**Machine Learning for   
Classifying Spiral and Elliptical Galaxies**

# Introduction

With the development of the technologies, astrophysics is exploring the night sky with an unprecedented speed. Larger CCDs are taking pictures at a faster rate, generating enormous amount of data. The Vera Rubin Observatory’s Legacy Survey of Space and Time (LSST), for example, which is just built and will soon be running, will generate about 20 TB data per night, containing millions of galaxies. Detecting and classifying the objects captured by the telescopes is one of the major topics. It is obviously no longer a realistic to analyze the data by human being solely and computer algorithm must take a major role. Advanced machine learning (ML) techniques, particularly convolutional neural networks (CNNs), with the ability to learn complex patterns that is difficult to be hard-coded, are well-suited for this purpose with an exceptional efficiency and precision.

This project is trying to use different ML techniques to classify spiral and elliptical galaxies, which are two of the most frequently found types of galaxies in the universe. Although it's not a particularly interesting task, it serves a good starting point for applying machine learning algorithm in classifying astronomical objects.

Spiral galaxies are…… elliptical galaxies are ……

In this project, I will firstly try a clustering algorithm to see how well the algorithms can tell the differences between the two types. And then try one shallow MLP and CNN to see how well they can candle the task. For capturing the delicate differences between the two types of galaxies, a deep CNN is supposed to be the best model. I will try a deep CNN model and fine tune it to see what would be the best result I can get.

# Loading and demonstrating the data

Explore the properties of the data, do some preprocessing if necessary.

# Unsupervised clustering analysis

Use Gaussian mixture to see

# Try classifying with shallow NN, both MLP and CNN

# Build a deeper CNN and see how much the performance is improved.

# Finetune the model and see how the performance is affected.

Things to try:

* Different optimizers
* Different activations
* Try dropout to limit overfitting