CNN-Copy1

2018 年 8 月 3 日

1 再論 CNN

1.1 介紹

上一章節我們已經試過了超簡化版的 VGG-16，我們再來加上一些深度，看看加上深度到底對我們整個模型的影響為何

In [1]:
```
from keras.datasets import cifar10
# MAC 一定要加入此行，才不會把對方伺服器的 SSL 證書視為無效
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
import matplotlib.pyplot as plt
%matplotlib inline

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Using TensorFlow backend.
```

一樣看看 shape

In [2]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

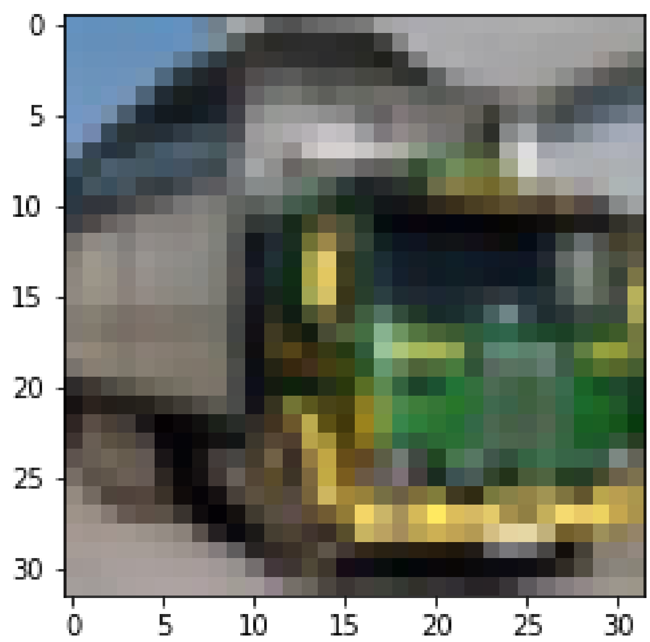
```
In [3]: label = {0:"飛機", 1:"車", 2:"鳥", 3:"貓", 4:"鹿",
                 5:"狗", 6:"青蛙", 7:"馬", 8:"船", 9:"卡車"}
```

可視化你想看的圖片

```
In [4]: a = int(input("請輸入你想可視化的圖片 [0-49999]:"))
        print("你想可視化的圖片號碼是", a)
        print("圖片答案是", label[y_train[a][0]])
        plt.imshow(x_train[a])
```

```
請輸入你想可視化的圖片 [0-49999]:14
你想可視化的圖片號碼是 14
圖片答案是 卡車
```

```
Out[4]: <matplotlib.image.AxesImage at 0x12490d6d8>
```



一樣對特徵做出標準化，並且對目標做出 One-hot 編碼

```
In [5]: from keras.utils import np_utils
        x_train_shaped = x_train.astype("float32") / 255
        x_test_shaped = x_test.astype("float32") / 255
        y_train_cat = np_utils.to_categorical(y_train)
        y_test_cat = np_utils.to_categorical(y_test)
```

這次我們真的遵照 VGG-16 的結構來構建卷積層，我們總共做了 6 次的卷積 (特徵萃取)，三次的池化 (減少計算量)，最後在接上全連接層做出分類

```
In [6]: from keras.models import Sequential
        from keras.layers import Dense, Dropout, Activation, Flatten
        from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D

        model = Sequential()



        model.add(Conv2D(filters=64,
                       kernel_size=(3, 3),
                       input_shape=(32, 32, 3),
                       activation='relu',
                       padding='same'))
        model.add(Conv2D(filters=64,
                       kernel_size=(3, 3),
                       activation='relu',
                       padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Dropout(0.25))
        model.add(Conv2D(filters=128,
                       kernel_size=(3, 3),
                       activation='relu',
                       padding='same'))
        model.add(Conv2D(filters=128,
                       kernel_size=(3, 3),
                       activation='relu',
                       padding='same'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
model.add(Conv2D(filters=256,
                 kernel_size=(3, 3),
                 activation='relu',
                 padding='same'))
model.add(Conv2D(filters=256,
                 kernel_size=(3, 3),
                 activation='relu',
                 padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dropout(rate=0.25))

model.add(Dense(1024, activation='relu'))
model.add(Dropout(rate=0.25))

model.add(Dense(10, activation='softmax'))
model.summary()
```

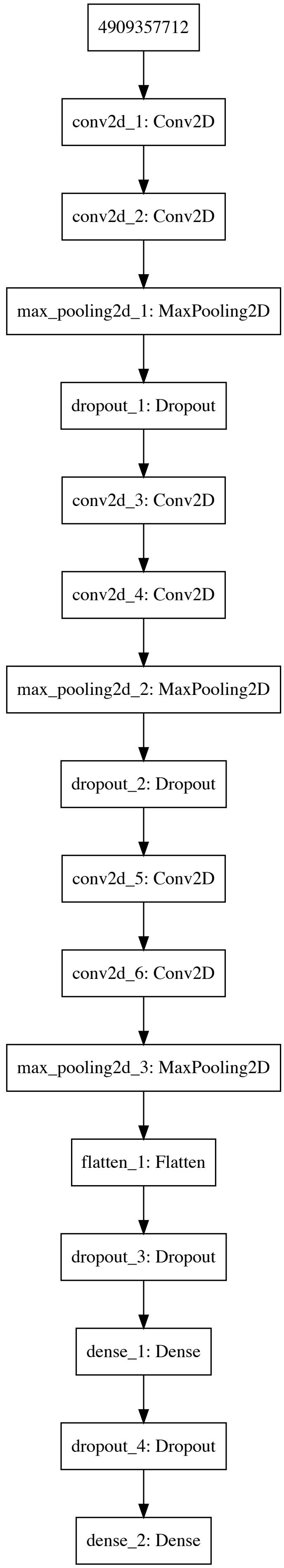
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 64)        1792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
```

```
dropout_2 (Dropout)          (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 256)         295168
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_3 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              4195328
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 5,350,986
Trainable params: 5,350,986
Non-trainable params: 0
_________________________________________________________________
```

```
In [ ]: from IPython.display import SVG
        from keras.utils.vis_utils import model_to_dot

        SVG(model_to_dot(model).create(prog='dot', format='svg'))
```

```
Out[None]:
```


4909357712
conv2d_1: Conv2D
conv2d_2: Conv2D
max_pooling2d_1: MaxPooling2D
dropout_1: Dropout
conv2d_3: Conv2D
conv2d_4: Conv2D
max_pooling2d_2: MaxPooling2D
dropout_2: Dropout
conv2d_5: Conv2D
conv2d_6: Conv2D
max_pooling2d_3: MaxPooling2D
flatten_1: Flatten
dropout_3: Dropout
dense_1: Dense
dropout_4: Dropout
dense_2: Dense

```
In [ ]: model.compile(loss="categorical_crossentropy",
                      optimizer = "adam",
                      metrics = ['accuracy'])
        train_history = model.fit(x = x_train_shaped, y = y_train_cat,
                                  validation_split = 0.1,
                                  epochs = 25,
                                  batch_size = 200,
                                  verbose = 2)
```

```
Train on 45000 samples, validate on 5000 samples
Epoch 1/25
 - 506s - loss: 1.7535 - acc: 0.3524 - val_loss: 1.3401 - val_acc: 0.5048
Epoch 2/25
 - 1332s - loss: 1.2987 - acc: 0.5338 - val_loss: 1.1006 - val_acc: 0.6030
Epoch 3/25
 - 530s - loss: 1.0836 - acc: 0.6148 - val_loss: 0.9409 - val_acc: 0.6720
Epoch 4/25
 - 524s - loss: 0.9304 - acc: 0.6695 - val_loss: 0.8318 - val_acc: 0.7104
Epoch 5/25
 - 548s - loss: 0.8294 - acc: 0.7091 - val_loss: 0.7503 - val_acc: 0.7418
Epoch 6/25
 - 572s - loss: 0.7383 - acc: 0.7391 - val_loss: 0.6894 - val_acc: 0.7628
Epoch 7/25
 - 577s - loss: 0.6673 - acc: 0.7651 - val_loss: 0.6526 - val_acc: 0.7760
Epoch 8/25
 - 562s - loss: 0.6147 - acc: 0.7840 - val_loss: 0.6549 - val_acc: 0.7792
Epoch 9/25
 - 560s - loss: 0.5583 - acc: 0.8043 - val_loss: 0.6476 - val_acc: 0.7808
Epoch 10/25
 - 549s - loss: 0.5181 - acc: 0.8192 - val_loss: 0.5955 - val_acc: 0.8032
Epoch 11/25
```

```
In [ ]: plt.plot(train_history.history["loss"])
        plt.plot(train_history.history["val_loss"])
```

```
plt.title("Loss Graph")
plt.legend(['loss', 'val_loss'], loc="upper left")
```

In [ ]:
```
model.evaluate(x_test_shaped, y_test_cat)
```