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, (5, 5),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)        2432      
_________________________________________________________________
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: 4,202,378
Trainable params: 4,202,378
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 [==============================] - 82s 2ms/step - loss:
1.7109 - accuracy: 0.3897 - val_loss: 1.4061 - val_accuracy: 0.5256
Epoch 2/25
40000/40000 [==============================] - 71s 2ms/step - loss:
1.3576 - accuracy: 0.5203 - val_loss: 1.4311 - val_accuracy: 0.5028
Epoch 3/25
40000/40000 [==============================] - 68s 2ms/step - loss:
1.2105 - accuracy: 0.5746 - val_loss: 1.1724 - val_accuracy: 0.5891
Epoch 4/25
40000/40000 [==============================] - 81s 2ms/step - loss:
1.1135 - accuracy: 0.6103 - val_loss: 1.3464 - val_accuracy: 0.5396
Epoch 5/25
40000/40000 [==============================] - 72s 2ms/step - loss:
1.0331 - accuracy: 0.6430 - val_loss: 1.1068 - val_accuracy: 0.6157
Epoch 6/25
40000/40000 [==============================] - 76s 2ms/step - loss:
0.9686 - accuracy: 0.6654 - val_loss: 1.0594 - val_accuracy: 0.6388
Epoch 7/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.9149 - accuracy: 0.6848 - val_loss: 1.1010 - val_accuracy: 0.6305
Epoch 8/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.8554 - accuracy: 0.7060 - val_loss: 1.1087 - val_accuracy: 0.6324
Epoch 9/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.8193 - accuracy: 0.7174 - val_loss: 1.0242 - val_accuracy: 0.6615
Epoch 10/25
40000/40000 [==============================] - 74s 2ms/step - loss:
0.7729 - accuracy: 0.7350 - val_loss: 1.0352 - val_accuracy: 0.6586
Epoch 11/25
40000/40000 [==============================] - 77s 2ms/step - loss:
0.7373 - accuracy: 0.7458 - val_loss: 0.9997 - val_accuracy: 0.6868
Epoch 12/25
40000/40000 [==============================] - 78s 2ms/step - loss:
0.6991 - accuracy: 0.7570 - val_loss: 1.0427 - val_accuracy: 0.6739
Epoch 13/25
40000/40000 [==============================] - 79s 2ms/step - loss:
0.6649 - accuracy: 0.7740 - val_loss: 1.0667 - val_accuracy: 0.6664
Epoch 14/25
40000/40000 [==============================] - 75s 2ms/step - loss:
0.6385 - accuracy: 0.7806 - val_loss: 1.1297 - val_accuracy: 0.6552
Epoch 15/25
40000/40000 [==============================] - 71s 2ms/step - loss:
0.6076 - accuracy: 0.7898 - val_loss: 1.1277 - val_accuracy: 0.6642
Epoch 16/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.5864 - accuracy: 0.8012 - val_loss: 1.0775 - val_accuracy: 0.6755
Epoch 17/25
40000/40000 [==============================] - 72s 2ms/step - loss:
0.5704 - accuracy: 0.8055 - val_loss: 1.2885 - val_accuracy: 0.6405
Epoch 18/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.5447 - accuracy: 0.8147 - val_loss: 1.0441 - val_accuracy: 0.6879
Epoch 19/25
40000/40000 [==============================] - 72s 2ms/step - loss:
0.5283 - accuracy: 0.8220 - val_loss: 1.1188 - val_accuracy: 0.6693
Epoch 20/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.5054 - accuracy: 0.8282 - val_loss: 1.2328 - val_accuracy: 0.6439
```

```
Epoch 21/25
40000/40000 [==============================] - 71s 2ms/step - loss:
0.4890 - accuracy: 0.8337 - val_loss: 1.1238 - val_accuracy: 0.6808
Epoch 22/25
40000/40000 [==============================] - 74s 2ms/step - loss:
0.4726 - accuracy: 0.8407 - val_loss: 1.4387 - val_accuracy: 0.6544
Epoch 23/25
40000/40000 [==============================] - 73s 2ms/step - loss:
0.4651 - accuracy: 0.8443 - val_loss: 1.2495 - val_accuracy: 0.6839
Epoch 24/25
40000/40000 [==============================] - 70s 2ms/step - loss:
0.4563 - accuracy: 0.8458 - val_loss: 1.1859 - val_accuracy: 0.6819
Epoch 25/25
40000/40000 [==============================] - 71s 2ms/step - loss:
0.4407 - accuracy: 0.8509 - val_loss: 1.3021 - val_accuracy: 0.6849


--------------------------------------------------------------------
-------
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 [ ]:

```
```