Galaxy Classification Using Galaxy Zoo Data

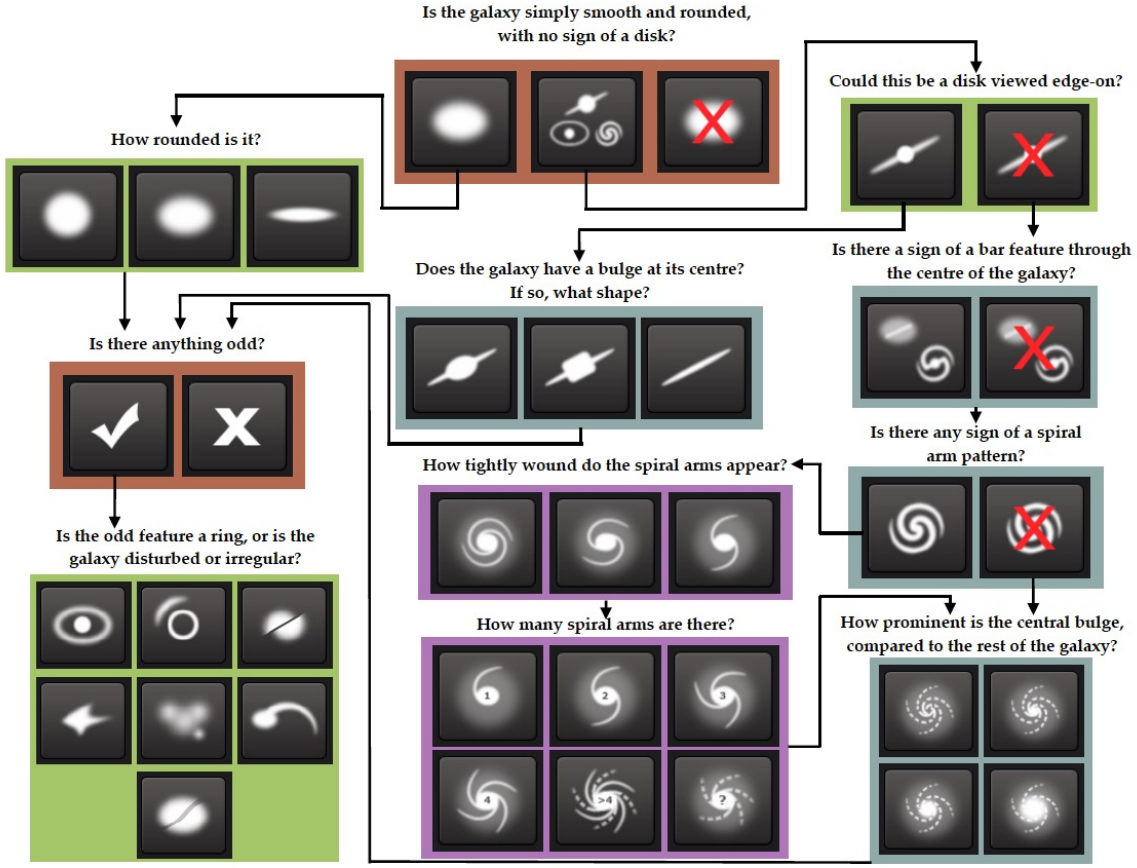
221AIG08 Jiwon Han[1]

[1]Departments of Artificial Intelligence·Software, Ewha Womans University

Introduction

Importance of morphological classification

(1) Galaxy Morphology: The Key to Understanding the Universe



Galaxy Zoo Decision Trees

Galaxy morphological classification holds immense importance in the field of astronomy for several reasons:

- **Understanding Galaxy Evolution**: Astronomers can trace evolutionary paths by categorizing galaxies based on their shapes, sizes, and structures,
- **Studying Galactic Dynamics**: Different galaxy morphologies indicate distinct dynamical properties.
- **Probing Environmental Influence**: The morphology of a galaxy can be influenced by its environment, such as interactions with neighboring galaxies.
- **Identifying Rare and Exotic Objects**: Morphological classification helps identify rare and exotic galaxies, such as irregular, peculiar, or merging systems.

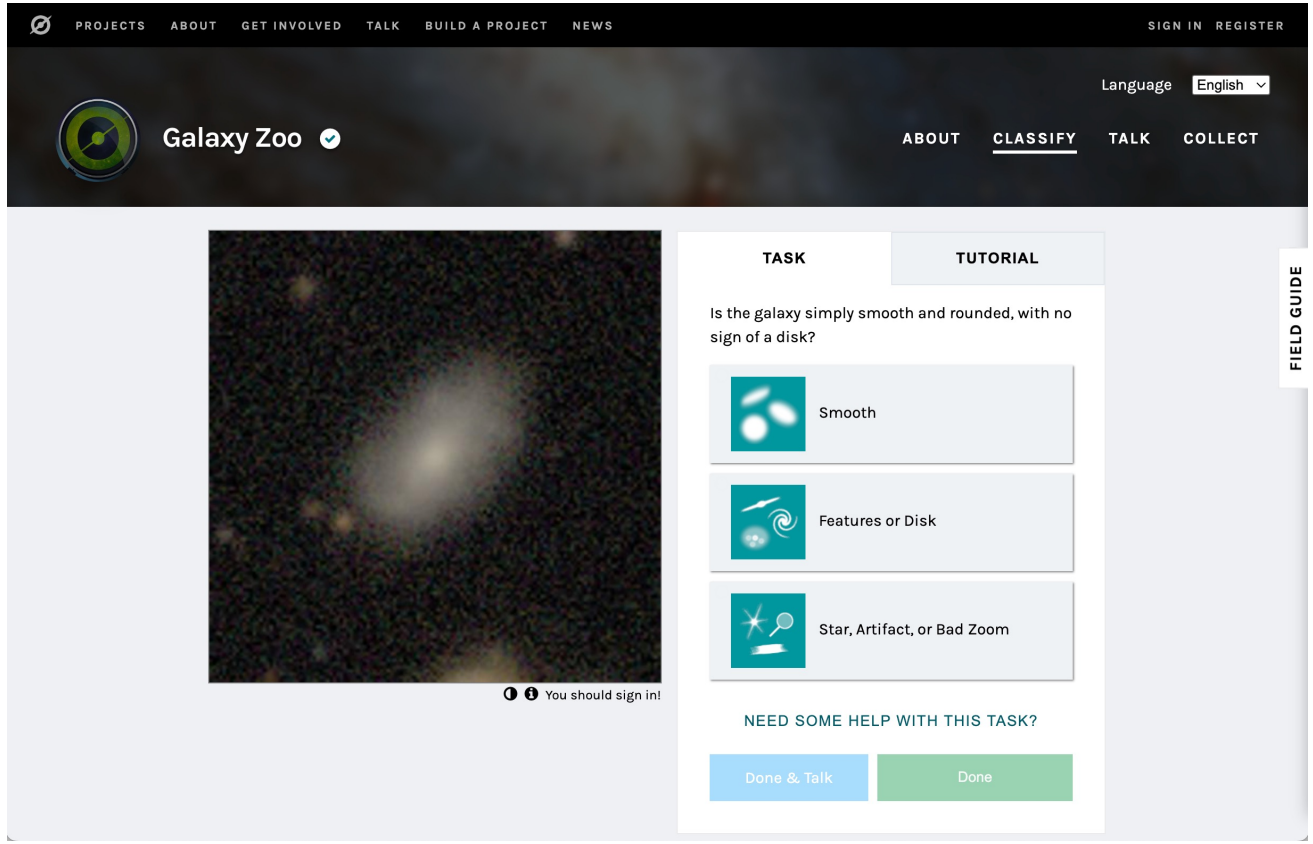
Introduction

Galaxy Zoo: Bridging science with citizen participation

(2) Inviting volunteers to Zooniverse

- Galaxy Zoo represents an innovative citizen science project, calling upon volunteers worldwide to assist astronomers in the classification of galaxies.

- Through the Galaxy Zoo platform, participants are presented with images of galaxies captured by telescopes. They are then tasked with categorizing these galaxies based on their shapes, sizes, and other characteristics.



Galaxy Zoo Webpage

https://kaggle.com/c/galaxy-zoo-the-galaxy-challenge/overview

Data
Kaggle, Galaxy Zoo Challenge

(1) Data Description

1) Input images (Training)

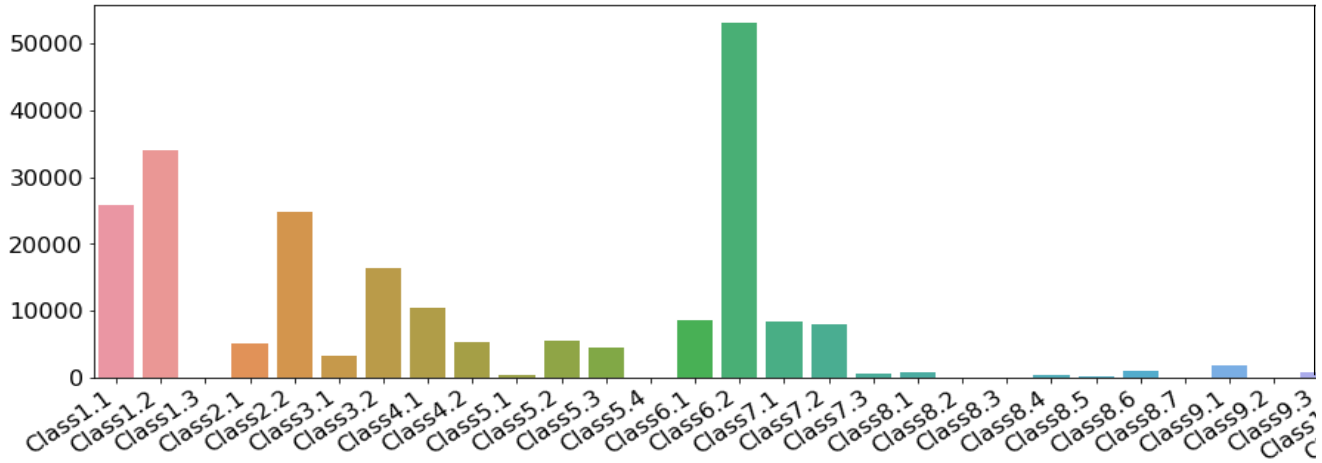
- JPG images of 61,578 galaxies.
- Files are named according to their Galaxy ID.

2) Output labels (Training)

- Probability distributions for the classifications for each of the training images.

3) Images with labels (Test)

- JPG images of 79,975 galaxies.
- Files are name according to their Galaxy ID.
- Probabilities for each of images are provided.



https://kaggle.com/c/galaxy-zoo-the-galaxy-challenge/overview

Data

Kaggle, Galaxy Zoo Challenge

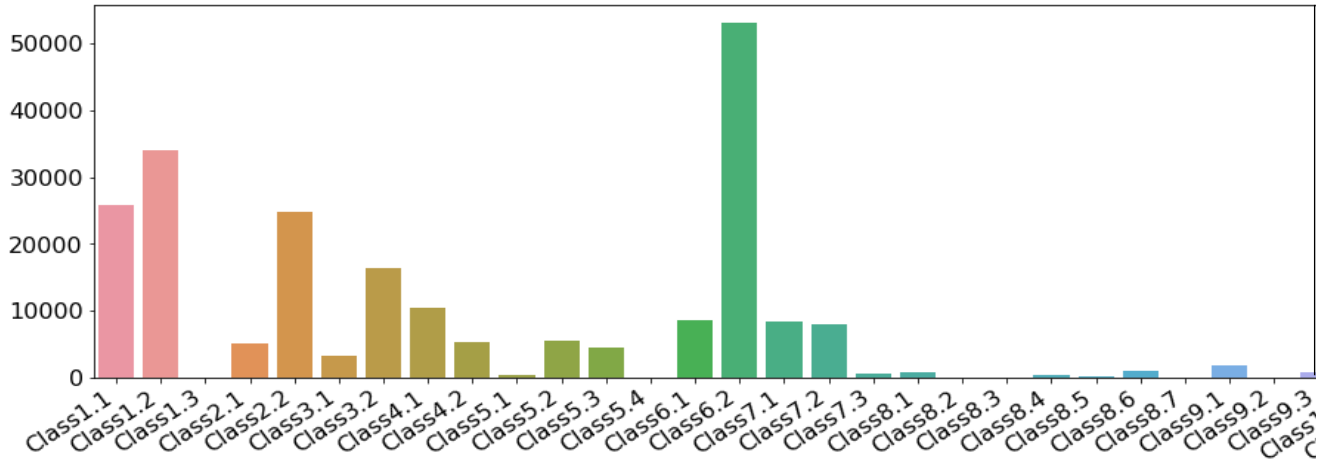
(1) Data Description

1) Input images (Training)

- JPG images of 61,578 galaxies.
- Files are named according to their Galaxy ID.

2) Output labels (Training)

- Probability distributions for the classifications for each of the training images.



Exp 1 (200 Epochs)	Train		Val
Exp 2 (20 Epochs)	Train	Val	Test

Data
Pre-processing Steps

(2) Load Data

1) Load Data

Load the probability data for each class. Add a new column, 'Label', which is assigned based on the class with the highest probability.

2) Extract Key Columns

Extract main two columns, ‘GalaxyID’ and ‘Label’, for faster computation and accessment.

3) Split Data

Split the entire dataset into training, validation, and test sets to ensure robust model evaluation.

```
Train set: 36946 samples
Validation set: 12316 samples
Test set: 12316 samples
```

```python
# Set Label (Train)
train_df = pd.read_csv(os.path.join(train_data_path, 'training_solutions_rev1.csv'))
print(train_df.shape)
print(train_df.columns)
```

✓ 2.4s Python

```
(61578, 38)
Index(['GalaxyID', 'Class1.1', 'Class1.2', 'Class1.3', 'Class2.1', 'Class2.2',
       'Class3.1', 'Class3.2', 'Class4.1', 'Class4.2', 'Class5.1', 'Class5.2',
       'Class5.3', 'Class5.4', 'Class6.1', 'Class6.2', 'Class7.1', 'Class7.2',
       'Class7.3', 'Class8.1', 'Class8.2', 'Class8.3', 'Class8.4', 'Class8.5',
       'Class8.6', 'Class8.7', 'Class9.1', 'Class9.2', 'Class9.3', 'Class10.1',
       'Class10.2', 'Class10.3', 'Class11.1', 'Class11.2', 'Class11.3',
       'Class11.4', 'Class11.5', 'Class11.6'],
      dtype='object')
```

```python
train_df['Label'] = train_df.iloc[:, 1:].idxmax(axis=1)
train_df = train_df[['GalaxyID', 'Label']]
print(train_df)
```

✓ 0.6s Python

```
       GalaxyID     Label
0        100008  Class6.2
1        100023  Class6.2
2        100053  Class6.2
3        100078  Class1.1
4        100090  Class6.2
...         ...       ...
61573    999948  Class6.2
61574    999950  Class1.1
61575    999958  Class6.1
61576    999964  Class1.2
61577    999967  Class1.1

[61578 rows x 2 columns]
```

Data
Pre-processing Steps

(3) Data Pre-Processing

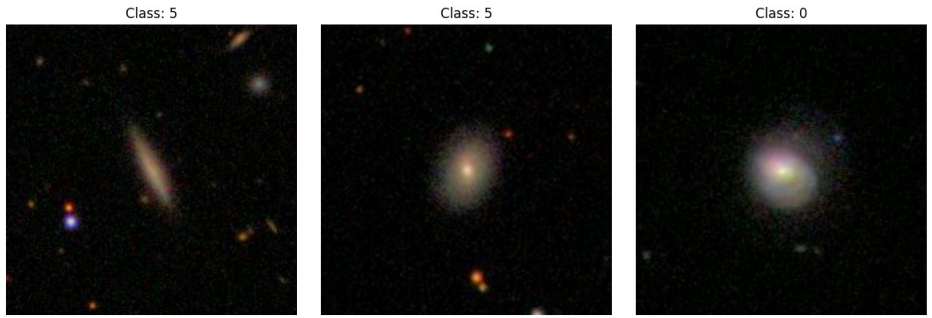
1) Define Transformations

Define transformation methods (e.g., resizing, normalization) to preprocess the images.

2) Create Custom Dataset

Create a custom Dataset class to efficiently load the data with PyTorch's DataLoader.

3) Visualize Images in the Dataset




```python
# Custom Dataset
class GalaxyDataset(Dataset):
    def __init__(self, image_dir, labels_df, transform=None):
        self.image_dir = image_dir
        self.labels_df = labels_df
        self.transform = transform

    def __len__(self):
        return len(self.labels_df)

    def __getitem__(self, idx):
        img_name = os.path.join(self.image_dir, f"{self.labels_df['GalaxyID'].iloc[idx]}.jp
        image = Image.open(img_name).convert('RGB')
        label = int(self.labels_df.iloc[idx, -1].split('.')[0].replace('Class', '')) - 1

        if self.transform:
            image = self.transform(image)

        return image, label

# Data Transform
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
])
```

✓ 0.0s Python

Model
CNN-based Classification Model

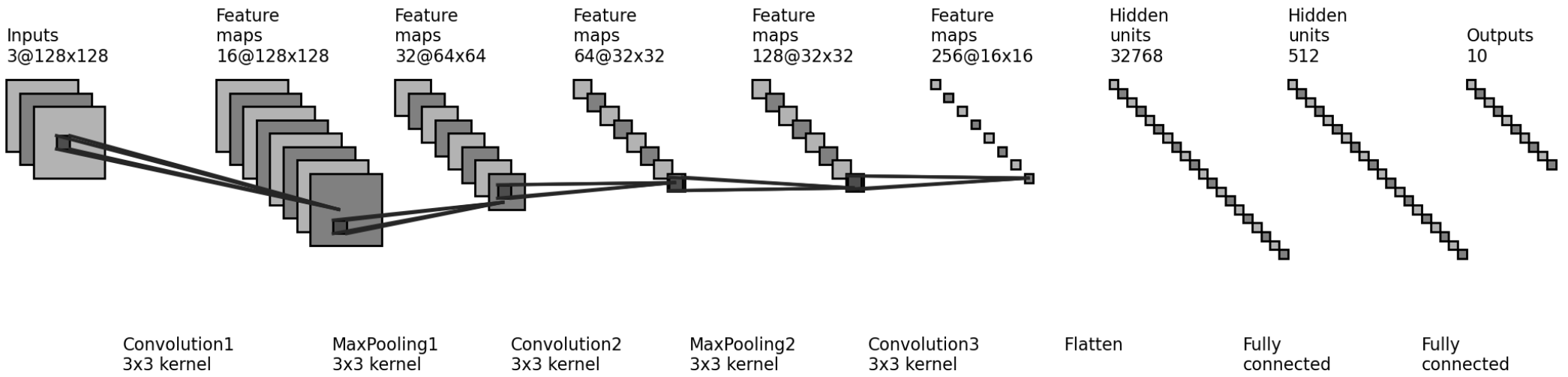
(1) Model Structure

1) Layer Configuration

- Input: (3, 128, 128)
- Conv Layer: [Conv, MaxPool] x 3
- FC Layer: [FullyConnected] x 2
- Output: 10 Classes

2) Hyper-Parameters Setting

- Epochs: 200
- Batch Size: 32
- Learning Rate: 1e-4
- Loss: Cross Entropy Loss
- Optimizer: Adam



Result
Experiment1 : Basic Settings

(1) Train : Val = 0.8 : 0.2

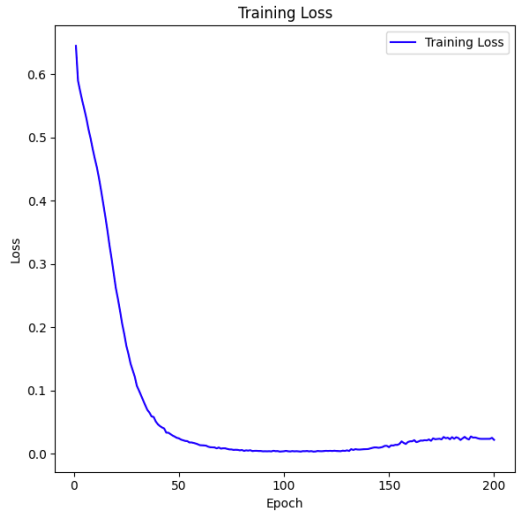
1) Basic Settings

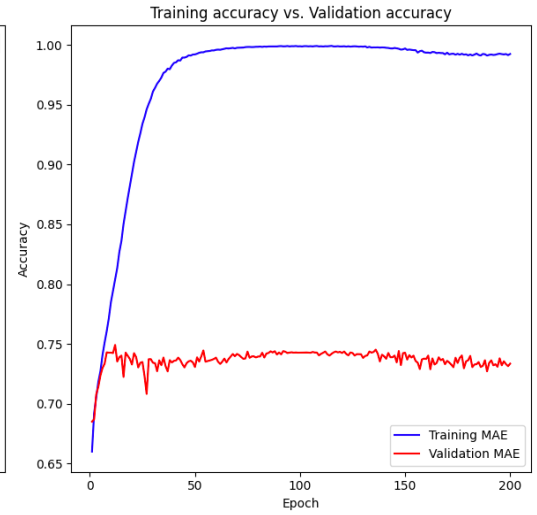
- Train : Val = 49262 : 12316
- Image: 3-Channel RGB
- Size: (128, 128)
- Epochs: 200
- Output: 10 Classes

Exp 1
(200 Epochs)

2) Train Results

- Best Result Epoch: 121
- Best Train Loss: 0.0045
- Best Train Accuracy: 0.9990
- Best Val Accuracy: 0.7437





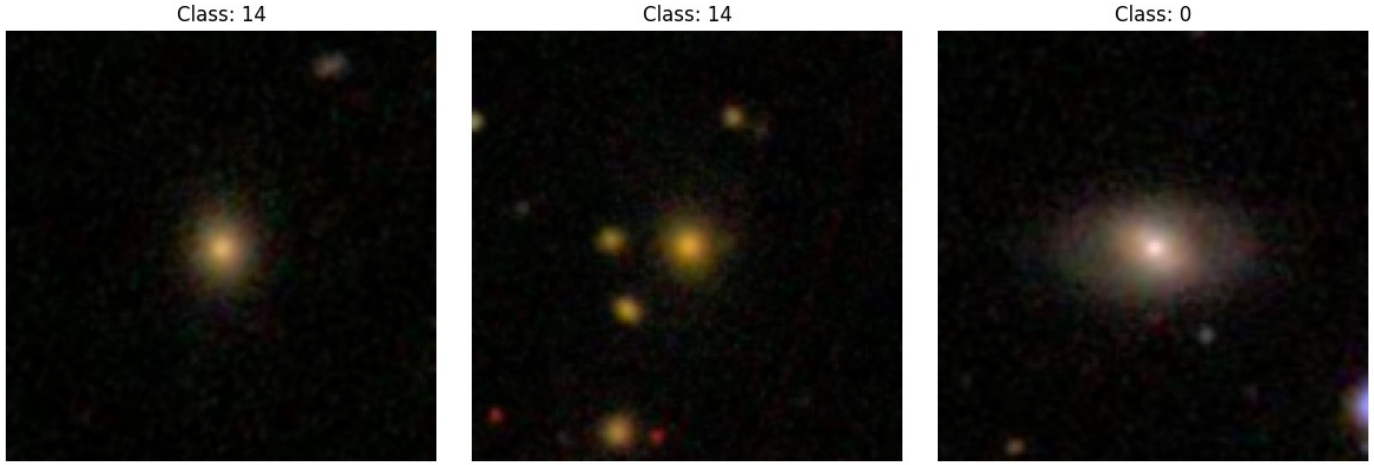
Result
Experiment2: Variations

Exp 2
(20 Epochs)

(2) Train : Val : Test = 0.6 : 0.2 : 0.2

1) Updated Parts

- Train : Val : Test = 36946 : 12316 : 12416
- Image: 3-Channel RGB
- Size: (64, 64) / (128, 128)
- Epochs: 20
- Output: 37 Classes



Result
Experiment2: Variations

Exp 2
(20 Epochs)

(2) Train : Val : Test = 0.6 : 0.2 : 0.2

1) Updated Parts

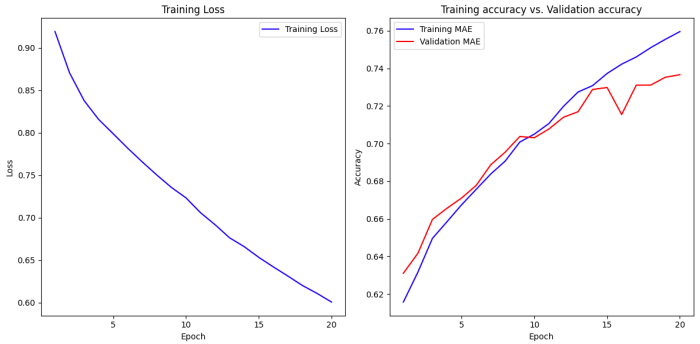
- Train : Val : Test = 36946 : 12316 : 12416
- Image: 3-Channel RGB
- Size: (64, 64) / (128, 128)
- Epochs: 20
- Output: 37 Classes

2) Result Comparison: Size

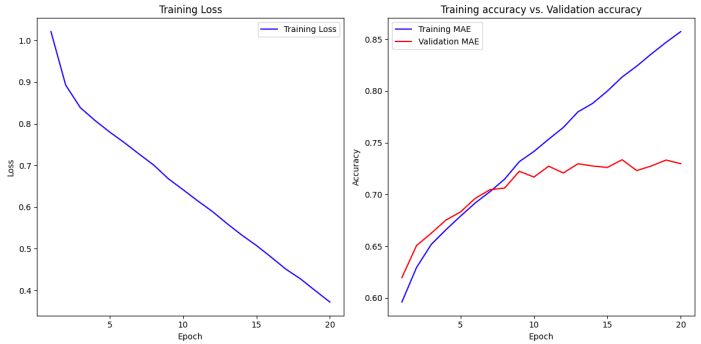
- Channel: 3-Channel RGB
- Size: (64, 64) / (128, 128)
- Test Accuracy:
  0.7337 / 0.7210

Gal_ID	Class	Label	Pred
348633	Class6.2	14	14
134877	Class6.2	14	14
568318	Class1.1	0	14
383087	Class6.2	14	14
367576	Class6.1	13	14

Gal_ID	Class	Label	Pred
348633	Class6.2	14	1
134877	Class6.2	14	14
568318	Class1.1	0	14
383087	Class6.2	14	14
367576	Class6.1	13	1



[64x64, 3-Channel RGB Images]



[128x128, 3-Channel RGB Images]

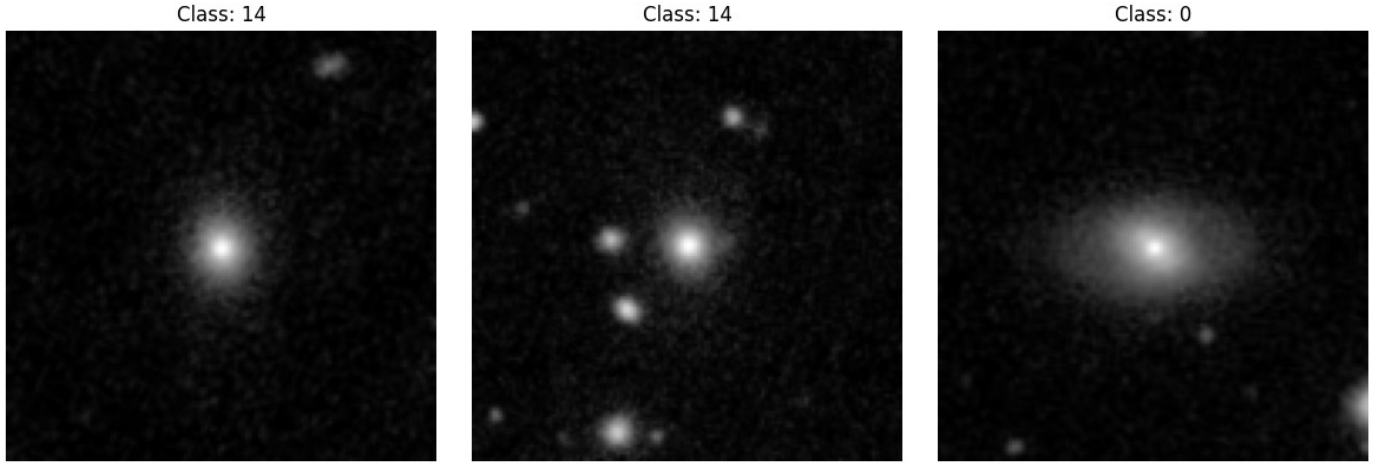
Result
Experiment2: Variations

Exp 2
(20 Epochs)

(3) Train : Val : Test = 0.6 : 0.2 : 0.2

1) Updated Parts

- Train : Val : Test = 36946 : 12316 : 12416
- Image: 1-Channel Grayscale
- Size: (64, 64) / (128, 128)
- Epochs: 20
- Output: 37 Classes



Result
Experiment2: Variations

Exp 2
(20 Epochs)

(3) Train : Val : Test = 0.6 : 0.2 : 0.2

1) Updated Parts

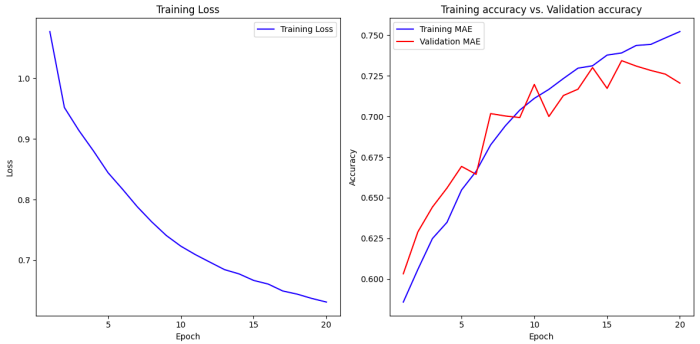
- Train : Val : Test = 36946 : 12316 : 12416
- Image: 1-Channel Grayscale
- Size: (64, 64) / (128, 128)
- Epochs: 20
- Output: 37 Classes

2) Result Comparison: Channel

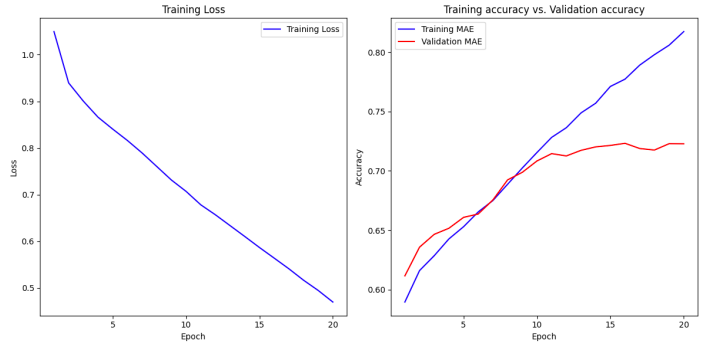
- Channel: 1-Channel Grayscale
- Size: (64, 64) / (128, 128)
- Test Accuracy:
  0.7201 / 0.7210

Gal_ID	Class	Label	Pred
348633	Class6.2	14	14
134877	Class6.2	14	14
568318	Class1.1	0	14
383087	Class6.2	14	14
367576	Class6.1	13	1

Gal_ID	Class	Label	Pred
348633	Class6.2	14	1
134877	Class6.2	14	1
568318	Class1.1	0	1
383087	Class6.2	14	14
367576	Class6.1	13	0



[64x64, 1-Channel Grayscale Images]



[128x128, 1-Channel Grayscale Images]