Star/Galaxy Classification Report

Written and implement by: Dang Cao Nguyen Pham

To produce the results locally: run script-local.sh
To produce the results with docker (working with cuda): run script-docker.sh

1.1

Code for the task: **prepare.py** with functions from:

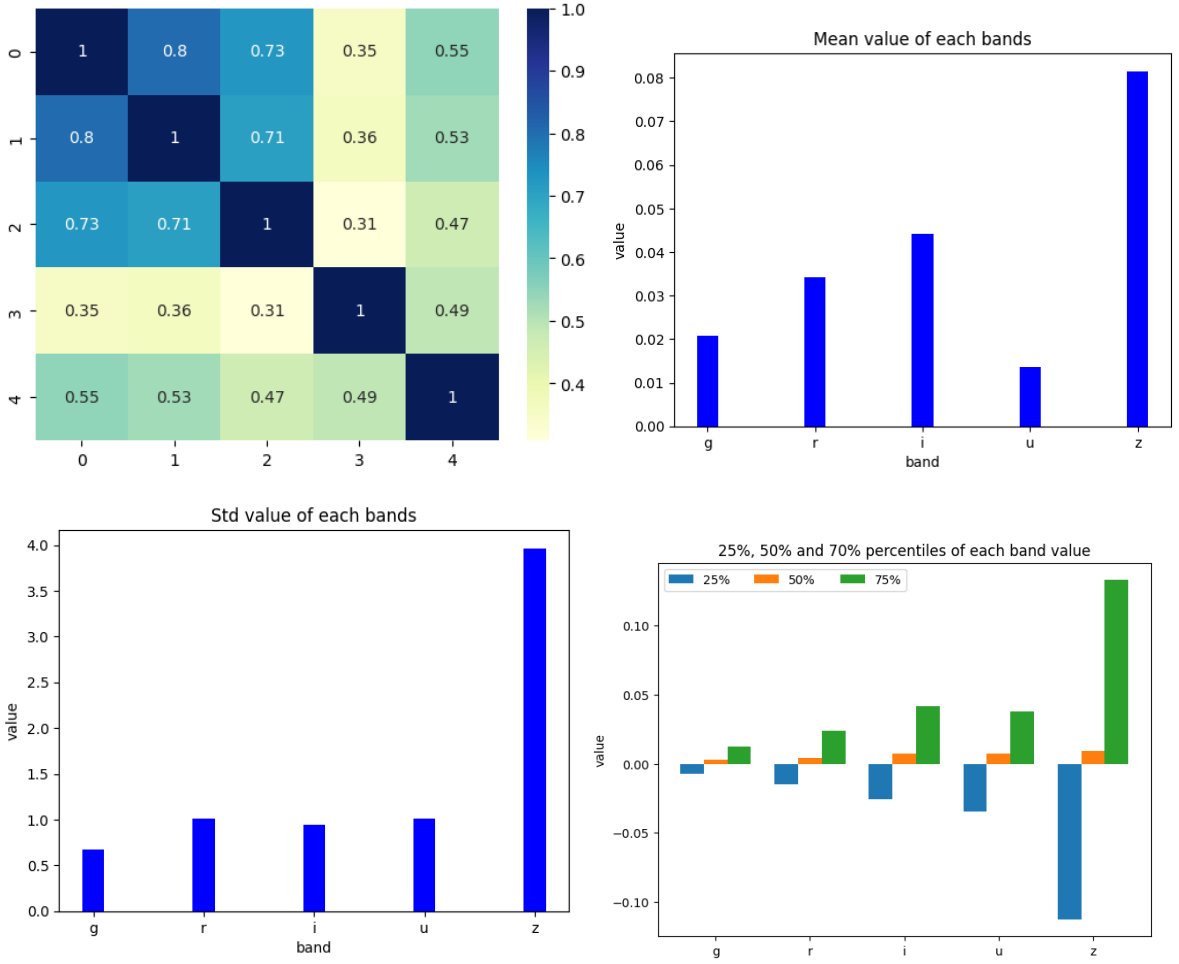
- **process.py** take data from data folder, align bands and process into tensor [N, C, H, W] for all images, [N, G, (x,y)] for all galaxy and [N, S, (x,y)] for all stars

Images 80 to 130 will be used for training, validation and testing.

Summary statistics:

- **Number of galaxies to stars:** 10545 to 26228 (ratio 1 to 2.6)
- **Recorded max shift vertical to align bands:** 13 pixels
- **Recorded max shift horizontal to align bands:** 4 pixels

Plots



Plot 1: Correlation betweens the bands (0,1,2,3,4) for (g, r, i, u, z):
Plot 2: Mean value of each band value
Plot 3: Standard deviation of each band value
Plot 4: 25%, 50% and 70% percentiles of each band values

1.2

Code for the task: **dataset.py**

SDSSData will be prepared on demand with regards to the distance from the centre pixel value, to extract images of according heights and width of each galaxy and stars.

SDSSData will then be split into **SDSSData_train** and **SDSSData_val** sets with the help of **sklearn.model_selection.train_test_split()** with option **stratify=labels** and the ratio 80:20, this ensures the distribution ratio of stars and galaxies.

Data related to the image **301/8162/6/frame-irg-008162-6-0080.jpg** has been excluded from the preparation step above. Only when the test dataset is called, image **80** will be processed and added to the test dataset.

1.3

Code for the task: **train.py and main.py**
Chosen loss function: **CrossEntropyLoss()**
Chosen evaluation metrics: **Accuracy** and **F1-score**

Model in use:
U-Net is a popular convolutional neural network (CNN) architecture used for image segmentation tasks.
The encoder pathway of U-Net is designed to capture the spatial information of an input image. It consists of several convolutional and pooling layers that progressively reduce the spatial dimensions while increasing the number of feature channels. This helps extract high-level features from the input image. **(U-Net: Convolutional Networks for Biomedical Image Segmentation Olaf Ronneberger, Philipp Fischer, and Thomas Brox)**
After the U-Net convolutional layer, the image will be flatten and feed into a hidden layer of adjustable number of hidden nodes and then output the likelihood of two nodes representing 2 classes of stars and galaxies

Hyperparameters to tune with the goal of maximising f-1 score (appropriate for imbalance dataset):

- **Learning_rate:** learning rate of the neural network
- **Optimizer:** optimization function to train
- **Dist_from_center:** distance from centre pixel, related to image size
- **Hidden_nodes:** number of nodes in the last hidden layers
- **Batch_size:** number of image per batch
- **Drop_out:** chance of a node to be dropped

ML pipeline

1. Data cleaning and preparation **(prepare.py)**
2. Dataset creation, division, and resampling for class balance **(dataset.py)**
3. Model construction and fine-tuning based on the F1 score **(train.py)**
4. Iterative optimization to identify the best-performing model **(train.py)**
5. Comprehensive testing of the optimised model **(main.py)**

Best parameters:

- "learning_rate": 0.0009,
- "optimizer": Adam,
- "dist_from_center": 20,
- "hidden_nodes": 256,
- "batch_size": 100,
- "drop_out": 0.282,

Model quality with and without tuning, as tested on image data 80:

Without tuning:

Test Loss: 0.457

Test Accuracy: 75.4%

Test f1: 0.61

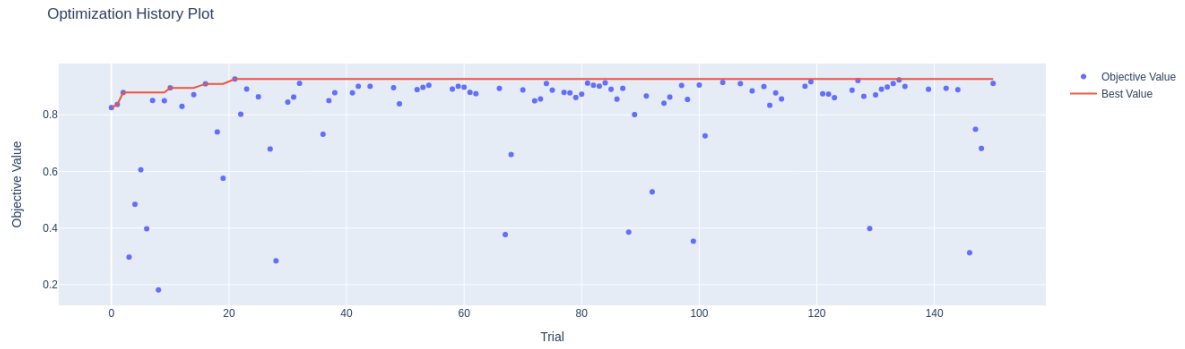
With tuning:

Test Loss: 0.213,

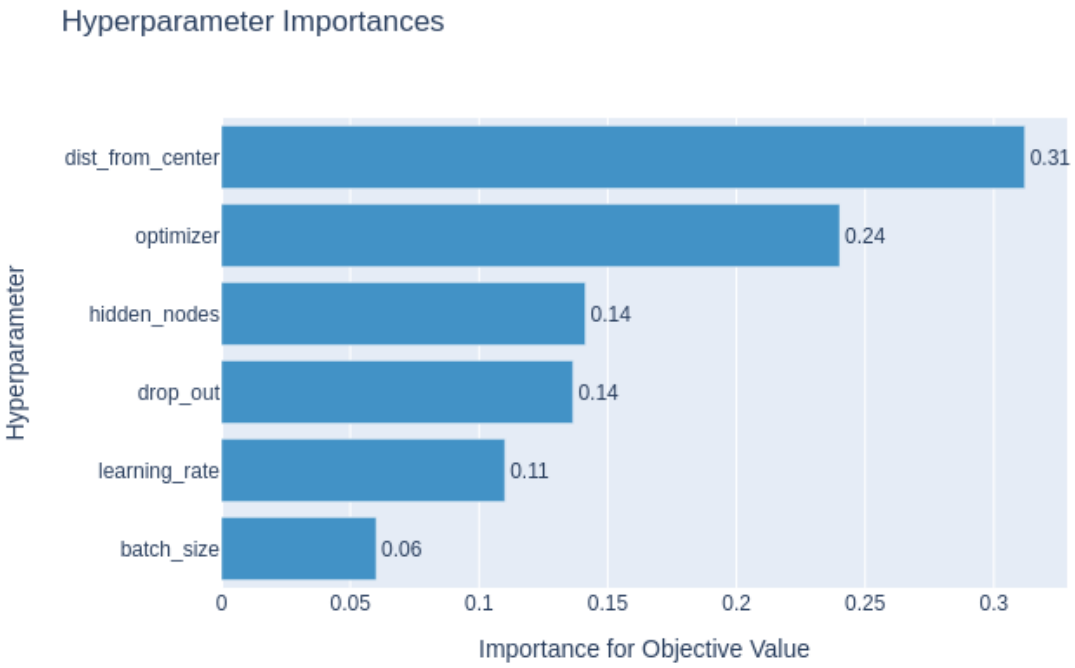
Test Accuracy: 89.8%

Test f1: 0.80

Plot 1: Histogram of the tuning process



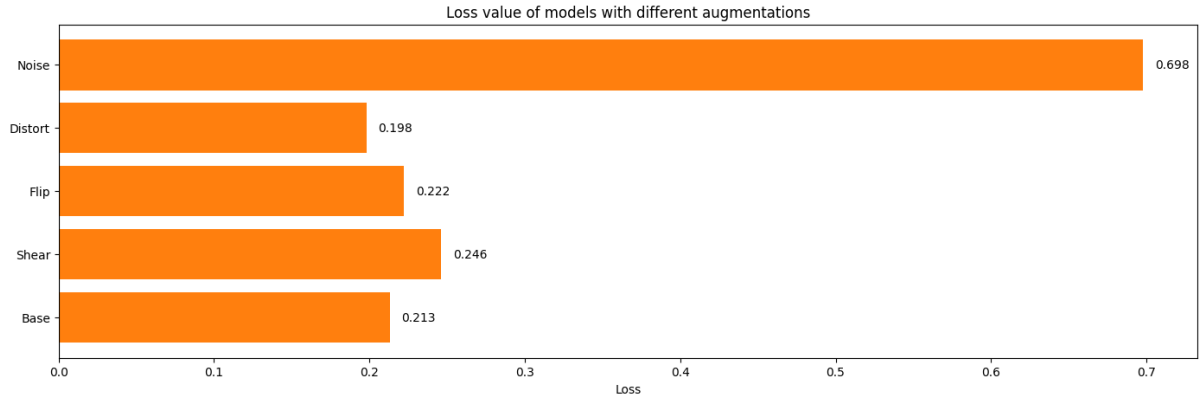
Plot 2: Hyperparameters ranked by importance

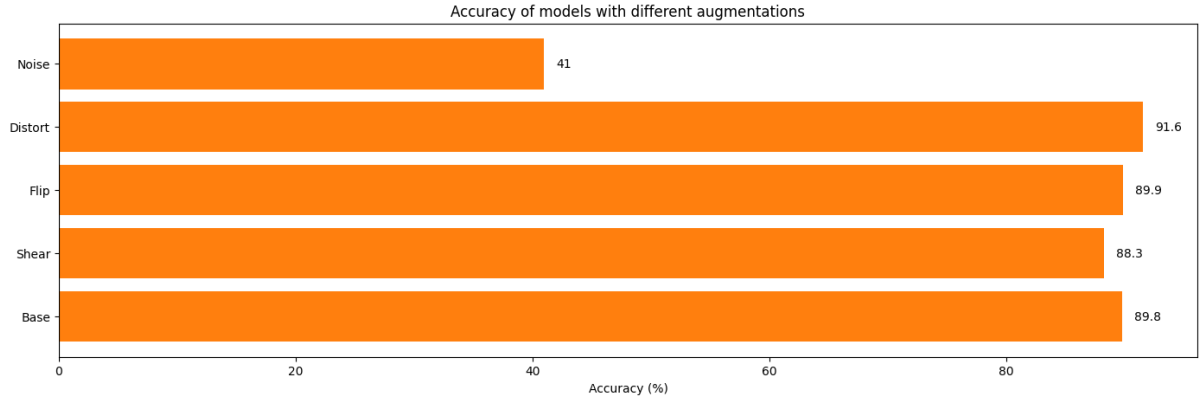


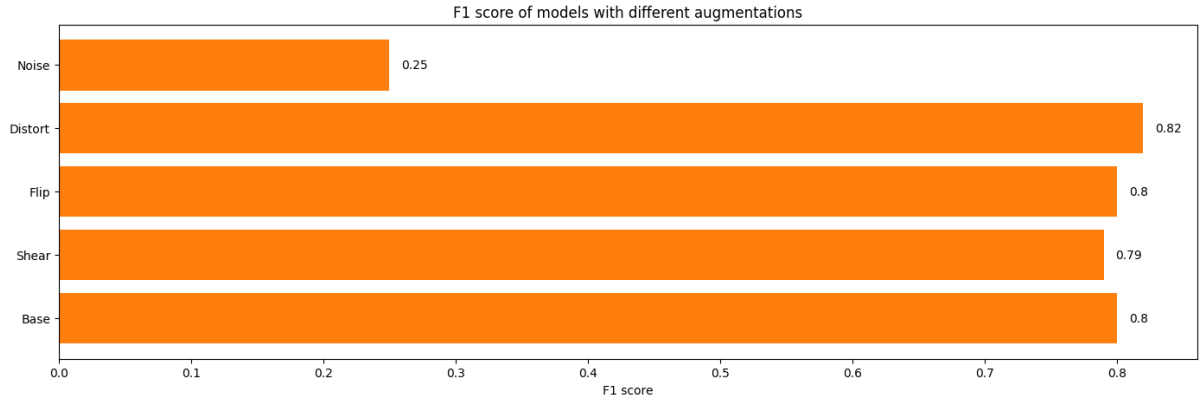
1.4

Code for the task: **main.py**
Data augmentation techniques used: **Shear, flip, optical distortion and Gaussian noise**

Results:







The inclusion of **optical distortion** as an augmentation technique resulted in a noteworthy improvement in the model's performance, with a 0.02 increase in the F1 score, a 2% boost in accuracy, and a reduction in loss by approximately 0.015.

Conversely, **Gaussian noise** had a significantly negative impact, leading to a one-third decrease in all metrics.

The other augmentation methods showed relatively similar results to the baseline model. These findings emphasise the effectiveness of optical distortion as a beneficial augmentation technique while highlighting the drawbacks of Gaussian noise for this classification problem.