add and Convolutional layers and finalize an optimal residual block.

★ **Week 2 (June 5 - June 11)** [Extract Patches and Positional Embeddings]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Improve upon the residual structure and test different activations.
- Start implementing vision transformers by programming classes to generate patches from images and flatten them to get a linear projection.
- Add positional embeddings to encode the location of each patch.

★ **Week 3 (June 12 - June 18)** [Multi-Head Attention Logic and ImageNet Flax Extraction]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Program the class the set up the transformer’s multi-head self-attention mechanism, which will include programming the logic to project the embeddings into query, key, and value followed by taking the dot product of query and key and using *tf.matmul(weights, value)* to get the weighted average over the softmax attention scores.
- Instantiate and design an outline of a transformer block class such that we can replicate it ‘n’ number of times to generate more transformer blocks.
- Extract the Flax keys from the pre-trained imagenet model so we can appropriately name our class layers.

★ **Week 4 (June 19 - June 25)** [Transformer Block with MLP]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Continue establishing the transformer block class. Apply layer normalizations and call the multi-head self-attention class implemented in the past week.
- Add dropout and residual connections and more normalizations.
- Build the MLP feed-forward network to learn more complex representations and use the Gaussian Error Linear Unit as activation.

★ **Week 5 (June 26 - July 2)** [Correspondence class to match pretrained weights]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Develop a correspondence class that iterates over the transformer blocks and gets the corresponding names for the blocks, pre_logits, embedding layers, dummy_inputs (cls), and encoder_norm layers.
- Get a mapping of the corresponding blocks with the Flax keys extracted in week 3 from the imagenet pretrained model.

★ **Week 6 (July 3 - July 9)** [Progressive Upsampling and More Residuals]

- July 10: Start submitting midterm evaluations
- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Construct a Laplacian pyramid for progressive upsampling by iteratively generating a series of intermediate upsampled feature maps with increasing resolution.
- Add residuals to refine the image and iterate it with the upsampling multiple times. Then extract the image from the top of the laplacian pyramid. This will generate and maintain the essence of the textures present in the ground truth images.

★ **Week 7 (July 10 - July 16)** [Self-Supervised Learning using Denoising Autoencoders]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Build the encoder of the denoising autoencoder using convolutions, residuals, and the developed transformer blocks.
- Implement a method to add Gaussian and other noises to the images in the encoder.
- Formulate the decoder layer using the previously built progressive upsampling layers.
- Create the custom loss function which will be a combination of MSE and SSIM.

★ **Week 8 (July 17 - July 23)** [GAN's Generator, content loss, and adversarial loss]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Configure the hybrid model that has learned useful representations using the autoencoder to reconstruct the high-res images as the generator of the custom GAN.
- Deploy perceptual loss or MSE as the content loss and compute it using a pre-trained VGG-19 network.
- Implement Wasserstein loss as an adversarial loss to handle mode collapse and vanishing gradients.

★ **Week 9 (July 24 - July 30)** [PatchGAN or SRGAN as GAN's Discriminator]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Construct the convolutions, activations, and outputs for PatchGAN and SRGAN.
- Implement binary cross-entropy loss.
- Use the two networks as the discriminator, get results and finalize one.

★ **Week 10 (July 31 - August 6)** [Hyperparameter Tuning using Bayesian Optimization]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Set up hyperparameter tuning using Bayesian Optimization over the network. This step will take about 1 week to train and test all the parameters over my cluster since the networks are deep and strong.
- Write a script to convert the model to JSON using Tensorflow.js if Tensorflow is chosen or we can use ONNX and PyTorch Javascript if PyTorch is chosen by the mentors.

★ **Week 11 (August 7 - August 13)** [Website/Tool Development]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Establish Flask, Javascript, HTML, and CSS to develop the UI mentioned in Fig. 3.
- Write logic to parse input zip files and generate new zip files of images.

★ **Week 12 (August 14 - August 20)** [Website Model Inference]

- Get feedback about previous weeks' work from the mentors and implement it.
- Discuss if any changes are necessary for the current week's work with the mentors.
- Deploy the custom GAN model that has been trained now on the website that will convert the input low-resolution zip file dataset into a high-resolution dataset.

★ **Final Submission Week (August 21 - August 28)**

- Clean up the code and make sure all bugs are fixed
- Neatly document the entire project including all classes and functions.
- Commit all the work to the mentor's repository or submit it to them as required.

Benefits to Community

Dark matter halos are structures made of dark matter, which is a theoretical concept that does not absorb, emit or even reflect electromagnetic radiation, it is not visible. However, from strong gravitational lensing data, we can study the distortion and magnification of light emitted by distant background galaxies that pass through a massive foreground object like a dark matter halo. The gravitational field of the halo appears to distort and bend the light, causing the background galaxy to appear stretched, magnified, or even multiplied. Astronomers analyze these effects on the visible matter to infer the presence and characteristics of the dark matter halo responsible for the lensing effect. A barrier arises towards this community due to the limitations of the instruments and observing conditions such that the lensing data is usually collected at low resolution.

To overcome this our superresolution model will generate high-resolution images from a low-resolution dataset. This will make it easier to identify and characterize substructures present in the images. It will also improve mass modeling and reduce bias and uncertainties that result in more reliable data for dark matter searches.

My Eligibility Towards the Project and Why Me?

- I am eligible to work on the project from GSoC's guidelines.
- I have the necessary knowledge and skillset required to successfully complete the project.
- I have completed and submitted 7 tests required to be eligible to work on the projects.
- I believe in investing my work into open source as it benefits the community as a whole. All of my past projects are available on GitHub for public use.
- On reading past papers and projects from DeepLens, I am impressed by their work to contribute towards unlocking the secrets of the universe. I am passionate to be a part of it.
- I have previously worked in research (at Max Planck) on physics simulations and deep learning methods, which will help me transfer and implement my prior knowledge on this.
- Moreover, I wish to learn from and know more about the work of the mentors (ma'am Anna Parul, sir Michael Toomey, sir Sergei Gleyzer, sir Saranga Mahanta and sir Karthik Sachdev)

What am I Expecting out of this Project?

To be as honest as possible, I want to get the chance to implement the architecture I have proposed in this project for the DeepLens pipeline. I want to contribute more towards the project as it not only is a great open-source idea towards the science community but also a huge step towards my personal improvement in the domain. I have already invested the past month working on the tests and this proposal which has piqued my interest even more towards completing it.

If possible, I want to also publish a paper with the mentors on this project as the architecture is a new and custom approach towards superresolution of lensing images, having many modern deep learning concepts amalgamated into one, with a website that shows its effectiveness.

To conclude, even if I am not given the opportunity through GSoC, I still desire and want to work with the mentors on this project regardless (it's okay if no funding is available).