學號：B06901180　系級：電機三　　姓名：鄭謹譯

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%)

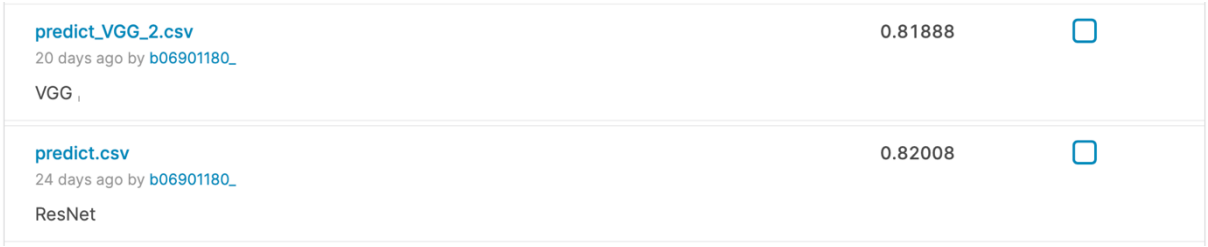
best model 是採用ensemble的方法將六個model，分別為四個類似ResNet但不同初始值（左下圖）與二個類似VGG但不同初始值（右下圖）組合而成

```
        self.fc = nn.Sequential(
            nn.Linear(256*8*8, 512),
            nn.ReLU(),
            nn.Linear(512, 11)
        )

        self.dropout = nn.Dropout2d(p=0.3)

    def forward(self, x):

        x = self.conv364(x)
        x = self.BN64_0(x)
        x_res64 = self.relu(x)
        x = self.conv6432(x_res64)
        x = self.BN32_0(x)
        x = self.relu(x)
        x = self.conv3264(x)
        x = self.BN64_1(x)
        x = self.relu(x)
        x = x+x_res64
        x = self.MP2D(x)
        x = self.conv64128(x)
        x = self.BN128_0(x)
        x_res128 = self.relu(x)
        x = self.conv12864(x_res128)
        x = self.BN64_2(x)
        x = self.relu(x)
        x = self.conv64128_1(x)
        x = self.BN128_1(x)
        x = self.relu(x)
        x = x+x_res128
        x = self.MP2D(x)
        x = self.conv128256(x)
        x = self.BN256_0(x)
        x_res256 = self.relu(x)
        x = self.conv256128(x_res256)
        x = self.BN128_2(x)
        x = self.relu(x)
        x = self.conv128256_1(x)
        x = self.BN256_1(x)
        x = self.relu(x)
        x = x+x_res256
        x_res256_1 = self.MP2D(x)
        x = self.conv256128(x_res256_1)
        x = self.BN128_3(x)
        x = self.relu(x)
        x = self.conv128256_1(x)
        x = self.BN256_2(x)
        x = self.relu(x)
        x = x+x_res256_1
        out = self.MP2D(x)
        out = out.view(out.size()[0], -1)
        out = self.dropout(out)
        return self.fc(out)
```

```
def __init__(self):
    super(Classifier, self).__init__()
    #torch.nn.Conv2d(in_channels, out_chan
    #torch.nn.MaxPool2d(kernel_size, strid
    #input 維度 [3, 128, 128]
    self.cnn = nn.Sequential(
        nn.Conv2d(3, 64, 3, 1, 1),  # [64,
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [64,

        #nn.Dropout2d(p=0.3),
        nn.Conv2d(64, 128, 3, 1, 1), # [12
        nn.BatchNorm2d(128),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [128

        #nn.Dropout2d(p=0.3),
        nn.Conv2d(128, 256, 3, 1, 1), # [2
        nn.BatchNorm2d(256),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),      # [256

        #nn.Dropout2d(p=0.3),
        nn.Conv2d(256, 512, 3, 1, 1), # [5
        nn.BatchNorm2d(512),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),       # [51

        #nn.Dropout2d(p=0.3),
        nn.Conv2d(512, 512, 3, 1, 1), # [5
        nn.BatchNorm2d(512),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),       # [51
    )
    self.fc = nn.Sequential(
        nn.Linear(512*4*4, 1024),
        nn.ReLU(),
        nn.Dropout(0.3),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.3),
        nn.Linear(512, 11)
    )
def forward(self, x):
    out = self.cnn(x)
    out = out.view(out.size()[0], -1)
    return self.fc(out)
```

左上圖參數量9543531，右上圖參數量12833803

類VGG Epoch 40，類Resnet Epoch 90，batch = 64



以下採用類VGG模型實作

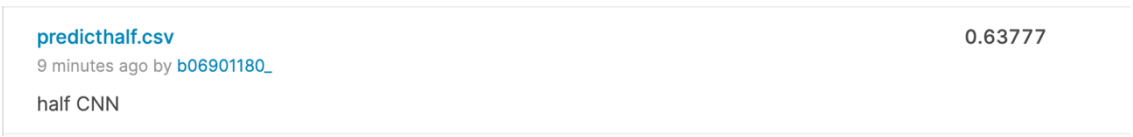
2. **請實作與第一題接近的參數量，但 CNN 深度（CNN 層數）減半的模型，並說明其模型架構、訓練參數量和準確率為何？(1%)**

```
Classifier(
  (cnn): Sequential(
    (0): Conv2d(3, 30, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(30, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(30, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(32, 47, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(47, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=12032, out_features=1024, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.3, inplace=False)
    (3): Linear(in_features=1024, out_features=512, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.3, inplace=False)
    (6): Linear(in_features=512, out_features=11, bias=True)
  )
)
Size of parameters =  12875548
```

訓練參數量 12875548，原本模型12833803，參數量誤差0.32%

Training epoch = 17 batch size=64

準確率



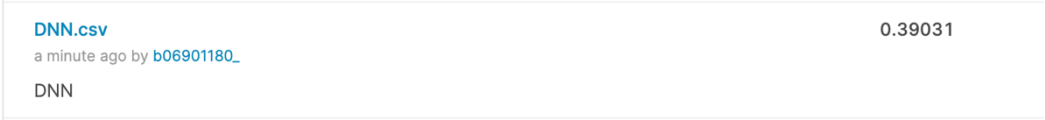
3. **請實作與第一題接近的參數量，簡單的 DNN 模型，同時也說明其模型架構、訓練參數和準確率為何？(1%)**

```
Classifier(
  (fc): Sequential(
    (0): Linear(in_features=49152, out_features=256, bias=True)
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=256, out_features=512, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.3)
    (6): Linear(in_features=512, out_features=256, bias=True)
    (7): ReLU()
    (8): Dropout(p=0.3)
    (9): Linear(in_features=256, out_features=11, bias=True)
  )
)
Size of parameters =  12849419
```

訓練參數量 12849419，原本模型12833803，參數量誤差0.12%

Training epoch = 17 batch size=64

準確率



4. **請說明由 1 ~ 3 題的實驗中你觀察到了什麼？(1%)**

實驗結果發現CNN結果優於半深度的CNN優於DNN，其中DNN又大幅差於CNN的方式。由於固定參數量，DNN每個連結的參數都不相同，需要非常多參數，相對上CNN可以共用參數，（同樣的圖形在不同處出現可以透過Filter移動來偵測）因此大幅減少非必要參數，額外留下的參數空間即可產生更多有用的Filters。

深度減半部分因為少了幾層Maxpooling，因此最後單一Filter通過FC的資料量仍大(16*16)，相較於較深的版本(4*4)能提供的Filter數量就較少，推測如此可以偵測的pattern減少，造成正確率降低。

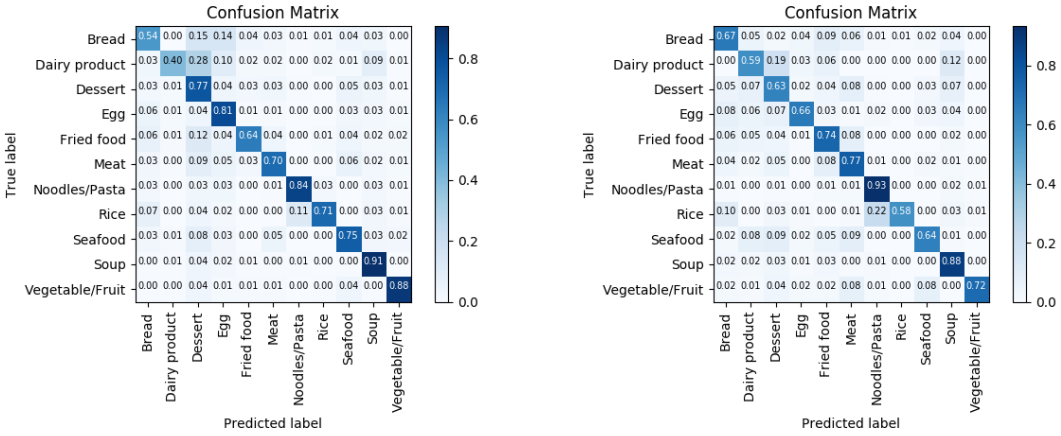
5. **請嘗試 data normalization 及 data augmentation，說明實作方法並且說明實行前後對準確率有什麼樣的影響？(1%)**

採用類Resnet對training data 和 validation data取 mean和variance

transforms.Normalize([0.55617539, 0.45159521, 0.34467578], [0.2722239
6, 0.27517131, 0.28156192])
準確率如下：

原本正確率0.82008，Normalize後略為降低，幾乎沒影響

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%)



左圖為類似VGG的模型結果、右圖為類似ResNet的模型結果。共同的弱點是D Diary product很容易辨識成Dessert，後來翻過圖片確實有些蠻像的，可能是容易答錯的原因。其次是Bread容易辨識成其他的，但沒有特定哪種類別。另外單就VGG討論，相對弱勢的是Fried Food，但相對在Soup和Vegetable/Frui t表現很好，以全體來看VGG對不擅長的類別效果很差(40%,54%)，但其高於75 %的項目又很多，較兩極化。ResNet則相反，最低仍有58%，大致正確率都維持穩定在65-75%，對種類的辨識度比較平均。

註：兩者不切割Validation Set單一筆丟上Kaggle的結果如下，相差不多