Rewrite

Take out **Tensorflow** from the code. And use mostly torch to tune the network. Using parallel computing to implement PIAL1.

Reasons: 1. multiple backend makes the code low in efficiency. 2. Easier to modifier later. 3. Using for loop for PIAL1 is low in efficiency as well.

Results: the training time decreases from 7 hours to no more than 1 hour (The device I used, Tesla P100, may contribute a lot)

Repeat the result on Overleaf

I’ve figured out some important technics that would make a difference of accuracy (L2 relative error)

1. **Activation,** if we use **Relu** for the **branch network**, the L2 relative error is larger. The reason is not the depth of network.

Activation	**L2 relative error** (one test only)
Relu	1.60e-2
Gelu	4.33e-3

2. **Dataset length scale,** the same as Min’s paper, using multiple LS could have better accuracy.

The result is sometime better than the article

Experience

Some mentionable mistakes I made that could cause training failure.

1. Data sampling Replacement. If we sample the data randomly and they might repeat within a single loop.

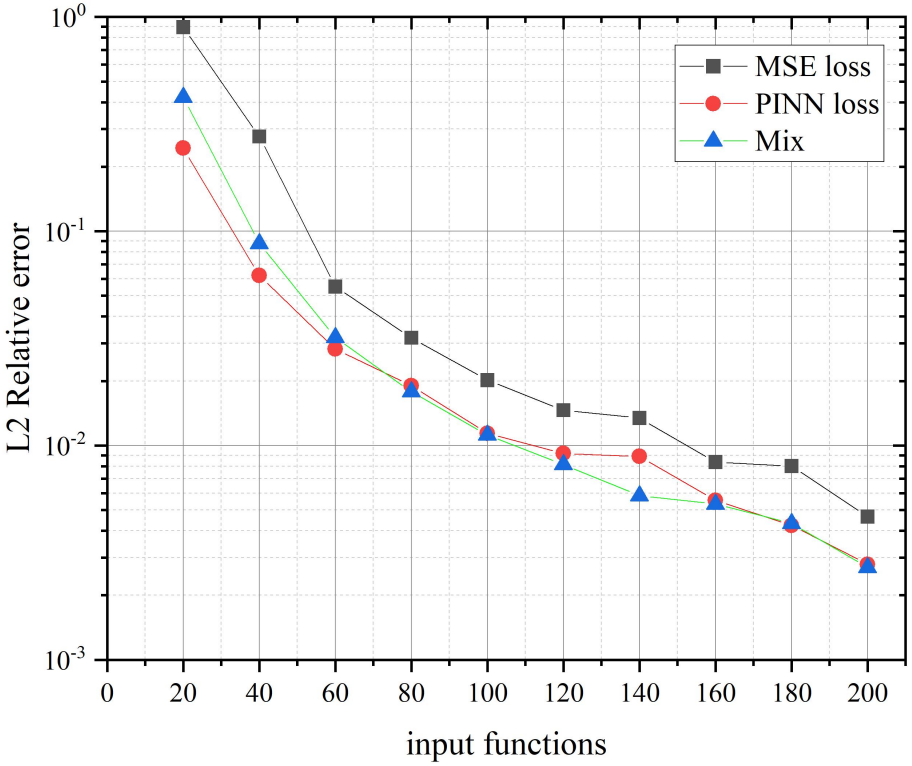
e.g. a = [1, 2, 3], step = 3. Without repeat: 3, 1, 2; With repeat: 3, 1, 3.

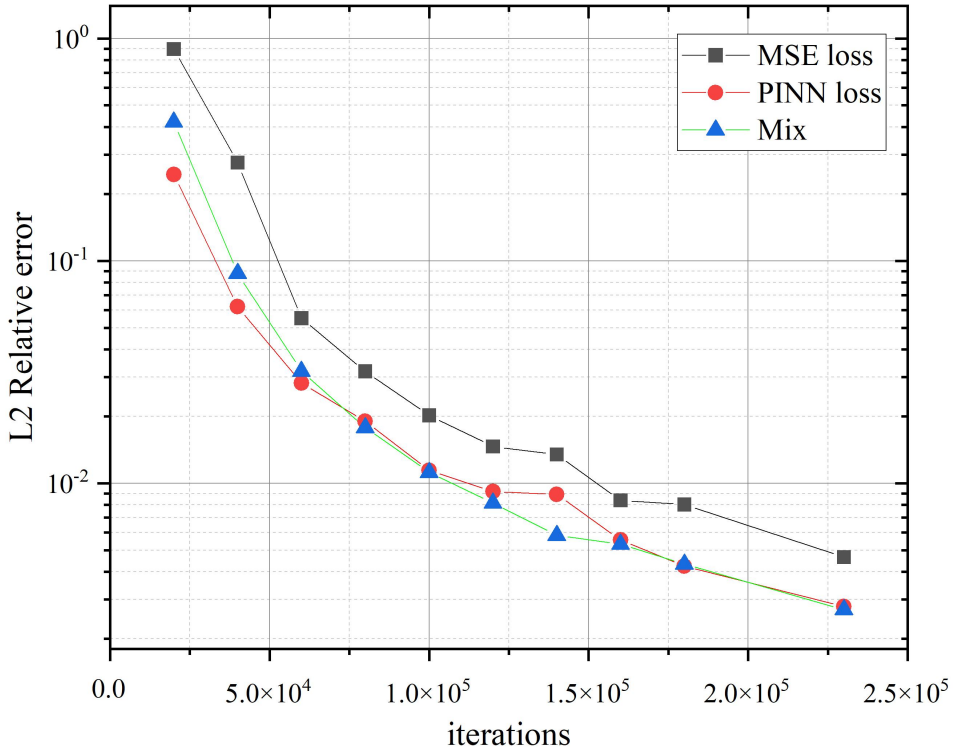
The accuracy would be worse.

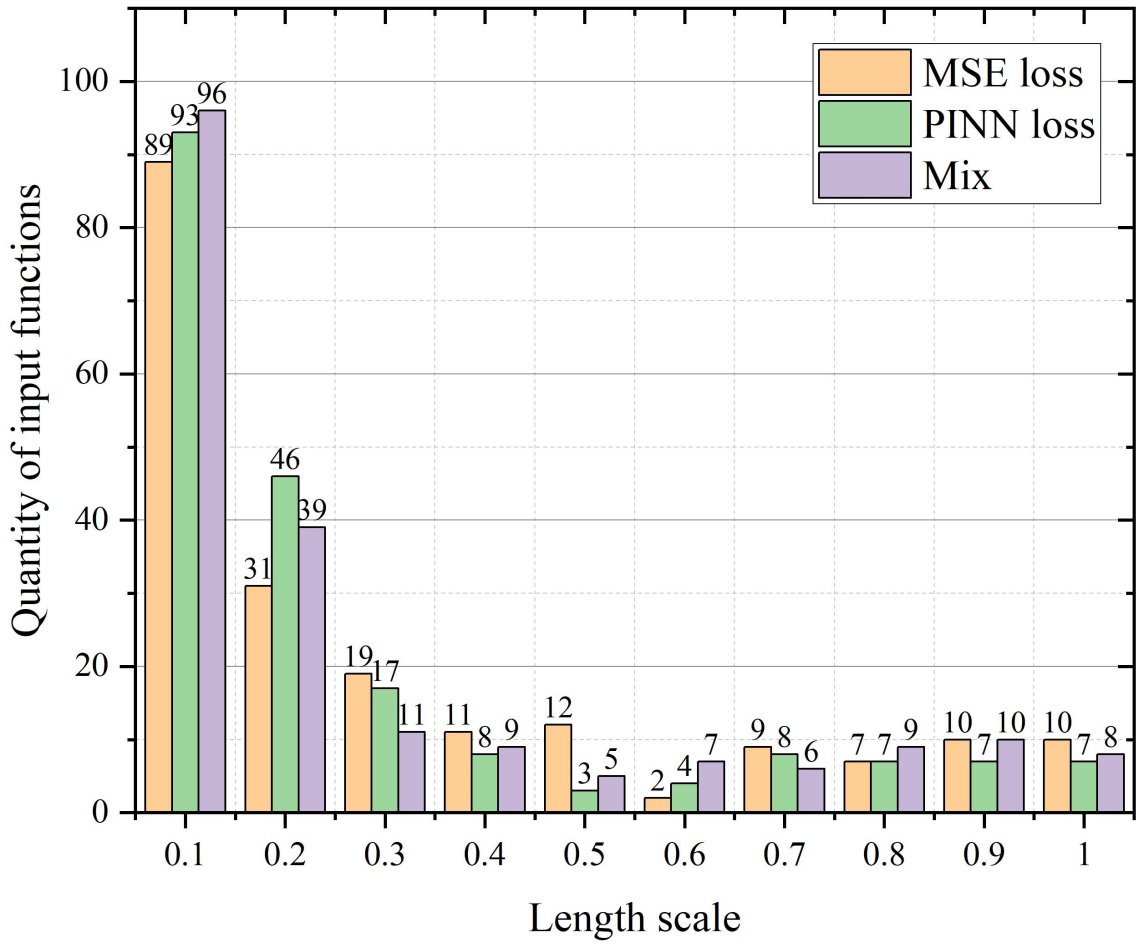
2. The result of using all data (1,000 input functions, LS = [0.1, 1]) is worse than PIAL1. One possible reason maybe the quality of data increases)

More tries

- I also tested the code using **(1.)** MSE between pd and gt **(2.)** 0.1 times PINN loss only (because the PINN loss is much larger than the MSE) **(3.)** 0.05 times PINN loss + MSE loss
- Their results are slightly different. One possible reason is the gradient norm after adding PINN loss is larger. So the parameters could be updated more efficiently.


MSE loss
PINN loss
Mix
L2 Relative error
input functions
10^0
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
120
140
160
180
200


MSE loss
PINN loss
Mix
L2 Relative error
iterations
10^0
10^{-1}
10^{-2}
0.0
5.0×10^4
1.0×10^5
1.5×10^5
2.0×10^5
2.5×10^5


MSE loss
PINN loss
Mix
Quantity of input functions
Length scale
0
20
40
60
80
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
89
93
96
31
46
39
19
17
11
11
8
9
12
3
5
2
4
7
9
8
6
7
7
9
10
7
10
10
7
8