PRIMORDIAL SPACE

SPACE: THE FINAL FRONTIER

AIDI2004 - 01
FINAL PROJECT REPORT

GROUP 13:
HIBBA IMTIAZ (100794061)
FAIZAN ALI (100518916)

Table of Contents

1. Introduction

The aim of Primordial Space is to help people learn more about space by using the power of AI. With this application we aim to allow the user to upload a picture of a galaxy and have the AI identify what type of galaxy it is and teach them about that type of galaxy.

2. Problem Statement and Rationale

For centuries people have been interested in what lies beyond the little blue ball we call home, and finally, we have the technology to start venturing into what lies beyond. But due to the sheer size of the infinitely expanding space and the number of new discoveries being made, we need some way to easily classify objects in space, while at the same time make it easy for people to learn more about what lies beyond.

3. Data Source and Acquisition

Our data was acquired from Galaxy10 [1] dataset that we got from astroNN library of python. This dataset is accumulated from two other datasets Galaxy Zoo [3] and Sloan Digital Sky Survey (SDSS) [2] dataset. It is an alternative to MNIST and CIFAR10 as a deep learning model for astronomers.

Galaxy Zoo dataset contains labeled images of over 90,000 galaxy images and SDSS dataset contains HD images of all types of galaxies.

The dataset contains 21785 images and 10 labels that are associated with images.

The images in the dataset were downscaled to 69x69 from the original size of 207x207, using bilinear interpolation. The dataset is not balanced, as it can be seen in Figure 2, we only have 15 images for Class 5 type of galaxy.

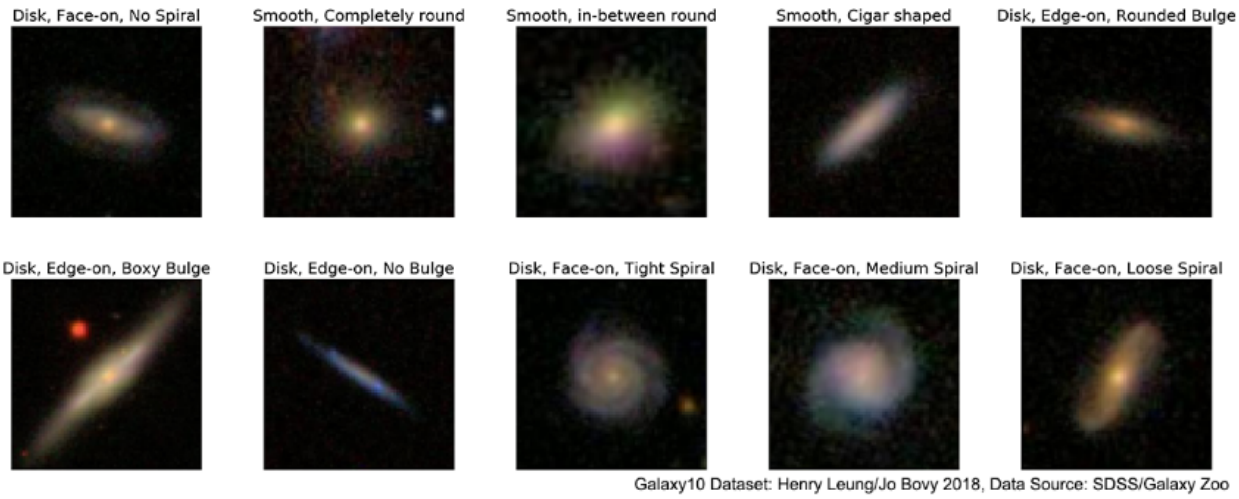


Figure 1: Images in the galaxy10 dataset

```
Galaxy10 dataset (21785 images)
├── Class 0 (3461 images): Disk, Face-on, No Spiral
├── Class 1 (6997 images): Smooth, Completely round
├── Class 2 (6292 images): Smooth, in-between round
├── Class 3 (394 images): Smooth, Cigar shaped
├── Class 4 (1534 images): Disk, Edge-on, Rounded Bulge
├── Class 5 (17 images): Disk, Edge-on, Boxy Bulge
├── Class 6 (589 images): Disk, Edge-on, No Bulge
├── Class 7 (1121 images): Disk, Face-on, Tight Spiral
├── Class 8 (906 images): Disk, Face-on, Medium Spiral
└── Class 9 (519 images): Disk, Face-on, Loose Spiral
```

Figure 2: Labels associated with the images

4. Model selection

A custom model was trained instead of using a pre-trained one as it allowed us to have greater control over the model architecture and hyperparameters. We experimented with different models to find the one that best fits the dataset. The selected model is a sequential CNN model with maxpooling and softmax activation function.

5. Exploratory Data Analysis

We conducted an extensive EDA on the Galaxy10 dataset, following are our findings. As noted previously, there are 21785 images and each image is of 69x69 dimension with three RGB channels. Corresponding to the images, there are 21785 labels. The images are split into 10 unique classes (Figure 2).

During the analysis we found that there were no missing values or corrupted images in the dataset and all the images are unique. Next, we examined the distribution of the galaxy types.

We noticed that we have a higher distribution of Class 1 and 2 type galaxies (Figure 3(a)) and way less for Class 3, 5 and 9. Handling this discrepancy is out of our project scope, hence it is reflected in our model as well.

When exploring the images, we noticed that all the images are of the same size, that is to say, they have the same height and width and all images are in RGB. Next step was to inspect the RGB distribution of the images, shown in figure 3(b). Since the images are mostly black with the galaxy as the bright yellowish spot, the RGB value distribution is concentrated on one side more. After examining the RGB value distribution of images,

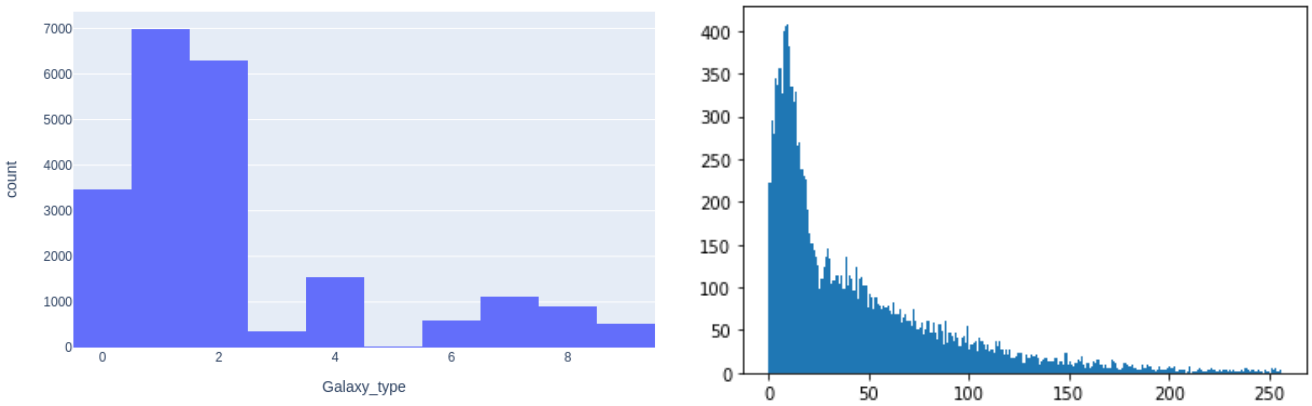


Figure 3 (a): Distribution of the galaxy types, (b) RGB distribution of an image from the dataset

we noticed that the distribution was almost identical for all the images, this is due to the fact that the images have been cropped in such a way that only the galaxy is the visible object. We got similar results when BGR distribution was analyzed (Appendix A).

Further, we inspected the contrast to get the sharpness of the images. An image with good contrast has its histogram spread across the graph. Figure 4 shows the pixel distribution over the cdf normalization, we see that the pixel values are concentrated within one region of the graph with high values, which tells us that the image is bright and the contrast is not good. This can be solved using histogram normalization.

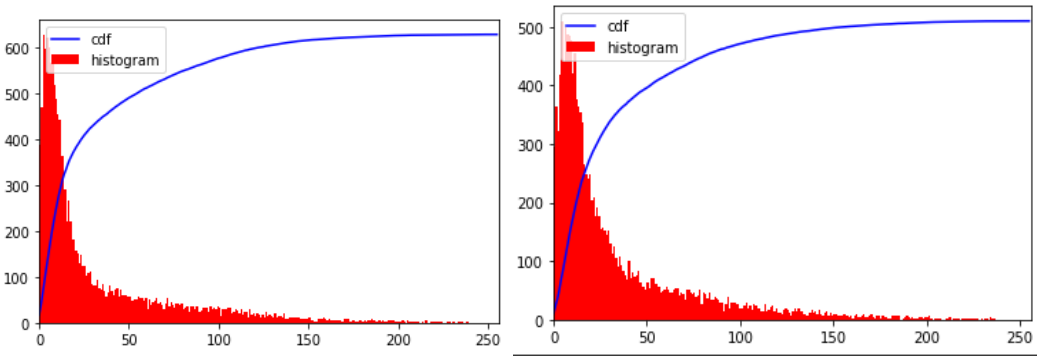


Figure 4: Distribution of pixel values over cdf normalization

6. Data Pre-processing

After analyzing the dataset in detail, the next step was to preprocess the dataset. We first one-hot encoded the labels and normalized the images. Afterwards, we split the labels into y variable and images in X variable (Figure 5).

Since the data is already clean, we split it into training and testing dataset and prepare for model training.

```
1 X

array([[[[16., 18., 15.],
         [18., 20., 15.],
         [16., 17., 12.],
         ...,
         [ 6.,  6.,  4.],
         [ 9.,  9.,  7.],
         [ 6.,  6.,  4.]],
```

```
1 y

array([[0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 1.],
       [0., 0., 1., ..., 0., 0., 0.]], dtype=float32)
```

Figure 5: (a) labels as y, (b) images as X

7. Model Training

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 69, 69, 64)        1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 69, 69, 64)        36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 17, 17, 256)       295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 17, 17, 256)       590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 17, 17, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 8, 8, 512)         1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 4, 4, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              8392704
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970
=================================================================
Total params: 39,929,674
Trainable params: 39,929,674
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Model summary of the selected model

VGG16 model:

The model was made using the VGG16 model as reference consisting of 13 Convolutional layers and 3 max pooling layers. This model performed better than the other CNN based models we compared and gave us an accuracy of 70% on the validation dataset.

It took about 2 hours to trained the model using GPU enabled Tensorflow 2.4

8. Model Prediction

The model takes in an image, with a minimum size of 69x69, of a galaxy as an input. We perform the same preprocessing steps to the model as was done to the model training images before feeding it into our predictor. The class with the highest probability predicted by the model is then returned.

9. Flask-Web App

The model predictor was integrated into the front end web application using the Flask framework (Figure 7).

```
app = Flask(__name__)

@app.route('/')
def home():
    #return render_template('MapGUI.html')
    return render_template('HomePage.html')

@app.route('/predictGalaxy', methods = ['GET','POST'])
def process_image():
    if request.method == 'POST':
        f = request.files['file']
        location = "static/img/upload/"+f.filename
        f.save(os.path.join('static/img/upload', secure_filename(f.filename)))
        output=prediction.predict(location)
        data = {'location': location, 'predict': output}
        return jsonify(data), 200
      # return jsonify({'status':'OK','location':'found'}) ,200

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Figure 7: Flask app integrated with trained model

The HTML web template was rendered using flask. The web application contains 3 different elements (Figure 8):

1. The homepage for users to learn more about primordial space
2. Information on the different galaxies
3. Prediction page that allows the user to upload their own picture of a galaxy and have a prediction made on it.

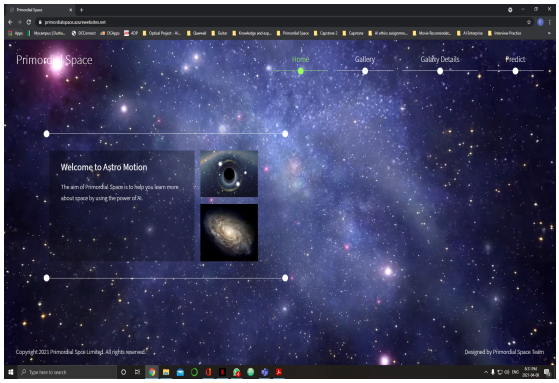


Figure 8(a): Website

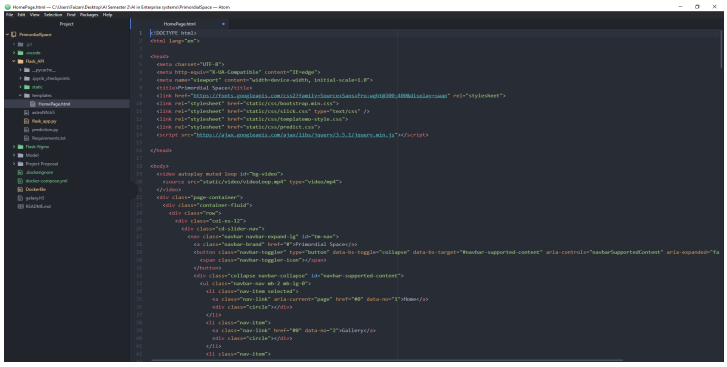


Figure8(b): Web app template code

10. Deployment - Docker/Azure

An image of the application was made using docker and then using that image the application was put into a docker container.

The image was uploaded onto Microsoft Azure Container registry, from where it was then integrated into the Azure Web App using the container built from the image. The application is hosted live at: https://primordialspace.azurewebsites.net/

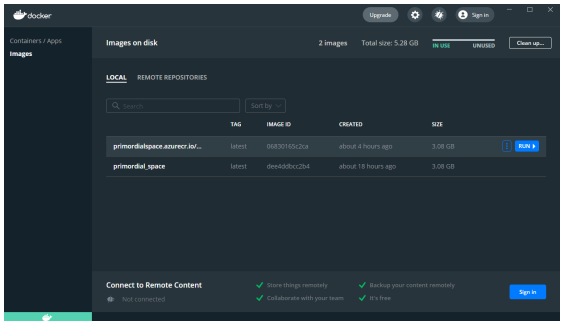


Figure 9(a):Docker

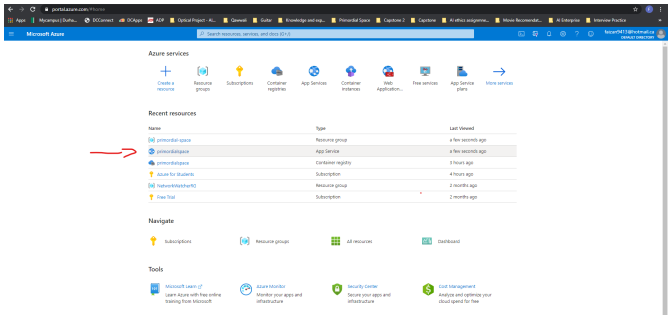


Figure 9(b): Azure deployment

11. Future Work

- Add more info about different galaxies
- Integrate Database to store user information
- Increase model accuracy
- Integrate support for mobile/tablet view on the website

12. Appendices

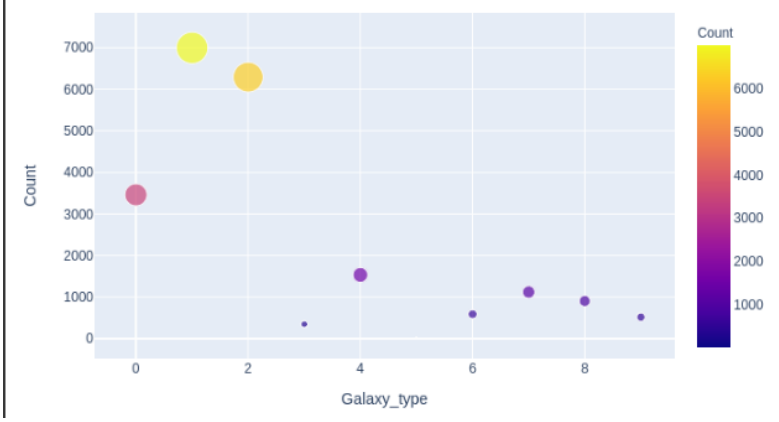
Appendix A: EDA of the dataset

```
#displaying random 20 labels with their descriptions
for counter, i in enumerate(range(np.random.randint(0, labels.shape[0], size=20).shape[0])):
  if counter<=20:
    print(i, galaxy10cls_lookup(labels[i]))
```

```
0 Smooth, in-between round
1 Smooth, in-between round
2 Disk, Edge-on, Rounded Bulge
3 Smooth, in-between round
4 Disk, Edge-on, Rounded Bulge
5 Smooth, Completely round
6 Smooth, Completely round
7 Smooth, Completely round
8 Smooth, Completely round
9 Disk, Face-on, Tight Spiral
10 Smooth, Completely round
11 Smooth, Cigar shaped
12 Smooth, Completely round
13 Disk, Face-on, Tight Spiral
14 Disk, Face-on, Tight Spiral
15 Disk, Face-on, Medium Spiral
16 Smooth, Completely round
17 Disk, Face-on, Loose Spiral
18 Smooth, Completely round
19 Smooth, in-between round
```

Figure 10: Displaying first 20 classes of the labels

```
#scatter plot of galaxy types by thier count
fig = px.scatter(galaxy_df, x='Galaxy_type', y='Count', color="Count",
                 size='Count')
fig.show("png")
```




```
#plotting data disribution of all galaxy types
fig = px.line(galaxy_df, x='Galaxy_type', y='Count')
fig.show("png")
```

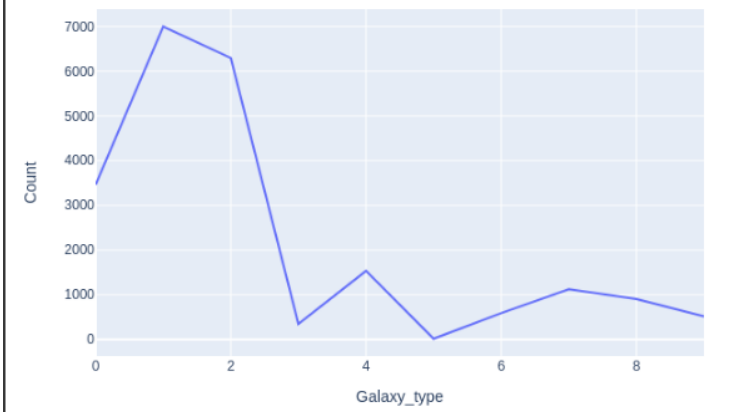


Figure 11: (a)Scatter plot of label classes distribution, (b) Line plot of label classes distribution

```
#BGR distribution of an image from the dataset
color = ('b','g','r')
for i,col in enumerate(color):
    histr = cv2.calcHist([images[1515]],[i],None,[256],[0,256])
    plt.plot(histr,color = col)
    plt.xlim([0,256])
plt.show()
```

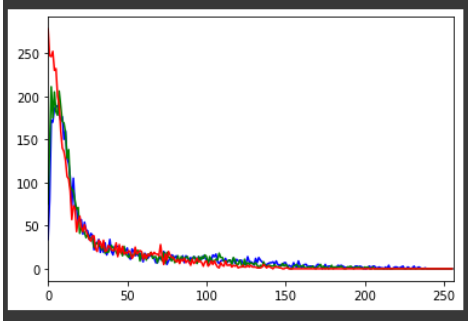


Figure 12: BGR distribution of an image from the dataset

13. References

[1] https://astronn.readthedocs.io/en/latest/galaxy10sdss.html

[2] https://www.sdss.org/

[3] https://www.zooniverse.org/projects/zookeeper/galaxy-zoo/