學號：**B06902125** 系級：資工三 姓名：黃柏瑋

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%)**

首先，先說明我對資料做了什麼preprocessing：

- 一開始照片讀入時會用256*256的規格，並加上Gaussian blur將畫素上的雜質去除
- 資料在training時會經過transform進行augmentation，在第5題中會再清楚說明

以下為模型架構：

- CNN的部分參照助教的模型，僅將ReLU改為PReLU，其餘都沒有改變：

```
self.cnn = nn.Sequential(
        nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, stride=1, padding=1),   # [64, 128, 128]
        nn.BatchNorm2d(64),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [64, 64, 64]

        nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1),   # [128, 64, 64]
        nn.BatchNorm2d(128),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [128, 32, 32]

        nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, stride=1, padding=1),   # [256, 32, 32]
        nn.BatchNorm2d(256),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [256, 16, 16]

        nn.Conv2d(in_channels=256, out_channels=512, kernel_size=3, stride=1, padding=1),   # [512, 16, 16]
        nn.BatchNorm2d(512),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [512, 8, 8]

        nn.Conv2d(in_channels=512, out_channels=512, kernel_size=3, stride=1, padding=1),   # [512, 8, 8]
        nn.BatchNorm2d(512),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0))   # [512, 4, 4]
```

- Fully-connected的部分也是參照助教的模型，但多加了幾層dropout，防止overfitting發生：

```
self.fc = nn.Sequential(
        nn.Dropout(0.3),
        nn.Linear(512*4*4, 1024),
        nn.PReLU(),
        nn.Dropout(0.2),
        nn.Linear(1024, 512),
        nn.PReLU(),
        nn.Dropout(0.1),
        nn.Linear(512, 11)
```

- 總參數量為12833810

最後，為此模型的準確率：

	TRAIN	**VAL**
best model	0.9680	0.7524

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%)**

首先，先將CNN的深度改為以下，總層數約為一伴而已：

```
self.cnn = nn.Sequential(
        nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, stride=1, padding=1),   # [64, 128, 128]
        nn.BatchNorm2d(64),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [64, 64, 64]

        nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1),   # [128, 64, 64]
        nn.BatchNorm2d(128),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0),   # [128, 32, 32]

        nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, stride=1, padding=1),   # [256, 32, 32]
        nn.BatchNorm2d(256),
        nn.PReLU(),
        nn.MaxPool2d(kernel_size=2, stride=2, padding=0))   # [256, 16, 16]
```

並將fully-connected的結構改為以下：

```
self.fc = nn.Sequential(
        nn.Dropout(0.3),
        nn.Linear(256*16*16, 256),
        nn.PReLU(),
        nn.Dropout(0.2),
        nn.Linear(256, 256),
        nn.PReLU(),
        nn.Dropout(0.1),
        nn.Linear(256, 11),
```

總參數量會為12217808

最後，為此模型的準確率：

	TRAIN	VAL
CNN_shallow	0.73570	0.6227

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%)

模型架構如下，直接把圖片的畫素都拉成一維丟入fully-connected NN：

```
self.fc = nn.Sequential(
        nn.Linear(3*128*128, 256),
        nn.PReLU(),
        nn.Dropout(0.2),
        nn.Linear(256, 128),
        nn.PReLU(),
        nn.Dropout(0.1),
        nn.Linear(128, 96),
        nn.PReLU(),
        nn.Dropout(0.1),
        nn.Linear(96, 11),
)
```

總參數量為12629518

最後，為此模型的準確率：

	TRAIN	VAL
DNN	0.3016	0.3085

4. 請說明由 1 ~ 3 題的實驗中你觀察到了什麼？(1%)

從第一題到第三題中，我發現CNN的效果明顯比全連接層顯著，當我們逐步減少CNN的層數，training的效果就越來越差。即使模型的參數維持差不多，但DNN減持慘不忍睹，無法有效地找出畫素之間的關係，造成underfitting。

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%)

首先，先做data normalization。原本的best model在做transform的時候有將所有像素都除以255，現在將他乘回去看看效果如何：

	TRAIN	VAL
best model w/ normalization	0.9680	0.7524
best model w/o normalization	0.9577	0.7245

我們能發現有做normalization的時候，訓練的效果比較好，無論是training accuracy或是validation accuracy都比較高。

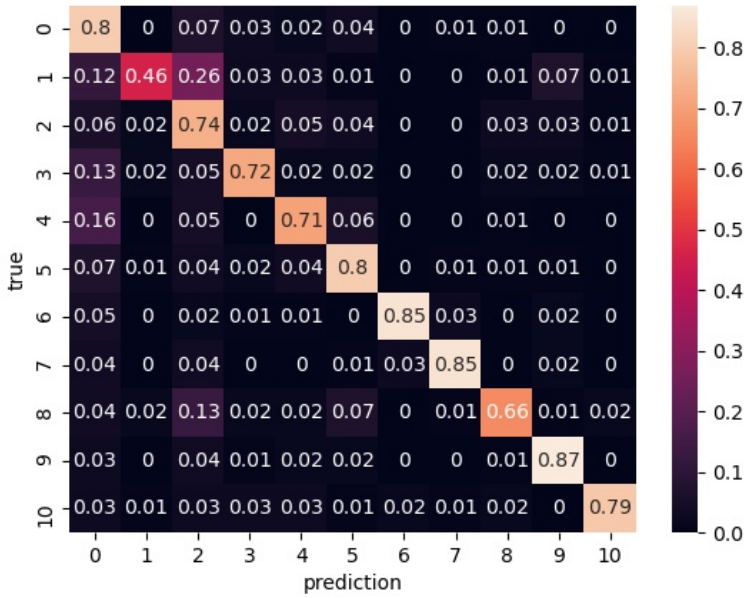
接著，比較data augmentation。原先的best model會先利用randomcrop將256*256隨機切成128*128，再隨機旋轉45度及水平翻轉，現在將這些效果拿掉，看看效果如何：

	TRAIN	VAL
best model w/ aug	0.9680	0.7524
best model w/o aug	0.9915	0.6869

我們能發現有做augmentation能有效減少overfitting，雖然training accuracy較低，但得到了較好的validation accuracy。

6. 觀察答錯的圖片中，哪些 class 彼此間容易用混？(1%)

由於我想要觀察某些原本為class k的會容易被誤解為哪一個class，因此我將confusion matrix按true label axis進行normalization：



- true label為第1類的很相當容易被誤判為第2類
- true label為第3類的蠻容易被誤判為第0類
- true label為第4類的蠻容易被誤判為第0類
- true label為第8類的蠻容易被誤判為第2類