IE 345 - K “Introduction to Deep Learning: Fundamentals Concepts”

Prof. Yuzo

Build a Convolutional Neural Network.

pg. 92 - 100

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

import keras
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
import os
```

```
Using TensorFlow backend.
```

In [2]:

```
# Set Hyperparameters

batch_size = 32
num_classes = 10
epochs = 25
data_augmentation = True
num_predictions = 20
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'keras_cifar10_trained_model.h5'
```

In [3]:

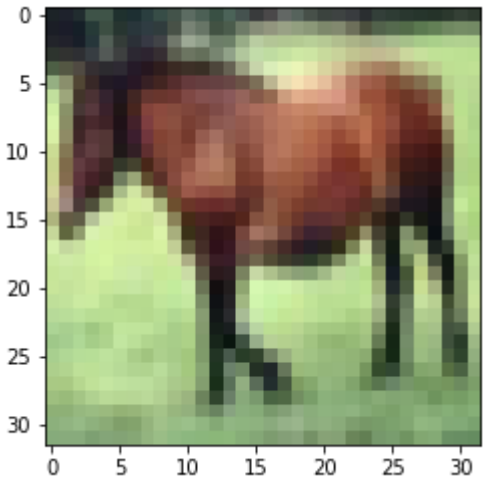
```
#Load Data and split into train and test sets

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train.shape:', x_train.shape)
print(x_train.shape[0], 'Train samples')
print(x_test.shape[0], 'Test samples')
```

```
x_train.shape: (50000, 32, 32, 3)
50000 Train samples
10000 Test samples
```

In [4]:

```
# Display image
sample_image = x_train[7]
plt.imshow(sample_image)
plt.show()
```



In [5]:

```
# Convert class vectors to binary class matrices

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

In [6]:

```
# Describe model using Keras sequential API
model = Sequential()
model.add(Conv2D(32, (3,3), padding='same', input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 30, 30, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
activation_4 (Activation)    (None, 13, 13, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1180160
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_6 (Activation)    (None, 10)                0
=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
# Initiate RMSprop optimizer
opt = keras.optimizers.rmsprop(lr=0.001, decay=1e-6)

#Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
             metrics=['accuracy'])
```

In [8]:

```
# Normalize image date
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```

In [15]:

```
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True)
else:
    print('Using real-time data augmentation.')
    datagen = ImageDataGenerator(featurewise_center=False,
                                 samplewise_center=False,
                                 featurewise_std_normalization=False,
                                 samplewise_std_normalization=False,
                                 zca_whitening=False,
                                 rotation_range=0,
                                 width_shift_range=0.1,
                                 height_shift_range=0.1,
                                 horizontal_flip=True,
                                 vertical_flip=False)

    datagen.fit(x_train)
    model.fit_generator(datagen.flow(x_train, y_train,
                                     batch_size=batch_size),
                        epochs=epochs,
                        validation_data=(x_test,y_test),
                        workers=4,
                        steps_per_epoch=5,)
```

```
Using real-time data augmentation.
Epoch 1/25
5/5 [==============================] - 11s 2s/step - loss: 1.5100 - acc:
0.4688 - val_loss: 1.4094 - val_acc: 0.4960
Epoch 2/25
5/5 [==============================] - 11s 2s/step - loss: 1.4071 - acc:
0.4562 - val_loss: 1.4543 - val_acc: 0.4824
Epoch 3/25
5/5 [==============================] - 11s 2s/step - loss: 1.3071 - acc:
0.5312 - val_loss: 1.5152 - val_acc: 0.4768
Epoch 4/25
5/5 [==============================] - 11s 2s/step - loss: 1.7368 - acc:
0.3875 - val_loss: 1.3567 - val_acc: 0.5066
Epoch 5/25
5/5 [==============================] - 11s 2s/step - loss: 1.3678 - acc:
0.4937 - val_loss: 1.4067 - val_acc: 0.4862
Epoch 6/25
5/5 [==============================] - 11s 2s/step - loss: 1.3066 - acc:
0.5625 - val_loss: 1.5283 - val_acc: 0.4592
Epoch 7/25
5/5 [==============================] - 12s 2s/step - loss: 1.5743 - acc:
0.4562 - val_loss: 1.3478 - val_acc: 0.5064
Epoch 8/25
5/5 [==============================] - 12s 2s/step - loss: 1.3666 - acc:
0.5312 - val_loss: 1.3678 - val_acc: 0.5049
Epoch 9/25
5/5 [==============================] - 12s 2s/step - loss: 1.6838 - acc:
0.4125 - val_loss: 1.4942 - val_acc: 0.4463
Epoch 10/25
5/5 [==============================] - 12s 2s/step - loss: 1.5408 - acc:
0.4625 - val_loss: 1.2973 - val_acc: 0.5378
Epoch 11/25
5/5 [==============================] - 12s 2s/step - loss: 1.3643 - acc:
0.4875 - val_loss: 1.3296 - val_acc: 0.5244
Epoch 12/25
5/5 [==============================] - 12s 2s/step - loss: 1.5870 - acc:
0.4500 - val_loss: 1.4043 - val_acc: 0.5052
Epoch 13/25
5/5 [==============================] - 12s 2s/step - loss: 1.4782 - acc:
0.4500 - val_loss: 1.7269 - val_acc: 0.4016
Epoch 14/25
5/5 [==============================] - 12s 2s/step - loss: 1.5164 - acc:
0.4188 - val_loss: 1.4053 - val_acc: 0.4978
Epoch 15/25
5/5 [==============================] - 12s 2s/step - loss: 1.5264 - acc:
0.4438 - val_loss: 1.4160 - val_acc: 0.4965
Epoch 16/25
5/5 [==============================] - 12s 2s/step - loss: 1.5124 - acc:
0.4375 - val_loss: 1.4340 - val_acc: 0.4988
Epoch 17/25
5/5 [==============================] - 12s 2s/step - loss: 1.6696 - acc:
0.3937 - val_loss: 1.3804 - val_acc: 0.4989
Epoch 18/25
5/5 [==============================] - 12s 2s/step - loss: 1.4861 - acc:
0.4500 - val_loss: 1.3941 - val_acc: 0.4964
Epoch 19/25
5/5 [==============================] - 12s 2s/step - loss: 1.4032 - acc:
0.5000 - val_loss: 1.3721 - val_acc: 0.4965
Epoch 20/25
5/5 [==============================] - 12s 2s/step - loss: 1.5003 - acc:
0.5000 - val_loss: 1.3384 - val_acc: 0.5104
```

```
Epoch 21/25
5/5 [==============================] - 12s 2s/step - loss: 1.5514 - acc:
0.4188 - val_loss: 1.4596 - val_acc: 0.4770
Epoch 22/25
5/5 [==============================] - 12s 2s/step - loss: 1.4399 - acc:
0.4937 - val_loss: 1.3241 - val_acc: 0.5183
Epoch 23/25
5/5 [==============================] - 12s 2s/step - loss: 1.4046 - acc:
0.4875 - val_loss: 1.3720 - val_acc: 0.5154
Epoch 24/25
5/5 [==============================] - 12s 2s/step - loss: 1.6158 - acc:
0.4875 - val_loss: 1.4696 - val_acc: 0.4816
Epoch 25/25
5/5 [==============================] - 12s 2s/step - loss: 1.7047 - acc:
0.4188 - val_loss: 1.3177 - val_acc: 0.5218
```

In [16]:

```
# Save model and weights for future use

if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
model_path = os.path.join(save_dir, model_name)
model.save(model_path)
print('Saved trained model at %s ' % model_path)
```

```
Saved trained model at C:\Users\pablo\Desktop\IE345_DeepLearning\saved_mod
els\keras_cifar10_trained_model.h5
```

In [17]:

```
#Score trained model
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss: ', scores[0])
print('Test accuracy: ', scores[1])
```

```
10000/10000 [==============================] - 12s 1ms/step
Test loss:  1.3176647438049316
Test accuracy:  0.5218
```

Pablo David Minango Negrete

$pablodavid218@gmail.com$