Theano and Machine Learning

Martin Meier

June 17, 2016

What is Keras?

- Frontend for Theano and Tensorflow
- Specialized for (deep) learning
- Has ready to use
  - Models
  - Layers
  - Optimizers
  - Datasets

Sequential models: simple MLP

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()

model.add(Dense(2, input_dim=2))
model.add(Activation('tanh'))

model.add(Dense(1, activation='tanh'))
```

Building the model

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()

model.add(Dense(2, input_dim=2))
model.add(Activation('tanh'))

model.add(Dense(1, activation='tanh'))

model.compile(loss='mse',
  optimizer='rmsprop', metrics=['accuracy'])
```

Training the model

```
import numpy as np
from keras.layers import Dense, Activation
from keras.models import Sequential

model = Sequential()
model.add(Dense(2, input_dim=2))
model.add(Activation('tanh'))
model.add(Dense(1, activation='tanh'))

model.compile(loss='mse',
  optimizer='rmsprop', metrics=['accuracy'])

X = np.array([[0,0],[0,1],[1,0],[1,1]])
Y = np.array([[0],[1],[1],[0]])

model.fit(X, Y, nb_epoch=1000, verbose=1, batch_size=1)
```

Available layers (some of them)

```
# fully connected layer
Dense(output_dim, activation='linear',
    bias=True, input_dim=None)

# applies dropout to the input p=[0,1]
Dropout(p)

# also as 1D and 3D
Convolution2D(nb_filter, nb_row, nb_col,
    border_mode='valid', dim_ordering='th')

MaxPooling2D(pool_size=(2, 2),
    border_mode='valid', dim_ordering='th')

# reshapes the input, use between conv and dense
Flatten()
```

Some activation functions

- relu
- tanh
- sigmoid
- softmax

Some loss functions (objectives in keras)

```
model.compile(loss='mse',
  optimizer='rmsprop', metrics=['accuracy'])
```

- mean_squared_error / mse
- mean_absolute_error / mae
- binary_crossentropy (logloss)
- categorical_crossentropy

Some optimizers functions

```
model.compile(loss='mse',
  optimizer='rmsprop', metrics=['accuracy'])

# 'sgd', nesterov is a special momentum
SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)

# 'rmsprop'
RMSprop(lr=0.001, rho=0.9, epsilon=1e-08)

# 'adagrad'
Adagrad(lr=0.01, epsilon=1e-08)

# 'adadelta'
Adadelta(lr=1.0, rho=0.95, epsilon=1e-08)
```

Included datasets: MNIST

- Handwritten digits
- 28×28 pixels
- 60000 training and 10000 test samples

Loading a dataset

```
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
img_rows, img_cols = 28, 28
X_train = X_train.reshape(X_train.shape[0], 1,
  img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1,
  img_rows, img_cols)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

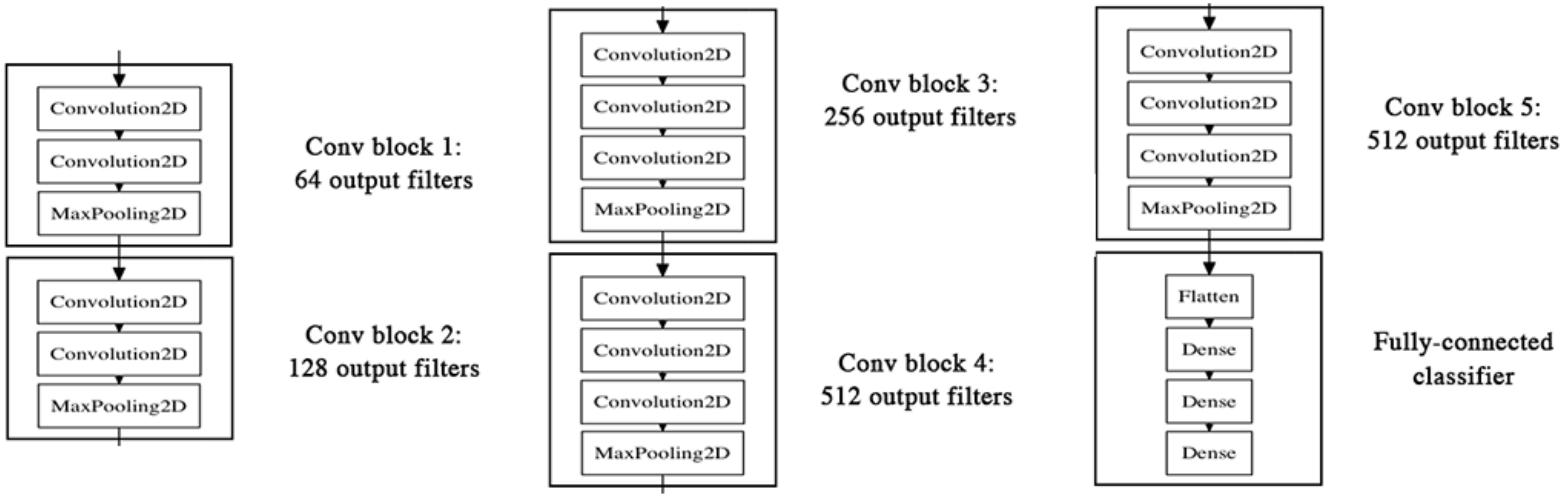
Using keras

```
mkdir src/nn
cd src/nn
source /vol/ni/share/theano/bin/active
```

Exercise

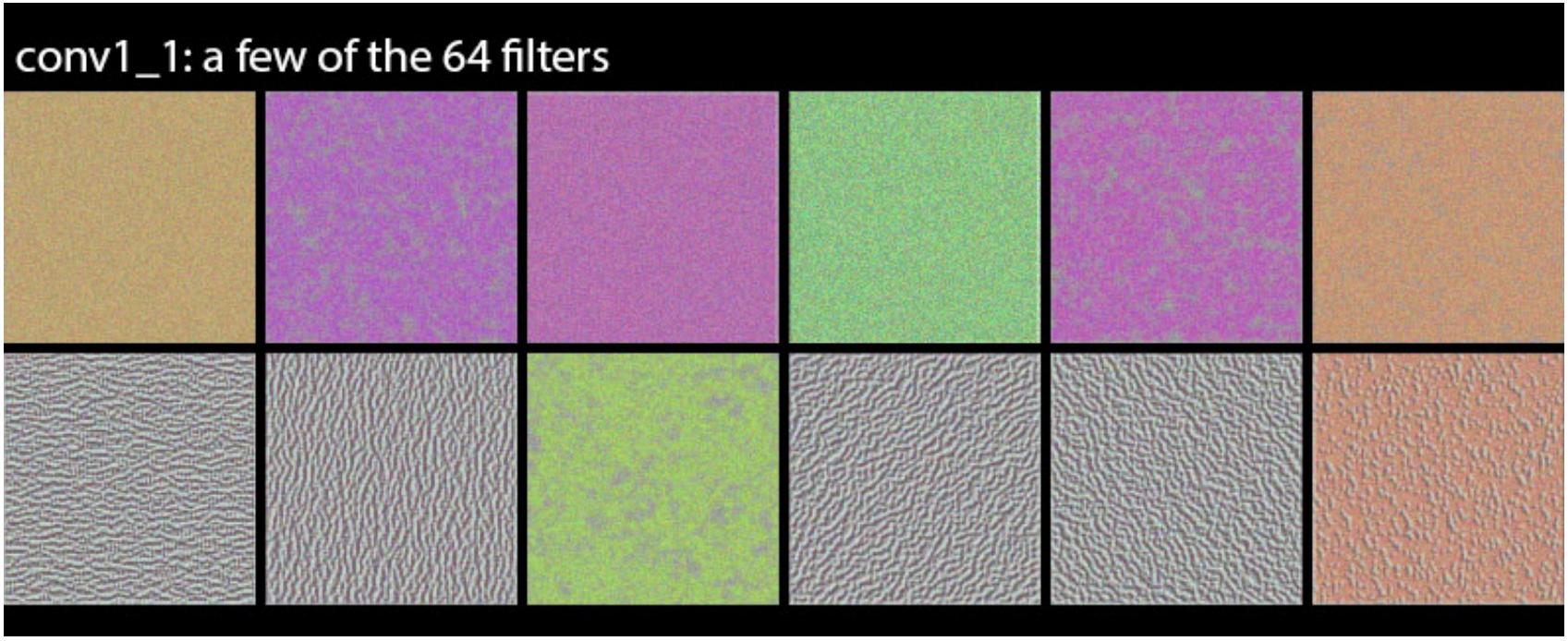
- Try to load the mnist dataset and build a small network with some Convolution2D and Dense layers.
- To save time during training, use only a subset of the available data, e.g. $X\_small = X\_train[0:5000]$
- Keep it reproducible: $numpy.random.seed(1)$
- Try:
  - Different convolution size
  - Activation functions
  - Optimizers

Using larger models: VGG16

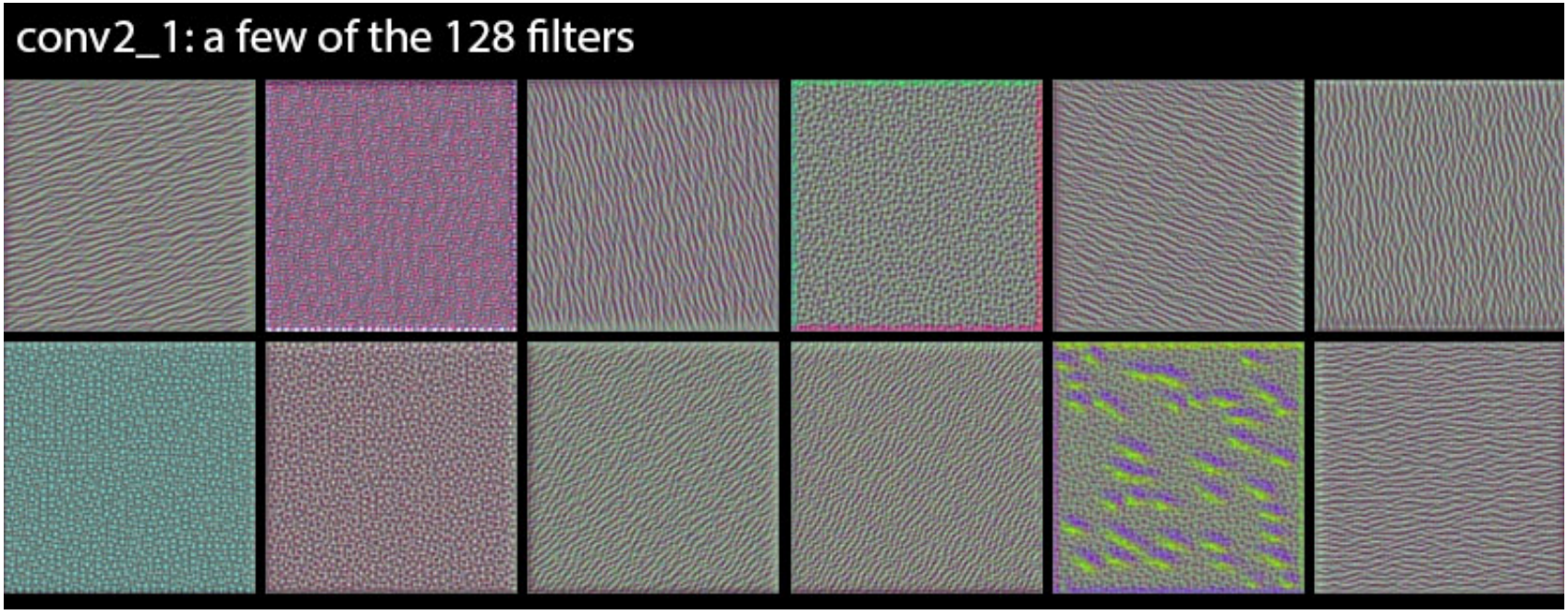


- ImageNet ILSVRC-2014 winner
- Pretrained network is available
- We will have a look at the learned filters

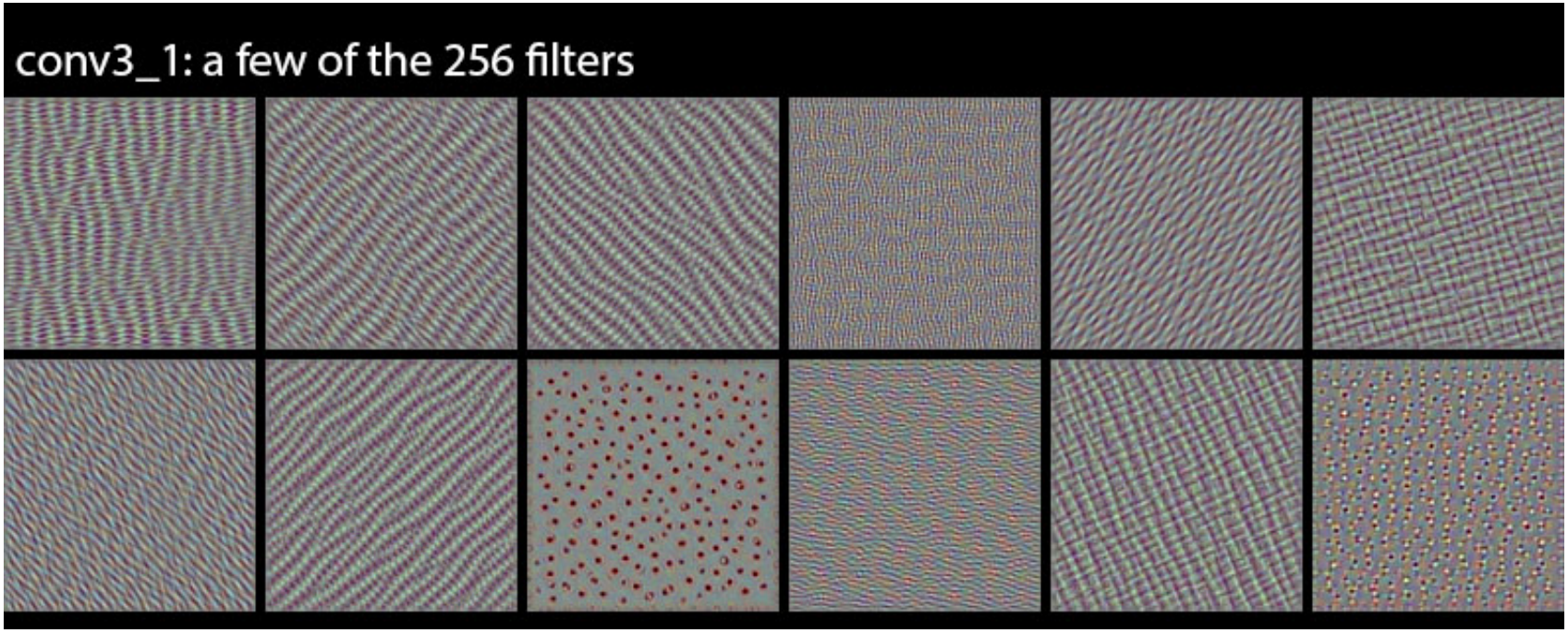
Some learned filters



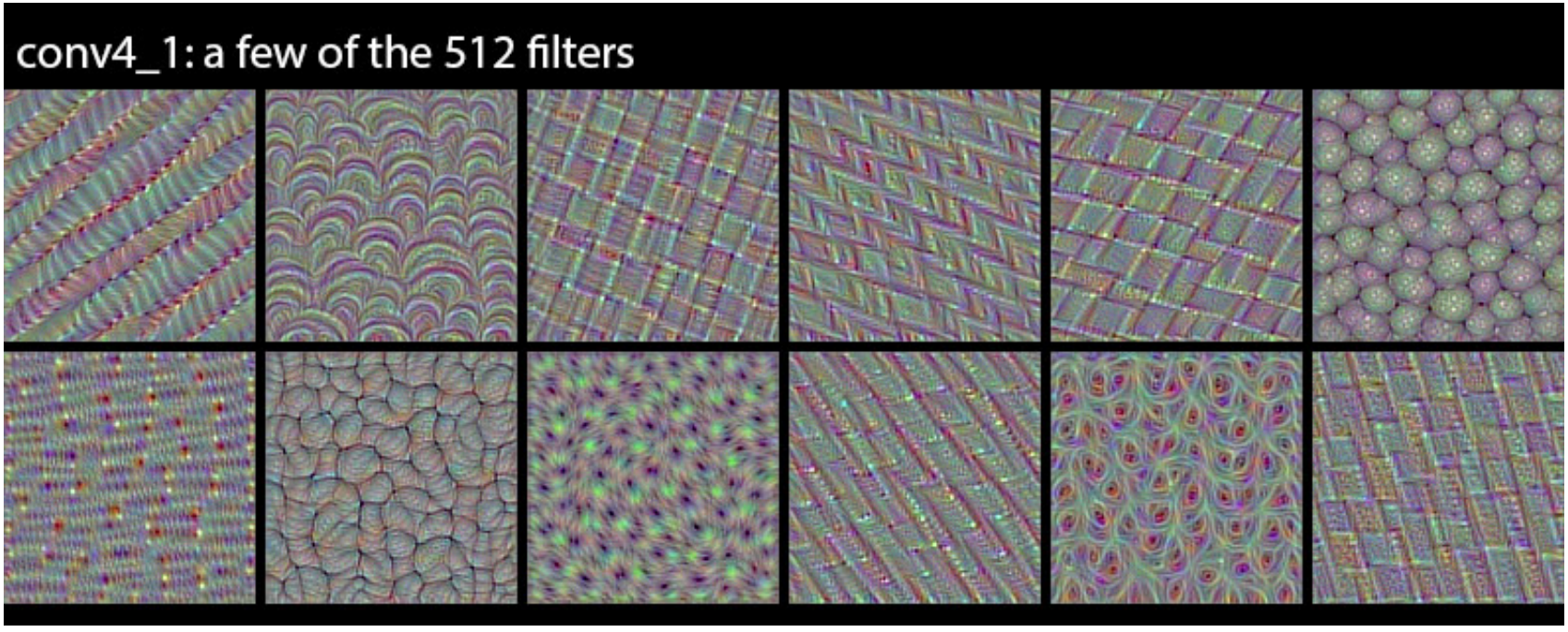
Some learned filters



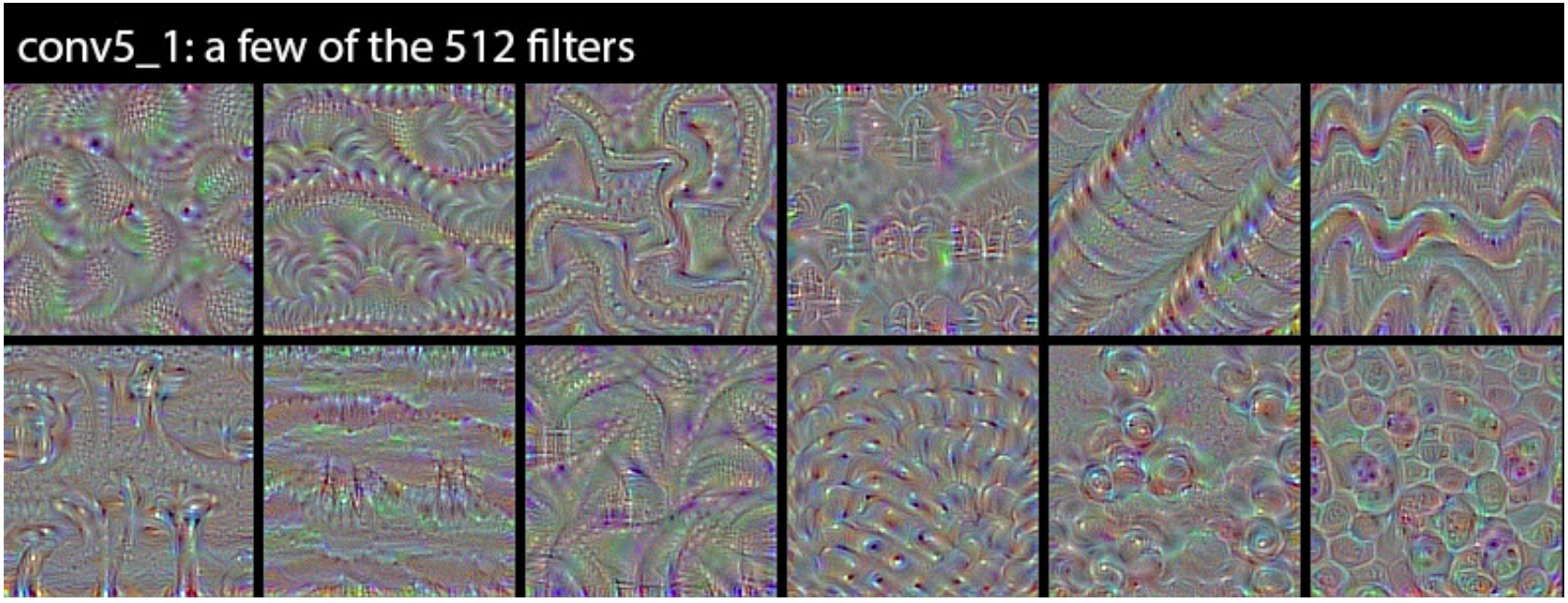
Some learned filters



Some learned filters



Some learned filters



Exercise: How are these filters activated on images?

- Let us see how these filters are activated on images
- Code: github.com/fchollet/keras/blob/master/examples/deep_dream.py
- Weights: /vol/ni/share/data/networks/vgg16_weights.h5