Machine Learning in Engineering Science

National Cheng Kung University Department of Engineering Science

Instructor: Chi-Hua Yu

Name:________________________________

Student ID:____________________________

Mock Exam

注意事項

1. 期中考試時間為 **09:10~12:00**。
2. 本次考試可以 **open book**，使用電子書者可以攜帶 ipad。
3. 考試時皆**不可使用網路查找答案**，發現有使用網路者一律視為作弊，以零分計算。
4. 程式題部分，請繳交 **ipynb 的檔案形式**，並輸入正確的檔名。
5. 請用**學號_Midterm** 為檔名做一個資料夾**(e.g., N96091350_Midterm)**，並將程式題之**.ipynb 檔案**放入資料夾中，壓縮後上傳至課程網站**(e.g., N96091350_Midterm.zip)**。
6. 如未依照上述規則繳交作業、繳交錯誤檔案，則以零分計算，不允許要分。
7. 手寫題可跳題作答，但必須標示清楚題號，若題號標示錯，該題也會視為零分，不允許要分。如字跡潦草至助教難以辨別，則會以助教辨視為主。
8. 程式題請依照題目規定作答，若無依照題目則將該題視為零分，不允許要分。
9. 請注意作答時不要抄襲網路或是同學的答案，助教會將程式碼放入自動比對程式，只要超過 70%相似度，以抄襲處置，抄襲者與被抄襲者都以零分計算。
10. 本次閱卷將採用自動批改，命名錯誤或是無法執行將被自動判定為失敗，失去該題的分數。成功通過自動閱卷的程式碼，助教會再進行人工判讀，確定程式邏輯是否恰當，因此添加最低限度的註解可以保障作答時候的分數。

請勿抄襲，抄襲者與被抄襲者本次考試皆 0 分計算

Machine Learning in Engineering Science

National Cheng Kung University Department of Engineering Science

Instructor: Chi-Hua Yu

Total (120%)

Part I (40%) Concept and Derivation.

1. (40%) Consider the simple network example with a single input $x = 2$ and a single output $y = 1$ shown in Figure 1 below.

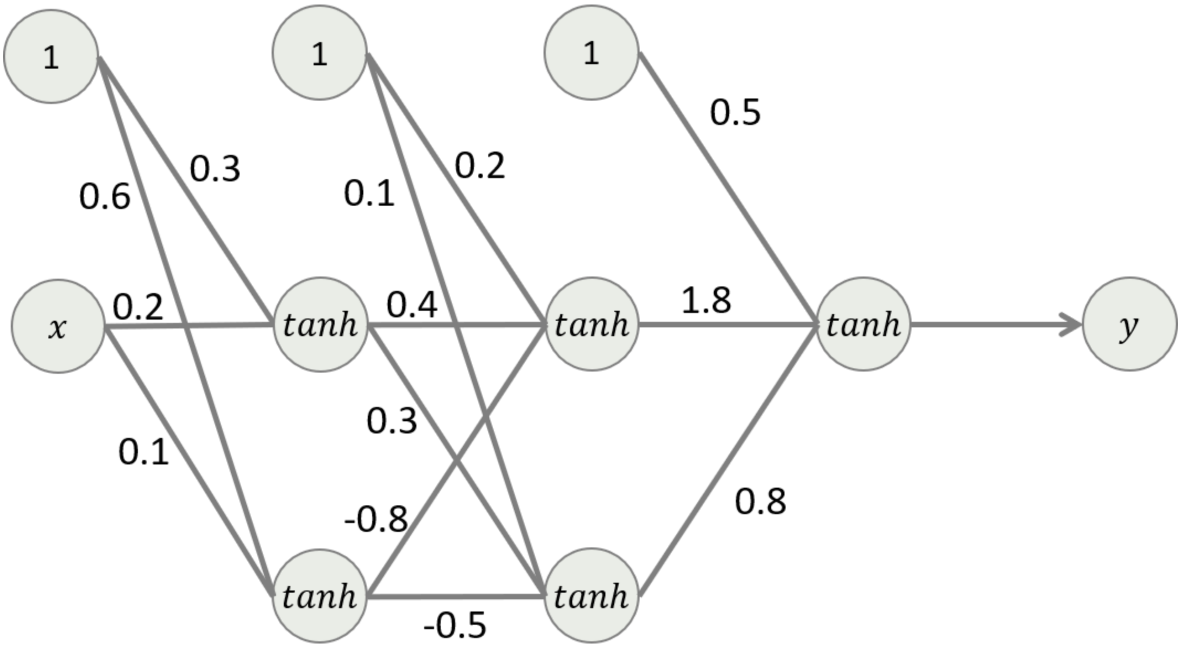


Figure 1

The weight matrices are:

$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.3 & 0.6 \\ 0.2 & 0.1 \end{bmatrix}; \ \mathbf{W}^{(2)} = \begin{bmatrix} 0.2 & 0.1 \\ 0.4 & 0.3 \\ -0.8 & -0.5 \end{bmatrix}; \ \mathbf{W}^{(3)} = \begin{bmatrix} 0.5 \\ 1.8 \\ 0.8 \end{bmatrix}$$

and the summation of weighted nodes for layer 1 can be expressed as $\mathbf{u}^{(1)} = (\mathbf{W}^{(1)})^T \mathbf{x}^{(0)}$; you can perform similar operation for other layers.

(a) (10%) Derive and compute $\mathbf{u}^{(1)}$, $\mathbf{z}^{(1)}$, $\mathbf{u}^{(2)}$, $\mathbf{z}^{(2)}$, and $\mathbf{y}^{(3)}$.

(b) (10%) Using the half of the sum square as our error function, derive and compute $\boldsymbol{\delta}^{(3)}$, $\boldsymbol{\delta}^{(2)}$, $\boldsymbol{\delta}^{(1)}$.

(c) (10%) Compute $\frac{\partial E_n}{\partial \mathbf{W}^{(1)}}$, $\frac{\partial E_n}{\partial \mathbf{W}^{(2)}}$, $\frac{\partial E_n}{\partial \mathbf{W}^{(3)}}$.

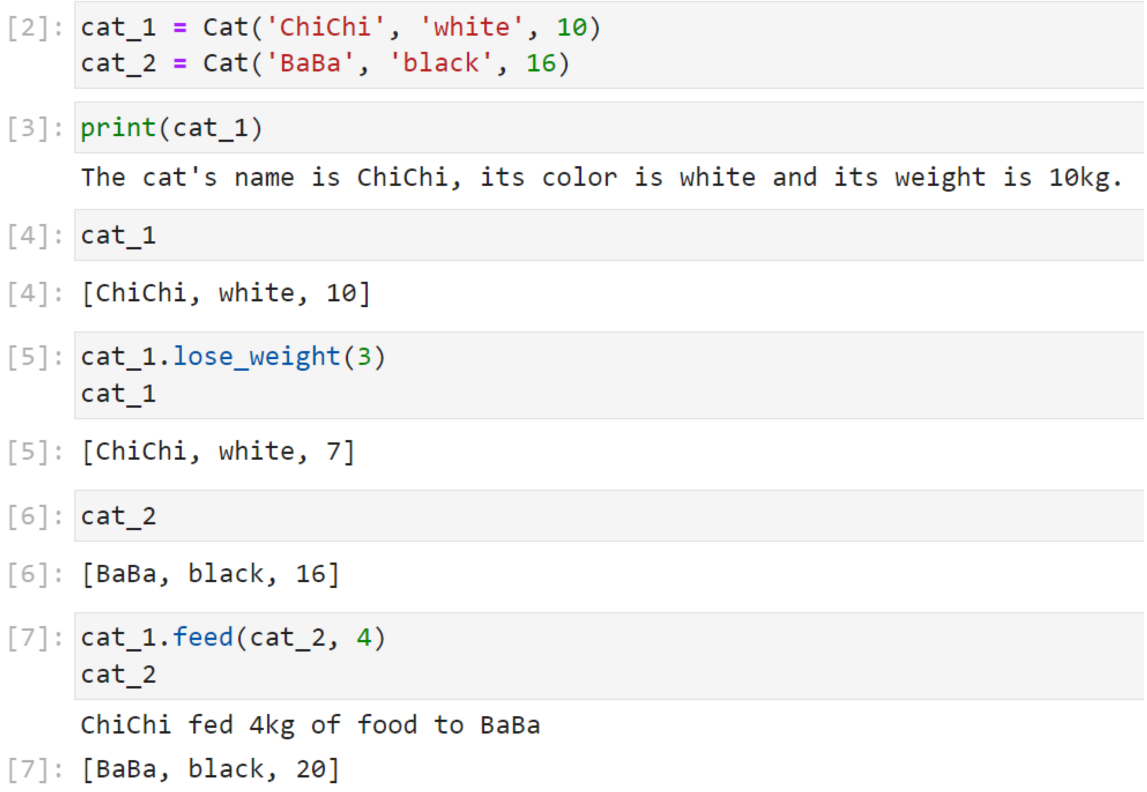
(d) (10%) Update the weight matrices using learning rate $\eta = 0.5$, repeat the forward propagation and compute $\mathbf{u}^{(1)}$, $\mathbf{z}^{(1)}$, $\mathbf{u}^{(2)}$, $\mathbf{z}^{(2)}$, and $\mathbf{y}^{(3)}$.

Part II (80%) Programming Problems.

1. **(30%)** Name your file `cat.ipynb`. Write a program to complete the cat class. The following functions need to be completed including `__str__()`, `__repr__()`, `lose_weight()`, and `feed()`. You can write functions according to the following example. Please refer to the example below to conduct user testing of `Cat` class.

   Below is the sample output:

```
class Cat():
    def __init__(self, name, color, weight):
        ...
```



```
[2]: cat_1 = Cat('ChiChi', 'white', 10)
     cat_2 = Cat('BaBa', 'black', 16)

[3]: print(cat_1)

     The cat's name is ChiChi, its color is white and its weight is 10kg.

[4]: cat_1

[4]: [ChiChi, white, 10]

[5]: cat_1.lose_weight(3)
     cat_1

[5]: [ChiChi, white, 7]

[6]: cat_2

[6]: [BaBa, black, 16]

[7]: cat_1.feed(cat_2, 4)
     cat_2

     ChiChi fed 4kg of food to BaBa
[7]: [BaBa, black, 20]
```

2. **(50%)** Name your Jupyter notebook `YourID_MNIST.ipynb(n96081494_MNIST)`. Please create an ANN model to classify images of handwritten digits. Please use from `torchvision.datasets` import `MNIST` to read the training dataset.

(a) (10%) Write the following class to create the neural network. Please pass the model's parameters such as input size and layer characteristics as arguments into the model.

```
class NeuralNet(nn.Module):
    def __init__(self ,input_size, hidden_1, …):
        super(NeuralNet, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(input_size, hidden_1),
            ...
        )

    def forward(self, x):
        x = x.view(x.shape[0], -1)
        return self.layers(x)

model = NeuralNet(input_size, hidden_1, …)
```

(b) (10%) Write the following function to complete the training loop. Please pass what the training loop needs as arguments to the trainer.

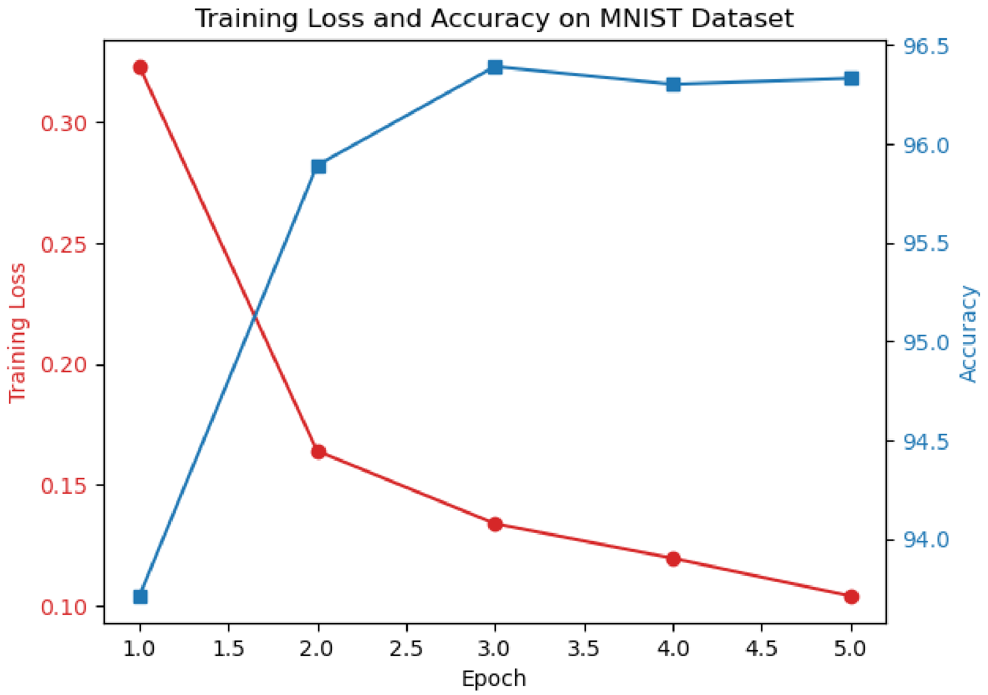
```
epochs = 5
training_losses = []
accuracies = []
def trainer(model,epochs=1,verbose=True, …):
    for epoch in range(epochs):
        ...

trainer(model,epochs=epochs,verbose=True, …)
```

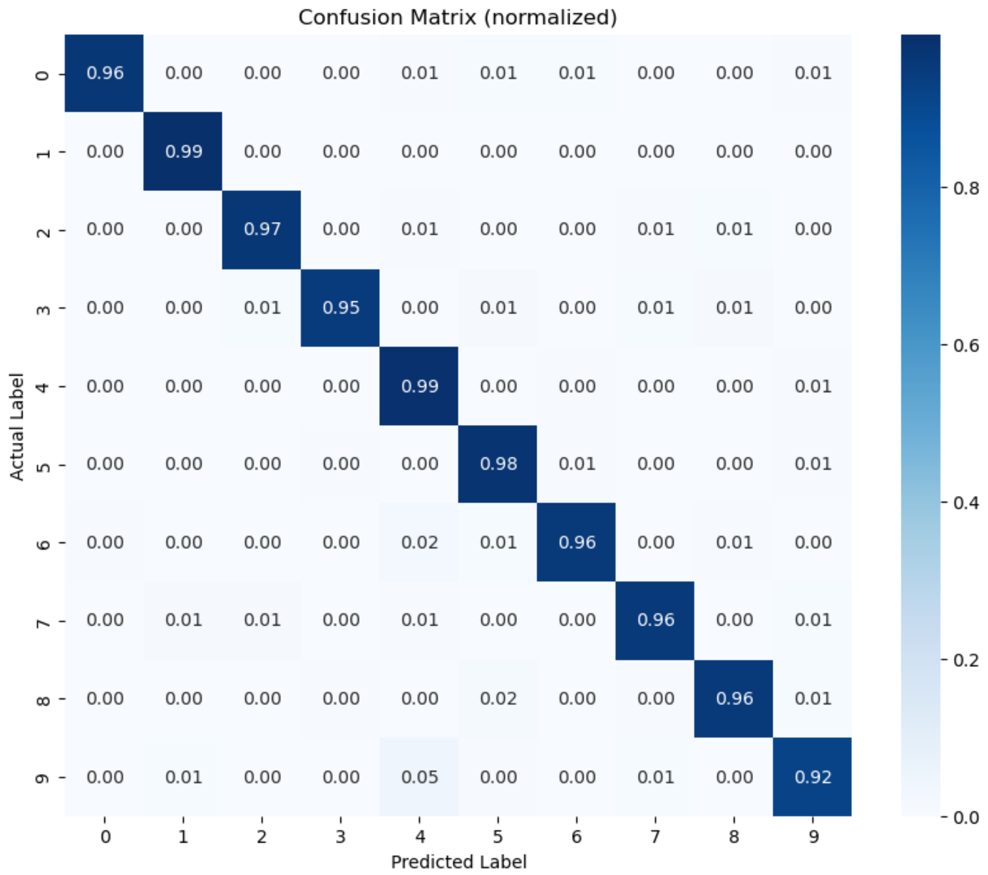
(c) (10%) Please build a model and set training parameters to make the model accuracy higher than 96% on the test set.

```
Accuracy of the network on the test images: 96.33%
```

(d) (10%) Please plot training history.



(e) (10%) Please plot confusion matrix on the test set.



	0	1	2	3	4	5	6	7	8	9
0	0.96	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.01
1	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.97	0.00	0.01	0.00	0.00	0.01	0.01	0.00
3	0.00	0.00	0.01	0.95	0.00	0.01	0.00	0.01	0.01	0.00
4	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.01
5	0.00	0.00	0.00	0.00	0.00	0.98	0.01	0.00	0.00	0.01
6	0.00	0.00	0.00	0.00	0.02	0.01	0.96	0.00	0.01	0.00
7	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.96	0.00	0.01
8	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.96	0.01
9	0.00	0.01	0.00	0.00	0.05	0.00	0.00	0.01	0.00	0.92