模型來源:[李宏毅教授深度學習課程](https://colab.research.google.com/drive/15A_8ilH-6-T3HOmSFrKbjDinBJl-s-16)
https://colab.research.google.com/drive/15A_8ilH-6-T3HOmSFrKbjDinBJl-s-16

模型任務:食物圖片分類

原本的模型架構:
5 層卷積層，使用 3X3 的卷積核，stride=1，padding=1
每一層卷積層後面接一個 2X2 的 MaxPooling

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

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

            nn.Conv2d(128, 256, 3, 1, 1), # [256, 32, 32]
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.MaxPool2d(2, 2, 0),      # [256, 16, 16]

            nn.Conv2d(256, 512, 3, 1, 1), # [512, 16, 16]
            nn.BatchNorm2d(512),
            nn.ReLU(),
            nn.MaxPool2d(2, 2, 0),       # [512, 8, 8]

            nn.Conv2d(512, 512, 3, 1, 1), # [512, 8, 8]
            nn.BatchNorm2d(512),
            nn.ReLU(),
            nn.MaxPool2d(2, 2, 0),       # [512, 4, 4]
        )
        self.fc = nn.Sequential(
            nn.Linear(512*4*4, 1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Linear(512, 11)
        )

    def forward(self, x):
        out = self.cnn(x)
        out = out.view(out.size()[0], -1)
        return self.fc(out)
```

分類效果:
準確率:61.8%

```
[ Valid | 007/008 ] loss = 1.17393, acc = 0.61800
[ Valid | 007/008 ] loss = 1.17393, acc = 0.61800 -> best
```

我修改過的模型架構:

6 層卷積層，前三層用 5X5 的卷積核，為了不改變圖的大小把 padding 改成 2

後三層用 3X3 的卷積核，最後一層我覺得 4X4 已經夠小了，所以不做 pooling

```
self.cnn = nn.Sequential(
        nn.Conv2d(3, 64, 5, 1, 2),    # [64, 128, 128]
        nn.BatchNorm2d(64),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),          # [64, 64, 64]

        nn.Conv2d(64, 128, 5, 1, 2),  # [128, 64, 64]
        nn.BatchNorm2d(128),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),          # [128, 32, 32]

        nn.Conv2d(128, 256, 5, 1, 2),  # [256, 32, 32]
        nn.BatchNorm2d(256),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),          # [256, 16, 16]

        nn.Conv2d(256, 512, 3, 1, 1),  # [512, 16, 16]
        nn.BatchNorm2d(512),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),          # [512, 8, 8]

        nn.Conv2d(512, 512, 3, 1, 1),  # [512, 8, 8]
        nn.BatchNorm2d(512),
        nn.ReLU(),
        nn.MaxPool2d(2, 2, 0),          # [512, 4, 4]

        nn.Conv2d(512, 1024, 3, 1, 1),  # [1024, 4, 4]
        nn.BatchNorm2d(1024),
        nn.ReLU()

)
self.fc = nn.Sequential(
        nn.Linear(1024*4*4, 1024),
        nn.ReLU(),
        nn.Dropout(0.5),
        nn.Linear(1024, 512),
        nn.ReLU(),
        nn.Dropout(0.5),
        nn.Linear(512, 256),
        nn.ReLU(),
        nn.Dropout(0.5),
        nn.Linear(256, 11)
)
```

分類效果:65.6%

```
[ Train | 020/020 ] loss = 0.78211, acc = 0.74582
100% |████████████████████████████████████████| 57/57 [00:19<00:00,  3.24it/s]
[ Valid | 020/020 ] loss = 1.07402, acc = 0.65636
[ Valid | 020/020 ] loss = 1.07402, acc = 0.65636 -> best
Best model found at epoch 19, saving model
```