學號：B07705049　系級：資管二　　姓名：林稜凱

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

```
self.cnn = nn.Sequential(
    nn.Conv2d(3, 64, 3, 1, 1),  # [64, 128, 128]  (3*3*3+1)*64 = 1792
    nn.BatchNorm2d(64),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [64, 64, 64]

    nn.Conv2d(64, 128, 3, 1, 1), # [128, 64, 64]  (3*3*64+1)*128 = 73856
    nn.BatchNorm2d(128),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [128, 32, 32]

    nn.Conv2d(128, 256, 3, 1, 1), # [256, 32, 32]  (3*3*128+1)*256 = 295168
    nn.BatchNorm2d(256),
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),      # [256, 16, 16]

    nn.Conv2d(256, 512, 3, 1, 1), # [512, 16, 16]  (3*3*256+1)*512 = 1180160
    nn.BatchNorm2d(512),
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),       # [512, 8, 8]

    nn.Conv2d(512, 512, 3, 1, 1), # [512, 8, 8]  (3*3*512+1)*512 = 2359808
    nn.BatchNorm2d(512),
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.MaxPool2d(2, 2, 0),       # [512, 4, 4]
)
self.fc = nn.Sequential(
    nn.Linear(512*4*4, 1024),  # (512*4*4+1)*1024 = 8389632
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.Linear(1024, 512),  # (1024+1)*512 = 524800
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.Linear(512, 256),  # (512+1)*256 = 131328
    nn.Dropout(0.3),
    nn.ReLU(),
    nn.Linear(256, 128),  # (256+1)*128 = 32896
    nn.ReLU(),
    nn.Linear(128, 11),  # (128+1)*11 = 1419
    nn.ReLU(),
)
```

CNN 架構如上圖，參數量約為 12990859，在以 training set 訓練 150 個 epochs 後在 validation set 上有 0.82 的準確度。

2. 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為何？(1%)

CNN 架構如右圖，參數量約為 2288187，在以 training set 訓練 90 個 epochs 後在 validation set 上有 0.72 的準確率。

因為參數減少許多，所以 train 的速度很快，在原 CNN 模型在 training set 上的準確率只有 0.65 時，此模型已經達到 0.85 的準確率，但是可能是因為參數少，導致在 training set 與 validation set 上的落差較大（接近 15%，原模型約為 6%）。

```
    self.cnn = nn.Sequential(
        nn.Conv2d(3, 64, 3, 1, 1),  # [64, 128, 128]  (3*3*3+1)*64 = 1792
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [64, 64, 64]

        nn.Conv2d(Loading..., 1, 1), # [32, 64, 64]  (3*3*64+1)*32 = 18464
        nn.BatchNorm2d(32),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [32, 32, 32]

        nn.Conv2d(32, 16, 3, 1, 1), # [16, 32, 32]  (3*3*32+1)*16 = 4624
        nn.BatchNorm2d(16),
        nn.Dropout(0.3),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [16, 16, 16]
    )
    self.fc = nn.Sequential(
        nn.Linear(16*16*16, 512),  # (16*16*16+1)*512 = 2097664
        nn.Dropout(0.3),
        nn.ReLU(),
        nn.Linear(512, 256),  # (512+1)*256 = 131328
        nn.Dropout(0.3),
        nn.ReLU(),
        nn.Linear(256, 128),  # (256+1)*128 = 32896
        nn.ReLU(),
        nn.Linear(128, 11),  # (128+1)*11 = 1419
        nn.ReLU(),
    )
# 2288187
```

3. 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、訓練參數和準確率為何？(1%)

DNN 架構如右圖，參數量約為 12098827，在以 training set 訓練 300 個 epochs 後在 validation set 上有 0.56 的準確率。

DNN 的模型 train 的速度很慢，在 CNN 已經達到在 training set 上 0.8 的準確率時，DNN 的模型仍只有 0.55，且在 training set 與 validation 上的準確率也差很多，可能是因為沒有 CNN 的局部辨識特性，使得 DNN 準確率較低。

```
    self.fc = nn.Sequential(
        nn.Linear(3*128*128, 64), # 3145728
        nn.Dropout(0.5),
        nn.ReLU(),
        nn.Linear(64, 256), # 16640
        nn.Dropout(0.3),
        nn.ReLU(),
        nn.Linear(256, 1024), # 263168
        nn.Dropout(0.3),
        nn.ReLU(),
        nn.Linear(1024, 4096), # 4198400
        nn.Dropout(0.5),
        nn.ReLU(),
        nn.Linear(4096, 1024), # 4195328
        nn.Dropout(0.5),
        nn.ReLU(),
        nn.Linear(1024, 256), # 262400
        nn.ReLU(),
        nn.Linear(256, 64), # 16448
        nn.ReLU(),
        nn.Linear(64, 11), # 715
        nn.ReLU(),
    )
# 12098827
```

4. 請說明由 1 ~ 3 題的實驗中你觀察到了什麼？(1%)

以接近的參數量訓練 DNN，訓練的速度慢，最後訓練出的模型辨識正確率也不高，說明了以 CNN 來訓練圖像辨識模型有其必要之處。層數減半的 CNN 模型訓練速度快，但在 validation set 上的表現不如原 CNN 模型，在應對各種任務時可能要以不同層數與參數多加實驗，才能找到表現最好的模型。

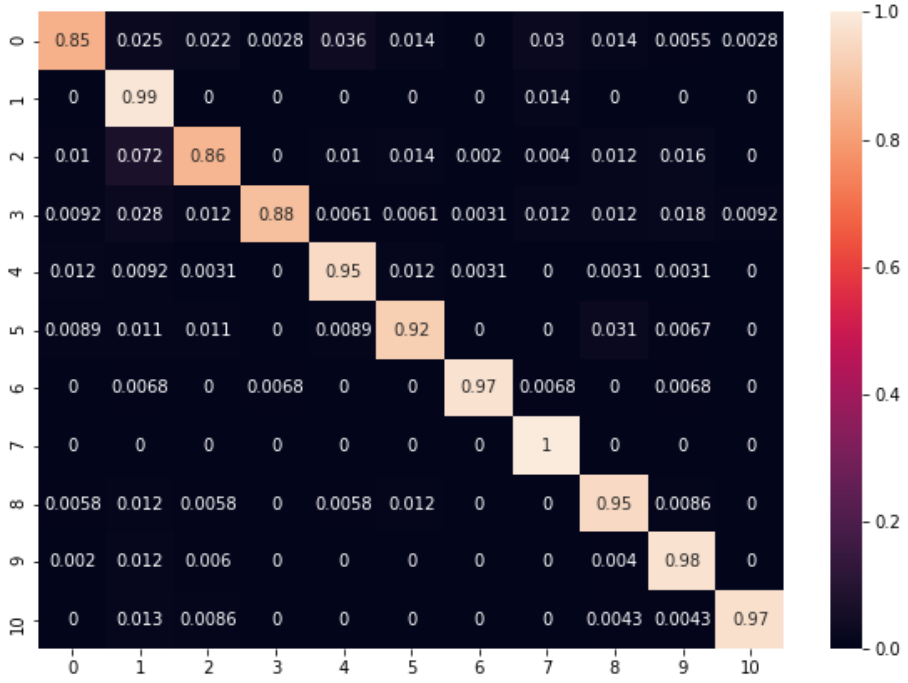
5. 請嘗試 data normalization 及 data augmentation，說明實作方法並且說明實行前後對準確率有什麼樣的影響？(1%)

原模型已做過如右的 normalization 與 augmentation，將這三行註解掉後再測試結果。

```
train_transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])
```

比起原模型，沒做 normalization 與 augmentation 的模型在 training set 與 validation 上的準確率相差較大（約 15%），說明了此措施確實能使模型更能正確辨識各種不同方向、角度的圖片。

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



以對 training set 訓練 150 個 epochs 後的模型來預設 validation set 的類別，比對結果是否正確畫出的 confusion matrix 如上圖。除了對角線外，最大的數值為 [2, 1] 的 0.072，說明模型最常將 dessert 類別誤判成 dairy product 類別，但 [1, 2] 的值為零，說明反向的情況較少發生。此外，較常互相搞混的類別有 [0, 4] = 0.036, [4, 0] = 0.012、[5, 8] = 0.031, [8, 5] = 0.012 等，說明模型常常將 bread 與 fried food、以及 meat 和 seafood 搞混。