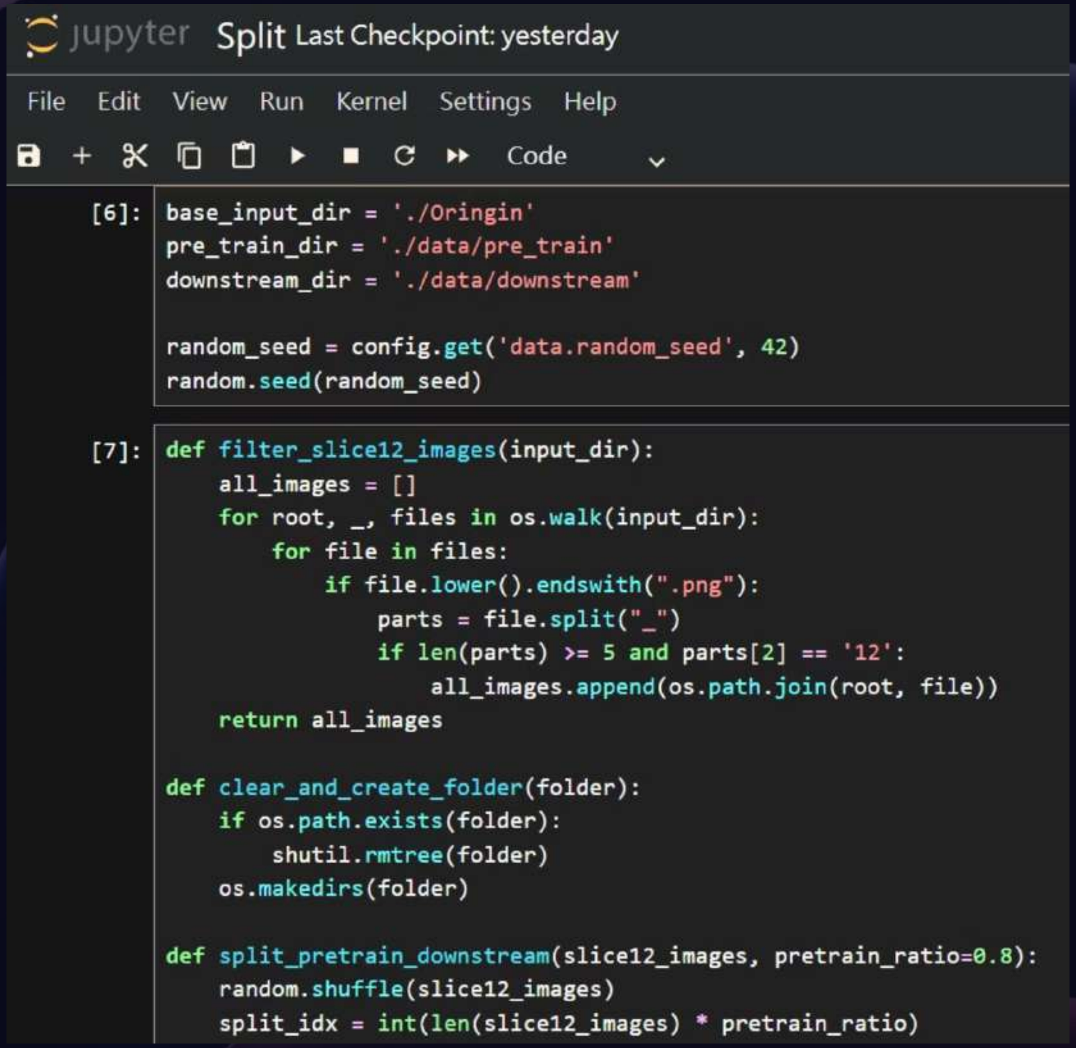
SimCLR 年齡預測

- 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
- Mean Absolute Error

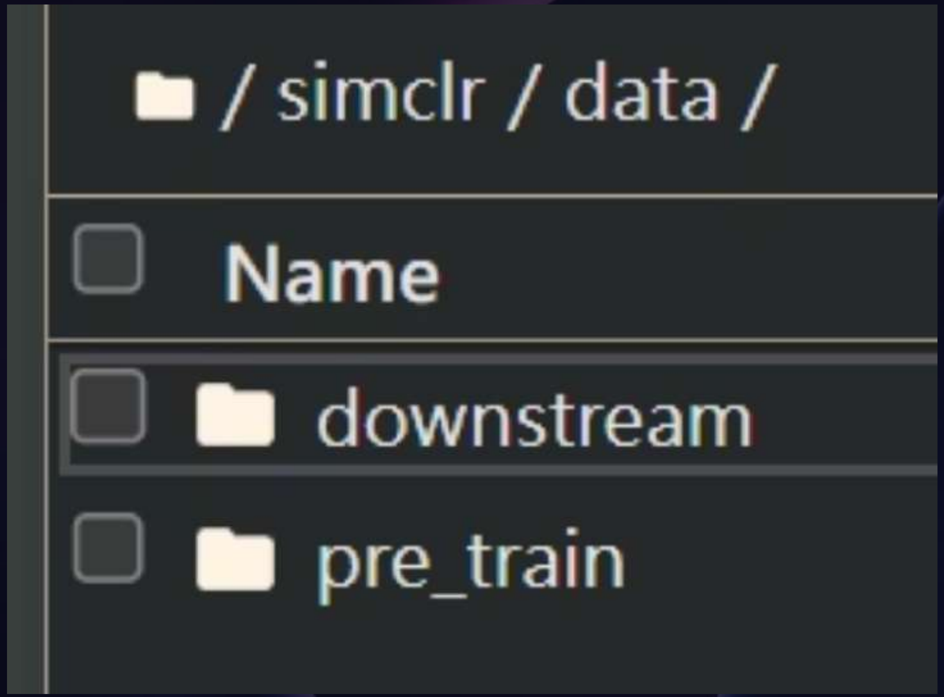
資料準備



```
[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

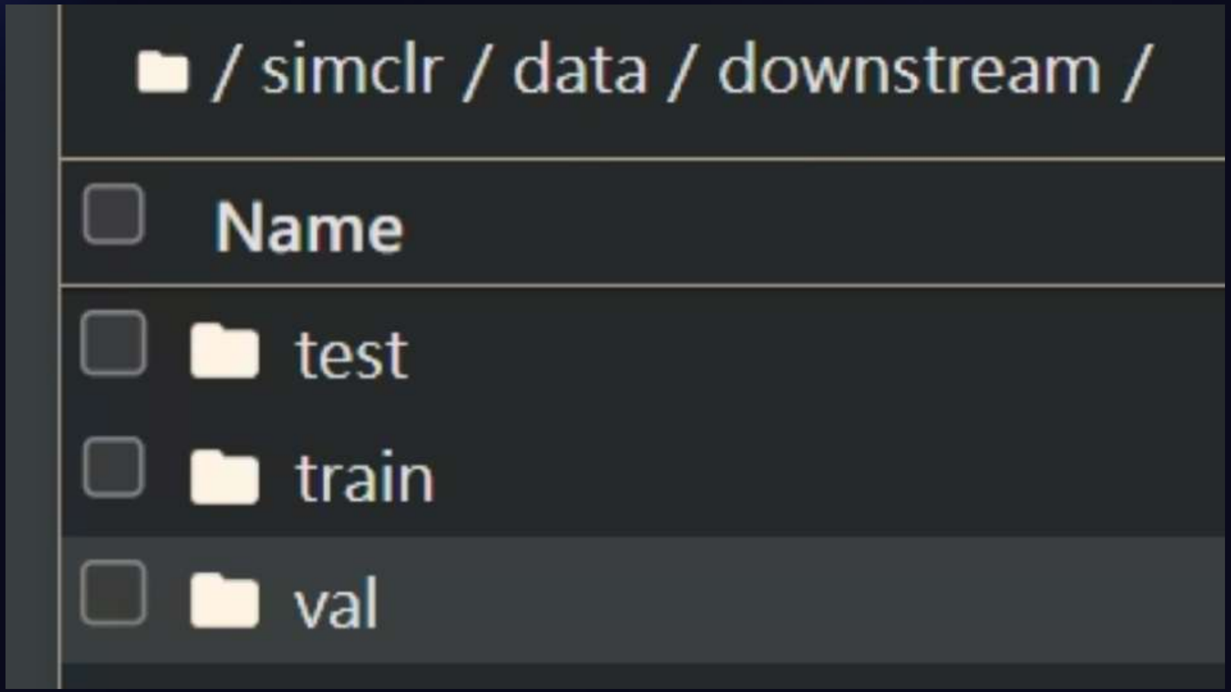


/ simclr / data /

□ Name

□ downstream

□ pre_train



/ simclr / data / downstream /

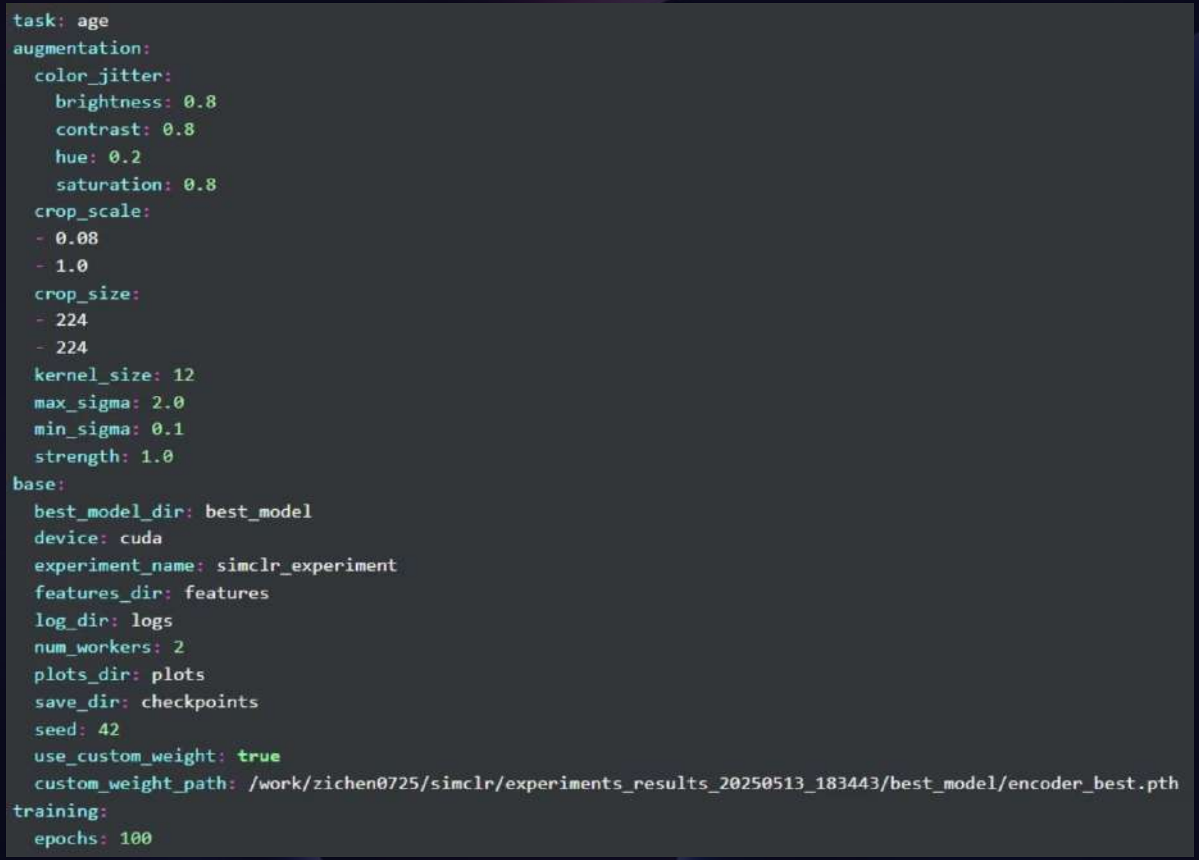
□ Name

□ test

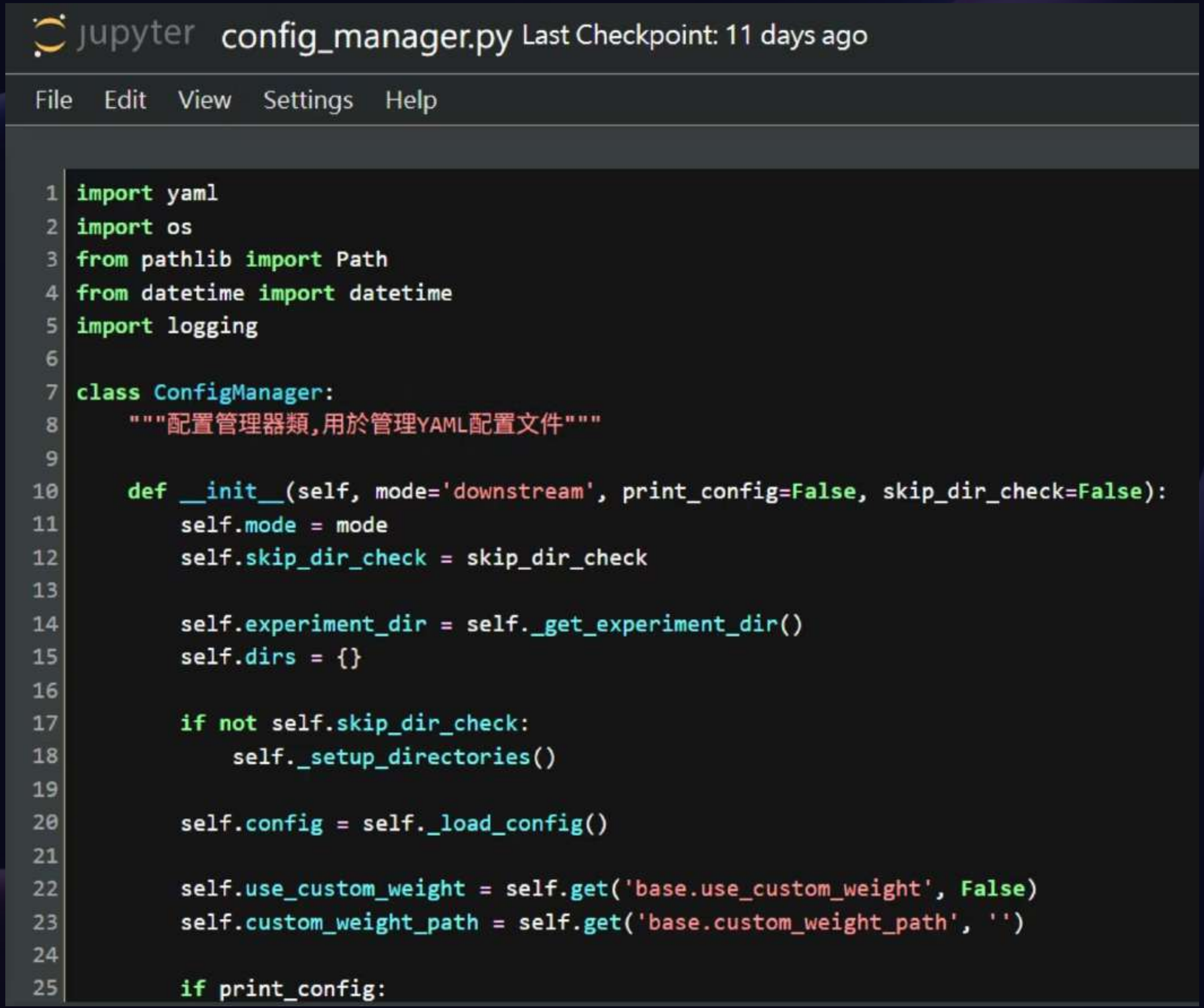
□ train

□ val

配置管理

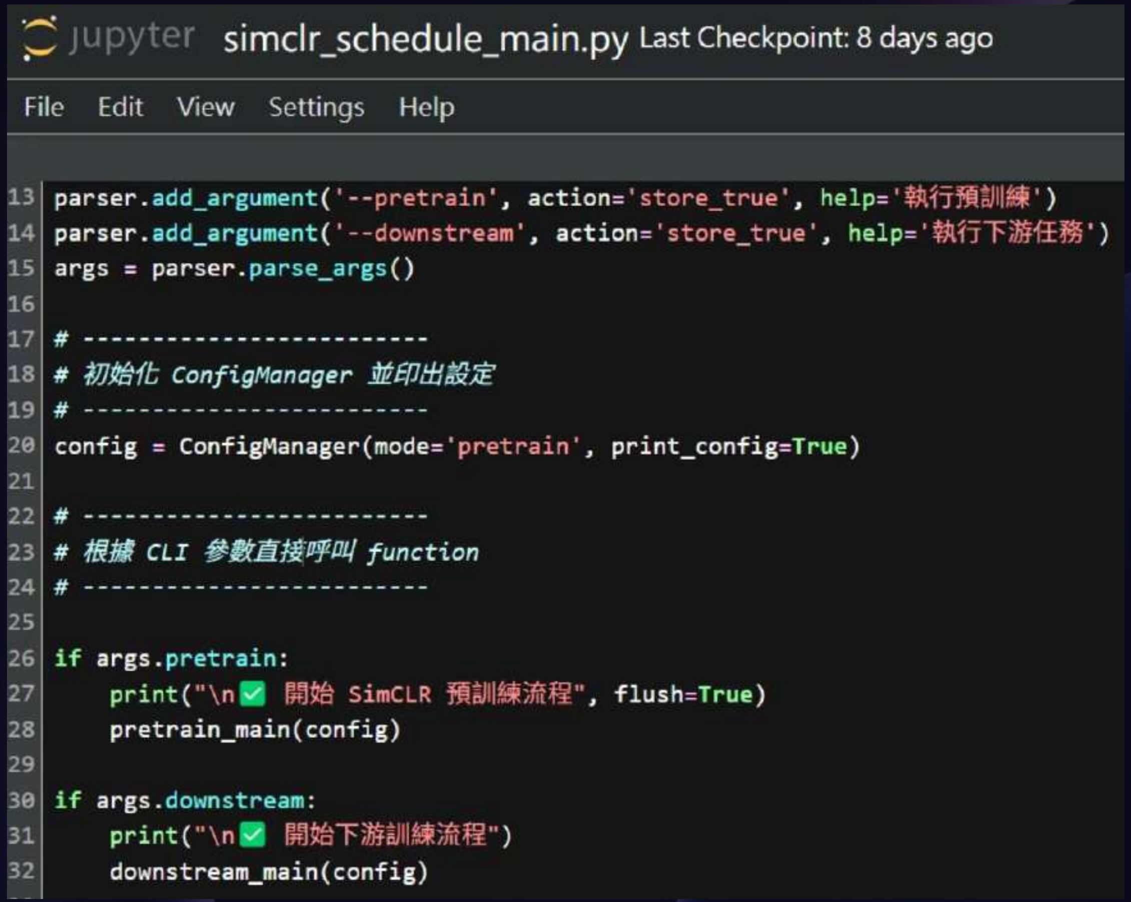


```yaml
task: age
augmentation:
  color_jitter:
    brightness: 0.8
    contrast: 0.8
    hue: 0.2
    saturation: 0.8
  crop_scale:
  - 0.08
  - 1.0
  crop_size:
  - 224
  - 224
  kernel_size: 12
  max_sigma: 2.0
  min_sigma: 0.1
  strength: 1.0
base:
  best_model_dir: best_model
  device: cuda
  experiment_name: simclr_experiment
  features_dir: features
  log_dir: logs
  num_workers: 2
  plots_dir: plots
  save_dir: checkpoints
  seed: 42
  use_custom_weight: true
  custom_weight_path: /work/zichen0725/simclr/experiments_results_20250513_183443/best_model/encoder_best.pth
training:
  epochs: 100
```

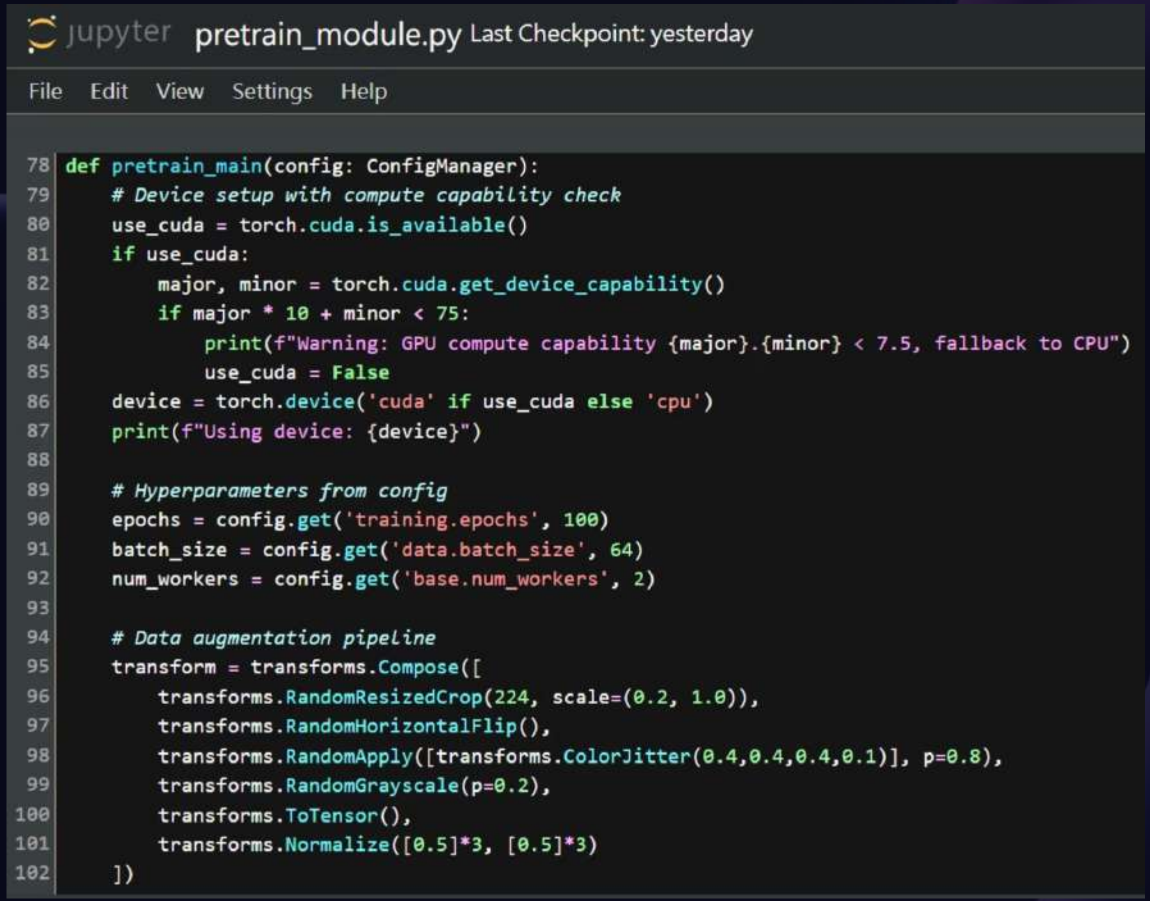


```python
import yaml
import os
from pathlib import Path
from datetime import datetime
import logging

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:
```

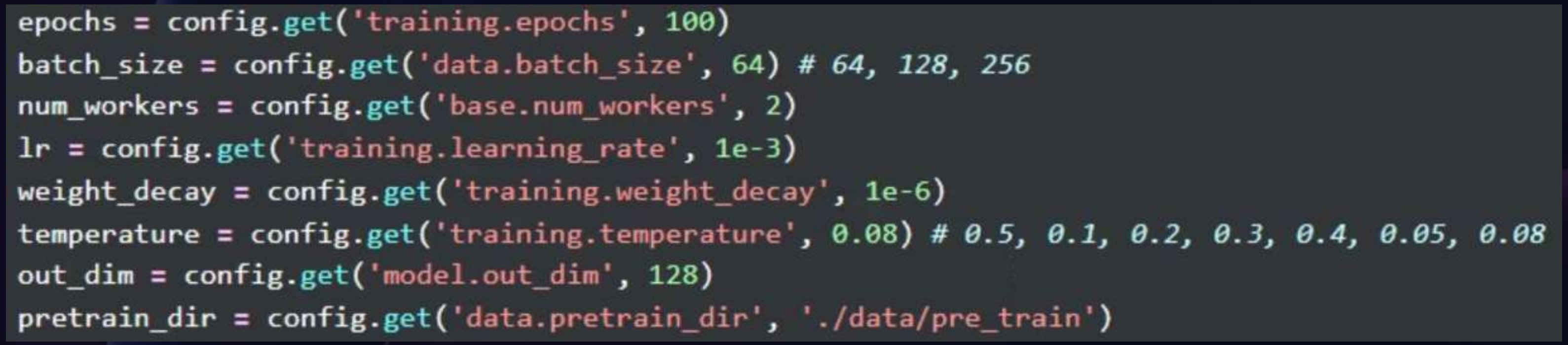
SimCLR 自監督預訓練



```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)
```

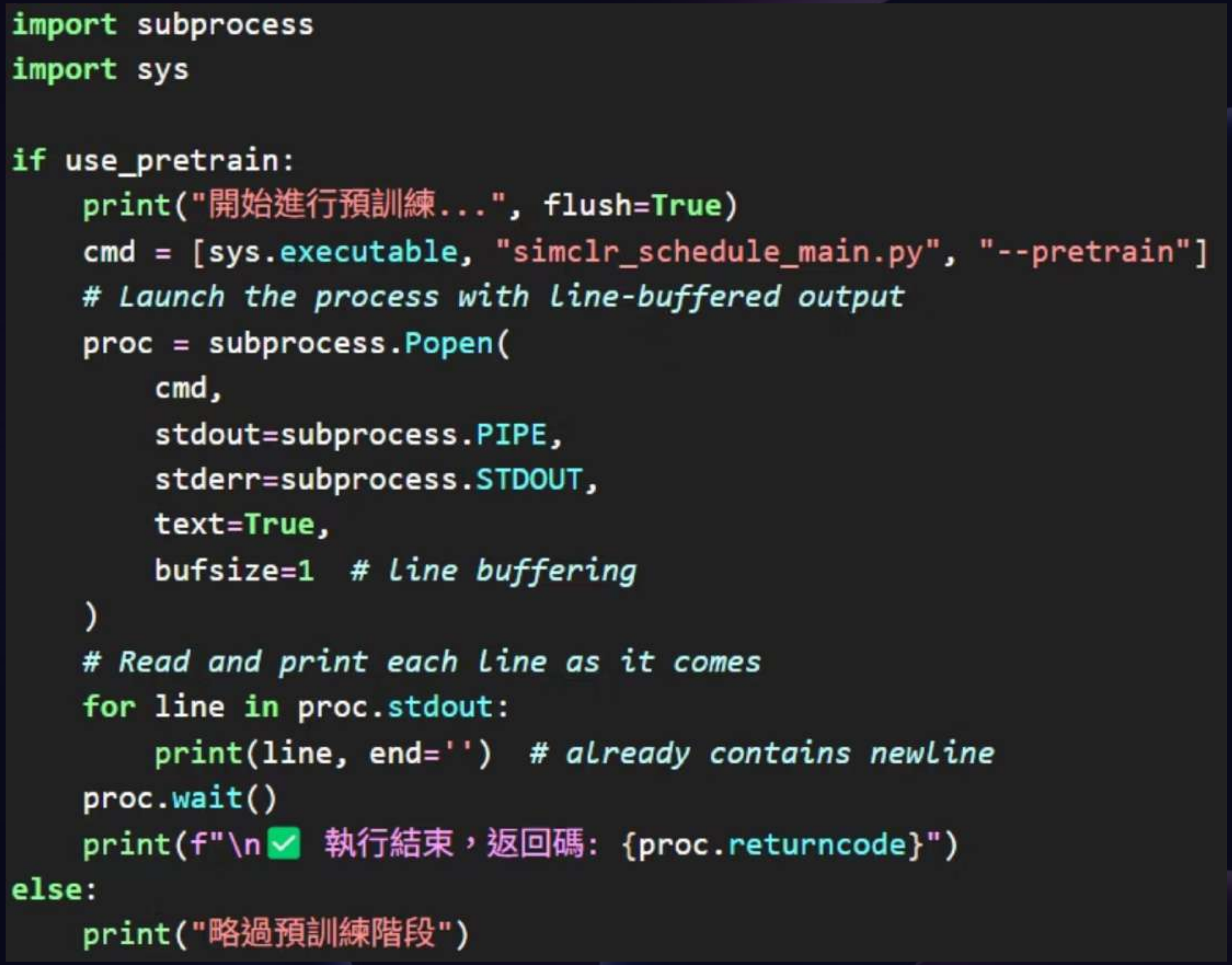


```python
def pretrain_main(config: ConfigManager):
    # Device setup with compute capability check
    use_cuda = torch.cuda.is_available()
    if use_cuda:
        major, minor = torch.cuda.get_device_capability()
        if major * 10 + minor < 75:
            print(f"Warning: GPU compute capability {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)
    ])
```

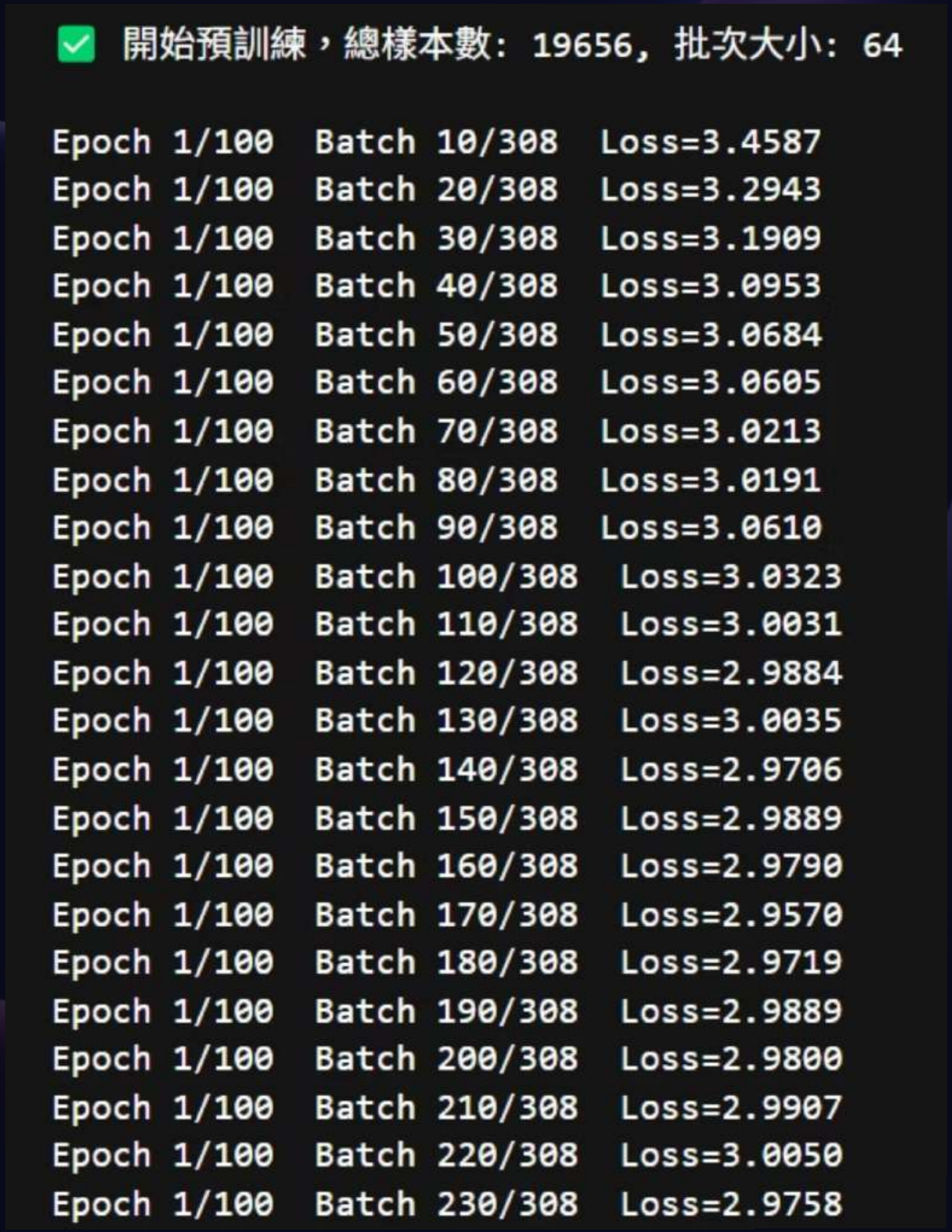


```python
epochs = config.get('training.epochs', 100)
batch_size = config.get('data.batch_size', 64) # 64, 128, 256
num_workers = config.get('base.num_workers', 2)
lr = config.get('training.learning_rate', 1e-3)
weight_decay = config.get('training.weight_decay', 1e-6)
temperature = config.get('training.temperature', 0.08) # 0.5, 0.1, 0.2, 0.3, 0.4, 0.05, 0.08
out_dim = config.get('model.out_dim', 128)
pretrain_dir = config.get('data.pretrain_dir', './data/pre_train')
```

SimCLR 自監督預訓練

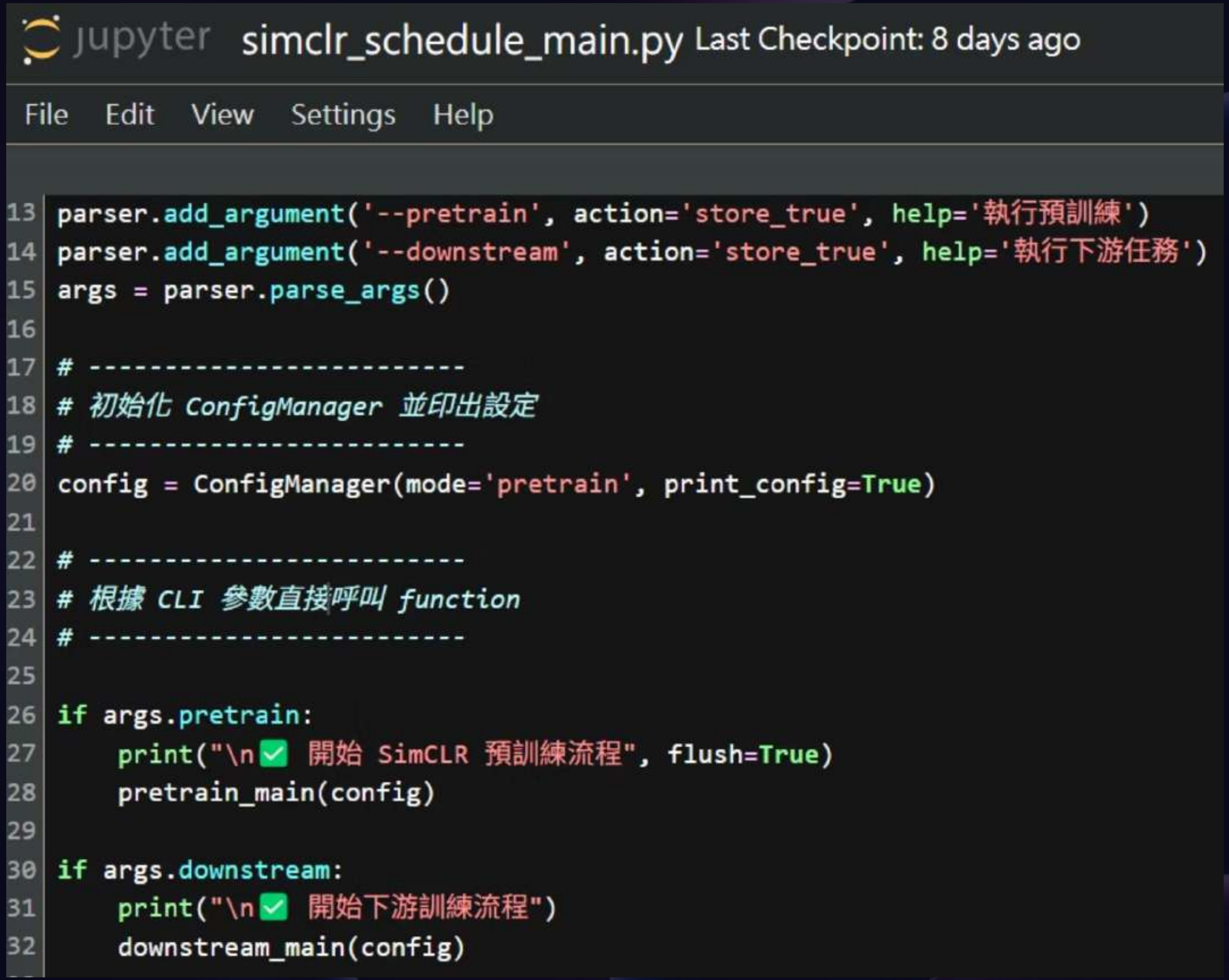


```python
import subprocess
import sys

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")
```

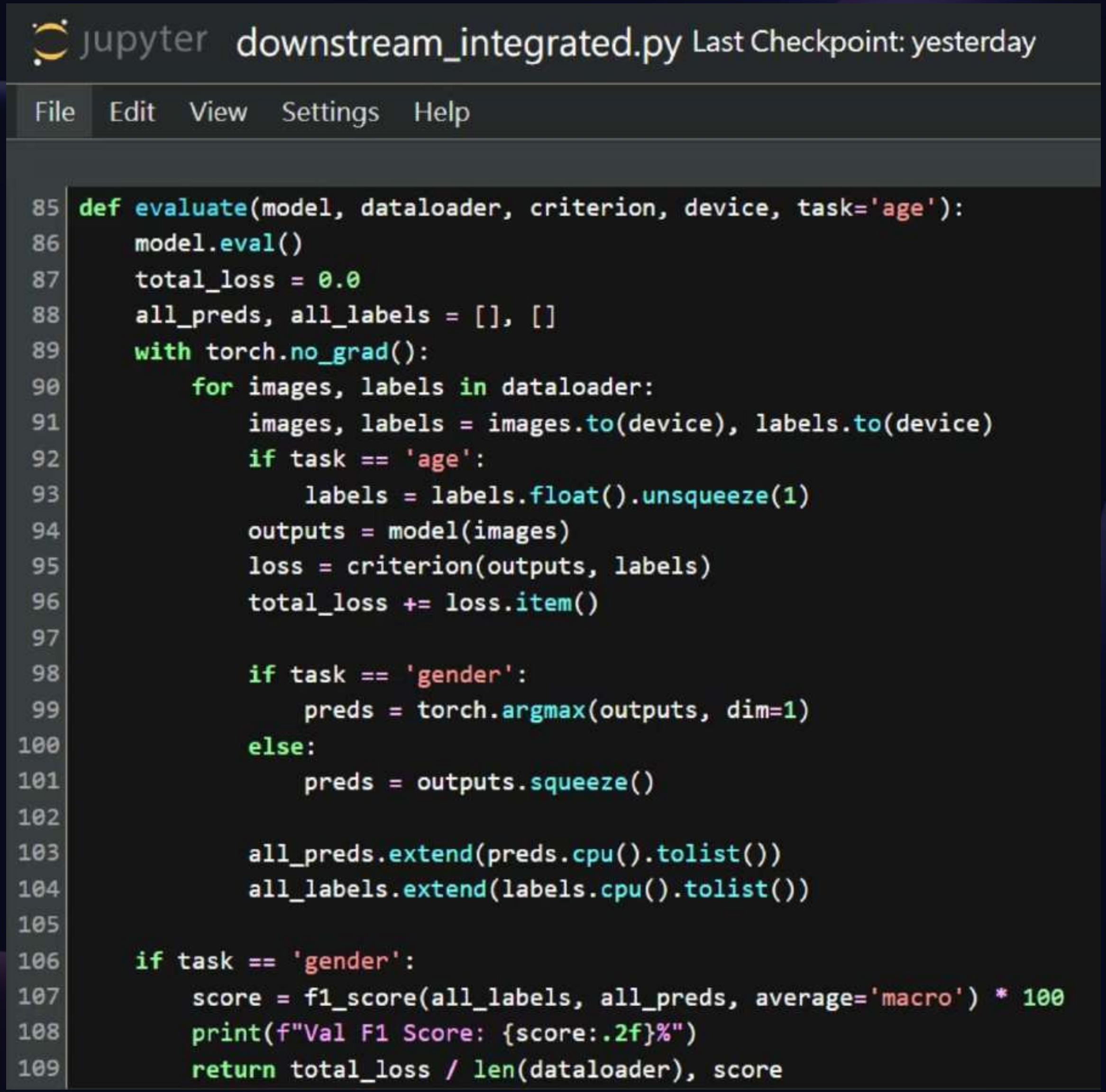


```
✅ 開始預訓練，總樣本數: 19656, 批次大小: 64

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=2.9758
```

下游任務訓練

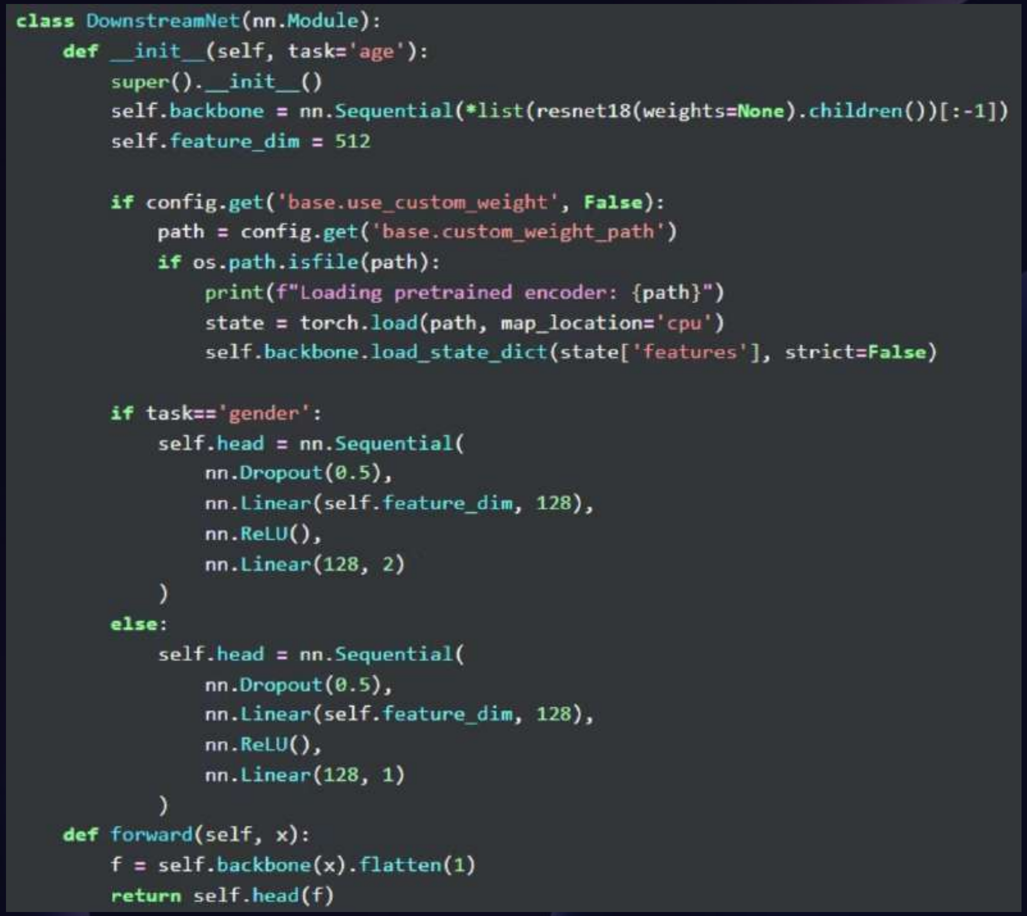


```python
parser.add_argument('--pretrain', action='store_true', help='執行預訓練')
parser.add_argument('--downstream', action='store_true', help='執行下游任務')
args = parser.parse_args()

# ------------------------
# 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)
```

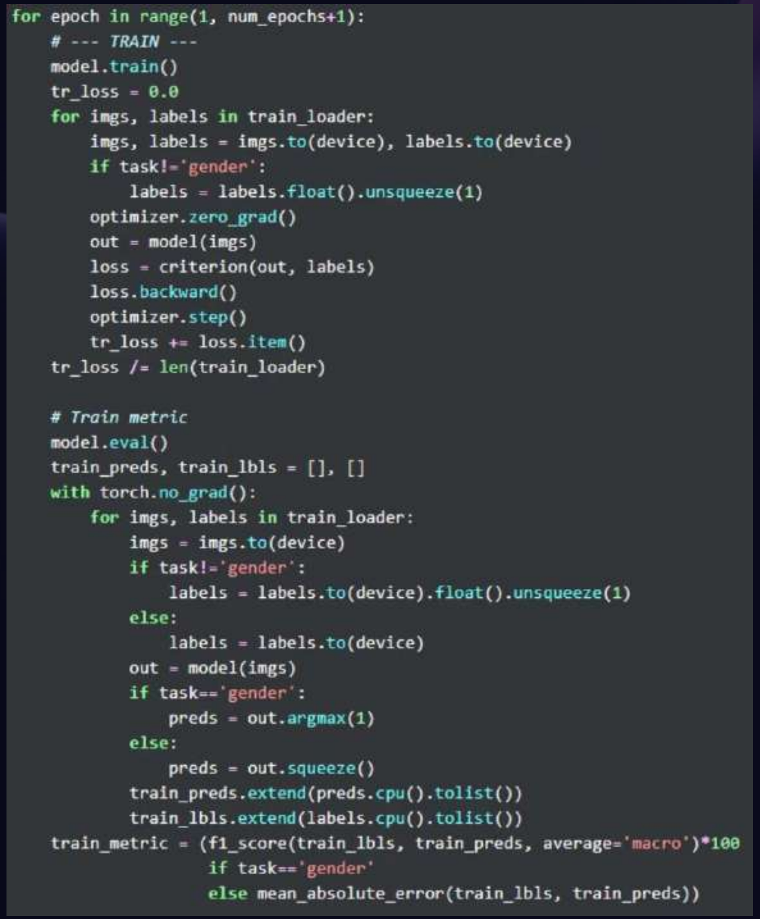


```python
def evaluate(model, dataloader, criterion, device, task='age'):
    model.eval()
    total_loss = 0.0
    all_preds, all_labels = [], []
    with torch.no_grad():
        for images, labels in dataloader:
            images, labels = images.to(device), labels.to(device)
            if task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

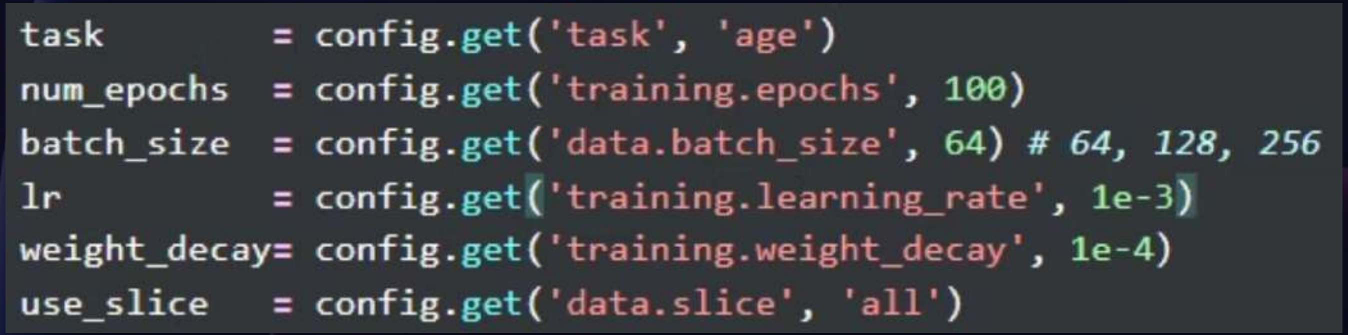
下游任務訓練



```
class DownstreamNet(nn.Module):
    def __init__(self, task='age'):
        super().__init__()
        self.backbone = nn.Sequential(*list(resnet18(weights=None).children())[:-1])
        self.feature_dim = 512

        if config.get('base.use_custom_weight', False):
            path = config.get('base.custom_weight_path')
            if os.path.isfile(path):
                print(f"Loading pretrained encoder: {path}")
                state = torch.load(path, map_location='cpu')
                self.backbone.load_state_dict(state['features'], strict=False)

        if task=='gender':
            self.head = nn.Sequential(
                nn.Dropout(0.5),
                nn.Linear(self.feature_dim, 128),
                nn.ReLU(),
                nn.Linear(128, 2)
            )
        else:
            self.head = nn.Sequential(
                nn.Dropout(0.5),
                nn.Linear(self.feature_dim, 128),
                nn.ReLU(),
                nn.Linear(128, 1)
            )
    def forward(self, x):
        f = self.backbone(x).flatten(1)
        return self.head(f)
```

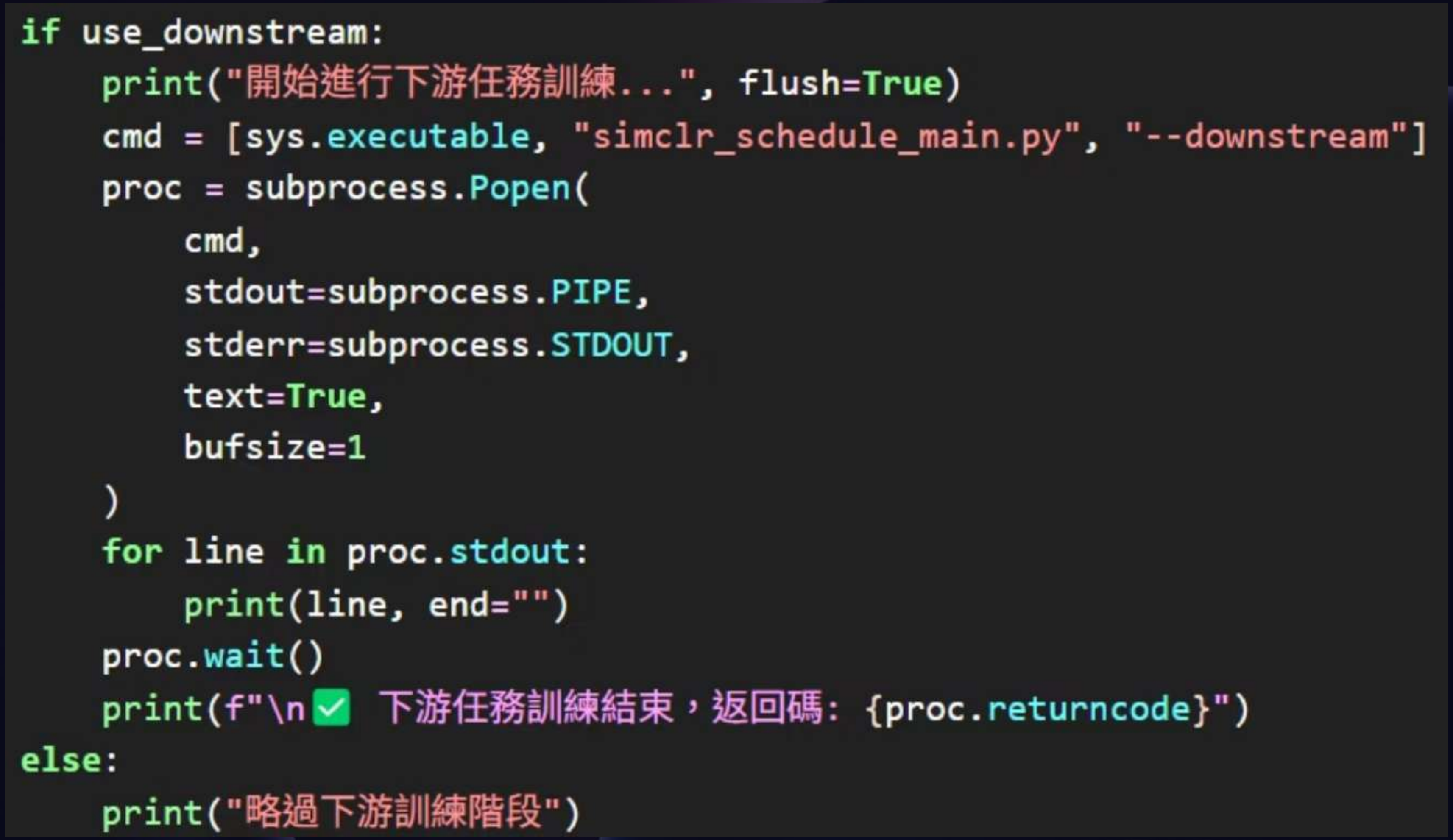


```
for epoch in range(1, num_epochs+1):
    # --- TRAIN ---
    model.train()
    tr_loss = 0.0
    for imgs, labels in train_loader:
        imgs, labels = imgs.to(device), labels.to(device)
        if task!='gender':
            labels = labels.float().unsqueeze(1)
        optimizer.zero_grad()
        out = model(imgs)
        loss = criterion(out, labels)
        loss.backward()
        optimizer.step()
        tr_loss += loss.item()
    tr_loss /= len(train_loader)

    # Train metric
    model.eval()
    train_preds, train_lbls = [], []
    with torch.no_grad():
        for imgs, labels in train_loader:
            imgs = imgs.to(device)
            if task!='gender':
                labels = labels.to(device).float().unsqueeze(1)
            else:
                labels = labels.to(device)
            out = model(imgs)
            if task=='gender':
                preds = out.argmax(1)
            else:
                preds = out.squeeze()
            train_preds.extend(preds.cpu().tolist())
            train_lbls.extend(labels.cpu().tolist())
    train_metric = (f1_score(train_lbls, train_preds, average='macro')*100
                    if task=='gender'
                    else mean_absolute_error(train_lbls, train_preds))
```

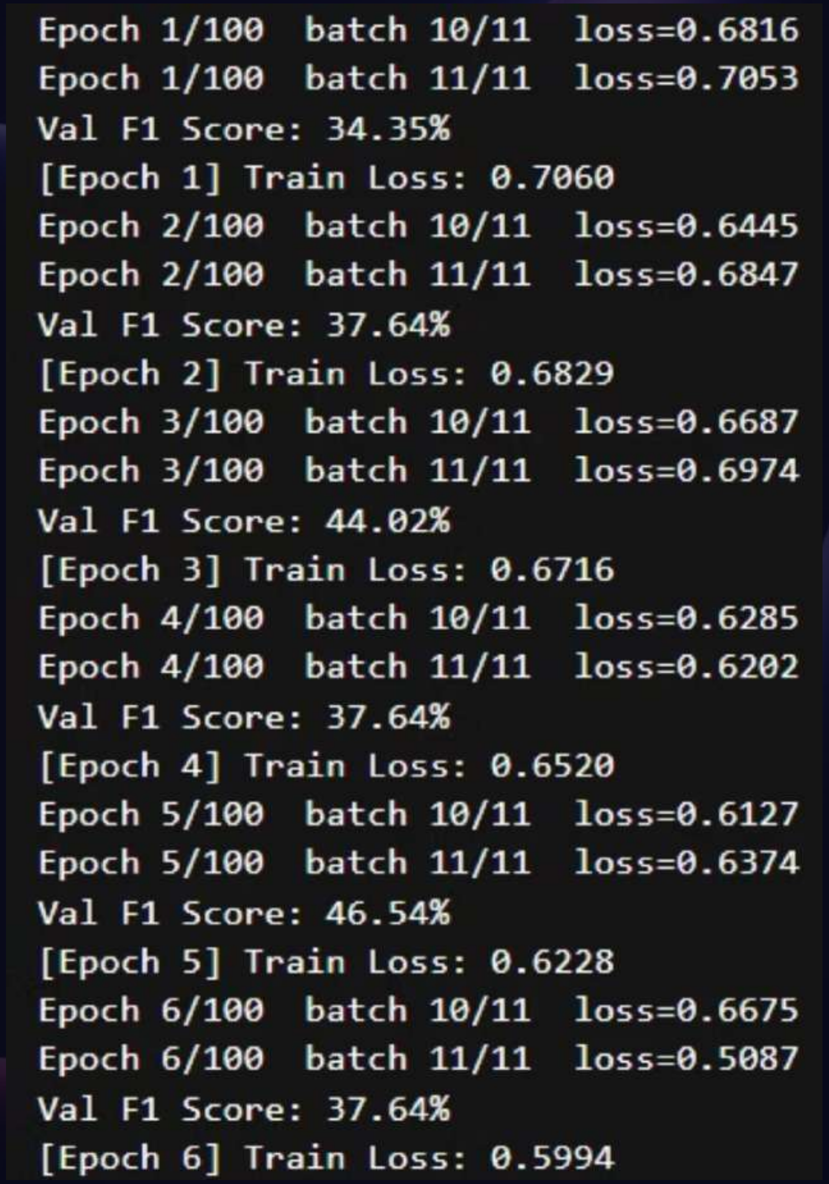


```
task         = config.get('task', 'age')
num_epochs   = config.get('training.epochs', 100)
batch_size   = config.get('data.batch_size', 64) # 64, 128, 256
lr           = config.get('training.learning_rate', 1e-3)
weight_decay= config.get('training.weight_decay', 1e-4)
use_slice    = config.get('data.slice', 'all')
```

下游任務訓練

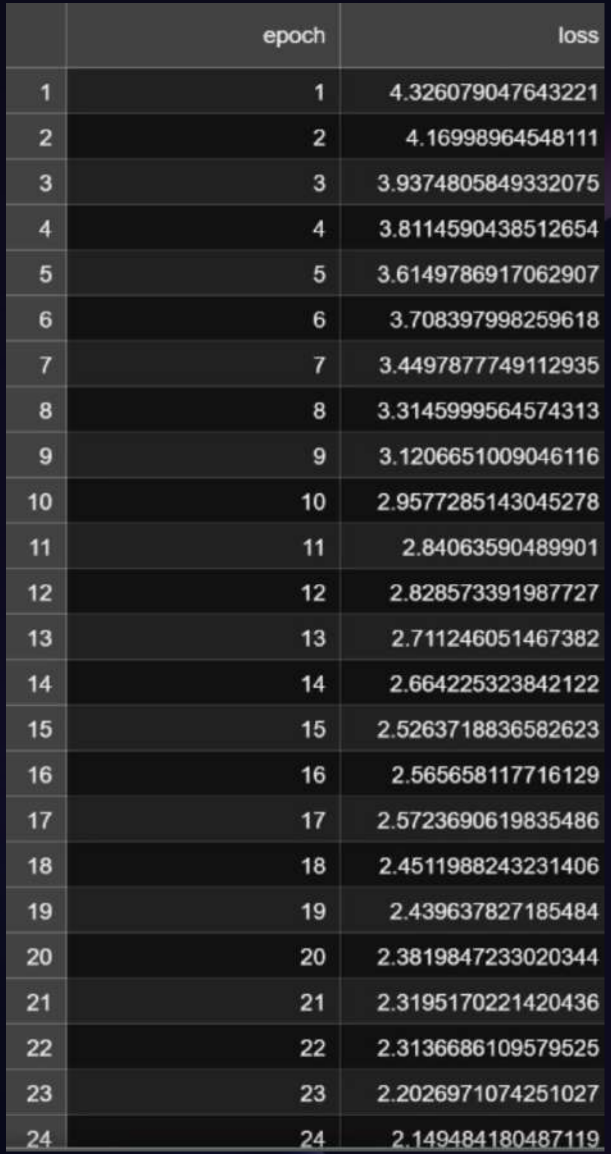


```
if use_downstream:
    print("開始進行下游任務訓練...", flush=True)
    cmd = [sys.executable, "simclr_schedule_main.py", "--downstream"]
    proc = subprocess.Popen(
        cmd,
        stdout=subprocess.PIPE,
        stderr=subprocess.STDOUT,
        text=True,
        bufsize=1
    )
    for line in proc.stdout:
        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")
```

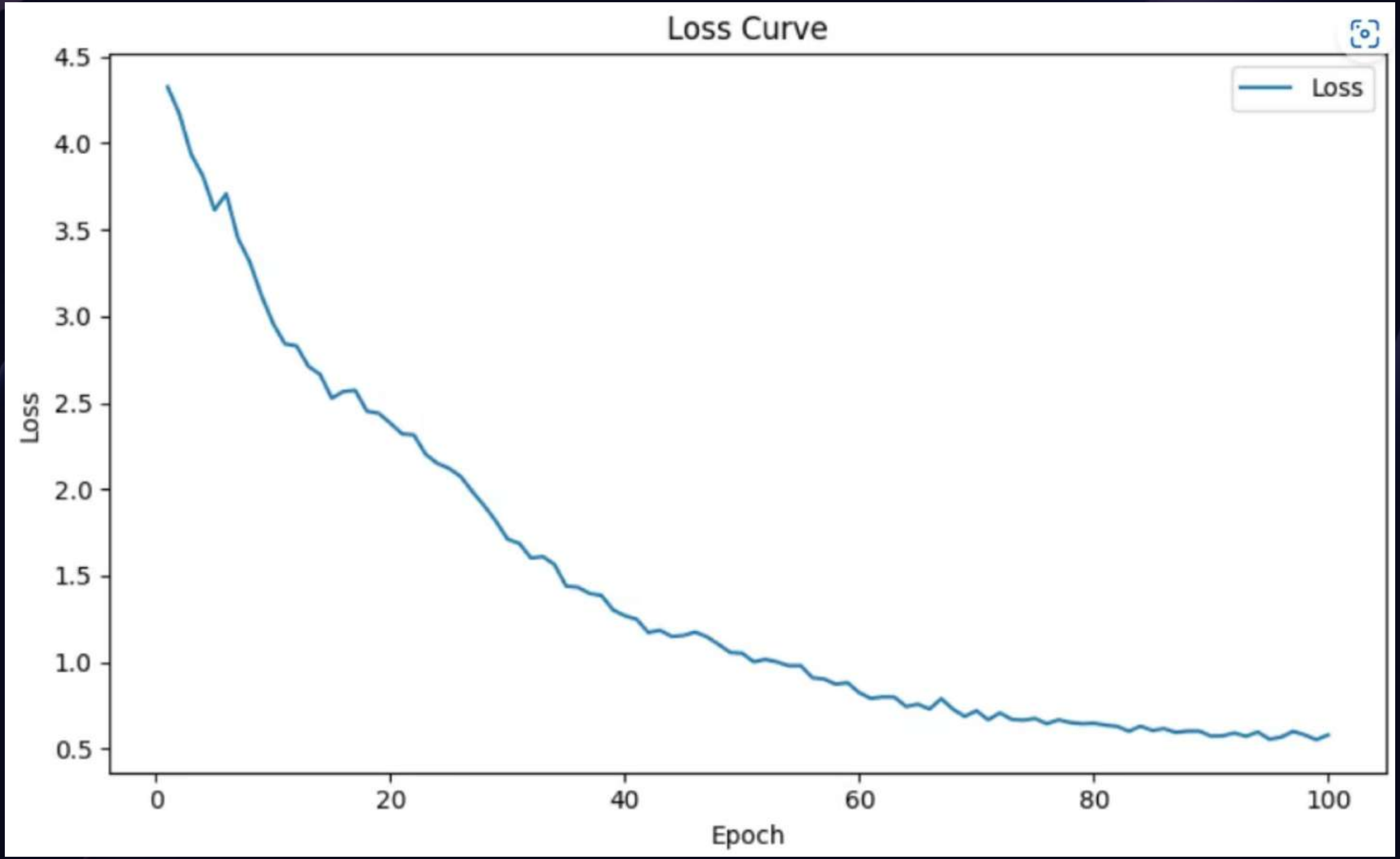


```
Epoch 1/100  batch 10/11  loss=0.6816
Epoch 1/100  batch 11/11  loss=0.7053
Val F1 Score: 34.35%
[Epoch 1] Train Loss: 0.7060
Epoch 2/100  batch 10/11  loss=0.6445
Epoch 2/100  batch 11/11  loss=0.6847
Val F1 Score: 37.64%
[Epoch 2] Train Loss: 0.6829
Epoch 3/100  batch 10/11  loss=0.6687
Epoch 3/100  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
```

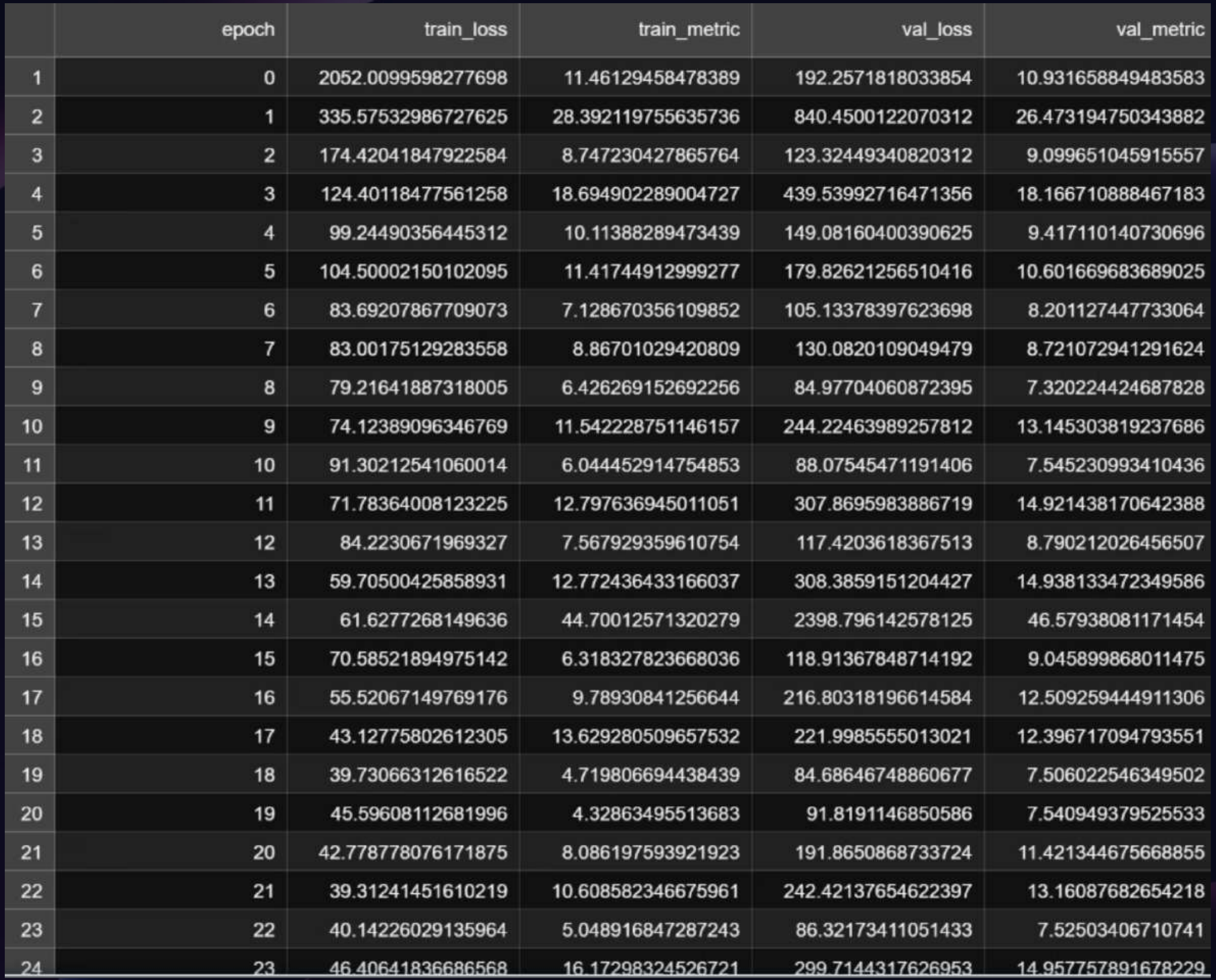
結果分析



	epoch	loss
1	1	4.326079047643221
2	2	4.16998964548111
3	3	3.9374805849332075
4	4	3.8114590438512654
5	5	3.6149786917062907
6	6	3.708397998259618
7	7	3.4497877749112935
8	8	3.3145999564574313
9	9	3.1206651009046116
10	10	2.9577285143045278
11	11	2.84063590489901
12	12	2.828573391987727
13	13	2.711246051467382
14	14	2.664225323842122
15	15	2.5263718836582623
16	16	2.565658117716129
17	17	2.5723690619835486
18	18	2.4511988243231406
19	19	2.439637827185484
20	20	2.3819847233020344
21	21	2.3195170221420436
22	22	2.3136686109579525
23	23	2.2026971074251027
24	24	2.149484180487119

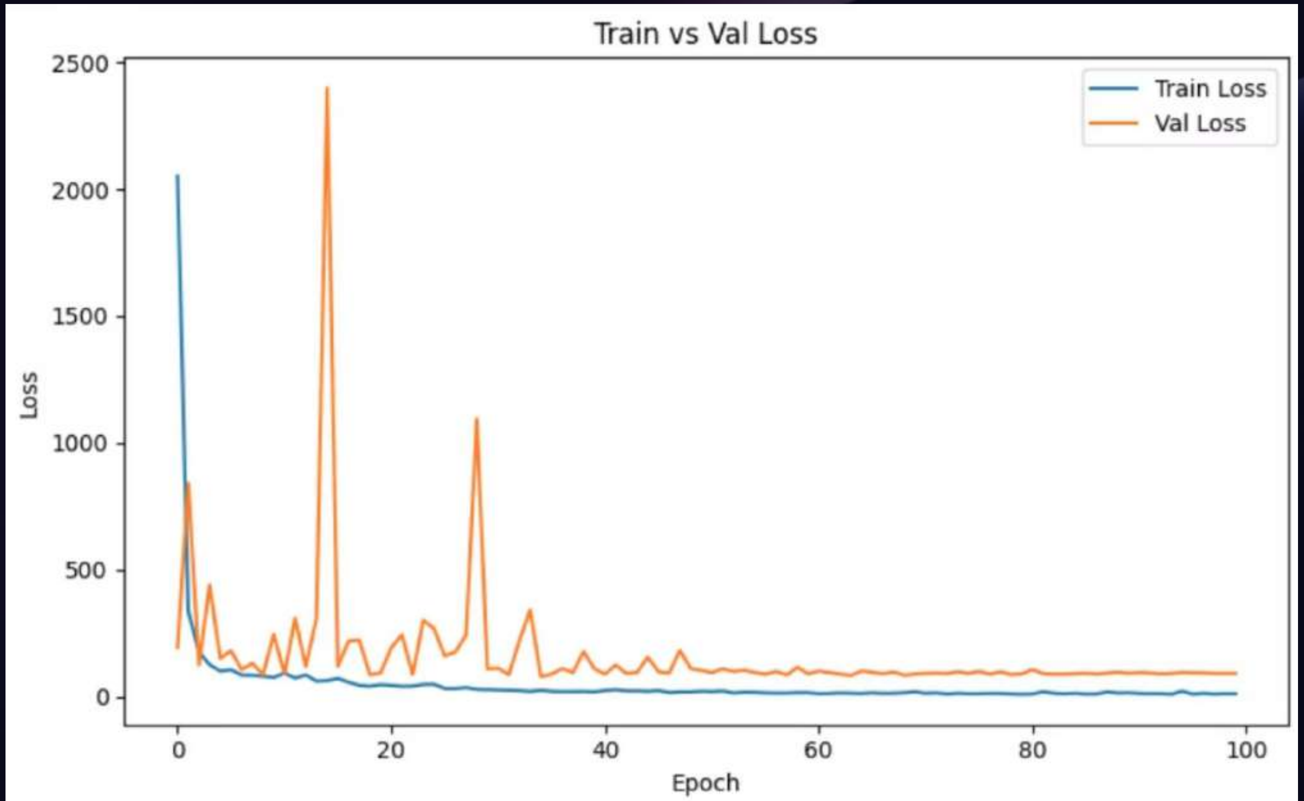


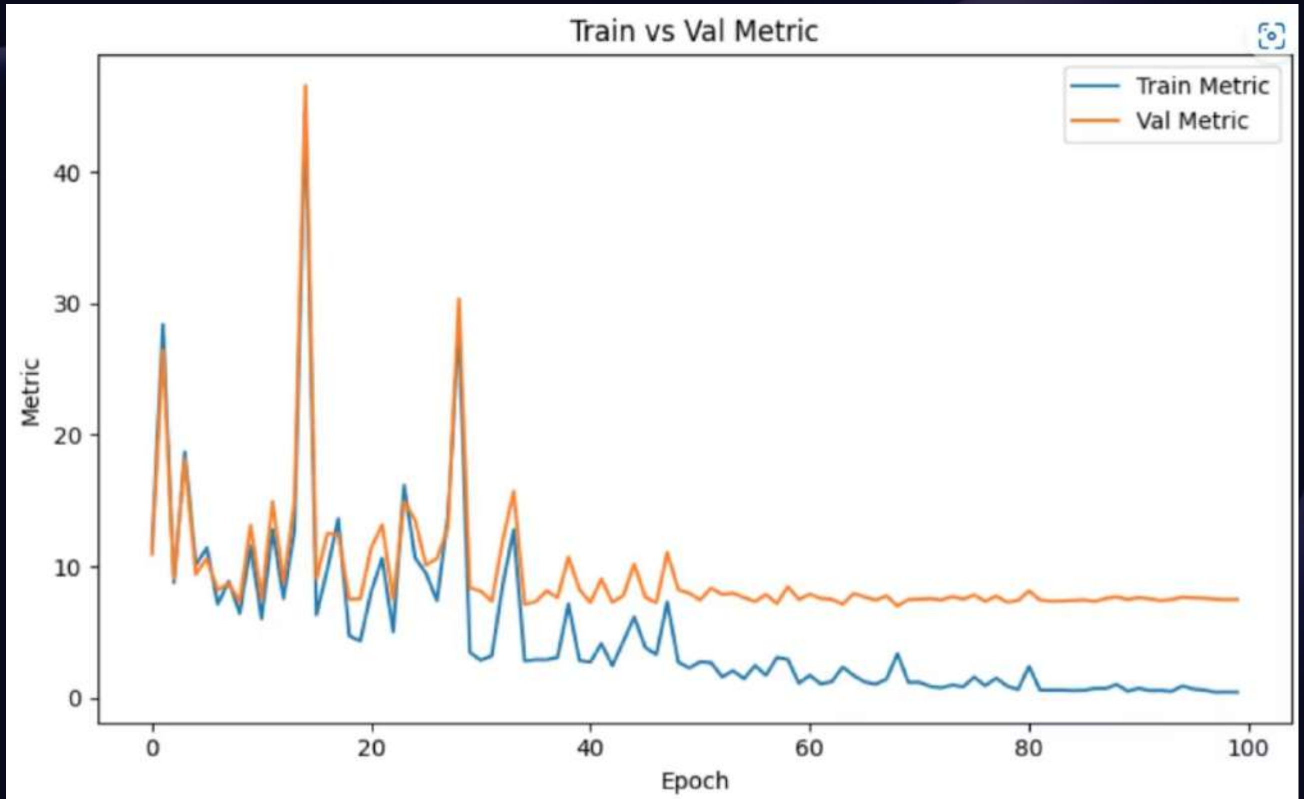
結果分析

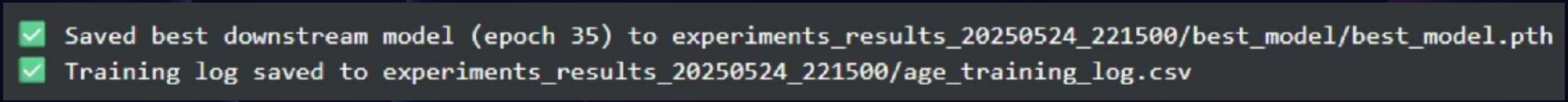


	epoch	train_loss	train_metric	val_loss	val_metric
1	0	2052.0099598277698	11.46129458478389	192.2571818033854	10.931658849483583
2	1	335.57532986727625	28.392119755635736	840.4500122070312	26.473194750343882
3	2	174.42041847922584	8.747230427865764	123.32449340820312	9.099651045915557
4	3	124.40118477561258	18.694902289004727	439.53992716471356	18.166710888467183
5	4	99.24490356445312	10.11388289473439	149.08160400390625	9.417110140730696
6	5	104.50002150102095	11.41744912999277	179.82621256510416	10.601669683689025
7	6	83.69207867709073	7.128670356109852	105.13378397623698	8.201127447733064
8	7	83.00175129283558	8.86701029420809	130.0820109049479	8.721072941291624
9	8	79.21641887318005	6.426269152692256	84.97704060872395	7.320224424687828
10	9	74.12389096346769	11.542228751146157	244.22463989257812	13.145303819237686
11	10	91.30212541060014	6.044452914754853	88.07545471191406	7.545230993410436
12	11	71.78364008123225	12.797636945011051	307.8695983886719	14.921438170642388
13	12	84.2230671969327	7.567929359610754	117.4203618367513	8.790212026456507
14	13	59.70500425858931	12.772436433166037	308.3859151204427	14.938133472349586
15	14	61.6277268149636	44.70012571320279	2398.796142578125	46.57938081171454
16	15	70.58521894975142	6.318327823668036	118.91367848714192	9.045899868011475
17	16	55.52067149769176	9.78930841256644	216.80318196614584	12.509259444911306
18	17	43.12775802612305	13.629280509657532	221.9985555013021	12.396717094793551
19	18	39.73066312616522	4.719806694438439	84.68646748860677	7.506022546349502
20	19	45.59608112681996	4.32863495513683	91.8191146850586	7.540949379525533
21	20	42.778778076171875	8.086197593921923	191.8650868733724	11.421344675668855
22	21	39.31241451610219	10.608582346675961	242.42137654622397	13.16087682654218
23	22	40.14226029135964	5.048916847287243	86.32173411051433	7.52503406710741
24	23	46.40641836686568	16.17298324526721	299.7144317626953	14.957757891678229

結果分析







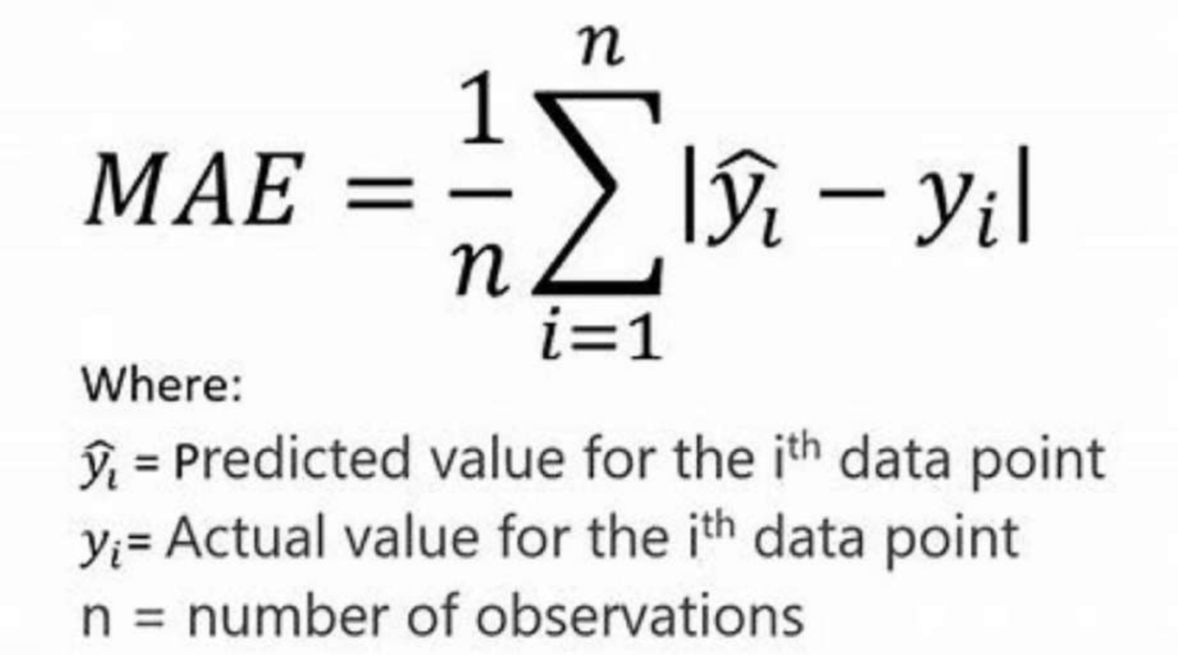
```
✅ Saved best downstream model (epoch 35) to experiments_results_20250524_221500/best_model/best_model.pth
✅ Training log saved to experiments_results_20250524_221500/age_training_log.csv
```

MAE (Mean Absolute Error)

- 預測值和實際值之間的平均平方誤差，|實際值-預測值|，相加後平均
- 能夠反映模型在預測中的整體準確性，值越小，模型的預測精度越高

$$MAE(y,\widehat{y}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y}_i|$$

MAE (Mean Absolute Error)



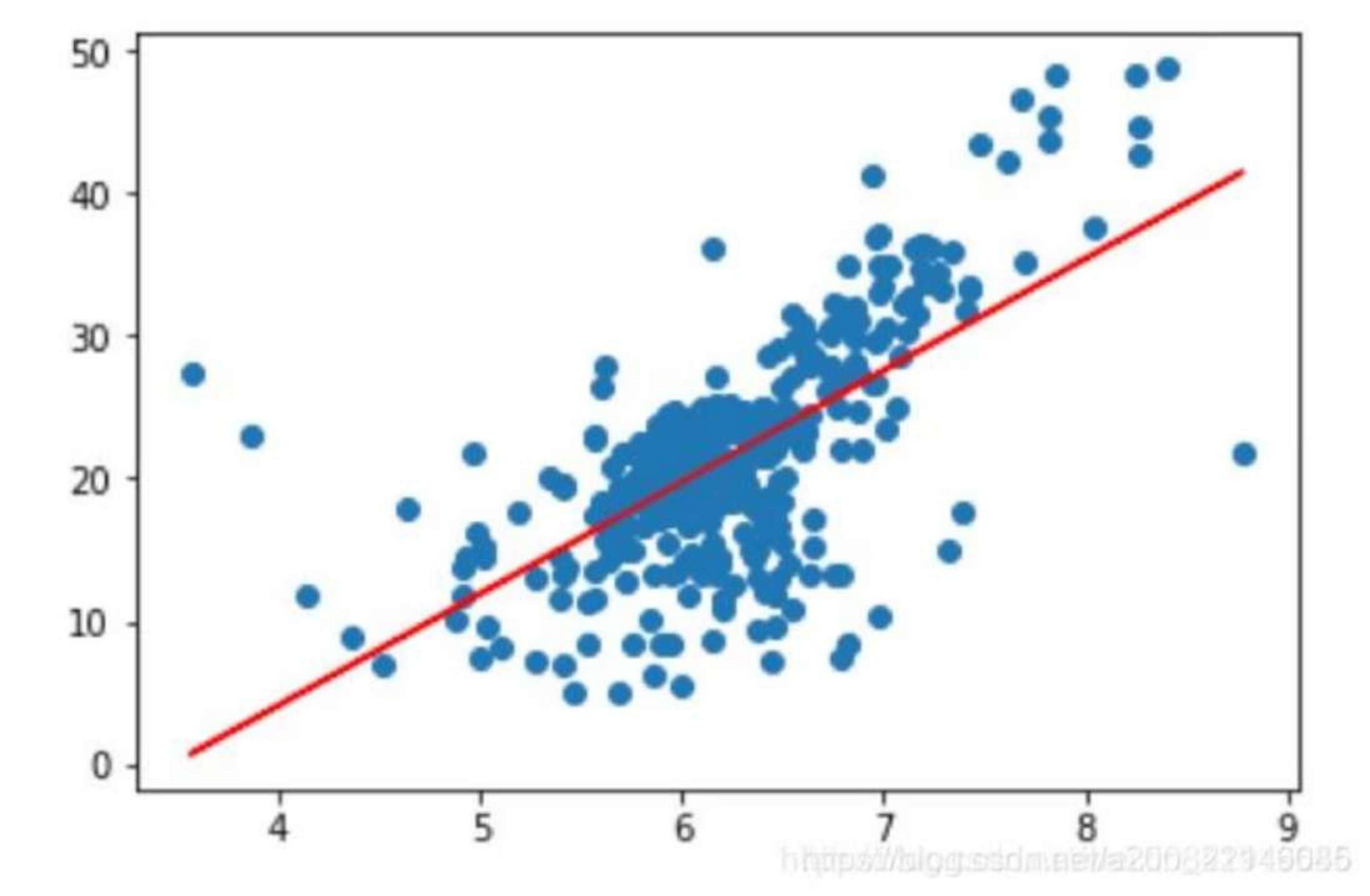
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y_i} - y_i|$$

Where:

$\widehat{y_i}$ = Predicted value for the ith data point

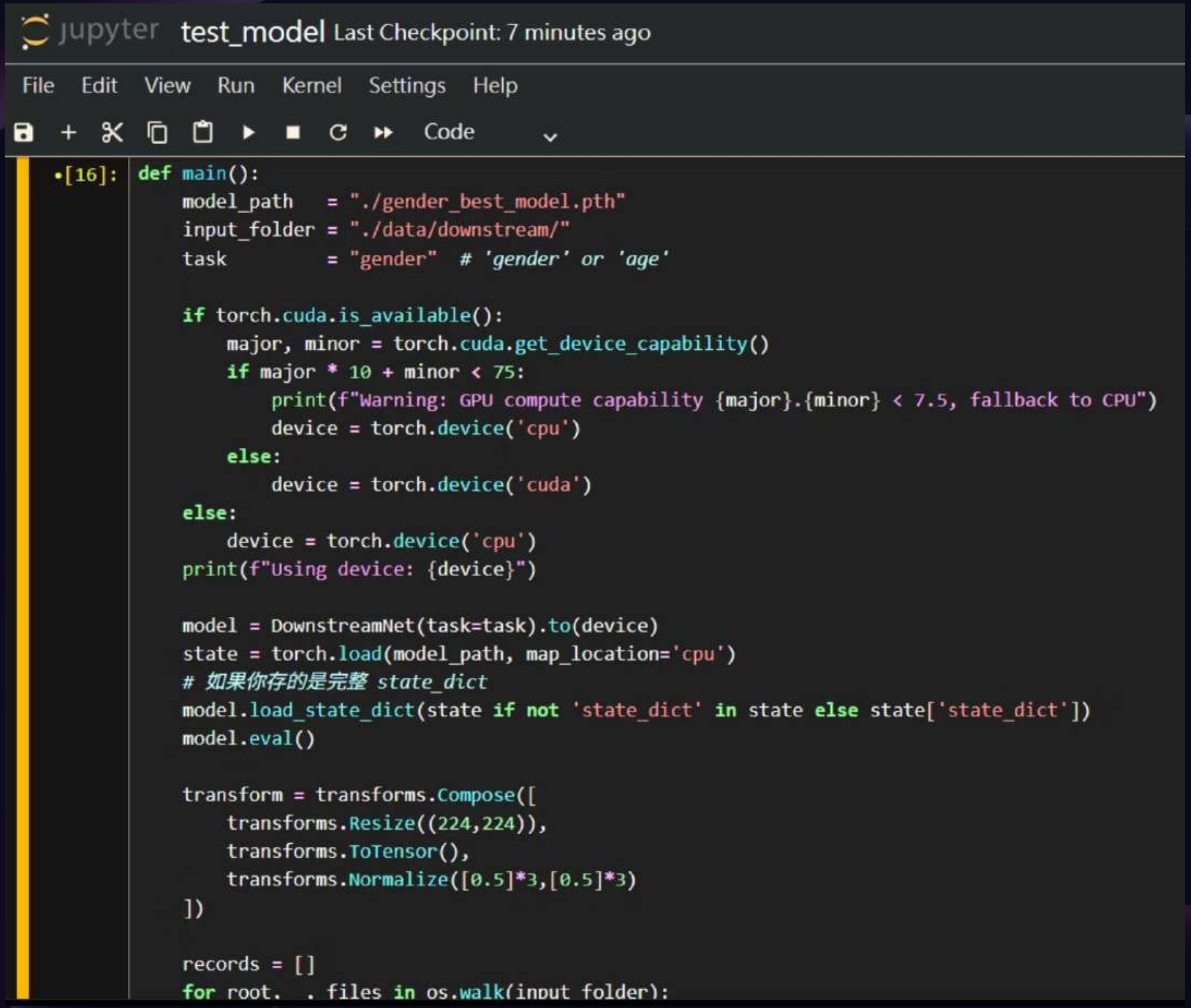
y_i= Actual value for the ith data point

n = number of observations



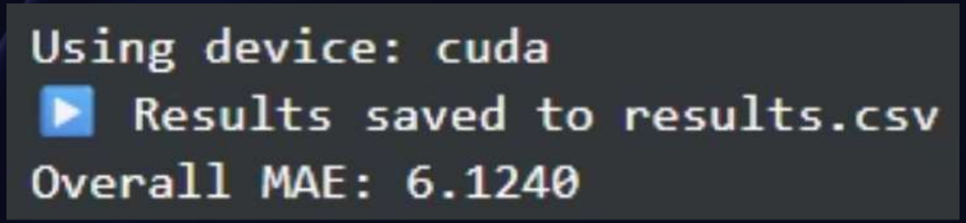
https://blog.csdn.net/a200822146085

結果分析



```
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):
```

結果分析



```
Using device: cuda
▶️ Results saved to results.csv
Overall MAE: 6.1240
```