In [1]:

```
from keras.datasets import cifar10
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import SGD, Adam, RMSprop
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

In [2]:

```
# CIFAR-10 содержит 60K изображений 32*32*3 канала цвета
IMG_CHANNELS = 3
IMG_ROWS = 32
IMG_COLS = 32

BATCH_SIZE = 128
NB_EPOCH = 25
NB_CLASSES = 10
VERBOSE = 1
VALIDATION_SPLIT = 0.2
OPTIM = RMSprop()
```

In [3]:

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

In [4]:

```
# Пребобразуем к категориальному виду
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)
```

In [5]:

```
# Преобразуем к формату с плавующей точкой и нормируем к диапазоу (0,1)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

In [6]:

```
# Описываем нейросеть
model = Sequential()
model.add(Conv2D(32, (7, 7),padding='same',
    input_shape=(IMG_ROWS,IMG_COLS, IMG_CHANNELS)))
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(NB_CLASSES))
model.add(Activation('softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        4736      
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 32)        0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 16, 16, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               4194816   
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130      
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0         
=================================================================
Total params: 4,204,682
Trainable params: 4,204,682
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
# Обучение модели
model.compile(loss='categorical_crossentropy', optimizer=OPTIM,
    metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=BATCH_SIZE,
    epochs=NB_EPOCH, validation_split=VALIDATION_SPLIT,
    verbose=VERBOSE)

score = model.evalaute(X_test, Y_test,
    batch_size=BATCH_SIZE, verbose=VERBOSE)
print("Test score:", score[0])
print("Test accuracy", score[1])
```

```
Train on 40000 samples, validate on 10000 samples
Epoch 1/25
40000/40000 [==============================] - 98s 2ms/step - loss:
1.7577 - accuracy: 0.3796 - val_loss: 1.5047 - val_accuracy: 0.4682
Epoch 2/25
40000/40000 [==============================] - 86s 2ms/step - loss:
1.4040 - accuracy: 0.5043 - val_loss: 1.2681 - val_accuracy: 0.5618
Epoch 3/25
40000/40000 [==============================] - 90s 2ms/step - loss:
1.2474 - accuracy: 0.5641 - val_loss: 1.2069 - val_accuracy: 0.5817
Epoch 4/25
40000/40000 [==============================] - 90s 2ms/step - loss:
1.1500 - accuracy: 0.5967 - val_loss: 1.1313 - val_accuracy: 0.6071
Epoch 5/25
40000/40000 [==============================] - 93s 2ms/step - loss:
1.0650 - accuracy: 0.6273 - val_loss: 1.2014 - val_accuracy: 0.5931
Epoch 6/25
40000/40000 [==============================] - 90s 2ms/step - loss:
1.0054 - accuracy: 0.6520 - val_loss: 1.1114 - val_accuracy: 0.6234
Epoch 7/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.9395 - accuracy: 0.6741 - val_loss: 1.0953 - val_accuracy: 0.6312
Epoch 8/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.8800 - accuracy: 0.6952 - val_loss: 1.0577 - val_accuracy: 0.6453
Epoch 9/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.8293 - accuracy: 0.7139 - val_loss: 1.0459 - val_accuracy: 0.6546
Epoch 10/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.7918 - accuracy: 0.7283 - val_loss: 1.2336 - val_accuracy: 0.6196
Epoch 11/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.7478 - accuracy: 0.7412 - val_loss: 1.0472 - val_accuracy: 0.6636
Epoch 12/25
40000/40000 [==============================] - 90s 2ms/step - loss:
0.7103 - accuracy: 0.7598 - val_loss: 1.1081 - val_accuracy: 0.6491
Epoch 13/25
40000/40000 [==============================] - 90s 2ms/step - loss:
0.6821 - accuracy: 0.7656 - val_loss: 1.0897 - val_accuracy: 0.6605
Epoch 14/25
40000/40000 [==============================] - 107s 3ms/step - loss:
0.6425 - accuracy: 0.7819 - val_loss: 1.0439 - val_accuracy: 0.6735
Epoch 15/25
40000/40000 [==============================] - 97s 2ms/step - loss:
0.6186 - accuracy: 0.7901 - val_loss: 1.0737 - val_accuracy: 0.6731
Epoch 16/25
40000/40000 [==============================] - 95s 2ms/step - loss:
0.5928 - accuracy: 0.8004 - val_loss: 1.1242 - val_accuracy: 0.6760
Epoch 17/25
40000/40000 [==============================] - 104s 3ms/step - loss:
0.5712 - accuracy: 0.8056 - val_loss: 1.1773 - val_accuracy: 0.6749
Epoch 18/25
40000/40000 [==============================] - 108s 3ms/step - loss:
0.5479 - accuracy: 0.8164 - val_loss: 1.2662 - val_accuracy: 0.6516
Epoch 19/25
40000/40000 [==============================] - 105s 3ms/step - loss:
0.5274 - accuracy: 0.8224 - val_loss: 1.1856 - val_accuracy: 0.6675
Epoch 20/25
40000/40000 [==============================] - 108s 3ms/step - loss:
0.5200 - accuracy: 0.8265 - val_loss: 1.2017 - val_accuracy: 0.6727
```

```
Epoch 21/25
40000/40000 [==============================] - 92s 2ms/step - loss:
0.5006 - accuracy: 0.8325 - val_loss: 1.2171 - val_accuracy: 0.6750
Epoch 22/25
40000/40000 [==============================] - 89s 2ms/step - loss:
0.4796 - accuracy: 0.8400 - val_loss: 1.2669 - val_accuracy: 0.6746
Epoch 23/25
40000/40000 [==============================] - 91s 2ms/step - loss:
0.4687 - accuracy: 0.8432 - val_loss: 1.2747 - val_accuracy: 0.6680
Epoch 24/25
40000/40000 [==============================] - 90s 2ms/step - loss:
0.4622 - accuracy: 0.8482 - val_loss: 1.3031 - val_accuracy: 0.6636
Epoch 25/25
40000/40000 [==============================] - 90s 2ms/step - loss:
0.4543 - accuracy: 0.8528 - val_loss: 1.2734 - val_accuracy: 0.6793


---------------------------------------------------------------------
-------
AttributeError                            Traceback (most recent cal
l last)
<ipython-input-7-d4eb89471226> in <module>
      6     verbose=VERBOSE)
      7
----> 8 score = model.evalaute(X_test, Y_test,
      9     batch_size=BATCH_SIZE, verbose=VERBOSE)
     10 print("Test score:", score[0])

AttributeError: 'Sequential' object has no attribute 'evalaute'
```

In [ ]:

```
model_json = model.to_json()
open('cifar10_arch.json','w').write(model_json)
model,save_weights('cifar10_weights.h5', overwrite = True)
```

In [ ]:

```

```

In [ ]:

```

```