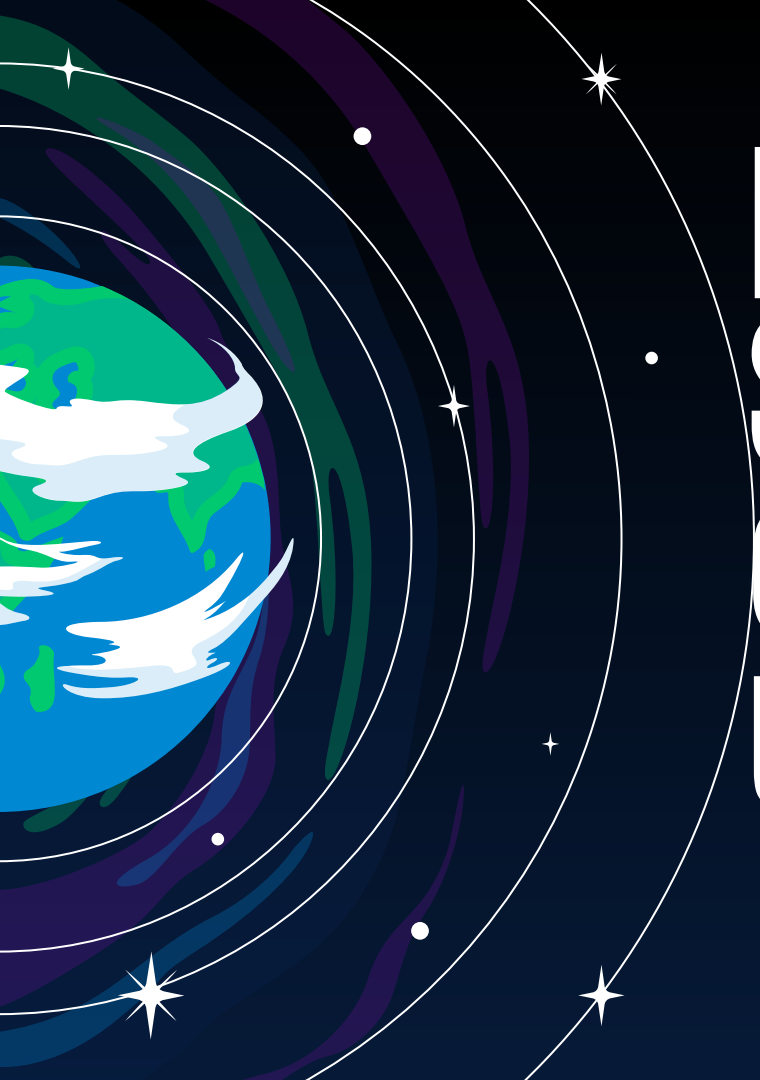


Pytroch Satellite image classification using networks

Wong Kai Yuan DC026157
Guan Jia Xi CC029721

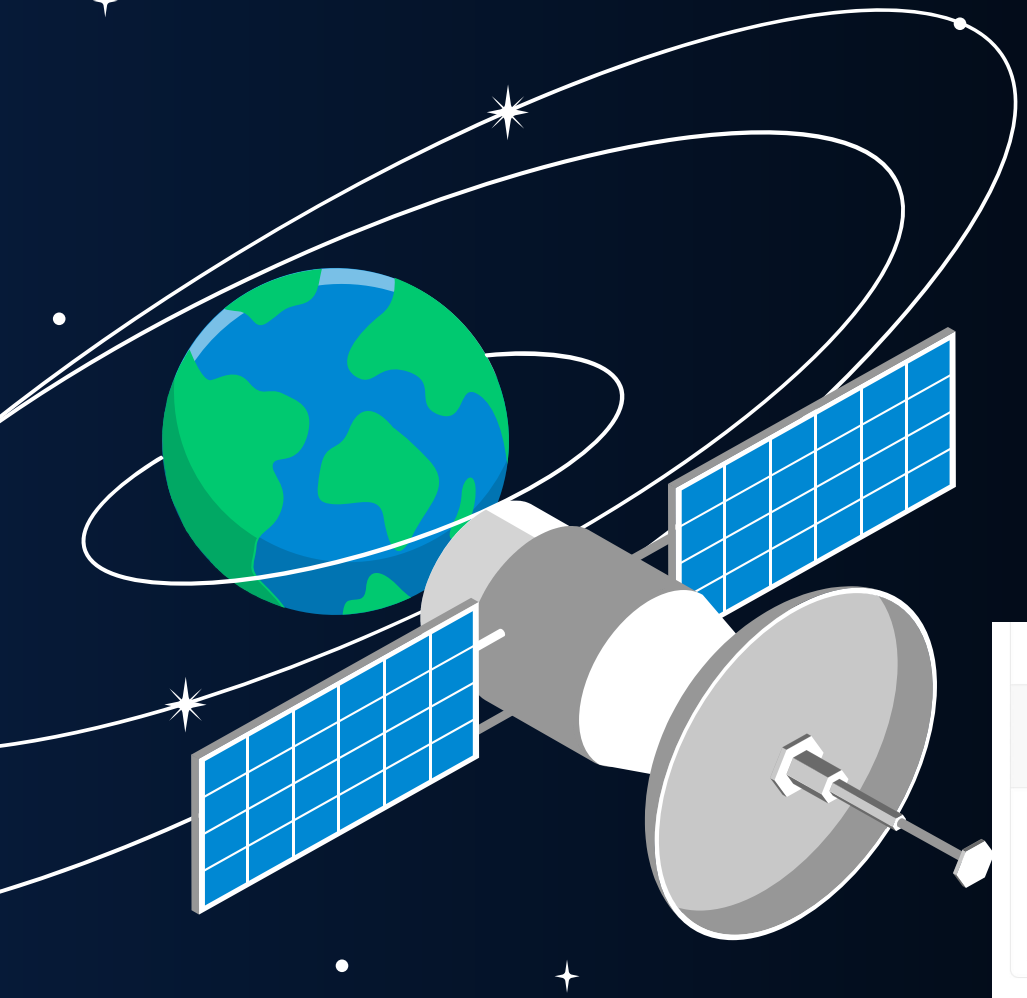
TABLE OF CONTENTS

01

INTRODUCTION

Pytorch

PyTorch is an open source machine learning library primarily used for Deep Learning applications, computer vision and natural language processing using GPUs and CPUs.



Dataset

Satellite image Dataset from Kaggle, contains four classes of satellite images which are: water, desert, cloudy and green area, with 5631 images in total and around 1500 images of each class.

data (4 directories)

About this directory

The whole dataset has 5631 images with jpg format

cloudy	desert	green_area	water
1500 files	1131 files	1500 files	1500 files

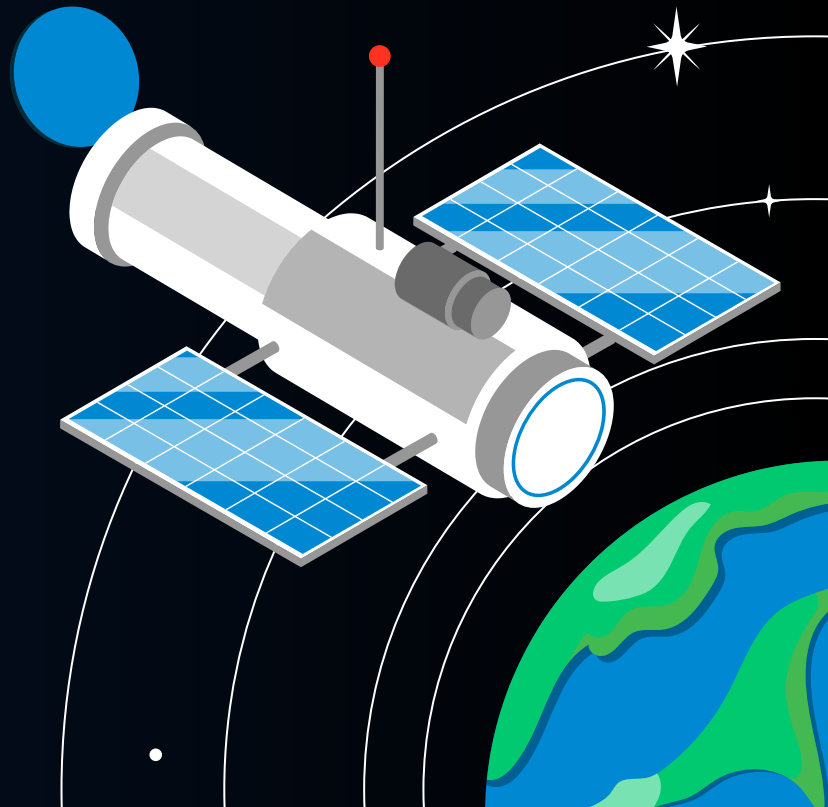
Neural Network Model

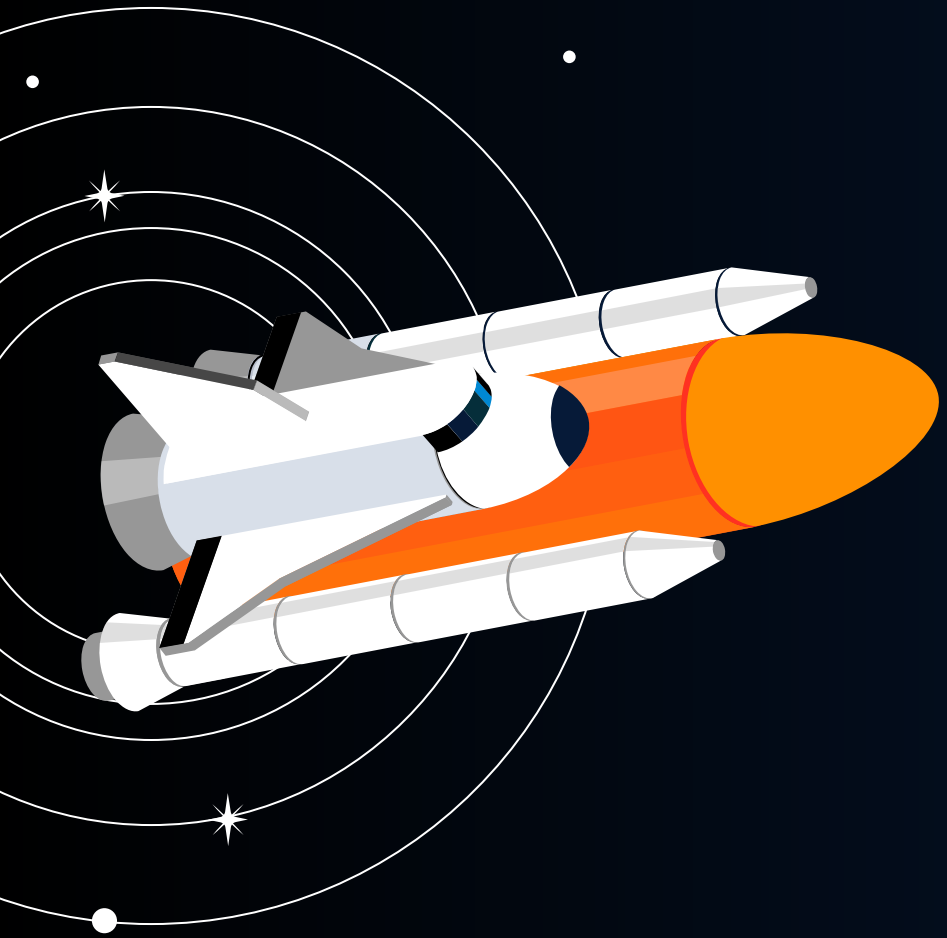
In this project, MLP and DenseNet model will be used to classify the dataset

02

Goal

To develop a deep learning or neural network model that can predict or classify satellite images into four classes using pytorch.





03

Project Body

For each Implementation :

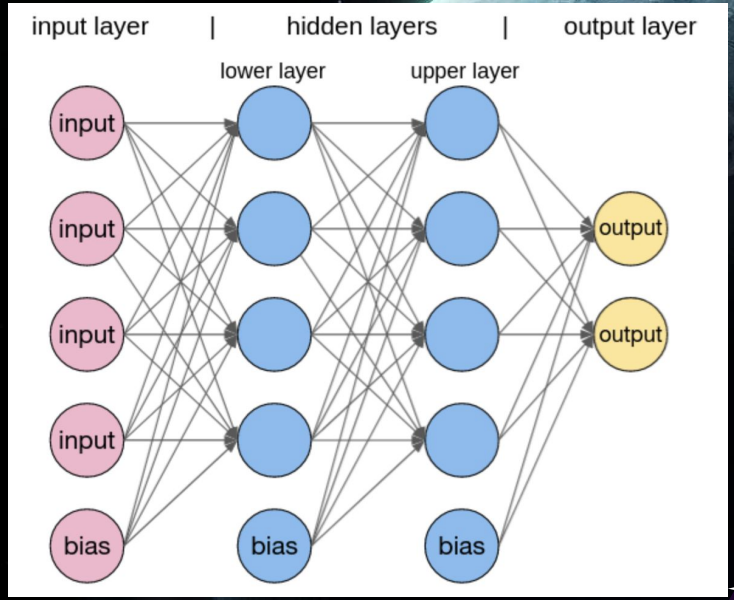
- Introduce our model
- Data Processing
- Define Model
- Train and evaluate the model
- Final result on testing

- **More details in report**

Implementation of Multilayer Perceptron (MLP)

What is MLP ?

A multilayer perceptron is a fully connected class of feedforward artificial neural network. It is one of the most basic neural network architectures.



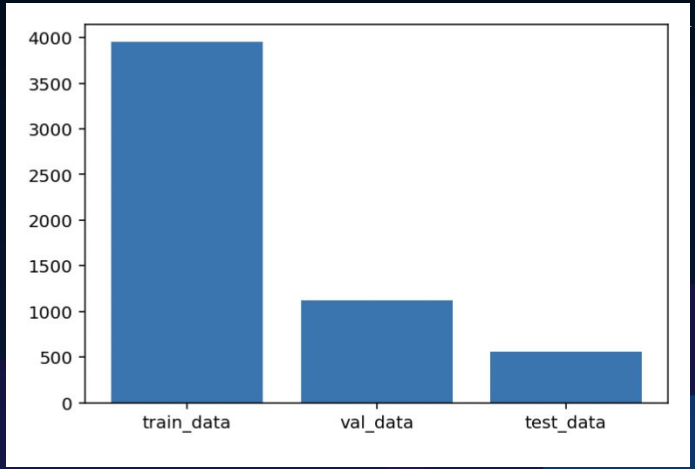
Implementation of MLP

01 Data processing

- Import all modules needed
- Define transform with known stats
- Load dataset with transform
- Split data into Training data, Validation data and Testing data in ratio of 7:2:1
- Define data loader with batch size of 64

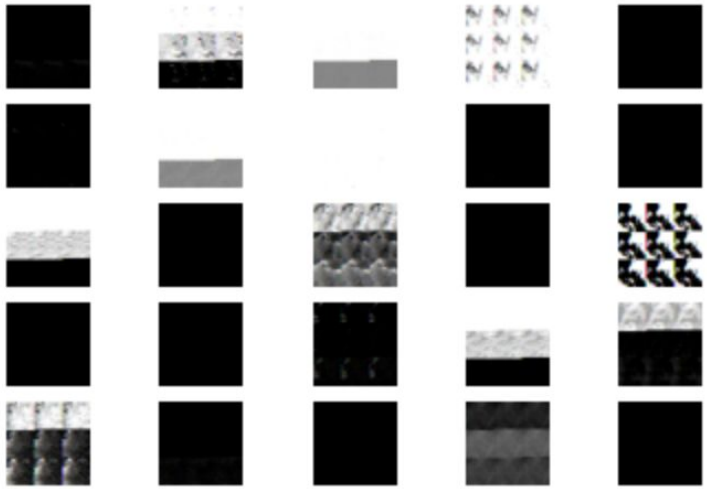
```
Split data to training, val and test in 7:2:1
Num training images:  3950
Num validating images:  1120
Num test images:  561
```

Bar chart of data after splitting



Visualization of Training Data

```
N_IMAGES = 25

images = [image for image, label in [train_data[i+200] for i in range(N_IMAGES)]]

plot_images(images)
```



Implementation of MLP

02 Define Model

Multilayer perceptron (MLP) with four hidden layers.

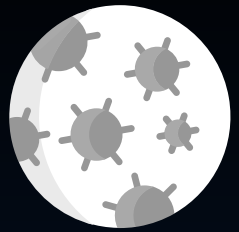
- Input dimension : 64*64*3
- Output dimension : 4 (total 4 class for classification)
- Hidden Layer 1 : 5000
- Hidden Layer 2 : 1500
- Hidden Layer 3 : 250
- Hidden Layer 4 : 50

Define trainable parameters :

```
def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requires_grad)

print(f'The model has {count_parameters(model):,} trainable parameters')
```

```
The model has 69,334,504 trainable parameters
```

Implementation of MLP



03 Train and evaluate the model

i) Define Optimizer, Criterion, Device
ii) Define Training Loop & Evaluating Loop
iii) Start Training !

```
Epoch: 01 | Epoch Time: 0m 52s
        Train Loss: 6.100 | Train Acc: 57.70%
         Val. Loss: 1.149 |  Val. Acc: 70.97%

Epoch: 02 | Epoch Time: 0m 31s
        Train Loss: 0.871 | Train Acc: 74.67%
         Val. Loss: 0.514 |  Val. Acc: 75.97%

Epoch: 03 | Epoch Time: 0m 30s
        Train Loss: 0.465 | Train Acc: 79.27%
         Val. Loss: 0.536 |  Val. Acc: 76.29%

Epoch: 04 | Epoch Time: 0m 30s
        Train Loss: 0.455 | Train Acc: 80.31%
         Val. Loss: 0.424 |  Val. Acc: 82.90%

Epoch: 05 | Epoch Time: 0m 30s
        Train Loss: 0.461 | Train Acc: 79.52%
         Val. Loss: 0.453 |  Val. Acc: 77.34%
```

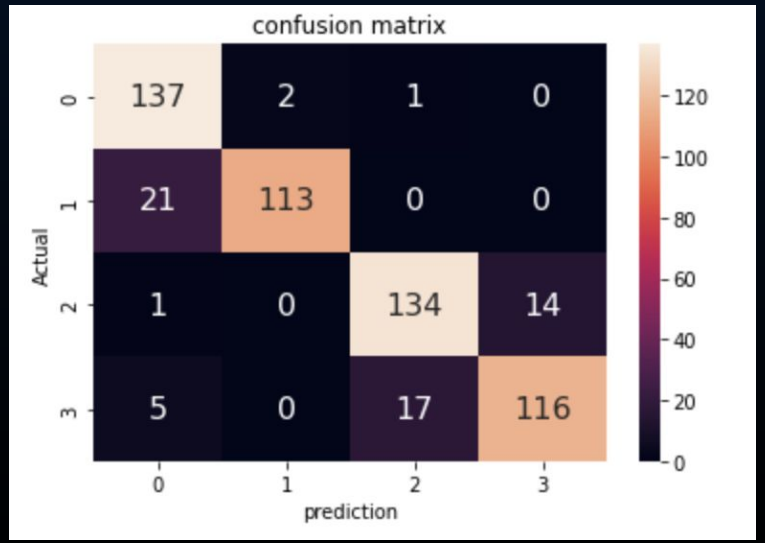
```
Epoch: 06 | Epoch Time: 0m 31s
        Train Loss: 0.420 | Train Acc: 81.74%
         Val. Loss: 0.422 |  Val. Acc: 78.39%

Epoch: 07 | Epoch Time: 0m 30s
        Train Loss: 0.366 | Train Acc: 84.21%
         Val. Loss: 0.377 |  Val. Acc: 83.55%

Epoch: 08 | Epoch Time: 0m 30s
        Train Loss: 0.386 | Train Acc: 83.31%
         Val. Loss: 0.376 |  Val. Acc: 82.74%

Epoch: 09 | Epoch Time: 0m 30s
        Train Loss: 0.362 | Train Acc: 84.11%
         Val. Loss: 0.367 |  Val. Acc: 84.03%

Epoch: 10 | Epoch Time: 0m 30s
        Train Loss: 0.337 | Train Acc: 84.94%
         Val. Loss: 0.350 |  Val. Acc: 85.16%
```

Implementation of MLP

04 Final result on testing

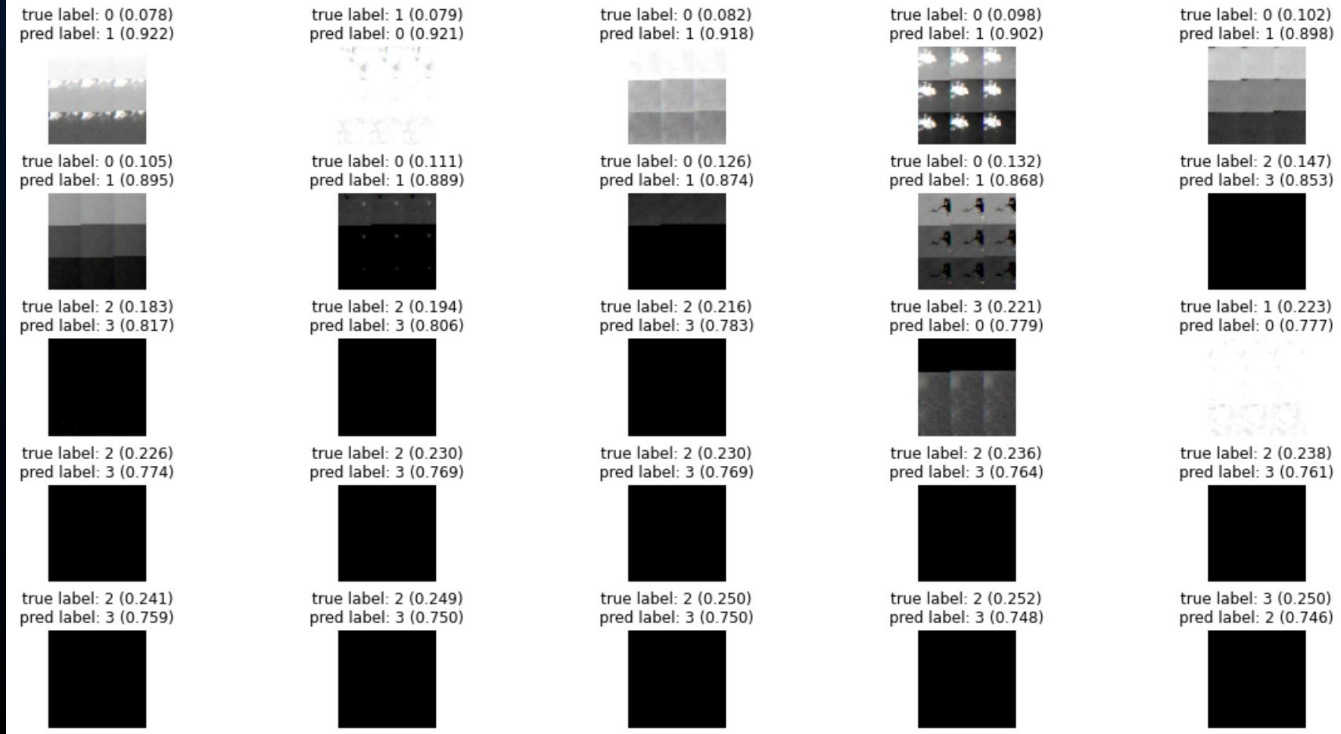
```
print(f'Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}%')
```

```
Test Loss: 0.265 | Test Acc: 88.26%
```



Plot incorrectly predicted label

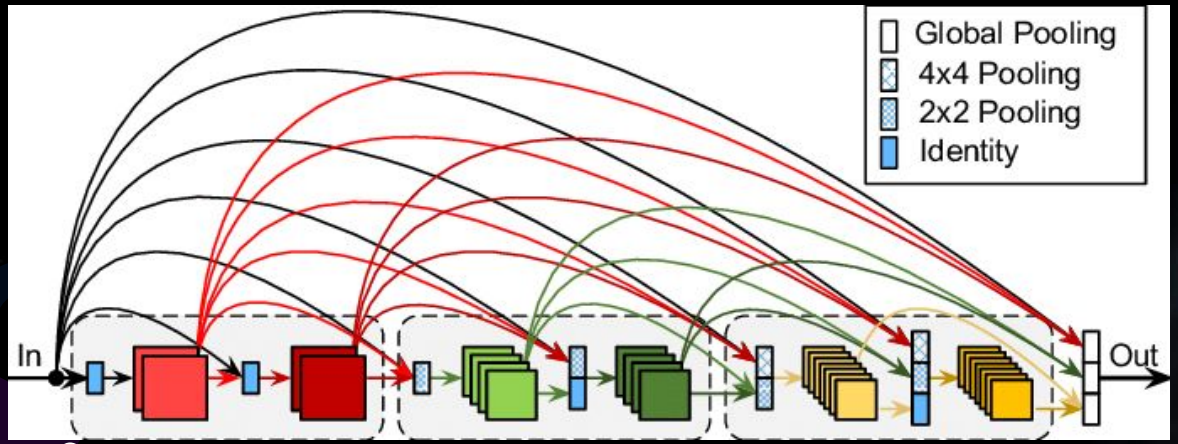
```
N_IMAGES = 25

plot_most_incorrect(incorrect_examples, N_IMAGES)
```



Implementation of DenseNet

What is DenseNet?

Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion.



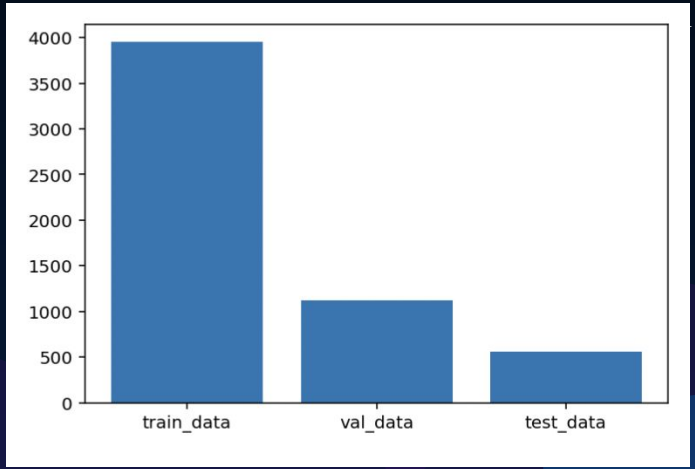
Implementation of DenseNet

01 Data processing

- **Define transform with known stats**
- **Load dataset with transform**
- **Split data into Training data, Validation data and Testing data in ratio of 7:2:1**
- **Define data loader with batch size of 64**

```
Split data to training, val and test in 7:2:1
Num training images:  3950
Num validating images:  1120
Num test images:  561
```

Bar chart of data after splitting



Implementation of DenseNet

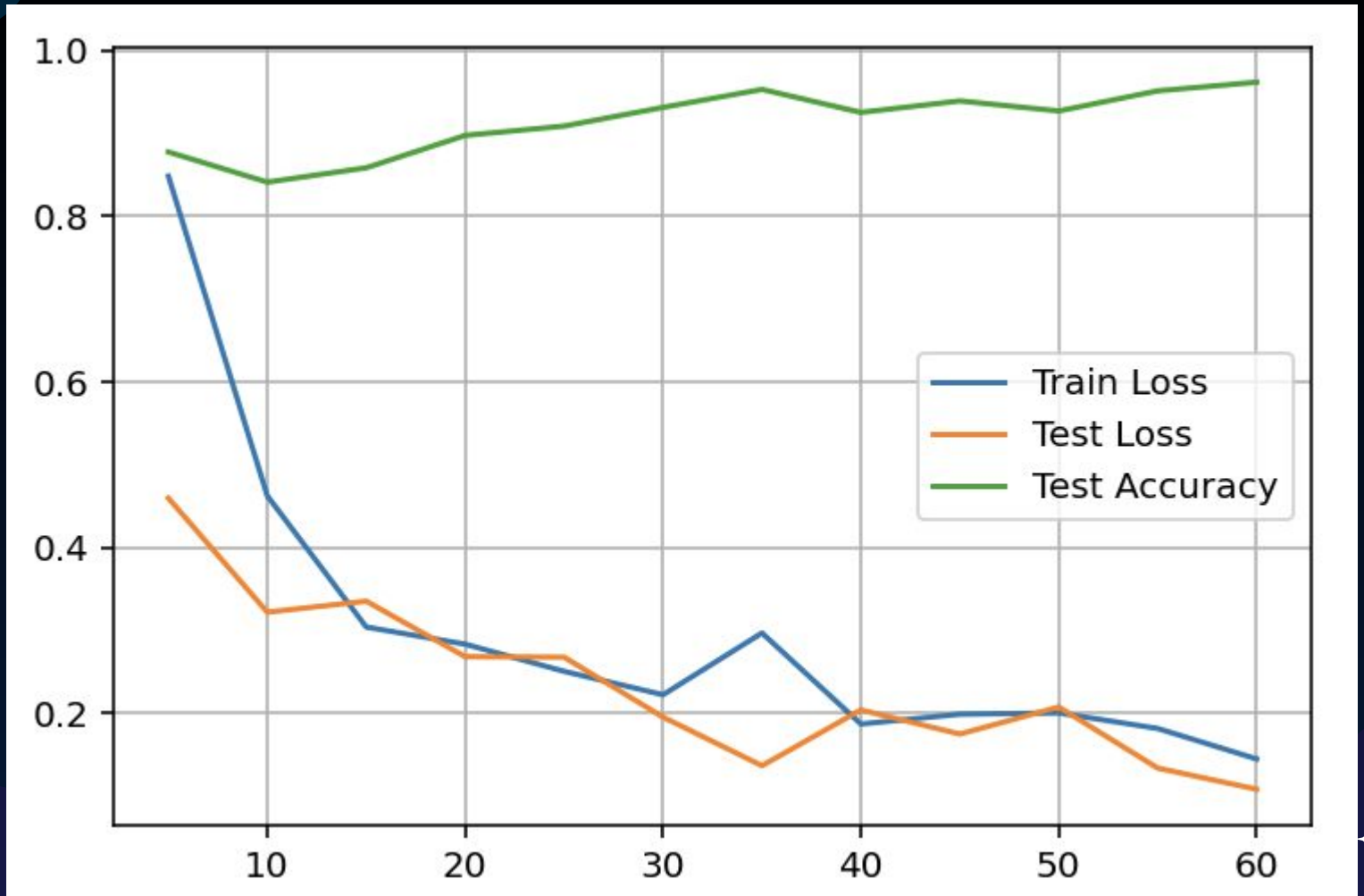
02 Define Model

Model will be a neural network, DenseNet-121. 121 means the depth of each layer in Dense Block.

Using the pytorch.nn module define the neural network using relu activation functions, and softmax.

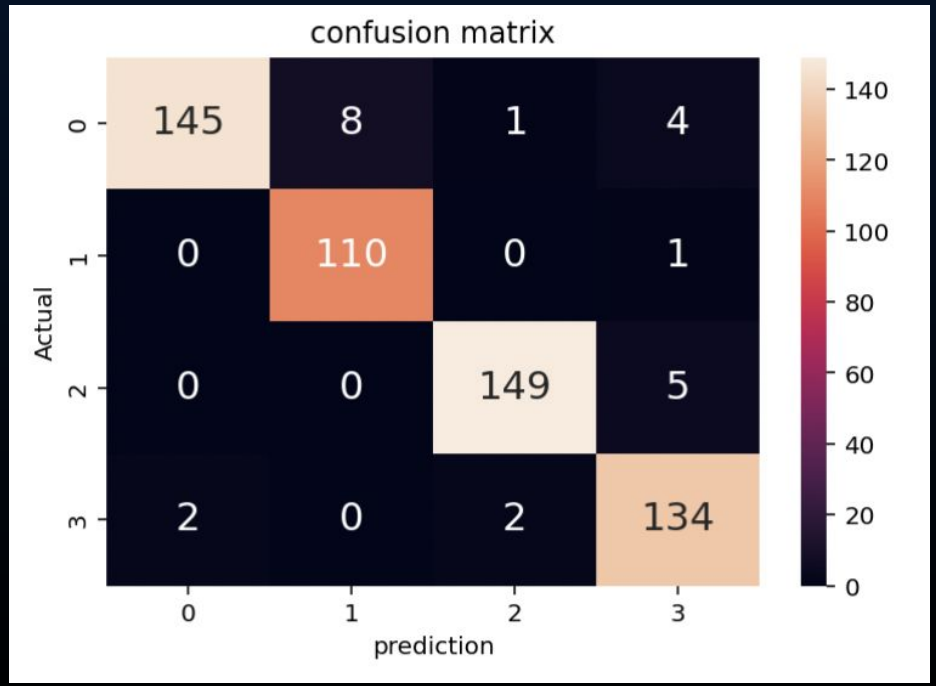
```
#print out the model architecture
model = models.densenet121(pretrained=True)
model
```

```
model.classifier = nn.Sequential(nn.Linear(1024, 512),
                                 nn.ReLU(),
                                 nn.Dropout(0.2),
```

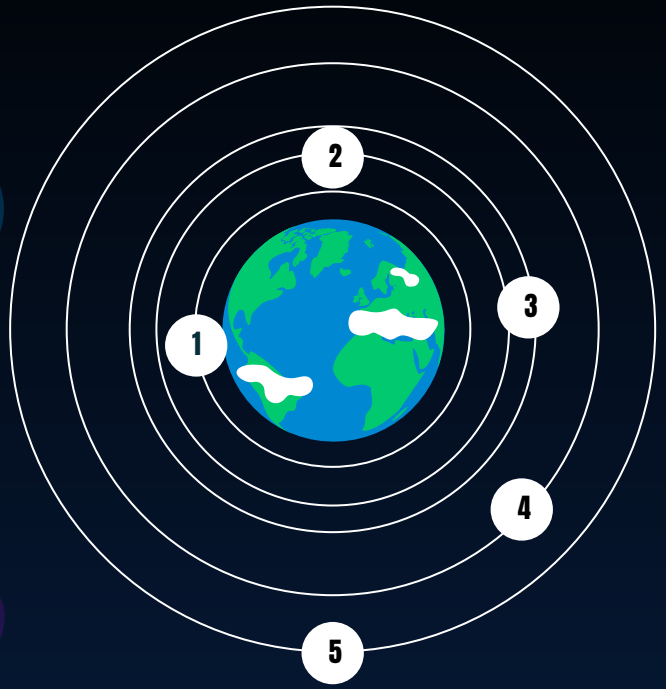

1.0
0.8
0.6
0.4
0.2
10
20
30
40
50
60
Train Loss
Test Loss
Test Accuracy

Implementation of DenseNet

04 Final result on testing



Conclusion



1. **The testing results on MLP is :**

   Error: 0.265 Accuracy: 88.26%

2. **The testing results on DenseNet is:**

   Error: 0.123 Accuracy: 96.2%

3. **Conclusion**

   This project explained the process of predicting a satellite image class with the pytorch library, with a comparison of MLP and a special model DenseNet in CNN.

Contribution

Wong Kai Yuan

DC026157

- **Implementation on MLP**
- **50% of Report**
- **50% of Presentation PPT**

Guan Jia Xi

CC029721

- **Implementation on DenseNet**
- **50% of Report**
- **50% of Presentation PPT**

We both work on each part, and integrate together during writing report and PPT.

THANKS!

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**