深度學習

HW6

學號 : B103012002

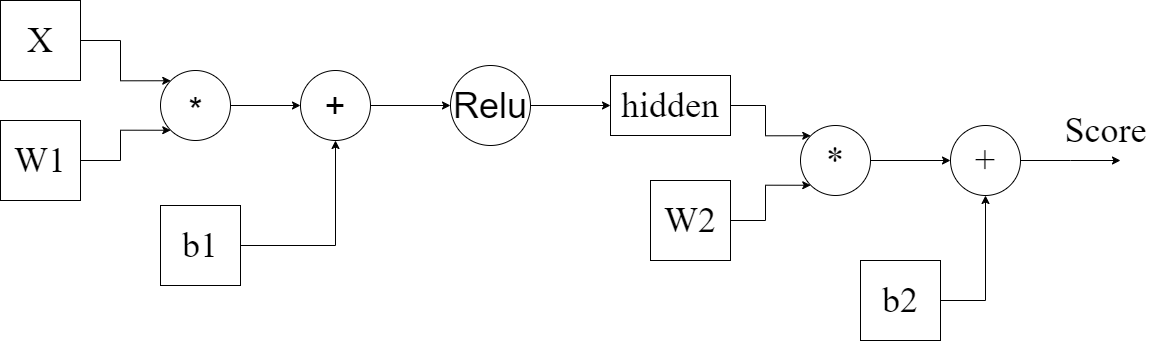
姓名 : 林凡皓

為了方便討論，因此將上次作業內容保留，並在後面加上這次作業內容。本次作業內容從 四、testing our NN on real dataset: CIFAR-10 開始。

1. Forward pass

* nn\_forward\_pass :

1. 解題思路 :

本題主要是計算神經網路對於每個類別計算出來的分數，整個神經網路的forward path如下圖

根據上圖，先利用torch.clamp( )來計算經過relu後的hidden。接著透過torch.mm( )來計算最後的score。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

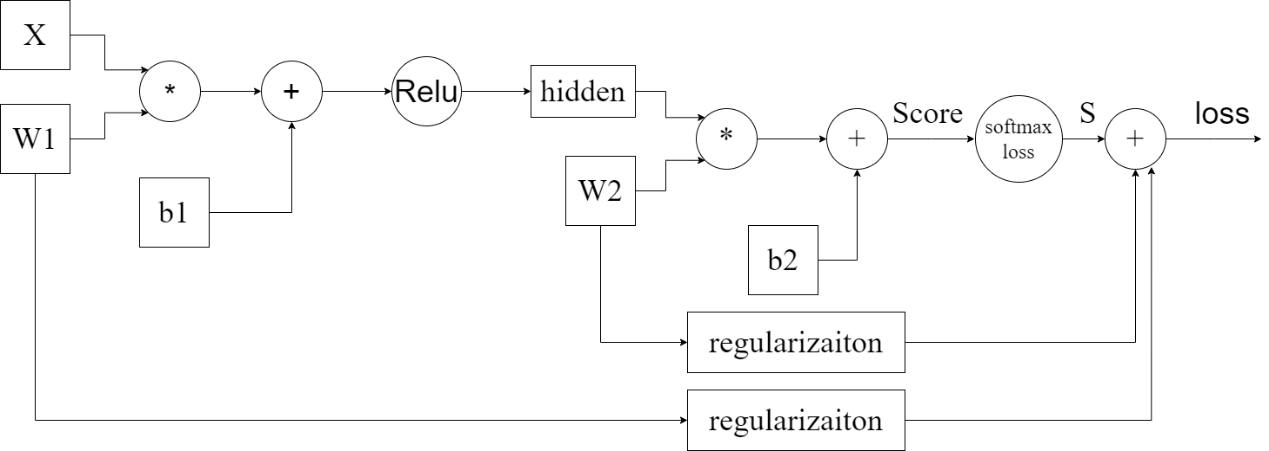
1. 額外討論 :

經過relu後，小於0的數字會被歸零，因此使用torch.clamp( )可以很好的去達到這樣的效果。torch.clamp( )的主要功能為限制tensor的範圍，使用的語法為torch.clamp(input, min, max)，其中min為tensor的最小值，max為tensor的最大值。

* nn\_forward\_backward :

1. 解題思路 :

本題主要是利用forward propagation的方式來計算loss。延伸nn\_forward\_path時的神經網路架構，如下圖



根據softmax loss的定義，如下圖，來完成一個function。

一張含有 字型, 文字, 白色, 符號 的圖片

自動產生的描述 一張含有 字型, 印刷術, 白色, 書法 的圖片

自動產生的描述

在這題，由於網路架構有兩層，因此會有兩個權重值tensor分別為W1和W2，因此regularization term會有兩項分別為R(W1)和R(W2)。

1. 執行結果 :

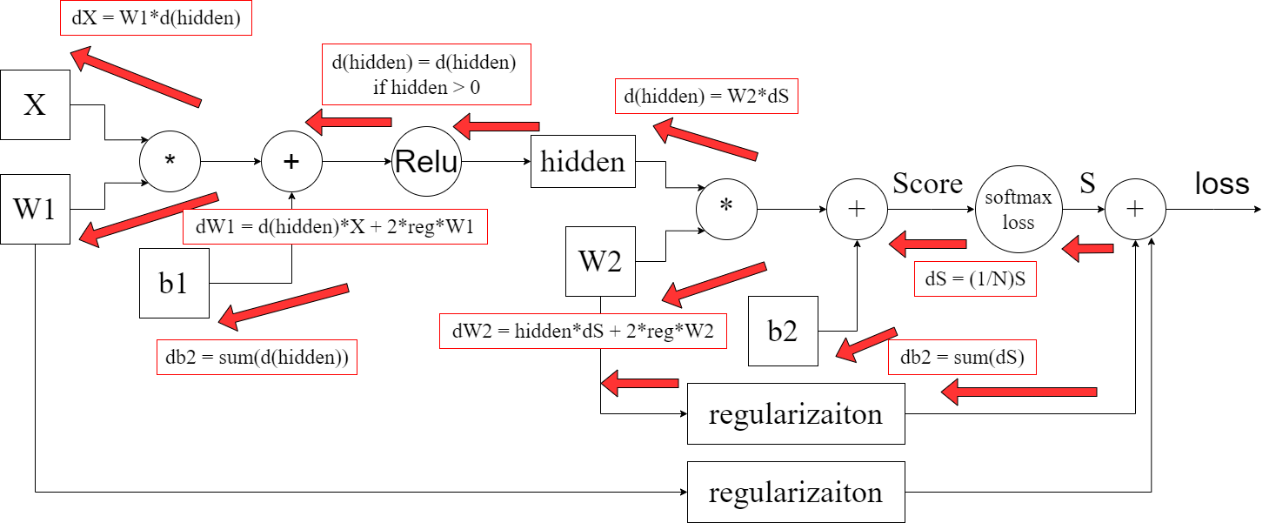
一張含有 文字, 字型, 白色 的圖片

自動產生的描述

1. Backward pass

* nn\_forward\_backward :

1. 解題思路 :

使用computation graph來實作back propagation，computational graph與計算之gradient如下

利用此計算結果加上矩陣相乘的規則(前面矩陣的column數目要等於後面矩陣的row數目)來判斷哪個矩陣要放前面，或是誰需要做轉置，來完成back propagation。

1. 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

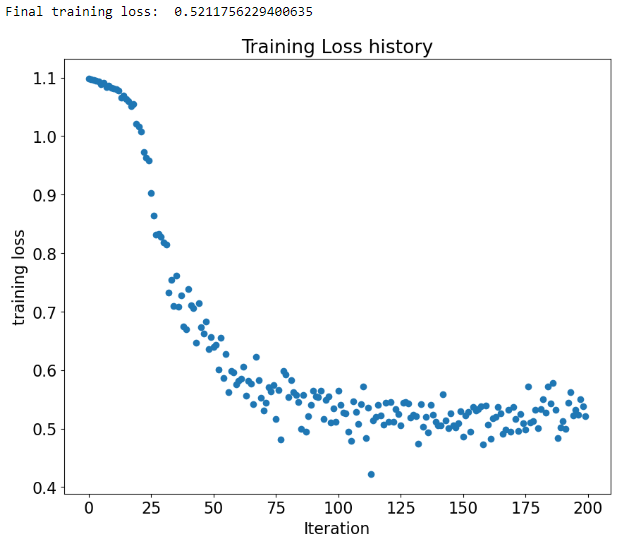
1. Train the network

* nn\_train :

1. 解題思路 :

Trianing的主要在做的事就是更新權重值。更新權重的方式是gradient descent，也就是往負的gradient方向去走。因此，寫法就是w = w – learning\_rate \* dw。

1. 執行結果 :



由上圖可以看出，因為gradient descent的影響，training loss會隨著訓練的次數增加而下降，直到最後趨近於飽和，代表說模型的極限已到。

* nn\_predict :

1. 解題思路 :

nn\_predict是用來當作nn\_train的pred\_func參數。將我們選擇好的參數、training data送進我們選擇的計算分數的function中，接著在計算的分數中找到最大值的index即為預測結果。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

由上圖可以看出準確度隨訓練次數增加而上升，最後趨近飽和，趨勢有點類似於loss。

1. 額外討論 :

為甚麼loss function可以用來計算分數?

關於這個問題，先看到nn\_predict傳入的loss\_func。這裡的loss\_func使用的是先前定義好的nn\_forward\_backward。在nn\_forward\_backward中可以看到有一段code是使用nn\_forward\_pass來計算分數，並在y = None時，nn\_forward\_backward會回傳score而不是loss和grad。接著看到定義nn\_pridect的地方。在定義nn\_predict的時候，我們並沒有傳入y，也就是說y是None，因此nn\_forward\_backward，就會變成是一個計算分數的函數。

1. Testing our NN on a real dataset: CIFAR-10

* Train a network :

根據之前定義的TwoLayerNet來對CIFAR-10做訓練。這裡建立一個有一層hidden layer (size = 36)的NN，接著對它做訓練，訓練的超參數如下 : num\_iters = 500、batch\_size = 1000、learning rate = 1e-2、learning\_rate\_decay = 0.95、reg = 0.25。訓練結果如下

一張含有 文字, 字型, 白色, 螢幕擷取畫面 的圖片

自動產生的描述

由結果可以看出，訓練基本上無效。接著下來討論原因。

* Debug and training :

為了方便觀察，將訓練過程視覺化，視覺化結果如下

一張含有 文字, 行, 繪圖, 字型 的圖片

自動產生的描述

將權重也視覺化，結果如下

一張含有 建築, 樣式, 正方形, Rectangle 的圖片

自動產生的描述

由訓練過程可以看到，validation accuracy和training accuracy基本上沒有提升，代表說現在模型處於underfitting的情況，因此我們嘗試一些方法，像是增加hidden layer size、增加訓練次數、調整learning rate等。

* What’s wrong

開始嘗試各種分析來找到目前模型的問題所在。

1. Hidden layer size :

首先嘗試不同大小的hidden layer。當hidden layer太小，模型會沒辦法很好的擬和training data，造成underfitting，因此嘗試不同的hidden layer size來觀察hidden layer size對於訓練的影響。結果如下

一張含有 文字, 行, 繪圖, 圖表 的圖片

自動產生的描述

由結果可以看出，當hidden layer size提升時，模型能夠有更好的validation accuracy，這主要是因為更大的hidden會有更多的參數可以做調整來更好的去擬合訓練資料。因此先前的模型訓練會underfit的原因可能為hidden layer size太小的關係。

1. Regularization strength :

接著探討regularization strength對訓練的影響。當regularization strength太大時，也會造成模型underfitting，因此也有必要觀察不同regularization strength對於訓練的影響。結果如下圖

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

由結果可以看到，regularization strength要小於0.001時，模型underfitting的情況才比較不會那麼嚴重。因此剛剛訓練沒有進展也有一部分原因是regularization strength太小。

1. Learning rate :

Learning rate對於訓練來說也有很大的影響，learning rate太大會成gradient descent的過程來回彈跳，如下圖

一張含有 行, 繪圖, 圖表, 字型 的圖片

自動產生的描述

由上圖可以看出，learning rate太大不但會造成loss無法順利下降，甚至會造成loss有不降反升的情形。Learning rate太小的話會造成loss下降過程很慢，或是訓練掉到local minimum等現象。嘗試改變learning rate，結果如下

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

由結果可以看出，最佳的learning rate為1。

1. 接著使用剛剛測試出來的參數再次訓練。使用參數如下 : num\_iter = 3000、hidden layer size = 128、regularization strength = 0、learning rate = 1。結果如下

一張含有 文字, 黑與白, 字型, 圖書 的圖片

自動產生的描述

可以看到經過一些參數的調整，模型準確度從原本的9.77 %提升到53.18 %。

* Tune your hyper parameters :

接著使用iteration的方式來尋找最佳的參數組合。

這裡會使用到find\_best\_net這個function。這個function的實現主要就是從get\_param\_set中取得learning rate、hidden\_sizes、regularization strengths、learning rate decays，並利用for loop分別去迭代這些參數，並在訓練完成後判斷目前的模型的validation accuracy是否有高於先前最佳的模型，如果有就更新最佳模型以及最佳的validation accuracy。執行結果如下

一張含有 文字, 圖表, 繪圖, 行 的圖片

自動產生的描述

當learning rate = 0.8、hidden layer size = 900、regularization strength = 1e-7、learning rate decay = 0.92時，得到最佳的validation accuracy = 55.71 %。接著將此模型的權重視覺化，結果如下

一張含有 樣式, 建築材料, Rectangle, 正方形 的圖片

自動產生的描述

最後將此模型對測試資料做測試，結果如下



最終testing accuracy為55.12 %。

1. 額外討論與嘗試

* NN v.s. SVM :

由這次作業與前幾次作業相比可以發現，這於同樣的資料及來說，一個只有一層hidden layer的NN的準確度可以來到54.19 %，而SVM的準確度大約是40 %左右。之所以一個非常簡單的神經網路的準確度可以高於SVM主要是因為它們的權重與所產生的模板的差異。

下圖為SVM所產生的模板

一張含有 文字, 鮮豔, 螢幕擷取畫面 的圖片

自動產生的描述

對於一個類別來說，SVM只會產生一個模板，這會使SVM無法應對太多種情況。下圖為NN的模板

一張含有 樣式, 建築材料, Rectangle, 正方形 的圖片

自動產生的描述

由此圖可以發現到，對於一個類別，NN會產生出多個對應的模板，這不只使NN可以應對更多種情況，也能使NN有更佳的判斷準確度。

如果去疊加更多層的NN，甚至是使用convolutional layer，神經網路會產生出更多的模板，甚至是可以更好的抓取關鍵特徵，這又會再讓神經網路的準確度上升。但是就如同教授上課所講，要訓練一個好的神經網路會需要大量的資料，如果資料不足，神經網路也沒有辦有效的訓練權重值，導致神經網路的功能無法完全被發揮之外，還要花費一堆時間訓練模型。因此，雖然說SVM或是KNN之類的架構的準確度無法與神經網路做比較，但是在一些情況下，SVM與KNN也會是不錯的選擇。

* Overfitting and underfitting :

Overfitting和underfitting是訓練模型時很重要的兩個指標，在這次訓練過程中，也有發生underfitting和overfitting的現象，因此拿出來做討論。

1. Underfitting : 也有人將其稱為high bias。Underfitting代表說目前的模型，即便是對訓練資料來說，也無法擬合的太好，如下圖所示

一張含有 文字, 行, 字型, 圖表 的圖片

自動產生的描述

underfitting的特色為training accuracy和validation accuracy都會很低。

對於這個現象，增加訓練資料是沒有用的，畢竟連目前的資料都無法有效擬合，更何況是再加入其他資料。對於underfitting，解決方法如下

1. 增加其他特徵 : 如果可以增加其他特徵讓模型去抓取，模型會有更好的機會去根據這些特徵找出資料中的規律，並更加有效的去擬合資料。
2. 讓模型更加複雜 : 不論是在linear classifier中加入一些高次多項式的項式，或是堆疊更深的神經網路，都會很有效的解決掉underfitting的問題，但是成本就是訓練與預測的時間與計算量都會增加。
3. 增加訓練次數。
4. 降低regularization strength。
5. Overfitting : 也有人稱為high variance。Overfitting代表說模型已經過度擬合訓練資料了，換句話說，模型已經成為專門為該訓練資料所設計的模型了，對於沒看過的資料有不好的泛化能力。此現象如下圖所示

一張含有 文字, 行, 圖表, 繪圖 的圖片

自動產生的描述

Overfitting的特色為training accuracy很高，但是validation accuracy沒那麼高，通常testing accuracy和validation accuracy會差不少。對於overfitting，解決方法如下

1. 增加訓練資料 : 假如訓練資瞭可以包含所有可能性，那就沒有overfitting的說法了，畢竟沒有情況是模型沒見過的，因此增加訓練資料可以幫助解決overfitting的問題。
2. 在訓練模型時使用early stopping的機制，讓模型在過度擬合時及時停止訓練。
3. 加大regularization strength。
4. 對於神經網路而言，可以使用dropout或是避免使用全連接層來避免overfitting的情況。

* Transfer learning

1. Transfer learning方法與分類

底下的source定義為已經存在的知識，target定義為欲訓練的領域。

1. Based on instance or said sample : 通過權重的分配，分別作用到source和target。例如，在source中有一個樣本和target中的樣本很像，那我們可以加大刺樣本對應到的權重。
2. Based on features : 將source和target中的特徵變換同一個空間。在source和target上的feature差異很大，我們可以透過將他們的特徵變換到同一個空間，如此一來我們就可以更加方便研究source和target的相關內容和性質。
3. Based on model : 將source的參數與target共享。將pre-trained的model修改一些layer的參數。這次嘗試做的也是based on model的transfer learning。
4. 嘗試

這次選擇使用兩種模型，分別為Xception與ResNet50，來做transfer learning。先對兩個模型作介紹。

1. Xception : 此模型是根據InceptionV3做改進，採用depthwise separable convolution來代替原本的Inception module，並加入了ResNet的Residual Learning方法。InceptionV3的主要概念是讓模型自己去找最好的特徵提取方式，架構如下

一張含有 圖表, 行, 寫生, 方案 的圖片

自動產生的描述

所謂的depthwise separable convolution可以分成兩個步驟，第一步為depthwise convolution，即對各個維度通道各自做卷積運算。第二步是pointwise convoltion，即對不同維度通道進行融合。Xception架構如下

一張含有 文字, 收據, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

1. ResNet50 : ResNet在VGG的基礎上，新增了skip connection。此動作不但部會新增任何參數，還能夠解決梯度消失的問題。ResNet中最關鍵的就是residual learning。Residual learning的是意圖如下圖

一張含有 文字, 圖表, 字型, 行 的圖片

自動產生的描述

上圖的概念就是residual mapping。在ResNet中，將目標H(x)換成F(x) = H(x) – x，即residual。優化residual會比直接學習恆等mapping ( x 🡪 H(x) )來的容易，有了residual mapping之後，模型只需要想辦法讓每個residual block學習出來的residual變成0 (等價於恆等映射H(x) = x)，然後移項就可以得到H(x) = F(x) + x，即我們原本要的mapping函數。

接著展示transfer lerning後的結果，首先是Xception，網路架構如下圖，

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

訓練過程如下

一張含有 文字, 螢幕擷取畫面, 功能表, 黑與白 的圖片

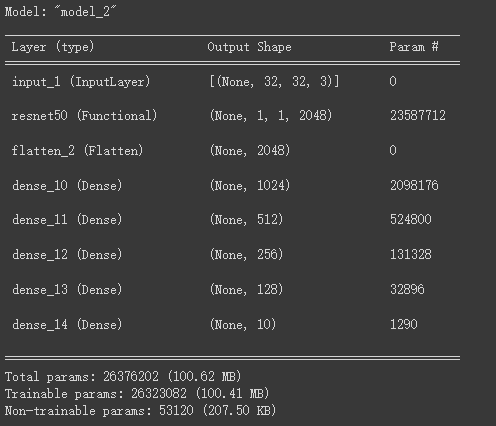
自動產生的描述

一張含有 文字, 螢幕擷取畫面, 繪圖, 行 的圖片

自動產生的描述

可以看到，validation accuracy可以來到86.7 %，比起這次作業的NN高出很多。

接著嘗試做ResNet50的transfer learning。網路架構如下



訓練過程如下

一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

自動產生的描述

由上圖可以看出，訓練到最後validation accuracy為68.5 %，而這時候不只是validation loss無法有效提升，連training loss都無法順利下降。這顯然是不合理的，情況有兩種可能，第一就是發生underfitting的問題，但是我認為不是這種可能，因為我有自己設置callbacks來防止overfitting的情況發生，因此訓練並非是迭代完成而停止，而是因為loss無法再順利下降而停止。第二種可能為原本的ResNet50的訓練資料與CIFAR-10相差很遠，這會造成說這組ResNet的參數是無法有效地應用到CIFAR-10的資料及上。為了解決這個情況，我讓原本的ResNet模型的參數變得可以做訓練，修改後結果如下

一張含有 文字, 螢幕擷取畫面, 功能表 的圖片

自動產生的描述

將訓練過程視覺化，結果如下

一張含有 文字, 螢幕擷取畫面, 繪圖, 行 的圖片

自動產生的描述

由上圖可以看到，最終的validation accuracy大落在78 %左右，而這一次的訓練，training accuracy也能夠順利提升到100 %左右，代表說模型有辦法去擬合CIFAR-10資料集了。

1. Reference

[1] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/>

[2] GreeksforGeeks “ML | Underfitting and Overfitting” <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>

[3] 陳明佐 “Transfer Learning 轉移學習“ <https://medium.com/%E6%88%91%E5%B0%B1%E5%95%8F%E4%B8%80%E5%8F%A5-%E6%80%8E%E9%BA%BC%E5%AF%AB/transfer-learning-%E8%BD%89%E7%A7%BB%E5%AD%B8%E7%BF%92-4538e6e2ffe4>

[4] TIN-WAI PHOEBE “Transfer Learning with Xception for CIFAR-10” <https://www.kaggle.com/code/phoebewongtw/transfer-learning-with-xception-for-cifar-10>

[5] 馨伊的閱讀筆記 “Inception系列 – Xception” <https://medium.com/ching-i/inception-%E7%B3%BB%E5%88%97-xception-fd2a4a4e7e82>

[5] stack overflow “Transfer learning with Keras, validation accuracy does not improve from outset” <https://stackoverflow.com/questions/58676652/transfer-learning-with-keras-validation-accuracy-does-not-improve-from-outset>

[6] Chi Ming Lee “值觀理解ResNet –簡介、觀念及實作(Python Keras)” <https://medium.com/@rossleecooloh/%E7%9B%B4%E8%A7%80%E7%90%86%E8%A7%A3resnet-%E7%B0%A1%E4%BB%8B-%E8%A7%80%E5%BF%B5%E5%8F%8A%E5%AF%A6%E4%BD%9C-python-keras-8d1e2e057de2>