3027020 - Noah Kuntner - Bemacs

I import from Keras' datasets the Cifar10 dataset, from Keras' models the sequential model and I am using the 2D Convolutional Layer Conv2D.

Additionally imports include from Keras' layers MaxPooling2D, Dense, Dropout, Activation and Flatten

In [1]:
```
from __future__ import print_function
import keras
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Activation, Flatte
from keras.preprocessing.image import ImageDataGenerator
import os

epochs = 15
batch_size = 16
classes = 10
save_dir = os.path.join("C:/Users/noah_/Thesis", "saved_models")
model_name = "cifar10_keras.h5"
```

In [2]:
```
save_dir
```

Out[2]:
```
'C:/Users/noah_/Thesis\\saved_models'
```

Followingly, the dataset is split between training and test dataset.

Then I print out the shape of X and the number of train and test samples.

In [3]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

print("x_train shape: {}".format(x_train.shape))
print("{} train samples".format(x_train.shape[0],))
print("{} test samples".format(x_test.shape[0], ))

# We need to transform the variables into binary class matrices
y_train = keras.utils.to_categorical(y_train, classes)
y_test = keras.utils.to_categorical(y_test, classes)
```

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

In [4]:
```
print(classes)
```

```
10
```

Now we create the Convolutional neural network and add four layers.

In [6]:
```
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), padding="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

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

model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(512))
model.add(Activation("sigmoid"))
model.add(Dropout(0.5))
model.add(Dense(classes))
model.add(Activation('softmax'))
```

Now I, finally, train the model

In [7]:
```
#I've set the loss function to categorical_crossentropy and the optimizer to
#stochastic optimizer Adam
model.compile(loss='categorical_crossentropy', optimizer="adam",
              metrics=["accuracy"])

x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
x_train = x_train / 200
x_test = x_test / 200


model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
              validation_data=(x_test, y_test), shuffle=True)
```

```
Epoch 1/15
3125/3125 [==============================] - 109s 35ms/step - loss: 1.6753 -
accuracy: 0.3796 - val_loss: 1.2889 - val_accuracy: 0.5314
Epoch 2/15
3125/3125 [==============================] - 110s 35ms/step - loss: 1.2972 -
accuracy: 0.5283 - val_loss: 1.0568 - val_accuracy: 0.6153
Epoch 3/15
3125/3125 [==============================] - 109s 35ms/step - loss: 1.1650 -
accuracy: 0.5841 - val_loss: 1.0011 - val_accuracy: 0.6425
Epoch 4/15
3125/3125 [==============================] - 104s 33ms/step - loss: 1.0806 -
accuracy: 0.6195 - val_loss: 1.0386 - val_accuracy: 0.6458
Epoch 5/15
3125/3125 [==============================] - 105s 34ms/step - loss: 1.0400 -
accuracy: 0.6340 - val_loss: 0.8736 - val_accuracy: 0.6956
Epoch 6/15
3125/3125 [==============================] - 107s 34ms/step - loss: 0.9972 -
accuracy: 0.6498 - val_loss: 0.8450 - val_accuracy: 0.7003
Epoch 7/15
3125/3125 [==============================] - 109s 35ms/step - loss: 0.9727 -
accuracy: 0.6593 - val_loss: 0.8197 - val_accuracy: 0.7170
Epoch 8/15
3125/3125 [==============================] - 108s 35ms/step - loss: 0.9545 -
accuracy: 0.6624 - val_loss: 0.9249 - val_accuracy: 0.6850
Epoch 9/15
3125/3125 [==============================] - 110s 35ms/step - loss: 0.9479 -
accuracy: 0.6690 - val_loss: 0.7917 - val_accuracy: 0.7225
Epoch 10/15
3125/3125 [==============================] - 111s 36ms/step - loss: 0.9329 -
accuracy: 0.6728 - val_loss: 0.7875 - val_accuracy: 0.7227
Epoch 11/15
3125/3125 [==============================] - 111s 36ms/step - loss: 0.9276 -
accuracy: 0.6771 - val_loss: 0.7994 - val_accuracy: 0.7189
Epoch 12/15
3125/3125 [==============================] - 116s 37ms/step - loss: 0.9300 -
accuracy: 0.6759 - val_loss: 0.7902 - val_accuracy: 0.7269
Epoch 13/15
3125/3125 [==============================] - 117s 38ms/step - loss: 0.9176 -
accuracy: 0.6814 - val_loss: 0.7810 - val_accuracy: 0.7270
Epoch 14/15
3125/3125 [==============================] - 110s 35ms/step - loss: 0.9285 -
```

```
accuracy: 0.6761 - val_loss: 0.7661 - val_accuracy: 0.7385
Epoch 15/15
3125/3125 [==============================] - 108s 34ms/step - loss: 0.9353 -
accuracy: 0.6741 - val_loss: 0.7545 - val_accuracy: 0.7362
```

Out[7]: <tensorflow.python.keras.callbacks.History at 0x23a80060448>