HW4_Changhao Xu

Tuesday, February 28, 2023 17:50

1. **Environment setup**

2. **Tensorized Fourier neural operator**

1) **Use tfno darcy config.yaml to train TFNO.**

number of parameters of TFNO are n_params: 137473, which is much smaller than that of FNO.

```
model = get_model(config)
model = model.to(device)

#Log parameter count
if is_logger:
    n_params = count_params(model)

    if config.verbose:
        print(f'\nn_params: {n_params}')
        sys.stdout.flush()
```

```
Given argument key='dropout' that is not in TFNO2d's signature.
Given argument key='tensor_lasso_penalty' that is not in TFNO2d's signature.
Keyword argument out_channels not specified for model TFNO2d, using default=1.
Keyword argument non_linearity not specified for model TFNO2d, using default=<built-in function gelu>.
Keyword argument decomposition_kwargs not specified for model TFNO2d, using default={}.

n_params: 137473
```

```
[115] time=2.11, avg_loss=1.3150, train_err=0.0657, 32_h1=0.0787, 32_l2=0.0431, 64_h1=0.1686, 64_l2=0.0588
[116] time=2.12, avg_loss=1.3268, train_err=0.0663, 32_h1=0.0716, 32_l2=0.0340, 64_h1=0.1601, 64_l2=0.0518
[117] time=2.12, avg_loss=1.3051, train_err=0.0653, 32_h1=0.0722, 32_l2=0.0350, 64_h1=0.1560, 64_l2=0.0487
[118] time=2.29, avg_loss=1.3331, train_err=0.0667, 32_h1=0.0748, 32_l2=0.0387, 64_h1=0.1660, 64_l2=0.0543
[119] time=2.24, avg_loss=1.3175, train_err=0.0659, 32_h1=0.0751, 32_l2=0.0393, 64_h1=0.1573, 64_l2=0.0518
[120] time=2.34, avg_loss=1.3322, train_err=0.0666, 32_h1=0.0732, 32_l2=0.0377, 64_h1=0.1576, 64_l2=0.0504
[121] time=2.22, avg_loss=1.3273, train_err=0.0664, 32_h1=0.0821, 32_l2=0.0535, 64_h1=0.1708, 64_l2=0.0664
[122] time=2.27, avg_loss=1.3329, train_err=0.0666, 32_h1=0.0746, 32_l2=0.0383, 64_h1=0.1621, 64_l2=0.0555
[123] time=2.25, avg_loss=1.3202, train_err=0.0660, 32_h1=0.0732, 32_l2=0.0362, 64_h1=0.1646, 64_l2=0.0524
[124] time=2.24, avg_loss=1.3057, train_err=0.0653, 32_h1=0.0719, 32_l2=0.0342, 64_h1=0.1580, 64_l2=0.0505
[125] time=2.33, avg_loss=1.2870, train_err=0.0643, 32_h1=0.0741, 32_l2=0.0378, 64_h1=0.1518, 64_l2=0.0485
[126] time=2.18, avg_loss=1.2895, train_err=0.0645, 32_h1=0.0719, 32_l2=0.0345, 64_h1=0.1598, 64_l2=0.0509
[127] time=2.32, avg_loss=1.3289, train_err=0.0664, 32_h1=0.0714, 32_l2=0.0337, 64_h1=0.1595, 64_l2=0.0492
[128] time=2.24, avg_loss=1.3020, train_err=0.0651, 32_h1=0.0727, 32_l2=0.0367, 64_h1=0.1577, 64_l2=0.0564
[129] time=2.27, avg_loss=1.2874, train_err=0.0644, 32_h1=0.0737, 32_l2=0.0364, 64_h1=0.1522, 64_l2=0.0504
[130] time=2.20, avg_loss=1.2890, train_err=0.0645, 32_h1=0.0722, 32_l2=0.0357, 64_h1=0.1601, 64_l2=0.0499
[131] time=2.13, avg_loss=1.3140, train_err=0.0657, 32_h1=0.0726, 32_l2=0.0352, 64_h1=0.1549, 64_l2=0.0457
[132] time=2.26, avg_loss=1.3198, train_err=0.0660, 32_h1=0.0720, 32_l2=0.0343, 64_h1=0.1598, 64_l2=0.0525
[133] time=2.25, avg_loss=1.2848, train_err=0.0642, 32_h1=0.0718, 32_l2=0.0340, 64_h1=0.1613, 64_l2=0.0502
[134] time=2.24, avg_loss=1.2786, train_err=0.0639, 32_h1=0.0715, 32_l2=0.0342, 64_h1=0.1572, 64_l2=0.0488
[135] time=2.16, avg_loss=1.2808, train_err=0.0640, 32_h1=0.0726, 32_l2=0.0359, 64_h1=0.1624, 64_l2=0.0543
[136] time=2.20, avg_loss=1.2777, train_err=0.0639, 32_h1=0.0742, 32_l2=0.0386, 64_h1=0.1571, 64_l2=0.0560
[137] time=2.47, avg_loss=1.2626, train_err=0.0631, 32_h1=0.0727, 32_l2=0.0366, 64_h1=0.1625, 64_l2=0.0564
[138] time=2.43, avg_loss=1.2684, train_err=0.0634, 32_h1=0.0724, 32_l2=0.0355, 64_h1=0.1595, 64_l2=0.0497
[139] time=2.28, avg_loss=1.2598, train_err=0.0630, 32_h1=0.0740, 32_l2=0.0399, 64_h1=0.1550, 64_l2=0.0554
[140] time=2.24, avg_loss=1.2740, train_err=0.0637, 32_h1=0.0728, 32_l2=0.0350, 64_h1=0.1625, 64_l2=0.0522
[141] time=2.42, avg_loss=1.2714, train_err=0.0636, 32_h1=0.0728, 32_l2=0.0358, 64_h1=0.1631, 64_l2=0.0538
[142] time=2.44, avg_loss=1.2842, train_err=0.0642, 32_h1=0.0728, 32_l2=0.0351, 64_h1=0.1566, 64_l2=0.0523
[143] time=2.45, avg_loss=1.2614, train_err=0.0631, 32_h1=0.0706, 32_l2=0.0332, 64_h1=0.1601, 64_l2=0.0508
[144] time=2.47, avg_loss=1.2510, train_err=0.0625, 32_h1=0.0722, 32_l2=0.0357, 64_h1=0.1582, 64_l2=0.0504
[145] time=2.49, avg_loss=1.2599, train_err=0.0630, 32_h1=0.0720, 32_l2=0.0345, 64_h1=0.1593, 64_l2=0.0525
[146] time=2.45, avg_loss=1.2488, train_err=0.0624, 32_h1=0.0709, 32_l2=0.0332, 64_h1=0.1592, 64_l2=0.0508
[147] time=2.44, avg_loss=1.2409, train_err=0.0620, 32_h1=0.0711, 32_l2=0.0346, 64_h1=0.1657, 64_l2=0.0528
[148] time=2.42, avg_loss=1.2526, train_err=0.0626, 32_h1=0.0736, 32_l2=0.0373, 64_h1=0.1490, 64_l2=0.0480
[149] time=2.40, avg_loss=1.2652, train_err=0.0633, 32_h1=0.0724, 32_l2=0.0359, 64_h1=0.1545, 64_l2=0.0497
```

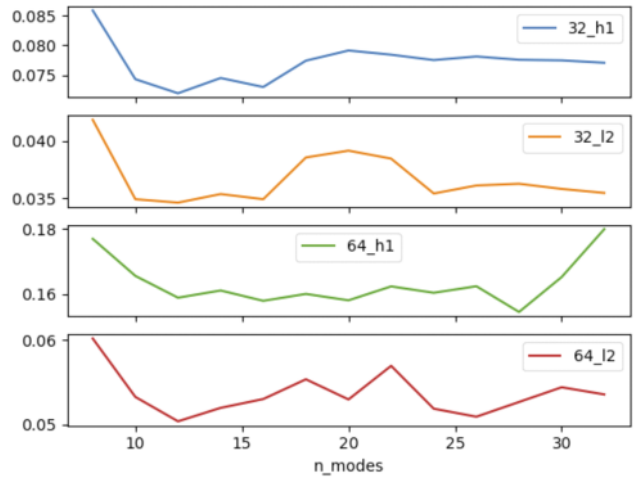
2) **Vary the number of modes of TFNO from 8 to 32,**
Plot the test L2 and H1 error on resolutions 32 and 64 change versus number of modes.

In order to reduce randomness, each parameter was run twice and the error was averaged.

	n_modes	32_h1	32_l2	64_h1	64_l2
0	8.0	0.08585	0.04185	0.17690	0.06020
1	10.0	0.07435	0.03490	0.16560	0.05325
2	12.0	0.07200	0.03460	0.15895	0.05035
3	14.0	0.07455	0.03535	0.16115	0.05195
4	16.0	0.07305	0.03490	0.15800	0.05300
5	18.0	0.07745	0.03855	0.16010	0.05535
6	20.0	0.07915	0.03915	0.15815	0.05295
7	22.0	0.07845	0.03845	0.16240	0.05695
8	24.0	0.07755	0.03540	0.16045	0.05185
9	26.0	0.07815	0.03610	0.16245	0.05090
10	28.0	0.07760	0.03625	0.15460	0.05265
11	30.0	0.07750	0.03580	0.16525	0.05440
12	32.0	0.07710	0.03545	0.17985	0.05355

```
df_1.plot.line(x='n_modes', subplots=True)
```

```
array([<AxesSubplot:xlabel='n_modes'>, <AxesSubplot:xlabel='n_modes'>,
       <AxesSubplot:xlabel='n_modes'>, <AxesSubplot:xlabel='n_modes'>],
      dtype=object)
```



From the results, we select n_modes_height = n_modes_width = 12 in terms of overall error.

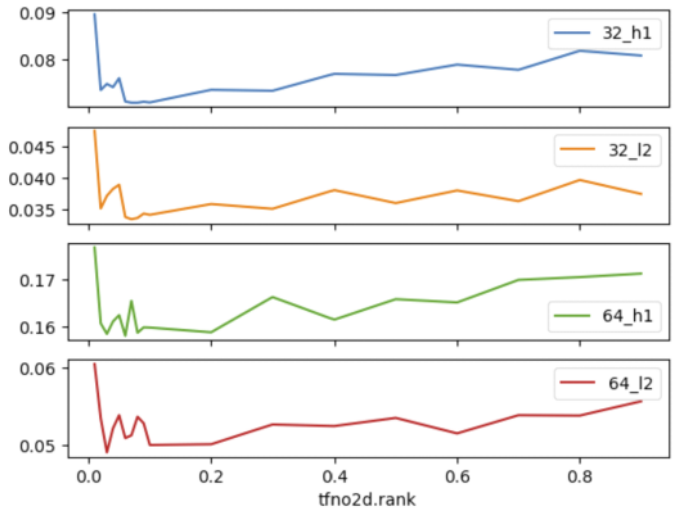
3) Use n_modes_height = n_modes_width = 12, vary the rank of tensor factorization from 0.01 to 0.9.
Plot the test L2 and H1 error on resolutions 32 and 64 change versus the number of paramters of TFNO.

In order to reduce randomness, each parameter was run twice and the error was averaged.

	tfno2d.rank	n_params	32_h1	32_l2	64_h1	64_l2
0	0.01	19537.0	0.08955	0.04745	0.17680	0.06045
1	0.02	26497.0	0.07365	0.03520	0.16065	0.05345
2	0.03	30849.0	0.07495	0.03720	0.15840	0.04905
3	0.04	35713.0	0.07425	0.03830	0.16105	0.05215
4	0.05	38337.0	0.07615	0.03895	0.16240	0.05385
5	0.06	56961.0	0.07125	0.03385	0.15800	0.05090
6	0.07	62161.0	0.07100	0.03350	0.16545	0.05125
7	0.08	62161.0	0.07100	0.03370	0.15865	0.05365
8	0.09	67649.0	0.07120	0.03440	0.15980	0.05280
9	0.10	67649.0	0.07105	0.03420	0.15975	0.05000
10	0.20	137473.0	0.07370	0.03590	0.15875	0.05010
11	0.30	173569.0	0.07350	0.03515	0.16625	0.05265
12	0.40	290385.0	0.07705	0.03810	0.16145	0.05245
13	0.50	311809.0	0.07680	0.03605	0.16580	0.05350
14	0.60	357057.0	0.07900	0.03805	0.16510	0.05150
15	0.70	380881.0	0.07790	0.03635	0.16990	0.05385
16	0.80	564097.0	0.08190	0.03970	0.17050	0.05380
17	0.90	600257.0	0.08090	0.03750	0.17125	0.05565

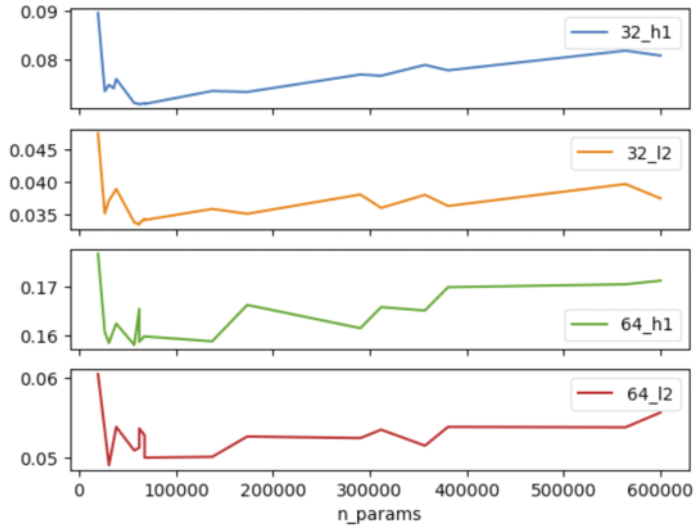
```
df_2.plot.line(x='tfno2d.rank', subplots=True)
```

```
array([<AxesSubplot:xlabel='tfno2d.rank'>,
       <AxesSubplot:xlabel='tfno2d.rank'>,
       <AxesSubplot:xlabel='tfno2d.rank'>,
       <AxesSubplot:xlabel='tfno2d.rank'>], dtype=object)
```




```
df_2.drop(columns=['tfno2d.rank']).plot.line(x='n_params', subplots=True)
```

```
array([<AxesSubplot:xlabel='n_params'>, <AxesSubplot:xlabel='n_params'>,
       <AxesSubplot:xlabel='n_params'>, <AxesSubplot:xlabel='n_params'>],
      dtype=object)
```



From the results, both rank=0.03 (n_params: 30849) and 0.1 (n_params: 67649)could yield small errors.