Machine Learning - Homework 3

資工四 B05902023 李澤諺

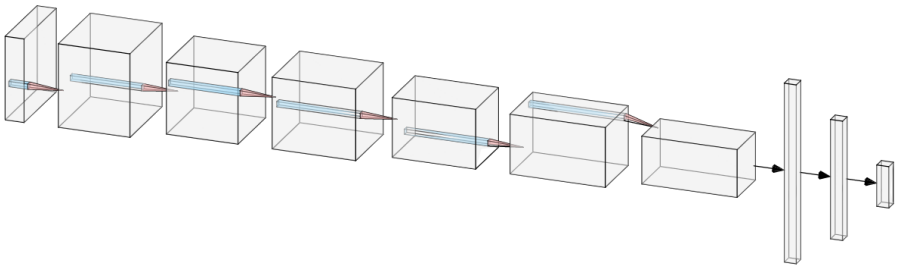
April 30, 2020

1. (1%) 請說明你實作的 CNN 模型，其模型架構、訓練參數量和準確率爲何?

我先將所有照片的大小皆調整爲 128×128，再將所有照片中 pixel 的值皆除以 255，normalize 到 0 和 1 之間，以此進行 data preprocessing，並且，我使用 torchvision 進行了以下的 data augmentation：

RandomAffine(15 , translate = (0.1 , 0.1) , scale = (0.9 , 1.1))
RandomHorizontalFlip()

接著，我參考了 VGG 的架構，使用 PyTorch 實作出了以下的 CNN：



Conv2d(3 , 64 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(64)
ReLU()
Conv2d(64 , 64 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(64)
ReLU()
MaxPool2d(2)
Dropout2d(0.1)
Conv2d(64 , 128 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(128)
ReLU()
Conv2d(128 , 128 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(128)
ReLU()
MaxPool2d(2)

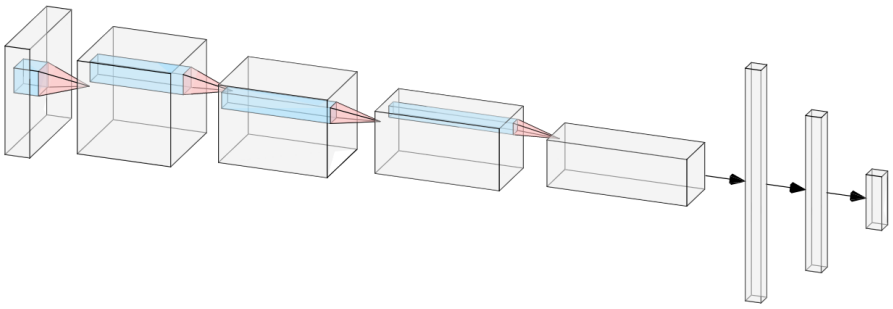
Dropout2d(0.1)
Conv2d(128 , 256 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(256)
ReLU()
Conv2d(256 , 256 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(256)
ReLU()
MaxPool2d(2)
Dropout2d(0.1)
Conv2d(256 , 512 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(512)
ReLU()
Conv2d(512 , 512 , kernel_size = 3 , stride = 1 , padding = 1)
BatchNorm2d(512)
ReLU()
MaxPool2d(2)
Dropout2d(0.5)
Linear(32768 , 1024 , bias = True)
BatchNorm1d(1024)
ReLU()
Dropout(0.5)
Linear(1024 , 11 , bias = True)

根據 torch-summary，以上的 CNN 中總共有 38257995 個 parameter。最後，我使用了 Adam 訓練 CNN，其中 batch size 爲 128，learning rate 爲 0.001，訓練了 150 個 epoch，以此得到最後的 model，其所得到的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.99878	0.82886	0.85176	0.84468

2. (1%) 請實作與第一題接近的參數量，但 CNN 深度 (CNN 層數) 減半的模型，並說明其模型架構、訓練參數量和準確率爲何?

我使用了和第 1 題相同的 data preprocessing 以及 data augmentation，並使用 PyTorch 實作出了以下的 CNN：



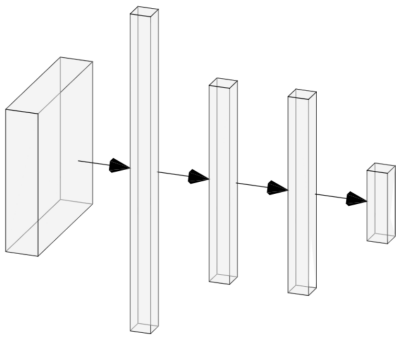
Conv2d(3 , 64 , kernel_size = 11 , stride = 1 , padding = 5)
BatchNorm2d(64)
ReLU()
MaxPool2d(2)
Dropout2d(0.1)
Conv2d(64 , 128 , kernel_size = 9 , stride = 1 , padding = 4)
BatchNorm2d(128)
ReLU()
MaxPool2d(2)
Dropout2d(0.1)
Conv2d(128 , 256 , kernel_size = 7 , stride = 1 , padding = 3)
BatchNorm2d(256)
ReLU()
MaxPool2d(2)
Dropout2d(0.1)
Conv2d(256 , 512 , kernel_size = 5 , stride = 1 , padding = 2)
BatchNorm2d(512)
ReLU()
MaxPool2d(2)
Dropout2d(0.5)
Linear(32768 , 1024 , bias = True)
BatchNorm1d(1024)
ReLU()
Dropout(0.5)
Linear(1024 , 11 , bias = True)

根據 torch-summary，以上的 CNN 中總共有 39140875 個 parameter。接著，我使用了 Adam 訓練 CNN，其中 batch size 爲 128，learning rate 爲 0.001，訓練了 150 個 epoch，以此得到最後的 model，其所得到的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.93361	0.74519	0.77405	0.76523

3. (1%) 請實作與第一題接近的參數量，簡單的 DNN 模型，同時也說明其模型架構、訓練參數和準確率爲何?

我先將所有照片的大小皆調整爲 128×128 之後，再將所有的照片皆變爲 1-dimensional array，以此進行 data preprocessing。接著，我實作出了以下的 DNN：



Linear(49152 , 800 , bias = True)
BatchNorm1d(800)
ReLU()
Dropout(0.5)
Linear(800 , 800 , bias = True)
BatchNorm1d(800)
ReLU()
Dropout(0.5)
Linear(800 , 11 , bias = True)

以上的 DNN 中總共有 39972011 個 parameter。接著，我使用了 Adam 訓練 DNN，其中 batch size 爲 128，learning rate 爲 0.001，訓練了 150 個 epoch，以此得到最後的 model，其所得到的 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.93331	0.32070	0.37298	0.37753

4. (1%) 請說明由 1 到 3 題的實驗中你觀察到了什麼?

由第 1 題到第 3 題可以看出，DNN 的 performance 比 CNN 還要差了許多，其大致是因爲 CNN 會使用 filter 對照片進行 convolution 運算，以找出照片中潛在的 pattern 作爲其 feature，並以此進行 classification，其爲 CNN 的強大之處，然而 DNN 沒辦法考慮到照片中潛在的 pattern，僅能對照片進行數值上的運算，以此進行 classification，因此 DNN 的 performance 遠遠不及 CNN。而第 3 題中的 CNN，其 layer 數目比第 1 題中的 CNN 還要少，並且爲了讓兩者的 parameter 數目相近，第 3 題中的 CNN 其 filter 大小皆比第 1 題中的 CNN 還要大，由兩者的 accuracy 可以看出，第 1 題中的 CNN 其 performance 比第 3 題中的 CNN 還要好，其大致是因爲第 1 題中的 CNN 是參考 VGG 的架構所實作出來的，VGG 的論文中指出，使用 filter 尺寸較小的 convolutional layer 重複堆疊，其亦能達到 filter 尺寸較大的 convolutional layer 的效果，甚至還能提高 non-linearity，增加 model 的 complexity，此外，使用 filter 尺寸較小的 convolutional layer，可以使得 CNN 的 layer 數目更多，即 model 變得更 deep，老師的課程中已有提過 deep model 的優點，故在此不再贅述，綜合以上所述，大致爲第 1 題中的 CNN 其 performance

比第 3 題中的 CNN 還要好的原因。(不過根據 ResNet 的論文中指出，當 model 的 layer 數目越多時，可能會使得資料在 model 中 forward 時逐漸遺失資訊，因此必須使用更多的 data augmentation 來彌補這一點，其爲在訓練第 1 題中的 CNN 時必須注意的地方)

5. (1%) 請嘗試 data normalization 及 data augmentation，說明實作方法並且說明實行前後對準確率有什麼樣的影響?

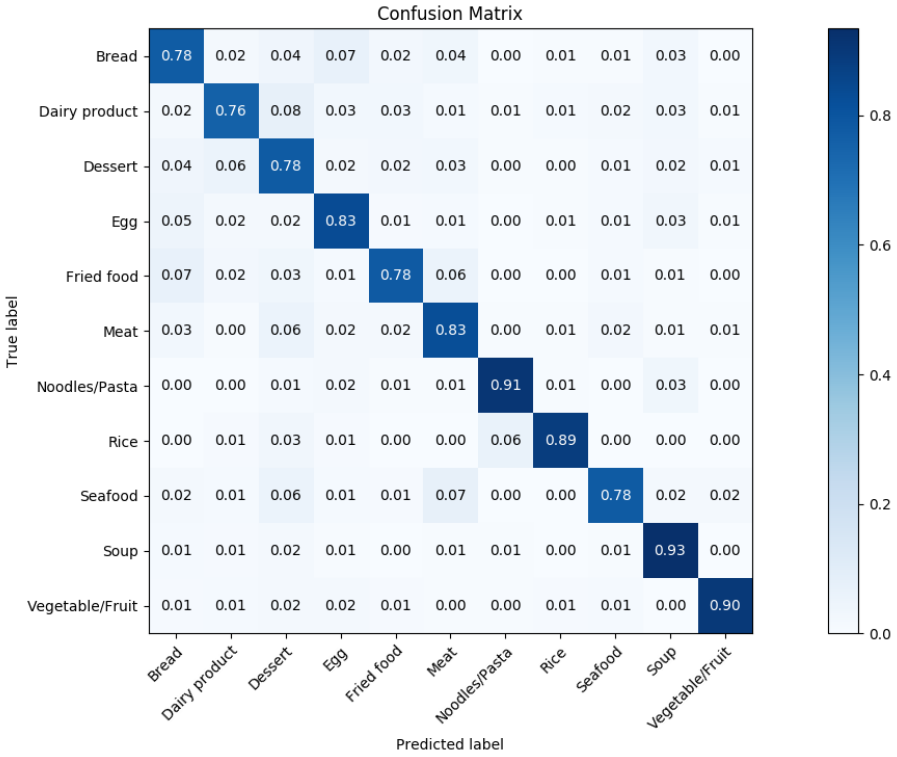
我使用的 data normalization 和 data augmentation 如第 1 題所述，此外，我也嘗試了不進行 data normalization 和 data augmentation，直接訓練第 1 題所述的 CNN，其 accuracy 如下表所示：

Train	Validation	Test	
		Public	Private
0.97223	0.65685	0.68200	0.65471

由此可以看出，沒有進行 data normalization 和 data augmentation 會使得 model 的 performance 變差，其大致是因爲，若沒有進行 data normalization，則會使得各個 feature 對 loss 的影響程度不同，其會使得 gradient descent 時 gradient 的方向會被數值較大的 feature 所主導，導致難以達到 optimization，使得 model 的 performance 變差，接著，如前一題所述，根據 ResNet 的論文中指出，當 model 的 layer 越多時，可能會使得資料在 model 中 forward 時逐漸遺失資訊，因此必須使用更多的 data augmentation 來彌補這一點，故若沒有進行 data augmentation，亦會使得 model 的 performance 變差，由此可以看出 data normalization 和 data augmentation 的影響。

6. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混?(繪出 confusion matrix 分析)

以下爲第 1 題中的 CNN 其在 validation dataset 上所得到的 confusion matrix：



由此可以看出，bread 和 egg 兩者容易被互相混淆，dairy product 和 dessert 兩者容易被互相混淆，此外，fried food 容易被分類爲 bread，meat 容易被分類爲 dessert，rice 容易被分類爲 noodles/pasta，seafood 容易被分類爲 dessert 或 meat。