使用 pytorch 實作 CIFAR-10 資料進行模型預測

朱家佑

andy32252@gmail.com

code: https://github.com/andy32252/CNNProject/blob/main/finally.ipynb

資料來源及內容:

使用 Kaggle 上的 CIFAR-10 資料集，包含了 60,000 張 32x32 的彩色圖片，共有 10 個類別:飛機、汽車、鳥類、貓、鹿、狗、青蛙、馬、船、卡車，每一個類別都有 6000 張圖片。

實作方法:

使用一般卷積神經網路 CNN 及 ResNet 做比較，對 CIFAR-10 進行模型的訓練，並且比較兩者差異，資料處理及模型訓練步驟如下:

1. 使用 PyTorch 中的 transforms 模組，使資料增強，有助於模型提升泛化性能，使用 RandomResizedCrop 將圖片進行隨機裁減及縮放，並重新調整成 224x224 像素。
2. 將圖片轉為 PyTorch 的張量(tensor)格式。
3. 使用 Normalize 對圖片 RGB 通道進行正規化，使用平均值為 0.5、標準差為 0.5，像素範圍由[0,255]縮至[-1,1]。
4. 將圖片指定為 32x32。
   - 隨機旋轉圖片最大角度為 20 度。
   - 以 10%機率隨機水平翻轉圖片。
   - 10%機率隨機調整圖片亮度、對比度、飽和度、清晰度。
   - 以 75%機率隨機去除圖片的某區域，為了增加模型對各種變化的適應。
5. 將資料集分為 50,000 張圖片的訓練集，10,000 張圖片的測試集。
6. 訓練設置：

- 批次大小(Batch Size)為 64。
- 訓練週期(Epochs)為 20。
- 最大學習率(Learning Rate)為 0.01。
- 損失函數(Loss Function)為交叉熵(Cross-Entropy)。
- 優化器(Optimizer)為 Adam。
- 設置權重衰減(Weight Decay)為 0.0001。
- 梯度裁減(grad_clip)為 0.1 防止梯度爆炸。

7. CNN 模型每層如下:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,728
BatchNorm2d-2	[-1, 64, 32, 32]	128
ReLU-3	[-1, 64, 32, 32]	0
Conv2d-4	[-1, 64, 32, 32]	36,928
BatchNorm2d-5	[-1, 64, 32, 32]	128
ReLU-6	[-1, 64, 32, 32]	0
Conv2d-7	[-1, 64, 32, 32]	36,928
BatchNorm2d-8	[-1, 64, 32, 32]	128
ReLU-9	[-1, 64, 32, 32]	0
AvgPool2d-10	[-1, 64, 16, 16]	0
Linear-11	[-1, 64]	1,048,640
Linear-12	[-1, 10]	650

Total params: 1,125,258

訓練準確能達到 99.9%

但是預測時將測試集的 10,000 張圖片帶入已訓練好的 CNN 模型，總共跑 100 次，最後將這 100 次的準確率加起來平均得到的準確率只有 44.53%。

```
Epoch: 1/20, last_lr: 0.00104, train_loss: 1.2251, val_loss: 1.1340, val_accuracy: 56.2%
Epoch: 2/20, last_lr: 0.00280, train_loss: 0.9565, val_loss: 0.9798, val_accuracy: 66.4%
Epoch: 3/20, last_lr: 0.00520, train_loss: 0.8654, val_loss: 0.8731, val_accuracy: 69.9%
Epoch: 4/20, last_lr: 0.00760, train_loss: 0.8022, val_loss: 0.8716, val_accuracy: 72.1%
Epoch: 5/20, last_lr: 0.00936, train_loss: 0.7753, val_loss: 0.8844, val_accuracy: 73.0%
Epoch: 6/20, last_lr: 0.01000, train_loss: 0.7631, val_loss: 0.9471, val_accuracy: 73.6%
Epoch: 7/20, last_lr: 0.00987, train_loss: 0.7496, val_loss: 0.8872, val_accuracy: 74.2%
Epoch: 8/20, last_lr: 0.00950, train_loss: 0.7370, val_loss: 0.8789, val_accuracy: 74.7%
Epoch: 9/20, last_lr: 0.00891, train_loss: 0.7177, val_loss: 0.8847, val_accuracy: 75.2%
Epoch: 10/20, last_lr: 0.00812, train_loss: 0.6827, val_loss: 0.8416, val_accuracy: 76.6%
Epoch: 11/20, last_lr: 0.00717, train_loss: 0.6359, val_loss: 0.8295, val_accuracy: 77.8%
Epoch: 12/20, last_lr: 0.00611, train_loss: 0.5726, val_loss: 0.7915, val_accuracy: 80.1%
Epoch: 13/20, last_lr: 0.00500, train_loss: 0.4922, val_loss: 0.8078, val_accuracy: 82.8%
Epoch: 14/20, last_lr: 0.00389, train_loss: 0.3957, val_loss: 0.7940, val_accuracy: 86.2%
Epoch: 15/20, last_lr: 0.00283, train_loss: 0.2800, val_loss: 0.8534, val_accuracy: 90.3%
Epoch: 16/20, last_lr: 0.00188, train_loss: 0.1811, val_loss: 0.9360, val_accuracy: 93.8%
Epoch: 17/20, last_lr: 0.00109, train_loss: 0.0898, val_loss: 0.9849, val_accuracy: 97.2%
Epoch: 18/20, last_lr: 0.00050, train_loss: 0.0370, val_loss: 1.0411, val_accuracy: 99.2%
Epoch: 19/20, last_lr: 0.00013, train_loss: 0.0186, val_loss: 1.0851, val_accuracy: 99.8%
Epoch: 20/20, last_lr: 0.00000, train_loss: 0.0135, val_loss: 1.0908, val_accuracy: 99.9%
```

8. ResNet 每層如下:

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 32, 32]	1,792
BatchNorm2d-2	[-1, 64, 32, 32]	128
ReLU-3	[-1, 64, 32, 32]	0
Conv2d-4	[-1, 128, 32, 32]	73,856
BatchNorm2d-5	[-1, 128, 32, 32]	256
ReLU-6	[-1, 128, 32, 32]	0
MaxPool2d-7	[-1, 128, 16, 16]	0
Conv2d-8	[-1, 128, 16, 16]	147,584
BatchNorm2d-9	[-1, 128, 16, 16]	256
ReLU-10	[-1, 128, 16, 16]	0

Conv2d-11	[-1, 128, 16, 16]	147,584
BatchNorm2d-12	[-1, 128, 16, 16]	256
ReLU-13	[-1, 128, 16, 16]	0
Conv2d-14	[-1, 256, 16, 16]	295,168
BatchNorm2d-15	[-1, 256, 16, 16]	512
ReLU-16	[-1, 256, 16, 16]	0
MaxPool2d-17	[-1, 256, 8, 8]	0

Conv2d-18	[-1, 512, 8, 8]	1,180,160
BatchNorm2d-19	[-1, 512, 8, 8]	1,024
ReLU-20	[-1, 512, 8, 8]	0
MaxPool2d-21	[-1, 512, 4, 4]	0

Conv2d-22	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-23	[-1, 512, 4, 4]	1,024
ReLU-24	[-1, 512, 4, 4]	0
Conv2d-25	[-1, 512, 4, 4]	2,359,808
BatchNorm2d-26	[-1, 512, 4, 4]	1,024
ReLU-27	[-1, 512, 4, 4]	0
MaxPool2d-28	[-1, 512, 1, 1]	0

Flatten-29	[-1, 512]	0
Linear-30	[-1, 128]	65,664
Linear-31	[-1, 10]	1,290

Total params: 6,637,194

訓練準確達到 99.8%

預測將測試集的 10,000 張圖片帶入已訓練好的 ResNet 模型，總共跑 100 次，最後將這 100 次的準確率加起來平均得到的準確率 66.02%

```
Epoch: 1/20, last_lr: 0.00104, train_loss: 1.0814, val_loss: 0.8003, val_accuracy: 61.7%
Epoch: 2/20, last_lr: 0.00280, train_loss: 0.7443, val_loss: 0.8172, val_accuracy: 74.4%
Epoch: 3/20, last_lr: 0.00520, train_loss: 0.7302, val_loss: 0.7646, val_accuracy: 75.3%
Epoch: 4/20, last_lr: 0.00760, train_loss: 0.7060, val_loss: 0.8534, val_accuracy: 76.4%
Epoch: 5/20, last_lr: 0.00936, train_loss: 0.6844, val_loss: 0.8469, val_accuracy: 76.9%
Epoch: 6/20, last_lr: 0.01000, train_loss: 0.6837, val_loss: 0.8605, val_accuracy: 77.0%
Epoch: 7/20, last_lr: 0.00987, train_loss: 0.6739, val_loss: 0.6773, val_accuracy: 77.0%
Epoch: 8/20, last_lr: 0.00950, train_loss: 0.6400, val_loss: 0.6904, val_accuracy: 78.1%
Epoch: 9/20, last_lr: 0.00891, train_loss: 0.6051, val_loss: 0.6205, val_accuracy: 79.4%
Epoch: 10/20, last_lr: 0.00812, train_loss: 0.5662, val_loss: 0.6822, val_accuracy: 80.7%
Epoch: 11/20, last_lr: 0.00717, train_loss: 0.5169, val_loss: 0.6018, val_accuracy: 82.2%
Epoch: 12/20, last_lr: 0.00611, train_loss: 0.4725, val_loss: 0.5606, val_accuracy: 83.8%
Epoch: 13/20, last_lr: 0.00500, train_loss: 0.4128, val_loss: 0.5183, val_accuracy: 85.7%
Epoch: 14/20, last_lr: 0.00389, train_loss: 0.3535, val_loss: 0.4917, val_accuracy: 88.0%
Epoch: 15/20, last_lr: 0.00283, train_loss: 0.2777, val_loss: 0.5101, val_accuracy: 90.2%
Epoch: 16/20, last_lr: 0.00188, train_loss: 0.1936, val_loss: 0.5050, val_accuracy: 93.3%
Epoch: 17/20, last_lr: 0.00109, train_loss: 0.1136, val_loss: 0.5287, val_accuracy: 96.1%
Epoch: 18/20, last_lr: 0.00050, train_loss: 0.0514, val_loss: 0.5536, val_accuracy: 98.5%
Epoch: 19/20, last_lr: 0.00013, train_loss: 0.0238, val_loss: 0.5673, val_accuracy: 99.6%
Epoch: 20/20, last_lr: 0.00000, train_loss: 0.0160, val_loss: 0.5701, val_accuracy: 99.8%
```

結論:

由於 ResNet 模型採用了殘差結構，成功地克服了梯度消失的問題，這使得在損失（val_loss）方面表現優於一般的 CNN。此外，ResNet 的跳躍連接有助於緩解梯度消失，進而促使模型更好地學習特徵。透過將 10000 張測試集圖片投入已訓練好的 ResNet 模型中，我們經過 100 次測試得出的平均準確率為 66.02%。相比之下，投入一般 CNN 已訓練模型的結果為 44.53%的準確率。綜合考量，ResNet 模型在處理此測試集上表現更為出色。