Assignment 01

Grades will be given based on:

(1) explanation of results
(2) spotting and explaining of any anomalies in your results

Submit to: IVLE/workbin/assignment 1 submission
Due Date: Sept. 28th, 11:59PM

Question #1

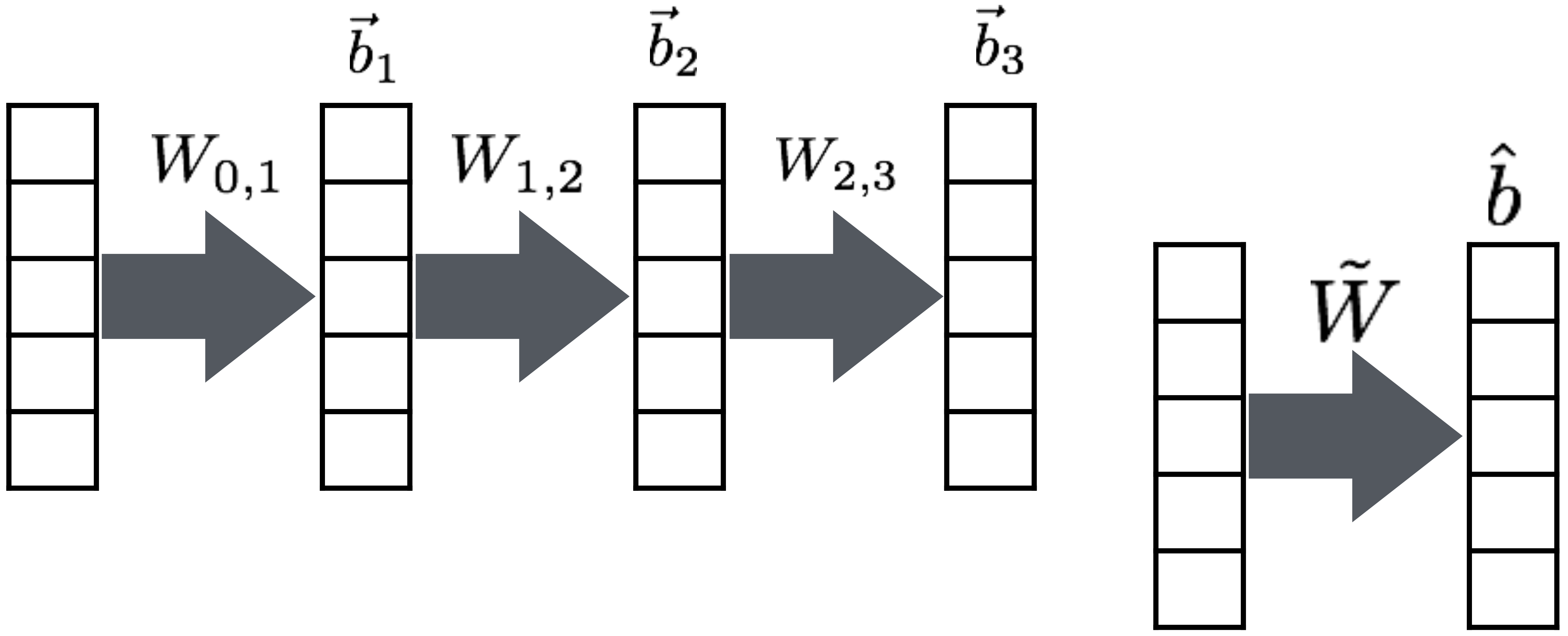
Two networks are said to be equivalent if, they have the same number of input and output nodes, for all inputs, the output of both networks are identical.

Given two neural networks as shown below, all activation functions are identity functions,

$$z = \sigma(z)$$

If the weights and biases are given in the first network with two hidden layers, how do you adjust the weights and biases of the second network with no hidden layers to make the two network equivalent. Provide closed form solution, using mathematical symbols.

Latex or MSWord your solutions and equations for submission in a pdf file.

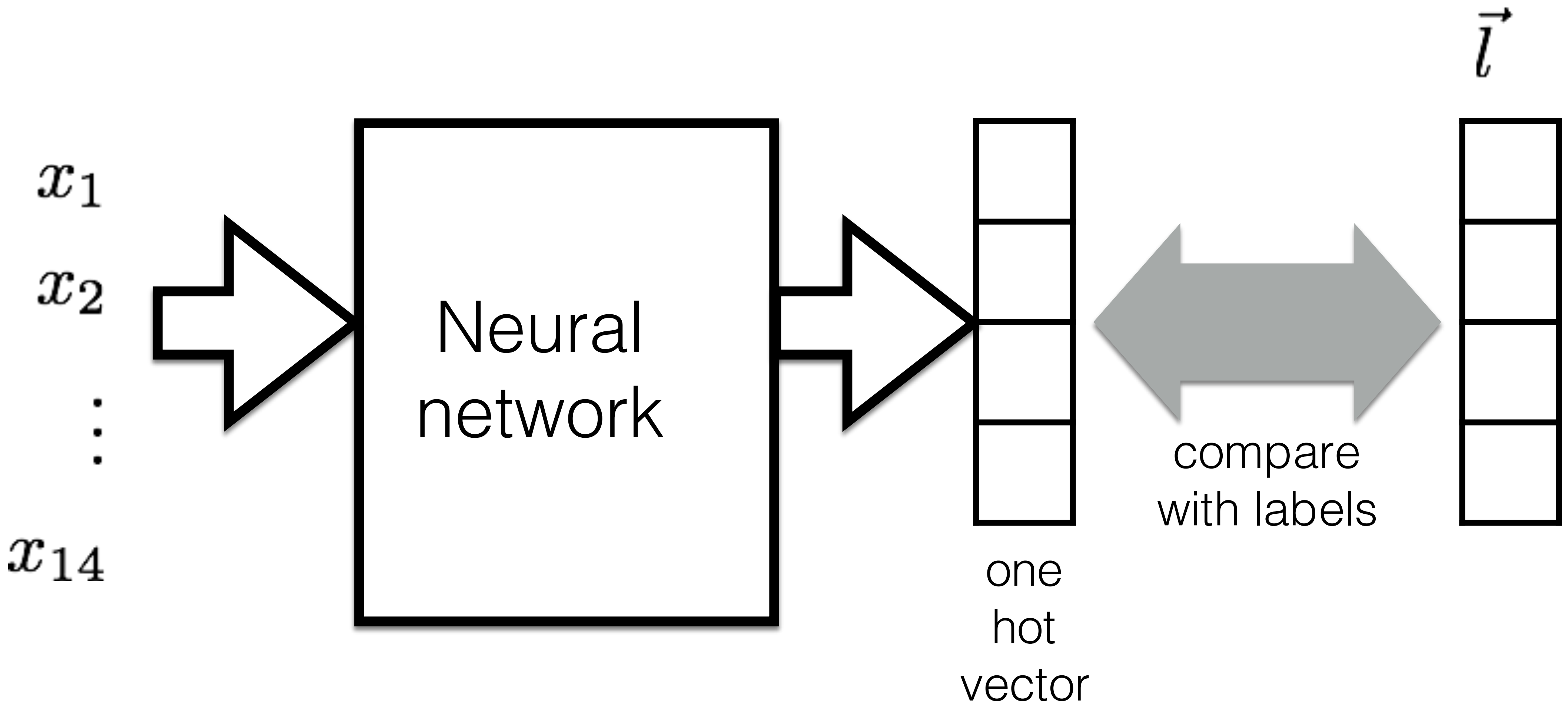


Question #2

Given a data set that represents a function that takes in 14 binary digits and output one of four numbers, 0,1,2,3. Construct a neural network to train for this function.

$$f : \{-1, 1\}^{14} \rightarrow \{0, 1, 2, 3\}$$

$$f(x_1, \ldots, x_{14}) \in \{0, 1, 2, 3\}$$



Question #2

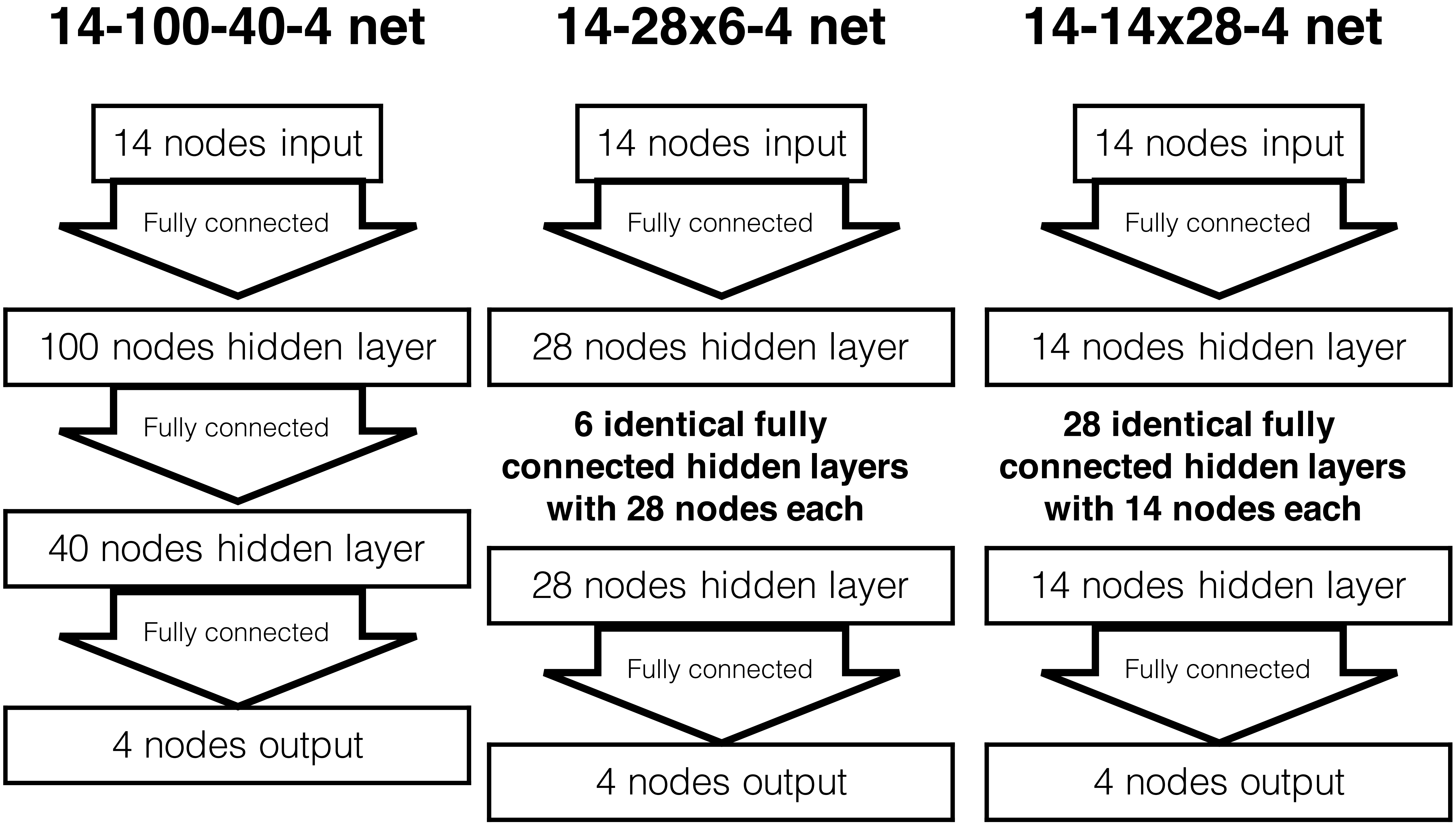
The label vectors are one-hot vectors such that

$$f(x_1, \cdots x_{14}) \in \{0, 1, 2, 3\}$$

$$\vec{l} = \begin{cases} (1,0,0,0) & \text{if } f(x_1, \cdots x_{14}) = 0 \\ (0,1,0,0) & \text{if } f(x_1, \cdots x_{14}) = 1 \\ (0,0,1,0) & \text{if } f(x_1, \cdots x_{14}) = 2 \\ (0,0,0,1) & \text{if } f(x_1, \cdots x_{14}) = 3 \end{cases}$$

For this assignment, construct three networks given in the next slide. Implement the back propagation algorithm and use gradient descend to train the networks. Use **cross entropy cost function** on a softmax function for training.

Question #2



All but the last Fully connected layers have ReLU as the activation function

Question #2

(1) Plot the training cost w.r.t. iterations
(2) Plot the testing cost w.r.t. iterations
(3) Plot the train & test accuracy scores w.r.t. iterations
(4) Check your back propagation intermediate results against known answers (see page 8 to 13 for details):
   a. You will be given one special data point d.
   b. You will be given one weight & bias set W0 together with correct gradients computed using data point d and W0.

$$\frac{\partial \text{output}}{\partial w_j} \qquad \frac{\partial \text{output}}{\partial b_j}$$

Note: Output here means the loss function

   c. You will be given another weight & bias set W1 with no corresponding gradients given.
   d. Use (a) and (b) to compute gradients and compare to the given correct gradients
   e. Use (a) and (c) to compute gradients and submit your gradient values for grading.

Your submissions will be automatically graded using a script. Be sure to format your output according to instructions in the next slide. Incorrect format will be graded as incorrect answers

Question #2

Give a half page discussion on why the three networks 14-100-40-4 net,14-28x6-4 net,14-14x28-4 net perform differently.

Which one performs better and why.

Given files

For Question#2(1)(2)(3)

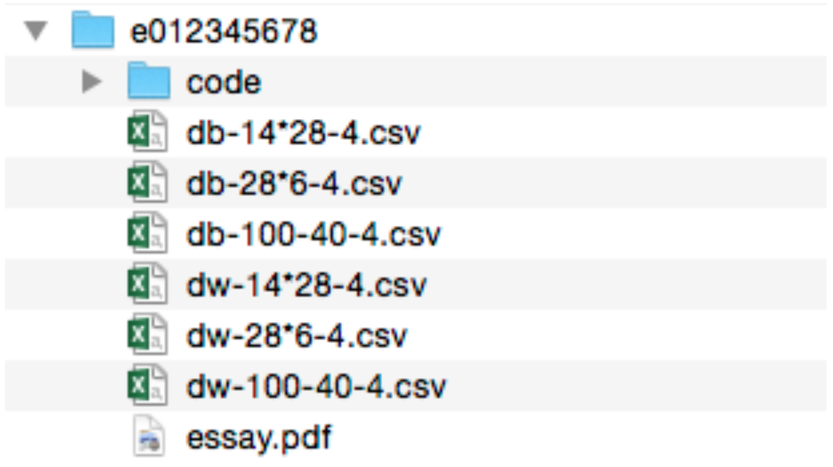
- The data given to you is under path Question2_123.
- It contains 4 csv files, (i.e. x_train, x_test, y_train, y_test). In the x_* files, each line is a datapoint of 14 dimensions, where in the y_* files, each line is the respective labels which are corresponding to the ones in x_*.

For Question#2(4)

- The data given to you is under path Question2_4
- You will be given weights and gradients for verification, which is under the '**b**' folder, where the weights you will work on is in the '**c**' folder. To ease your understanding, in the weights csv, there's one heading column introducing the weights/biases, while there is **NO** such column for the gradients. And you should **NOT** include the headings in your submission as well.
- The given data point x and label y is in the '**a.txt**' file. For both verification and test, we use the same data point and label.
- Your submission should be the same format as the given 'true-d*' files.
- If possible, use **np.float32** to control the granularity of your gradients, otherwise, round your results to at least **16** digits, or **e-16**.

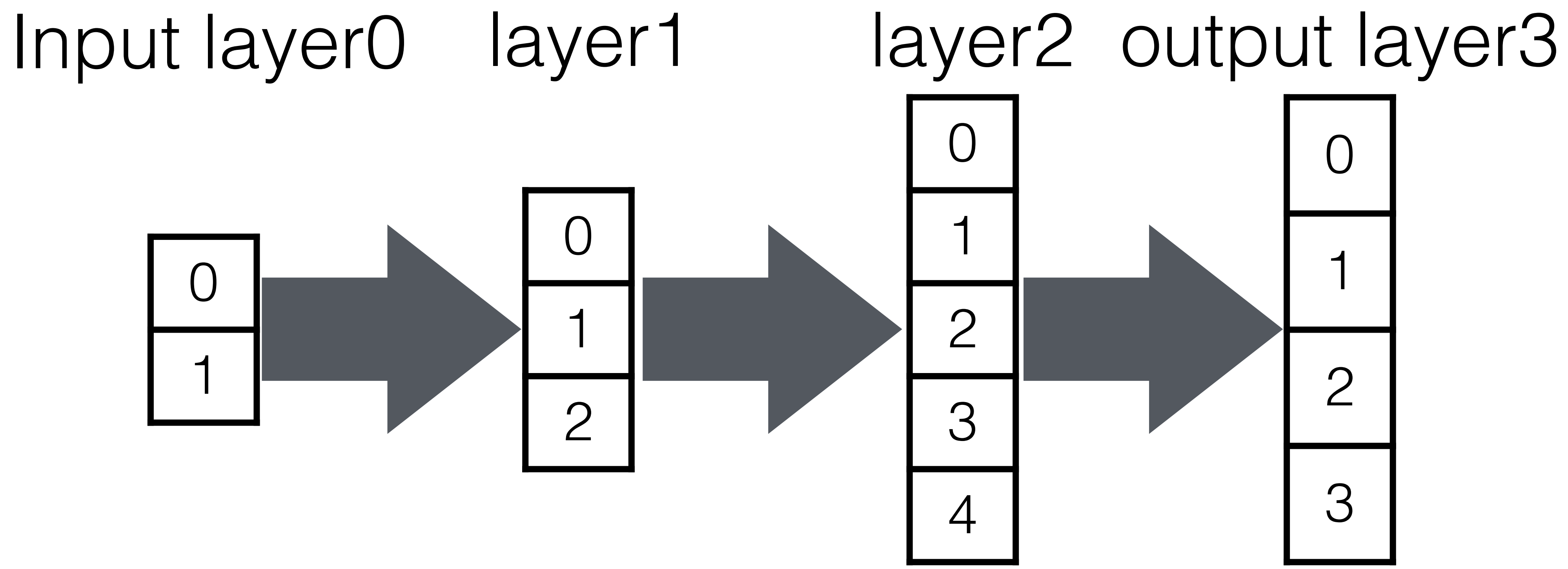
Submission

- Your submission is a single .zip file. Other compressions are **NOT** acceptable.
- The naming of the .zip file is your ID as shown on the IVLE/class. We will use it for grading. For most cases, it's your NUS NETID, which will be e******.zip ('e' in lowercase).
- Inside the .zip file, there are 8 files. They are 1 pdf, 6 csv and 1 folder containing your code.
- For the essay part of question1 and question 2. You will submit a single pdf file. Please section your document properly.
- For question2(4), the output files are csv files, with comma (i.e.',') as the delimiter. Using space or other delimiters are **NOT** acceptable.
- Csv namings are as follow: `dw-100-40-4.csv`, `db-100-40-4.csv`, `dw-28-6-4.csv`, `db-28-6-4.csv`, `dw-14-28-4.csv`, `db-14-28-4.csv`,which correspond to the gradients of weights and biases for the three network configurations: *14-100-40-4*, *14-28*6-4*, *14-14*28-4*. Other naming are **NOT** acceptable. (**The naming has changed to be compatible with win & *nix**)
- Since this task you are not expected to work with platforms, so aside from the 6 csv file, please upload your code. You should pack your code (only codes) in a folder and compress along with the other files



```
e012345678
  code
  db-14*28-4.csv
  db-28*6-4.csv
  db-100-40-4.csv
  dw-14*28-4.csv
  dw-28*6-4.csv
  dw-100-40-4.csv
  essay.pdf
```

More Details for Question#2(4)



- Suppose layer(t) has N_t number of nodes, so the weights from layer(t) to layer(t+1) form a N_t by N_{t+1}matrix, where the (i, j)-th entry of this matrix represents the weight connecting the i-th node of layer(t) to the j-th node of layer(t+1).

- The bias then is simply a length-N_t vector as for layer(t). Noted that the input layer 0 has no corresponding bias.

- Softmax is not considered to be a layer in this context, so the output layer output logits.

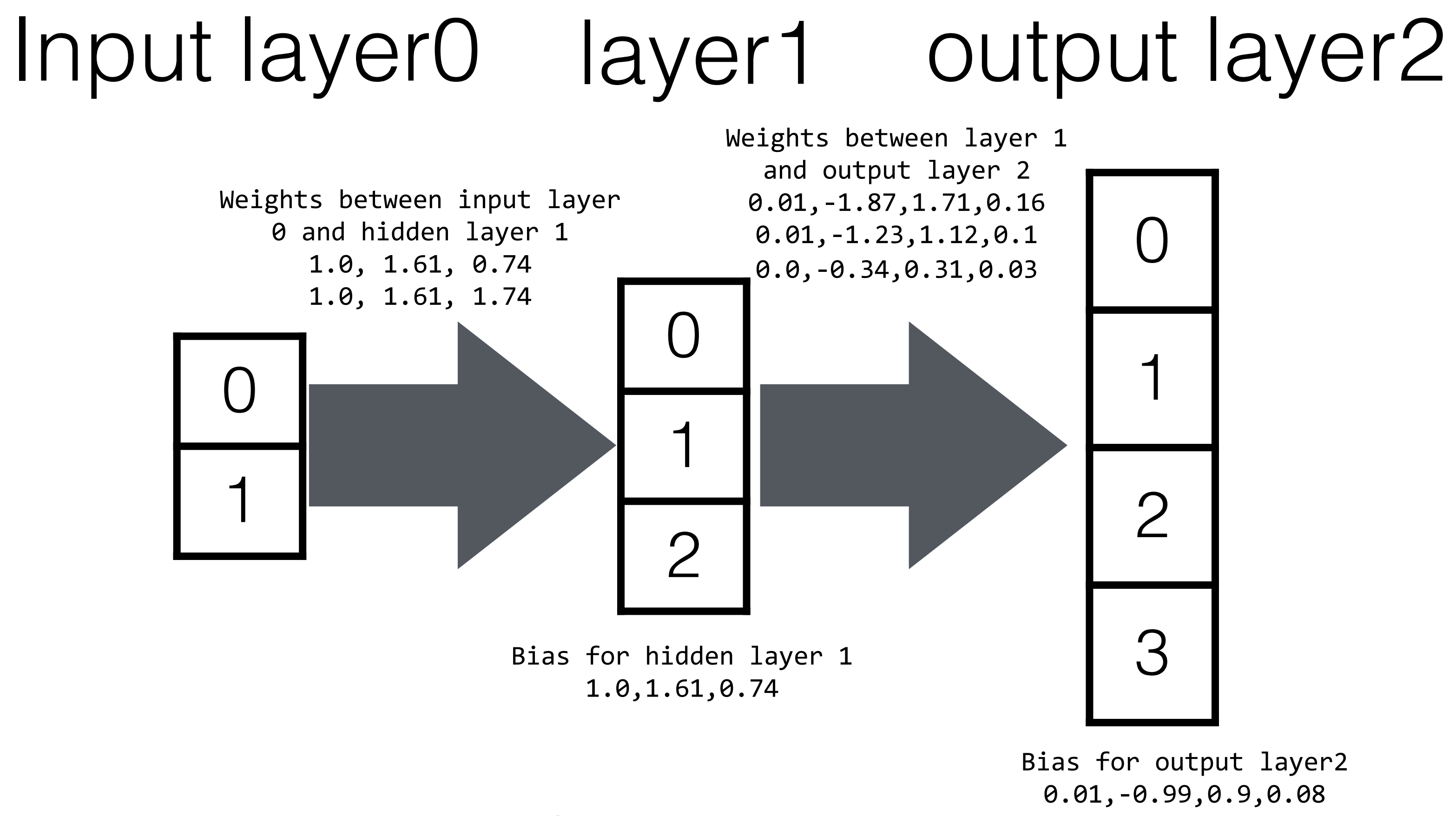
The given weights and corresponding gradients output file is of the following format:

- Totally $\sum_t^{\# of layers-1} N_t$rows.
- The first N_0 rows are the matrix from input layer0 to hidden layer1, the following N_1 rows are the matrix from layer1 to layer2, and so on so forth. There's **NO blank line** between matrix(t) and matrix(t+1)
- Then of each matrix, the (i, j) item is the derivative, which is d (output)/ d w(i,j) of the corresponding node.

The given biases and corresponding gradients output file is similar:

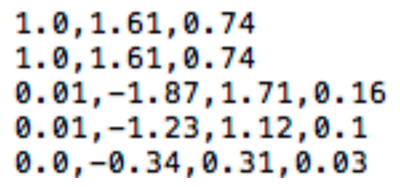
- Totally $\sum_t^{\# of layers-1} 1$rows.
- The first row is the bias vector for layer1, and so has N_1 items. The second line is the bias vector for layer2 with N_2 items, and so on so forth.
- Of each row, the i-th item is d(output)/d b(i) of the corresponding node.

Example



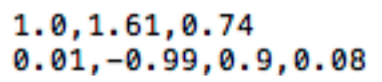
Suppose we have a 2-3-4 fully connected network, and a data point X=(x1,x2) as input, and the output logits vector is of 4 dimension.

Then, the weight matrix looks like the following:



```
1.0,1.61,0.74
1.0,1.61,0.74
0.01,-1.87,1.71,0.16
0.01,-1.23,1.12,0.1
0.0,-0.34,0.31,0.03
```

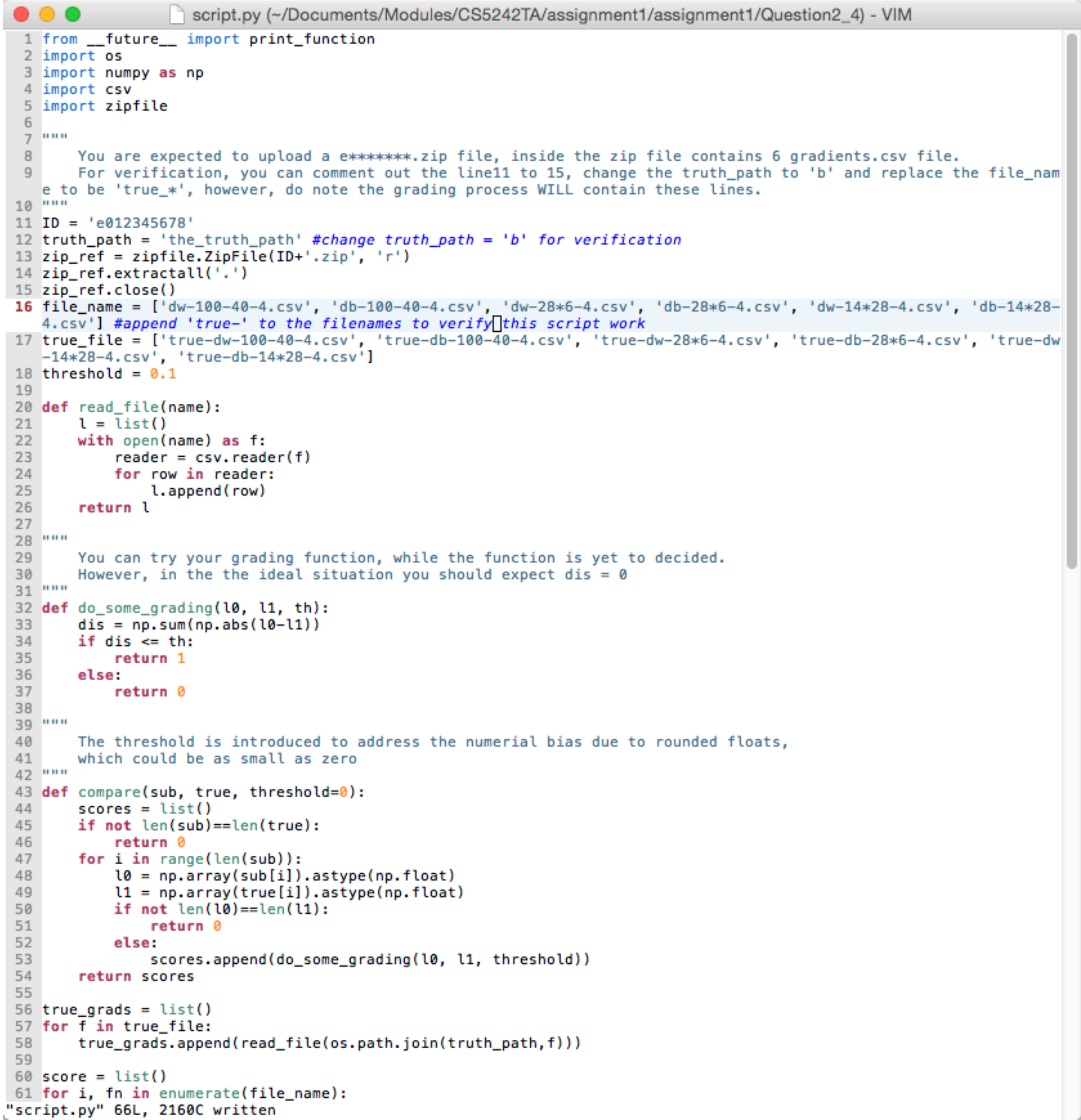
And the biases:



```
1.0,1.61,0.74
0.01,-0.99,0.9,0.08
```

Script

- Your result will be graded using the script similar to the one on the right. If you run it for verification for '**b**', you would get an output **781** .
- This script will be upload along with the other files.
- Do try running your code with this script to adjust your formatting. Otherwise you are likely to receive no point.☺



script.py (~/Documents/Modules/CS5242TA/assignment1/assignment1/Question2_4) - VIM

```
 1 from __future__ import print_function
 2 import os
 3 import numpy as np
 4 import csv
 5 import zipfile
 6 
 7 """
 8     You are expected to upload a e*******.zip file, inside the zip file contains 6 gradients.csv file.
 9     For verification, you can comment out the line11 to 15, change the truth_path to 'b' and replace the file_nam
e to be 'true_*', however, do note the grading process WILL contain these lines.
10 """
11 ID = 'e012345678'
12 truth_path = 'the_truth_path' #change truth_path = 'b' for verification
13 zip_ref = zipfile.ZipFile(ID+'.zip', 'r')
14 zip_ref.extractall('.')
15 zip_ref.close()
16 file_name = ['dw-100-40-4.csv', 'db-100-40-4.csv', 'dw-28*6-4.csv', 'db-28*6-4.csv', 'dw-14*28-4.csv', 'db-14*28-
4.csv'] #append 'true-' to the filenames to verify this script work
17 true_file = ['true-dw-100-40-4.csv', 'true-db-100-40-4.csv', 'true-dw-28*6-4.csv', 'true-db-28*6-4.csv', 'true-dw
-14*28-4.csv', 'true-db-14*28-4.csv']
18 threshold = 0.1
19 
20 def read_file(name):
21     l = list()
22     with open(name) as f:
23         reader = csv.reader(f)
24         for row in reader:
25             l.append(row)
26     return l
27 
28 """
29     You can try your grading function, while the function is yet to decided.
30     However, in the the ideal situation you should expect dis = 0
31 """
32 def do_some_grading(l0, l1, th):
33     dis = np.sum(np.abs(l0-l1))
34     if dis <= th:
35         return 1
36     else:
37         return 0
38 
39 """
40     The threshold is introduced to address the numerial bias due to rounded floats,
41     which could be as small as zero
42 """
43 def compare(sub, true, threshold=0):
44     scores = list()
45     if not len(sub)==len(true):
46         return 0
47     for i in range(len(sub)):
48         l0 = np.array(sub[i]).astype(np.float)
49         l1 = np.array(true[i]).astype(np.float)
50         if not len(l0)==len(l1):
51             return 0
52         else:
53             scores.append(do_some_grading(l0, l1, threshold))
54     return scores
55 
56 true_grads = list()
57 for f in true_file:
58     true_grads.append(read_file(os.path.join(truth_path,f)))
59 
60 score = list()
61 for i, fn in enumerate(file_name):
"script.py" 66L, 2160C written
```