In [2]:

```
!pip install numpy pandas matplotlib seaborn plotly requests tqdm opencv-python
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: numpy in /environment/miniconda3/lib/python3.10/si
te-packages (1.24.1)
Requirement already satisfied: pandas in /environment/miniconda3/lib/python3.10/s
ite-packages (2.1.2)
Requirement already satisfied: matplotlib in /environment/miniconda3/lib/python3.
10/site-packages (3.8.1)
Requirement already satisfied: seaborn in /environment/miniconda3/lib/python3.10/
site-packages (0.13.0)
Requirement already satisfied: plotly in /environment/miniconda3/lib/python3.10/s
ite-packages (5.19.0)
Requirement already satisfied: requests in /environment/miniconda3/lib/python3.1
0/site-packages (2.31.0)
Requirement already satisfied: tqdm in /environment/miniconda3/lib/python3.10/sit
e-packages (4.65.0)
Requirement already satisfied: opencv-python in /environment/miniconda3/lib/pytho
n3.10/site-packages (4.8.1.78)
Requirement already satisfied: pillow in /environment/miniconda3/lib/python3.10/s
ite-packages (9.3.0)
Requirement already satisfied: wandb in /environment/miniconda3/lib/python3.10/si
te-packages (0.16.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /environment/miniconda3/
lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /environment/miniconda3/lib/python
3.10/site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /environment/miniconda3/lib/pyth
on3.10/site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in /environment/miniconda3/lib/py
thon3.10/site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /environment/miniconda3/lib/python
3.10/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /environment/miniconda3/lib/p
ython3.10/site-packages (from matplotlib) (4.44.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /environment/miniconda3/lib/p
ython3.10/site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from matplotlib) (23.0)
Requirement already satisfied: pyparsing>=2.3.1 in /environment/miniconda3/lib/py
thon3.10/site-packages (from matplotlib) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from plotly) (8.2.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /environment/miniconda
3/lib/python3.10/site-packages (from requests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /environment/miniconda3/lib/python
3.10/site-packages (from requests) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /environment/miniconda3/lib/
python3.10/site-packages (from requests) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /environment/miniconda3/lib/
python3.10/site-packages (from requests) (2023.7.22)
Requirement already satisfied: Click!=8.0.0,>=7.1 in /environment/miniconda3/lib/
python3.10/site-packages (from wandb) (7.1.2)
Requirement already satisfied: GitPython!=3.1.29,>=1.0.0 in /environment/minicond
a3/lib/python3.10/site-packages (from wandb) (3.1.42)
Requirement already satisfied: psutil>=5.0.0 in /environment/miniconda3/lib/pytho
n3.10/site-packages (from wandb) (5.9.5)
Requirement already satisfied: sentry-sdk>=1.0.0 in /environment/miniconda3/lib/p
ython3.10/site-packages (from wandb) (1.40.5)
Requirement already satisfied: docker-pycreds>=0.4.0 in /environment/miniconda3/l
ib/python3.10/site-packages (from wandb) (0.4.0)
Requirement already satisfied: PyYAML in /environment/miniconda3/lib/python3.10/s
```

```
ite-packages (from wandb) (6.0.1)
Requirement already satisfied: setproctitle in /environment/miniconda3/lib/python
3.10/site-packages (from wandb) (1.3.3)
Requirement already satisfied: setuptools in /environment/miniconda3/lib/python3.
10/site-packages (from wandb) (67.8.0)
Requirement already satisfied: appdirs>=1.4.3 in /environment/miniconda3/lib/pyth
on3.10/site-packages (from wandb) (1.4.4)
Requirement already satisfied: protobuf!=4.21.0,<5,>=3.19.0 in /environment/minic
onda3/lib/python3.10/site-packages (from wandb) (4.23.4)
Requirement already satisfied: six>=1.4.0 in /environment/miniconda3/lib/python3.
10/site-packages (from docker-pycreds>=0.4.0->wandb) (1.16.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from GitPython!=3.1.29,>=1.0.0->wandb) (4.0.11)
Requirement already satisfied: smmap<6,>=3.0.1 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from gitdb<5,>=4.0.1->GitPython!=3.1.29,>=1.0.0->wandb)
(5.0.1)
```

Download and install Pytorch

In [3]:

```
!pip3 install torch torchvision torchaudio --extra-index-url https://download.py
```

```
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple, https://download.py
torch.org/whl/cu113
Requirement already satisfied: torch in /environment/miniconda3/lib/python3.10/si
te-packages (2.0.1+cu118)
Requirement already satisfied: torchvision in /environment/miniconda3/lib/python
3.10/site-packages (0.15.2+cu118)
Requirement already satisfied: torchaudio in /environment/miniconda3/lib/python3.
10/site-packages (2.0.2+cu118)
Requirement already satisfied: filelock in /environment/miniconda3/lib/python3.1
0/site-packages (from torch) (3.9.0)
Requirement already satisfied: typing-extensions in /environment/miniconda3/lib/p
ython3.10/site-packages (from torch) (4.8.0)
Requirement already satisfied: sympy in /environment/miniconda3/lib/python3.10/si
te-packages (from torch) (1.11.1)
Requirement already satisfied: networkx in /environment/miniconda3/lib/python3.1
0/site-packages (from torch) (3.0)
Requirement already satisfied: jinja2 in /environment/miniconda3/lib/python3.10/s
ite-packages (from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /environment/miniconda3/lib/pytho
n3.10/site-packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /environment/miniconda3/lib/python3.10/si
te-packages (from triton==2.0.0->torch) (3.25.0)
Requirement already satisfied: lit in /environment/miniconda3/lib/python3.10/site
-packages (from triton==2.0.0->torch) (15.0.7)
Requirement already satisfied: numpy in /environment/miniconda3/lib/python3.10/si
te-packages (from torchvision) (1.24.1)
Requirement already satisfied: requests in /environment/miniconda3/lib/python3.1
0/site-packages (from torchvision) (2.31.0)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /environment/miniconda3/l
ib/python3.10/site-packages (from torchvision) (9.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /environment/miniconda3/lib/pyt
hon3.10/site-packages (from jinja2->torch) (2.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /environment/miniconda
3/lib/python3.10/site-packages (from requests->torchvision) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /environment/miniconda3/lib/python
3.10/site-packages (from requests->torchvision) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /environment/miniconda3/lib/
python3.10/site-packages (from requests->torchvision) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /environment/miniconda3/lib/
python3.10/site-packages (from requests->torchvision) (2023.7.22)
Requirement already satisfied: mpmath>=0.19 in /environment/miniconda3/lib/python
3.10/site-packages (from sympy->torch) (1.2.1)
```

In [4]:

```
!wget https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/20220716-mmclassif
```

```
--2024-02-26 16:29:54--  https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/
20220716-mmclassification/dataset/SimHei.ttf
Connecting to 172.16.0.13:5848... connected.
Proxy request sent, awaiting response... 200 OK
Length: 10050868 (9.6M) [application/x-font-ttf]
Saving to: 'SimHei.ttf.2'

SimHei.ttf.2        100%[===================>]   9.58M  16.7MB/s    in 0.6s

2024-02-26 16:29:55 (16.7 MB/s) - 'SimHei.ttf.2' saved [10050868/10050868]
```

Create a catalogue

In [5]:
```python
import os
```

In [6]:
```python
# Store the results file
# os.mkdir('output')

# Store the trained model weights
os.mkdir('checkpoint')


# Store the generated charts
os.mkdir('diagrams')
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
Cell In[6], line 5
      1 # Store the results file
      2 # os.mkdir('output')
      3
      4 # Store the trained model weights
----> 5 os.mkdir('checkpoint')
      8 # Store the generated charts
      9 os.mkdir('diagrams')

FileExistsError: [Errno 17] File exists: 'checkpoint'
```

Setting matplotlib Chinese and English fonts

In [ ]:
```python
## Font Environment Settings
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.font_manager import FontProperties

# global font settings
SimSun = FontProperties(fname='/home/featurize/SimHei.ttf')  # Used to display C
plt.rcParams['axes.unicode_minus'] = False  # Used to display the negative sign
Times_New_Roman = FontProperties(fname='/home/featurize/times.ttf')

# mixed font settings
config = {
#     "font.family":'serif',
#     "font.size": 80,
      "mathtext.fontset":'stix',
#     "font.serif": ['SimSun'],
}
rcParams.update(config)

#Canvas Settings
fig = plt.figure(num=1, figsize=(9, 6),dpi=180)
ax = plt.axes((0.23,0.23,0.6,0.6))


# Application of font effects
ax.set_title('中文宋体 $\mathrm{Times}$ $\mathrm{New}$ $\mathrm{Roman}$ $\mathrm
                                            ,fontproperties=SimSun,fontsize=12)

ax.set_xlabel('测试测试',fontproperties=SimSun,fontsize=12)

ax.set_ylabel('TestTest',fontproperties=Times_New_Roman,fontsize=12)
```

```
plt.show()
```

In [7]:
```
!sudo snap install tree
```

```
snap "tree" is already installed, see 'snap help refresh'
```

In [8]:
```
!tree /home/featurize/data -L 2
```

```
/home/featurize/data
├── train
│   ├── CherryTomatoes
│   ├── Mangosteen
│   ├── MomordicaCharantia
│   ├── NavelOrange
│   ├── Sandsugaroranges
│   ├── apple
│   ├── banana
│   ├── carrot
│   ├── cherries
│   ├── cucumber
│   ├── durian
│   ├── grape
│   ├── hamimelon
│   ├── kiwi
│   ├── lemon
│   ├── lichee
│   ├── longan
│   ├── mango
│   ├── pear
│   ├── pineapple
│   ├── pitaya
│   ├── pomegranate
│   ├── strawberry
│   ├── tomato
│   └── watermelon
└── val
    ├── Cherrytomatoes
    ├── Mangosteen
    ├── MomordicaCharantia
    ├── NavelOrange
    ├── Sandsugaroranges
    ├── apple
    ├── banana
    ├── carrot
    ├── cherries
    ├── cucumber
    ├── durian
    ├── grape
    ├── hamimelon
    ├── kiwi
    ├── lemon
    ├── lichee
    ├── longan
    ├── mango
    ├── pear
    ├── pineapple
    ├── pitaya
    ├── pomegranate
    ├── strawberry
    ├── tomato
    └── watermelon

52 directories, 0 files
```

change

In [ ]:

```python

```

In [9]:

```python
import time
import os
from tqdm import tqdm

import pandas as pd
import numpy as np

import torch
import torchvision
import torch.nn as nn
import torch.nn.functional as F

import matplotlib.pyplot as plt
%matplotlib inline


import warnings
warnings.filterwarnings("ignore")


device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print('device', device)
```

```
device cuda:0
```

In [11]:

```python
from torchvision import transforms

# Training Set Image Preprocessing - RCTN: Scaling, Cropping, Turn Tensor, Norma
train_transform = transforms.Compose([transforms.RandomResizedCrop(224),
                                      transforms.RandomHorizontalFlip(),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406]
                                     ])

# Test Set Image Preprocessing - RCTN: Scaling, Cropping, Turn Tensor, Normalisa
test_transform = transforms.Compose([transforms.Resize(256),
                                     transforms.CenterCrop(224),
                                     transforms.ToTensor(),
                                     transforms.Normalize(
                                         mean=[0.485, 0.456, 0.406],
                                         std=[0.229, 0.224, 0.225])
                                    ])
```

In [12]:

```python
# Dataset folder path
dataset_dir = '/home/featurize/data'
```

In [13]:

```python
train_path = os.path.join(dataset_dir, 'train')
test_path = os.path.join(dataset_dir, 'val')
print('Training_set_path', train_path)
print('Testing_set_path', test_path)
```

```
Training_set_path /home/featurize/data/train
Testing_set_path /home/featurize/data/val
```

In [14]:

```python
from torchvision import datasets

# Load training set
```

```python
train_dataset = datasets.ImageFolder(train_path, train_transform)

# Load Test Set
test_dataset = datasets.ImageFolder(test_path, test_transform)
```

In [15]:
```python
print('Number of images in the training set', len(train_dataset))
print('Number of categories', len(train_dataset.classes))
print('Name of each category', train_dataset.classes)
```

```
Number of images in the training set 3649
Number of categories 25
Name of each category ['CherryTomatoes', 'Mangosteen', 'MomordicaCharantia', 'Nav
elOrange', 'Sandsugaroranges', 'apple', 'banana', 'carrot', 'cherries', 'cucumbe
r', 'durian', 'grape', 'hamimelon', 'kiwi', 'lemon', 'lichee', 'longan', 'mango',
'pear', 'pineapple', 'pitaya', 'pomegranate', 'strawberry', 'tomato', 'watermelo
n']
```

In [16]:
```python
print('Number of test set images', len(test_dataset))
print('Number of categories', len(test_dataset.classes))
print('Name of each category', test_dataset.classes)
```

```
Number of test set images 898
Number of categories 25
Name of each category ['Cherrytomatoes', 'Mangosteen', 'MomordicaCharantia', 'Nav
elOrange', 'Sandsugaroranges', 'apple', 'banana', 'carrot', 'cherries', 'cucumbe
r', 'durian', 'grape', 'hamimelon', 'kiwi', 'lemon', 'lichee', 'longan', 'mango',
'pear', 'pineapple', 'pitaya', 'pomegranate', 'strawberry', 'tomato', 'watermelo
n']
```

In [17]:
```python
# Name of each category
class_names = train_dataset.classes
n_class = len(class_names)
```

In [18]:
```python
class_names
```

Out[18]:
```
['CherryTomatoes',
 'Mangosteen',
 'MomordicaCharantia',
 'NavelOrange',
 'Sandsugaroranges',
 'apple',
 'banana',
 'carrot',
 'cherries',
 'cucumber',
 'durian',
 'grape',
 'hamimelon',
 'kiwi',
 'lemon',
 'lichee',
 'longan',
 'mango',
 'pear',
 'pineapple',
 'pitaya',
 'pomegranate',
 'strawberry',
 'tomato',
 'watermelon']
```

In [19]:
```python
# Mapping relationship: category to index number
train_dataset.class_to_idx
```

Out[19]:
```
{'CherryTomatoes': 0,
 'Mangosteen': 1,
 'MomordicaCharantia': 2,
 'NavelOrange': 3,
 'Sandsugaroranges': 4,
 'apple': 5,
 'banana': 6,
 'carrot': 7,
 'cherries': 8,
 'cucumber': 9,
 'durian': 10,
 'grape': 11,
 'hamimelon': 12,
 'kiwi': 13,
 'lemon': 14,
 'lichee': 15,
 'longan': 16,
 'mango': 17,
 'pear': 18,
 'pineapple': 19,
 'pitaya': 20,
 'pomegranate': 21,
 'strawberry': 22,
 'tomato': 23,
 'watermelon': 24}
```

In [20]:
```python
# Mapping relationship: index number to category
idx_to_labels = {y:x for x,y in train_dataset.class_to_idx.items()}
```

In [21]:
```python
idx_to_labels
```

Out[21]:
```
{0: 'CherryTomatoes',
 1: 'Mangosteen',
 2: 'MomordicaCharantia',
 3: 'NavelOrange',
 4: 'Sandsugaroranges',
 5: 'apple',
 6: 'banana',
 7: 'carrot',
 8: 'cherries',
 9: 'cucumber',
 10: 'durian',
 11: 'grape',
 12: 'hamimelon',
 13: 'kiwi',
 14: 'lemon',
 15: 'lichee',
 16: 'longan',
 17: 'mango',
 18: 'pear',
 19: 'pineapple',
 20: 'pitaya',
 21: 'pomegranate',
 22: 'strawberry',
 23: 'tomato',
 24: 'watermelon'}
```

In [78]:
```
# Save as local npy file
np.save('idx_to_labels.npy', idx_to_labels)
np.save('labels_to_idx.npy', train_dataset.class_to_idx)
```

Define the data loader DataLoader

In [22]:
```
from torch.utils.data import DataLoader
```

In [23]:
```
BATCH_SIZE = 32

# Data loader for the training set
train_loader = DataLoader(train_dataset,
                          batch_size=BATCH_SIZE,
                          shuffle=True,
                          num_workers=4
                         )

# Data Loader for Test Sets
test_loader = DataLoader(test_dataset,
                         batch_size=BATCH_SIZE,
                         shuffle=False,
                         num_workers=4
                        )
```

View images and annotations for a batch

In [24]:
```
from torchvision import models
import torch.optim as optim
from torch.optim import lr_scheduler
```

Choosing a Transfer Learning Training Approach

In [25]:
```
model = models.resnet18(pretrained=False) # Load only the model structure, not t

model.fc = nn.Linear(model.fc.in_features, n_class)

optimizer = optim.Adam(model.parameters())
```

Training configuration

In [26]:
```
model = model.to(device)

# Cross Entropy Loss Function
criterion = nn.CrossEntropyLoss()

# Training rounds Epoch
EPOCHS = 30

# Learning rate reduction strategies
lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.5)
```

Functions: training on a training set

In [27]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
```

In [28]:
```
def train_one_batch(images, labels):
    '''
    Run a batch of training and return the training log for the current batch.
    '''

    # Get a batch of data and annotations
    images = images.to(device)
    labels = labels.to(device)

    outputs = model(images) # Input model to perform forward prediction
    loss = criterion(outputs, labels) # Calculate the average cross-entropy loss

    # Optimising update weights
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Get the label category and prediction category of the current batch
    _, preds = torch.max(outputs, 1) # Get the prediction categories for all ima
    preds = preds.cpu().numpy()
    loss = loss.detach().cpu().numpy()
    outputs = outputs.detach().cpu().numpy()
    labels = labels.detach().cpu().numpy()

    log_train = {}
    log_train['epoch'] = epoch
    log_train['batch'] = batch_idx
    # Calculation of disaggregated assessment indicators
    log_train['train_loss'] = loss
    log_train['train_accuracy'] = accuracy_score(labels, preds)
    # log_train['train_precision'] = precision_score(labels, preds, average='mac
    # log_train['train_recall'] = recall_score(labels, preds, average='macro')
    # log_train['train_f1-score'] = f1_score(labels, preds, average='macro')

    return log_train
```

Functions: Evaluate on the whole test set

In [29]:
```
def evaluate_testset():
    '''
    Evaluate on the entire test set and return a log of categorised evaluation m
    '''

    loss_list = []
    labels_list = []
    preds_list = []

    with torch.no_grad():
```

```
        for images, labels in test_loader: # Generate a batch of data and annota
            images = images.to(device)
            labels = labels.to(device)
            outputs = model(images) # Input model to perform forward prediction

            # Get label categories and prediction categories for the entire test
            _, preds = torch.max(outputs, 1) # Get the prediction categories for
            preds = preds.cpu().numpy()
            loss = criterion(outputs, labels) # From logit, calculate the averag
            loss = loss.detach().cpu().numpy()
            outputs = outputs.detach().cpu().numpy()
            labels = labels.detach().cpu().numpy()

            loss_list.append(loss)
            labels_list.extend(labels)
            preds_list.extend(preds)

    log_test = {}
    log_test['epoch'] = epoch

    # Calculation of disaggregated assessment indicators
    log_test['test_loss'] = np.mean(loss_list)
    log_test['test_accuracy'] = accuracy_score(labels_list, preds_list)
    log_test['test_precision'] = precision_score(labels_list, preds_list, averag
    log_test['test_recall'] = recall_score(labels_list, preds_list, average='mac
    log_test['test_f1-score'] = f1_score(labels_list, preds_list, average='macro

    return log_test
```

Before training starts, keep a log

In [30]:
```
epoch = 0
batch_idx = 0
best_test_accuracy = 0
```

In [33]:
```
# Training log - training sets
df_train_log = pd.DataFrame()
log_train = {}
log_train['epoch'] = 0
log_train['batch'] = 0
images, labels = next(iter(train_loader))
log_train.update(train_one_batch(images, labels))
df_train_log = df_train_log._append(log_train, ignore_index=True)
```

In [34]:
```
df_train_log
```

Out[34]:

	epoch	batch	train_loss	train_accuracy
0	0	0	3.3938172	0.03125

In [36]:
```
# Training log - test set
df_test_log = pd.DataFrame()
log_test = {}
log_test['epoch'] = 0
log_test.update(evaluate_testset())
df_test_log = df_test_log._append(log_test, ignore_index=True)
```

In [37]:
```
df_test_log
```

Out[37]:

	epoch	test_loss	test_accuracy	test_precision	test_recall	test_f1-score
0	0.0	3.697069	0.046771	0.022265	0.046261	0.01971

Create wandb visualisation project

In [ ]:

In [43]:
```
import wandb

wandb.init(project='fruit25', name=time.strftime('%m%d%H%M%S'))
```

```
wandb: Currently logged in as: 13048152760 (ual_student_xintianyin). Use `wandb l
ogin --relogin` to force relogin
```

Tracking run with wandb version 0.16.3

Run data is saved locally in `/home/featurize/wandb/run-20240226_165721-n415gyvo`

Syncing run **0226165721** to Weights & Biases (docs)

View project at https://wandb.ai/ual_student_xintianyin/fruit25

View run at https://wandb.ai/ual_student_xintianyin/fruit25/runs/n415gyvo

Out[43]: Display W&B run

Run the training

In [47]:
```
for epoch in range(1, EPOCHS+1):

    print(f'Epoch {epoch}/{EPOCHS}')

    ## training phase
    model.train()
    for images, labels in tqdm(train_loader): # Get a batch of data and annotati
        batch_idx += 1
        log_train = train_one_batch(images, labels)
        df_train_log = df_train_log._append(log_train, ignore_index=True)
        wandb.log(log_train)

    lr_scheduler.step()

    ## testing phase
    model.eval()
    log_test = evaluate_testset()
    df_test_log = df_test_log._append(log_test, ignore_index=True)
    wandb.log(log_test)

    # Save the latest best model files
    if log_test['test_accuracy'] > best_test_accuracy:
        # Delete old best model files (if any)
        old_best_checkpoint_path = 'checkpoint/best-{:.3f}.pth'.format(best_test
        if os.path.exists(old_best_checkpoint_path):
            os.remove(old_best_checkpoint_path)
        # Save the new best model file
```

```
        best_test_accuracy = log_test['test_accuracy']
        new_best_checkpoint_path = 'checkpoint/best-{:.3f}.pth'.format(log_test[
        torch.save(model, new_best_checkpoint_path)
        print('Save the new best model', 'checkpoint/best-{:.3f}.pth'.format(bes
        # best_test_accuracy = log_test['test_accuracy']

df_train_log.to_csv('TraininglogTrainingSets.csv', index=False)
df_test_log.to_csv('TrainingLogTestSet.csv', index=False)
```

```
Epoch 1/30
100%|██████████| 115/115 [00:05<00:00, 21.28it/s]
Epoch 2/30
100%|██████████| 115/115 [00:05<00:00, 21.56it/s]
Epoch 3/30
100%|██████████| 115/115 [00:05<00:00, 21.72it/s]
Epoch 4/30
100%|██████████| 115/115 [00:05<00:00, 21.84it/s]
Epoch 5/30
100%|██████████| 115/115 [00:05<00:00, 21.76it/s]
Epoch 6/30
100%|██████████| 115/115 [00:05<00:00, 22.13it/s]
Epoch 7/30
100%|██████████| 115/115 [00:05<00:00, 21.11it/s]
Save the new best model checkpoint/best-0.675.pth
Epoch 8/30
100%|██████████| 115/115 [00:05<00:00, 21.97it/s]
Epoch 9/30
100%|██████████| 115/115 [00:05<00:00, 21.55it/s]
Epoch 10/30
100%|██████████| 115/115 [00:05<00:00, 21.62it/s]
Epoch 11/30
100%|██████████| 115/115 [00:05<00:00, 21.24it/s]
Epoch 12/30
100%|██████████| 115/115 [00:05<00:00, 21.81it/s]
Epoch 13/30
100%|██████████| 115/115 [00:05<00:00, 21.90it/s]
Epoch 14/30
100%|██████████| 115/115 [00:05<00:00, 21.52it/s]
Epoch 15/30
100%|██████████| 115/115 [00:05<00:00, 21.98it/s]
Epoch 16/30
100%|██████████| 115/115 [00:05<00:00, 22.17it/s]
Epoch 17/30
100%|██████████| 115/115 [00:05<00:00, 21.83it/s]
Epoch 18/30
100%|██████████| 115/115 [00:05<00:00, 21.62it/s]
Epoch 19/30
100%|██████████| 115/115 [00:05<00:00, 21.59it/s]
Epoch 20/30
100%|██████████| 115/115 [00:05<00:00, 21.69it/s]
Epoch 21/30
100%|██████████| 115/115 [00:05<00:00, 21.71it/s]
Epoch 22/30
100%|██████████| 115/115 [00:05<00:00, 21.03it/s]
```

```
Epoch 23/30
100%|██████████| 115/115 [00:05<00:00, 21.54it/s]
Epoch 24/30
100%|██████████| 115/115 [00:05<00:00, 21.49it/s]
Epoch 25/30
100%|██████████| 115/115 [00:05<00:00, 21.65it/s]
Epoch 26/30
100%|██████████| 115/115 [00:05<00:00, 21.39it/s]
Epoch 27/30
100%|██████████| 115/115 [00:05<00:00, 21.52it/s]
Epoch 28/30
100%|██████████| 115/115 [00:05<00:00, 21.46it/s]
Epoch 29/30
100%|██████████| 115/115 [00:05<00:00, 21.00it/s]
Epoch 30/30
100%|██████████| 115/115 [00:05<00:00, 21.69it/s]
```

Evaluation on the test set

In [45]:

```
# Load the best model as the current model
model = torch.load('checkpoint/best-{:.3f}.pth'.format(best_test_accuracy))
```

Evaluate on a test set

In [46]:

```
model.eval()
print(evaluate_testset())
```

```
{'epoch': 30, 'test_loss': 1.0977473, 'test_accuracy': 0.6648106904231625, 'test_precision': 0.6735762147736243, 'test_recall': 0.66314819409647, 'test_f1-score': 0.6627346390295513}
```