mlflow

安装与使用

- 安装

```
pip install mlflow # 在命令行中输入
```

- 使用

```
import mlflow
```

开启训练

```
mlflow.set_experiment('your_experiment_name') # 设置实验名称

with mlflow.start_run(run_name='your_run_name'): # 设置训练名称
    model.train()
```

- 可以在训练中嵌入子训练

```
with mlflow.start_run(run_name='parent_run_name'): # 设置母训练名称
    with mlflow.start_run(run_name='child_run_name1', nested=True): # 设置子训练1的名称
        model1.train()

    with mlflow.start_run(run_name='child_run_name2', nested=True): # 设置子训练2的名称
        model2.train()
```

- 指定run_id可以继续之前的训练记录

```
with mlflow.start_run(run_id='past_run_id'): # 当指定run_id，其他的参数都将无效
    model.train()
```

> 注意：继续训练记录并不等于继续训练，只是将训练过程中需要记录的东西继续记录到同一个训练目录下。

超参记录

- 每次训练记录一次，例如学习速率、epoch数量等

```
mlflow.log_param(param, 'param_name') # 记录一个参数
mlflow.log_params({'param_name1':param1,
                   'param_name2':param2
                   }) # 记录多个参数
```

指标记录

- 每个epoch记录一次，例如loss、accuracy等

```
mlflow.log_metric(metric, 'metric_name') # 记录一个指标
mlflow.metrics({'metric_name1':metric1,
                'metric_name2':metric2
                }) # 记录多个指标
```

文件记录

- 记录训练过程中需要保存的文件

```
mlflow.log_artifact('artifact_path') # 记录一个文件
mlflow.log_artifacts('folder_path') # 记录一个文件夹下所有文件
```

- 可以直接记录matplotilib或者plotly中的Figure

```
mlflow.log_figure(Figure, 'figure_path') # 记录图片
```

- 也可以直接记录文字

```
mlflow.log_text(text, 'figure_path') # 记录文字
```

> 以上所有文件都将存在对应训练目录中的/artifacts文件夹下，所以输入的路径都是相对路径

训练标签与训练id搜索

- 训练标签
    - 除了记录超参、指标、文件，还可以给不同的训练贴标签
    - 默认记录的标签有

mlflow.runName	Human readable name that identifies this run.
mlflow.note.content	A descriptive note about this run. This reserved tag is not set automatically and can be overridden by the user to include additional information about the run. The content is displayed on the run's page under the Notes section.
mlflow.parentRunId	The ID of the parent run, if this is a nested run.
mlflow.user	Identifier of the user who created the run.
mlflow.source.type	Source type. Possible values: "NOTEBOOK", "JOB", "PROJECT", "LOCAL", and "UNKNOWN"
mlflow.source.name	Source identifier (e.g., GitHub URL, local Python filename, name of notebook)
mlflow.source.git.commit	Commit hash of the executed code, if in a git repository.
mlflow.source.git.branch	Name of the branch of the executed code, if in a git repository.
mlflow.source.git.repoURL	URL that the executed code was cloned from.
mlflow.project.env	The runtime context used by the MLflow project. Possible values: "docker" and "conda".
mlflow.project.entryPoint	Name of the project entry point associated with the current run, if any.
mlflow.docker.image.name	Name of the Docker image used to execute this run.
mlflow.docker.image.id	ID of the Docker image used to execute this run.
mlflow.log-model.history	(Experimental) Model metadata collected by log-model calls. Includes the serialized form of the MLModel model files logged to a run, although the exact format and information captured is subject to change.

    - 自定义标签

```
mlflow.set_tag('tag_name', tag) # 设置一个标签
mlflow.set_tags({'tag_name1':tag1,
                 'tag_name2':tag2
                 }) # 设置多个标签
```

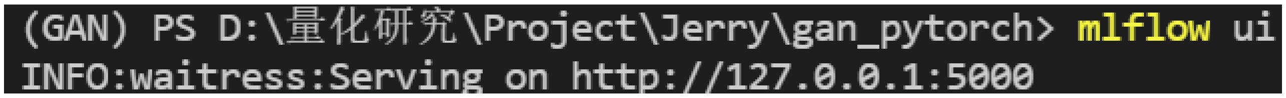
- 训练id搜索
    - 训练的id是一串难以记忆的hash key，可以用训练记录的标签来搜索对应的id

```
run_name = 'name_to_search'
histories = mlflow.search_runs(filter_string=f'tags.mlflow.runName = "{run_name}"')
# 返回pandas.DataFrame，columns是所有标签，index为0的是最新的训练
recent_run_id = histories['run_id'][0]
```

训练记录可视化

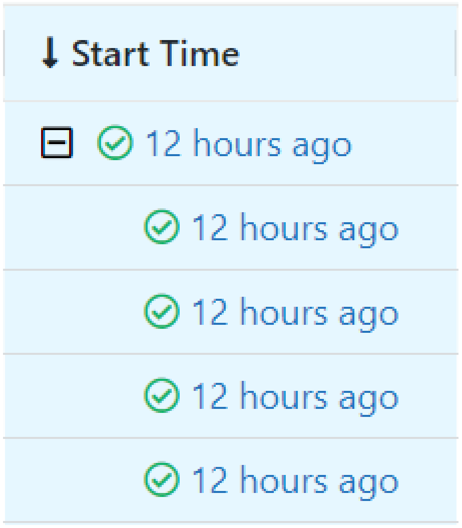
1. 在命令行中输入

```
mlflow ui
```



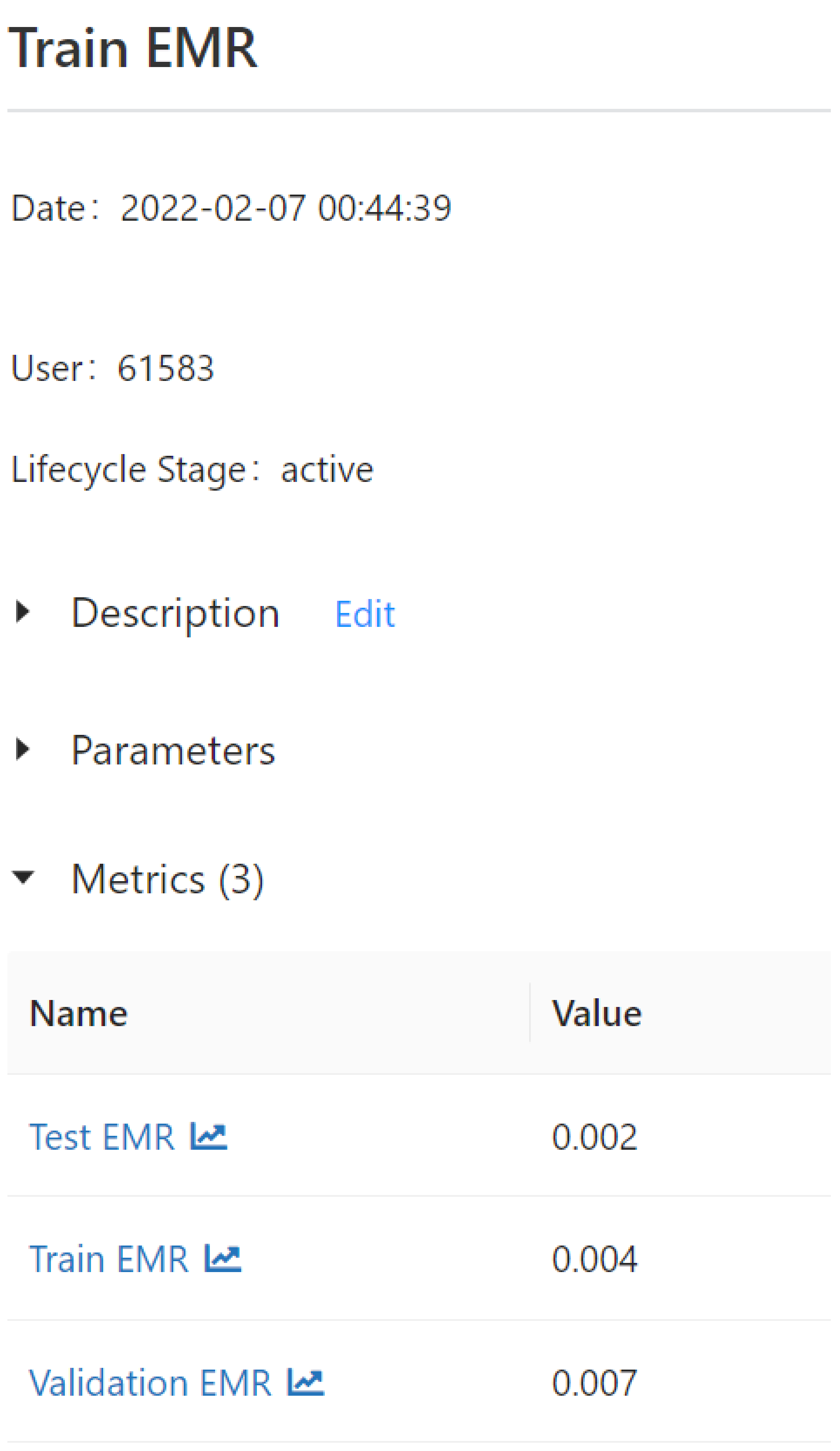
```
(GAN) PS D:\量化研究\Project\Jerry\gan_pytorch> mlflow ui
INFO:waitress:Serving on http://127.0.0.1:5000
```

2. 将网址复制到浏览器打开（vscode可以直接按住Ctrl并点击链接）



↓ Start Time
⊟ ✓ 12 hours ago
✓ 12 hours ago
✓ 12 hours ago
✓ 12 hours ago
✓ 12 hours ago

- ui中这种蓝色的超链接都是可以点击进去看详细信息的，上图是每个训练开始的时间，点进去可以看到每个训练的详细记录情况


Train EMR

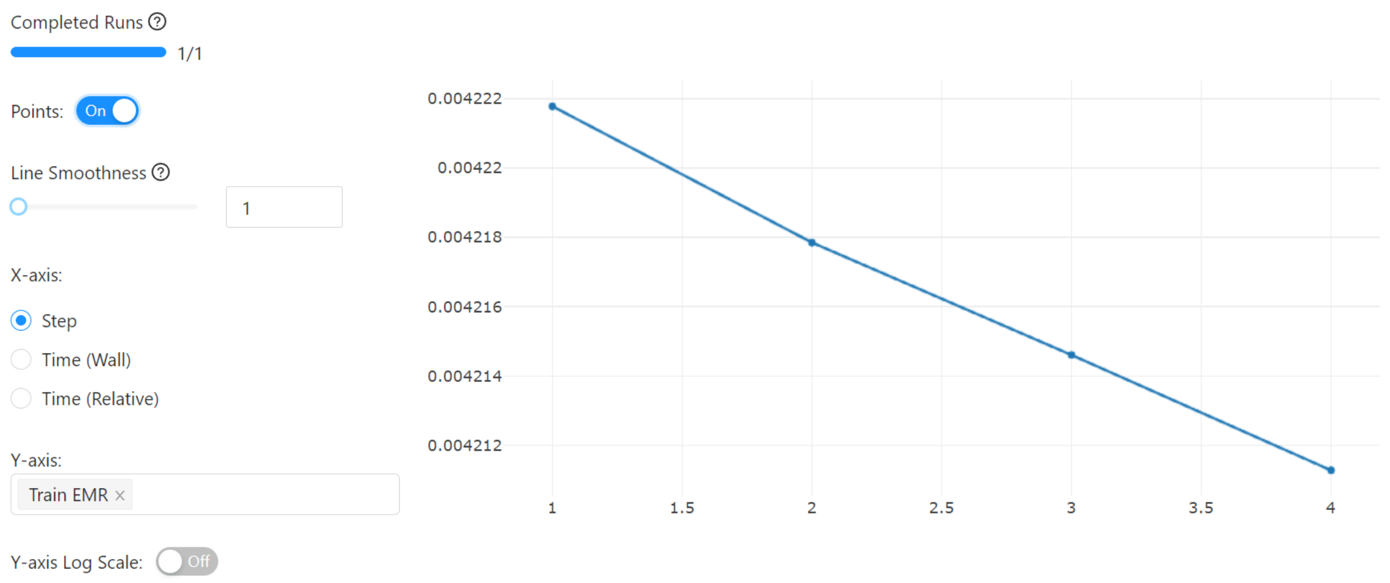
Date: 2022-02-07 00:44:39

User: 61583

Lifecycle Stage: active

Name	Value
Test EMR	0.002
Train EMR	0.004
Validation EMR	0.007

- 点击记录的指标可以显示指标的变化，横坐标可以选step也可以选时间


Completed Runs ⓘ 1/1

Points: On

Line Smoothness ⓘ 1

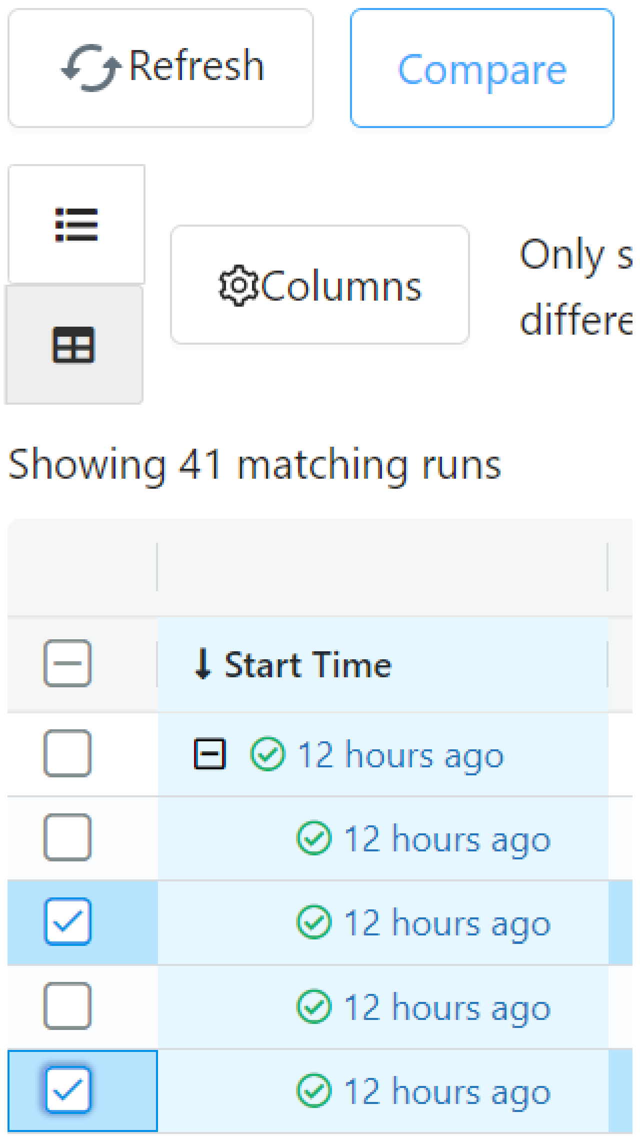
X-axis:

[✓ Step] Time (Wall) Time (Relative)

Y-axis: Train EMR ×

Y-axis Log Scale: Off

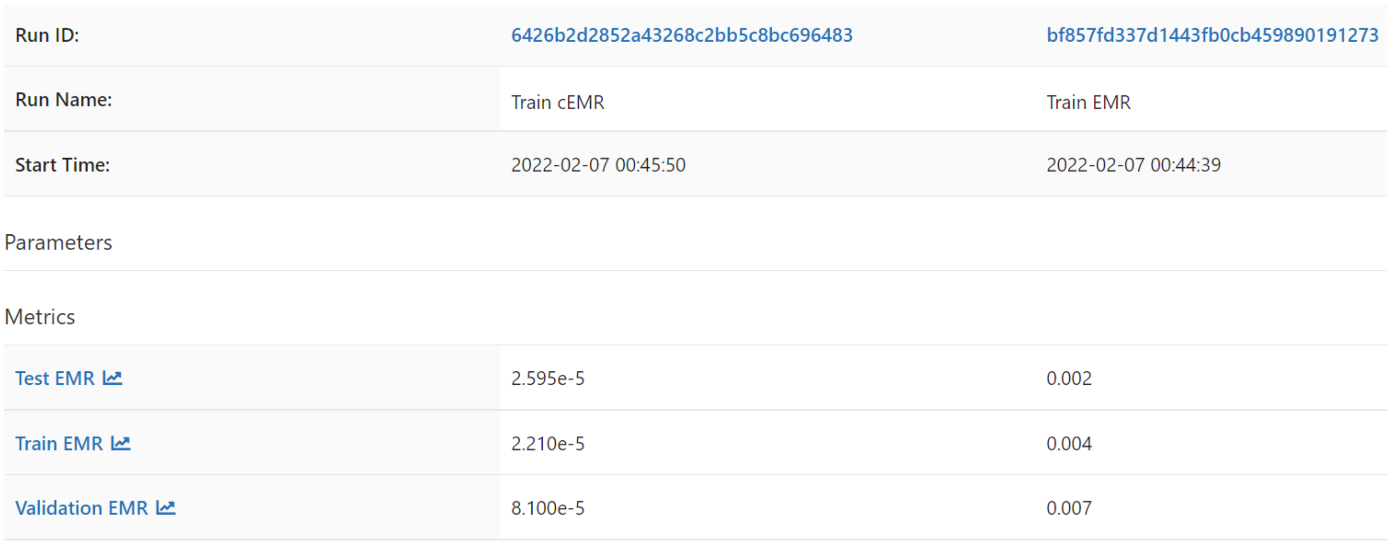
- 回到主界面，不同的训练之间还可以进行比较



Showing 41 matching runs

	Start Time
☐	12 hours ago
☐	12 hours ago
✓	12 hours ago
☐	12 hours ago
✓	12 hours ago

- 选中训练并点击Compare后，可以看到比对的详细信息



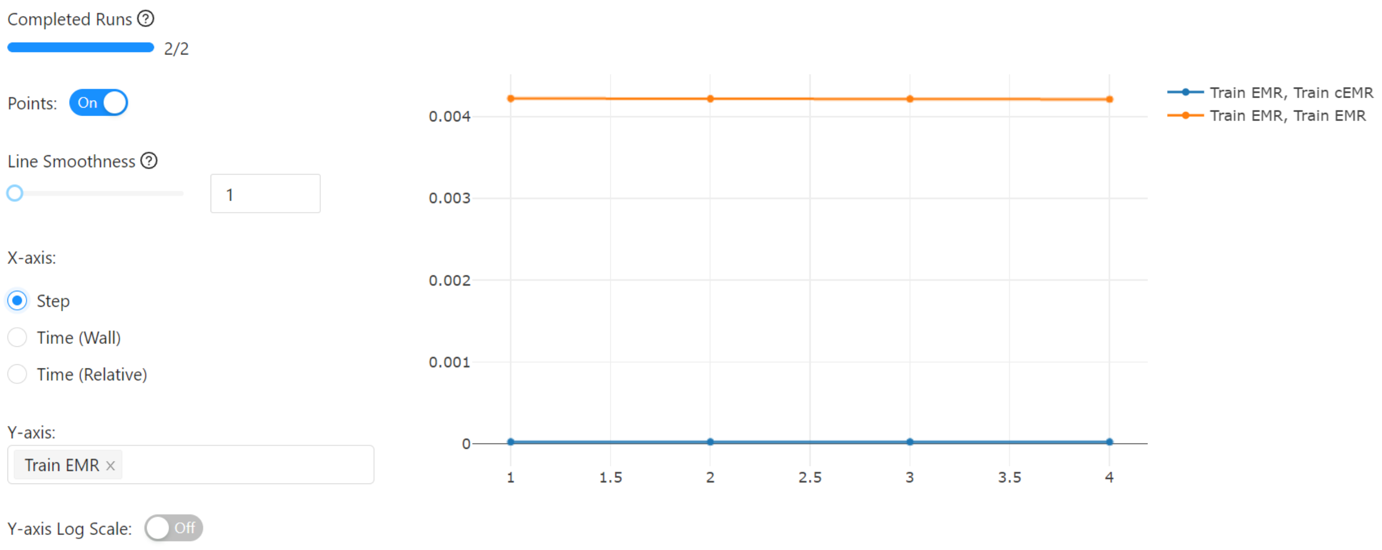
Run ID:	6426b2d2852a43268c2bb5c8bc696483	bf857fd337d1443fb0cb459890191273
Run Name:	Train cEMR	Train EMR
Start Time:	2022-02-07 00:45:50	2022-02-07 00:44:39

Parameters

Metrics

Test EMR	2.595e-5	0.002
Train EMR	2.210e-5	0.004
Validation EMR	8.100e-5	0.007

- 点击记录的指标可以看到指标变化的比对



Completed Runs 2/2

Points: On

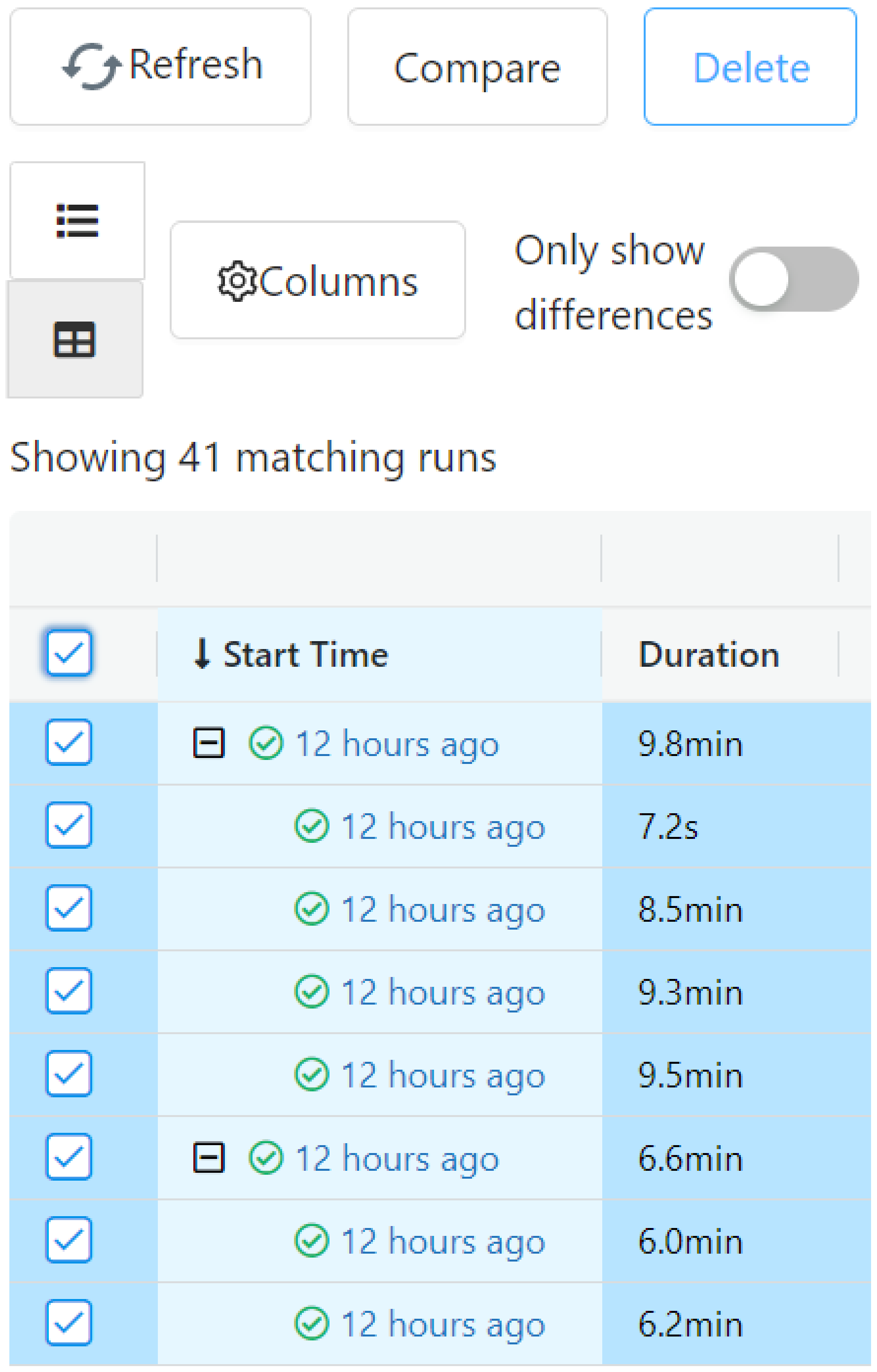
Line Smoothness 1

X-axis:
- [✓ Step]
- Time (Wall)
- Time (Relative)

Y-axis: Train EMR

Y-axis Log Scale: Off

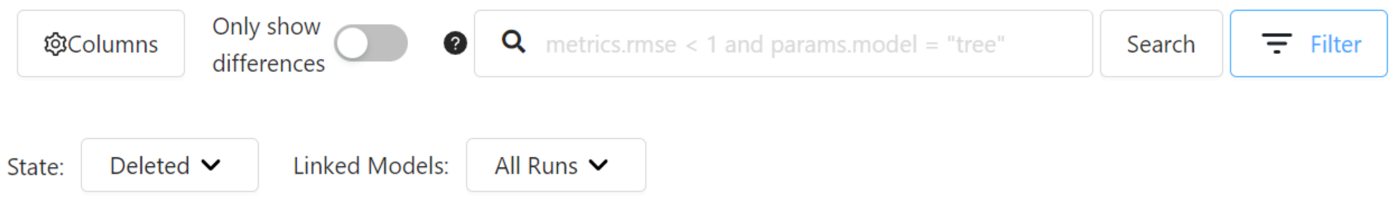
- 在ui中可以删除训练



Showing 41 matching runs

✓	Start Time	Duration
✓	⊟ 12 hours ago	9.8min
✓	12 hours ago	7.2s
✓	12 hours ago	8.5min
✓	12 hours ago	9.3min
✓	12 hours ago	9.5min
✓	⊟ 12 hours ago	6.6min
✓	12 hours ago	6.0min
✓	12 hours ago	6.2min

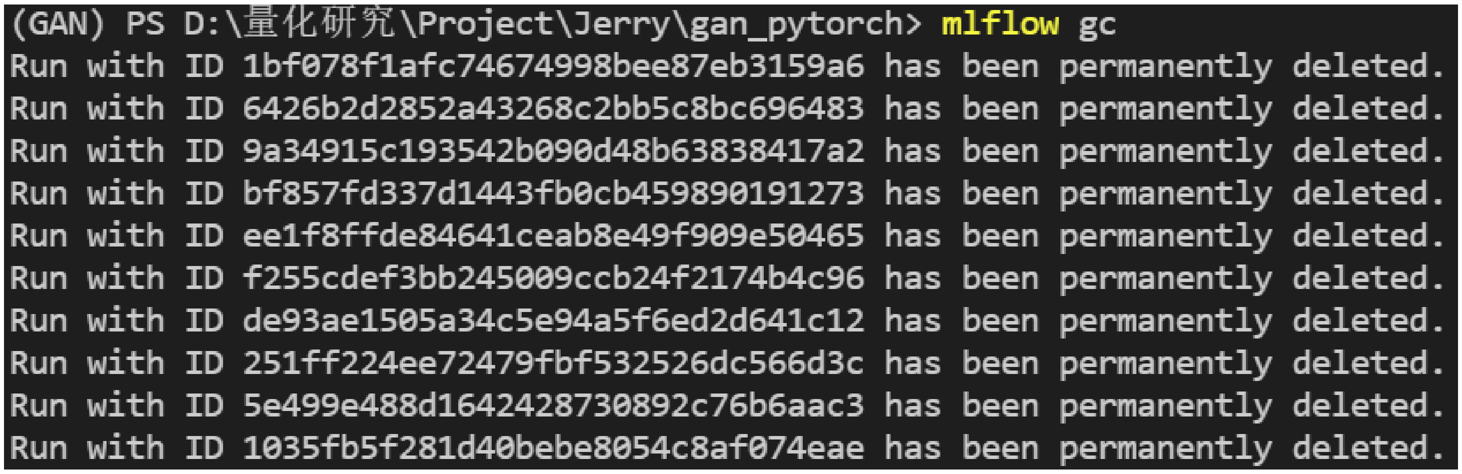
- 但删除只是让训练的State变成Deleted，还是可以在Filter中筛选出来。这个状态下的训练会保留30天，30天后自动删除



State: Deleted

- 如果想要立马删除，则在ui中删除后，到命令行中键入

```
mlflow gc
```



```
(GAN) PS D:\量化研究\Project\Jerry\gan_pytorch> mlflow gc
Run with ID 1bf078f1afc74674998bee87eb3159a6 has been permanently deleted.
Run with ID 6426b2d2852a43268c2bb5c8bc696483 has been permanently deleted.
Run with ID 9a34915c193542b090d48b63838417a2 has been permanently deleted.
Run with ID bf857fd337d1443fb0cb459890191273 has been permanently deleted.
Run with ID ee1f8ffde84641ceab8e49f909e50465 has been permanently deleted.
Run with ID f255cdef3bb245009ccb24f2174b4c96 has been permanently deleted.
Run with ID de93ae1505a34c5e94a5f6ed2d641c12 has been permanently deleted.
Run with ID 251ff224ee72479fbf532526dc566d3c has been permanently deleted.
Run with ID 5e499e488d1642428730892c76b6aac3 has been permanently deleted.
Run with ID 1035fb5f281d40bebe8054c8af074eae has been permanently deleted.
```

与其他机器学习包的协作

- mlflow支持与各种机器学习包的协作，比如pytorch、tensorflow、sklearn、keras等等
- 使用autolog可以自动记录模型的参数和指标

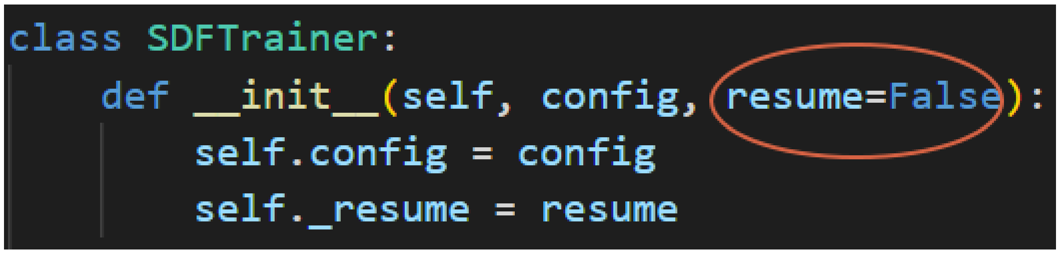
```
mlflow.sklearn.autolog()
mlflow.pytorch.autolog() # 似乎只支持pytorch_lightening，只装pytorch用不了
...
```

- 每种机器学习包都有特定的函数可以使用，以pytorch为例，可以用mlflow.pytorch.log_state_dict来记录checkpoint（记录到artifacts中），用mlflow.pytorch.load_state_dict来加载checkpoint，替代pytorch自带的函数

```
mlflow.pytorch.log_state_dict(checkpoint, 'folder_path') # 将pth文件存到artifacts中的指定文件夹内

path = mlflow.get_artifact_uri('folder_path') # 获取指定文件夹的artifact uri（不是本地路径）
mlflow.pytorch.load_state_dict(checkpoint, path) # 加载指定路径中的pth文件
```

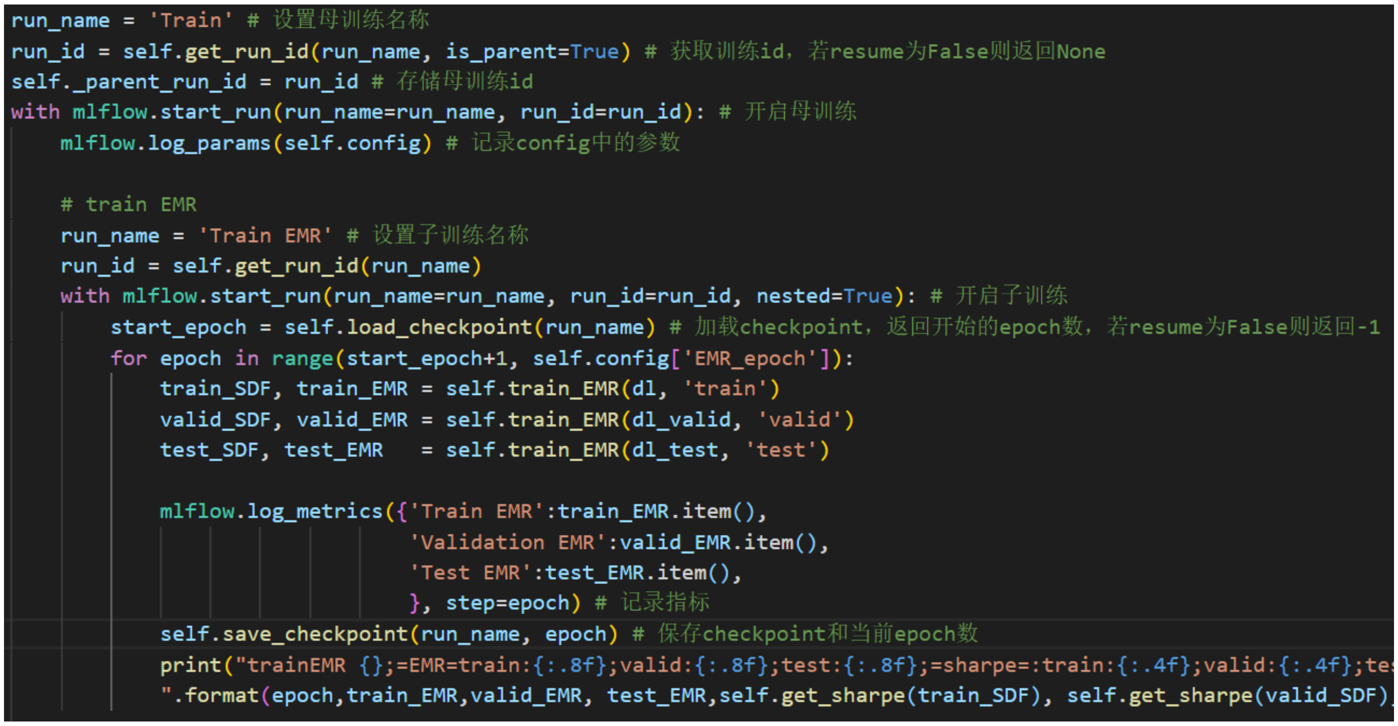
应用实例

- 以gan_pytorch为例，使用mlflow记录参数与指标，并实现断点续训
- 加入resume参数，True为断点续训，False为从头训练



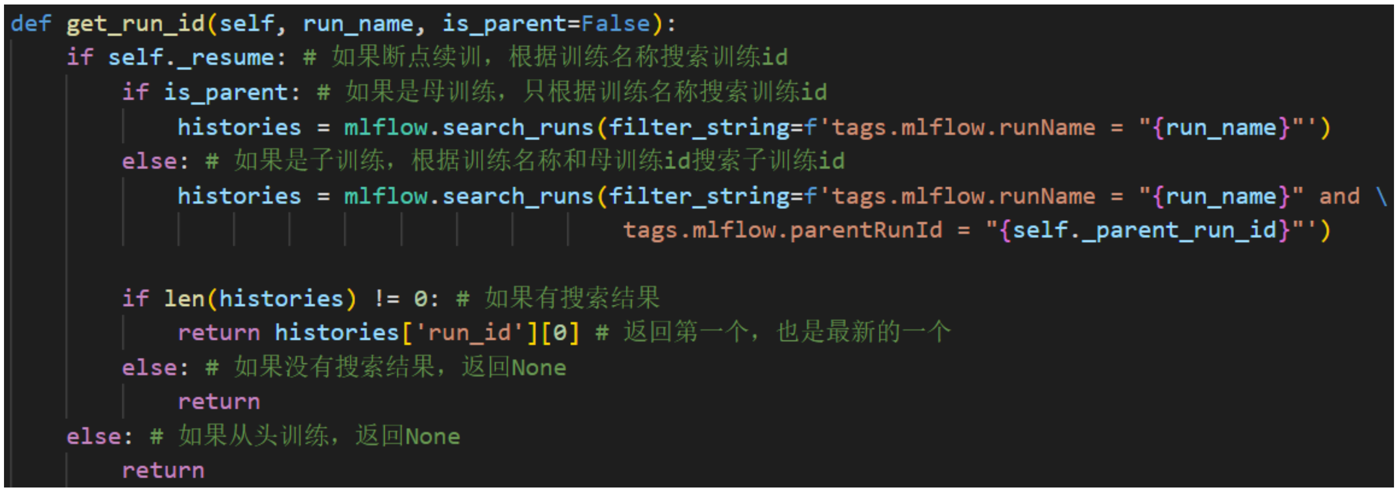
```
class SDFTrainer:
    def __init__(self, config, resume=False):
        self.config = config
        self._resume = resume
```

- 部分主代码

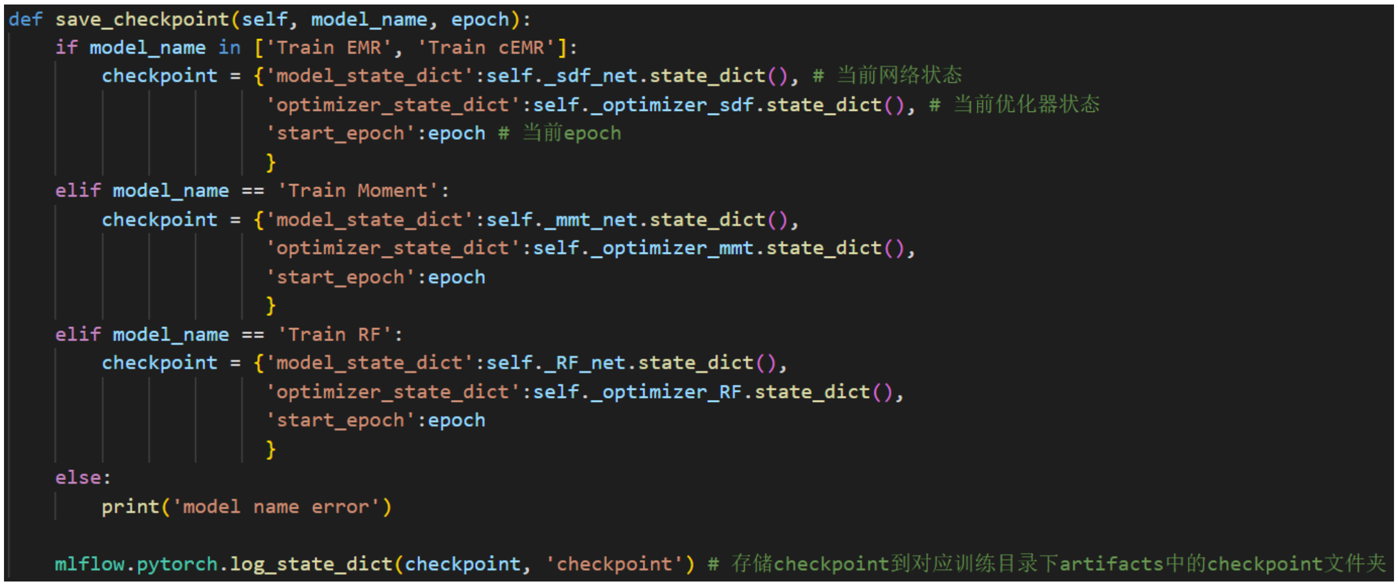


```python
run_name = 'Train' # 设置母训练名称
run_id = self.get_run_id(run_name, is_parent=True) # 获取训练id，若resume为False则返回None
self._parent_run_id = run_id # 存储母训练id
with mlflow.start_run(run_name=run_name, run_id=run_id): # 开启母训练
    mlflow.log_params(self.config) # 记录config中的参数

    # train EMR
    run_name = 'Train EMR' # 设置子训练名称
    run_id = self.get_run_id(run_name)
    with mlflow.start_run(run_name=run_name, run_id=run_id, nested=True): # 开启子训练
        start_epoch = self.load_checkpoint(run_name) # 加载checkpoint，返回开始的epoch数，若resume为False则返回-1
        for epoch in range(start_epoch+1, self.config['EMR_epoch']):
            train_SDF, train_EMR = self.train_EMR(dl, 'train')
            valid_SDF, valid_EMR = self.train_EMR(dl_valid, 'valid')
            test_SDF, test_EMR   = self.train_EMR(dl_test, 'test')

            mlflow.log_metrics({'Train EMR':train_EMR.item(),
                                'Validation EMR':valid_EMR.item(),
                                'Test EMR':test_EMR.item(),
                                }, step=epoch) # 记录指标
            self.save_checkpoint(run_name, epoch) # 保存checkpoint和当前epoch数
            print("trainEMR {};=EMR=train:{:.8f};valid:{:.8f};test:{:.8f};=sharpe=:train:{:.4f};valid:{:.4f};te
            ".format(epoch,train_EMR,valid_EMR, test_EMR,self.get_sharpe(train_SDF), self.get_sharpe(valid_SDF)
```

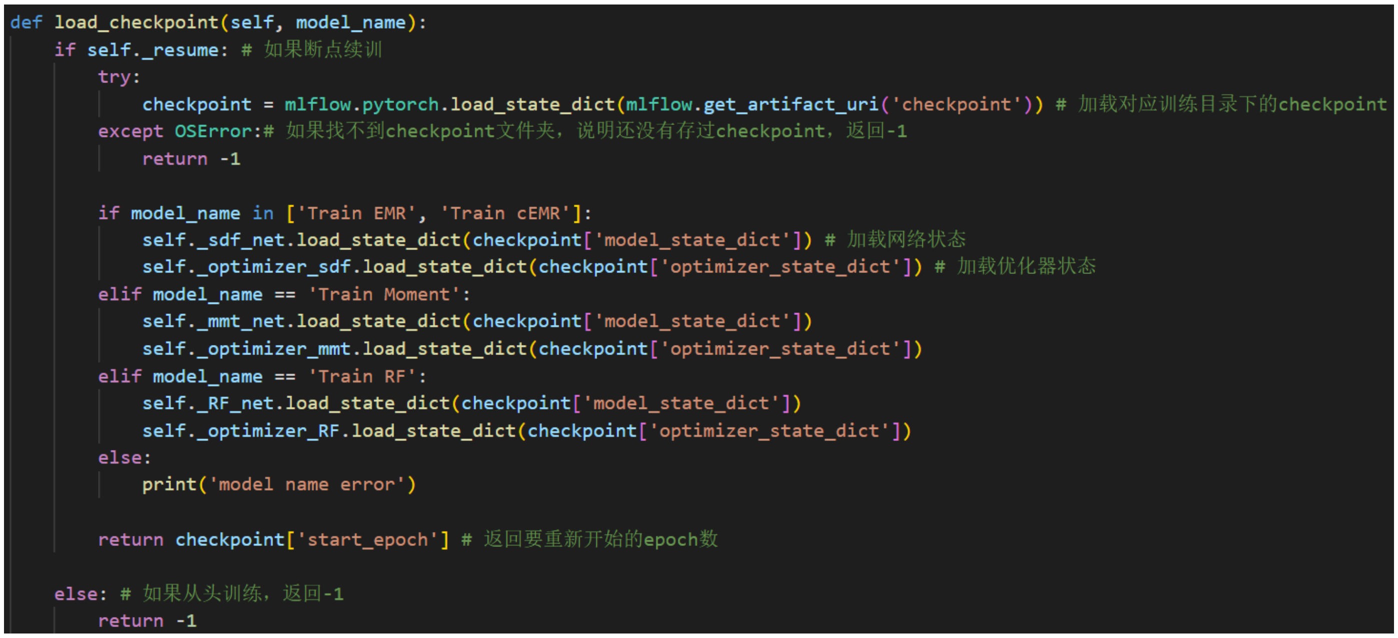
- 相关函数定义



```python
def get_run_id(self, run_name, is_parent=False):
    if self._resume: # 如果断点续训，根据训练名称搜索训练id
        if is_parent: # 如果是母训练，只根据训练名称搜索训练id
            histories = mlflow.search_runs(filter_string=f'tags.mlflow.runName = "{run_name}"')
        else: # 如果是子训练，根据训练名称和母训练id搜索子训练id
            histories = mlflow.search_runs(filter_string=f'tags.mlflow.runName = "{run_name}" and \
                                        tags.mlflow.parentRunId = "{self._parent_run_id}"')

        if len(histories) != 0: # 如果有搜索结果
            return histories['run_id'][0] # 返回第一个，也是最新的一个
        else: # 如果没有搜索结果，返回None
            return
    else: # 如果从头训练，返回None
        return
```



```python
def save_checkpoint(self, model_name, epoch):
    if model_name in ['Train EMR', 'Train cEMR']:
        checkpoint = {'model_state_dict':self._sdf_net.state_dict(), # 当前网络状态
                      'optimizer_state_dict':self._optimizer_sdf.state_dict(), # 当前优化器状态
                      'start_epoch':epoch # 当前epoch
                      }
    elif model_name == 'Train Moment':
        checkpoint = {'model_state_dict':self._mmt_net.state_dict(),
                      'optimizer_state_dict':self._optimizer_mmt.state_dict(),
                      'start_epoch':epoch
                      }
    elif model_name == 'Train RF':
        checkpoint = {'model_state_dict':self._RF_net.state_dict(),
                      'optimizer_state_dict':self._optimizer_RF.state_dict(),
                      'start_epoch':epoch
                      }
    else:
        print('model name error')

    mlflow.pytorch.log_state_dict(checkpoint, 'checkpoint') # 存储checkpoint到对应训练目录下artifacts中的checkpoint文件夹
```


```python
def load_checkpoint(self, model_name):
    if self._resume: # 如果断点续训
        try:
            checkpoint = mlflow.pytorch.load_state_dict(mlflow.get_artifact_uri('checkpoint')) # 加载对应训练目录下的checkpoint
        except OSError:# 如果找不到checkpoint文件夹，说明还没有存过checkpoint，返回-1
            return -1

        if model_name in ['Train EMR', 'Train cEMR']:
            self._sdf_net.load_state_dict(checkpoint['model_state_dict']) # 加载网络状态
            self._optimizer_sdf.load_state_dict(checkpoint['optimizer_state_dict']) # 加载优化器状态
        elif model_name == 'Train Moment':
            self._mmt_net.load_state_dict(checkpoint['model_state_dict'])
            self._optimizer_mmt.load_state_dict(checkpoint['optimizer_state_dict'])
        elif model_name == 'Train RF':
            self._RF_net.load_state_dict(checkpoint['model_state_dict'])
            self._optimizer_RF.load_state_dict(checkpoint['optimizer_state_dict'])
        else:
            print('model name error')

        return checkpoint['start_epoch'] # 返回要重新开始的epoch数

    else: # 如果从头训练，返回-1
        return -1
```