Subject: Phase Reconstruction with II using Generative Adversarial Network (GAN)

1 What we want to do

- We apply machine learning methods to the phase retrieval problem that arises in Intensity Interferometry due to square visibility.
- With II, we have sparsely sampled phase-subtracted Fourier transform of stellar objects, i.e., square visibility.
- A conditional GAN on simulated images of stellar objects is used to reconstruct the shape, size, and brightness distribution of a fast-rotating star using the signal from II.

2 Create N number of Images

Create images of stellar object

2.1 ellipse.py

- Define a function to create 2 dimensional (x, y) N number of grids
- Define a function to draw a contour map
- Define a function to plot the elliptic or other shape of a stellar object.

2.2 generate image.py

Define the range of parameters of the ellipse to plot the image with different sizes, angles, and shapes, and save them as .jpg format

- Radius; the size of the ellipsoid
- Inclination; Inclination of the ellipsoid [0, 2pi]
- Pa; Rotation around x/y-axis [0, 2pi]
- Sq; Thickness of the ellipsoid along the z-axis [0.6, 1]

3 The baselines to simulate II signal

Simulate the baselines with M number of telescope for the L number of observational period to get the signal

3.1 aperture.py

- Define the shape of the aperture for each telescope
- Introduce a function if there is any masking (Not here)
- Plot the telescopes on an observational plane for visualization of baseline with respect to a stellar object
- Define a function to create the array of baseline's name and length on XY plane

3.2 obstime.py

It defines the day of observation

- Calculate the number of steps at each observational night due to observational interval.
- Start and end time after each observational interval.
- creates an array of Julian days for a given observational interval.

3.3 obspoint.py

- Define a function to convert the given degree to radian.
- Define a function to convert the given hours to radian.
- defines a function to change the baseline with time due to earth rotation

3.4 generate baseline.py

The generated variational baseline with time will be useful to simulate the signal for any source and telescope's array

- Define the array of starting times for the observational day
- Define the array of ending times for the observational day
- Calculate the number of steps for a given observational interval (average time) so the array of Julian time
- Define the telescope position name and get the baseline's name and length
- Define the position of source Polaris (assumed) in the sky coordinate
- Calculate the variational baseline according to the earth's rotation
- Plot the covered (u,v) plane with baselines in a .png file for visualization
- Convert the observational plane to grayscale, re-arrange pixels, and save the baseline in .npy format

4 Create input data for GAN

It creates the input (II signal) for the GAN using the baselines for each ellipsoid image (i.e., a stellar source which is ground truth). It saves it for training, testing, and validating folders containing merged images (stellar objects in the sky and signals on the observational plane). These three folders act as datasets for GAN.

4.1 image noise.py

- Define a function that adds both images. The ellipsoid (i.e., a stellar source) and II signal for this ellipsoid using the baselines
- Add salt and paper noise to the ellipsoid image so that it creates noisy II data for input to GAN

4.2 generate pspec.py

- Stellar objects (ellipsoid's image) are the ground truth
- Calculate the power spectrum (the absolute value of Fourier transform) of each noised image using salt and paper noise def in "image noise.py."
- Calculate the visibility for given baselines (power spectrum x baseline)
- Normalise the signal (i.e. visibility)
- Add both ground truth and signal side by side on an image using add image def in "image noise.py."
- Save the images in the train, test, and validation folders

5 Train the signal with GAN

It is time to create the Generator and Discriminator network along with other necessary definitions to train the model for image reconstruction.

5.1 main def.py

This script prepares the set of definitions used for the GAN structure while training.

- There is a function that breaks the image and creates the input (signal) and ground truth (the image of a stellar object)
- Define a function to resize both the signal and ground truth to change the shape of an array without changing its data using the Nearest-neighbor interpolation method.

- Define a function to crop both the signal and ground truth to the image size.
- Define a function to normalize the images to [-1, 1]. This normalization is done to ensure that the input data has a mean of 0 and a standard deviation of 1, which can help improve the performance of machine learning models.
- Define a function to Jitter the images.
- Define a function to load the concatenated image files from the train folder and separate them, then Jitter (flipping half images as phase reconstruction exists in visibility) and normalize them.
- Define a function to load the concatenated image files from the test folder and separate, then resize, and normalize them (for the testing model, flipping is not done as the testing will be done).
- Define a function to add salt and pepper noise to an image using tf.where(). Extra values (from 0 to 1) are added in elements where conditions apply in tf.where()
- Define a function to calculate the difference in two input signal
- Define a function to scale down the network
- Define a function to scale up the network

5.2 main def test.py

This script tests the definition used for the GAN structure while training in "main def.py."

- There is a baseline on the observational plane (in terms of npy)
- There is a source in the sky (in terms of npy)
- The Fourier transform of ground truth (the image of a stellar object) is the visibility signal on an observational plane
- Multiplication of this signal with baseline returns the observed signal for the telescopes
- Check the separation of the added image is working with the "load" function
- Check that random Jitter is working so that we can get plus and minus visibility both
- Salt and paper noise is working or not
- Downsampling reduces the size of the image
- Upsampling increases the size of the image

5.3 gan def.py

This script describes the GAN network as a generator and Discriminator. Using these networks, train the model

- Define the Generator with different layers of neurons using upsampling and downsampling.
- Define the loss function for the Generator where it calculates the loss in output of the DiscriminatorDiscriminator after not being fooled and the loss in target and Generator output
- Define the Discriminator with different layers of neurons that take input signal and target as input and return the predicted result based on inputs
- Define the loss function for DiscriminatorDiscriminator, which tries to prove that the generated result from the Generator is wrong
- Define a function to check prediction or its 180 rotated form is matching with ground truth

Now, enough definition is prepared and ready for the image generation using the input signal, which will be checked as the predicted image or its rotation is closer to the target image ("generate images").

- Using a Generator, the model creates images
- It uses input signal and ground truth (target), then
- Produce predicted image or its 180 rotation based on matching with Ground Truth
- It also shows the difference between ground truth and predicted image (generated from the Generator using input signal) after image generation.

Define a function that trains the Generator and Discriminator

- There will be a model for the Generator and its output
- Use gradient and then optimizer of it to reduce the loss of Generator
- There will be another model for DiscriminatorDiscriminator and its output
- Use gradient and then optimizer of DiscriminatorDiscriminator as well to reduce the loss of DiscriminatorDiscriminator

Write a function to train the model using training data with a test data set

- unpack the test dataset, which has signal and ground truth
- test the Generator using this
- unpacks the training dataset, which also has signal and ground truth
- train the Generator and Discriminator using this to give the best result

5.4 gan def test.py

Test the script “gan def.py” here

- Define the parameters to train the Generator and Discriminator
- Load the image to be trained in float32 tensor
- Get the structure of the Generator according to the neuron using plot.model
- Test the Generator to generate an image using input signal (visibility)
- Get the structure of the Discriminator according to the neuron
- Test the Discriminator to predict the image (result) based on the signal and generated image as input while keeping training=False.
- Plots four images based on the model using def “generate image”: Input, Ground truth, Prediction, and (optionally) difference in the predicted image and ground truth (if show diff = True).
- Load image for training and testing
- Train the model using def “fit”
- Test the model using def “generate image.”