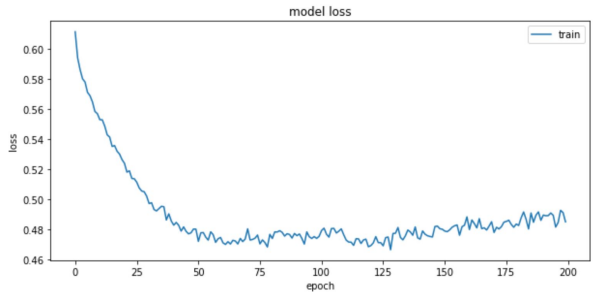
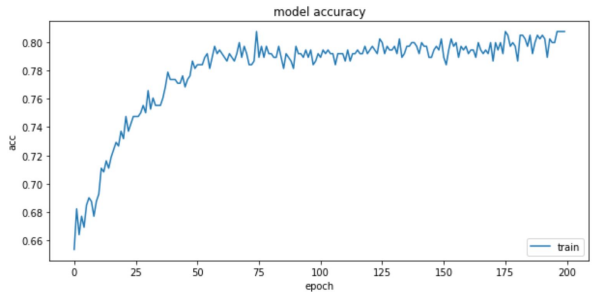
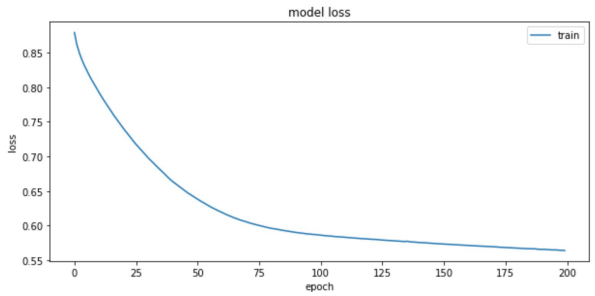
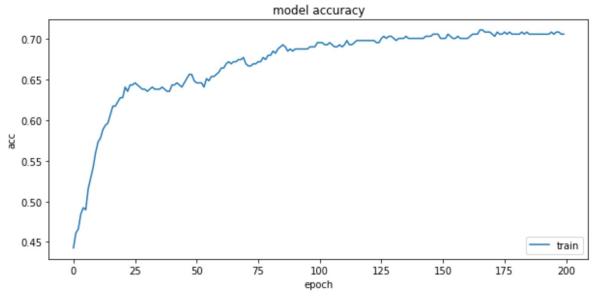
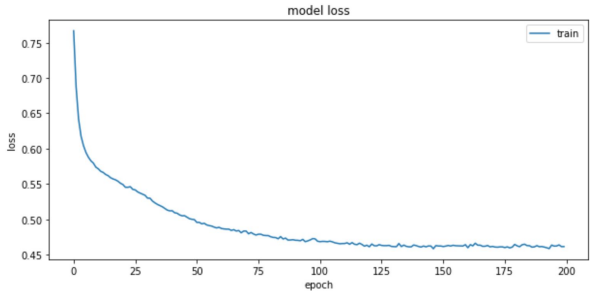
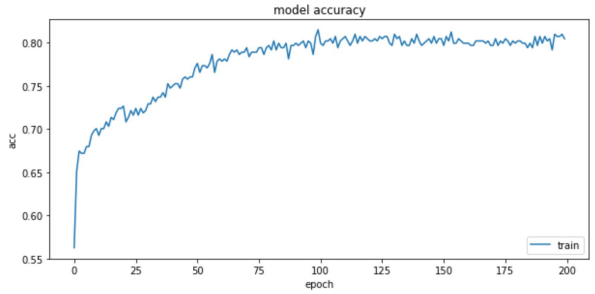
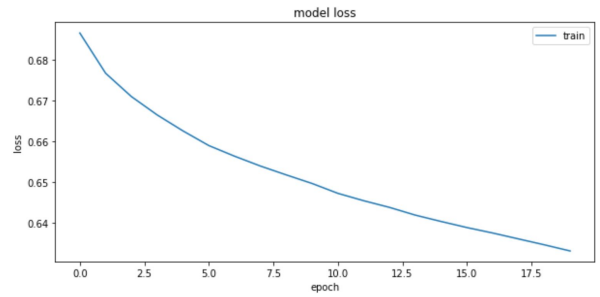
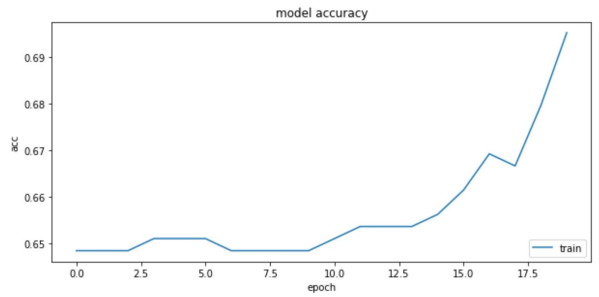
Update the pcn.py using Keras MLP model, and do the following tasks:

1.1 Demonstrate your Pima Results Here (Cycles, Accuracy)

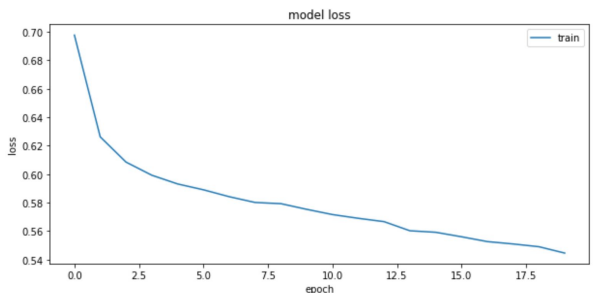
Epoch:200, Batch:1 (best acc:0.8, time=215.589s)	
	

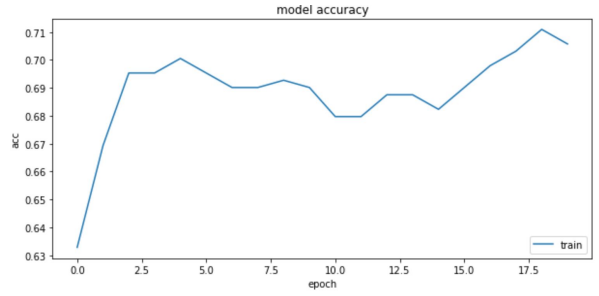
Epoch:200, Batch:256 (best acc:0.7, time=3.271s)	
	
Epoch:200, Batch:2 (best acc:0.8, time=109.814s)	
	
Epoch:20, Batch:256 (best acc:0.69, time=1.065s)	





Epoch:20, Batch:2 (best acc:0.71, time=11.57s)

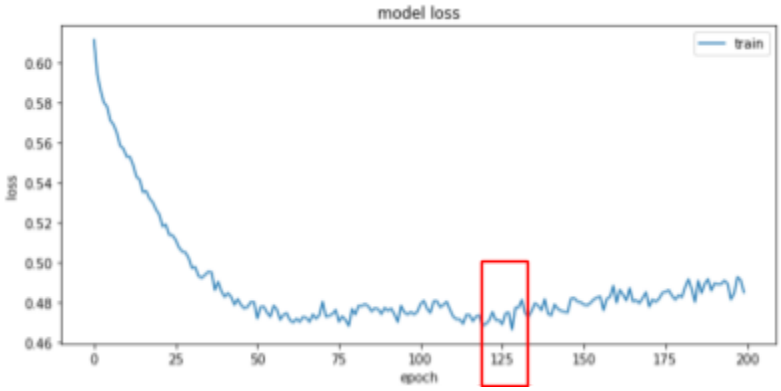




1.2 Compare the batch training and sequential (single-instance) training.

由1.1結果可知，若batch愈小、epoch愈多時，訓練的準確度會逐漸提升，但所花的時間會愈多，因batch是把原始資料拆成很多個大小為n的batch，由於此pima資料有768筆資料，當batch的大小太大時，每一次的梯度更新會更新太多導致準確率較低。

選擇正確的epoch也是相當重要的，下圖可看出在epoch 125時，loss不斷上升，發生over-fitting的情況，因此下次訓練應該改epoch為125較恰當。

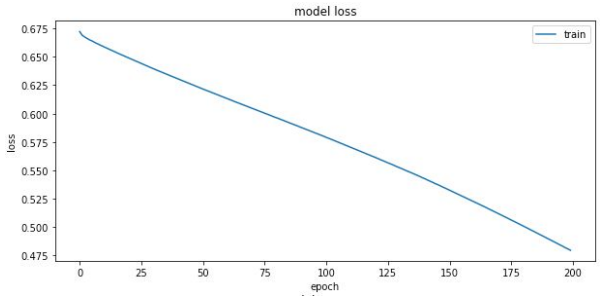


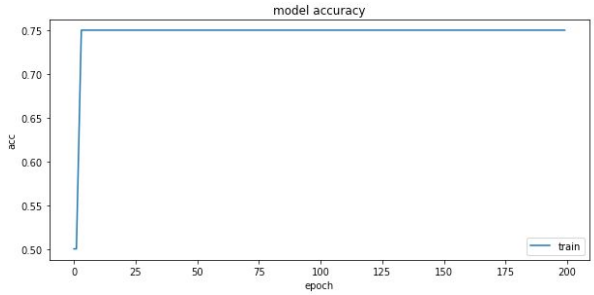
參考資料1 參考資料2

2. Run the pcn.py to the learning self-mapping problem (as shown in the following picture), and find the best result you can.

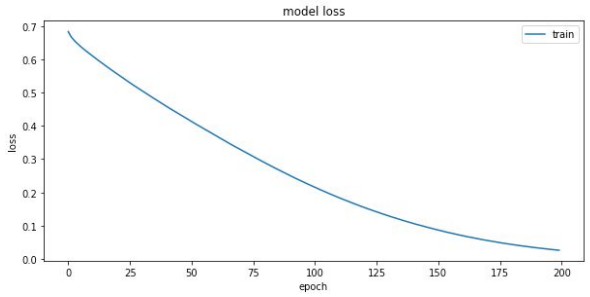
以下epoch:200、batch:1

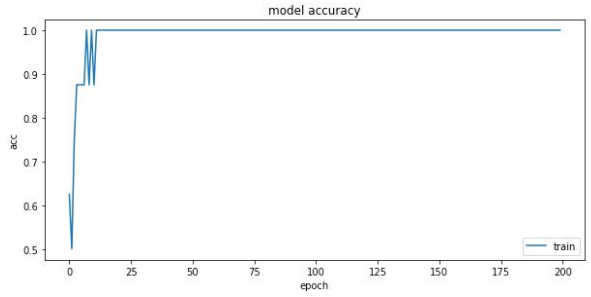
神經元層數：1(10個)
Loss:0.4795, Acc:0.75, time:3.491
Dense
Activation(relu)



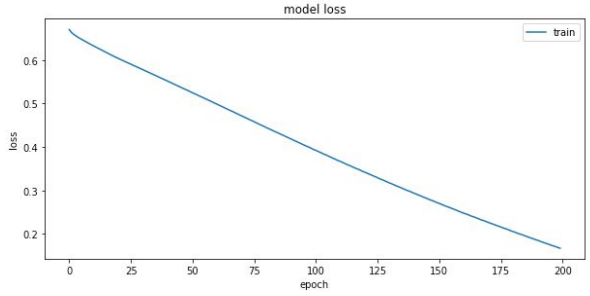


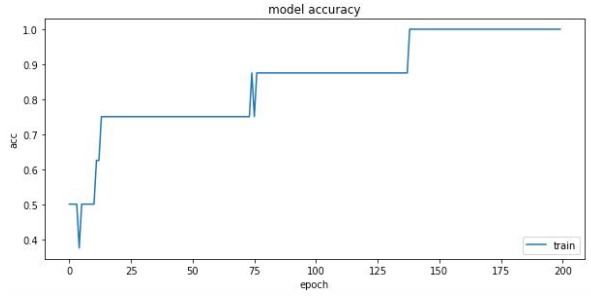
神經元層數：1(100個) 最佳
Loss:0.0256, Acc:1.0, time:3.550
Dense
Activation(relu)



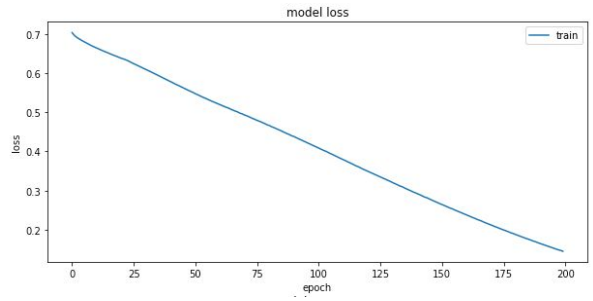


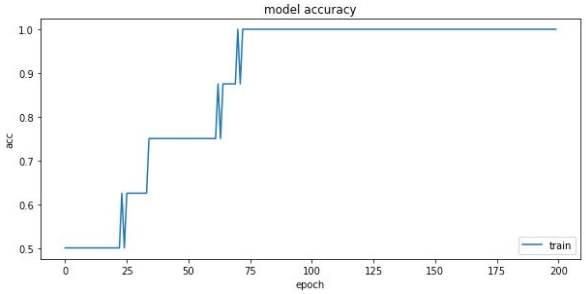
神經元層數：2(10個, 10個)
Loss:0.167, Acc:1.0, time:3.899
Dense
Activation(relu)
Dense



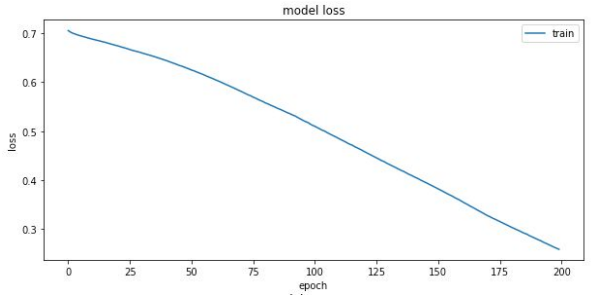


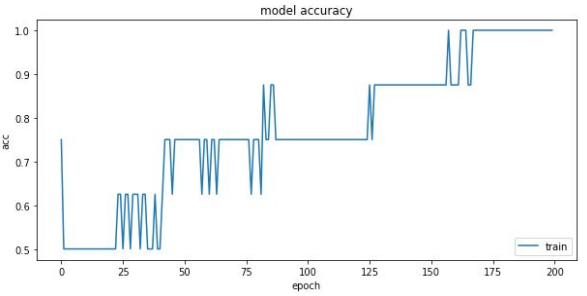
神經元層數：2(10個, 10個)
Loss:0.143, Acc:1.0, time:3.801
Dense
Activation(relu)
Dense
Activation(relu)





神經元層數：3(10個, 10個, 10個)
Loss:0.2591, Acc:1.0, time:3.801
Dense
Activation(relu)
Dense
Activation(relu)
Dense
Activation(relu)





比較可發現，我們使用單層（100個神經元）的訓練loss最低，所需的時間也較少。模型深度太深除了訓練較久，準確率也沒有比單層多個神經元高。

3. Run the pcn.py to the learning OR mapping problem (as shown in the following picture), choose the parameters (e.g. epochs) to find the best result you can.

x1 x2 Output

---

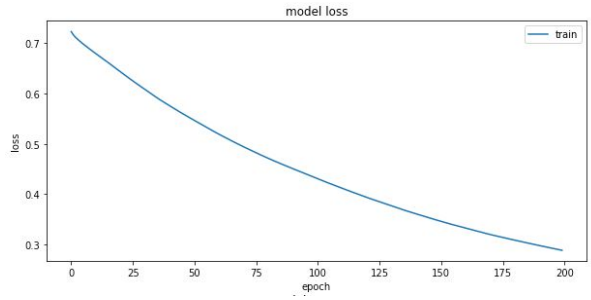
0 0 0

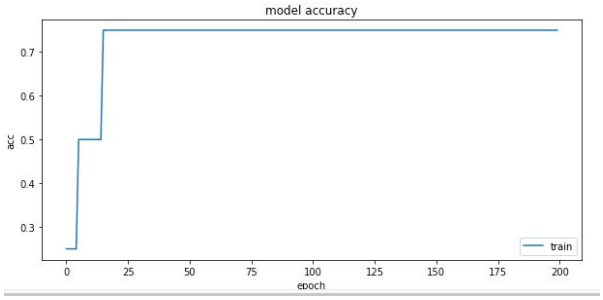
0 1 1

1 0 1

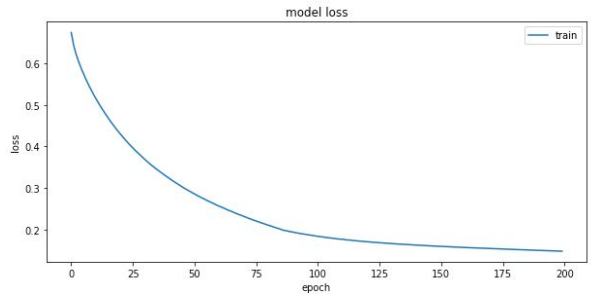
1 1 1

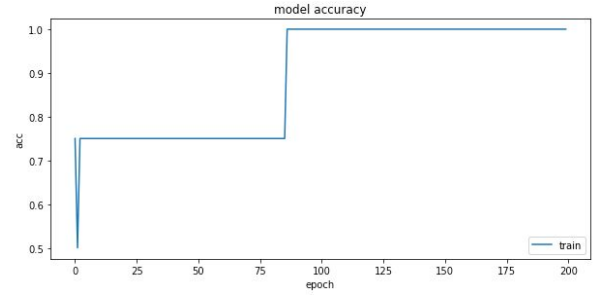
神經元層數：1(10個)
Loss:0.2885, Acc:0.75, time:11.50
Dense
Activation(relu)



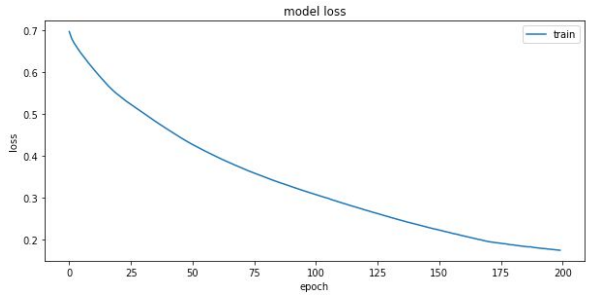


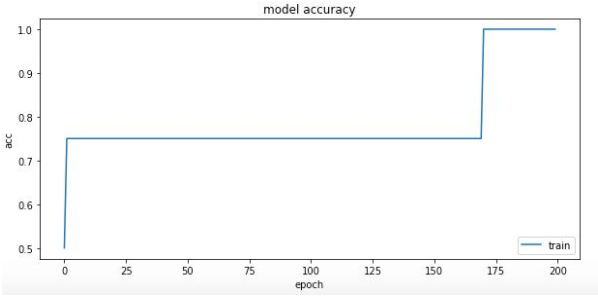
神經元層數：1(100個) 最佳
Loss:0.149, Acc:1.0, time:3.45
Dense
Activation(relu)





神經元層數：2(10個,10個)
Loss:0.175, Acc:1.0, time:3.65
Dense
Activation(relu)
Dense
Activation(relu)





最佳也是和上一題一樣。

4. Run the pcn.py to the learning XOR mapping problem (as shown in the following picture), choose the parameters (e.g. epochs) to find the best result you can.

x1 x2 Output

---

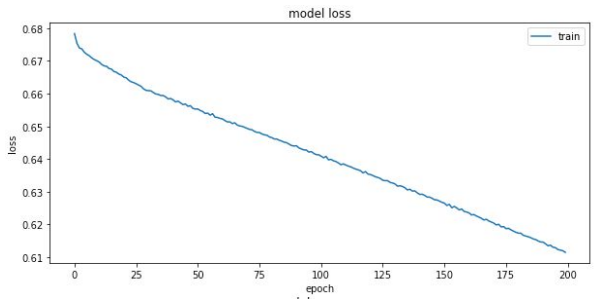
0 0 0

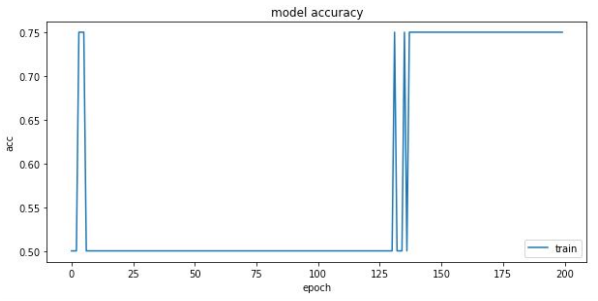
0 1 1

1 0 1

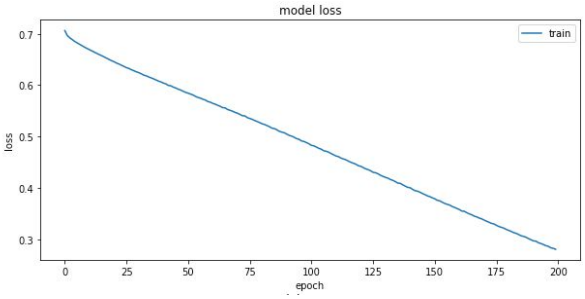
1 1 0

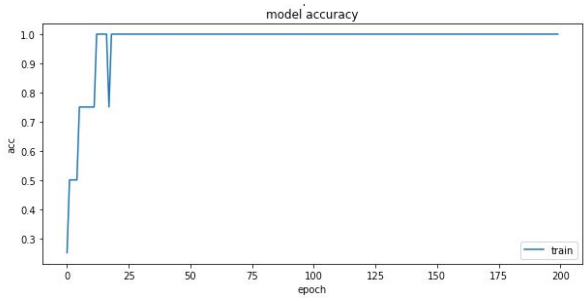
神經元層數：1(10個)
Loss:0.6114, Acc:0.750, time:3.65
Dense
Activation(relu)



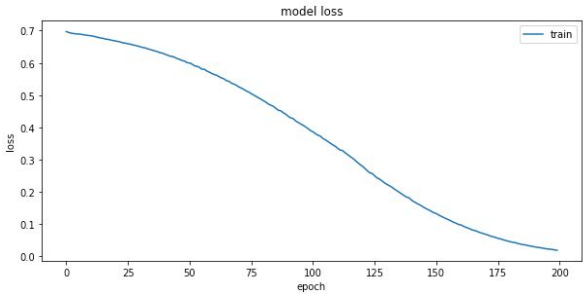


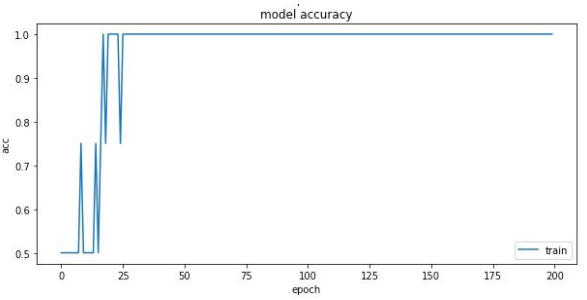
神經元層數：1(100個)
Loss:0.280, Acc:1.0, time:3.745
Dense
Activation(relu)





神經元層數：4(10個,10個,10個) 最佳
Loss:0.0188, Acc:1.0, time:4.612
Dense
Activation(relu)
Dense
Activation(relu)
Dense
Activation(relu)
Dense
Activation(relu)





由上可知，遇到XOR問題時，是不可分割的問題，當單層增加神經元數，但loss還是>0.1，當把模型改深一點時，loss接近0.01，因此模型愈深，在XOR分類中可能導致模型準確率較高。