Deep Neural Nets and Keras

Pavel Krömer[1]

Data Science Summer School @ Uni Vienna

[1]Dept. of Computer Science,
VŠB - Technical University of Ostrava,
Ostrava, Czech Republic
pavel.kromer@vsb.cz

Outline

About

Introduction

(Deep) artificial neural networks are among the most successful machine–learning models.

They are universal tools that can be used for supervised and/or unsupervised learning.

Artificial neural networks

Artificial neural network

- a computational model evaluating a parametric function composed of many other parametric (sub)functions
- composed of many information processing units, organized into interconnected layers
- one unit solves a linearly separable problem, i.e. draws a hyperplane in an $n-$dimensional space

Keras

Keras

Keras is a high-level neural networks API written in Python.

Keras

Keras is a high-level neural networks API written in Python.

- easy prototyping
- support for convolutional and recurrent nets
- accellerated by multicore and GPU

Keras

Keras is a high-level neural networks API written in Python.

- easy prototyping
- support for convolutional and recurrent nets
- accellerated by multicore and GPU

Powered by a backend

- Tensorflow (default)
- Theano
- others (CNTK)

Keras (cont.)

My favourite because

- sufficiently high–level (for my taste)
- allows mixing–in with the wonderfull Python ecosystem (scikit, matplotlib, ...)
- is programmer oriented
- well–documented, with lots of examples

Keras (cont.)

My favourite because

- sufficiently high–level (for my taste)
- allows mixing–in with the wonderfull Python ecosystem (scikit, matplotlib, ...)
- is programmer oriented
- well–documented, with lots of examples
- one can cheat in it **`https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Keras_Cheat_Sheet_Python.pdf`**

Keras components

Model

- THE (deep) neural network you want to use
- a stack of connected layers
- sequential API $\times$ the bare MODEL class

Keras components

Model

- THE (deep) neural network you want to use
- a stack of connected layers
- sequential API $\times$ the bare MODEL class

Layers

- individual levels that define the architecture and functionality of the Model
- different types, properties, params, functions
  - DENSE layers (this is the normal, fully-connected layer)
  - CONVOLUTIONAL layers (applies convolution operations on the previous layer)
  - POOLING layers (used after convolutional layers)
  - DROPOUT layers (regularization, prevent overfitting)

Keras components (cont.)

Loss functions

- compare the predicted output with the real output in each pass of the training algorithm
- tell the model how the weights should be updated
- mean–squared error, cross–entropy, ...

Keras components (cont.)

Loss functions

- compare the predicted output with the real output in each pass of the training algorithm
- tell the model how the weights should be updated
- mean–squared error, cross–entropy, ...

Optimizers

- weight update strategies in the training process
- stochastic gradient descent, RMSProp, Adagrad

Keras hands–on

Installation

(Fairly) easy steps

- Get Python (Anaconda highly recommended: **`https://www.anaconda.com/download/`**)
- Get TensorFlow (**`https://www.tensorflow.org/install/`**)
- Get Keras (**`https://keras.io/`**)

Installation

(Fairly) easy steps

- Get Python (Anaconda highly recommended: **`https://www.anaconda.com/download/`**)
- Get TensorFlow (**`https://www.tensorflow.org/install/`**)
- Get Keras (**`https://keras.io/`**)

```
pip install tensorflow
pip install keras
```

Installation

(Fairly) easy steps

- Get Python (Anaconda highly recommended: **`https://www.anaconda.com/download/`**)
- Get TensorFlow (**`https://www.tensorflow.org/install/`**)
- Get Keras (**`https://keras.io/`**)

```
pip install tensorflow
pip install keras
pip install msgpack argparse pydot
```

Installation

(Fairly) easy steps

- Get Python (Anaconda highly recommended: **`https://www.anaconda.com/download/`**)
- Get TensorFlow (**`https://www.tensorflow.org/install/`**)
- Get Keras (**`https://keras.io/`**)

```
pip install tensorflow
pip install keras
pip install msgpack argparse pydot
```

```
conda install keras
conda install pydot
```

The mother of all classification demos: cats vs. dogs

The mother of all classification demos: cats vs. dogs

Published on Kaggle in 2014, contains 25,000 images of cats and dogs.

The mother of all classification demos: cats vs. dogs

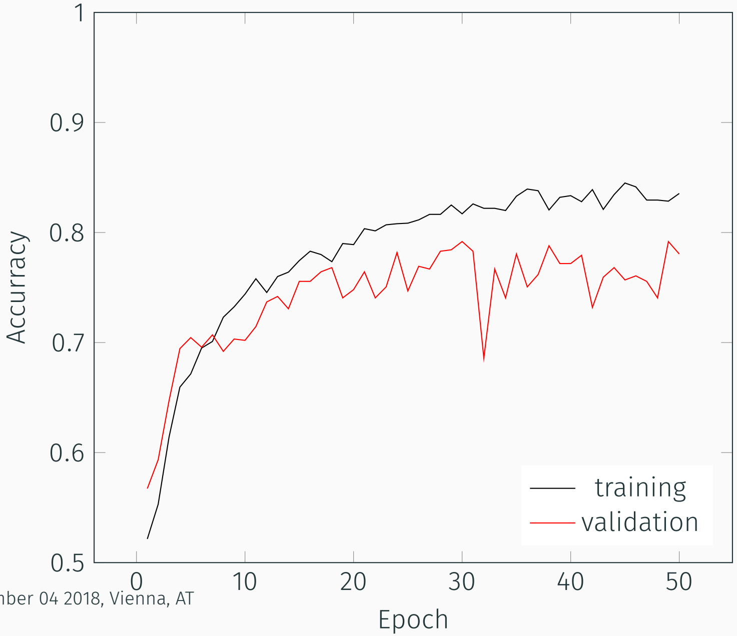
Published on Kaggle in 2014, contains 25,000 images of cats and dogs.

To make it a bit harder, we use only 1000 training images of each class.

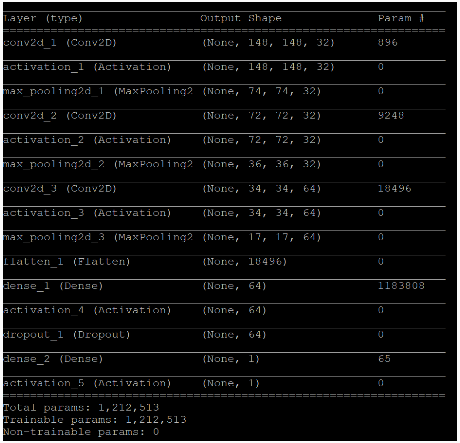
Computer demo …

`https://goo.gl/M5ShF3`

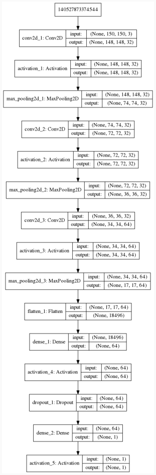
From scratch™



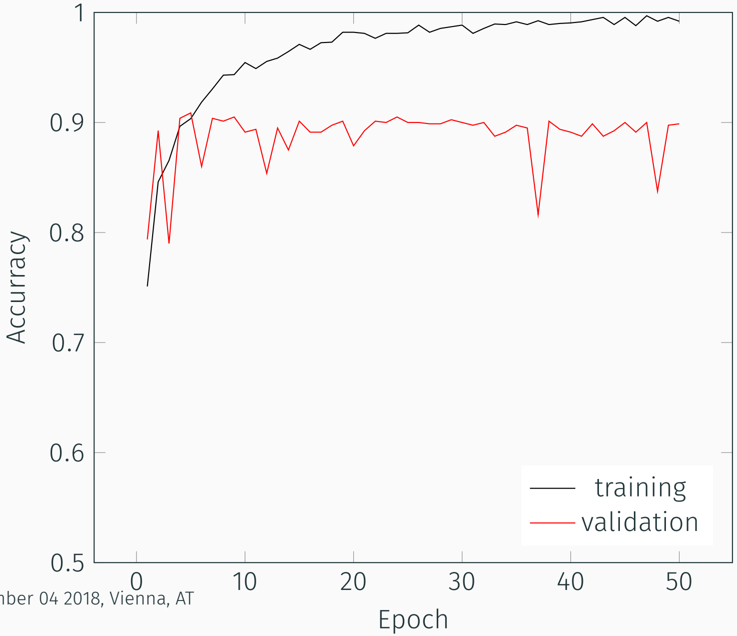
From scratch^{TM}



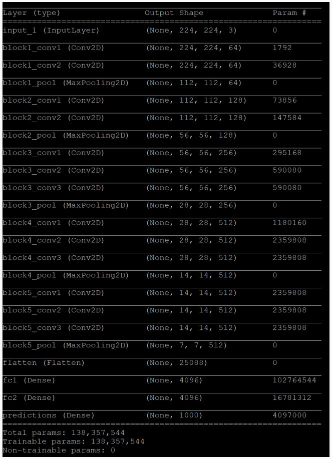
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 1,212,513
Trainable params: 1,212,513
Non-trainable params: 0
```


140527873374544
conv2d_1: Conv2D | input: (None, 150, 150, 3) | output: (None, 148, 148, 32)
activation_1: Activation | input: (None, 148, 148, 32) | output: (None, 148, 148, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 148, 148, 32) | output: (None, 74, 74, 32)
conv2d_2: Conv2D | input: (None, 74, 74, 32) | output: (None, 72, 72, 32)
activation_2: Activation | input: (None, 72, 72, 32) | output: (None, 72, 72, 32)
max_pooling2d_2: MaxPooling2D | input: (None, 72, 72, 32) | output: (None, 36, 36, 32)
conv2d_3: Conv2D | input: (None, 36, 36, 32) | output: (None, 34, 34, 64)
activation_3: Activation | input: (None, 34, 34, 64) | output: (None, 34, 34, 64)
max_pooling2d_3: MaxPooling2D | input: (None, 34, 34, 64) | output: (None, 17, 17, 64)
flatten_1: Flatten | input: (None, 17, 17, 64) | output: (None, 18496)
dense_1: Dense | input: (None, 18496) | output: (None, 64)
activation_4: Activation | input: (None, 64) | output: (None, 64)
dropout_1: Dropout | input: (None, 64) | output: (None, 64)
dense_2: Dense | input: (None, 64) | output: (None, 1)
activation_5: Activation | input: (None, 1) | output: (None, 1)

VGG16 / ImageNet

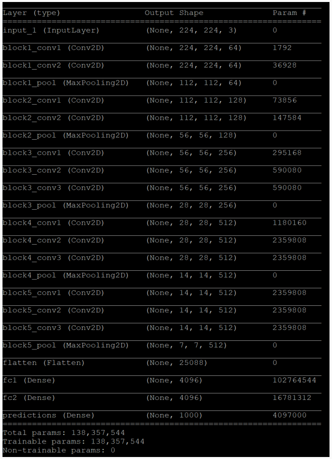


What VGG16 dreams about?



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

What VGG16 dreams about?



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```