Galaxy Morphology Classification Using Convolutional Neural Networks

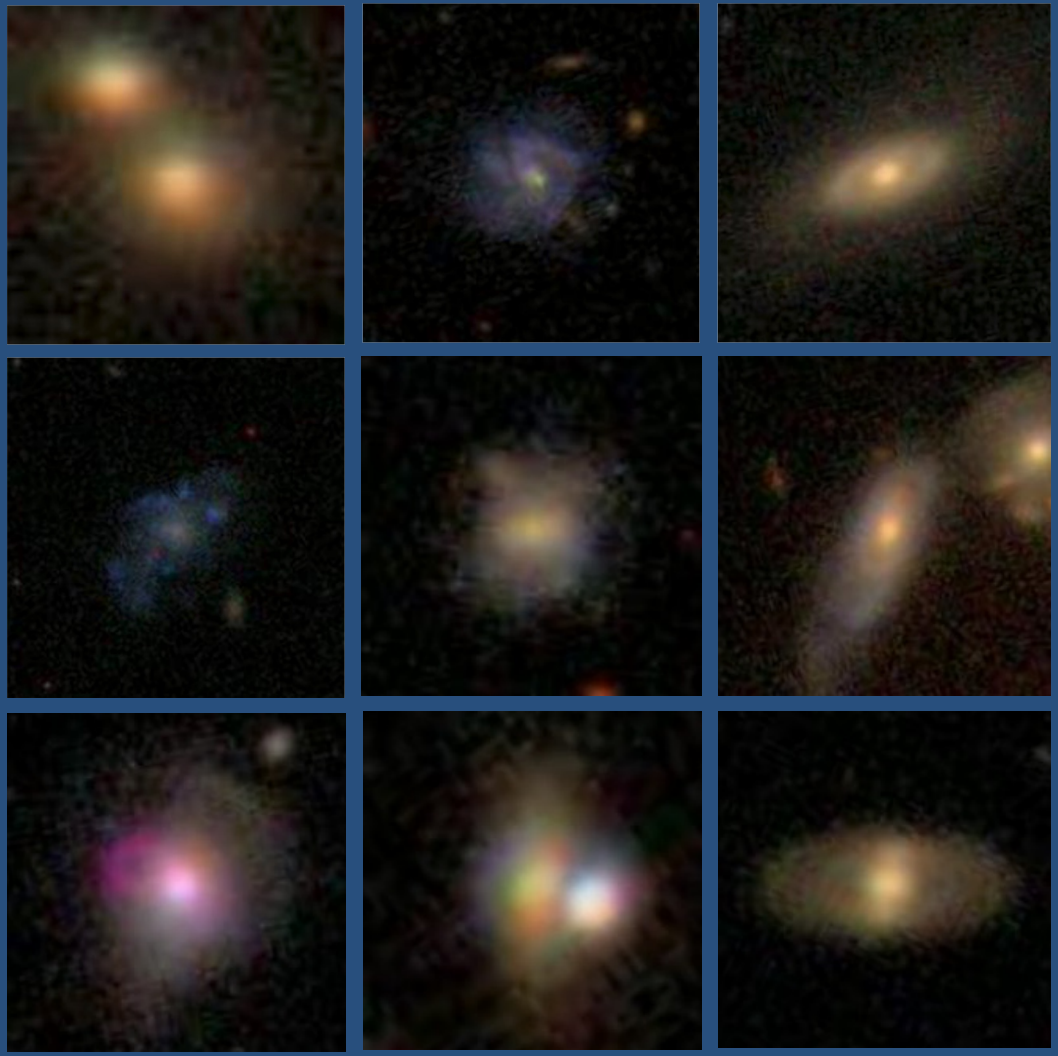
COMP-4990 Group 36

April 8th, 2022

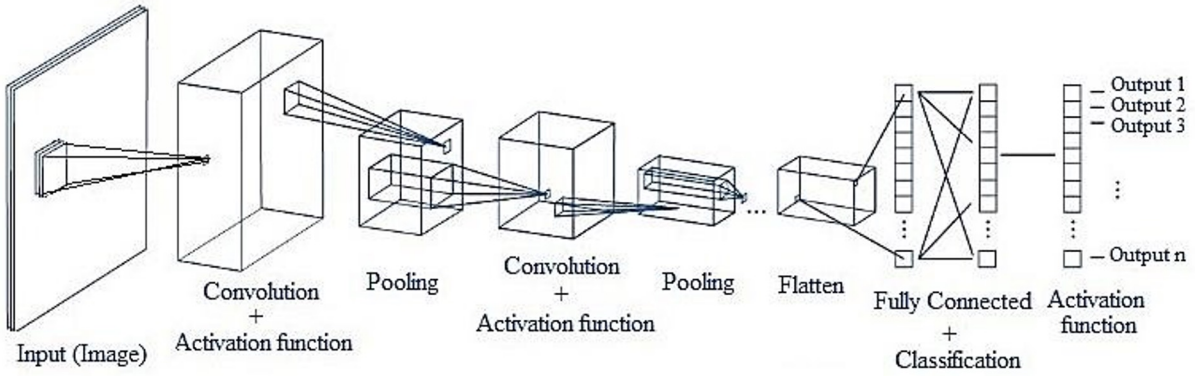
Overview

Visual galaxy morphologies are used by astronomers to study the dynamical structure of the systems for about 90 years. A wide range of morphological features provide information about the history of the host systems.

Our project is aiming for automatically classification of the galaxy images based on their morphological odd features–ring, irregular, merger, and other features.



Convolutional Neural Network



CNN ALGOTHIRM

```
# My CNN model
model = Sequential()
#, kernel_initializer=tf.keras.initializers.HeNormal()
model.add(Conv2D(input_shape=(SIZE[0],SIZE[1],3),filters=224,kernel_size=(3,3), activation="relu", padding="same"))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
#model.add(Dropout(0.25))

model.add(Conv2D(filters=32, kernel_size=(3,3), activation="relu", ))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
#model.add(Dropout(0.15))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation="relu"))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
#model.add(Dropout(0.2))

model.add(Conv2D(filters=192, kernel_size=(3,3), activation="relu"))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(5,5), activation="relu"))
#model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(units=256,activation="relu"))
#model.add(Dropout(0.2))
#model.add(Dense(units=128,activation="relu"))
#model.add(Dropout(0.2))
model.add(Dense(units=4, activation="softmax"))
```

We built a Convolutional Neural Network with

- 5 convolutional layers each followed by a polling layer
- Flatten layer
- 2 dense layers

Technologies

Programing Language
Python

Platform/API/Library
Tensorflow, keras
numpy, pandas, sklearn

Google Colab
Connect to and compute on GPU

PyQt5 + Qt Designer
Implement user interface

Dataset

Train/Test Split

```
from sklearn.model_selection import train_test_split

labels_train, labels_test = train_test_split(labels, test_size=.2)
labels_test.to_csv('test_true.csv', index=False)

print('Split traning labels: ')
labels_train.shape, labels_test.shape
```

```
Split traning labels:
((3499, 2), (875, 2))
```

4374 images

- Training: 3499
- Validation: 872

GalaxyID	Category
490849	2
320131	4
426761	4
743187	1
158247	4
...	...
658975	2
249105	2
938675	2
536807	4
243024	1

GalaxyID	Category_1	Category_2	Category_3	Category_4
490849	0	1	0	0
320131	0	0	0	1
426761	0	0	0	1
743187	1	0	0	0
158247	0	0	0	1
...	...	...	...	...
658975	0	1	0	0
249105	0	1	0	0
938675	0	1	0	0
536807	0	0	0	1
243024	1	0	0	0

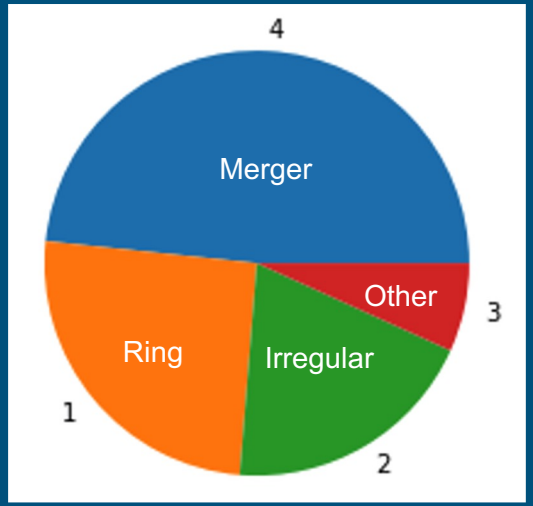
One_hot Encoding

```
one_hot_train = pd.get_dummies(labels_train, columns = ['Category'])
one_hot_test = pd.get_dummies(labels_test, columns = ['Category'])
```

Preprocess the data

- Original size: 424*424
- After resized: 224*224
- img = tf.image.resize_with_crop_or_pad(img,

  SIZE[0], SIZE[1])



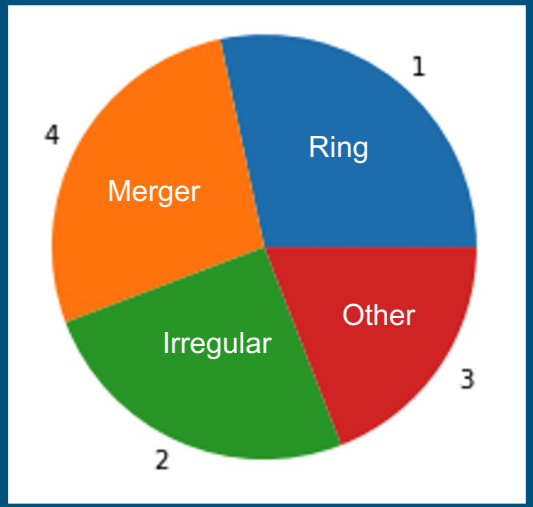


Image data augmentation

To solve the data imbalance problem, rotate image randomly to generate new images.

```
def rotate_image(img_path, angle):
  img = plt.imread(img_path)
  img = imutils.rotate(img, angle)
  img = tf.image.resize_with_crop_or_pad(img, SIZE[0], SIZE[1])
  img = img/255
  return img

def rotate_images(labels):
  data = labels.values
  img_ids = data[:,0].astype(int).astype(str)

  for i in tqdm(img_ids):
    angle = 90*random.randint(1, 4)+random.randint(-20, 20)
    img = rotate_image('/content/input/training/'+i+'.jpg', angle)
    train_imgs.append(img)

  images = train_imgs
  return images
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 224)	6272
max_pooling2d (MaxPooling2D)	(None, 112, 112, 224)	0
conv2d_1 (Conv2D)	(None, 110, 110, 32)	64544
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 32)	0
conv2d_2 (Conv2D)	(None, 53, 53, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
conv2d_3 (Conv2D)	(None, 24, 24, 192)	110784
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 192)	0
conv2d_4 (Conv2D)	(None, 8, 8, 128)	614528
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dense_1 (Dense)	(None, 4)	1028

Total params: 1,340,196
Trainable params: 1,340,196

Build CNN Model

KerasTuner is used for hyperparameter search.

- Conv2D filters: min_value=32, max_value=256, step=32
- Kernel: 3*3 or 5*5
- Number of conv layers: 2-5
- Number of dense layers: 2-3

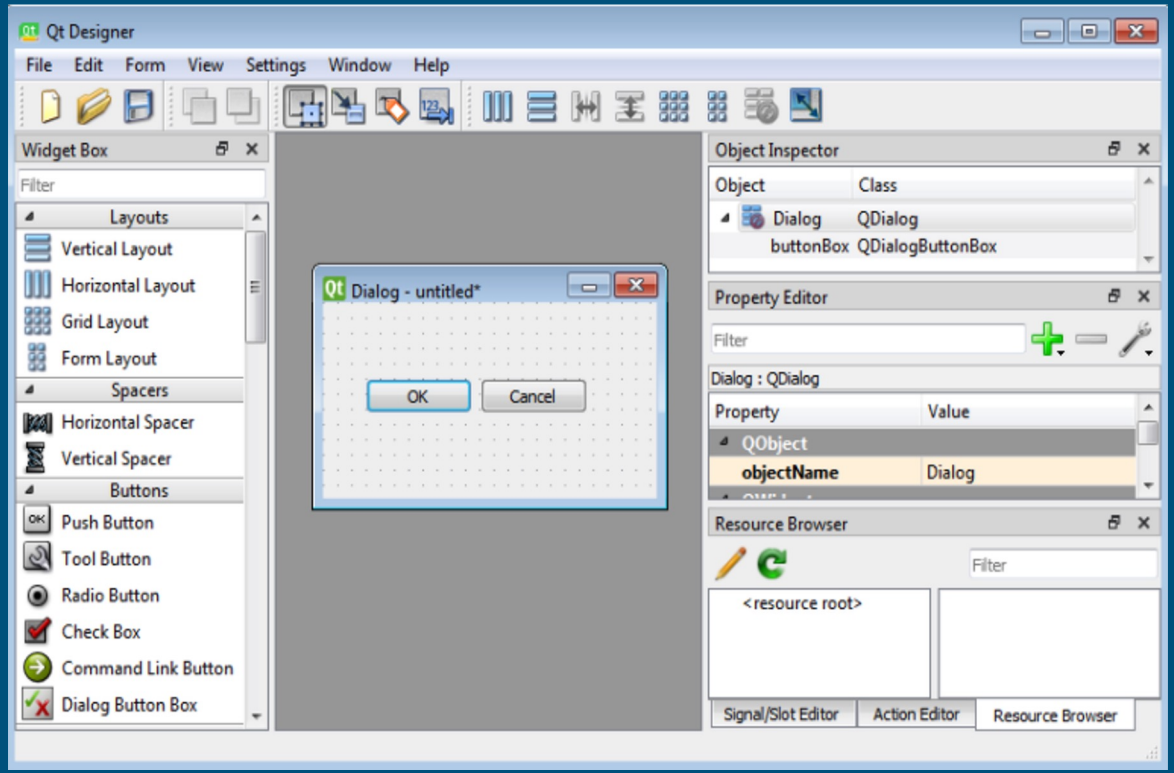
```
Epoch 21/25
191/191 [==============================] - 57s 299ms/step - loss: 0.4832 - precision: 0.8622 - val_loss: 0.6316 - val_precision: 0.8497
Epoch 22/25
191/191 [==============================] - 58s 301ms/step - loss: 0.4627 - precision: 0.8671 - val_loss: 0.6111 - val_precision: 0.8483
Epoch 23/25
191/191 [==============================] - 57s 300ms/step - loss: 0.4652 - precision: 0.8668 - val_loss: 0.6639 - val_precision: 0.8348
Epoch 24/25
191/191 [==============================] - 58s 301ms/step - loss: 0.4519 - precision: 0.8713 - val_loss: 0.6072 - val_precision: 0.8369
Epoch 25/25
191/191 [==============================] - 57s 300ms/step - loss: 0.4215 - precision: 0.8848 - val_loss: 0.6211 - val_precision: 0.8480
```

```
datagen = ImageDataGenerator(
            featurewise_center = False,
            samplewise_center  = False,
            featurewise_std_normalization = False,
            samplewise_std_normalization  = False,
            zca_whitening = False,
            rotation_range = 20,
            width_shift_range  = 0.2,
            height_shift_range = 0.2,
            horizontal_flip = True,
            vertical_flip = False)

datagen.fit(train_imgs)
```

Train the model

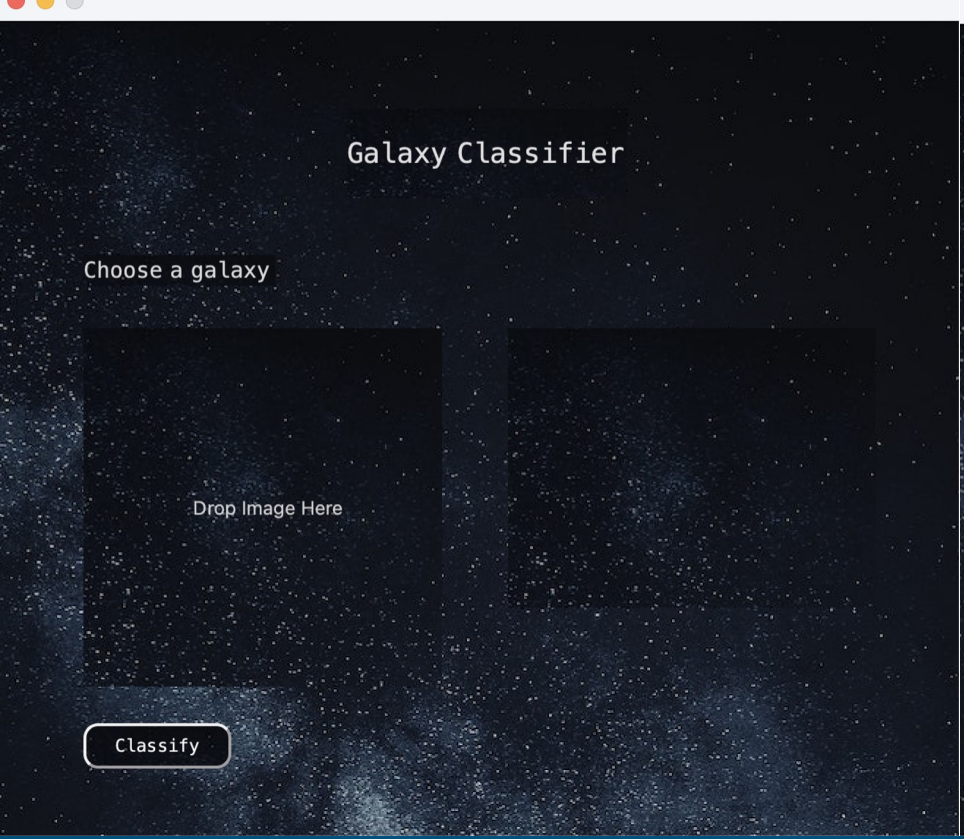
- Randomly shift, rotate or flip the images during training
- Validation precision reaches 84.97%

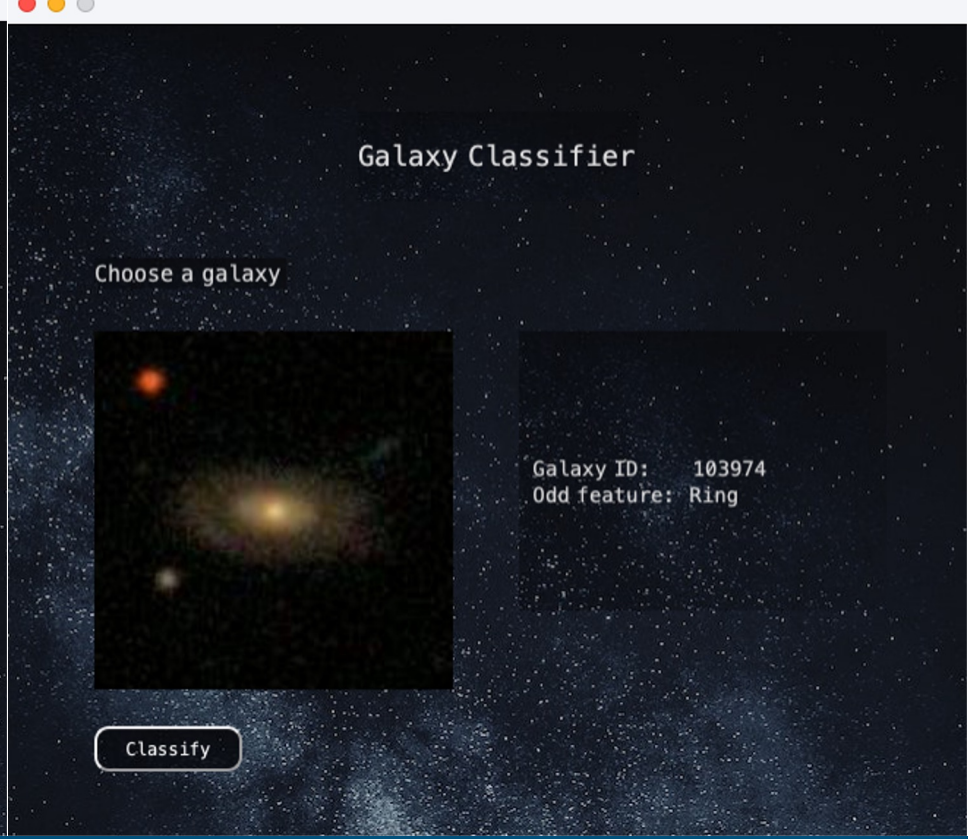
```python
def classify(self):
    model = load_model("galaxyClassifier-model.h5")
    categories = {
        0: "Ring",
        1: "Irregular",
        2: "Other",
        3: "Merger"
    }

    SIZE = (224, 224)

    def preprocess_image(path):
        img = plt.imread(path)
        img = tf.image.resize_with_crop_or_pad(img, SIZE[0], SIZE[1])
        img = img / 255
        return img

    image = []
    img = preprocess_image(self.img_path)
    image.append(img)
    image = np.array(image)

    pred = model.predict(image)
    prediction = np.array(pred)
    prediction = np.argmax(prediction, axis=1)

    self.result.setText("\n  Galaxy ID:      "+self.img_path[-10:-4]+
                        "\n  Odd feature:  " + categories[prediction[0]])
```



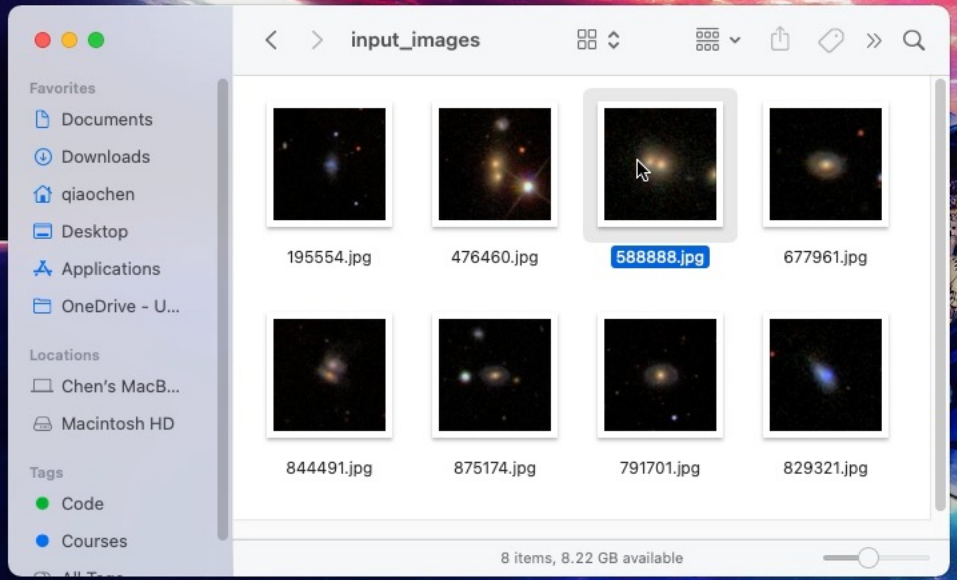
Design User Interface

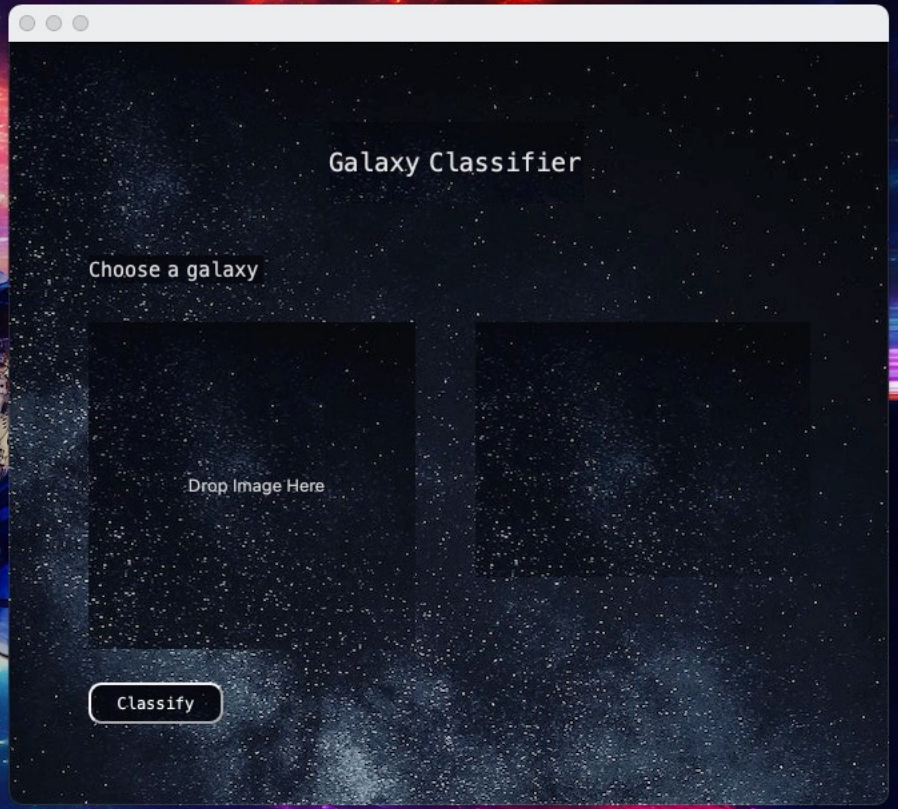
Qt designer and PyQt5 are used to design the basic user interface





User Interface


input_images
Favorites
Documents
Downloads
qiaochen
Desktop
Applications
OneDrive - U...
Locations
Chen's MacB...
Macintosh HD
Tags
Code
Courses
195554.jpg
476460.jpg
588888.jpg
677961.jpg
844491.jpg
875174.jpg
791701.jpg
829321.jpg
8 items, 8.22 GB available


Galaxy Classifier
Choose a galaxy
Drop Image Here
Classify

Reference

- Melanie R Beck, Claudia Scarlata, Lucy F Fortson, Chris J Lintott, B D Simmons, Melanie A Galloway, Kyle W Willett, Hugh Dickinson, Karen L Masters, Philip J Marshall, Darryl Wright. Integrating human and machine intelligence in galaxy morphology classification tasks. Monthly Notices of the Royal Astronomical Society, Volume 476, Issue 4, June 2018, Pages 5516–5534. https://academic.oup.com/mnras/article/476/4/5516/4923080

- M. H UERTAS -C OMPANY, R. G RAVET, G. C ABRERA -V IVES, P.G.P ÉREZ -G ONZÁLEZ, J. S. K ARTALTEPE, G. BARRO, M. B ERNARDI, S. M EI, F. S HANKAR, P. D IMAURO, E.F. B ELL, D. KOCEVSKI, D. C. KOO, S. M. FABER, D. H. M CINTOSH. A catalog of visual-like morphologies in the 5 CANDELS fields using deep-learning. The Astrophysical Journal Supplement Series, Volume 221, Issue 1, article id. 8, 23 pp. (2015)

- Jason Brownlee. 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. Machine Learning Mastery(2015). https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Thank you !