Folder structure

In the folder, there are
Four jupyter notebooks(.ipynb),
Four folders
a readme,
a csv table,
a report.

Four jupyter notebooks(.ipynb):
Model.ipynb: This is the main code of the model
Proving that the cross entropy is ok.ipynb: This is the code to prove that
using y_hat-y in cross-entropy function as the output delta and then skip the backward of softmax,
using -y/y_hat(the derivative of cross-entropy function) and then via the backward of softmax, have same result.
dataset prepare.ipynb: This is to transform labels to one-hot encode and separate test dataset to test and validation.
Best Model to predict test dataset.ipynb: This dataset is to run the Dataset of Assignment1. To be specific, the raw dataset.

Four folders:
Pic: The folder to save the output pictures (loss and accuracy). The index of picture is related to the csv table.
Assignment1-Dataset: The raw dataset from this assignment
Train-Test-Val: The dataset from dataset prepare.ipynb, including train, test and validation datasets
Prediction of Test dataset: This folder includes Test prediction.csv, Test result.csv and two output pictures about loss and accuracy. Test prediction.csv records the label and the prediction. Test result.csv includes time cost and loss and more.

A readme:
This file. README.pdf

A csv table:
analyse.csv includes the index of analyse, the description of change and the output results. The index in the table maybe different to the report because we change it in report to make sure it is more readable.

A report:
490429716_530210018.pdf
The final report to show our work.

IMPORTANT:

1. Version: Python 3.9.16/ numpy 1.23.5/ pandas 1.5.3/ matplotlib 3.7.0/
2. Before all, running dataset prepare.ipynb first.
3. Before running the Model.ipynb, please check the dataset load part first.

   If you want to run with train, test and validation. You do not need to change anything.

   If you just want to run the raw dataset(Train, Test), I suggest to run the Best Model to predict test dataset.ipynb. Or using the second cell.

```
# # load raw dataset
# train_data=np.load('Assignment1-Dataset\\train_data.npy')
# train_label=np.load('Assignment1-Dataset\\train_label.npy')
# valid_data=np.load('Assignment1-Dataset\\test_data.npy')
# valid_label=np.load('Assignment1-Dataset\\test_label.npy')
# Transforming label from number to one-hot
# new_test_label = np.zeros([len(valid_label), 10])
# i = 0
# for label in valid_label:
#     new_test_label[i, label] = 1
#     i = i+1
# valid_label = new_test_label
# new_train_label = np.zeros([len(train_label), 10])
# i = 0
# for label in train_label:
#     new_train_label[i, label] = 1
#     i = i+1
# train_label = new_train_label
```

Then, annotating the third cell

```
# load dataset
test_data=np.load('Train-Test-Val\\test_data.npy')
test_label=np.load('Train-Test-Val\\test_label.npy')
train_data=np.load('Train-Test-Val\\train_data.npy')
train_label=np.load('Train-Test-Val\\train_label.npy')
valid_data=np.load('Train-Test-Val\\valid_data.npy')
valid_label=np.load('Train-Test-Val\\valid_label.npy')
```

Then, annotating the first line in forth cell

```
test_data = (test_data - np.min(test_data)+1e-4) / \
    (0.1*(np.max(test_data)-np.min(test_data)))
```

Then, do not running the cell after fifth from the bottom cell. To be specific, do not run the code after plotting