SimCLR 性別預測

- 資料準備
- 配置管理
- SinCLR 自監督預訓練
- 下游任務訓練
- 結果分析
- F1 Score

資料準備

資料目錄結構

建立 data/ 根目錄，包含 pre_train/ 和 downstream/ 子目錄

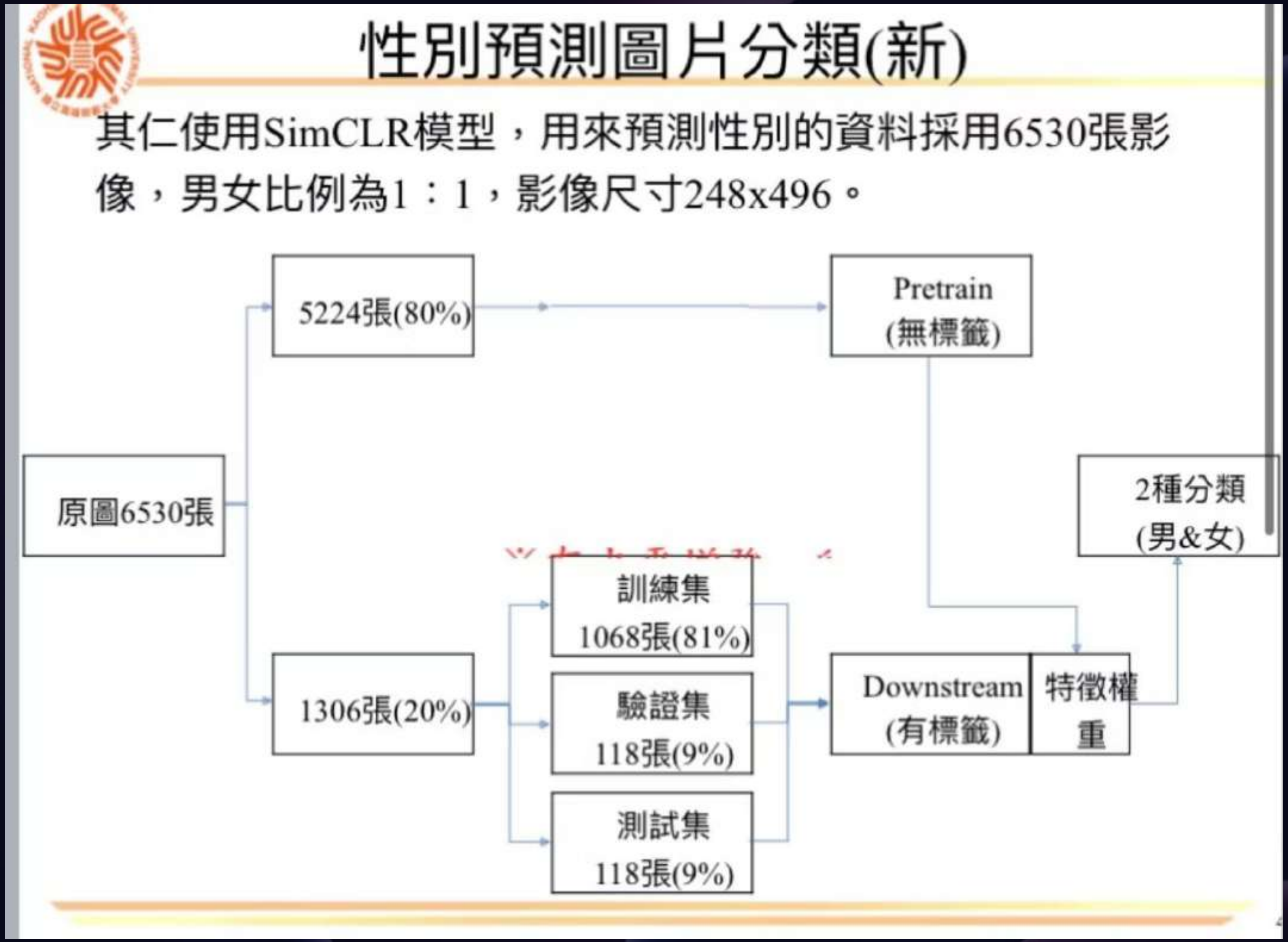
檔案命名規則

檔名格式為 <...>.png，可直接解析性別、年齡、切片編號

資料集分割

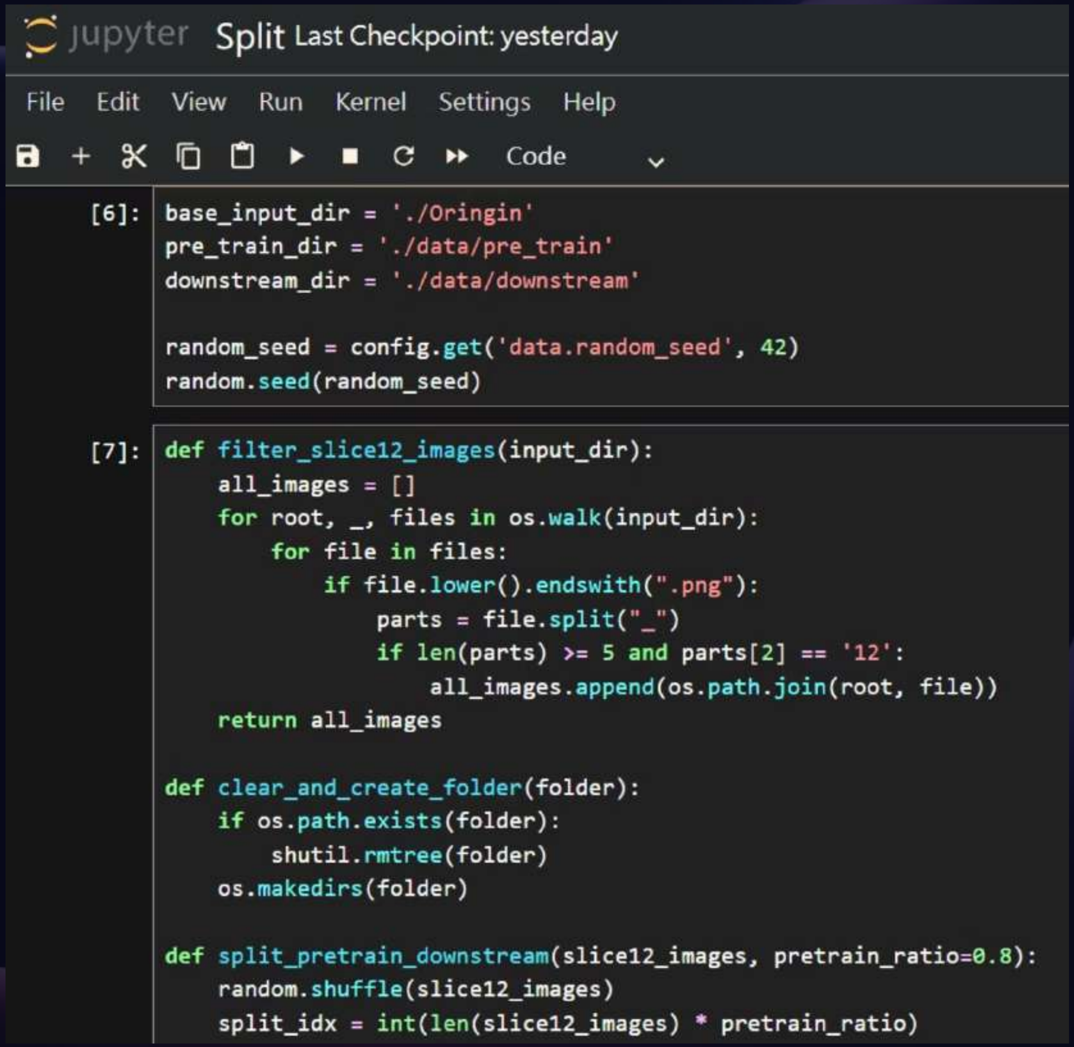
使用 train_test_split 劃分訓練/驗證集

資料準備



性別預測圖片分類(新)

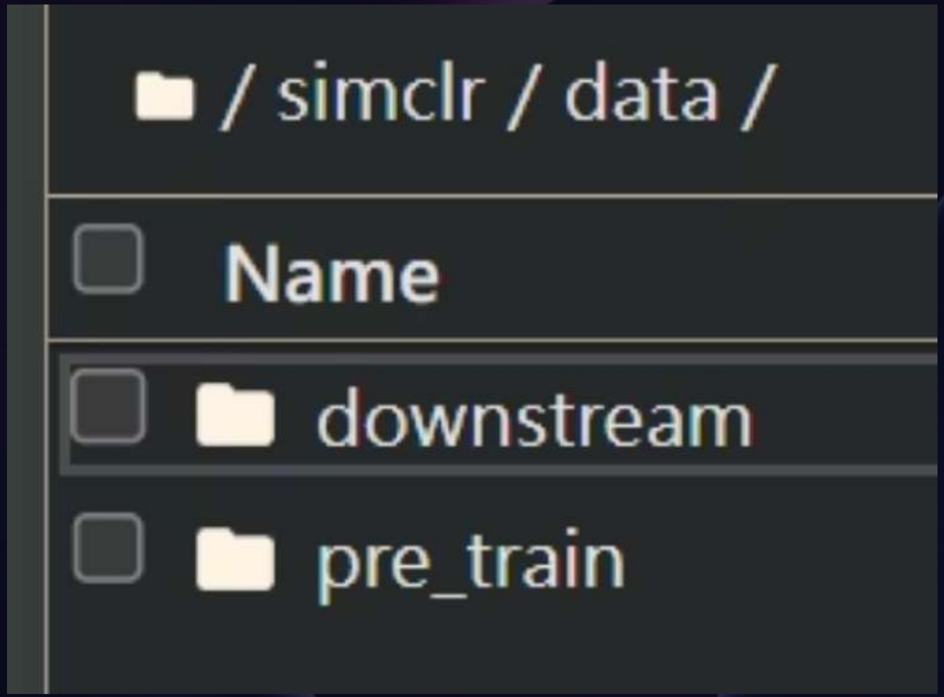
其仁使用SimCLR模型，用來預測性別的資料採用6530張影像，男女比例為1：1，影像尺寸248x496。

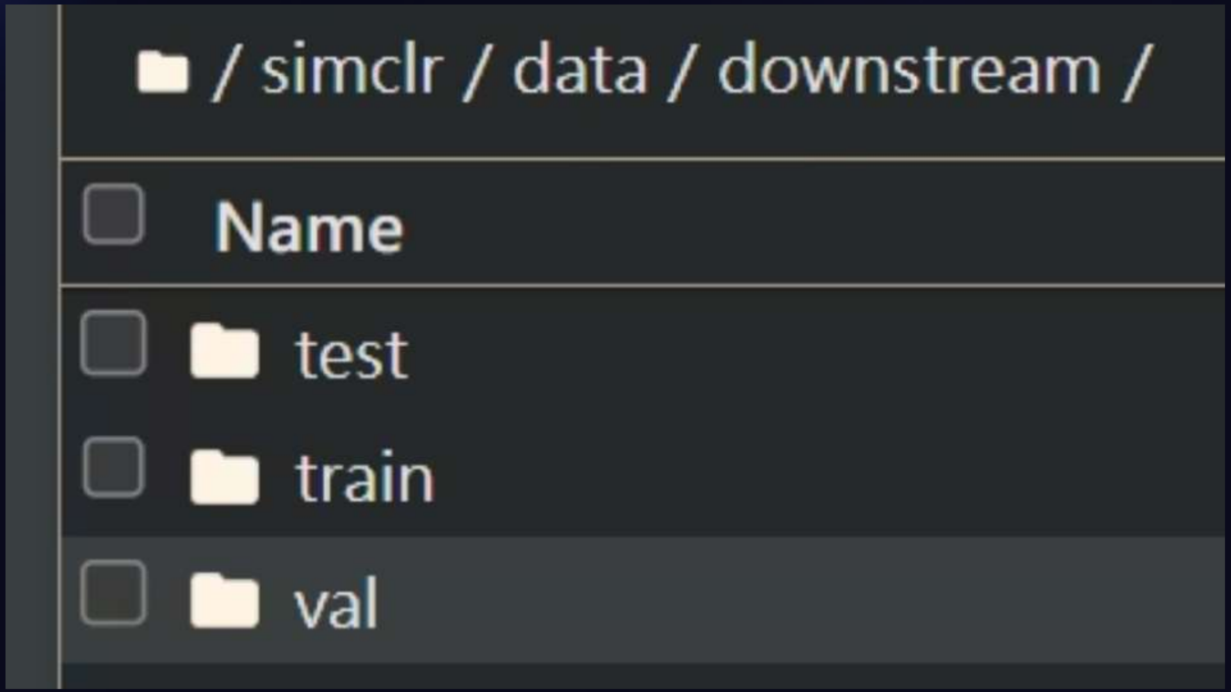


jupyter Split Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Code

```
[6]: base_input_dir = './Oringin'
     pre_train_dir = './data/pre_train'
     downstream_dir = './data/downstream'

     random_seed = config.get('data.random_seed', 42)
     random.seed(random_seed)

[7]: def filter_slice12_images(input_dir):
         all_images = []
         for root, _, files in os.walk(input_dir):
             for file in files:
                 if file.lower().endswith(".png"):
                     parts = file.split("_")
                     if len(parts) >= 5 and parts[2] == '12':
                         all_images.append(os.path.join(root, file))
         return all_images

     def clear_and_create_folder(folder):
         if os.path.exists(folder):
             shutil.rmtree(folder)
         os.makedirs(folder)

     def split_pretrain_downstream(slice12_images, pretrain_ratio=0.8):
         random.shuffle(slice12_images)
         split_idx = int(len(slice12_images) * pretrain_ratio)
```

資料準備


/ simclr / data /
Name
downstream
pre_train


/ simclr / data / downstream /
Name
test
train
val

配置管理

config.yaml

集中定義所有超參數、路徑、augmentation 強度等。

ConfigManager

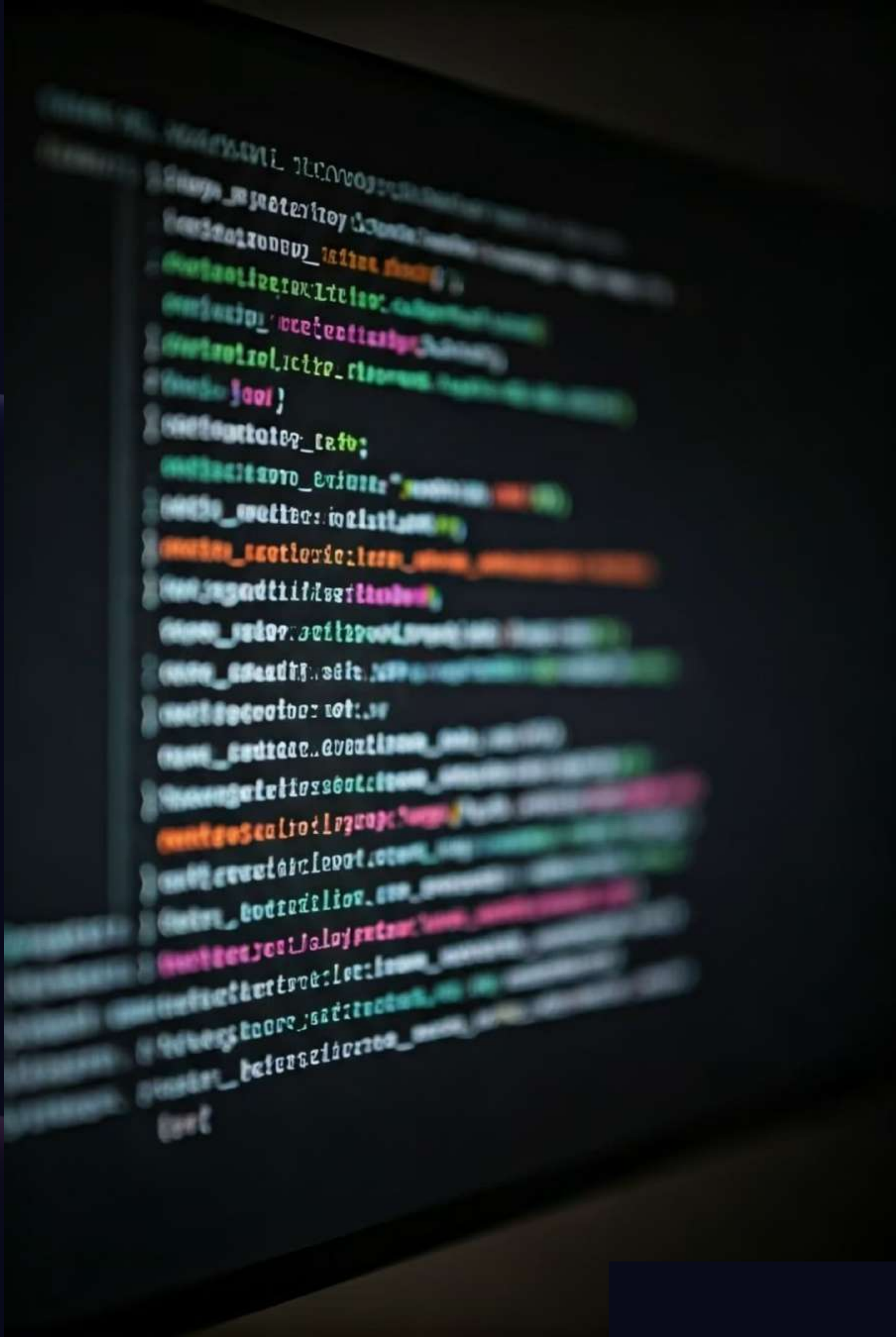
自動搜尋或建立實驗目錄，讀取、更新設定。

目錄結構

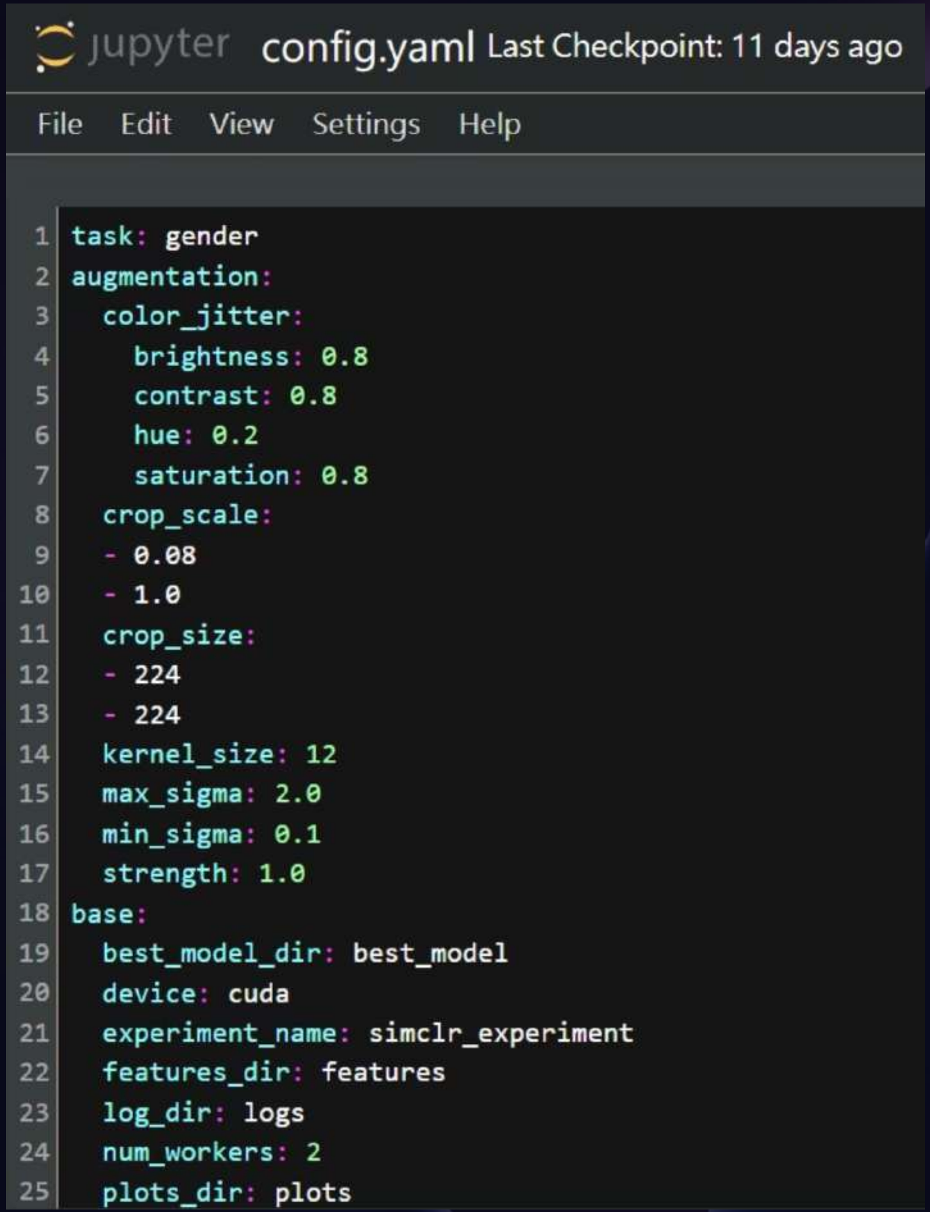
建立必要的子目錄：plots、logs、checkpoints、features。

日誌格式

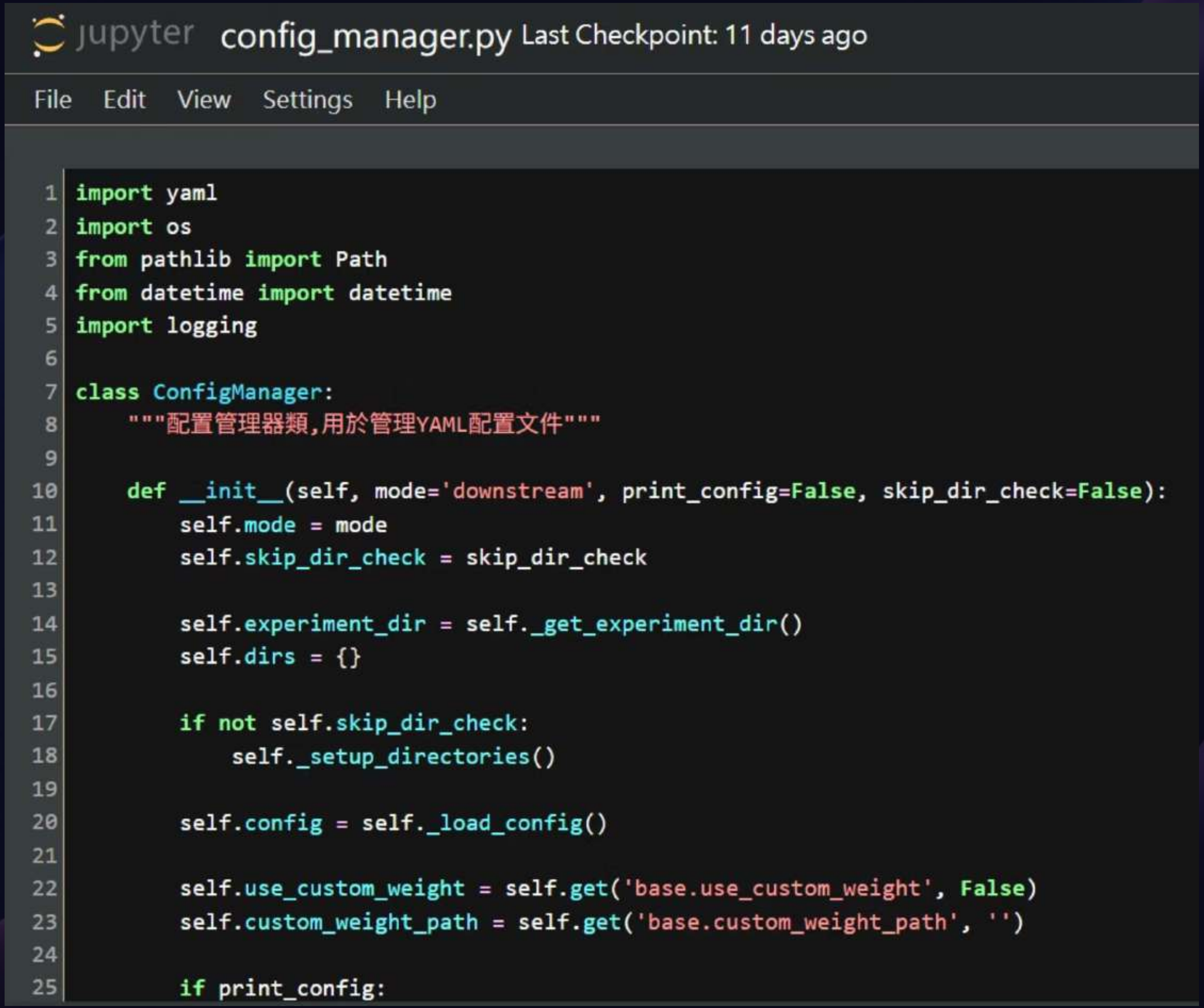
設定統一的日誌記錄格式。



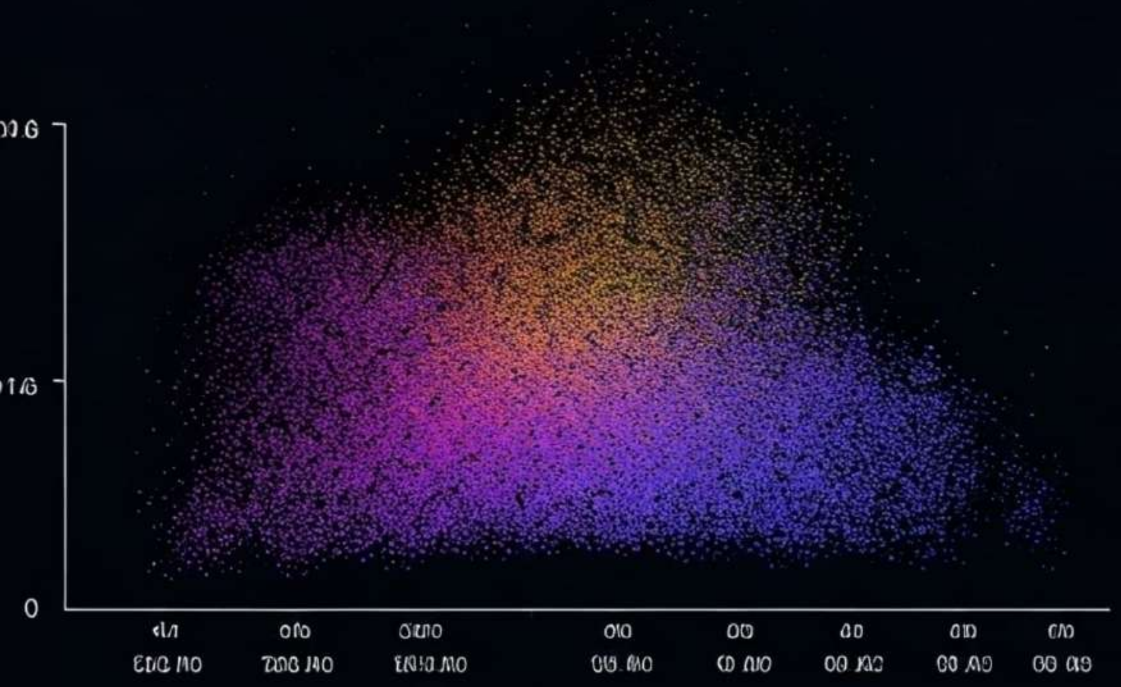
配置管理


jupyter config.yaml Last Checkpoint: 11 days ago
File Edit View Settings Help

```yaml
task: gender
augmentation:
  color_jitter:
    brightness: 0.8
    contrast: 0.8
    hue: 0.2
    saturation: 0.8
  crop_scale:
  - 0.08
  - 1.0
  crop_size:
  - 224
  - 224
  kernel_size: 12
  max_sigma: 2.0
  min_sigma: 0.1
  strength: 1.0
base:
  best_model_dir: best_model
  device: cuda
  experiment_name: simclr_experiment
  features_dir: features
  log_dir: logs
  num_workers: 2
  plots_dir: plots
```


jupyter config_manager.py Last Checkpoint: 11 days ago
File Edit View Settings Help

```python
import yaml
import os
from pathlib import Path
from datetime import datetime
import logging

class ConfigManager:
    """配置管理器類,用於管理YAML配置文件"""

    def __init__(self, mode='downstream', print_config=False, skip_dir_check=False):
        self.mode = mode
        self.skip_dir_check = skip_dir_check

        self.experiment_dir = self._get_experiment_dir()
        self.dirs = {}

        if not self.skip_dir_check:
            self._setup_directories()

        self.config = self._load_config()

        self.use_custom_weight = self.get('base.use_custom_weight', False)
        self.custom_weight_path = self.get('base.custom_weight_path', '')

        if print_config:
```



SimCLR 自監督預訓練

1. **資料集準備**

   SimCLRDummyDataset 載入影像，進行雙重增強，回傳正對比樣本對。

2. **模型結構**

   使用 ResNet18 作為 Backbone，加上兩層 MLP 投影頭。

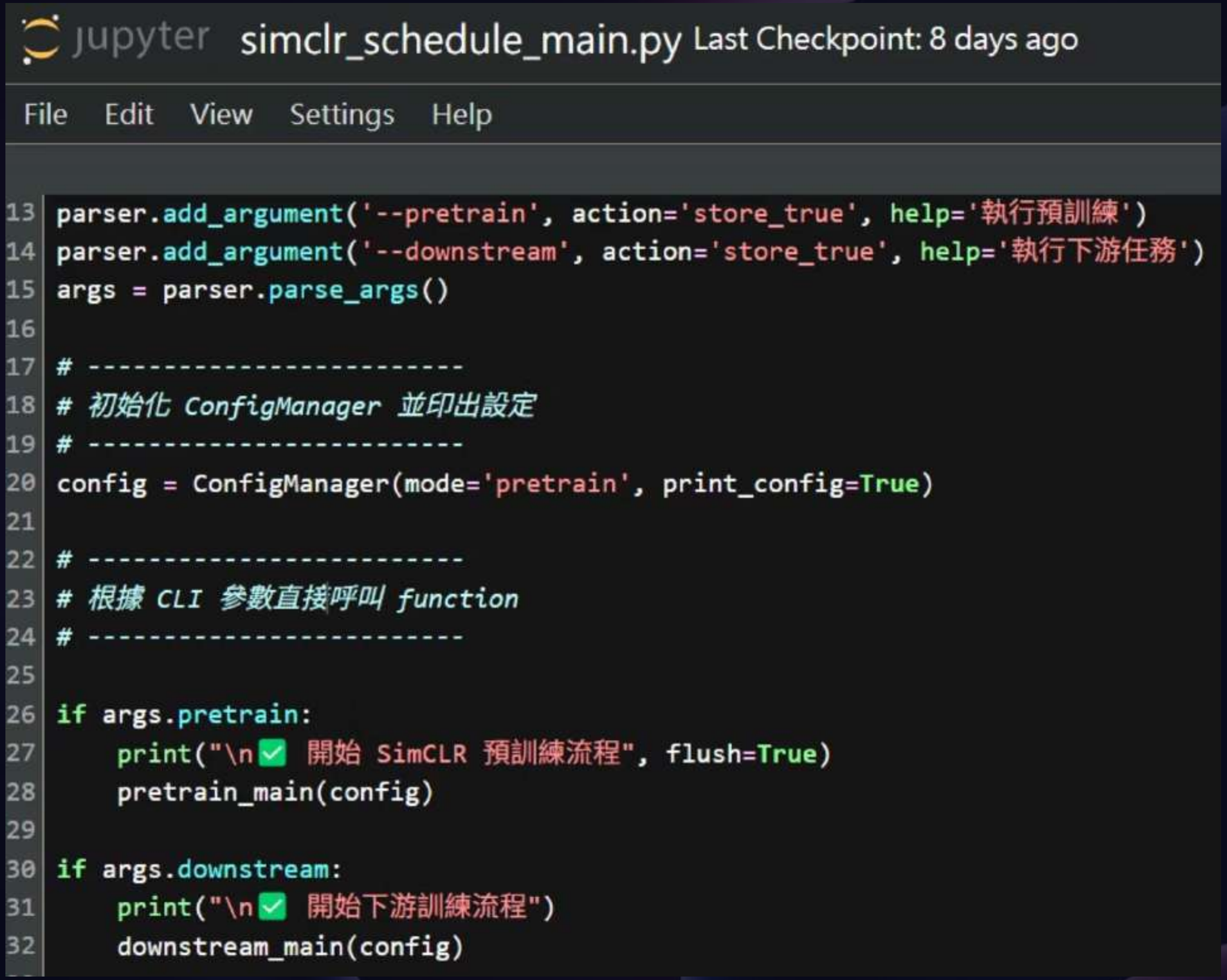
3. **損失函數**

   採用 NTXentLoss 計算對比損失。

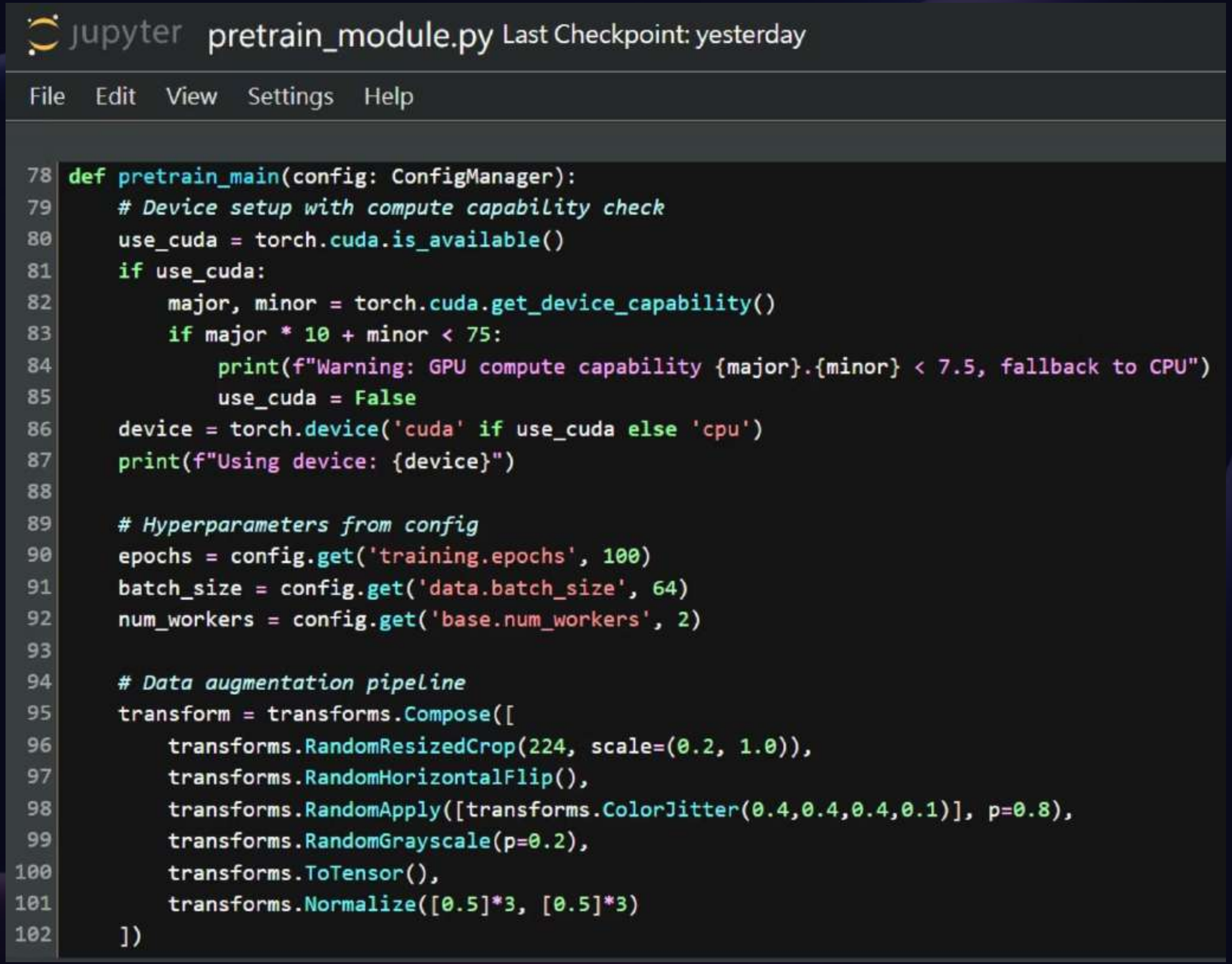
4. **訓練流程**

   支援批次大小排程、多 GPU 訓練、早停機制。

SimCLR 自監督預訓練

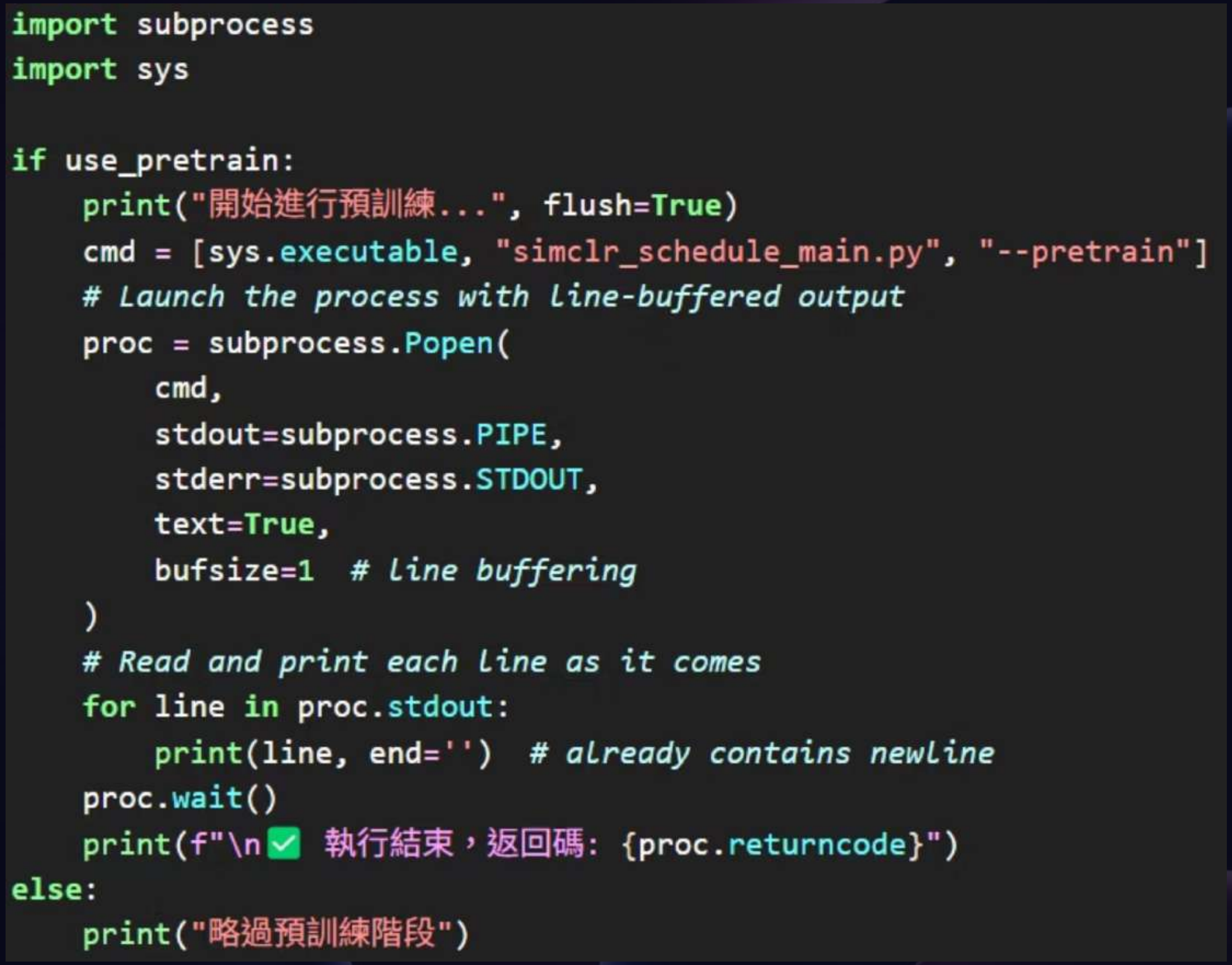

jupyter simclr_schedule_main.py Last Checkpoint: 8 days ago
File Edit View Settings Help

```python
parser.add_argument('--pretrain', action='store_true', help='執行預訓練')
parser.add_argument('--downstream', action='store_true', help='執行下游任務')
args = parser.parse_args()

# ------------------------
# 初始化 ConfigManager 並印出設定
# ------------------------
config = ConfigManager(mode='pretrain', print_config=True)

# ------------------------
# 根據 CLI 參數直接呼叫 function
# ------------------------

if args.pretrain:
    print("\n✅ 開始 SimCLR 預訓練流程", flush=True)
    pretrain_main(config)

if args.downstream:
    print("\n✅ 開始下游訓練流程")
    downstream_main(config)
```

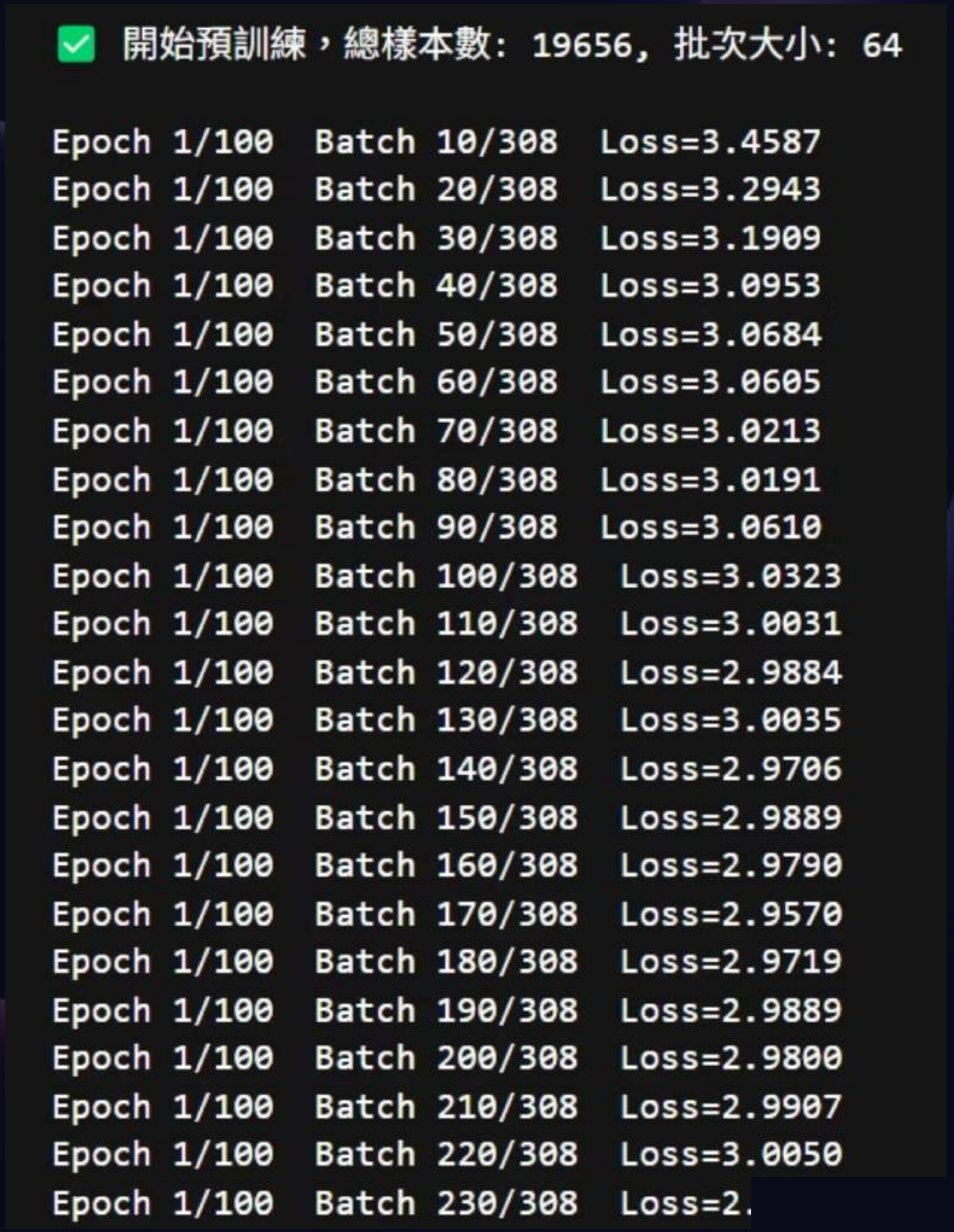

jupyter pretrain_module.py Last Checkpoint: yesterday
File Edit View Settings Help

```python
def pretrain_main(config: ConfigManager):
    # Device setup with compute capability check
    use_cuda = torch.cuda.is_available()
    if use_cuda:
        major, minor = torch.cuda.get_device_capability()
        if major * 10 + minor < 75:
            print(f"Warning: GPU compute capability {major}.{minor} < 7.5, fallback to CPU")
            use_cuda = False
    device = torch.device('cuda' if use_cuda else 'cpu')
    print(f"Using device: {device}")

    # Hyperparameters from config
    epochs = config.get('training.epochs', 100)
    batch_size = config.get('data.batch_size', 64)
    num_workers = config.get('base.num_workers', 2)

    # Data augmentation pipeline
    transform = transforms.Compose([
        transforms.RandomResizedCrop(224, scale=(0.2, 1.0)),
        transforms.RandomHorizontalFlip(),
        transforms.RandomApply([transforms.ColorJitter(0.4,0.4,0.4,0.1)], p=0.8),
        transforms.RandomGrayscale(p=0.2),
        transforms.ToTensor(),
        transforms.Normalize([0.5]*3, [0.5]*3)
    ])
```

SimCLR 自監督預訓練



```python
import subprocess
import sys

if use_pretrain:
    print("開始進行預訓練...", flush=True)
    cmd = [sys.executable, "simclr_schedule_main.py", "--pretrain"]
    # Launch the process with line-buffered output
    proc = subprocess.Popen(
        cmd,
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT,
        text=True,
        bufsize=1  # line buffering
    )
    # Read and print each line as it comes
    for line in proc.stdout:
        print(line, end='')  # already contains newline
    proc.wait()
    print(f"\n✅ 執行結束，返回碼: {proc.returncode}")
else:
    print("略過預訓練階段")
```



```
✅ 開始預訓練，總樣本數: 19656, 批次大小: 64

Epoch 1/100  Batch 10/308  Loss=3.4587
Epoch 1/100  Batch 20/308  Loss=3.2943
Epoch 1/100  Batch 30/308  Loss=3.1909
Epoch 1/100  Batch 40/308  Loss=3.0953
Epoch 1/100  Batch 50/308  Loss=3.0684
Epoch 1/100  Batch 60/308  Loss=3.0605
Epoch 1/100  Batch 70/308  Loss=3.0213
Epoch 1/100  Batch 80/308  Loss=3.0191
Epoch 1/100  Batch 90/308  Loss=3.0610
Epoch 1/100  Batch 100/308  Loss=3.0323
Epoch 1/100  Batch 110/308  Loss=3.0031
Epoch 1/100  Batch 120/308  Loss=2.9884
Epoch 1/100  Batch 130/308  Loss=3.0035
Epoch 1/100  Batch 140/308  Loss=2.9706
Epoch 1/100  Batch 150/308  Loss=2.9889
Epoch 1/100  Batch 160/308  Loss=2.9790
Epoch 1/100  Batch 170/308  Loss=2.9570
Epoch 1/100  Batch 180/308  Loss=2.9719
Epoch 1/100  Batch 190/308  Loss=2.9889
Epoch 1/100  Batch 200/308  Loss=2.9800
Epoch 1/100  Batch 210/308  Loss=2.9907
Epoch 1/100  Batch 220/308  Loss=3.0050
Epoch 1/100  Batch 230/308  Loss=2.
```

下游任務訓練

資料集準備

OCTFilenameDataset 依檔名抽取切片、年齡或性別作為標籤。

模型結構

載入 SimCLR encoder 權重，接一個小型 MLP 分類/回歸頭。

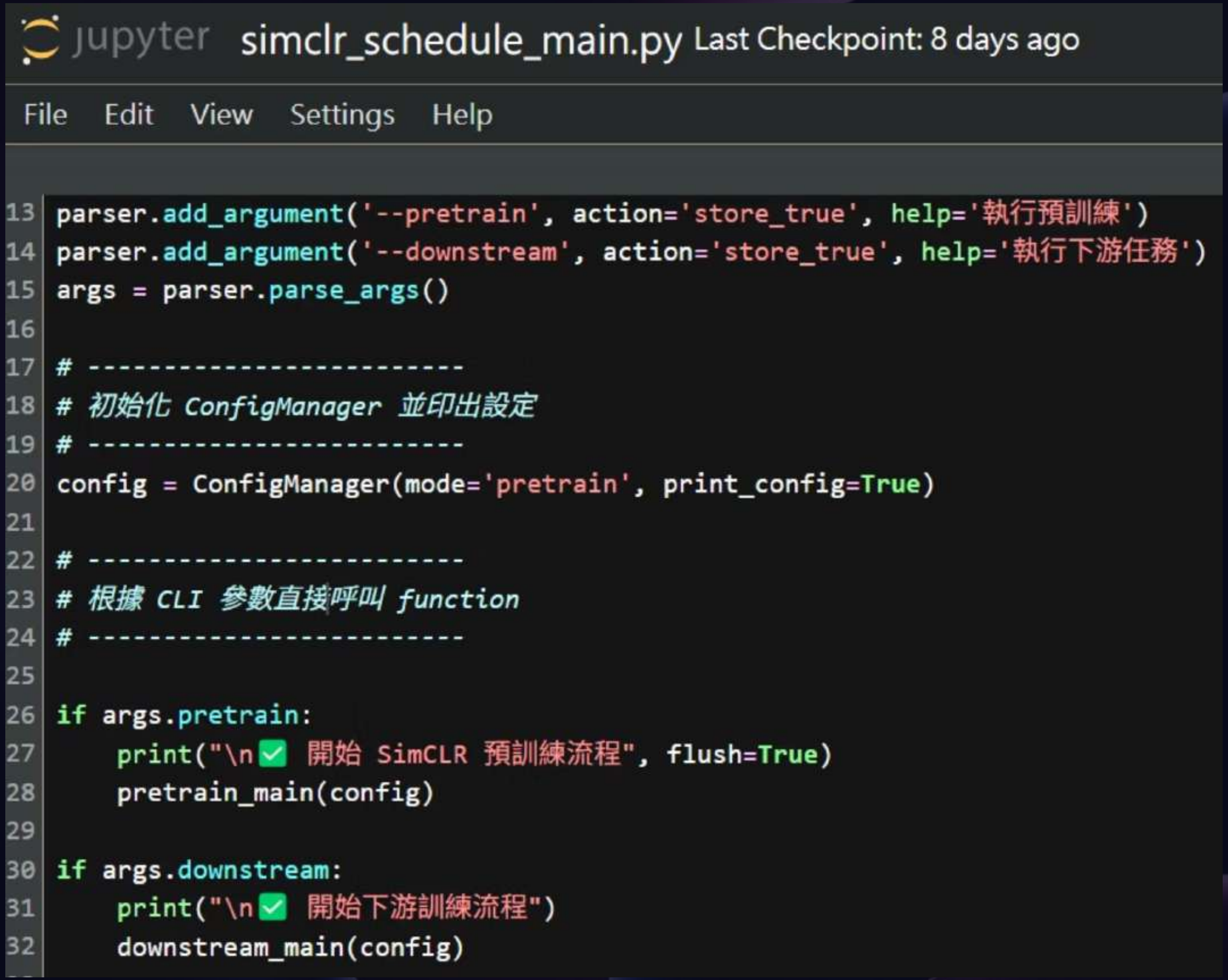
訓練與驗證

使用 CrossEntropyLoss 或 MSELoss，Adam 優化器。

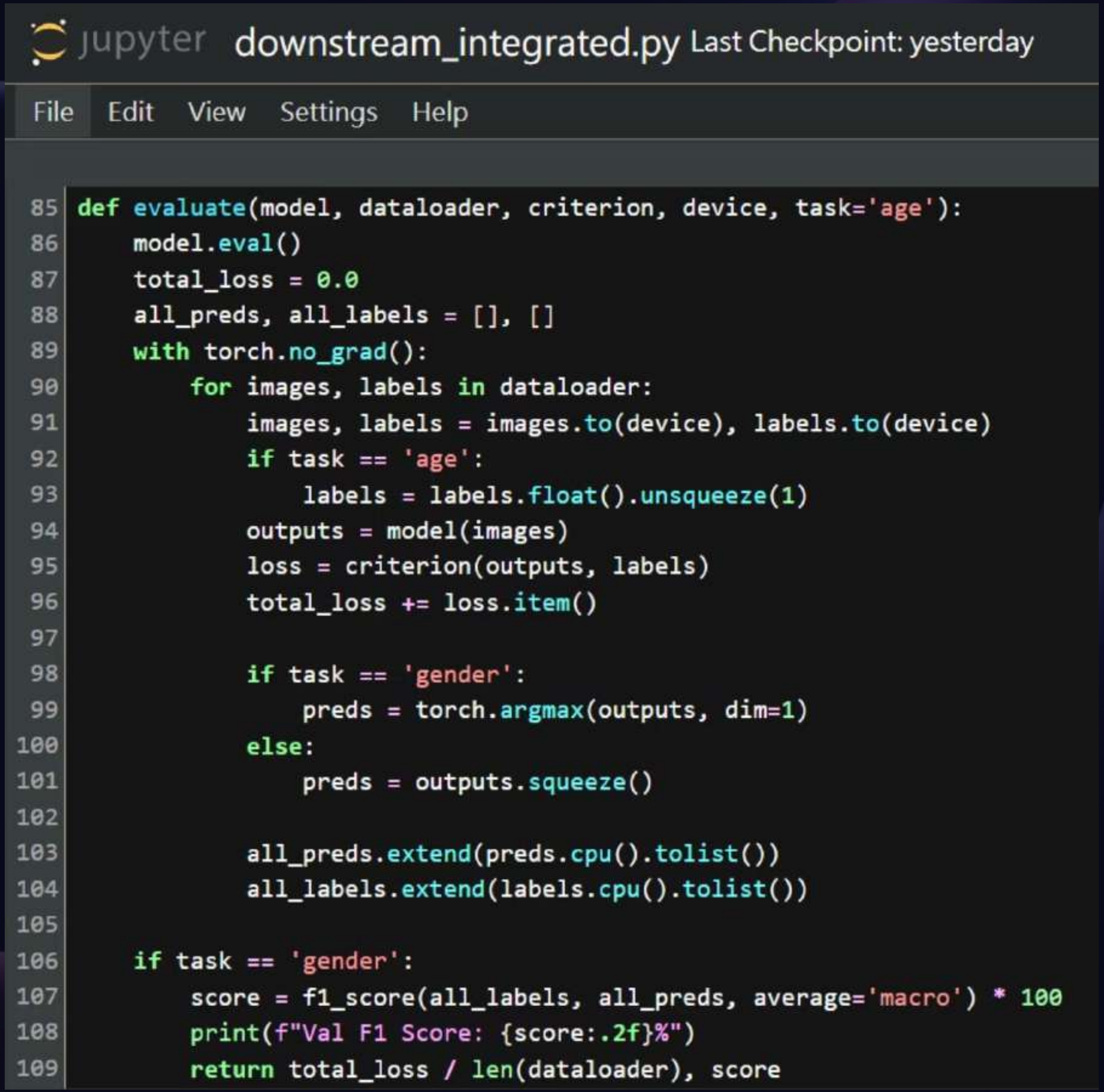
模型儲存

最佳模型儲存為 _best_model.pth，訓練記錄輸出到 CSV。

下游任務訓練

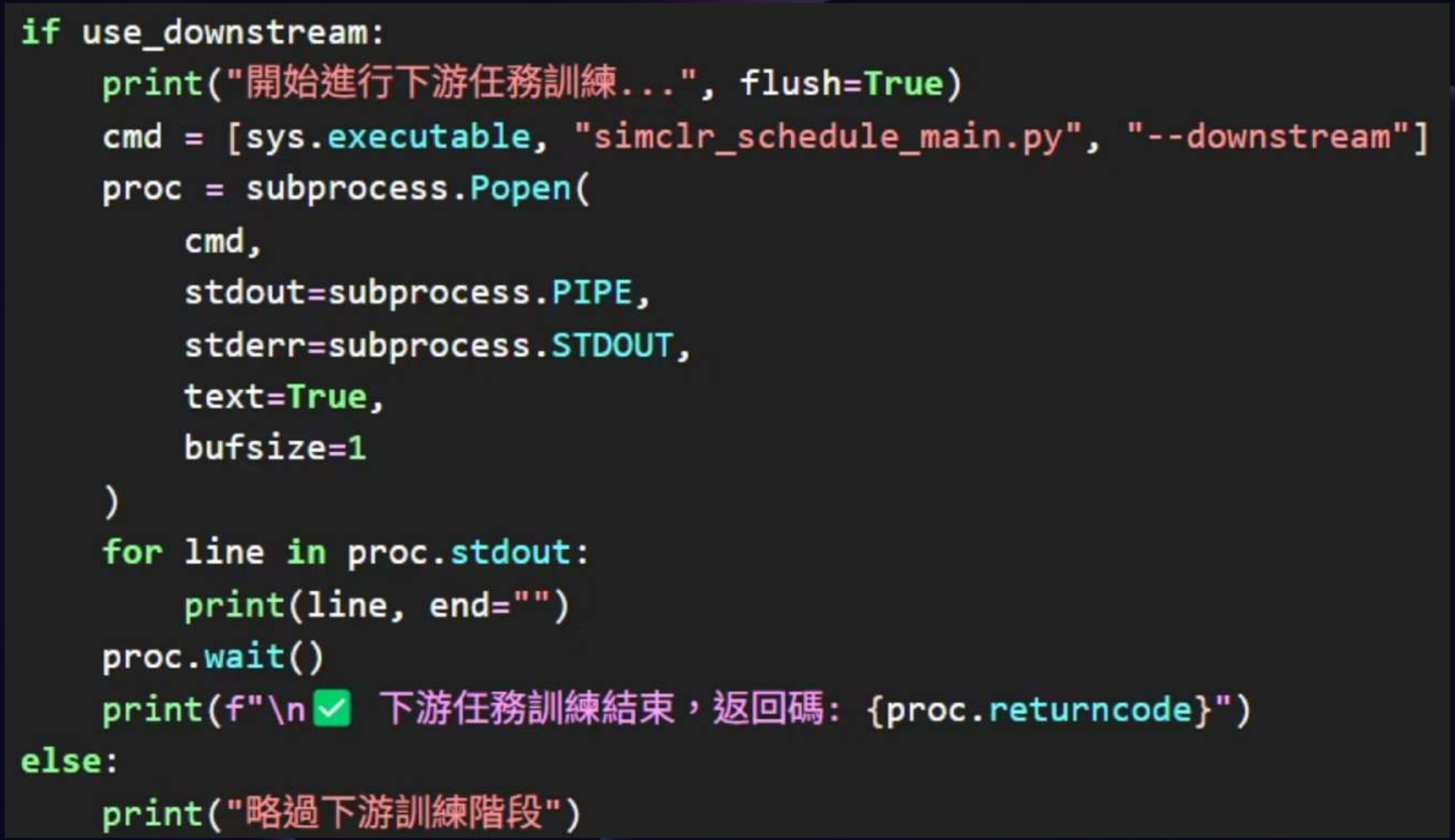

jupyter simclr_schedule_main.py Last Checkpoint: 8 days ago
File Edit View Settings Help

```python
parser.add_argument('--pretrain', action='store_true', help='執行預訓練')
parser.add_argument('--downstream', action='store_true', help='執行下游任務')
args = parser.parse_args()

# ------------------------
# 初始化 ConfigManager 並印出設定
# ------------------------
config = ConfigManager(mode='pretrain', print_config=True)

# ------------------------
# 根據 CLI 參數直接呼叫 function
# ------------------------

if args.pretrain:
    print("\n✅ 開始 SimCLR 預訓練流程", flush=True)
    pretrain_main(config)

if args.downstream:
    print("\n✅ 開始下游訓練流程")
    downstream_main(config)
```

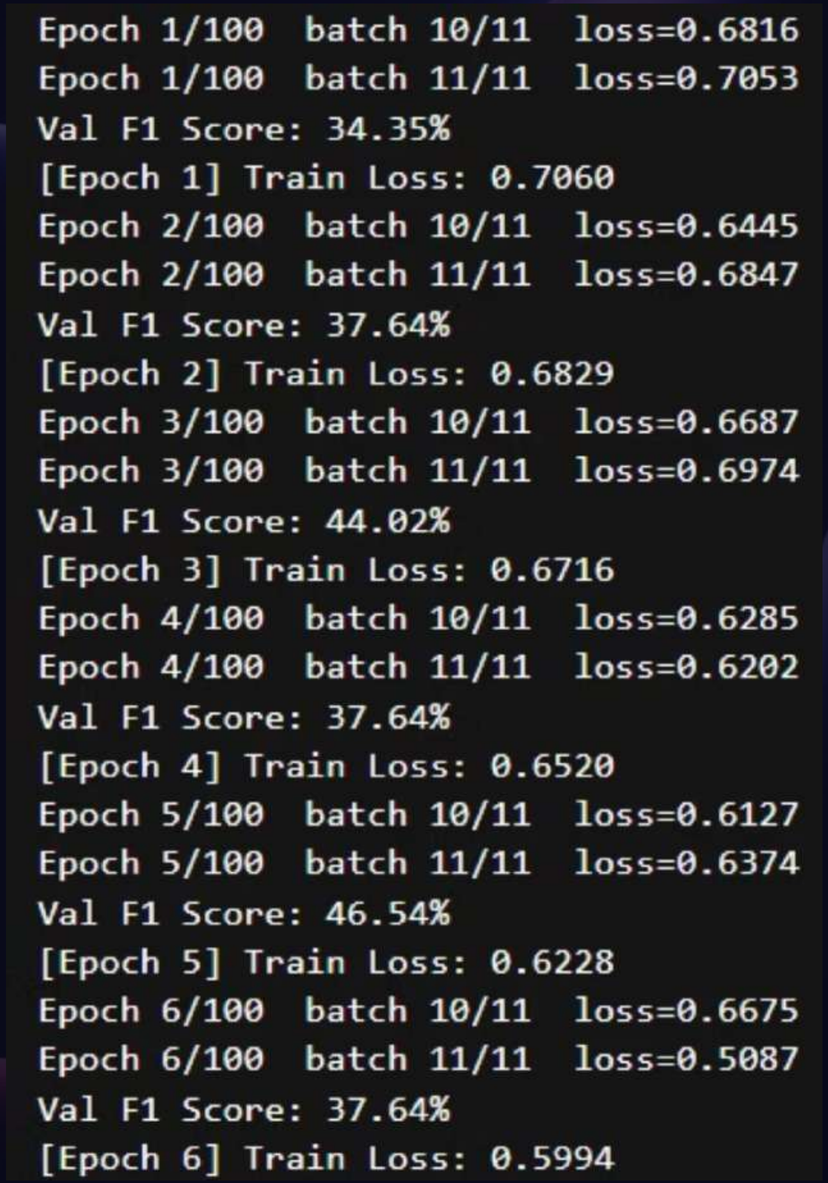

jupyter downstream_integrated.py Last Checkpoint: yesterday
File Edit View Settings Help

```python
def evaluate(model, dataloader, criterion, device, task='age'):
    model.eval()
    total_loss = 0.0
    all_preds, all_labels = [], []
    with torch.no_grad():
        for images, labels in dataloader:
            images, labels = images.to(device), labels.to(device)
            if task == 'age':
                labels = labels.float().unsqueeze(1)
            outputs = model(images)
            loss = criterion(outputs, labels)
            total_loss += loss.item()

            if task == 'gender':
                preds = torch.argmax(outputs, dim=1)
            else:
                preds = outputs.squeeze()

            all_preds.extend(preds.cpu().tolist())
            all_labels.extend(labels.cpu().tolist())

    if task == 'gender':
        score = f1_score(all_labels, all_preds, average='macro') * 100
        print(f"Val F1 Score: {score:.2f}%")
        return total_loss / len(dataloader), score
```

下游任務訓練

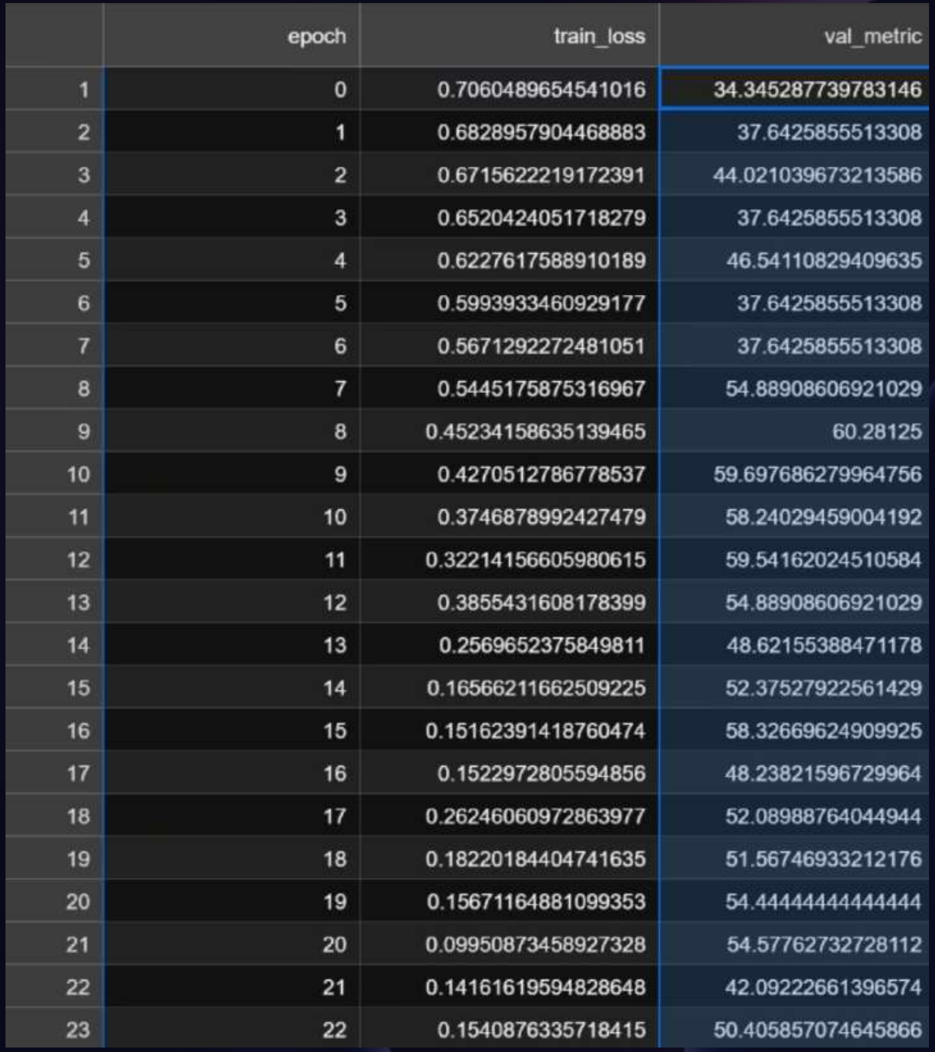


```python
if use_downstream:
    print("開始進行下游任務訓練...", flush=True)
    cmd = [sys.executable, "simclr_schedule_main.py", "--downstream"]
    proc = subprocess.Popen(
        cmd,
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT,
        text=True,
        bufsize=1
    )
    for line in proc.stdout:
        print(line, end="")
    proc.wait()
    print(f"\n✅ 下游任務訓練結束，返回碼: {proc.returncode}")
else:
    print("略過下游訓練階段")
```

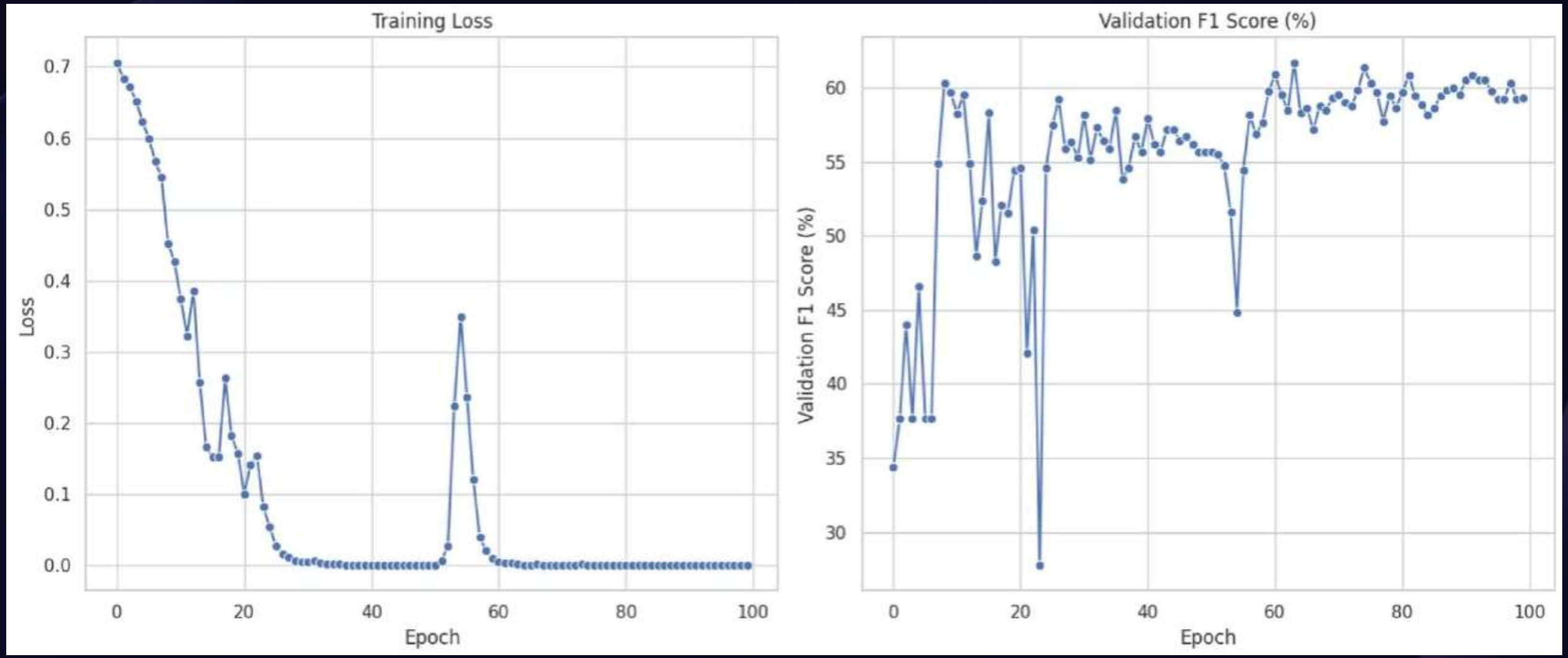


```
Epoch 1/100  batch 10/11  loss=0.6816
Epoch 1/100  batch 11/11  loss=0.7053
Val F1 Score: 34.35%
[Epoch 1] Train Loss: 0.7060
Epoch 2/100  batch 10/11  loss=0.6445
Epoch 2/100  batch 11/11  loss=0.6847
Val F1 Score: 37.64%
[Epoch 2] Train Loss: 0.6829
Epoch 3/100  batch 10/11  loss=0.6687
Epoch 3/100  batch 11/11  loss=0.6974
Val F1 Score: 44.02%
[Epoch 3] Train Loss: 0.6716
Epoch 4/100  batch 10/11  loss=0.6285
Epoch 4/100  batch 11/11  loss=0.6202
Val F1 Score: 37.64%
[Epoch 4] Train Loss: 0.6520
Epoch 5/100  batch 10/11  loss=0.6127
Epoch 5/100  batch 11/11  loss=0.6374
Val F1 Score: 46.54%
[Epoch 5] Train Loss: 0.6228
Epoch 6/100  batch 10/11  loss=0.6675
Epoch 6/100  batch 11/11  loss=0.5087
Val F1 Score: 37.64%
[Epoch 6] Train Loss: 0.5994
```

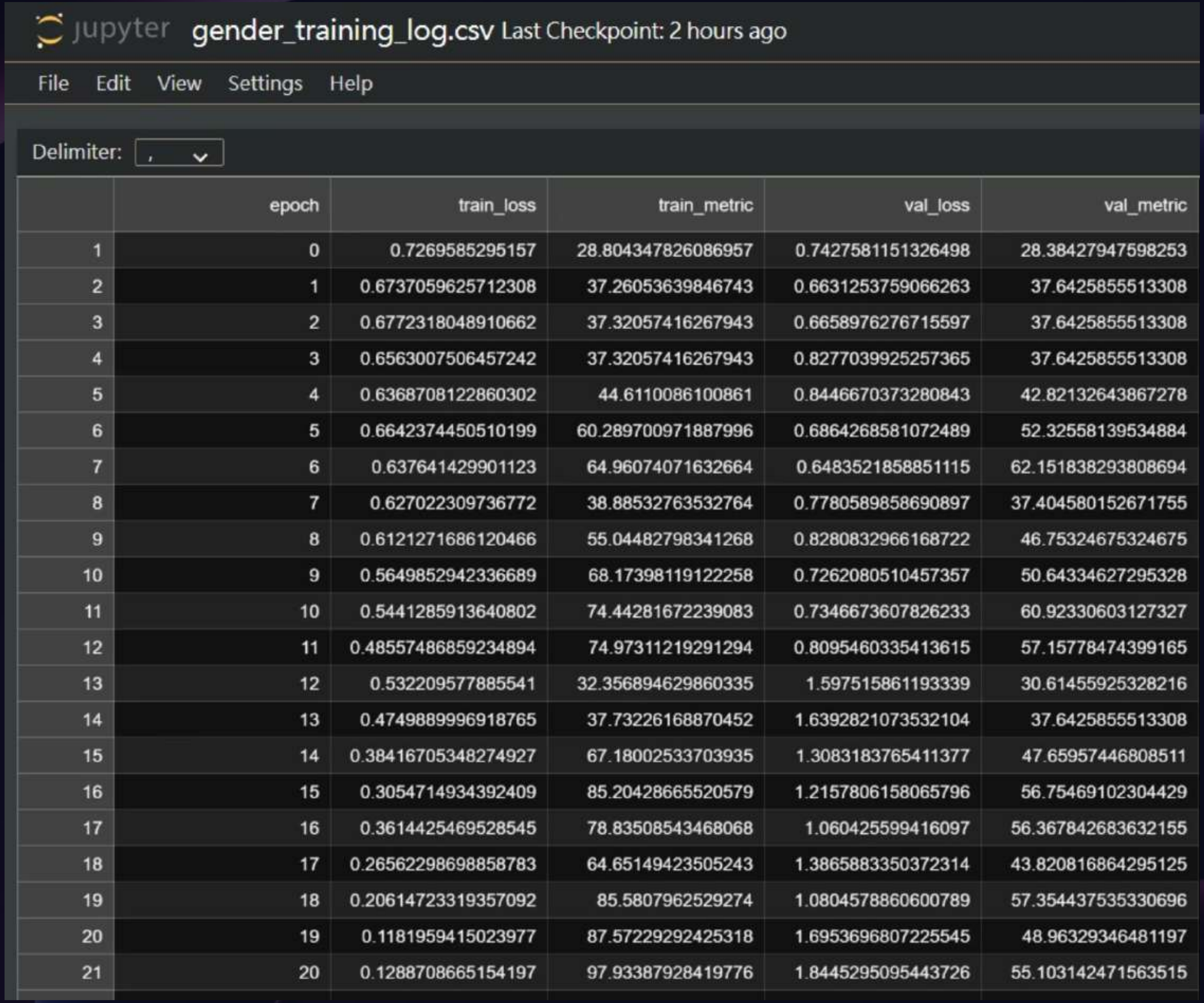
結果分析



	epoch	train_loss	val_metric
1	0	0.7060489654541016	34.345287739783146
2	1	0.6828957904468883	37.6425855513308
3	2	0.6715622219172391	44.021039673213586
4	3	0.6520424051718279	37.6425855513308
5	4	0.6227617588910189	46.54110829409635
6	5	0.5993933460929177	37.6425855513308
7	6	0.5671292272481051	37.6425855513308
8	7	0.5445175875316967	54.88908606921029
9	8	0.45234158635139465	60.28125
10	9	0.4270512786778537	59.697686279964756
11	10	0.3746878992427479	58.24029459004192
12	11	0.32214156605980615	59.54162024510584
13	12	0.3855431608178399	54.88908606921029
14	13	0.2569652375849811	48.62155388471178
15	14	0.16566211662509225	52.37527922561429
16	15	0.15162391418760474	58.32669624909925
17	16	0.1522972805594856	48.23821596729964
18	17	0.26246060972863977	52.08988764044944
19	18	0.18220184404741635	51.56746933212176
20	19	0.15671164881099353	54.44444444444444
21	20	0.09950873458927328	54.57762732728112
22	21	0.14161619594828648	42.09222661396574
23	22	0.1540876335718415	50.405857074645866

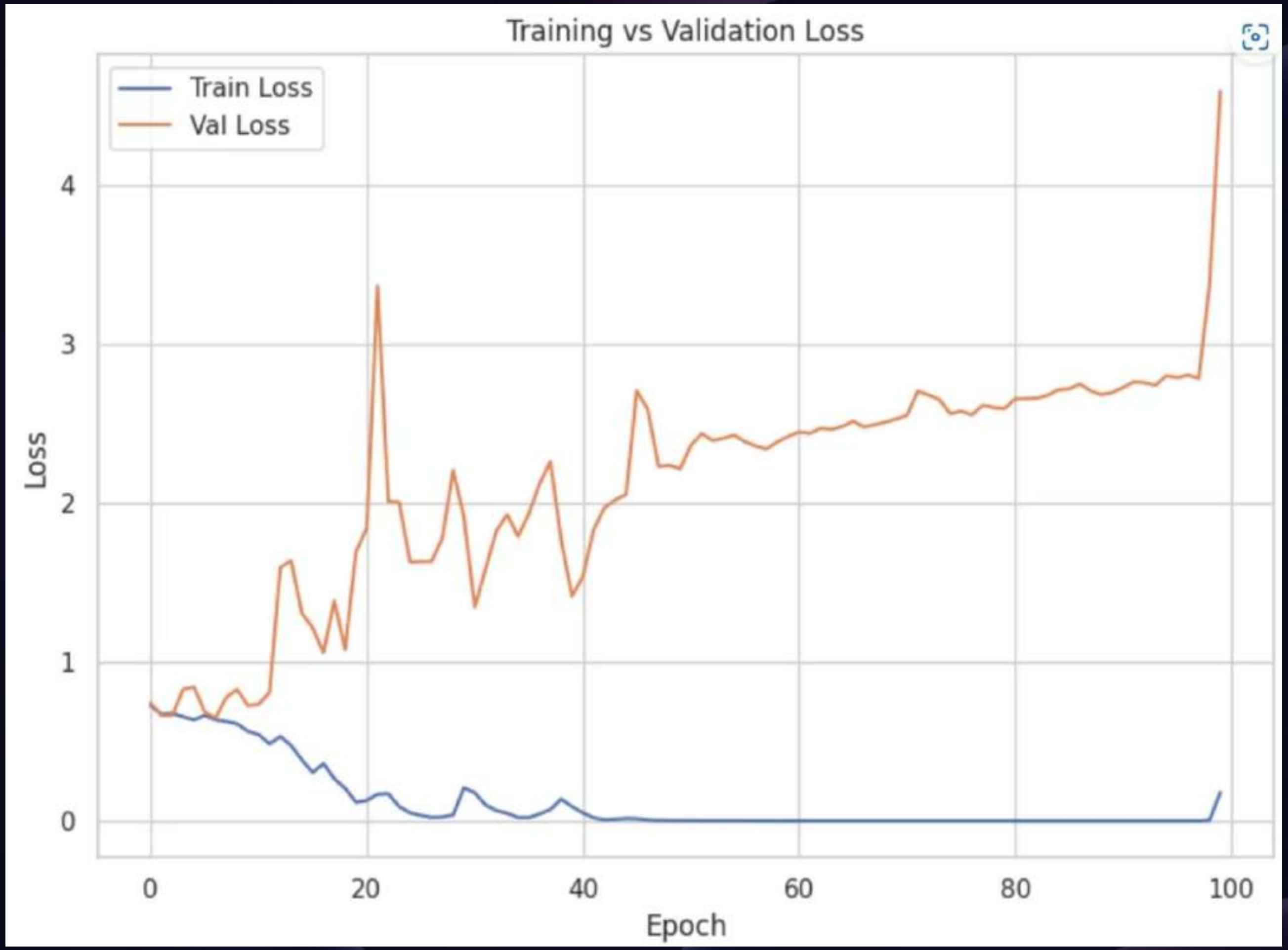


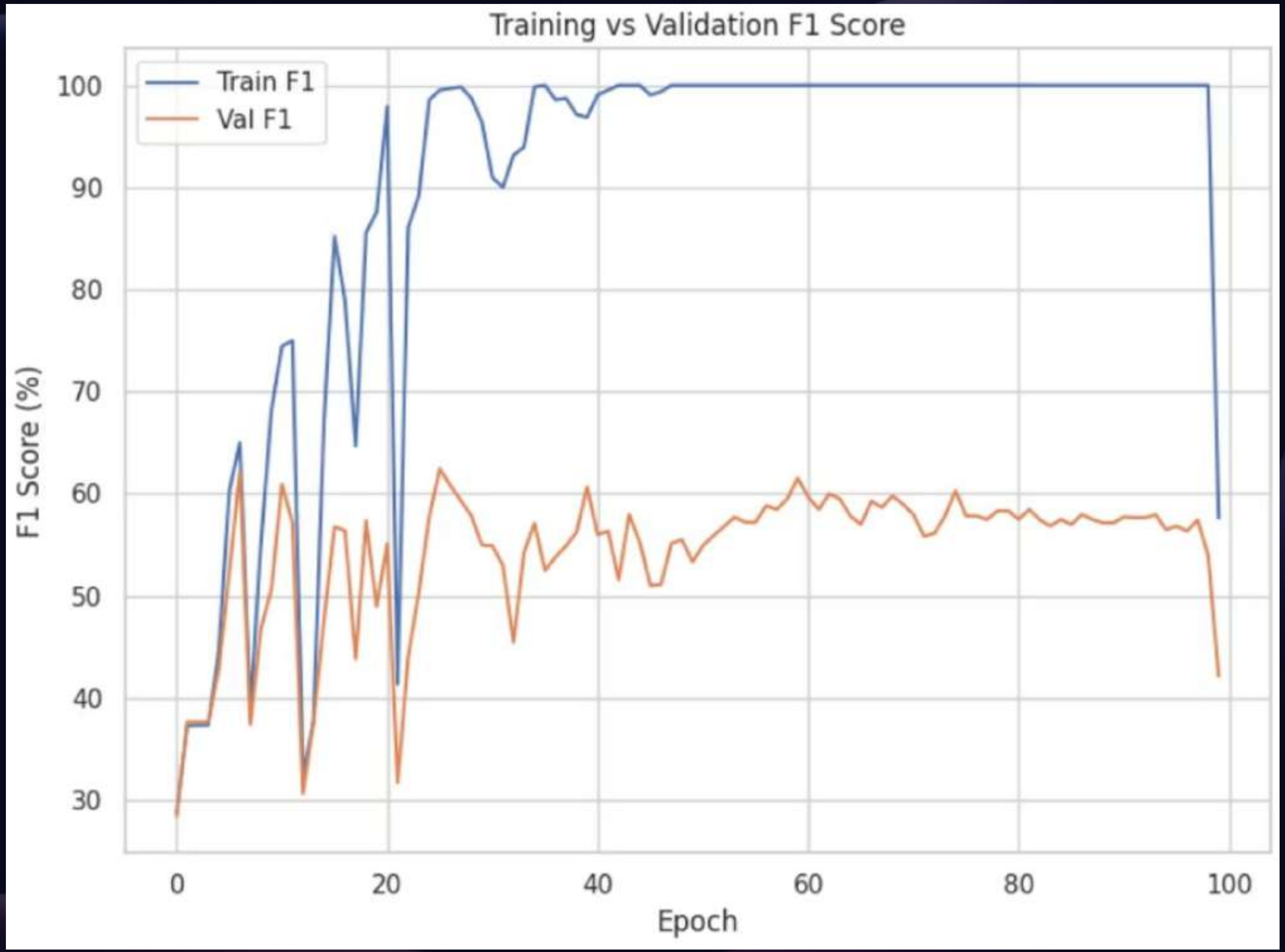
結果分析



	epoch	train_loss	train_metric	val_loss	val_metric
1	0	0.7269585295157	28.804347826086957	0.7427581151326498	28.38427947598253
2	1	0.6737059625712308	37.26053639846743	0.6631253759066263	37.6425855513308
3	2	0.6772318048910662	37.32057416267943	0.6658976276715597	37.6425855513308
4	3	0.6563007506457242	37.32057416267943	0.8277039925257365	37.6425855513308
5	4	0.6368708122860302	44.6110086100861	0.8446670373280843	42.82132643867278
6	5	0.6642374450510199	60.289700971887996	0.6864268581072489	52.32558139534884
7	6	0.637641429901123	64.96074071632664	0.6483521858851115	62.151838293808694
8	7	0.627022309736772	38.88532763532764	0.7780589858690897	37.404580152671755
9	8	0.6121271686120466	55.04482798341268	0.8280832966168722	46.75324675324675
10	9	0.5649852942336689	68.17398119122258	0.7262080510457357	50.64334627295328
11	10	0.5441285913640802	74.44281672239083	0.7346673607826233	60.92330603127327
12	11	0.48557486859234894	74.97311219291294	0.8095460335413615	57.15778474399165
13	12	0.532209577885541	32.356894629860335	1.597515861193339	30.61455925328216
14	13	0.4749889996918765	37.73226168870452	1.6392821073532104	37.6425855513308
15	14	0.38416705348274927	67.18002533703935	1.3083183765411377	47.65957446808511
16	15	0.3054714934392409	85.20428665520579	1.2157806158065796	56.75469102304429
17	16	0.3614425469528545	78.83508543468068	1.060425599416097	56.367842683632155
18	17	0.26562298698858783	64.65149423505243	1.3865883350372314	43.820816864295125
19	18	0.20614723319357092	85.5807962529274	1.0804578860600789	57.354437535330696
20	19	0.1181959415023977	87.57229292425318	1.6953696807225545	48.96329346481197
21	20	0.1288708665154197	97.93387928419776	1.8445295095443726	55.103142471563515

結果分析


Training vs Validation Loss
Train Loss
Val Loss
Loss
Epoch
0
1
2
3
4
0
20
40
60
80
100


Training vs Validation F1 Score
Train F1
Val F1
F1 Score (%)
Epoch
30
40
50
60
70
80
90
100
0
20
40
60
80
100

F1 Score

- 衡量二元(或多元)分類模型在「Precision」與「Recall」之間綜合表現的指標
- Precision → 模型預測為正例(Positive)中，實際正例的比例

$$Precision = TP/(TP + FP)$$

- Recall → 在所有實際為正例的樣本中，被模型正確抓出的比例

$$Recall = TP/(TP + FN)$$

TP → True Positives

FP → False Positives

FN → False Negatives

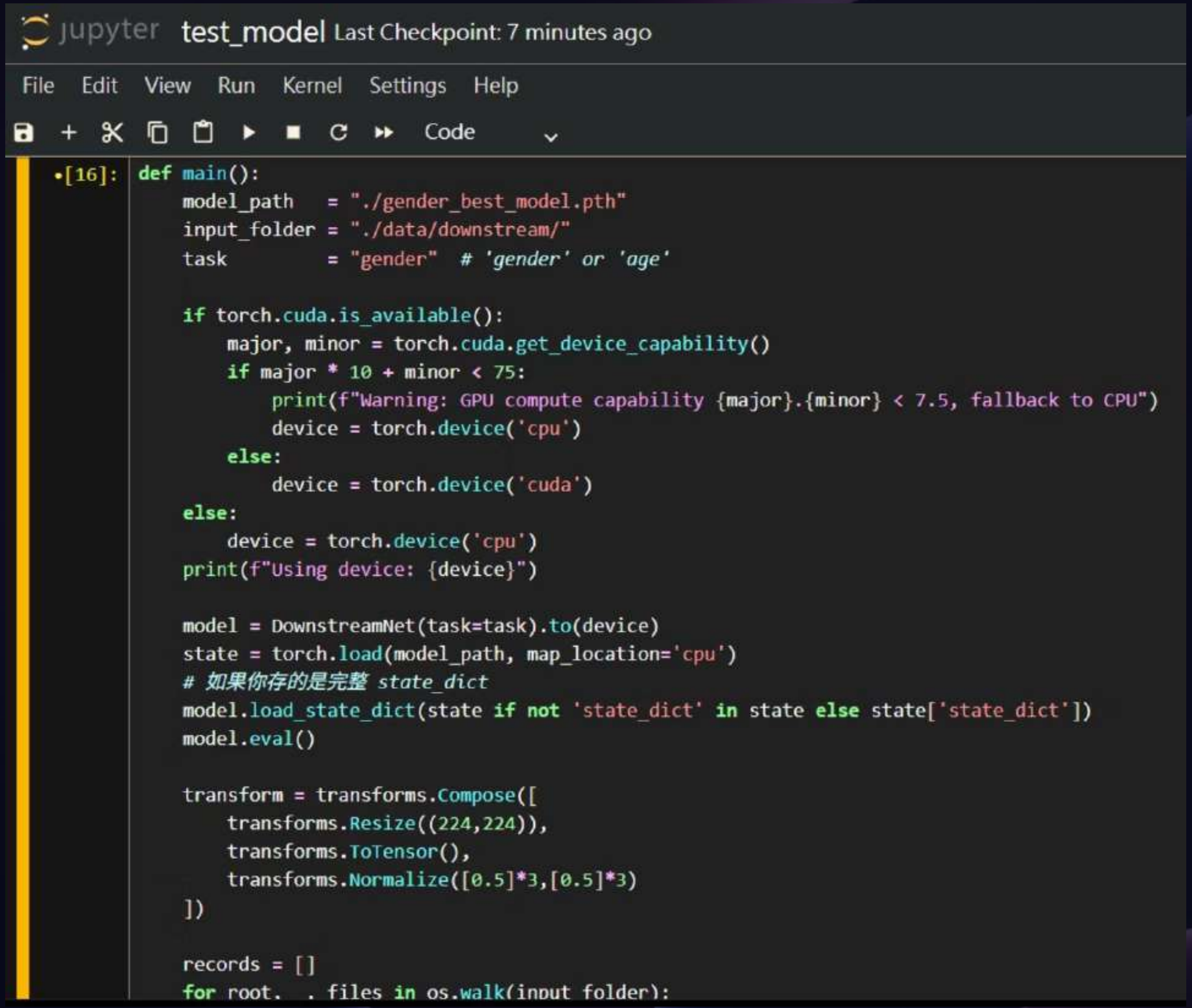
F1 Score

- F1 Score 定義 :

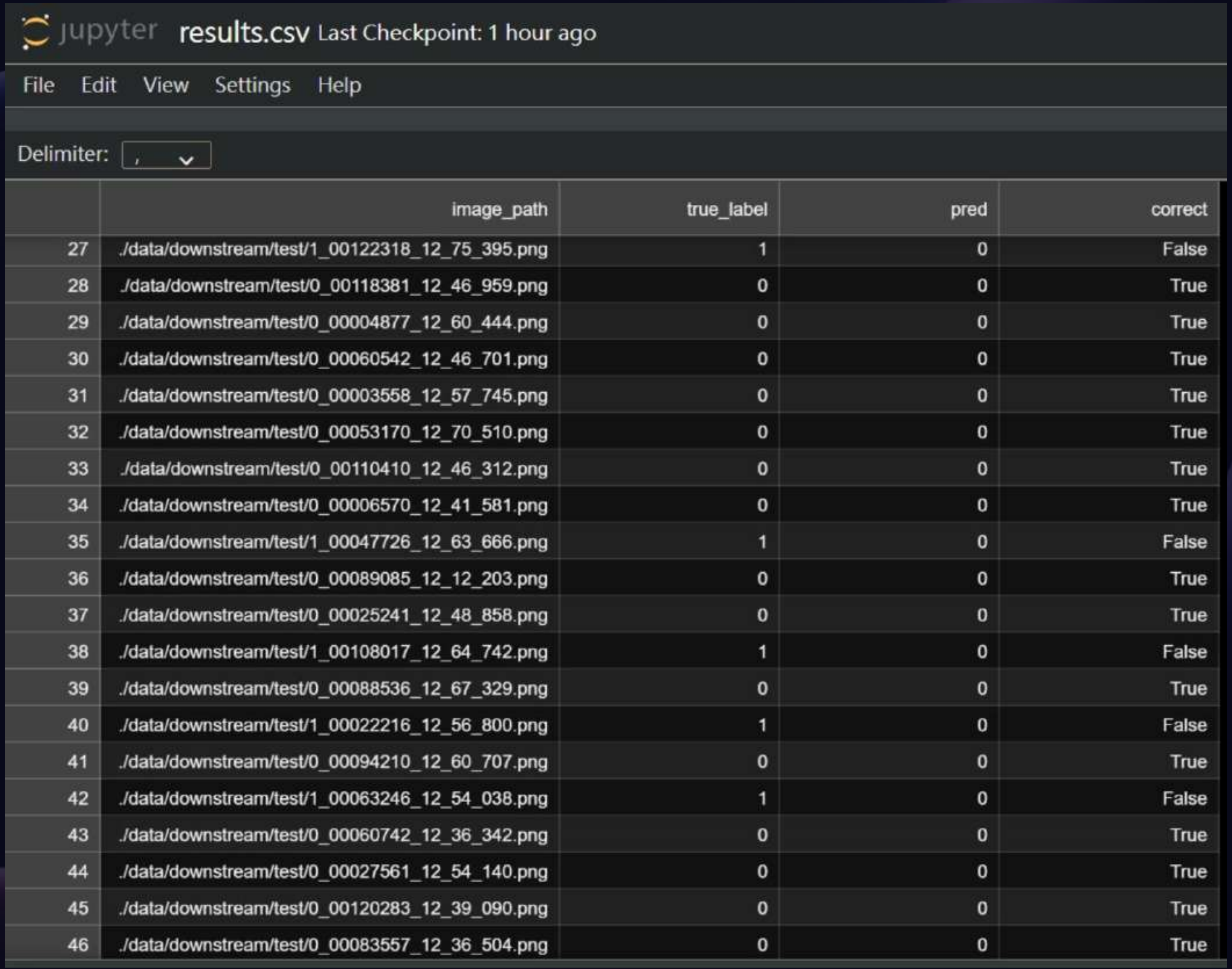
$$F1 = 2 \times (Precision \times Recall)/(Precision + Recall)$$

- 使用場景 : 男女性別分辨、確診陽性陰性
- 模型評分 : 取值範圍 0 到 1，分數越高模型越優

結果分析


jupyter test_model Last Checkpoint: 7 minutes ago
File Edit View Run Kernel Settings Help
Code

```
•[16]: def main():
           model_path   = "./gender_best_model.pth"
           input_folder = "./data/downstream/"
           task         = "gender"  # 'gender' or 'age'

           if torch.cuda.is_available():
               major, minor = torch.cuda.get_device_capability()
               if major * 10 + minor < 75:
                   print(f"Warning: GPU compute capability {major}.{minor} < 7.5, fallback to CPU")
                   device = torch.device('cpu')
               else:
                   device = torch.device('cuda')
           else:
               device = torch.device('cpu')
           print(f"Using device: {device}")

           model = DownstreamNet(task=task).to(device)
           state = torch.load(model_path, map_location='cpu')
           # 如果你存的是完整 state_dict
           model.load_state_dict(state if not 'state_dict' in state else state['state_dict'])
           model.eval()

           transform = transforms.Compose([
               transforms.Resize((224,224)),
               transforms.ToTensor(),
               transforms.Normalize([0.5]*3,[0.5]*3)
           ])

           records = []
           for root, _, files in os.walk(input_folder):
```


jupyter results.csv Last Checkpoint: 1 hour ago
File Edit View Settings Help
Delimiter: ,

	image_path	true_label	pred	correct
27	./data/downstream/test/1_00122318_12_75_395.png	1	0	False
28	./data/downstream/test/0_00118381_12_46_959.png	0	0	True
29	./data/downstream/test/0_00004877_12_60_444.png	0	0	True
30	./data/downstream/test/0_00060542_12_46_701.png	0	0	True
31	./data/downstream/test/0_00003558_12_57_745.png	0	0	True
32	./data/downstream/test/0_00053170_12_70_510.png	0	0	True
33	./data/downstream/test/0_00110410_12_46_312.png	0	0	True
34	./data/downstream/test/0_00006570_12_41_581.png	0	0	True
35	./data/downstream/test/1_00047726_12_63_666.png	1	0	False
36	./data/downstream/test/0_00089085_12_12_203.png	0	0	True
37	./data/downstream/test/0_00025241_12_48_858.png	0	0	True
38	./data/downstream/test/1_00108017_12_64_742.png	1	0	False
39	./data/downstream/test/0_00088536_12_67_329.png	0	0	True
40	./data/downstream/test/1_00022216_12_56_800.png	1	0	False
41	./data/downstream/test/0_00094210_12_60_707.png	0	0	True
42	./data/downstream/test/1_00063246_12_54_038.png	1	0	False
43	./data/downstream/test/0_00060742_12_36_342.png	0	0	True
44	./data/downstream/test/0_00027561_12_54_140.png	0	0	True
45	./data/downstream/test/0_00120283_12_39_090.png	0	0	True
46	./data/downstream/test/0_00083557_12_36_504.png	0	0	True