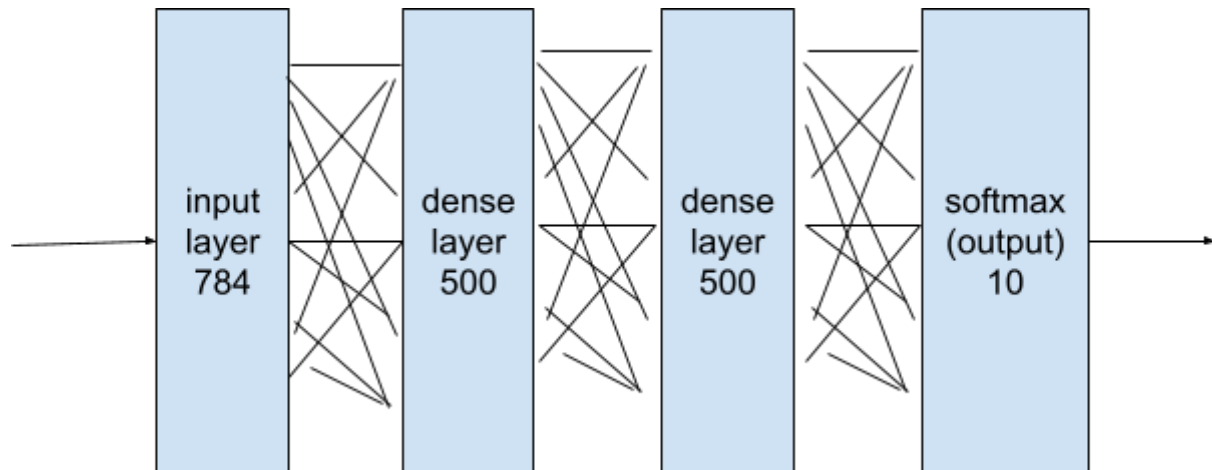


1. Show the model architecture, implementation detail, and testing accuracy. Please describe the methods to achieve the final result in detail, e.g., epochs, batch size, etc. (10%)

- Model architecture:

input layer有784個neurons, 中間有2層dense layer各有500個neurons, 最後經過softmax後predict出屬於10個class中的哪個機率較高



- Implementation:

主要functions:

- one_hot_encode: 將label的一個數字轉換成一個list, 並將x所屬的class給1, 其餘給0. ex. label = 5 \rightarrow [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
- activation: 根據ppt定義relu、tanh、sigmoid的運算公式
- activation_d: 根據ppt定義relu、tanh、sigmoid微分的運算公式
- softmax: 根據ppt定義softmax計算公式

$$\text{Softmax activation: } \Phi(v_1 \dots v_k) = \frac{1}{\sum_{i=1}^k \exp(v_i)} [\exp(v_1) \dots \exp(v_k)]$$

- cross_entropy_loss: 為了怕log裡的值=0, 因此有加一個epsilon進去

$$H = \sum_{c=1}^C \sum_{i=1}^n -y_{c,i} \log_2(p_{c,i})$$

- forward_pass: 回傳從input經過每層layer的weight跟bias的運算後, 再經過activation function的階段性結果
- back_pass: 根據forward pass回傳的結果, 利用chain rule從output layer一直往前去計算loss對每層layer的weight跟bias的微分值, 以便後面來更新weight跟bias
- train: 每次會是一個batch進來, y需先做one_hot_encode, batch中的每筆data都會先經過forward pass再經過back pass來完成一個完整的backpropagation, 最後每筆資料會得到loss對每層weight跟bias的微分值, 我把同一個batch的weight跟bias微分值加起來做平均後, 再搭配learning rate來更新weight跟bias
- test: 單純的把想測試的x和y丟進來此function, x做forward pass即可得到predict結果, 再跟one_hot_encode過的y比較並計算準確率(accuracy)
- save_model: 將neuron個數、activation function type、learning rate、weight 跟 bias 寫入file中, 以便未來load這個train好的model來用

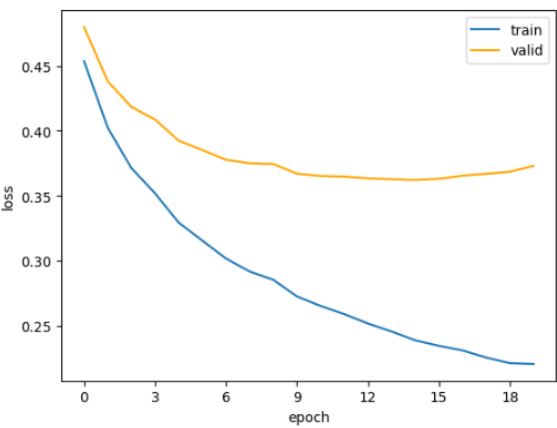
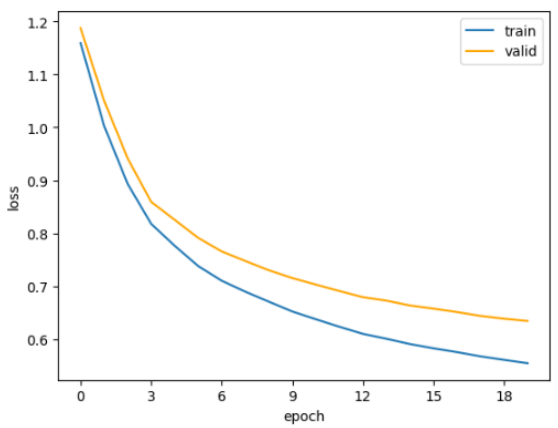
- k) load_model: 將上述model相關的info讀出來, 重建model
- l) random_shuffle: 將training data的順序打亂、在之後取batch去train時會達到 mini_batch的效果
- Methods:
 - 一開始會讀取train_image跟train_label, 接著我先將它們以8:2的比例切分成 training set跟validation set, 也會讀取t10k_image跟t10k_label作為testing set , 接著會將training set做random_shuffle以達到後來取一定batch數在train的時候有隨機mini_batch的概念。接著, 新增一個HW1()的class, 接著用settings() 去對model做設定, 例: learning rate、activation_type、neuron數...等, 特別提到 weight跟bias, 在這裡我是用 $\sqrt{2/\text{neuron個數}}$ 去做初始化。設定完之後便開始用training set去train model, train的步驟已在上面介紹function時提到, 我設定epoch為20, batch size為20, training set會被分成多個batch去做weight和 bias的更新, 當training set全部被輪過一次後才算是一個epoch, 經過20個 epoch後, 會利用save_model這個function把model相關資訊寫入file, 然後用 validation set跟testing set去得出此model的accuracy。
- Testing accuracy: 88.17%

2. During testing, the model might overfit training data to achieve poor performance. How do you check overfitting and underfitting during training? Please describe the methods and experiments to prove that. (10%)

如何檢查overfitting 和 underfitting ?

我會看 validation set 跟 training set 在 train model 所產生的 loss的變化。

- 1) Overfitting: training set 的 loss 一直有在持續下降, 但validation set卻降到一半後就沒有在持續下降
 在overfitting的部分我的settings有:
 - 初始化 weight 跟 bias為 $\sqrt{2 / \text{neurons}}$
 - model layer: input, dense, dense, dense, output
 - 每層神經元個數: 784, 128, 128, 128, 10
 - activation function: relu
 - epoch: 20
 - batch: 20
- 2) Underfitting: training set 跟 validation set 的 loss 下降趨勢差不多, 但是中間仍有一段 gap, 且loss值整體偏大
 在underfitting的部分我大部分的settings跟上述的一樣, 只是未初始化 weight 跟 bias (全部設0)

Overfitting	Underfitting
	
有初始化weight跟bias	未初始化weight跟bias (全部設0)

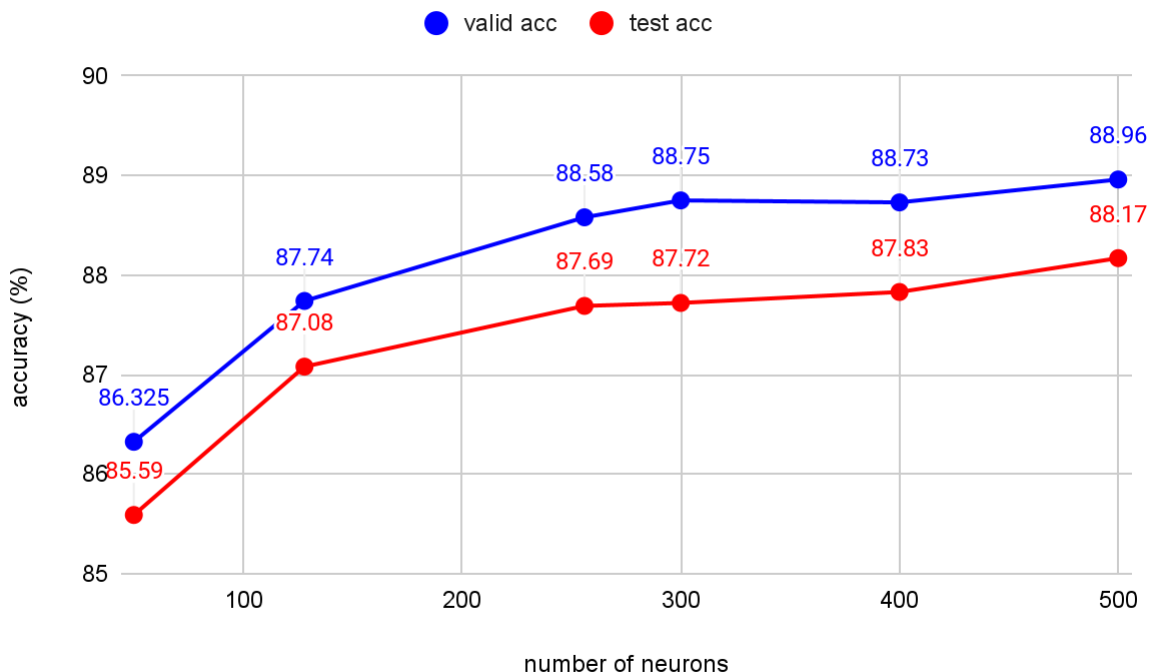
3. During training, you need to adjust the hyperparameters (epochs, batch size, ...) to achieve higher accuracy. How do you choose these hyperparameters? Please conduct experiments to show that these hyperparameters are suitable to achieve higher accuracy for the final test results. (10%)

- 1) Epochs: epoch太大的話可能會發生overfitting, 且並不是epoch越大, accuracy越高, 因為在實驗時有發現當epoch到達某個值時accuracy並沒有顯著的提升, 因此覺得沒有必要再繼續後面的epoch, 反而有可能overfitting, 因此就訂了epoch為20
- 2) Batch size: batch size過小, 花費時間多, 同時梯度震盪嚴重, 不利於收斂; batch size過大, 不同batch的梯度方向沒有任何變化, 容易陷入局部極小值。因此在經過多次嘗試後, 決定batch size為20。
- 3) Learning rate: model學習的速度, 我在一開始給他0.0001, 但是發現學習的太慢 accuracy沒有顯著的提升, 因此慢慢的x10去取一個適當的learning rate, 使model學習的速度適當以調到適合的參數, 因此最後取0.01當作learning rate。
- 4) Neurons: 起初在自己調整參數時是決定固定128這個neuron數並去嘗試要用多少層dense layer, 最後決定為3層dense layer, 有得到87.42%的testing accuracy, 後來在做report時用2層dense layer並分別用50, 128, 256, 300, 400, 500的neuron數去做測試, 用500去訓練時在testing accuracy得到了88.17%的accuracy。
- 5) Weight & bias initialization: 起初設全部為0, 發現model的accuracy一直停滯於76%上下, 後來翻了ppt後將它們初始化成 $\sqrt{2/\text{neurons}}$ 後, accuracy就可以上升到80%了。

4. If we use a very deep NN with a large number of neurons, will the accuracy increase? Why or why not? Please conduct experiments to prove that. (10%)

Settings:

初始化 weight 跟 bias 為 $\sqrt{2 / n}$ (n 為 number of neurons)
model layer: input, dense, dense, output
每層神經元個數: 784, n, n, 10 (n 為 number of neurons)
activation function: relu
epoch: 20
batch: 20



以實驗結果來說，accuracy不一定會隨著neuron個數增加而上升，就以neuron個數等於300和400時在validation set的accuracy來說，反而還下降了0.02%，我覺得是因為neuron個數越多可能會導致overfitting而導致accuracy降低。

5. Please do experiments to compare at least other two activation functions in the model. Analyze the activation functions you used and show if these activation functions help improve the performance during testing. (10%)

Settings:

初始化 weight 跟 bias為 $\sqrt{2 / \text{neurons}}$

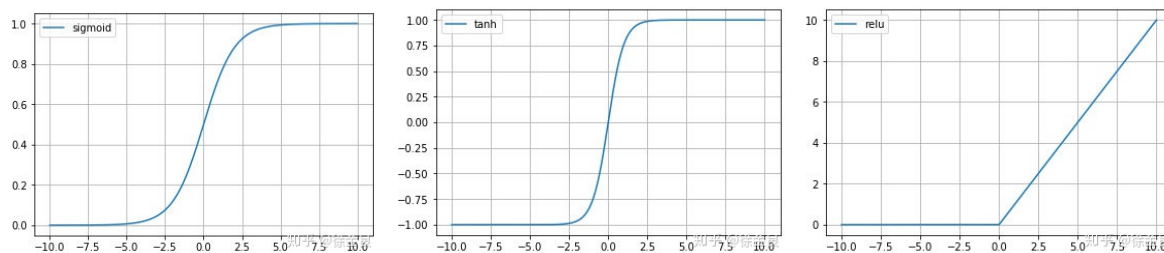
model layer: input, dense, dense, output

每層神經元個數: 784, 128, 128, 10

epoch: 20

batch: 20

	valid acc	test acc
relu	87.86	86.86
sigmoid	86	84.94
tanh	87.94	86.97



以實驗結果來說，sigmoid會導致整體的accuracy下降，包括validation accuracy跟testing accuracy，而tanh有使兩個accuracy稍稍上升。但sigmoid和tanh都會有梯度爆炸和梯度消失問題，因此在訓練時大部分還是都用relu居多。