NCTU Pattern Recognition, Homework 5

Deadline: June 12, 23:59

Coding (100%):

In this coding assignment, you need to implement the deep neural network by any deep learning fr amework, e.g., Pytorch, TensorFlow, or Keras, then train the DNN model by the Cifar-10 dataset a nd try to beat the baseline performance.

Download dataset HERE.

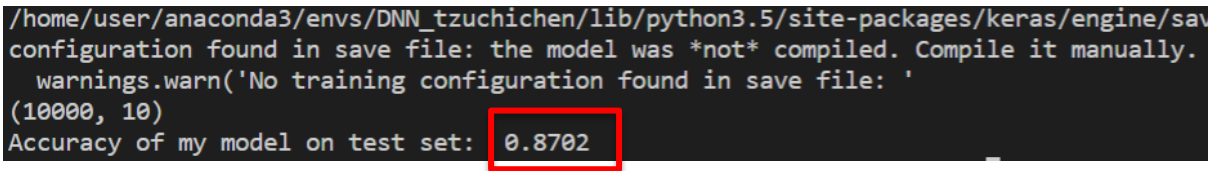
Please note that you should only train and evaluate your model on the provided dataset.

DO NOT download the data from other resources.

If you are a newbie in a deep learning framework, we recommend learning Keras or Pytorch.

- Pytorch tutorial
- Keras tutorial
- TensorFlow tutorial

1. (100%) Show your accuracy of your model on the provided test data by screenshot the res ults of your code and paste them on your report



Evaluation:

Accuracy	Your scores
acc >= 0.95	100 points
0.9 <= acc < 0.95	90 points
0.80 <= acc < 0.90	80 points
0.75 <= acc < 0.80	70 points
0.65 <= acc < 0.75	60 points
0.6 <= acc < 0.65	50 points
acc <0.6	No points

Note: Keyword to boost your model performance

1. Data augmentation
2. Hyperparameter searches for model structure (number of filters, number of

convolution/dense layer) and optimizer (learning rate)

3. Regularization

Note: If your result is bad, check this tutorial first to debug your model

NOTE: 如果助教需要測試程式，請手動安裝以下這些套件(或參考requirements.txt)
tensorflow
numpy
matplotlib
keras
h5py
sklearn
Pillow
只要讓它自動選擇最新版本即可(例如: conda install tensorflow)。

以下為我增加效能的小方法

1.Data augmentation:

```
166     # set up image augmentation
167     datagen = ImageDataGenerator(
168         rotation_range=15,
169         horizontal_flip=True,
170         #vertical_flip=True,
171         width_shift_range=0.1,
172         height_shift_range=0.1
173         #zoom_range=0.3
174         )
```

旋轉15度、左右翻轉、左右平移0.1倍的width、上下平移0.1倍的height。

Regularization:

```
38      model = Sequential()
39
40      model.add(Conv2D(num_filters, (3, 3), activation=ac, kernel_regularizer=reg, input_shape=(img_r
41      model.add(BatchNormalization(axis=-1))
42      model.add(Conv2D(num_filters, (3, 3), activation=ac,kernel_regularizer=reg,padding='same'))
43      model.add(BatchNormalization(axis=-1))
44      model.add(MaxPooling2D(pool_size=(2, 2)))   # reduces to 16x16x3xnum_filters
45      model.add(Dropout(drop_conv))
46
47      model.add(Conv2D(2*num_filters, (3, 3), activation=ac,kernel_regularizer=reg,padding='same'))
48      model.add(BatchNormalization(axis=-1))
49      model.add(Conv2D(2*num_filters, (3, 3), activation=ac,kernel_regularizer=reg,padding='same'))
50      model.add(BatchNormalization(axis=-1))
51      model.add(MaxPooling2D(pool_size=(2, 2)))   # reduces to 8x8x3x(2*num_filters)
52      model.add(Dropout(drop_conv))
53
54      model.add(Conv2D(4*num_filters, (3, 3), activation=ac,kernel_regularizer=reg,padding='same'))
55      model.add(BatchNormalization(axis=-1))
```

```
55      model.add(BatchNormalization(axis=-1))
56      model.add(Conv2D(4*num_filters, (3, 3), activation=ac,kernel_regularizer=reg padding='same'))
57      model.add(BatchNormalization(axis=-1))
58      model.add(MaxPooling2D(pool_size=(2, 2)))   # reduces to 4x4x3x(4*num_filters)
59      model.add(Dropout(drop_conv))
60
61      model.add(Flatten())
62      model.add(Dense(512, activation=ac kernel_regularizer=reg))
63      model.add(BatchNormalization())
64      model.add(Dropout(drop_dense))
65      model.add(Dense(num_classes, activation='softmax'))
66
67      #model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='Adam')
68      model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=keras.optimizers.ad
```

每一層都加上regularizer來防止overfitting。

另外，因為訓練時間太長，我沒有使用Hyperparameter searches。我直接選用推薦的optimizer (learning rate): keras.optimizers.adamax(lr=5e-3)。

我選擇參考VGG模型架構，架構如下:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_1 (Batch	(None, 32, 32, 32)	128
conv2d_2 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_2 (Batch	(None, 32, 32, 32)	128
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv2d_3 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_3 (Batch	(None, 16, 16, 64)	256
conv2d_4 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_4 (Batch	(None, 16, 16, 64)	256
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 64)	0

dropout_2 (Dropout)	(None, 8, 8, 64)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_5 (Batch	(None, 8, 8, 128)	512
conv2d_6 (Conv2D)	(None, 8, 8, 128)	147584
batch_normalization_6 (Batch	(None, 8, 8, 128)	512
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 128)	0
dropout_3 (Dropout)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
batch_normalization_7 (Batch	(None, 512)	2048
dropout_4 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130

Total params: 1,345,066
Trainable params: 1,343,146
Non-trainable params: 1,920

我的架構比VGG簡單一些，一般VGG16有13層Convolution layer、5層Maxpooling layer 、3層Fully connected layer。VGG16或VGG19太多層，訓練時間太長，因此我縮短成6層Convolution layer、3層Maxpooling layer、1層Fully connected layer。每兩層Convolution layer後面都有一層3層Maxpooling layer，最後再攤平完成Fully connected layer。