耳朵关键点检测-可视化训练日志

训练模型时在 `work_dirs` 目录生成记录训练日志，解析其中损失函数、评估指标等信息，并可视化。

同济子豪兄：[https://space.bilibili.com/1900783 (https://space.bilibili.com/1900783)](https://space.bilibili.com/1900783)

进入mmdetection主目录

In [1]:

```
import os
os.chdir('mmpose')
```

导入工具包

In [2]:

```
import pandas as pd
from tqdm import tqdm

import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['axes.unicode_minus']=False  # 用来正常显示负号
```

载入训练日志

In [3]:

```
# 日志文件路径
log_path = 'work_dirs/rtmpose-s-ear/20230604_163650/vis_data/scalars.json'
```

In [4]:

```
with open(log_path, "r") as f:
    json_list = f.readlines()
```

In [5]:

```
len(json_list)
```

Out[5]:

```
1830
```

In [6]:

```
eval(json_list[4])
```

Out[6]:

```
{'lr': 0.0008421368421052633,
 'data_time': 4.806794834136963,
 'loss': 0.41893109679222107,
 'loss_kpt': 0.41893109679222107,
 'acc_pose': 0.1130952380952381,
 'time': 6.185840511322022,
 'epoch': 1,
 'memory': 1601,
 'step': 5}
```

In [7]:

```
df_train = pd.DataFrame()
df_test = pd.DataFrame()
for each in tqdm(json_list):
    if 'coco/AP' in each:
        df_test = df_test.append(eval(each), ignore_index=True)
    else:
        df_train = df_train.append(eval(each), ignore_index=True)
```

```
  0%|
| 0/1830 [00:00<?, ?it/s]C:\Users\leaf8\AppData\Local\Temp\ipykernel_13240\14
20932411.py:7: FutureWarning: The frame.append method is deprecated and will
be removed from pandas in a future version. Use pandas.concat instead.
  df_train = df_train.append(eval(each), ignore_index=True)
C:\Users\leaf8\AppData\Local\Temp\ipykernel_13240\1420932411.py:7: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  df_train = df_train.append(eval(each), ignore_index=True)
C:\Users\leaf8\AppData\Local\Temp\ipykernel_13240\1420932411.py:7: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  df_train = df_train.append(eval(each), ignore_index=True)
C:\Users\leaf8\AppData\Local\Temp\ipykernel_13240\1420932411.py:7: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
a future version. Use pandas.concat instead.
  df_train = df_train.append(eval(each), ignore_index=True)
C:\Users\leaf8\AppData\Local\Temp\ipykernel_13240\1420932411.py:7: FutureWarn
ing: The frame.append method is deprecated and will be removed from pandas in
```

In [8]:

```
df_train
```

Out[8]:

	lr	data_time	loss	loss_kpt	acc_pose	time	epoch	memory	st
0	4.000000e-08	7.957206	0.422016	0.422016	0.010417	13.061284	1.0	1532.0	1
1	2.105642e-04	4.301051	0.422321	0.422321	0.004464	6.952642	1.0	1601.0	2
2	4.210884e-04	4.643302	0.422624	0.422624	0.007440	6.609260	1.0	1601.0	3
3	6.316126e-04	4.796461	0.421855	0.421855	0.028274	6.407934	1.0	1601.0	4
4	8.421368e-04	4.806795	0.418931	0.418931	0.113095	6.185841	1.0	1601.0	5
...	...	...	...	...	...	...	...	...	
1795	2.002894e-04	4.080988	0.036799	0.036799	0.998512	4.496029	300.0	1601.0	1796
1796	2.001852e-04	4.066762	0.036885	0.036885	0.998512	4.481530	300.0	1601.0	1797
1797	2.001042e-04	4.069154	0.037018	0.037018	0.994048	4.484229	300.0	1601.0	1798
1798	2.000463e-04	4.068344	0.037003	0.037003	1.000000	4.483686	300.0	1601.0	1799
1799	2.000116e-04	3.999151	0.036328	0.036328	1.000000	4.406959	300.0	568.0	1800

1800 rows × 9 columns

In [9]:

```
df_test
```

Out[9]:

	coco/AP	coco/AP .5	coco/AP .75	coco/AP (M)	coco/AP (L)	coco/AR	coco/AR .5	coco/AR .75	coco/AR (M)
0	0.000000	0.000000	0.000000	-1.0	0.000000	0.000000	0.000000	0.000000	-1.0
1	0.001782	0.005941	0.000000	-1.0	0.001782	0.007143	0.023810	0.000000	-1.0
2	0.000000	0.000000	0.000000	-1.0	0.000000	0.000000	0.000000	0.000000	-1.0
3	0.079205	0.376148	0.000000	-1.0	0.079205	0.135714	0.547619	0.000000	-1.0
4	0.000000	0.000000	0.000000	-1.0	0.000000	0.000000	0.000000	0.000000	-1.0
5	0.245761	0.925860	0.002700	-1.0	0.245761	0.300000	0.952381	0.023810	-1.0
6	0.316543	0.898760	0.071521	-1.0	0.316543	0.361905	0.904762	0.166667	-1.0
7	0.485500	1.000000	0.307443	-1.0	0.485500	0.519048	1.000000	0.428571	-1.0
8	0.469171	1.000000	0.285624	-1.0	0.469171	0.511905	1.000000	0.428571	-1.0
9	0.539317	1.000000	0.554833	-1.0	0.539317	0.573810	1.000000	0.666667	-1.0
10	0.533655	1.000000	0.541970	-1.0	0.533655	0.569048	1.000000	0.619048	-1.0
11	0.569373	1.000000	0.643258	-1.0	0.569373	0.607143	1.000000	0.738095	-1.0
12	0.010451	0.084708	0.000000	-1.0	0.010451	0.019048	0.142857	0.000000	-1.0
13	0.528553	1.000000	0.415899	-1.0	0.528553	0.569048	1.000000	0.547619	-1.0
14	0.553844	1.000000	0.478268	-1.0	0.553844	0.588095	1.000000	0.571429	-1.0
15	0.594174	1.000000	0.704250	-1.0	0.594174	0.626190	1.000000	0.761905	-1.0
16	0.635103	1.000000	0.783028	-1.0	0.635103	0.661905	1.000000	0.809524	-1.0
17	0.565623	1.000000	0.602407	-1.0	0.565623	0.604762	1.000000	0.690476	-1.0
18	0.661626	1.000000	0.869740	-1.0	0.661626	0.680952	1.000000	0.880952	-1.0
19	0.649978	1.000000	0.780500	-1.0	0.649978	0.695238	1.000000	0.833333	-1.0
20	0.690555	1.000000	0.919059	-1.0	0.690555	0.711905	1.000000	0.928571	-1.0
21	0.728529	1.000000	0.942912	-1.0	0.728529	0.757143	1.000000	0.952381	-1.0
22	0.721614	1.000000	0.946413	-1.0	0.721614	0.750000	1.000000	0.952381	-1.0
23	0.736554	1.000000	0.947195	-1.0	0.736554	0.764286	1.000000	0.952381	-1.0
24	0.699394	1.000000	0.940987	-1.0	0.699394	0.738095	1.000000	0.952381	-1.0
25	0.741375	1.000000	0.969118	-1.0	0.741375	0.773810	1.000000	0.976190	-1.0
26	0.736882	1.000000	0.968647	-1.0	0.736882	0.776190	1.000000	0.976190	-1.0
27	0.738619	1.000000	0.946868	-1.0	0.738619	0.773810	1.000000	0.952381	-1.0
28	0.745215	1.000000	0.946110	-1.0	0.745215	0.778571	1.000000	0.952381	-1.0
29	0.733089	1.000000	0.945599	-1.0	0.733089	0.776190	1.000000	0.952381	-1.0

导出训练日志表格

In [10]:

```
df_train.to_csv('训练日志-训练集.csv', index=False)
df_test.to_csv('训练日志-测试集.csv', index=False)
```

设置Matplotlib中文字体

In [11]:

```
# # windows操作系统
# plt.rcParams['font.sans-serif']=['SimHei']  # 用来正常显示中文标签
# plt.rcParams['axes.unicode_minus']=False  # 用来正常显示负号
```

In [12]:

```
# Mac操作系统，参考 https://www.ngui.cc/51cto/show-727683.html
# 下载 simhei.ttf 字体文件
# !wget https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/20220716-mmclassification/datase
```

In [13]:

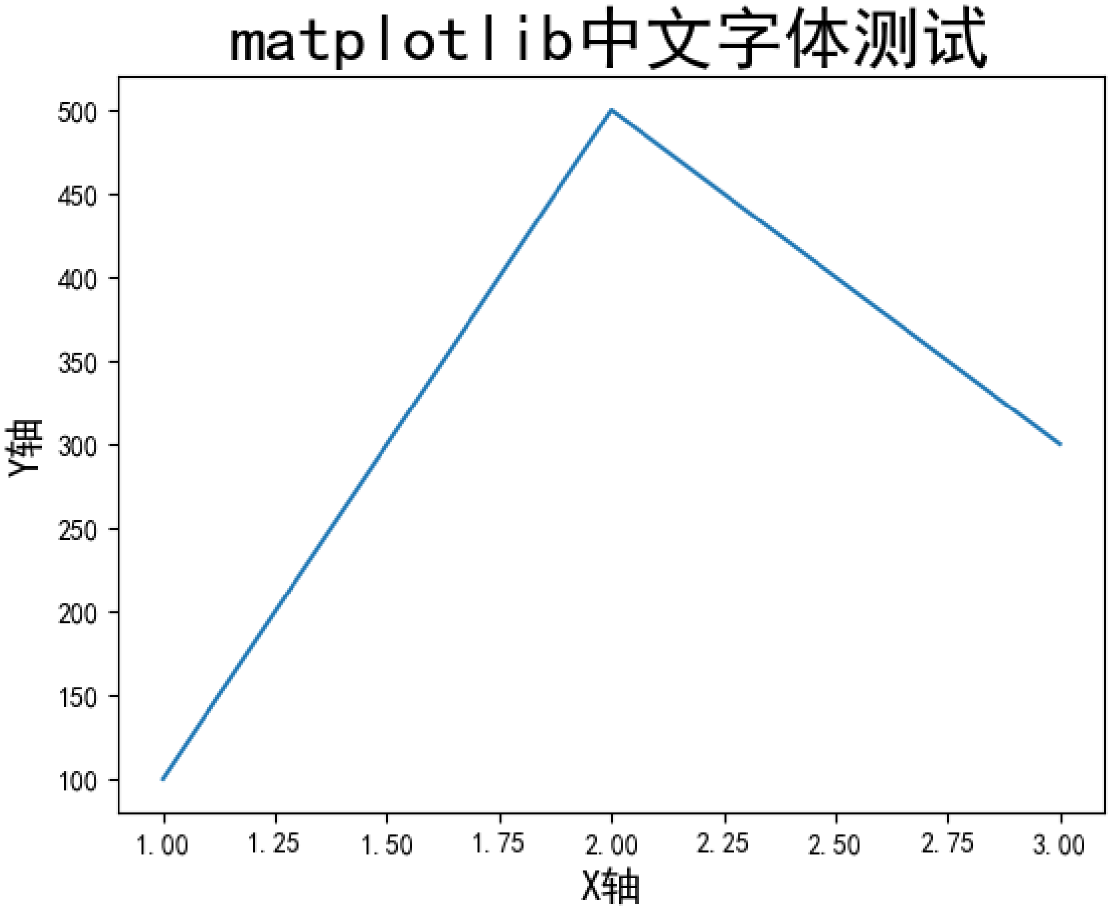
```
# Linux操作系统，例如 云GPU平台：https://featurize.cn/?s=d7ce99f842414bfcaea5662a97581bd1
# 如果遇到 SSL 相关报错，重新运行本代码块即可
!wget https://zihao-openmmlab.obs.cn-east-3.myhuaweicloud.com/20220716-mmclassification/dataset
!rm -rf /home/featurize/.cache/matplotlib

import matplotlib
import matplotlib.pyplot as plt
matplotlib.rc("font",family='SimHei') # 中文字体
```

```
/environment/miniconda3/lib/python3.7/site-packages/matplotlib/mpl-data/fonts/tt
f/SimHei.ttf: No such file or directory
'rm' 不是内部或外部命令，也不是可运行的程序
或批处理文件。
```

In [14]:

```
plt.plot([1,2,3], [100,500,300])
plt.title('matplotlib中文字体测试', fontsize=25)
plt.xlabel('X轴', fontsize=15)
plt.ylabel('Y轴', fontsize=15)
plt.show()
```



可视化辅助函数

In [15]:

```
from matplotlib import colors as mcolors
import random
random.seed(124)
colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'tab:blue', 'tab:orange', 'tab:green', 'tab:red', '
markers = [".",",","o","v","^","<",">","1","2","3","4","8","s","p","P","*","h","H","+","x","X","
linestyle = ['--', '-.', '-']

def get_line_arg():
    '''
    随机产生一种绘图线型
    '''
    line_arg = {}
    line_arg['color'] = random.choice(colors)
    # line_arg['marker'] = random.choice(markers)
    line_arg['linestyle'] = random.choice(linestyle)
    line_arg['linewidth'] = random.randint(1, 4)
    # line_arg['markersize'] = random.randint(3, 5)
    return line_arg
```

训练集损失函数

In [16]:

```
df_train.columns
```

Out[16]:

```
Index(['lr', 'data_time', 'loss', 'loss_kpt', 'acc_pose', 'time', 'epoch',
       'memory', 'step'],
      dtype='object')
```

In [17]:

```
metrics = ['loss', 'loss_kpt']
```

In [18]:

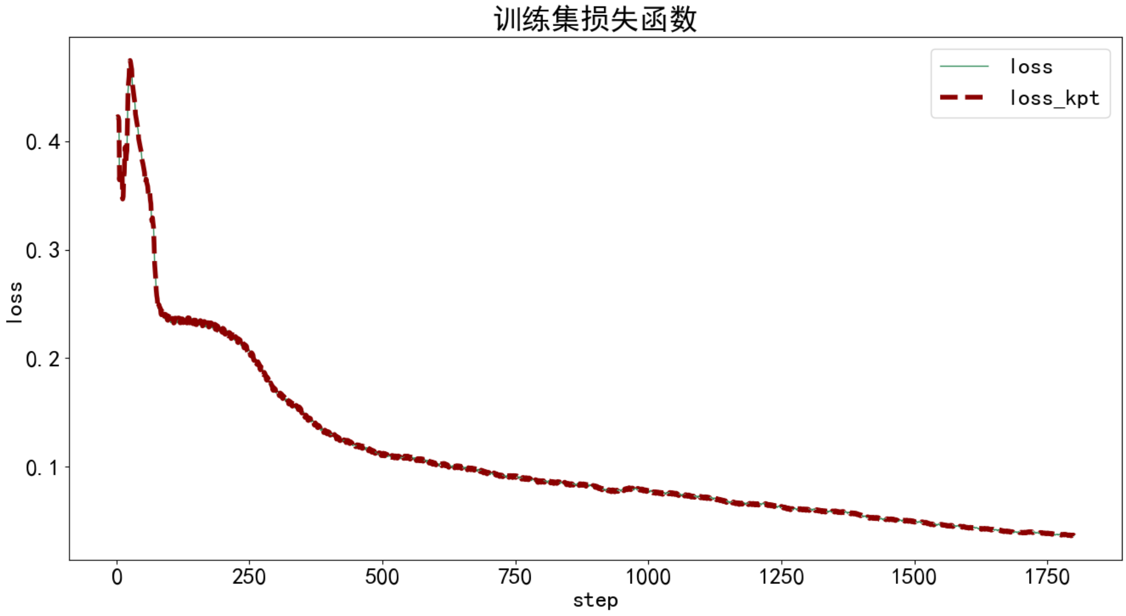
```
plt.figure(figsize=(16, 8))

x = df_train['step']
for y in metrics:
    plt.plot(x, df_train[y], label=y, **get_line_arg())

plt.tick_params(labelsize=20)
plt.xlabel('step', fontsize=20)
plt.ylabel('loss', fontsize=20)
plt.title('训练集损失函数', fontsize=25)
plt.savefig('训练集损失函数.pdf', dpi=120, bbox_inches='tight')

plt.legend(fontsize=20)

plt.show()
```



训练集准确率

In [19]:

```
metrics = ['acc_pose']
```

In [20]:

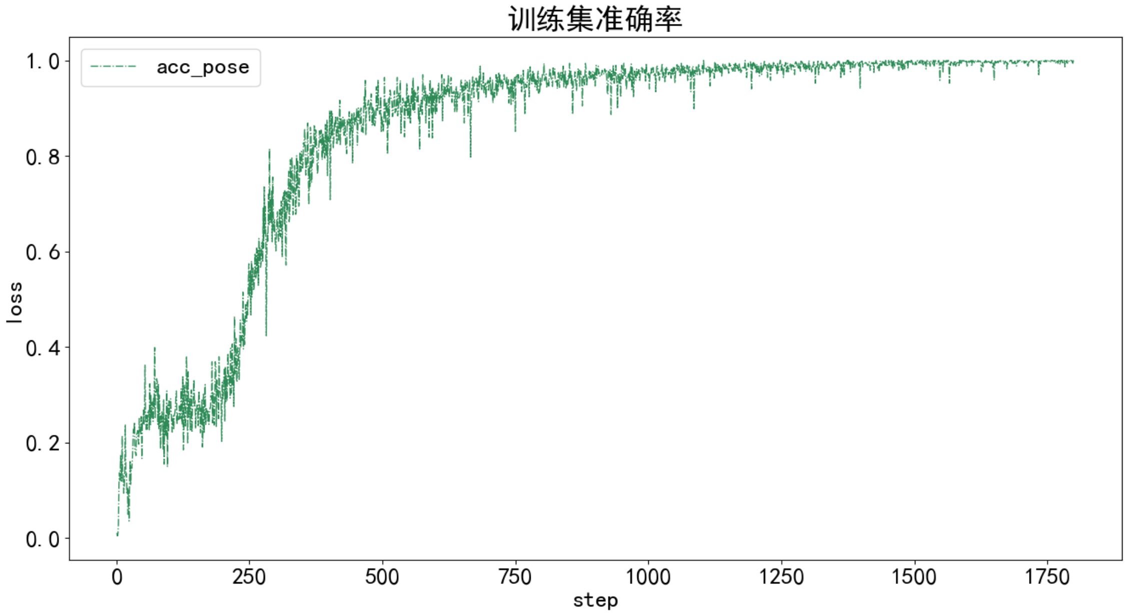
```
plt.figure(figsize=(16, 8))

x = df_train['step']
for y in metrics:
    plt.plot(x, df_train[y], label=y, **get_line_arg())

plt.tick_params(labelsize=20)
plt.xlabel('step', fontsize=20)
plt.ylabel('loss', fontsize=20)
plt.title('训练集准确率', fontsize=25)
plt.savefig('训练集准确率.pdf', dpi=120, bbox_inches='tight')

plt.legend(fontsize=20)

plt.show()
```



测试集评估指标-MS COCO Metric

In [21]:

```
df_test.columns
```

Out[21]:

```
Index(['coco/AP', 'coco/AP .5', 'coco/AP .75', 'coco/AP (M)', 'coco/AP (L)',
       'coco/AR', 'coco/AR .5', 'coco/AR .75', 'coco/AR (M)', 'coco/AR (L)',
       'PCK', 'AUC', 'NME', 'data_time', 'time', 'step'],
      dtype='object')
```

In [22]:

```
metrics = ['coco/AP', 'coco/AP .5', 'coco/AP .75', 'coco/AP (M)', 'coco/AP (L)', 'coco/AR', 'coc
```

In [23]:

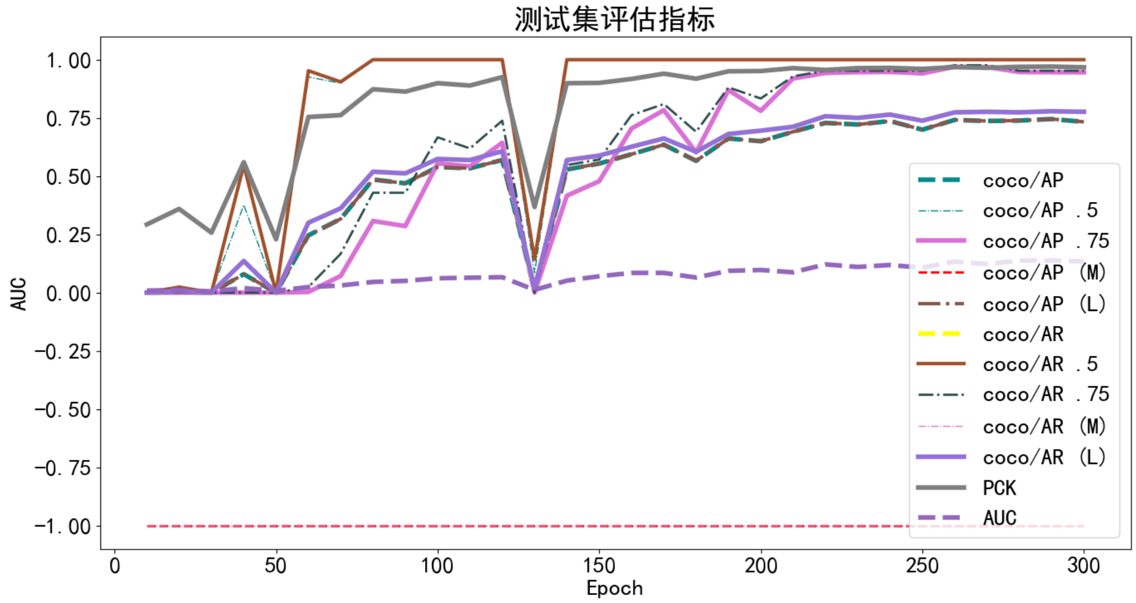
```
plt.figure(figsize=(16, 8))

x = df_test['step']
for y in metrics:
    plt.plot(x, df_test[y], label=y, **get_line_arg())

plt.tick_params(labelsize=20)
# plt.ylim([0, 100])
plt.xlabel('Epoch', fontsize=20)
plt.ylabel(y, fontsize=20)
plt.title('测试集评估指标', fontsize=25)
plt.savefig('测试集分类评估指标.pdf', dpi=120, bbox_inches='tight')

plt.legend(fontsize=20)

plt.show()
```



测试集评估指标-NME

In [24]:

```
metrics = ['NME']
```

In [25]:

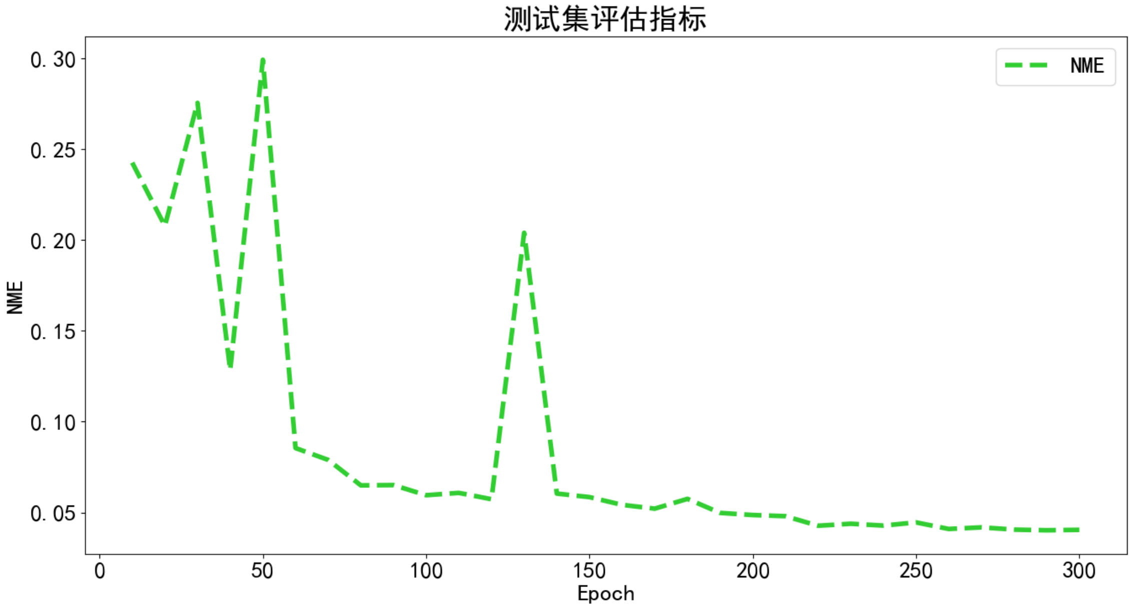
```
plt.figure(figsize=(16, 8))

x = df_test['step']
for y in metrics:
    plt.plot(x, df_test[y], label=y, **get_line_arg())

plt.tick_params(labelsize=20)
# plt.ylim([0, 100])
plt.xlabel('Epoch', fontsize=20)
plt.ylabel(y, fontsize=20)
plt.title('测试集评估指标', fontsize=25)
plt.savefig('测试集分类评估指标.pdf', dpi=120, bbox_inches='tight')

plt.legend(fontsize=20)

plt.show()
```



In [ ]:

In [ ]:

In [ ]: