D:\ProgramData\anaconda3\envs\tf-jie\python.exe "D:/Program Files/JetBrains/PyCharm2024.1.4/plugins/python/helpers/pydev/pydevconsole.py" --mode=client --host=127.0.0.1 --port=64597

import sys; print('Python %s on %s' % (sys.version, sys.platform))

sys.path.extend(['E:\\jk\\offline\\step3\_Transformer'])

PyDev console: starting.

Python 3.10.14 | packaged by Anaconda, Inc. | (main, Mar 21 2024, 16:20:14) [MSC v.1916 64 bit (AMD64)] on win32

runfile('E:\\jk\\offline\\step3\_Transformer\\test\_StepByStep\_3\_241106\_depth8.py', wdir='E:\\jk\\offline\\step3\_Transformer')

E:\jk\offline\step3\_Transformer\timesformer\_pytorch\timesformer\_pytorch.py:28: SyntaxWarning: "is" with a literal. Did you mean "=="?

if amt is 0:

E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:103: FutureWarning: You are using `torch.load` with `weights\_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights\_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add\_safe\_globals`. We recommend you start setting `weights\_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.

model.load\_state\_dict(torch.load('./model\_500\_3\_241106.pth'))

0it [00:00, ?it/s]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 20:

Accuracy : 0.5867

Precision: 0.7924

Recall : 0.5053

F1-Score : 0.3795

1it [04:06, 246.95s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 40:

Accuracy : 0.7289

Precision: 0.7529

Recall : 0.6921

F1-Score : 0.6940

2it [08:13, 246.50s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 60:

Accuracy : 0.7244

Precision: 0.7176

Recall : 0.7093

F1-Score : 0.7119

3it [12:19, 246.42s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 80:

Accuracy : 0.7778

Precision: 0.7736

Recall : 0.7656

F1-Score : 0.7685

4it [16:26, 246.66s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 100:

Accuracy : 0.8333

Precision: 0.8295

Recall : 0.8268

F1-Score : 0.8280

5it [20:32, 246.48s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 120:

Accuracy : 0.8711

Precision: 0.8684

Recall : 0.8660

F1-Score : 0.8671

6it [24:39, 246.57s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 140:

Accuracy : 0.9000

Precision: 0.8999

Recall : 0.8938

F1-Score : 0.8965

7it [28:46, 246.71s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

8it [32:54, 247.00s/it]Step 160:

Accuracy : 0.9356

Precision: 0.9323

Recall : 0.9364

F1-Score : 0.9341

E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 180:

Accuracy : 0.9711

Precision: 0.9741

Recall : 0.9669

F1-Score : 0.9701

9it [37:01, 247.04s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 200:

Accuracy : 0.9756

Precision: 0.9745

Recall : 0.9753

F1-Score : 0.9749

10it [41:08, 247.08s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 220:

Accuracy : 0.9756

Precision: 0.9745

Recall : 0.9753

F1-Score : 0.9749

11it [45:21, 248.89s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 240:

Accuracy : 0.9733

Precision: 0.9720

Recall : 0.9733

F1-Score : 0.9726

12it [49:28, 248.38s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 260:

Accuracy : 0.9733

Precision: 0.9720

Recall : 0.9733

F1-Score : 0.9726

13it [54:07, 257.80s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 280:

Accuracy : 0.9733

Precision: 0.9720

Recall : 0.9733

F1-Score : 0.9726

14it [59:14, 272.40s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 300:

Accuracy : 0.9733

Precision: 0.9720

Recall : 0.9733

F1-Score : 0.9726

15it [1:03:21, 264.87s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 320:

Accuracy : 0.9844

Precision: 0.9821

Recall : 0.9866

F1-Score : 0.9841

16it [1:07:28, 259.62s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 340:

Accuracy : 0.9844

Precision: 0.9821

Recall : 0.9866

F1-Score : 0.9841

17it [1:11:35, 255.71s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 360:

Accuracy : 0.9867

Precision: 0.9845

Recall : 0.9885

F1-Score : 0.9864

18it [1:15:40, 252.55s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 380:

Accuracy : 0.9911

Precision: 0.9896

Recall : 0.9924

F1-Score : 0.9909

19it [1:19:45, 250.06s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 400:

Accuracy : 0.9911

Precision: 0.9896

Recall : 0.9924

F1-Score : 0.9909

20it [1:23:49, 248.35s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 420:

Accuracy : 0.9933

Precision: 0.9921

Recall : 0.9943

F1-Score : 0.9932

21it [1:27:54, 247.23s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 440:

Accuracy : 0.9933

Precision: 0.9921

Recall : 0.9943

F1-Score : 0.9932

22it [1:31:58, 246.41s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 460:

Accuracy : 0.9933

Precision: 0.9921

Recall : 0.9943

F1-Score : 0.9932

23it [1:36:01, 245.51s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 480:

Accuracy : 0.9933

Precision: 0.9921

Recall : 0.9943

F1-Score : 0.9932

24it [1:40:05, 244.93s/it]E:\jk\offline\step3\_Transformer\test\_StepByStep\_3\_241106\_depth8.py:46: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires\_grad\_(True), rather than torch.tensor(sourceTensor).

data = torch.tensor(self.\_load\_txt(self.data\_paths[idx]), dtype=torch.float32)

Step 500:

Accuracy : 0.9933

Precision: 0.9921

Recall : 0.9943

F1-Score : 0.9932

25it [1:44:09, 249.97s/it]

Step Accuracy Precision Recall F1-Score

0 20 0.586667 0.792411 0.505319 0.379540

1 40 0.728889 0.752941 0.692058 0.693973

2 60 0.724444 0.717579 0.709274 0.711872

3 80 0.777778 0.773590 0.765592 0.768519

4 100 0.833333 0.829465 0.826823 0.828033

5 120 0.871111 0.868360 0.866026 0.867119

6 140 0.900000 0.899905 0.893840 0.896478

7 160 0.935556 0.932278 0.936394 0.934095

8 180 0.971111 0.974118 0.966928 0.970094

9 200 0.975556 0.974548 0.975252 0.974895

10 220 0.975556 0.974548 0.975252 0.974895

11 240 0.973333 0.971964 0.973343 0.972633

12 260 0.973333 0.971964 0.973343 0.972633

13 280 0.973333 0.971964 0.973343 0.972633

14 300 0.973333 0.971964 0.973343 0.972633

15 320 0.984444 0.982051 0.986641 0.984092

16 340 0.984444 0.982051 0.986641 0.984092

17 360 0.986667 0.984536 0.988550 0.986355

18 380 0.991111 0.989583 0.992366 0.990891

19 400 0.991111 0.989583 0.992366 0.990891

20 420 0.993333 0.992147 0.994275 0.993163

21 440 0.993333 0.992147 0.994275 0.993163

22 460 0.993333 0.992147 0.994275 0.993163

23 480 0.993333 0.992147 0.994275 0.993163

24 500 0.993333 0.992147 0.994275 0.993163