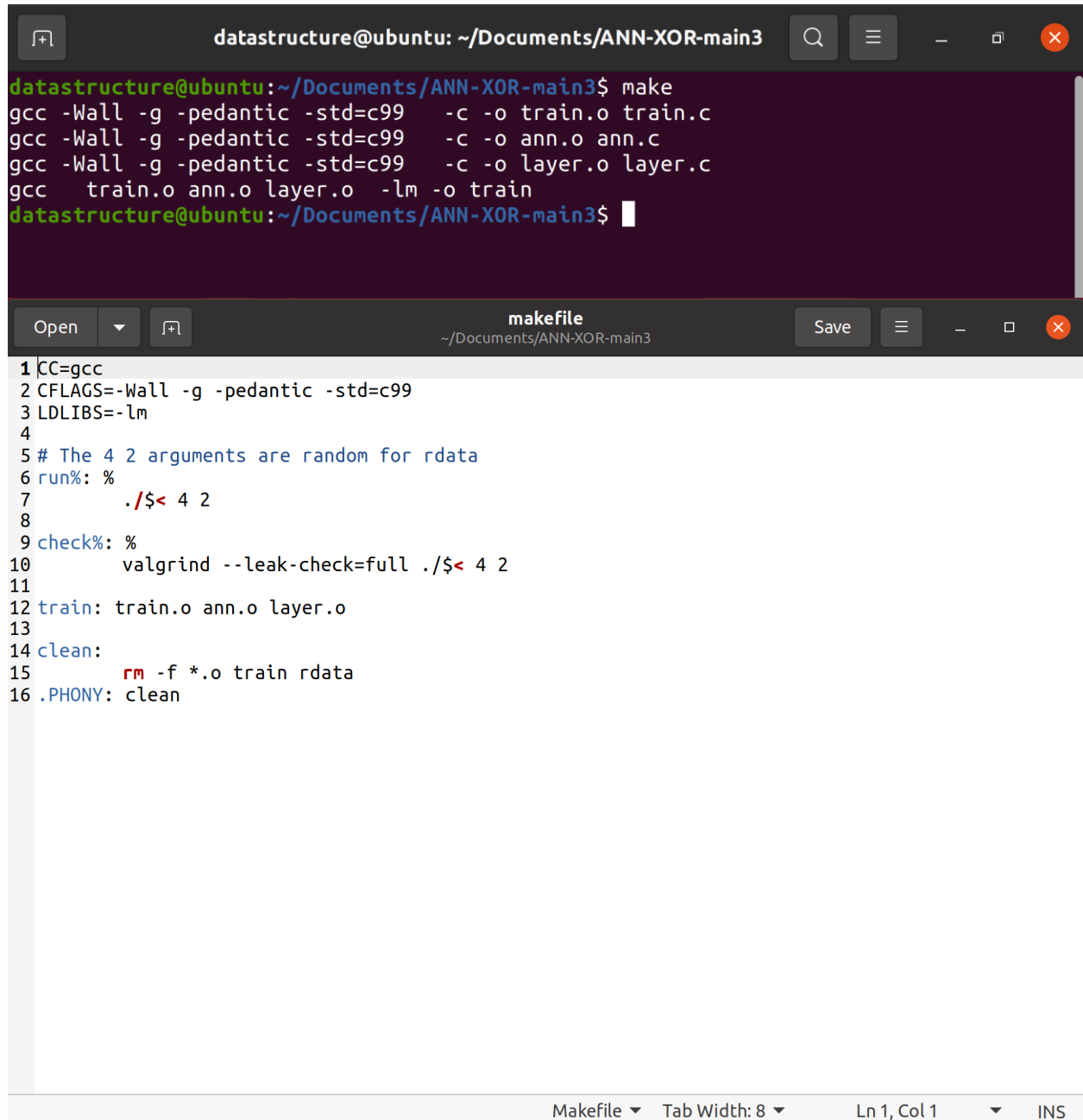


Assighment #0

1. 編譯結果



The image shows a terminal window and a code editor. The terminal window, titled 'datastructure@ubuntu: ~/Documents/ANN-XOR-main3', displays the output of a 'make' command. It shows the compilation of three C files (train.c, ann.c, layer.c) into object files (train.o, ann.o, layer.o) and then linking them into an executable named 'train'. The code editor, titled 'makefile', shows the contents of the Makefile, which defines the compiler (C=gcc), flags (CFLAGS=-Wall -g -pedantic -std=c99), libraries (LDLIBS=-lm), and the build rules for the program.

```
datastructure@ubuntu: ~/Documents/ANN-XOR-main3$ make
gcc -Wall -g -pedantic -std=c99 -c -o train.o train.c
gcc -Wall -g -pedantic -std=c99 -c -o ann.o ann.c
gcc -Wall -g -pedantic -std=c99 -c -o layer.o layer.c
gcc train.o ann.o layer.o -lm -o train
datastructure@ubuntu: ~/Documents/ANN-XOR-main3$
```

```
1 C=gcc
2 CFLAGS=-Wall -g -pedantic -std=c99
3 LDLIBS=-lm
4
5 # The 4 2 arguments are random for rdata
6 run%: %
7     ./${< 4 2
8
9 check%: %
10     valgrind --leak-check=full ./${< 4 2
11
12 train: train.o ann.o layer.o
13
14 clean:
15     rm -f *.o train rdata
16 .PHONY: clean
```

Makefile ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

2. 執行結果

```
datastructure@ubuntu:~/Documents/ANN-XOR-main3$ ./train
Big data machine learning.

-----
PART I - Creating a layer.

Trying to layer_create.
Running layer_init.
Here are some of the properties:
  num_outputs: 2
  num_inputs: 0
  outputs[0]: 0.000000
  outputs[1]: 0.000000

Creating second layer.
Running layer_init on second layer.
Here are some of the properties:
  num_outputs: 1
  num_inputs: 2
  weights[0]: -0.466530
  weights[1]: -0.170036
  biases[0]: 0.000000
  outputs[0]: 0.000000

Computing second layer outputs:
Here is the new output:
  outputs[0]: 0.500000

Freeing both layers.
```



PART II - Creating a neural network.
2 inputs, 8 hidden neurons and 1 output.

```
* - * \  
          * -  
* - * /
```

Initialising network with random weights...
The current state of the hidden layer:

```
weights[0][0]: 0.190636  
weights[0][1]: -0.077513  
weights[0][2]: -0.293735  
weights[0][3]: -0.249872  
weights[0][4]: 0.136559  
weights[0][5]: 0.363622  
weights[0][6]: -0.198344  
weights[0][7]: -0.475076  
weights[1][0]: -0.135007  
weights[1][1]: 0.265381  
weights[1][2]: -0.182140  
weights[1][3]: -0.364272  
weights[1][4]: -0.393255  
weights[1][5]: 0.260672  
weights[1][6]: -0.418328  
weights[1][7]: 0.050956  
biases[0]: 0.000000  
biases[1]: 0.000000  
biases[2]: 0.000000  
biases[3]: 0.000000  
biases[4]: 0.000000  
biases[5]: 0.000000  
biases[6]: 0.000000  
biases[7]: 0.000000  
outputs[0]: 0.000000  
outputs[1]: 0.000000  
outputs[2]: 0.000000  
outputs[3]: 0.000000  
outputs[4]: 0.000000  
outputs[5]: 0.000000  
outputs[6]: 0.000000  
outputs[7]: 0.000000
```

Current random outputs of the network:

```
[0, 0] -> 0.575744  
[0, 1] -> 0.579658  
[1, 0] -> 0.568582  
[1, 1] -> 0.572521
```

Training the network...

The current state of the hidden layer:

```
weights[0][0]: 805.945472
weights[0][1]: -682.349229
weights[0][2]: -660.547427
weights[0][3]: -607.748128
weights[0][4]: 476.325807
weights[0][5]: 743.268419
weights[0][6]: -186.919007
weights[0][7]: -696.847884
weights[1][0]: -808.984264
weights[1][1]: 679.236940
weights[1][2]: 657.421482
weights[1][3]: -607.764541
weights[1][4]: -479.572707
weights[1][5]: -746.344704
weights[1][6]: -186.927165
weights[1][7]: 693.744485
biases[0]: 2.073698
biases[1]: 2.174701
biases[2]: 2.193165
biases[3]: 606.070053
biases[4]: 2.354378
biases[5]: 2.125172
biases[6]: 185.300611
biases[7]: 2.162651
outputs[0]: 0.290233
outputs[1]: 0.293651
outputs[2]: 0.294258
outputs[3]: 0.000000
outputs[4]: 0.299416
outputs[5]: 0.291989
outputs[6]: 0.000000
outputs[7]: 0.293252
```

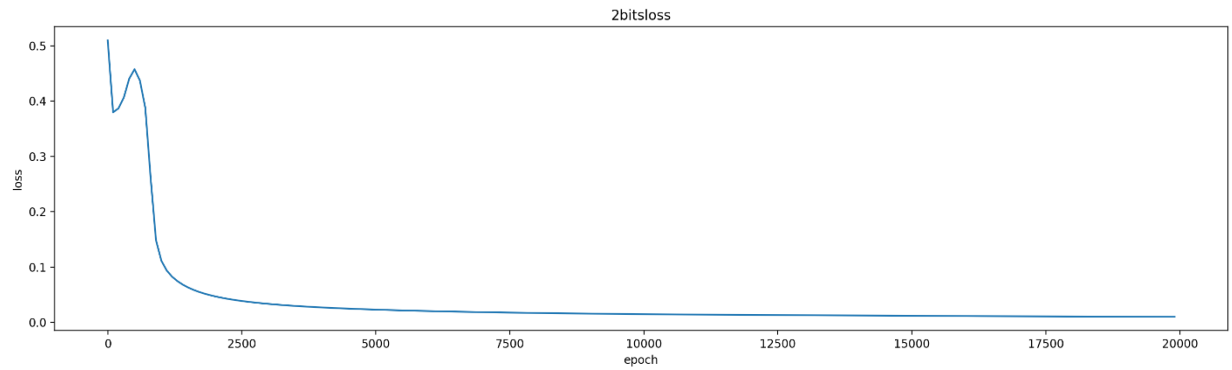
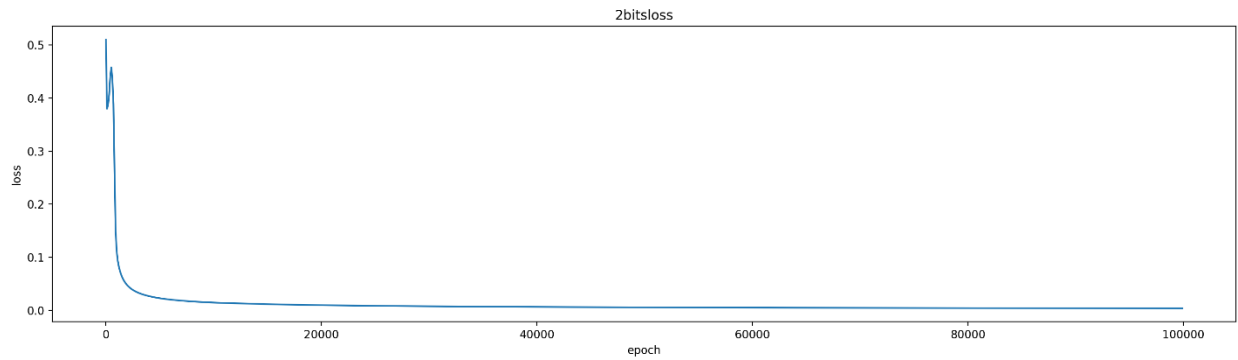
After training magic happened the outputs are:

```
[0, 0] -> 0.002612
[0, 1] -> 0.997584
[1, 0] -> 0.997284
[1, 1] -> 0.002324
```

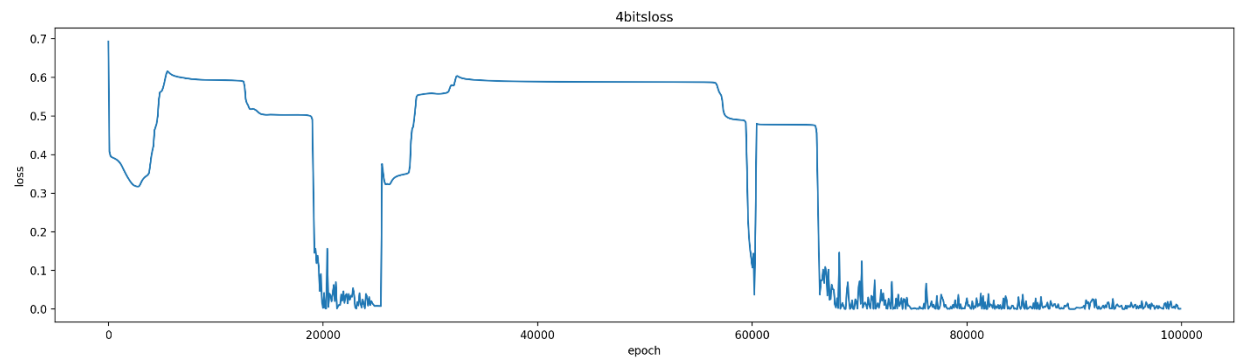
datastructure@ubuntu:~/Documents/ANN-XOR-main3\$

3. 分析

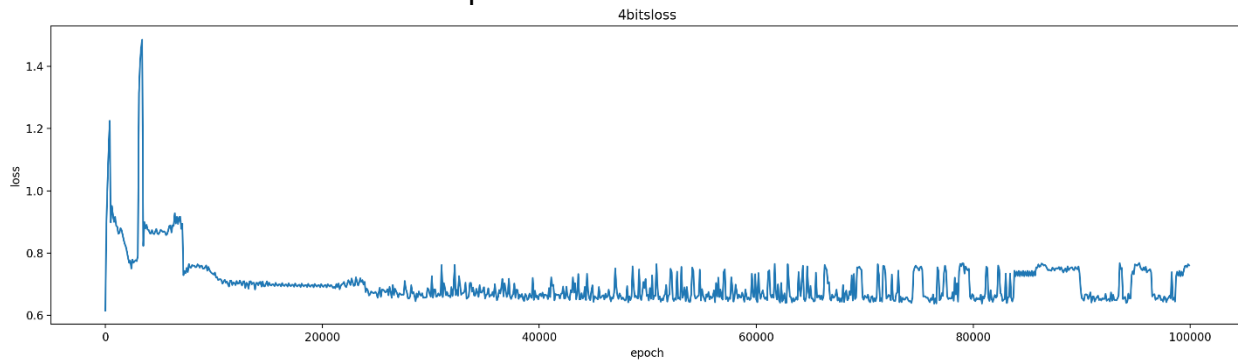
2bits 2x8x1 Neural network



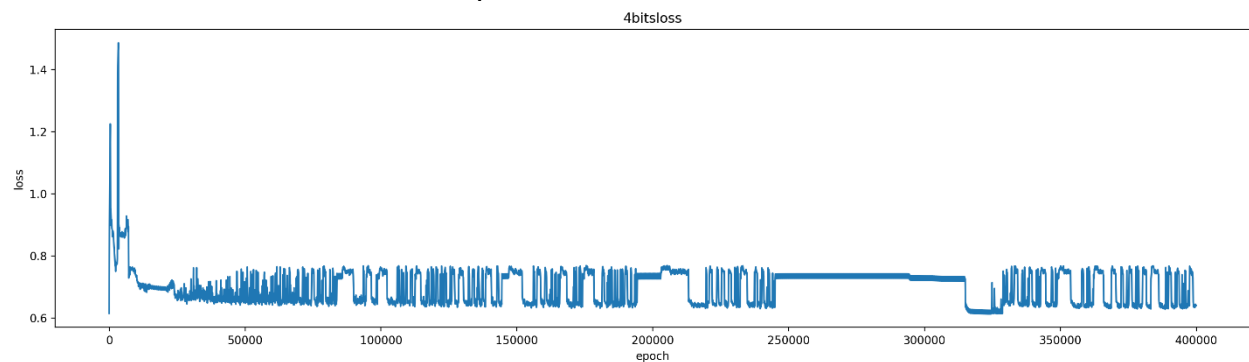
4bits 4x32x1 Neural network



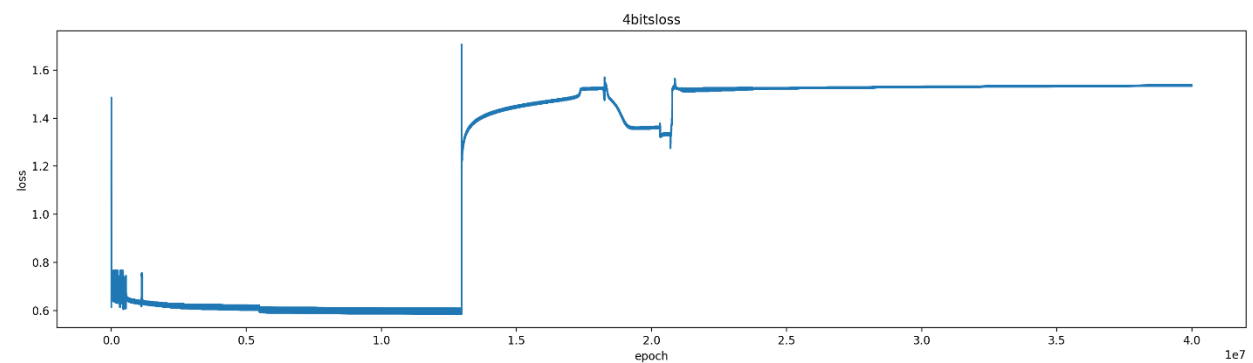
4bits 4x16x8x1 Neural network epoch 100000



4bits 4x16x8x1 Neural network epoch 400000



4bits 4x16x8x1 Neural network epoch 40000000



- 2bits
 - 資料用一層 8 個 neuron 即可 train 起來。
 - 在 1000epoch 左右 loss 有提升，推測可能是找到 local minima 後又進入搜尋。
- 4bits
 - 最後用一層 32 個的 neuron 有 train 起來。
 - 原本已 2bits 的想法去推估 4x16x8x1 的應該要 train 出來，但結果與實際願為，不管 train 多久，好像都沒有辦法達到很好的表現。

結論：

Fully connected 的 network 可能不太適合做 XOR 運算，若運用 RNN 或者 CNN 這類的網路應該可以降低複雜度提升準確度。