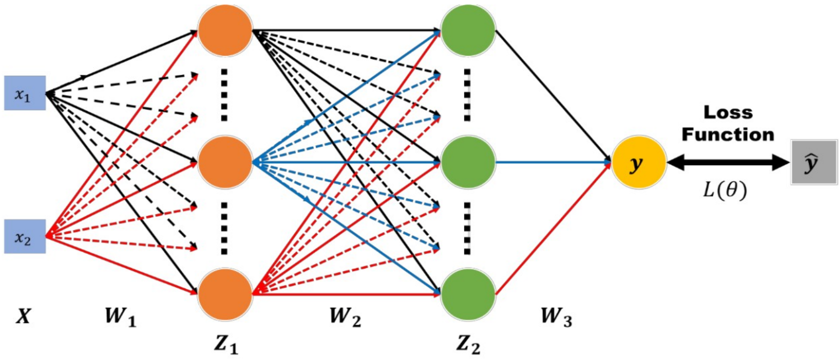
Lab 1 Report

309551064 張凱翔

1. Introduction

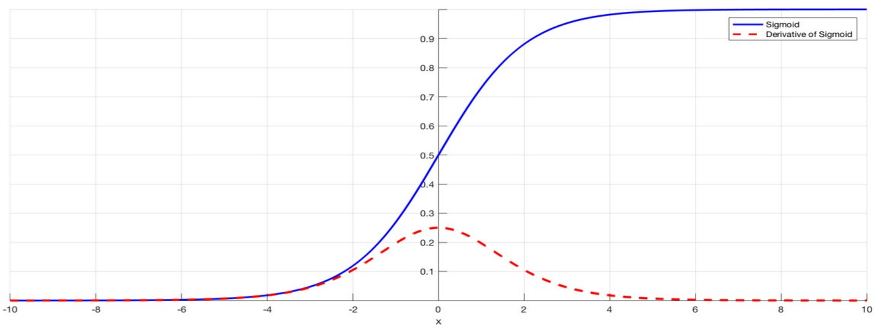
用兩層 hidden layer 來做兩種點分類問題:Linear 和 XOR, 並使用 numpy 實做 backpropagation 來更新 weight。Model 的架構如下圖表示。



2. Experiment setups:

A. Sigmoid functions

使用 lab1 spec 上提供的 sigmoid function



B. Neural network

Input: 2 dimension
Hidden Layer: 10 dimension, 2 layer
Output: 1 dimension
Loss Function: MSE loss
Learning Rate: 0.1
Epochs: 5000

C. Backpropagation

X-> W1-> z1-> a1-> W2-> z2-> a2-> W3-> z3-> a3==y

上面是 `forwarding pass` 的順序，`backpropagation` 則是從最後一層往前計算。

從最後一層的 weight 開始更新，使用 chain rule 來更加快速的計算 gradient。我的作法是以 neuron 為單位計算這個 neuron 所有 weight 的 gradient，再將所有 neuron 的 gradient concatenate 起來，最後將原始的 weight 減去 gradient * LR。

下圖是第二層 weight 的更新方式:

```
# net[1]
C_W1 = np.zeros((len(net[1]), 1))
C_z1 = np.zeros((1, 1))
for i in range(len(net[2])):
    z2_a1 = np.array([net[2][i]]).reshape(1, 1)
    a1_z1 = np.array([derivative_sigmoid(outputs[1]).reshape(-1)[i]]).reshape(1, 1)
    z1_W1 = outputs[0].T
    C_z1 = np.concatenate((C_z1, a1_z1 @ z2_a1 @ a2_z2 @ C_a2), axis=1)
    C_W1 = np.concatenate((C_W1, z1_W1 @ a1_z1 @ z2_a1 @ a2_z2.T @ C_a2.T), axis=1)

C_W1 = C_W1[:, 1:]
C_z1 = C_z1[:, 1:]
```

```
update_net[1] -= LR * C_W1
```

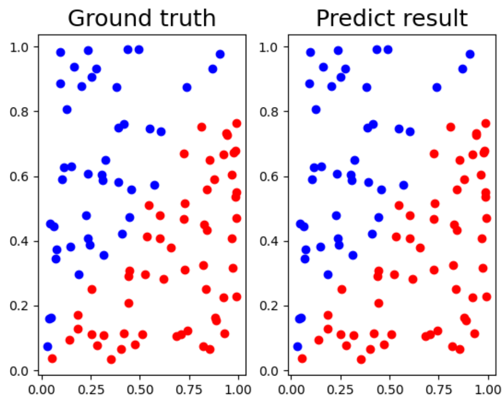
3. Results of your testing:

A. Screenshot and comparison figure

Linear:

```
epoch 4965 loss : 7.613245045370869e-05
epoch 4966 loss : 7.611085365818966e-05
epoch 4967 loss : 7.608926784820474e-05
epoch 4968 loss : 7.606769301575001e-05
epoch 4969 loss : 7.604612915283412e-05
epoch 4970 loss : 7.602457625146534e-05
epoch 4971 loss : 7.600303430366536e-05
epoch 4972 loss : 7.598150330146136e-05
epoch 4973 loss : 7.595998323688652e-05
epoch 4974 loss : 7.593847410198571e-05
epoch 4975 loss : 7.591697588880799e-05
epoch 4976 loss : 7.589548858941076e-05
epoch 4977 loss : 7.587401219585764e-05
epoch 4978 loss : 7.58525467002233e-05
epoch 4979 loss : 7.583109209458542e-05
epoch 4980 loss : 7.580964837103284e-05
epoch 4981 loss : 7.578821552166044e-05
epoch 4982 loss : 7.576679353857069e-05
epoch 4983 loss : 7.574538241387134e-05
epoch 4984 loss : 7.572398213968114e-05
epoch 4985 loss : 7.570259270812427e-05
epoch 4986 loss : 7.568121411133182e-05
epoch 4987 loss : 7.565984634144554e-05
epoch 4988 loss : 7.563848939061002e-05
epoch 4989 loss : 7.561714325097993e-05
epoch 4990 loss : 7.559580791471733e-05
epoch 4991 loss : 7.55744833739907e-05
epoch 4992 loss : 7.555316962097672e-05
epoch 4993 loss : 7.553186664786024e-05
epoch 4994 loss : 7.551057444683044e-05
epoch 4995 loss : 7.548929301008571e-05
epoch 4996 loss : 7.546802232983217e-05
epoch 4997 loss : 7.544676239828428e-05
epoch 4998 loss : 7.542551320766003e-05
epoch 4999 loss : 7.540427475018815e-05
epoch 5000 loss : 7.538304701810515e-05
```

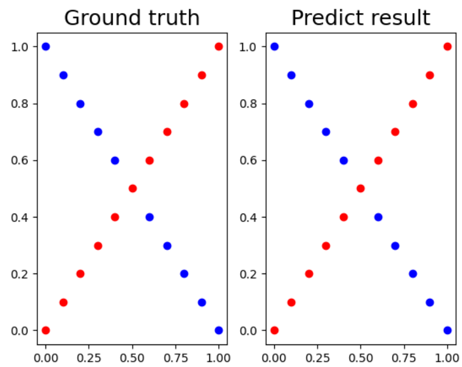
```
[9.99999996e-01]
[9.99999997e-01]
[9.99997746e-01]
[6.69926938e-07]
[9.55433928e-01]
[9.99999998e-01]
[2.13974245e-08]
[9.99999999e-01]
[9.99999997e-01]
[9.99999989e-01]
[6.04083437e-08]
[1.50130441e-08]
[9.99411030e-01]
[9.99999999e-01]
[9.99999999e-01]
[1.89400032e-07]
[9.99999992e-01]
[1.43664465e-02]
[1.30830686e-08]
[9.99999999e-01]
[3.03155567e-08]
[1.67078197e-08]
[1.97024131e-08]
[9.99999886e-01]
[9.99999997e-01]
[9.99999999e-01]
[2.53519907e-04]
[9.99999998e-01]
[1.24719480e-08]
[9.99999998e-01]
[9.99999998e-01]]
```



XOR:

```
epoch 4971 loss : 0.0002915166541196464
epoch 4972 loss : 0.00029139752523972735
epoch 4973 loss : 0.0002912784837854105
epoch 4974 loss : 0.0002911595296573643
epoch 4975 loss : 0.00029104066276406087
epoch 4976 loss : 0.0002909218830140973
epoch 4977 loss : 0.0002908031903161966
epoch 4978 loss : 0.0002906845845791977
epoch 4979 loss : 0.00029056606571207133
epoch 4980 loss : 0.00029044763362391255
epoch 4981 loss : 0.0002903292882239307
epoch 4982 loss : 0.000290211029421465
epoch 4983 loss : 0.00029009285712597966
epoch 4984 loss : 0.0002899747712470496
epoch 4985 loss : 0.00028985677169438825
epoch 4986 loss : 0.0002897388583778176
epoch 4987 loss : 0.0002896210312072875
epoch 4988 loss : 0.00028950329009287164
epoch 4989 loss : 0.0002893856349447613
epoch 4990 loss : 0.00028926806567326546
epoch 4991 loss : 0.0002891505821888248
epoch 4992 loss : 0.00028903318440199493
epoch 4993 loss : 0.00028891587222345
epoch 4994 loss : 0.000288798645563985
epoch 4995 loss : 0.00028868150433452287
epoch 4996 loss : 0.0002885644484460957
epoch 4997 loss : 0.00028844747780986627
epoch 4998 loss : 0.00028833059233710455
epoch 4999 loss : 0.00028821379193921446
epoch 5000 loss : 0.00028809707652770704
```

```
[[0.00875944]
 [0.9999543 ]
 [0.00759748]
 [0.99994671]
 [0.00898509]
 [0.99992732]
 [0.01747357]
 [0.99978932]
 [0.03198917]
 [0.96176353]
 [0.02883998]
 [0.01733844]
 [0.96008786]
 [0.01077311]
 [0.99968571]
 [0.00773847]
 [0.99984076]
 [0.0062416 ]
 [0.99981439]
 [0.00542407]
 [0.99973242]]
```



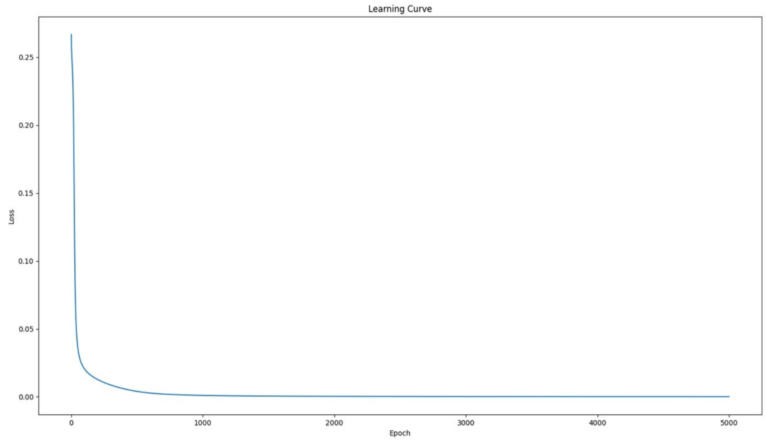
B. Show the accuracy of your prediction

Linear: `Accuracy  1.0`

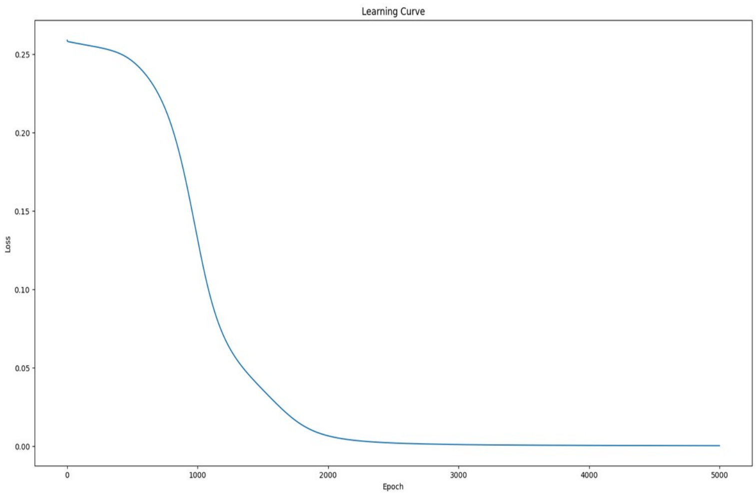
XOR: `Accuracy  1.0`

C. Learning curve (loss, epoch curve)

Linear:



XOR:

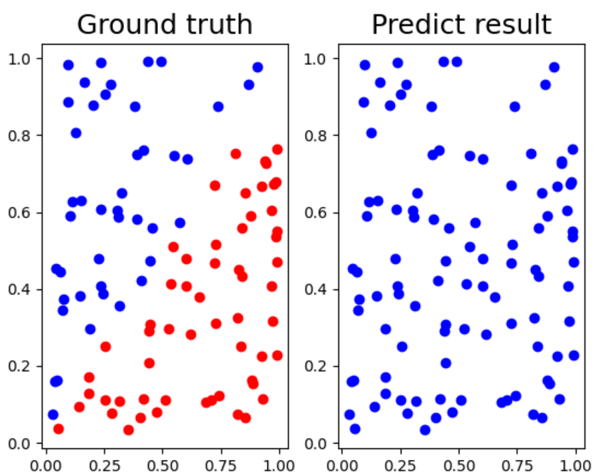


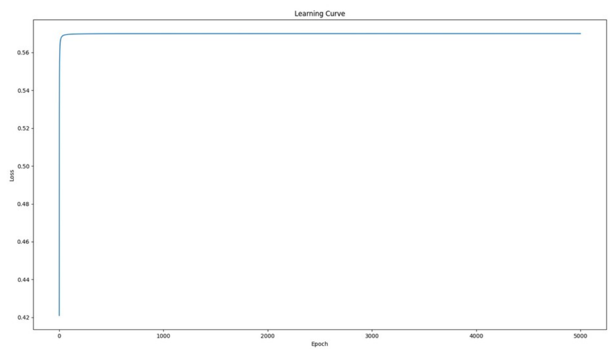
D. anything you want to present

一開始將 prediction 對 loss 的式子寫錯，導致 model train 不起來

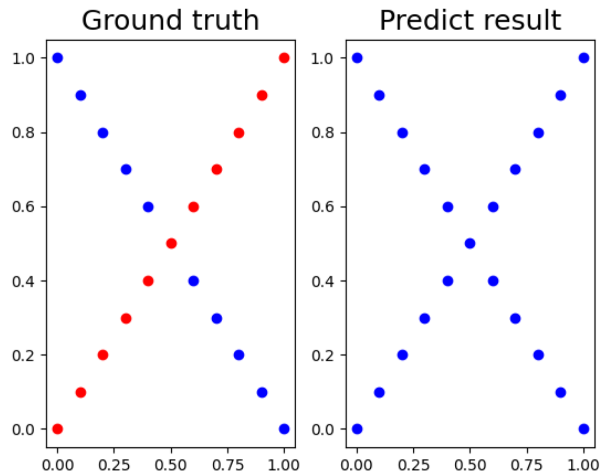
```
C_a2 = y - pred
```

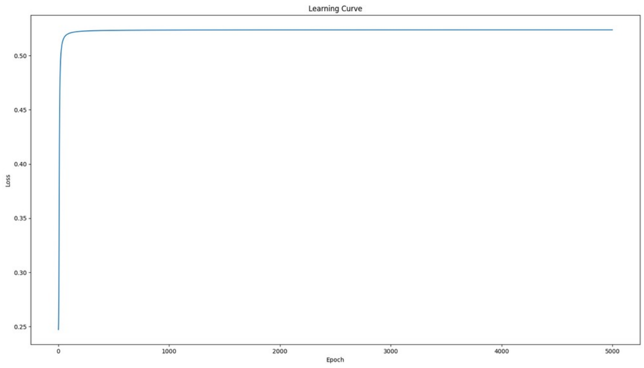
Linear:





XOR:



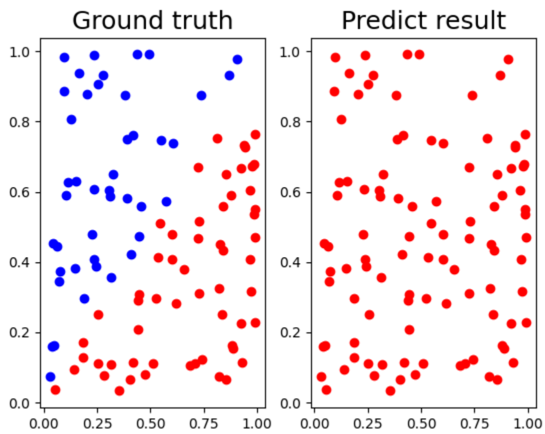


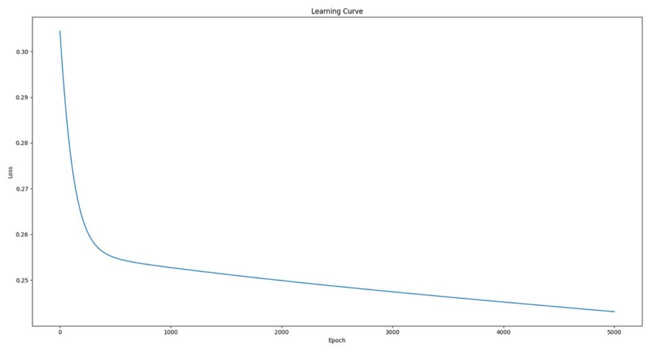
4. Discussion:

A. Try different learning rates

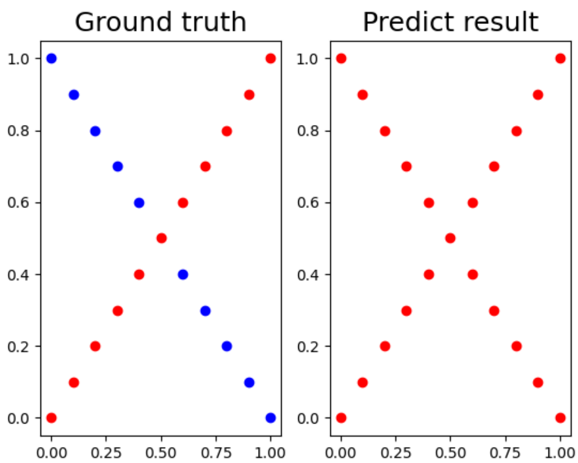
I replace the **1e-1** learning rate with **1e-4** and the result is shown below.

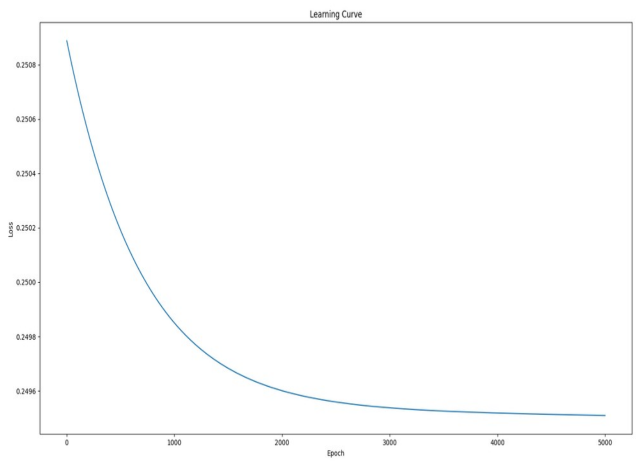
Linear:



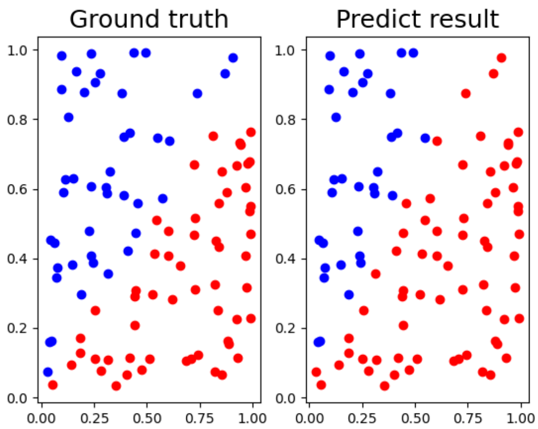


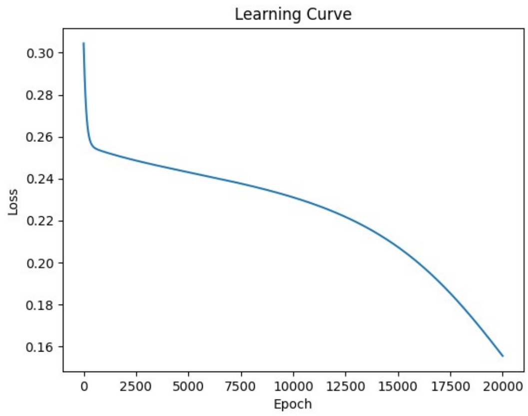
XOR:





可以看到如果將 learning rate 從 1e-1 改成 1e-4 後準確度變得很不好，雖然 loss 有在持續下降，但下降的幅度並不大，因此在同樣 epoch 的情況下並不能像 1e-1 達到很好的訓練效果。



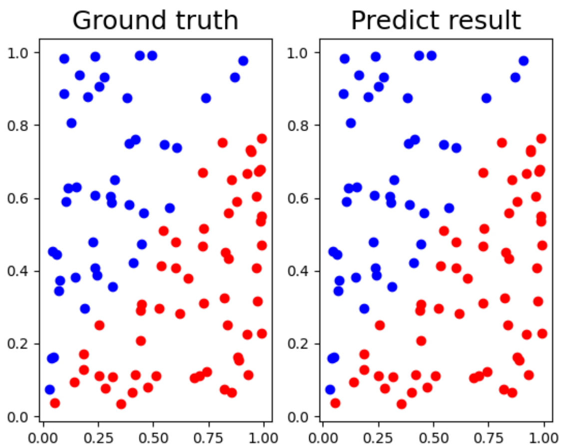


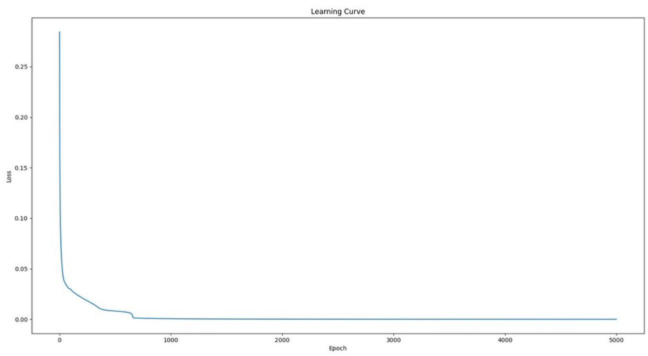
但是將 epoch 的數量加大後，便能達到不錯的訓練效果。

B. Try different numbers of hidden units

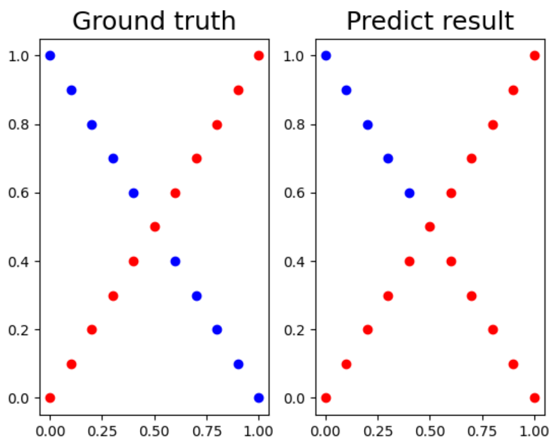
I replace the **10** hidden units with **100** and the result is shown below.

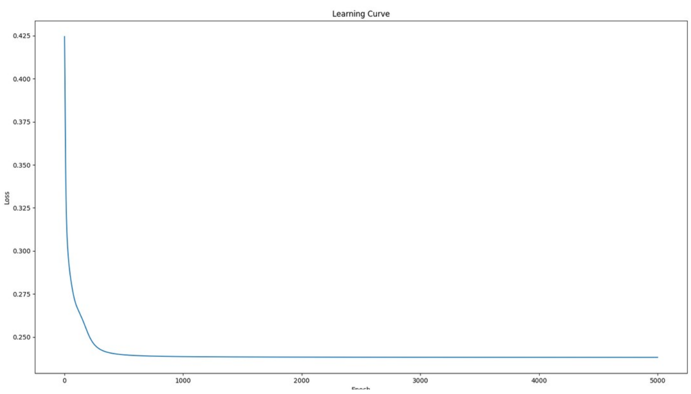
Linear:





XOR:





加大 hidden neuron 的數量並不能保證能有較好或是較快收斂，以 XOR 為例加大 hidden neuron 的數量反而使結果變得不好。因此 hidden neuron 的數量對於 model 的準確率也是很重要的變因之一，要根據想要解決問題的難易度來進行調整，而不是越多越好。

C. Try without activation functions

```
epoch 1 loss : nan
epoch 2 loss : nan
epoch 3 loss : nan
epoch 4 loss : nan
epoch 5 loss : nan
epoch 6 loss : nan
epoch 7 loss : nan
epoch 8 loss : nan
epoch 9 loss : nan
```

```
lab1.py:115: RuntimeWarning: overflow encountered in matmul
  C_W1 = np.concatenate((C_W1, z1_W1 @ a1_z1 @ z2_a1 @ a2_z2.T @
 C_a2.T), axis=1)
```

試著將 activation function 拿掉，會造成 overflow 的問題。因為缺少了 activation function 將輸出壓縮在 0 與 1 之間，造成梯度爆炸，loss 變為 nan

D. Anything you want to share

從這兩個 case 的 learning curve 來看，XOR 的 case 是比較難 train 的。我認為是因為 XOR 的 case 缺乏一個直觀的界線，不像 linear 的 case，兩個 class 有一個很明顯的界線，因此所需要的訓練時間是較長的。

在這次的作業中並沒有實做 bias 但還是可以將 model 訓練的很好，我認為是因為這次的 task 是相對簡單的，因此有沒有加 bias 並沒有太大的區別，可能還會因此增加訓練的時間，但相信在一些困難的 task 上，加上 bias 會對 model 的訓練有很大的影響。

5. Extra

A. Implement different optimizers

None

B. Implement different activation functions

I replace the sigmoid function with **LeakReLU** and the code is shown below.

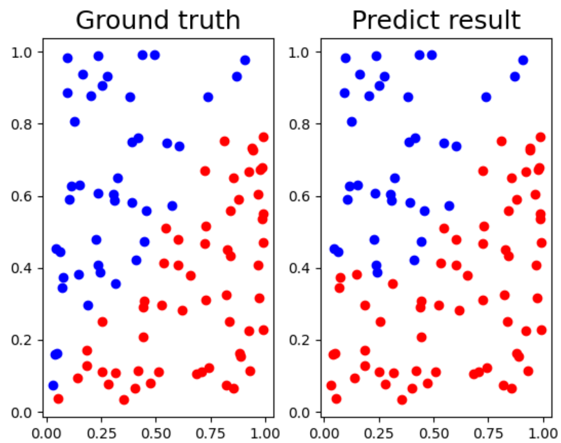
```
def ReLU(x):
    return np.where(x < 0, 0.07 * x, x)

def derivative_ReLU(x):
    return np.where(x < 0, 0.07, 1)
```

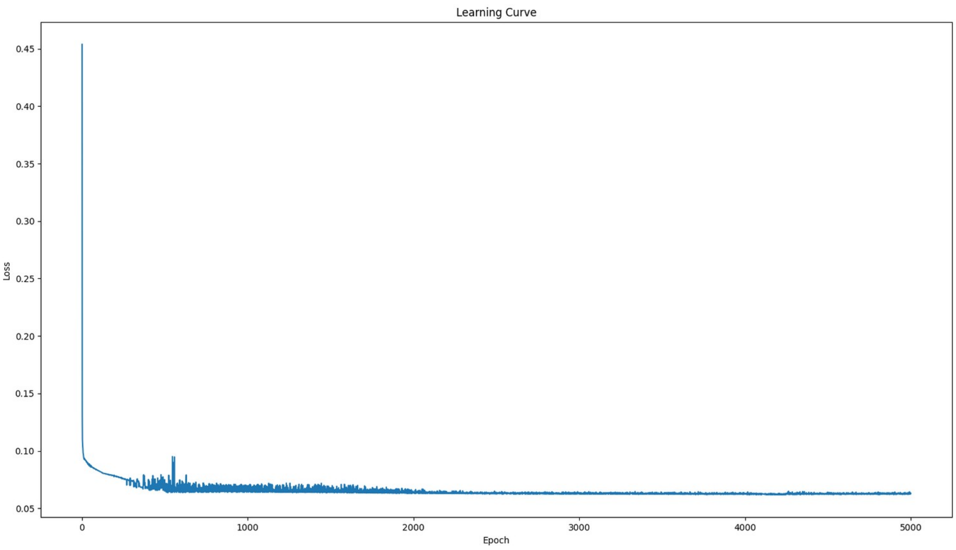
Linear:

```
epoch 4972 loss : 0.06286475213856725
epoch 4973 loss : 0.06238659492504293
epoch 4974 loss : 0.06260776508544139
epoch 4975 loss : 0.06263795876427639
epoch 4976 loss : 0.06239668543236106
epoch 4977 loss : 0.06344192832802178
epoch 4978 loss : 0.06322333580232574
epoch 4979 loss : 0.06279945817831221
epoch 4980 loss : 0.06266576779082145
epoch 4981 loss : 0.06224248452517396
epoch 4982 loss : 0.06256684063234977
epoch 4983 loss : 0.0625039751258828
epoch 4984 loss : 0.06217326932881412
epoch 4985 loss : 0.06274197049983324
epoch 4986 loss : 0.06380583375528923
epoch 4987 loss : 0.062328146918094254
epoch 4988 loss : 0.06253079189565974
epoch 4989 loss : 0.06260370349048568
epoch 4990 loss : 0.06336685419194663
epoch 4991 loss : 0.06231093401485985
epoch 4992 loss : 0.06287447562730152
epoch 4993 loss : 0.06435459338362803
epoch 4994 loss : 0.06263092786042945
epoch 4995 loss : 0.06325244283243343
epoch 4996 loss : 0.06317055133439482
epoch 4997 loss : 0.062338631017545464
epoch 4998 loss : 0.06340087020377963
epoch 4999 loss : 0.06268868836861706
epoch 5000 loss : 0.06306248617932357
```

```
[ 7.29537613e-01]
[-7.67109156e-03]
[ 1.02739333e+00]
[ 6.81050335e-01]
[ 7.95853119e-01]
[-3.37355673e-03]
[-1.19235050e-02]
[ 4.43820570e-01]
[ 1.25162824e+00]
[ 1.13567756e+00]
[-2.34531410e-03]
[ 9.66270166e-01]
[ 2.10470057e-02]
[-1.62368830e-02]
[ 9.89747378e-01]
[-5.10377165e-03]
[-9.73373989e-03]
[-9.15525298e-03]
[ 1.21508100e+00]
[ 9.90625502e-01]
[ 1.03943500e+00]
[-2.90189149e-04]
[ 7.12983622e-01]
[-1.89965322e-02]
[ 1.17074946e+00]
[ 1.03089498e+00]]
```



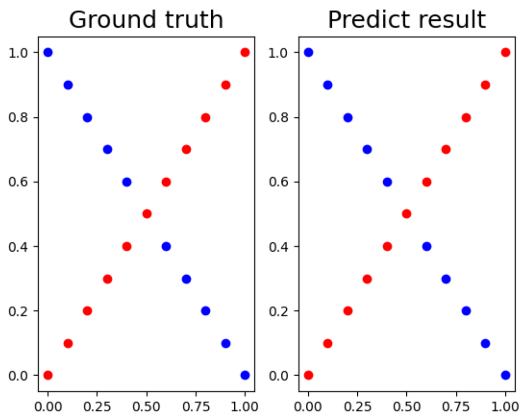

```
Accuracy  0.92
```



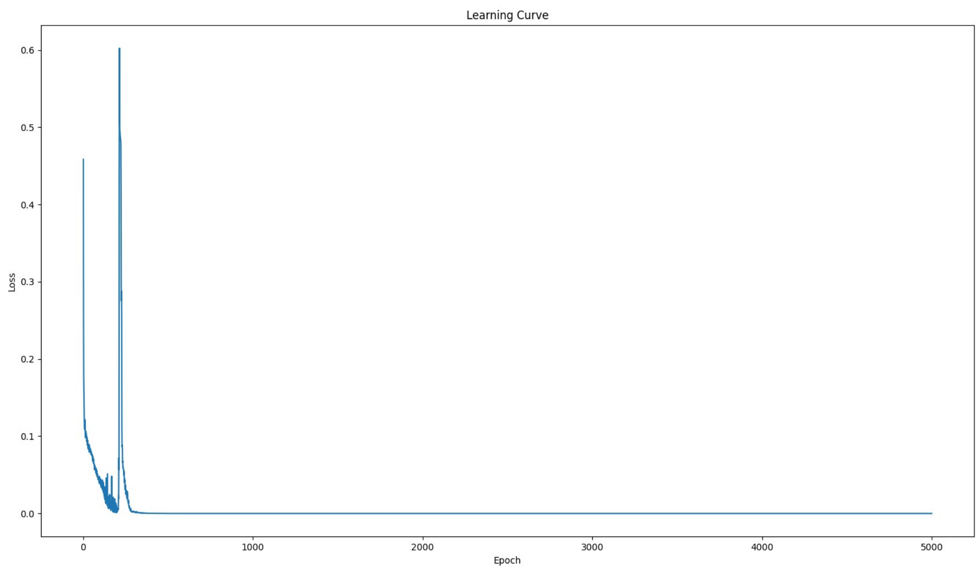
XOR:

```
epoch 4976 loss : 5.816868118755159 4e-31
epoch 4977 loss : 9.444286313128755e-31
epoch 4978 loss : 7.838532145255295e-31
epoch 4979 loss : 6.998038409242187e-31
epoch 4980 loss : 3.3926839680916624e-31
epoch 4981 loss : 3.649719908851547e-31
epoch 4982 loss : 2.6637132070997765e-31
epoch 4983 loss : 4.047174603398250-31
epoch 4984 loss : 5.056845240768311e-31
epoch 4985 loss : 3.789991719342348e-31
epoch 4986 loss : 5.526607526044027e-31
epoch 4987 loss : 8.147174965474466e-31
epoch 4988 loss : 3.841400668229921e-31
epoch 4989 loss : 6.038177066275630-31
epoch 4990 loss : 1.1069194063514752e-30
epoch 4991 loss : 3.434152851486159e-31
epoch 4992 loss : 7.014649158362674e-31
epoch 4993 loss : 5.559783017487058e-31
epoch 4994 loss : 2.170505985075411e-31
epoch 4995 loss : 3.403068995624266e-31
epoch 4996 loss : 2.4246318359450382e-31
epoch 4997 loss : 3.574301147962634e-31
epoch 4998 loss : 2.3841754786931644e-31
epoch 4999 loss : 3.004776998448693e-31
epoch 5000 loss : 2.9794741491815236e-31
```

```
[[ 0.00000000e+00]
 [ 1.00000000e+00]
 [-1.72431514e-17]
 [ 1.00000000e+00]
 [-3.44863027e-17]
 [ 1.00000000e+00]
 [-4.08006962e-17]
 [ 1.00000000e+00]
 [-6.89726054e-17]
 [ 1.00000000e+00]
 [-7.18869408e-17]
 [-8.16013923e-17]
 [ 1.00000000e+00]
 [-1.24344979e-16]
 [ 1.00000000e+00]
 [-1.37945211e-16]
 [ 1.00000000e+00]
 [-1.55431223e-16]
 [ 1.00000000e+00]
 [-1.43773882e-16]
 [ 1.00000000e+00]]
```




```
Accuracy  1.0
```



C. Implement convolutional layers

None