**Project Breakdown**

**Task 1 – Game of Life (GOL)**

* Contains two files
  + GOL.py
  + GOLTests.py
* GOL.py contains three important methods
  + returnNeighbors(arr, row, column) takes the GOL board into account along with the row and column of a particular cell in the board. It then returns the value of all the cells surrounding that cell
  + returnNextGen(arr) takes a GOL board into account and returns the state of the board after one generation is simulated. This method is used when creating the Keras model in part 2
  + playGame(generations, fileName) takes into account the number of generations to simulate in addition to the pickle filename where the GOL board is stored. It then simulates the GOL board for the number of generations passed in and returns the resulting board
* GOLTests.py has 10 unit tests that test various scenarios
  + They test different board sizes, including square and rectangular sizes
  + They test different unique cases such as the glider pattern, and several static patterns (board configurations that don’t change no matter how many simulations are run).
  + All 10 unit tests were passed

**Task 2 – Keras Model**

* Directory Structure
  + GOL.py (Same file from Task 1, included in this folder as a dependency)
  + GOLModelTest.py
  + model.h5
  + model.json
* GOLModelTest.py includes a method that loads in the keras model from the last 2 files and returns a prediction on whether two inputted 7x7 GOL boards are sequential or not
* GOLNeuralNet.ipynb contains the training code that was used to train the Keras model to detect if two GOL frames are sequential
* model.h5 contains all the weights for the artificial neural network (ANN) that was trained to detect whether or not two 7x7 GOL boards are sequential
* model.json contains information about the layers of the keras ANN and the different activation functions
* Model Information
  + The model was trained in Keras and is an Artificial Neural Network
  + The ANN contains three layers (1 input layer, 1 output layer, 1 hidden layer)
  + The first two layers use the RELU activation function to account for nonlinearity in the training data and the last layer uses the sigmoid activation function to actually classify the training data
  + It inputs two 7x7 GOL boards and outputs 0 if they’re sequential and 1 if they’re not sequential
* Training the Model
  + To obtain the training data, I generated 3000 random 7x7 GOL boards
  + I then used my GOL.py file to return the next frame of all these GOL boards and labelled this sequence of boards as sequential frames
  + For the non-sequential frames, I generated 3000 random 7x7 GOL boards. Then for each GOL board I generated another random 7x7 GOL board and made sure that the newly generated board is not sequential to the initially generated board. I labelled this sequence of boards as nonsequential frames
  + The dataset had 3000 sequences that were sequential and 3000 sequences that were not sequential
  + I then split this set into a training set and a test set with 80% of the data being used for training data
  + After training the neural net on the training data, I compiled the metrics for both the training data and test data. Metrics involved accuracy, precision, and recall in order to provide a holistic review on the performance of the model. The results are below.

A screenshot of a cell phone

Description automatically generated