cnn-cifar10-keras-v0.2.0

July 24, 2018

1 Classification Using Keras

Richard Kuo, 20180706 ver. 0.2.0
notebook - cnn-cifar10-keras.ipynb pdf dir - /Users/rkuo/code/tensorflow/cnn-cifar10

This is very similar to ccn-cifar10-tf model, all the housekeeping, import statements are the same; we will copy them here and replace the model building with Keras API. We will refactor some code too.

We will build a simple model of
2 convolution layer ,
1 pooling layer and
a fully connected layer.

Code borrowed from: - Cifar-10 Classification using Keras Tutorial - Object Recognition with Convolutional Neural Networks in the Keras Deep Learning Library - Convolutional Neural Networks (CNN) for CIFAR-10 Dataset - Deep-math-machine-learning.ai - Keras code example

1.1 Load and display dataset

After data loading, to verify and better understand the dataset; sample some them. For more complicate dataset, plot, explore the contents. - shapes - sizes - sample values

In [1]:
```
# Loading the CIFAR-10 datasets
import keras
from keras.datasets import cifar10

# load data, instead of using the built-in function, this can be done with pyth
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
```

```
Using TensorFlow backend.
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 90s 1us/step
```

In [2]:
```
# Plot
import matplotlib.pyplot as plt
% matplotlib inline
```

```
        from scipy.misc import toimage

        import numpy as np

        X_train  # tensor type
        Y_train
        print('X_train shape:', X_train.shape)
        print('Y_train shape:', Y_train.shape)
        print(X_train.shape[0], ' train samples')
        print(X_test.shape[0],  ' test samples')
        print("Value of the first element of X_train:")
        print(X_train[0])
        print("Value of the first element of Y_train:")
        print(Y_train[0])
        # create a grid of 3x3 images
        print("X can be converted back to original images via utility function:")
        for i in range(0, 9):
          plt.subplot(330 + 1 + i)
          plt.imshow(toimage(X_train[i]))
        # show the plot
        plt.show()
```

```
X_train shape: (50000, 32, 32, 3)
Y_train shape: (50000, 1)
50000  train samples
10000  test samples
Value of the first element of X_train:
[[[ 59  62  63]
  [ 43  46  45]
  [ 50  48  43]
  ...,
  [158 132 108]
  [152 125 102]
  [148 124 103]]

 [[ 16  20  20]
  [  0   0   0]
  [ 18   8   0]
  ...,
  [123  88  55]
  [119  83  50]
  [122  87  57]]

 [[ 25  24  21]
  [ 16   7   0]
  [ 49  27   8]
  ...,
  [118  84  50]
```

```
  [120  84  50]
  [109  73  42]]

 ...,
 [[208 170  96]
  [201 153  34]
  [198 161  26]
  ...,
  [160 133  70]
  [ 56  31   7]
  [ 53  34  20]]

 [[180 139  96]
  [173 123  42]
  [186 144  30]
  ...,
  [184 148  94]
  [ 97  62  34]
  [ 83  53  34]]

 [[177 144 116]
  [168 129  94]
  [179 142  87]
  ...,
  [216 184 140]
  [151 118  84]
  [123  92  72]]]
Value of the first element of Y_train:
[6]
X can be converted back to original images via utility function:


/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:23: DeprecationWarning: `toimage` i
`toimage` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use Pillow's ``Image.fromarray`` directly instead.
```