深度學習

HW2

學號 : B103012002

姓名 : 林凡皓

1. **Compute distance : Naïve implementation**

* compute\_distances\_two\_loops :

1. 解題思路 : 先透過x.view( )將輸入的兩個矩陣變成二為矩陣，再透過兩個for loop計算每個訓練資料與測試資料的歐式距離。

2. 執行結果 :

­­一張含有 螢幕擷取畫面, 文字, 太陽能電池 的圖片

自動產生的描述

1. **Compute distances : vectorization**

* compute\_distances\_one\_loop :

1. 解題思路 : 先透過x.view( )將輸入的兩個矩陣變成二為矩陣，透過一個for loop迭代訓練資料以及broadcast來符合測試資料。

2. 執行結果 :



* compute\_distances\_no\_loop :

1. 解題思路 : 先透過x.view( )將輸入的兩個矩陣變成二為矩陣，並利用(x - y)^2 = x^2 + y^2 – dot(x, y)的概念，分別計算出x^2、y^2、dot(x, y)，即可完成本題。

2. 執行結果 :



* Two\_loops v.s. one\_loop v.s. no\_loops :

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

由結果可以看出，一個迴圈會比兩個迴圈快上14倍左右，而不使用迴圈會比一個迴圈快上700倍左右。

1. **Predict labels**

* predict\_labels :

1. 解題思路 : 透過torch.topk( )取得最近的k個鄰居的index，並透過index取得label。接著計算每個label出現的次數並取出現最多的label。

2. 執行結果 :



3. 額外討論 : torch.topk( )傳入三個參數，第一個為dists，第二個為k，第三個是largest，也就是說要在dists這個tensor中找到k個值，如果largest=False，就會找最小值。

* KnnClassifier :

1. 解題思路 : 這一部分要定義一個KnnClassifier的class。一開始要對KnnClassifier做初始化，接著定義predict函數。predict函數只需要先呼compute\_distances\_no\_loops計算出與其他樣本的距離，接著用predict\_labels取得最近的距離即可。

2. 執行結果 :

一張含有 螢幕擷取畫面, 鮮豔 的圖片

自動產生的描述

一張含有 地圖, 螢幕擷取畫面, 鮮豔 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 地圖, 鮮豔 的圖片

自動產生的描述

由以上三張圖可以看出，當k逐漸變大，切割不同類別的邊界會變滑順，且切割出來的區塊也會減少。



由上圖可以得知，這個簡單的Knn model(k = 1)大約有27.4 %的準確度，比一般人類預測的準確度(10 %)還要高。

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

當k=5時，準確度稍微提升一點，但是進步很少。

1. Cross-validation

* knn\_cross\_validate :

1. 解題思路 : 先利用chunk將訓練資料與訓練標籤分割成指定fold數目，並將結果存成list。接著利用for loop走過k\_choices中每一個k值，對於每一個k值，利用for loop讓分割後的訓練資料與訓練標籤中的每一個fold皆有機會當validation set，可以透過每一次都選擇list的最後一個元素，結束驗證後將被選中的元素放到list的第一個位置。

2. 執行結果 :

一張含有 文字, 黑與白, 文件, 樣式 的圖片

自動產生的描述

一張含有 文字, 行, 圖表, 螢幕擷取畫面 的圖片

自動產生的描述

由上圖可以看出，最佳的k大約落在10，左右。

* knn\_get\_best\_k :

1. 解題思路 : 先創建一個dictionary，透過for loop計算上一題k\_to\_accuracies的對於每一個k而言的平均，並將此平均與k值儲存到剛才創建的dictionary。最後透過max( )取得擁有最大平均值的k值。

2. 執行結果 :

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

最佳k值為10，與上一題結果相同。

* 利用訓練好的模型預測整的資料集 :

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

預測後的準確度為33.86 %。

1. **額外嘗試**

* 使用更多樣本訓練knn :

我嘗試使用50000個樣本來做訓練。

1. k = 1 :

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

k = 1時，準確度為35.39 %。

2. k = 5 :



k = 5時，準確度為33.98 %。不管k = 1 or 5，準確度都有些微提升。

3. cross validation :

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

最後得到的準確度為35.39 %，雖然有少量資料訓練出來的模型好，但是提升幅度有限。

* 使用ResNet50 :

由於KNN做出來的準確度大約只有30 %，因此我嘗試使用tensorflow.keras.applications中的ResNet50來訓練。

1. ResNet50 : 採用深度殘差學習的思想，透過引入跳要連接來建構模型，使訓練更加容易。由於ResNet50使用卷積層來提取圖像特徵，因此我認為將它用在本次作業是不錯的選擇。

2. 訓練結果 :

一張含有 文字, 行, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

最終validation accuracy大約停留在68 %左右便無法再上升，此準確度比KNN高出許多。

* 使用CNN :

CNN為影像辨識常用的深度學習架構，因此本次作業我也嘗試建立一個簡單的CNN來做比較。

CNN架構如下 :

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

訓練結果如下 :

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述

最高的準確度為74.03 %，為目前最高的模型。

* 心得 :

由這次的簡單嘗試，我得到一下幾點結論 :

1. 增加訓練資料可以增加模型的準確度，但是增加幅度有限，如果要求更高的準確度，需要對模型做進一步的優化或是使用其他更強大的模型。

2. 比較大的模型雖然可以有效提升準確度，但訓練時間會比較長，以這次的嘗試來說，KNN只需要花幾秒鐘便可以訓練完成，而CNN卻需要花上30分鐘才能夠訓練完成，因此我認為選擇模型不只是要考慮到準確度，還要針對自己的需求在各個方面做取捨。

1. Reference :

[1] Carlos Andres Polania “Transfer learning with ResNet50 from keras and CIFAR-10” <https://www.linkedin.com/pulse/transfer-learning-resnet50-from-keras-cifar-10-carlos-andres-polania/?trk=article-ssr-frontend-pulse_more-articles_related-content-card>

[2] Devashree Madhugiri “Using CNN for Image Classification on CIFAR-10 Dataset” <https://devashree-madhugiri.medium.com/using-cnn-for-image-classification-on-cifar-10-dataset-7803d9f3b983>

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