Deep Neural Nets and Keras

Pavel Krömer¹

Data Science Summer School @ Uni Vienna

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

About
Keras
Components

Keras hands-on
Installation
Fun with puppies, kitties, and DNNs
About
(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 is a high-level neural networks API written in Python.
Keras is a **high-level neural networks API** written in **Python**.

- easy prototyping
- support for convolutional and recurrent nets
- accelerated by multicore and GPU
Keras is a high-level neural networks API written in Python.

- easy prototyping
- support for convolutional and recurrent nets
- accelerate 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
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 × the bare Model class
Keras components

Model

• THE (deep) neural network you want to use
• a stack of connected layers
• sequential API × 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)
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, ...
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/)
(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
```
(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

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.
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
Epoch | Accuracy
---|---
Training | Validation
0 | 0.55 | 0.54
10 | 0.70 | 0.60
20 | 0.75 | 0.65
30 | 0.80 | 0.70
40 | 0.85 | 0.75
50 | 0.90 | 0.80
Layer (type)                  Output Shape      Param #
-----------------------------------------------------------------------------------
conv2d_1 (Conv2D)             (None, 148, 148, 32) 896  
activation_1 (Activation)    (None, 148, 148, 32) 0  
max_pooling2d_1 (MaxPooling2D) (None, 74, 74, 32) 0  
conv2d_2 (Conv2D)             (None, 72, 72, 32) 9248  
activation_2 (Activation)    (None, 72, 72, 32) 0  
max_pooling2d_2 (MaxPooling2D) (None, 36, 36, 32) 0  
conv2d_3 (Conv2D)             (None, 34, 34, 64) 18496  
activation_3 (Activation)    (None, 34, 34, 64) 0  
max_pooling2d_3 (MaxPooling2D) (None, 17, 17, 64) 0  
flattlen_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
<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_1 (InputLayer)</td>
<td>(None, 224, 224, 3)</td>
<td>0</td>
</tr>
<tr>
<td>block1_conv1 (Conv2D)</td>
<td>(None, 224, 224, 64)</td>
<td>1792</td>
</tr>
<tr>
<td>block1_conv2 (Conv2D)</td>
<td>(None, 224, 224, 64)</td>
<td>36928</td>
</tr>
<tr>
<td>block1_pool1 (MaxPooling2D)</td>
<td>(None, 112, 112, 64)</td>
<td>0</td>
</tr>
<tr>
<td>block2_conv1 (Conv2D)</td>
<td>(None, 112, 112, 128)</td>
<td>73856</td>
</tr>
<tr>
<td>block2_conv2 (Conv2D)</td>
<td>(None, 112, 112, 128)</td>
<td>147704</td>
</tr>
<tr>
<td>block2_pool1 (MaxPooling2D)</td>
<td>(None, 56, 56, 128)</td>
<td>0</td>
</tr>
<tr>
<td>block3_conv1 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>295160</td>
</tr>
<tr>
<td>block3_conv2 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>590080</td>
</tr>
<tr>
<td>block3_conv3 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>590080</td>
</tr>
<tr>
<td>block3_pool1 (MaxPooling2D)</td>
<td>(None, 28, 28, 256)</td>
<td>0</td>
</tr>
<tr>
<td>block4_conv1 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>1180160</td>
</tr>
<tr>
<td>block4_conv2 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>2359008</td>
</tr>
<tr>
<td>block4_conv3 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>2359008</td>
</tr>
<tr>
<td>block4_pool1 (MaxPooling2D)</td>
<td>(None, 14, 14, 512)</td>
<td>0</td>
</tr>
<tr>
<td>block5_conv1 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359008</td>
</tr>
<tr>
<td>block5_conv2 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359008</td>
</tr>
<tr>
<td>block5_conv3 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359008</td>
</tr>
<tr>
<td>block5_pool1 (MaxPooling2D)</td>
<td>(None, 7, 7, 512)</td>
<td>0</td>
</tr>
<tr>
<td>flatten (Flatten)</td>
<td>(None, 25088)</td>
<td>0</td>
</tr>
<tr>
<td>fc1 (Dense)</td>
<td>(None, 4096)</td>
<td>102764544</td>
</tr>
<tr>
<td>fc2 (Dense)</td>
<td>(None, 4096)</td>
<td>16781112</td>
</tr>
<tr>
<td>predictions (Dense)</td>
<td>(None, 1000)</td>
<td>4097000</td>
</tr>
</tbody>
</table>

Total params: 138,357,564
Trainable params: 138,357,564
Non-trainable params: 0
What VGG16 dreams about?

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_1 (InputLayer)</td>
<td>(None, 224, 224, 3)</td>
<td>0</td>
</tr>
<tr>
<td>block1_conv1 (Conv2D)</td>
<td>(None, 224, 224, 64)</td>
<td>1792</td>
</tr>
<tr>
<td>block1_conv2 (Conv2D)</td>
<td>(None, 224, 224, 64)</td>
<td>36320</td>
</tr>
<tr>
<td>block1_pool1 (MaxPooling2D)</td>
<td>(None, 112, 112, 64)</td>
<td>0</td>
</tr>
<tr>
<td>block2_conv1 (Conv2D)</td>
<td>(None, 112, 112, 128)</td>
<td>73856</td>
</tr>
<tr>
<td>block2_conv2 (Conv2D)</td>
<td>(None, 112, 112, 128)</td>
<td>147704</td>
</tr>
<tr>
<td>block2_pool1 (MaxPooling2D)</td>
<td>(None, 56, 56, 128)</td>
<td>0</td>
</tr>
<tr>
<td>block3_conv1 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>295160</td>
</tr>
<tr>
<td>block3_conv2 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>590080</td>
</tr>
<tr>
<td>block3_conv3 (Conv2D)</td>
<td>(None, 56, 56, 256)</td>
<td>590080</td>
</tr>
<tr>
<td>block3_pool1 (MaxPooling2D)</td>
<td>(None, 28, 28, 256)</td>
<td>0</td>
</tr>
<tr>
<td>block4_conv1 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>1180160</td>
</tr>
<tr>
<td>block4_conv2 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>2359808</td>
</tr>
<tr>
<td>block4_conv3 (Conv2D)</td>
<td>(None, 28, 28, 512)</td>
<td>2359808</td>
</tr>
<tr>
<td>block4_pool1 (MaxPooling2D)</td>
<td>(None, 14, 14, 512)</td>
<td>0</td>
</tr>
<tr>
<td>block5_conv1 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359808</td>
</tr>
<tr>
<td>block5_conv2 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359808</td>
</tr>
<tr>
<td>block5_conv3 (Conv2D)</td>
<td>(None, 14, 14, 512)</td>
<td>2359808</td>
</tr>
<tr>
<td>block5_pool1 (MaxPooling2D)</td>
<td>(None, 7, 7, 512)</td>
<td>0</td>
</tr>
<tr>
<td>flatten (Flatten)</td>
<td>(None, 25088)</td>
<td>0</td>
</tr>
<tr>
<td>fc1 (Dense)</td>
<td>(None, 4096)</td>
<td>102764544</td>
</tr>
<tr>
<td>fc2 (Dense)</td>
<td>(None, 4096)</td>
<td>16781112</td>
</tr>
<tr>
<td>predictions (Dense)</td>
<td>(None, 1000)</td>
<td>40570000</td>
</tr>
</tbody>
</table>

Total params: 138,157,564
Trainable params: 138,157,564
Non-trainable params: 0