In [2]:
```
! shred -u setup_google_colab.py
! wget https://raw.githubusercontent.com/hse-aml/intro-to-dl/master/setup_goo
import setup_google_colab
# please, uncomment the week you're working on
# setup_google_colab.setup_week1()
# setup_google_colab.setup_week2()
# setup_google_colab.setup_week2_honor()
setup_google_colab.setup_week3()
# setup_google_colab.setup_week4()
# setup_google_colab.setup_week5()
# setup_google_colab.setup_week6()
# set tf 1.x for colab
# set tf 1.x for colab
```

```
--2022-04-11 20:41:32--  https://raw.githubusercontent.com/hse-aml/intro-to-d
l/master/setup_google_colab.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.10
8.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.10
8.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3636 (3.6K) [text/plain]
Saving to: 'setup_google_colab.py'

setup_google_colab. 100%[===================>]   3.55K  --.-KB/s    in 0s

2022-04-11 20:41:32 (33.0 MB/s) - 'setup_google_colab.py' saved [3636/3636]

**************************************************
102flowers.tgz
**************************************************
imagelabels.mat
**************************************************
inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
**************************************************
cifar-10-batches-py.tar.gz
**************************************************
mnist.npz
```

Your first CNN on CIFAR-10

In this task you will:

- define your first CNN architecture for CIFAR-10 dataset
- train it from scratch
- visualize learnt filters

CIFAR-10 dataset contains 32x32 color images from 10 classes: **airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck**:

Import stuff

In [3]:
```
import sys
sys.path.append("..")
import grading
import download_utils
```

In [4]:
```
# !!! remember to clear session/graph if you rebuild your graph to avoid out-
```

In [5]:
```
download_utils.link_all_keras_resources()
```

In [6]:
```
!pip uninstall -y tensorflow
!pip install keras==2.0.6
!pip install tensorflow==1.15.0
import tensorflow as tf
import keras
from keras import backend as K
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
print(tf.__version__)
print(keras.__version__)
import grading_utils
import keras_utils
from keras_utils import reset_tf_session
```

```
Found existing installation: tensorflow 1.15.0
Uninstalling tensorflow-1.15.0:
  Successfully uninstalled tensorflow-1.15.0
Requirement already satisfied: keras==2.0.6 in /usr/local/lib/python3.7/dist-p
ackages (2.0.6)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from keras==2.0.6) (1.16.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-package
s (from keras==2.0.6) (3.13)
Requirement already satisfied: theano in /usr/local/lib/python3.7/dist-package
s (from keras==2.0.6) (1.0.5)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.7/dist-pa
ckages (from theano->keras==2.0.6) (1.4.1)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-p
ackages (from theano->keras==2.0.6) (1.21.5)
Collecting tensorflow==1.15.0
  Using cached tensorflow-1.15.0-cp37-cp37m-manylinux2010_x86_64.whl (412.3 M
B)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.7/dist
-packages (from tensorflow==1.15.0) (1.0.0)
Requirement already satisfied: tensorboard<1.16.0,>=1.15.0 in /usr/local/lib/p
ython3.7/dist-packages (from tensorflow==1.15.0) (1.15.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-pa
ckages (from tensorflow==1.15.0) (1.16.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/py
thon3.7/dist-packages (from tensorflow==1.15.0) (1.1.2)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/di
st-packages (from tensorflow==1.15.0) (1.1.0)
Requirement already satisfied: tensorflow-estimator==1.15.1 in /usr/local/lib/
python3.7/dist-packages (from tensorflow==1.15.0) (1.15.1)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-pa
ckages (from tensorflow==1.15.0) (0.37.1)
```

```
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==1.15.0) (1.14.0)
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.
7/dist-packages (from tensorflow==1.15.0) (0.2.0)
Requirement already satisfied: gast==0.2.2 in /usr/local/lib/python3.7/dist-pa
ckages (from tensorflow==1.15.0) (0.2.2)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==1.15.0) (1.44.0)
Requirement already satisfied: keras-applications>=1.0.8 in /usr/local/lib/pyt
hon3.7/dist-packages (from tensorflow==1.15.0) (1.0.8)
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.7/
dist-packages (from tensorflow==1.15.0) (1.21.5)
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.7/dist-p
ackages (from tensorflow==1.15.0) (0.8.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/d
ist-packages (from tensorflow==1.15.0) (3.3.0)
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/python3.7/dis
t-packages (from tensorflow==1.15.0) (3.17.3)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages
(from keras-applications>=1.0.8->tensorflow==1.15.0) (2.10.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/
dist-packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.0) (57.4.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/d
ist-packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.0) (1.0.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dis
t-packages (from tensorboard<1.16.0,>=1.15.0->tensorflow==1.15.0) (3.3.6)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/pytho
n3.7/dist-packages (from markdown>=2.6.8->tensorboard<1.16.0,>=1.15.0->tensorf
low==1.15.0) (4.11.3)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pack
ages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<1.16.0,>=1.1
5.0->tensorflow==1.15.0) (3.7.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/pyth
on3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboar
d<1.16.0,>=1.15.0->tensorflow==1.15.0) (3.10.0.2)
Installing collected packages: tensorflow
ERROR: pip's dependency resolver does not currently take into account all the
 packages that are installed. This behaviour is the source of the following de
pendency conflicts.
kapre 0.3.7 requires tensorflow>=2.0.0, but you have tensorflow 1.15.0 which i
s incompatible.
Successfully installed tensorflow-1.15.0
```

```
Using TensorFlow backend.
1.15.0
2.0.6
```

Fill in your Coursera token and email

To successfully submit your answers to our grader, please fill in your Coursera submission token and email

In [7]:

```
grader = grading.Grader(assignment_key="s1B1I5DuEeeyLAqI7dCYkg",
                        all_parts=["7W4tu", "nQOsg", "96eco"])
```

In [36]:

```
# token expires every 30 min
COURSERA_TOKEN = 'Z8BsPJDNjMo6wrNI' ### YOUR TOKEN HERE
COURSERA_EMAIL = 'e0321294@u.nus.edu'### YOUR EMAIL HERE
```

Load dataset

In [9]:
```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [10]:
```
print("Train samples:", x_train.shape, y_train.shape)
print("Test samples:", x_test.shape, y_test.shape)
```

```
Train samples: (50000, 32, 32, 3) (50000, 1)
Test samples: (10000, 32, 32, 3) (10000, 1)
```

In [11]:
```
NUM_CLASSES = 10
cifar10_classes = ["airplane", "automobile", "bird", "cat", "deer",
                   "dog", "frog", "horse", "ship", "truck"]
```

In [12]:
```
# show random images from train
cols = 8
rows = 2
fig = plt.figure(figsize=(2 * cols - 1, 2.5 * rows - 1))
for i in range(cols):
    for j in range(rows):
        random_index = np.random.randint(0, len(y_train))
        ax = fig.add_subplot(rows, cols, i * rows + j + 1)
        ax.grid('off')
        ax.axis('off')
        ax.imshow(x_train[random_index, :])
        ax.set_title(cifar10_classes[y_train[random_index, 0]])
plt.show()
```

Prepare data

We need to normalize inputs like this:

$$x_{norm} = \frac{x}{255} - 0.5$$

We need to convert class labels to one-hot encoded vectors. Use **keras.utils.to_categorical**.

In [13]:
```
# normalize inputs
x_train2 = x_train/255. - 0.5 ### YOUR CODE HERE
x_test2 = x_test/255. - 0.5 ### YOUR CODE HERE

# convert class labels to one-hot encoded, should have shape (?, NUM_CLASSES)
```

```
y_train2 = keras.utils.to_categorical(y_train, num_classes=NUM_CLASSES)  ###
y_test2 =  keras.utils.to_categorical(y_test, num_classes=NUM_CLASSES) ### YO
```

Define CNN architecture

In [14]:

```
# import necessary building blocks
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dr
from keras.layers.advanced_activations import LeakyReLU
```

Convolutional networks are built from several types of layers:

- Conv2D - performs convolution:
  - **filters**: number of output channels;
  - **kernel_size**: an integer or tuple/list of 2 integers, specifying the width and height of the 2D convolution window;
  - **padding**: padding="same" adds zero padding to the input, so that the output has the same width and height, padding='valid' performs convolution only in locations where kernel and the input fully overlap;
  - **activation**: "relu", "tanh", etc.
  - **input_shape**: shape of input.
- MaxPooling2D - performs 2D max pooling.
- Flatten - flattens the input, does not affect the batch size.
- Dense - fully-connected layer.
- Activation - applies an activation function.
- LeakyReLU - applies leaky relu activation.
- Dropout - applies dropout.

You need to define a model which takes **(None, 32, 32, 3)** input and predicts **(None, 10)** output with probabilities for all classes. **None** in shapes stands for batch dimension.

Simple feed-forward networks in Keras can be defined in the following way:

```
model = Sequential()  # start feed-forward model definition
model.add(Conv2D(..., input_shape=(32, 32, 3)))  # first layer needs
to define "input_shape"

...  # here comes a bunch of convolutional, pooling and dropout
layers

model.add(Dense(NUM_CLASSES))  # the last layer with neuron for each
class
model.add(Activation("softmax"))  # output probabilities
```

Stack **4** convolutional layers with kernel size **(3, 3)** with growing number of filters **(16, 32, 32, 64)**, use "same" padding.

Add **2x2** pooling layer after every 2 convolutional layers (conv-conv-pool scheme).

Use **LeakyReLU** activation with recommended parameter **0.1** for all layers that need it (after convolutional and dense layers):

```
model.add(LeakyReLU(0.1))
```

Add a dense layer with **256** neurons and a second dense layer with **10** neurons for classes. Remember to use **Flatten** layer before first dense layer to reshape input volume into a flat vector!

Add **Dropout** after every pooling layer (**0.25**) and between dense layers (**0.5**).

In [15]:
```
def make_model():
    """
    Define your model architecture here.
    Returns `Sequential` model.
    """
    model = Sequential()

    # CONV 1
    model.add(Conv2D(16, (3, 3), strides = (1, 1), padding="same", name = 'co
    model.add(LeakyReLU(0.1))

    # CONV 2
    model.add(Conv2D(32, (3, 3), strides = (1, 1), padding="same", name = 'co
    model.add(LeakyReLU(0.1))

    # MaxPooling2D 1
    model.add(MaxPooling2D((2, 2), name='max_pool_1'))

    # Dropout
    model.add(Dropout(0.25, noise_shape=None, seed=0))

    # CONV 3
    model.add(Conv2D(32, (3, 3), strides = (1, 1), padding="same", name = 'co
    model.add(LeakyReLU(0.1))

    # CONV 4
    model.add(Conv2D(64, (3, 3), strides = (1, 1), padding="same", name = 'co
    model.add(LeakyReLU(0.1))

    # MaxPooling2D 2
    model.add(MaxPooling2D((2, 2), name='max_pool_2'))

    # Dropout
    model.add(Dropout(0.25, noise_shape=None, seed=0))

    # Flatten
    model.add(Flatten())

    # FC
    model.add(Dense(256, name='fc1'))
    model.add(Dropout(0.5, noise_shape=None, seed=0))

    # FC
    model.add(Dense(NUM_CLASSES))
    model.add(Activation("softmax"))

    ### YOUR CODE HERE

    return model
```

In [16]:
```
# describe model
s = reset_tf_session()  # clear default graph
model = make_model()
model.summary()
```

```
WARNING:tensorflow:From /content/keras_utils.py:68: The name tf.get_default_se
ssion is deprecated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:79: The name tf.reset_default_graph is deprecated. Please
use tf.compat.v1.reset_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:82: The name tf.placeholder is deprecated. Please use tf.
compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:84: The name tf.get_default_graph is deprecated. Please u
se tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /content/keras_utils.py:75: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /content/keras_utils.py:77: The name tf.InteractiveSes
sion is deprecated. Please use tf.compat.v1.InteractiveSession instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:3535: The name tf.random_uniform is deprecated. Please us
e tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:3378: The name tf.nn.max_pool is deprecated. Please use t
f.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:2878: calling dropout (from tensorflow.python.ops.nn_ops)
with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - ke
ep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:1210: calling reduce_prod_v1 (from tensorflow.python.ops.
math_ops) with keep_dims is deprecated and will be removed in a future versio
n.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
```

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 32, 32, 16)	448
leaky_re_lu_1 (LeakyReLU)	(None, 32, 32, 16)	0
conv2 (Conv2D)	(None, 32, 32, 32)	4640
leaky_re_lu_2 (LeakyReLU)	(None, 32, 32, 32)	0
max_pool_1 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_1 (Dropout)	(None, 16, 16, 32)	0
conv3 (Conv2D)	(None, 16, 16, 32)	9248
leaky_re_lu_3 (LeakyReLU)	(None, 16, 16, 32)	0
conv4 (Conv2D)	(None, 16, 16, 64)	18496
leaky_re_lu_4 (LeakyReLU)	(None, 16, 16, 64)	0

```
max_pool_2 (MaxPooling2D)    (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               1048832
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 1,084,234
Trainable params: 1,084,234
Non-trainable params: 0
_________________________________________________________________
```

In [17]:

```
## GRADED PART, DO NOT CHANGE!
# Number of model parameters
grader.set_answer("7W4tu", grading_utils.model_total_params(model))
```

In [18]:

```
# you can make submission with answers so far to check yourself at this stage
grader.submit(COURSERA_EMAIL, COURSERA_TOKEN)
```

```
You used an invalid email or your token may have expired. Please make sure you
have entered all fields correctly. Try generating a new token if the issue sti
ll persists.
```

Train model

Training of your model can take approx. 4-8 minutes per epoch.

During training you should observe the decrease in reported loss on training and validation.

If the loss on training is not decreasing with epochs you should revise your model definition and learning rate.

In [19]:

```
INIT_LR = 5e-3  # initial learning rate
BATCH_SIZE = 32
EPOCHS = 10

s = reset_tf_session()  # clear default graph
# don't call K.set_learning_phase() !!! (otherwise will enable dropout in tra
model = make_model()  # define our model

# prepare model for fitting (loss, optimizer, etc)
model.compile(
    loss='categorical_crossentropy',  # we train 10-way classification
    optimizer=keras.optimizers.adamax(lr=INIT_LR),  # for SGD
    metrics=['accuracy']  # report accuracy during training
)

# scheduler of learning rate (decay with epochs)
def lr_scheduler(epoch):
    return INIT_LR * 0.9 ** epoch

# callback for printing of actual learning rate used by optimizer
```

```
class LrHistory(keras.callbacks.Callback):
    def on_epoch_begin(self, epoch, logs={}):
        print("Learning rate:", K.get_value(model.optimizer.lr))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/optimizer
s.py:697: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.t
rain.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:2745: calling reduce_sum_v1 (from tensorflow.python.ops.m
ath_ops) with keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:2749: The name tf.log is deprecated. Please use tf.math.l
og instead.
```

Training takes approximately **1.5 hours**. You're aiming for ~0.80 validation accuracy.

In [20]:

```
# we will save model checkpoints to continue training in case of kernel death
model_filename = 'cifar.{0:03d}.hdf5'
last_finished_epoch = None

#### uncomment below to continue training from model checkpoint
#### fill `last_finished_epoch` with your latest finished epoch
# from keras.models import load_model
# s = reset_tf_session()
# last_finished_epoch = 7
# model = load_model(model_filename.format(last_finished_epoch))
```

In [21]:

```
# fit model
model.fit(
    x_train2, y_train2,  # prepared data
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=[keras.callbacks.LearningRateScheduler(lr_scheduler),
               LrHistory(),
               keras_utils.TqdmProgressCallback(),
               keras_utils.ModelSaveCallback(model_filename)],
    validation_data=(x_test2, y_test2),
    shuffle=True,
    verbose=0,
    initial_epoch=last_finished_epoch or 0
)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:2289: The name tf.Session is deprecated. Please use tf.co
mpat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow_cor
e/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) i
s deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:879: The name tf.assign_add is deprecated. Please use tf.
compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:602: calling Constant.__init__ (from tensorflow.python.op
s.init_ops) with dtype is deprecated and will be removed in a future version.
```

```
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:866: The name tf.assign is deprecated. Please use tf.comp
at.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:333: The name tf.global_variables is deprecated. Please u
se tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/keras/backend/t
ensorflow_backend.py:341: The name tf.variables_initializer is deprecated. Ple
ase use tf.compat.v1.variables_initializer instead.

Learning rate: 0.005

Epoch 1/10
**************************************************
loss: 1.2982; acc: 0.5380; val_loss: 0.9746; val_acc: 0.6582
Model saved in cifar.000.hdf5
Learning rate: 0.0045

Epoch 2/10
**************************************************
loss: 0.9428; acc: 0.6721; val_loss: 0.8393; val_acc: 0.7042
Model saved in cifar.001.hdf5
Learning rate: 0.00405

Epoch 3/10
**************************************************
loss: 0.8349; acc: 0.7098; val_loss: 0.7665; val_acc: 0.7395
Model saved in cifar.002.hdf5
Learning rate: 0.003645

Epoch 4/10
**************************************************
loss: 0.7664; acc: 0.7343; val_loss: 0.7125; val_acc: 0.7577
Model saved in cifar.003.hdf5
Learning rate: 0.0032805

Epoch 5/10
**************************************************
loss: 0.7100; acc: 0.7545; val_loss: 0.7080; val_acc: 0.7556
Model saved in cifar.004.hdf5
Learning rate: 0.00295245

Epoch 6/10
**************************************************
loss: 0.6677; acc: 0.7686; val_loss: 0.6715; val_acc: 0.7727
Model saved in cifar.005.hdf5
Learning rate: 0.002657205

Epoch 7/10
**************************************************
loss: 0.6348; acc: 0.7796; val_loss: 0.6640; val_acc: 0.7723
Model saved in cifar.006.hdf5
Learning rate: 0.0023914846

Epoch 8/10
**************************************************
loss: 0.5999; acc: 0.7909; val_loss: 0.6425; val_acc: 0.7802
Model saved in cifar.007.hdf5
Learning rate: 0.002152336
```

```
Epoch 9/10
**************************************************
loss: 0.5744; acc: 0.8009; val_loss: 0.6402; val_acc: 0.7862
Model saved in cifar.008.hdf5
Learning rate: 0.0019371024

Epoch 10/10
**************************************************
loss: 0.5520; acc: 0.8097; val_loss: 0.6220; val_acc: 0.7915
Model saved in cifar.009.hdf5
```

Out[21]:
```
<keras.callbacks.History at 0x7f8c084eaad0>
```

In [22]:
```
! pip install h5py==2.10.0 --force-reinstall
```

```
Collecting h5py==2.10.0
  Using cached h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
Collecting six
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting numpy>=1.7
  Using cached numpy-1.21.5-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86
_64.whl (15.7 MB)
Installing collected packages: six, numpy, h5py
  Attempting uninstall: six
    Found existing installation: six 1.16.0
    Uninstalling six-1.16.0:
      Successfully uninstalled six-1.16.0
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.5
    Uninstalling numpy-1.21.5:
      Successfully uninstalled numpy-1.21.5
  Attempting uninstall: h5py
    Found existing installation: h5py 2.10.0
    Uninstalling h5py-2.10.0:
      Successfully uninstalled h5py-2.10.0
ERROR: pip's dependency resolver does not currently take into account all the
 packages that are installed. This behaviour is the source of the following de
pendency conflicts.
tensorflow-probability 0.16.0 requires gast>=0.3.2, but you have gast 0.2.2 wh
ich is incompatible.
kapre 0.3.7 requires tensorflow>=2.0.0, but you have tensorflow 1.15.0 which i
s incompatible.
google-colab 1.0.0 requires six~=1.15.0, but you have six 1.16.0 which is inco
mpatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is
 incompatible.
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.9
which is incompatible.
Successfully installed h5py-2.10.0 numpy-1.21.5 six-1.16.0
```

In [23]:
```
# save weights to file
model.save_weights("weights.h5")
# load weights from file (can call without model.fit)
model.load_weights("weights.h5")
```

Evaluate model

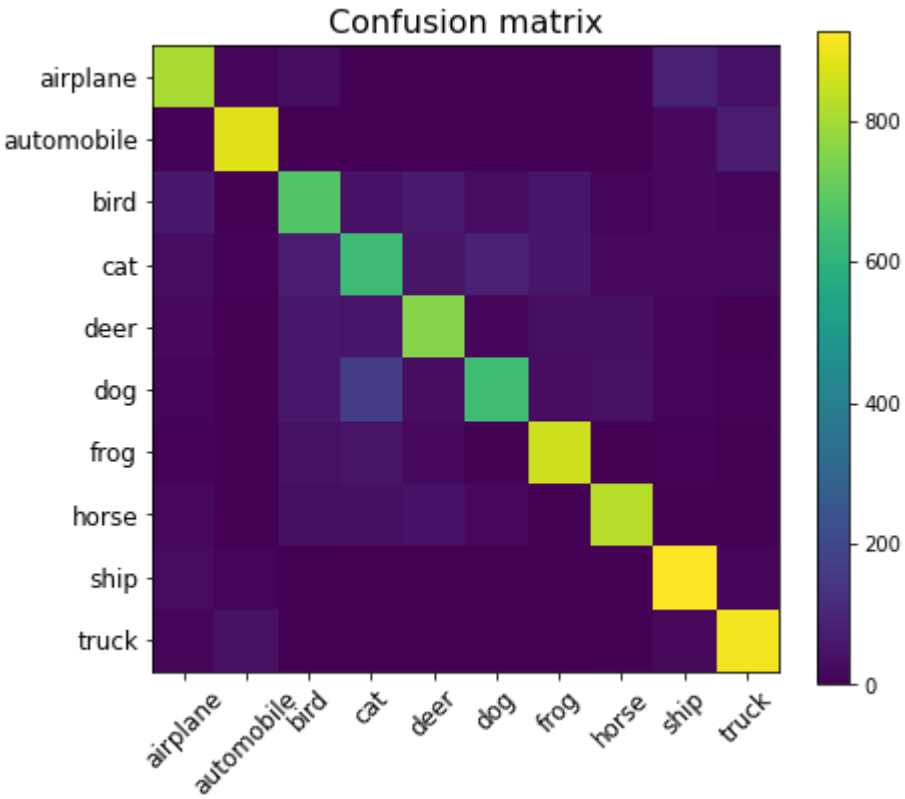
In [24]:
```
# make test predictions
```

```
y_pred_test = model.predict_proba(x_test2)
y_pred_test_classes = np.argmax(y_pred_test, axis=1)
y_pred_test_max_probas = np.max(y_pred_test, axis=1)
```

```
9984/10000 [============================>.] - ETA: 0s
```

In [25]:

```
# confusion matrix and accuracy
from sklearn.metrics import confusion_matrix, accuracy_score
plt.figure(figsize=(7, 6))
plt.title('Confusion matrix', fontsize=16)
plt.imshow(confusion_matrix(y_test, y_pred_test_classes))
plt.xticks(np.arange(10), cifar10_classes, rotation=45, fontsize=12)
plt.yticks(np.arange(10), cifar10_classes, fontsize=12)
plt.colorbar()
plt.show()
print("Test accuracy:", accuracy_score(y_test, y_pred_test_classes))
```




```
Test accuracy: 0.7915
```

In [26]:

```
## GRADED PART, DO NOT CHANGE!
# Accuracy on validation data
grader.set_answer("nQOsg", accuracy_score(y_test, y_pred_test_classes))
```

In [27]:

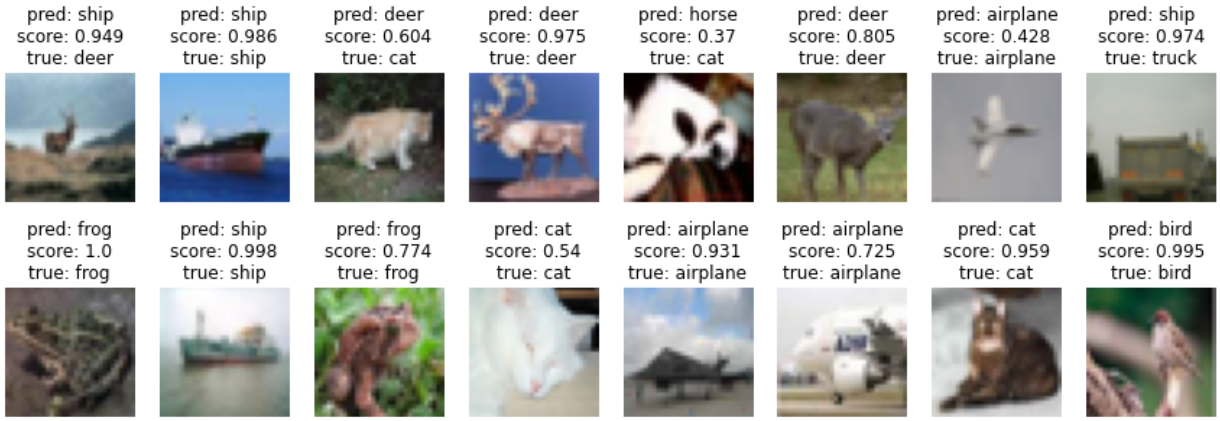
```
# you can make submission with answers so far to check yourself at this stage
grader.submit(COURSERA_EMAIL, COURSERA_TOKEN)
```

```
You used an invalid email or your token may have expired. Please make sure you
have entered all fields correctly. Try generating a new token if the issue sti
ll persists.
```

In [28]:

```
# inspect preditions
cols = 8
rows = 2
fig = plt.figure(figsize=(2 * cols - 1, 3 * rows - 1))
for i in range(cols):
```

```
    for j in range(rows):
        random_index = np.random.randint(0, len(y_test))
        ax = fig.add_subplot(rows, cols, i * rows + j + 1)
        ax.grid('off')
        ax.axis('off')
        ax.imshow(x_test[random_index, :])
        pred_label = cifar10_classes[y_pred_test_classes[random_index]]
        pred_proba = y_pred_test_max_probas[random_index]
        true_label = cifar10_classes[y_test[random_index, 0]]
        ax.set_title("pred: {}\nscore: {:.3}\ntrue: {}".format(
               pred_label, pred_proba, true_label
        ))
plt.show()
```



Visualize maximum stimuli

We want to find input images that provide maximum activations for particular layers of our network.

We will find those maximum stimuli via gradient ascent in image space.

For that task we load our model weights, calculate the layer output gradient with respect to image input and shift input image in that direction.

In [29]:
```
s = reset_tf_session()  # clear default graph
K.set_learning_phase(0)  # disable dropout
model = make_model()
model.load_weights("weights.h5")  # that were saved after model.fit
```

In [30]:
```
# all weights we have
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 32, 32, 16)        448
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 32, 32, 16)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 32, 32, 32)        4640
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 32, 32)        0
_________________________________________________________________
max_pool_1 (MaxPooling2D)    (None, 16, 16, 32)        0
_________________________________________________________________
```

```
dropout_1 (Dropout)          (None, 16, 16, 32)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 16, 16, 32)        9248
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 16, 16, 32)        0
_________________________________________________________________
conv4 (Conv2D)               (None, 16, 16, 64)        18496
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 16, 16, 64)        0
_________________________________________________________________
max_pool_2 (MaxPooling2D)    (None, 8, 8, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4096)              0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               1048832
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                2570
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
=================================================================
Total params: 1,084,234
Trainable params: 1,084,234
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
def find_maximum_stimuli(layer_name, is_conv, filter_index, model, iterations

    def image_values_to_rgb(x):
        # normalize x: center on 0 (np.mean(x_train2)), ensure std is 0.25 (n
        # so that it looks like a normalized image input for our network
        x = (x-np.mean(x_train2))/np.std(x_train2)### YOUR CODE HERE

        # do reverse normalization to RGB values: x = (x_norm + 0.5) * 255
        x = (x + 0.5) * 255### YOUR CODE HERE

        # clip values to [0, 255] and convert to bytes
        x = np.clip(x, 0, 255).astype('uint8')
        return x

    # this is the placeholder for the input image
    input_img = model.input
    img_width, img_height = input_img.shape.as_list()[1:3]

    # find the layer output by name
    layer_output = list(filter(lambda x: x.name == layer_name, model.layers))

    # we build a loss function that maximizes the activation
    # of the filter_index filter of the layer considered
    if is_conv:
        # mean over feature map values for convolutional layer
        loss = K.mean(layer_output[:, :, :, filter_index])
    else:
        loss = K.mean(layer_output[:, filter_index])

    # we compute the gradient of the loss wrt input image
    grads = K.gradients(loss, input_img)[0]  # [0] because of the batch dimen

    # normalization trick: we normalize the gradient
    grads = grads / (K.sqrt(K.sum(K.square(grads))) + 1e-10)
```

```
    # this function returns the loss and grads given the input picture
    iterate = K.function([input_img], [loss, grads])

    # we start from a gray image with some random noise
    input_img_data = np.random.random((1, img_width, img_height, 3))
    input_img_data = (input_img_data - 0.5) * (0.1 if is_conv else 0.001)

    # we run gradient ascent
    for i in range(iterations):
        loss_value, grads_value = iterate([input_img_data])
        input_img_data += grads_value * step
        if verbose:
            print('Current loss value:', loss_value)

    # decode the resulting input image
    img = image_values_to_rgb(input_img_data[0])

    return img, loss_value
```

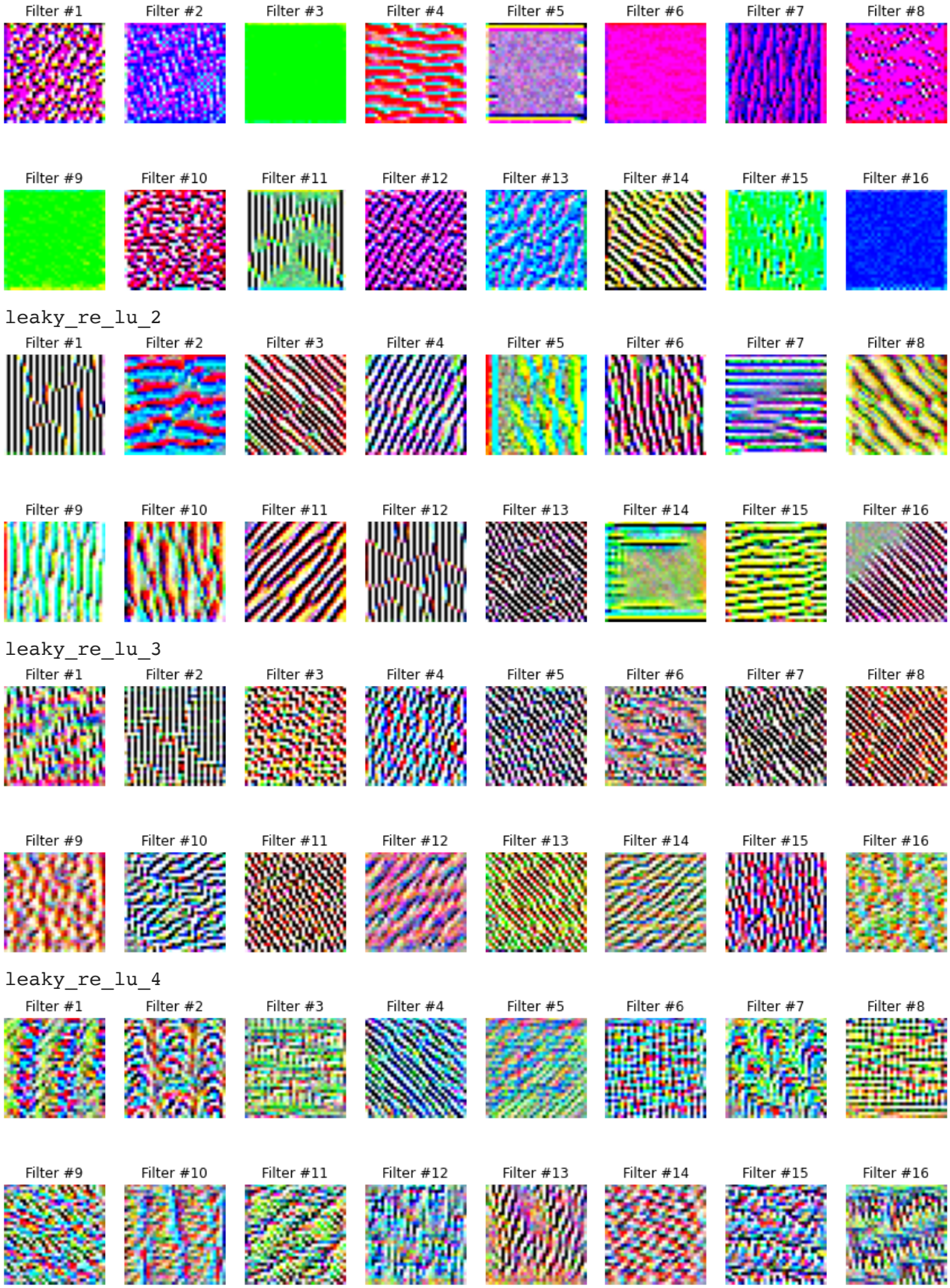
In [32]:

```
# sample maximum stimuli
def plot_filters_stimuli(layer_name, is_conv, model, iterations=20, step=1., 
    cols = 8
    rows = 2
    filter_index = 0
    max_filter_index = list(filter(lambda x: x.name == layer_name, model.laye
    fig = plt.figure(figsize=(2 * cols - 1, 3 * rows - 1))
    for i in range(cols):
        for j in range(rows):
            if filter_index <= max_filter_index:
                ax = fig.add_subplot(rows, cols, i * rows + j + 1)
                ax.grid('off')
                ax.axis('off')
                loss = -1e20
                while loss < 0 and filter_index <= max_filter_index:
                    stimuli, loss = find_maximum_stimuli(layer_name, is_conv,
                                                         iterations, step, ve
                    filter_index += 1
                if loss > 0:
                    ax.imshow(stimuli)
                    ax.set_title("Filter #{}".format(filter_index))
    plt.show()
```

In [35]:

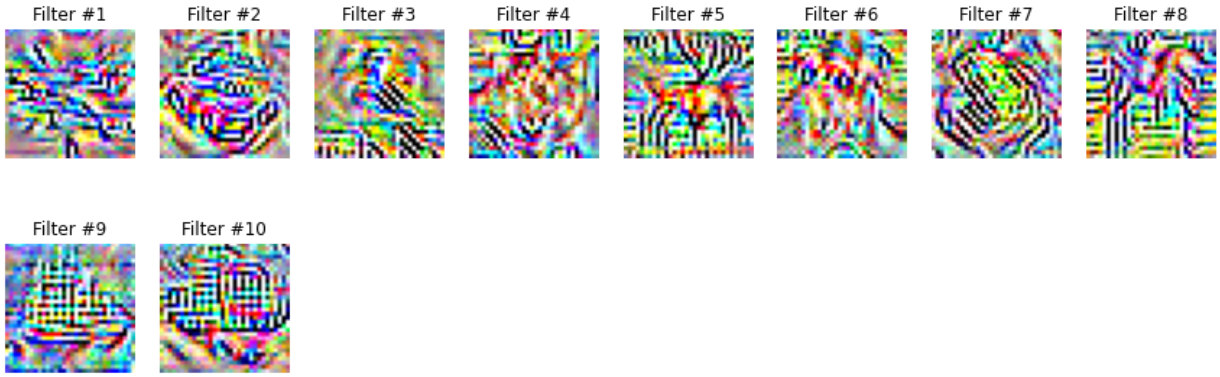
```
# maximum stimuli for convolutional neurons
conv_activation_layers = []
for layer in model.layers:
    if isinstance(layer, LeakyReLU):
        prev_layer = layer.inbound_nodes[0].inbound_layers[0]
        if isinstance(prev_layer, Conv2D):
            conv_activation_layers.append(layer)

for layer in conv_activation_layers:
    print(layer.name)
    plot_filters_stimuli(layer_name=layer.name, is_conv=True, model=model)
```

```
leaky_re_lu_1
```



In [37]:

```
# maximum stimuli for last dense layer
last_dense_layer = list(filter(lambda x: isinstance(x, Dense), model.layers))
plot_filters_stimuli(layer_name=last_dense_layer.name, is_conv=False,
                     iterations=200, step=0.1, model=model)
```



In [38]:
```
def maximum_stimuli_test_for_grader():
    layer = list(filter(lambda x: isinstance(x, Dense), model.layers))[-1]
    output_index = 7
    stimuli, loss = find_maximum_stimuli(
        layer_name=layer.name,
        is_conv=False,
        filter_index=output_index,
        model=model,
        verbose=False
    )
    return model.predict_proba(stimuli[np.newaxis, :])[0, output_index]
```

In [39]:
```
## GRADED PART, DO NOT CHANGE!
# Maximum stimuli test
grader.set_answer("96eco", maximum_stimuli_test_for_grader())
```

```
1/1 [==============================] - 0s
```

In [40]:
```
# you can make submission with answers so far to check yourself at this stage
grader.submit(COURSERA_EMAIL, COURSERA_TOKEN)
```

```
Submitted to Coursera platform. See results on assignment page!
```

That's it! Congratulations!

What you've done:

- defined CNN architecture
- trained your model
- evaluated your model
- visualised learnt filters