CSC 580 Final Project

Bohan Li

May 2020

Introduction

The study of celestial bodies surrounding stars is crucial to the understanding of star systems distant from the solar system. However, images of such objects often contain high amounts of starlight noise from the corresponding star. A common approach to resolving this issue is to employ point spread function (PSF) subtraction, which involves using a set of reference images to construct an estimate of the star, which is then subtracted from a target image containing a faint signal. The most widely accepted method of performing PSF subtraction is the Karhunen-Loeve Image Projection (KLIP) algorithm [2], which computes an eigenbasis of the reference set using the eigenvectors of the covariance matrix of the images. This algorithm has many open-source implementations available; here, the focus of the analysis will be on PynPoint [3], one such implementation of the KLIP algorithm written in python. PynPoint offers minor improvements on the algorithm, such as the implementation of pixel masking to filter out known points of poor quality data, but does not differ in the method of the eigenbasis construction that defines KLIP. However, the unsophisticated method by which KLIP computes the basis for PSF subtraction suggests that there is potential room for improvement. A recent publication of the Spectral Inference Networks (SpIN) algorithm [1] revealed a deep learning approach to computing eigenfunctions of linear operators. The publication showed promising results of a next-frame predictor of various Atari video games using successor features, a similar application to computing an estimate of a set of reference images. Here, a preliminary analysis of the applications of SpIN towards computations of eigenfunctions and PSF subtraction is provided.

Methods and Research Process

The initial goal of this project was to use the eigenfunctions computed from SpIN directly in the PynPoint algorithm, comparing results of the PynPoint implementation, which utilizes PCA in computing the eigenfunctions, to that of SpIN. This approach was based on the assumption that the SpIN algorithm was simply an eigenfunction computer, as the algorithm was presented so in Pfau et al [1]. However, further examination of the provided example usages and the source code of the algorithm revealed that SpIN was far more powerful than initially thought, capable of providing solutions to a variety of mathematical and machine learning problems due to its high degree of customizability. In Pfau et al., the algorithm is presented as having three parameters, the linear operator, decay rates, and the first order optimizer. However, the example source code applications reveal that the user is given a higher degree of freedom than simply toggling three variables. The SpIN library contains three examples, an bare-minimum end-to-end example, an example of the solutions to the Schrodinger Equation modeling a hydrogen atom, and the example of the next-frame predictor for the Atari game. In all three examples, the user specifies the convergence criteria, implemented as the number of iterations, the sampling method of the data at each iteration, and the network structure. A summary of the parameters for each example is given in table 1.

To develop an understanding for the various results that these examples output, a single dataset drawn from the PynPoint library, the beta pic M' set, was loaded into the end-to-end and Atari examples. Since hydrogen required inputs of points in space, this was not compatible with the image stack data and was thus not considered. The PynPoint algorithm was also run using the example data, with eigenvectors plotted in figure 1. For the minimum example, the problem was formulated as each pixel having a feature vector represented by its value in each reference image. The first 20 eigenfunctions are shown in figure 2. For the

Example	Linear Operator	Optimizer	Network Structure	Sampling Schema
Minimum	Kernel	Adam	2-layer neural network	Random feature vectors
Hydrogen	Hamiltonian	RMSProp	4-layer neural network	Random points in space
Atari	Slowness	RMSProp	3-layer convolutional neural network	Random sequential frames in episode

Table 1: Parameters for each example

Atari example, the image stack was treated as if it were a set of video data from a game. Figure 3 shows results, with average frames in each eigenfunction shown on the left and eigenfunction features on the right. Pfau et al. notes that noticeable differences between the left and right images indicates that learning is occurring.

Discussion and Conclusion

The differences in the example eigenfunction images shows that the eigenfunctions computed represent solutions to very different problem sets, and therefore are not comparable. This shows the ability and limitations of the SpIN algorithm, which requires careful definition of a machine learning problem to use. It prevents the simple integration of the PynPoint and SnIP algorithms, since there exists a large degree of freedom when choosing the correct implementation of the SnIP algorithm. Extensions of this project could involve the construction of such machine learning problems, in which the output of SnIP is is tied closely with some aspect of the PSF subtraction algorithm. For example, rather than taking eigenfunctions of the covariance matrix as in KLIP, SnIP could potentially apply some sort of transform to the image set and retrieve eigenfunctions from that.

References

[1] David Pfau, Stig Petersen, Ashish Agarwal, David G. T. Barrett, and Kimberly L. Stachenfeld. Spectral inference networks: Unifying deep and spectral learning, 2018.

[2] Rémi Soummer, Laurent Pueyo, and James Larkin. Detection and characterization of exoplanets and disks using projections on karhunen-loève eigenimages. *The Astrophysical Journal Letters*, 755(2):L28, 2012.

[3] T. Stolker, M. J. Bonse, S. P. Quanz, A. Amara, G. Cugno, A. J. Bohn, and A. Boehle. PynPoint: a modular pipeline architecture for processing and analysis of high-contrast imaging data. , 621:A59, January 2019.

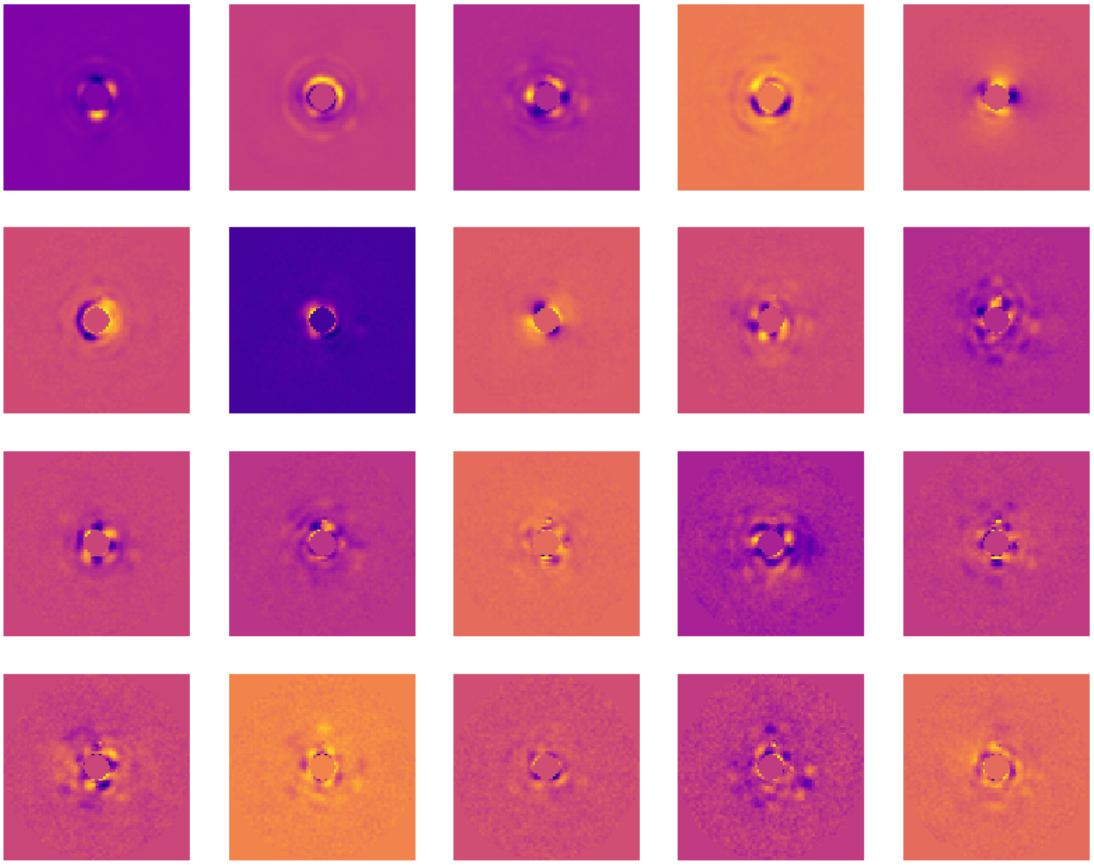

Figure 1: First 20 eigenvalues of beta pic dataset with PynPoint PCA approach

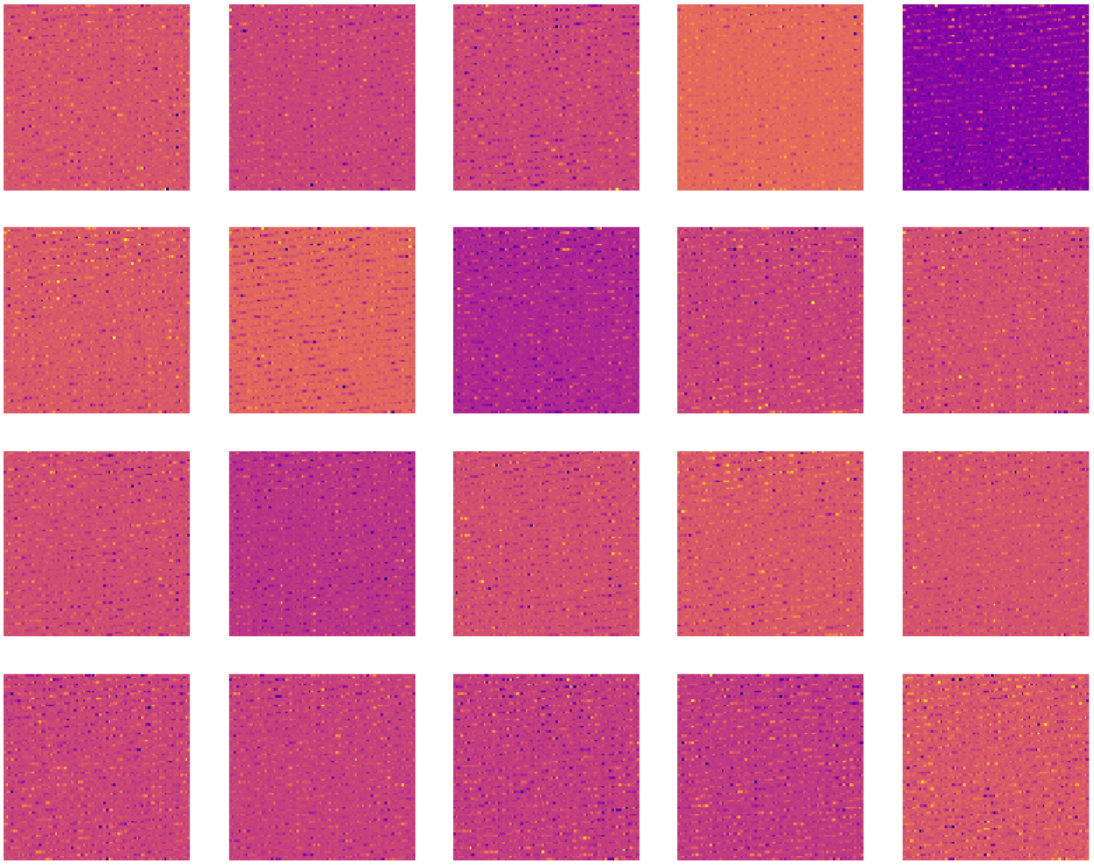

Figure 2: First 20 eigenvalues of beta pic dataset run using SpIN minimum end-to-end example code

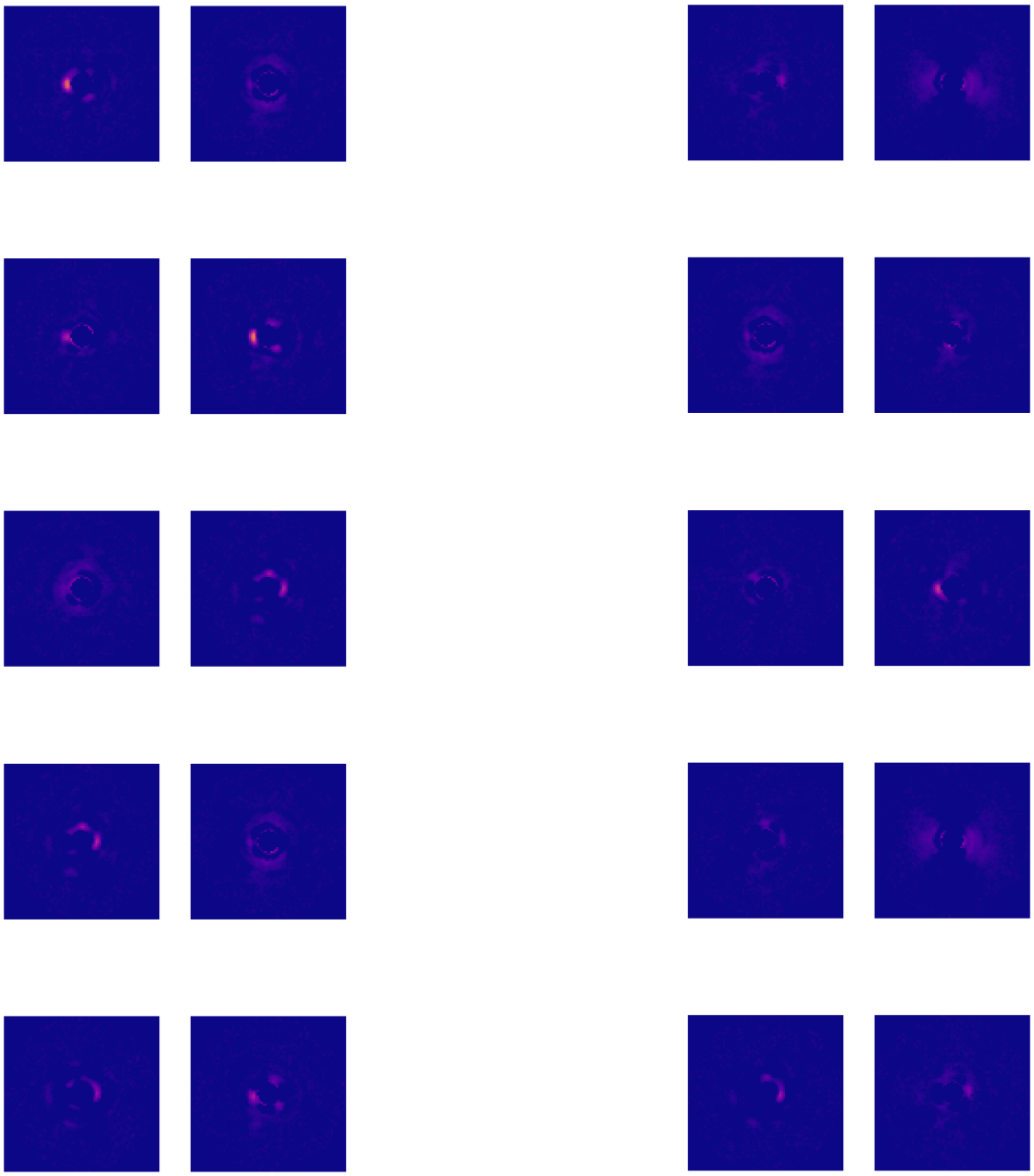

Figure 3: First 10 eigenfunctions of Atari example: average of 100 frames with greatest magnitude (left), eigenfunction encoded features (right)