PS8

January 23, 2023

1 Simplified AlexNet implementation

1.1 Loading CIFAR10

[1]:
```
import tensorflow as tf
import tensorflow.keras as keras
```

[2]:
```
from tensorflow.keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 266s 2us/step
```

[3]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
(50000, 1)
(10000, 1)
```

[4]:
```
# normalization
X_train, X_test = X_train/255.0, X_test/255.0
```

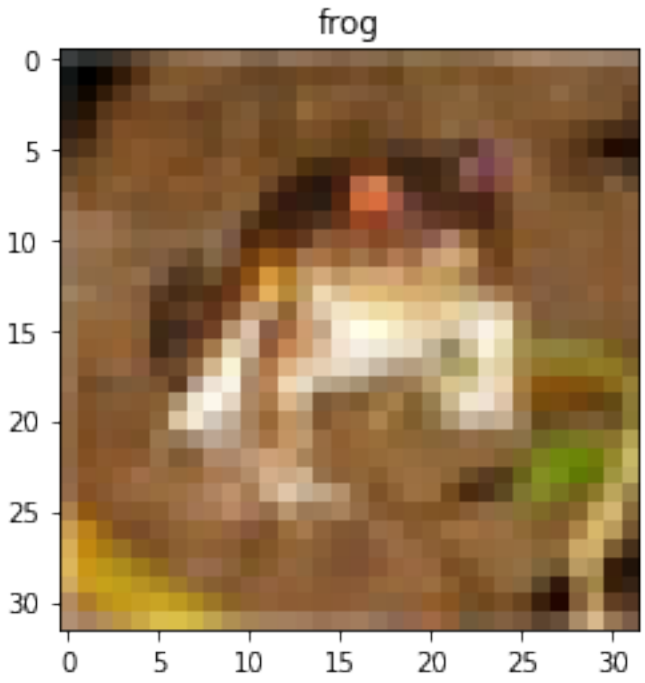
1.2 Data visualization

[5]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(5,5), dpi=150)
plt.imshow(X_train[0])
plt.axis('off')
plt.show()
```

1.3 Data visualization with legend

[6]:
```
import matplotlib.pyplot as plt

label_dict = {0: 'airplane', 1: 'automobile', 2: 'bird',
              3: 'cat',      4: 'deer',       5: 'dog',
              6: 'frog',     7: 'horse',      8: 'ship',
              9: 'truck'}
print(y_train[0])
print(label_dict[6])
plt.imshow(X_train[0])
plt.title(label_dict[6])
plt.show()
```

```
[6]
frog
```



1.4 Construct a model

[7]:
```
## Simplifed Alexnet

from keras.models import Sequential
from keras.layers import Flatten, Dense
from keras.layers import Conv2D
from keras.layers import MaxPool2D

model_alexnet = Sequential()

# 1st stack: [conv]+[ReLU]+[pool]
model_alexnet.add(Conv2D(input_shape = (32,32,3),
                         kernel_size = (3,3),
                         strides=(1,1),
                         filters=48,
                         padding='same',
                         activation='relu'))
model_alexnet.add(MaxPool2D(pool_size = (2,2),
                            strides = (2,2),
                            padding = 'valid'))

# 2nd stack: [conv]+[ReLU]+[pool]
model_alexnet.add(Conv2D(kernel_size = (3,3),
```

```
                          strides=(1,1),
                          filters=96,
                          padding='same',
                          activation='relu'))
model_alexnet.add(MaxPool2D(pool_size = (2,2),
                            strides = (2,2),
                            padding = 'valid'))

# 3rd stack: [conv]+[ReLU]
model_alexnet.add(Conv2D(kernel_size = (3,3),
                          strides=(1,1),
                          filters=192,
                          padding='same',
                          activation='relu'))

# 4th stack: [conv]+[ReLU]
model_alexnet.add(Conv2D(kernel_size = (3,3),
                          strides=(1,1),
                          filters=192,
                          padding='same',
                          activation='relu'))

# 5th stack: [conv]+[ReLU]+[pool]
model_alexnet.add(Conv2D(kernel_size = (3,3),
                          strides=(1,1),
                          filters=256,
                          padding='same',
                          activation='relu'))
model_alexnet.add(MaxPool2D(pool_size = (2,2),
                            strides = (2,2),
                            padding = 'valid'))

# 6th layer: 512 fully connected
model_alexnet.add(Flatten())
model_alexnet.add(Dense(512,activation='relu'))

# 7th layer: 256 fully connected
model_alexnet.add(Dense(256,activation='relu'))

# 8th layer: 10 output
model_alexnet.add(Dense(10,activation='softmax'))

model_alexnet.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
```

```
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 48)        1344

 max_pooling2d (MaxPooling2D  (None, 16, 16, 48)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 96)        41568

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 96)         0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 192)         166080

 conv2d_3 (Conv2D)           (None, 8, 8, 192)         331968

 conv2d_4 (Conv2D)           (None, 8, 8, 256)         442624

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 256)        0
 2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 512)               2097664

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 10)                2570

=================================================================
Total params: 3,215,146
Trainable params: 3,215,146
Non-trainable params: 0
_________________________________________________________________
```

1.5 Compile

[8]:
```
from tensorflow.keras.optimizers import Adam

opt = Adam(learning_rate = 0.001,
           beta_1 = 0.9,
           beta_2 = 0.999)

model_alexnet.compile(optimizer = opt,
                      loss='sparse_categorical_crossentropy',
                      metrics=['acc'])
```

1.6 Training

[9]:
```
# Training
model_alexnet.fit(X_train,y_train,epochs=20)
```

```
Epoch 1/20
1563/1563 [==============================] - 66s 42ms/step - loss: 1.6316 - acc:
0.3868
Epoch 2/20
1563/1563 [==============================] - 64s 41ms/step - loss: 1.1409 - acc:
0.5900
Epoch 3/20
1563/1563 [==============================] - 65s 42ms/step - loss: 0.9297 - acc:
0.6716
Epoch 4/20
1563/1563 [==============================] - 66s 42ms/step - loss: 0.7896 - acc:
0.7206
Epoch 5/20
1563/1563 [==============================] - 66s 42ms/step - loss: 0.6829 - acc:
0.7607
Epoch 6/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.5896 - acc:
0.7906
Epoch 7/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.5070 - acc:
0.8206
Epoch 8/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.4409 - acc:
0.8432
Epoch 9/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.3736 - acc:
0.8677
Epoch 10/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.3351 - acc:
0.8806
Epoch 11/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.2887 - acc:
0.8982
Epoch 12/20
1563/1563 [==============================] - 68s 44ms/step - loss: 0.2606 - acc:
0.9092
Epoch 13/20
1563/1563 [==============================] - 70s 45ms/step - loss: 0.2367 - acc:
0.9177
Epoch 14/20
1563/1563 [==============================] - 68s 44ms/step - loss: 0.2218 - acc:
0.9236
Epoch 15/20
```

```
1563/1563 [==============================] - 66s 43ms/step - loss: 0.1959 - acc:
0.9318
Epoch 16/20
1563/1563 [==============================] - 67s 43ms/step - loss: 0.1940 - acc:
0.9323
Epoch 17/20
1563/1563 [==============================] - 90s 57ms/step - loss: 0.1749 - acc:
0.9402
Epoch 18/20
1563/1563 [==============================] - 104s 66ms/step - loss: 0.1678 -
acc: 0.9435
Epoch 19/20
1563/1563 [==============================] - 105s 67ms/step - loss: 0.1667 -
acc: 0.9442
Epoch 20/20
1563/1563 [==============================] - 108s 69ms/step - loss: 0.1524 -
acc: 0.9480
```

[9]:
```
<keras.callbacks.History at 0x2a50e880df0>
```

[10]:
```
# Evaluation
test_performance = model_alexnet.evaluate(X_test,y_test)
print(test_performance)
```

```
313/313 [==============================] - 7s 20ms/step - loss: 1.5235 - acc:
0.7277
[1.523450493812561, 0.7276999950408936]
```