What is Deep Learning

The Big Picture – From History to Application using Keras

Christian Heumann, Matthias Aßenmacher, Daniel Schalk

October 24, 2018

History of Deep Learning

Deep Learning Timeline - 1

1943

Warren S. McCulloch and **Walter Pitts** published tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1943

Deep Learning Timeline - 1

1943
Warren S. McCulloch and **Walter Pitts** published tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1943 | 1950

1950
Alan Turing predicted the impact of Machine Learning. Additionally, he crafted what has been dubbed The Turing Test.

1957
Frank Rosenblatt declared that he construct a system which works similar to the perceptual processes of a biological brain - The Perceptron.

Deep Learning Timeline - 1

1943

Warren S. McCulloch and **Walter Pitts** published tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1960

Henry J. Kelley published “Gradient Theory of Optimal Flight Paths”. Many of his ideas were used to develop the basics of a continuous backpropagation model (aka the backward propagation of errors) used in training neural networks.

1950

Alan Turing predicted the impact of Machine Learning. Additionally, he crafted what has been dubbed The Turing Test.

1957

Frank Rosenblatt declared that he construct a system which works similar to the perceptual processes of a biological brain - The Perceptron.

Deep Learning Timeline - 1

1943 | 1950 | 1960 | 1965

1943
Warren S. McCulloch and **Walter Pitts** published tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1950
Alan Turing predicted the impact of Machine Learning. Additionally, he crafted what has been dubbed The Turing Test.

1957
Frank Rosenblatt declared that he construct a system which works similar to the perceptual processes of a biological brain - The Perceptron.

1960
Henry J. Kelley published "Gradient Theory of Optimal Flight Paths". Many of his ideas were used to develop the basics of a continuous backpropagation model (aka the backward propagation of errors) used in training neural networks.

1965
Alexey Ivakhnenko and **V. G. Lapa** created the first *working* deep learning networks, applying what had been only theories and ideas up to that point.

Ivakhnenko developed a learning algorithm using deep feedforward multilayer perceptrons. For that reason alone, many consider Ivakhnenko the father of modern deep learning.

Deep Learning Timeline - 1

1943 | 1950 | 1960 | 1965 | 1970

1943
Warren S. McCulloch and **Walter Pitts** published tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1950
Alan Turing predicted the impact of Machine Learning. Additionally, he crafted what has been dubbed The Turing Test.

1957
Frank Rosenblatt declared that he construct a system which works similar to the perceptual processes of a biological brain - The Perceptron.

1960
Henry J. Kelley published "Gradient Theory of Optimal Flight Paths". Many of his ideas were used to develop the basics of a continuous backpropagation model (aka the backward propagation of errors) used in training neural networks.

1965
Alexey Ivakhnenko and **V. G. Lapa** created the first *working* deep learning networks, applying what had been only theories and ideas up to that point.

Ivakhnenko developed a learning algorithm using deep feedforward multilayer perceptrons. For that reason alone, many consider Ivakhnenko the father of modern deep learning.

1970 - First AI Winter
AI was subject to critiques and financial setbacks. AI researchers had failed to appreciate the difficulty of the problems they faced.

AI was claimed to only be suitable for solving "toy" versions.

Deep Learning Timeline - 2

1980

Kunihiko Fukushima creates Neocognitron, an artificial neural network that learned how to recognize visual patterns.

His work led to the development of the first convolutional neural networks, which are based on the visual cortex organization found in animals.

1980

Deep Learning Timeline - 2

1980

Kunihiko Fukushima creates Neocognitron, an artificial neural network that learned how to recognize visual patterns.

His work led to the development of the first convolutional neural networks, which are based on the visual cortex organization found in animals.

1980 1987

1987 - Second AI Winter

Many AI projects expectations had run much higher than what was actually possible.

Over 300 AI companies had shutdown, gone bankrupt, or been acquired by the end of 1993, effectively ending the first commercial wave of AI.

Deep Learning Timeline - 2

1980
Kunihiko Fukushima creates Neocognitron, an artificial neural network that learned how to recognize visual patterns.

His work led to the development of the first convolutional neural networks, which are based on the visual cortex organization found in animals.

1993
Increasing computing power and focusing on specific isolates problems let AI be more successful than ever.

1997
IBMs chess-playing computer Deep Blue beats the world chess champion Garry Kasparov.

1980 | 1987 | 1993

1987 - Second AI Winter
Many AI projects expectations had run much higher than what was actually possible.

Over 300 AI companies had shutdown, gone bankrupt, or been acquired by the end of 1993, effectively ending the first commercial wave of AI.

Deep Learning Timeline - 2

1980

Kunihiko Fukushima creates Neocognitron, an artificial neural network that learned how to recognize visual patterns.

His work led to the development of the first convolutional neural networks, which are based on the visual cortex organization found in animals.

1987 - Second AI Winter

Many AI projects expectations had run much higher than what was actually possible.

Over 300 AI companies had shutdown, gone bankrupt, or been acquired by the end of 1993, effectively ending the first commercial wave of AI.

1993

Increasing computing power and focusing on specific isolates problems let AI be more successful than ever.

1997

IBMs chess-playing computer Deep Blue beats the world chess champion Garry Kasparov.

1997

Algorithms originally developed by AI researchers began to appear as parts of larger systems. AI had solved a lot of very difficult problems and their solutions proved to be useful throughout the technology industry, such as data mining, industrial robotics, logistics, speech recognition, banking software, medical diagnosis and Google's search engine.

1980 | 1987 | 1993 | 1997

Deep Learning Timeline - 2

1980
Kunihiko Fukushima creates Neocognitron, an artificial neural network that learned how to recognize visual patterns.

His work led to the development of the first convolutional neural networks, which are based on the visual cortex organization found in animals.

1987 - Second AI Winter
Many AI projects expectations had run much higher than what was actually possible.

Over 300 AI companies had shutdown, gone bankrupt, or been acquired by the end of 1993, effectively ending the first commercial wave of AI.

1993
Increasing computing power and focusing on specific isolates problems let AI be more successful than ever.

1997
IBMs chess-playing computer Deep Blue beats the world chess champion Garry Kasparov.

1997
Algorithms originally developed by AI researchers began to appear as parts of larger systems. AI had solved a lot of very difficult problems and their solutions proved to be useful throughout the technology industry, such as data mining, industrial robotics, logistics, speech recognition, banking software, medical diagnosis and Google's search engine.

2011 - Deep Learning
Deep Learning appeared as Machine Learning method which uses deep graphs with many processing layers.

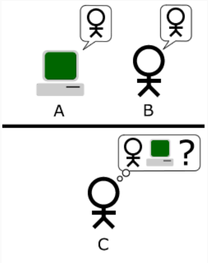
Today Deep Learning is everywhere and often wrongly equated with AI.

1980 1987 1993 1997 2011

Fascination Deep Learning

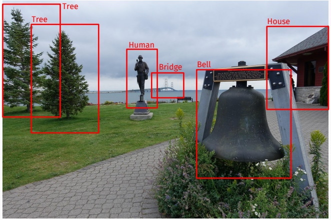
Imitating Humans - 1

- Turing test:
  - Developed by Alan Turing in 1950
  - Test of a machine's ability to exhibit intelligent behavior
  - Player C, the interrogator, is given the task of trying to determine which player, A or B, is a computer and which is a human

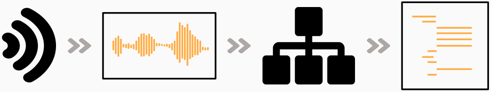


Imitating Humans - 2

- Image Recognition (Seeing):



- Speech Recognition and Text Mining (Hearing and understanding text):



Imitating Humans - 3

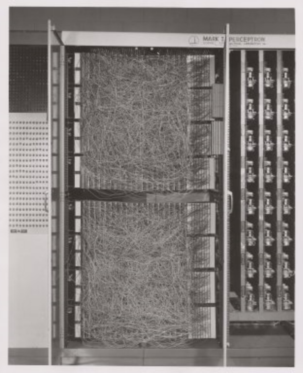
- And know we try to learn them being creative:
  - Music and Text Generation
  - Neural Style Transfer:

  - . . .

Why Deep Learning is so Powerful?

The Perceptron

- The perceptron was invented by Frank Rosenblatt 1957.

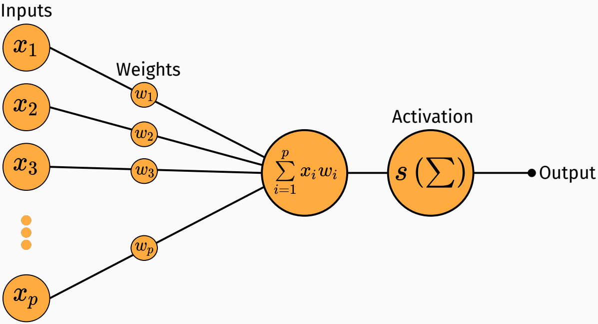


The Mark I Perceptron

- It is the basic computational unit for neural networks.

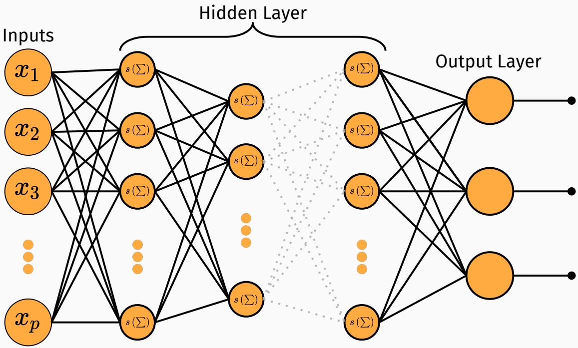
Singlelayer Perceptron

- Weighted sum of input values transformed by an activation function s
- If s is the sigmoid function $(1 + \exp \sum)^{-1}$, then the perceptron does exactly the same as the logistic regression



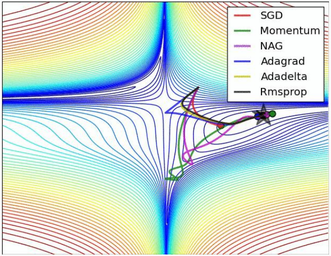
Multilayer Perceptron

- Stacking of multiple perceptrons
- Corresponds to stacking GLM models
- Number of parameter grows very fast
  $\rightarrow$ Optimizing becomes more difficult



Optimizer

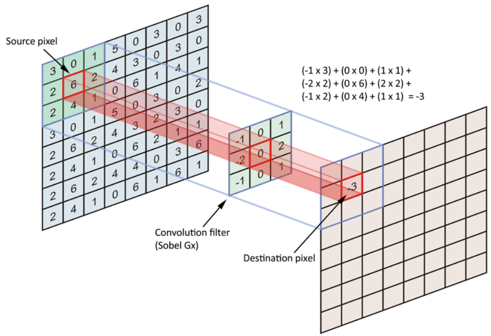
- Having that much parameter/weights to find, standard optimizer like Gradient Descent may fail
- Therefore, much effort was spend to get faster optimizer like momentum, adagrad, etc.:



Source: Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

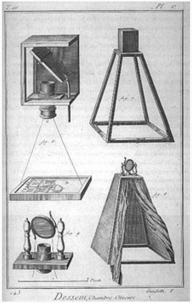
Convolution

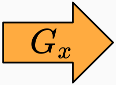
- Generating of new, hopefully meaningful, features of the input (commonly images)



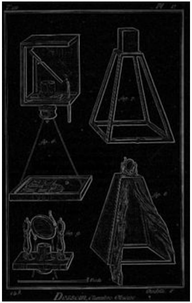
Convolution

Input Image





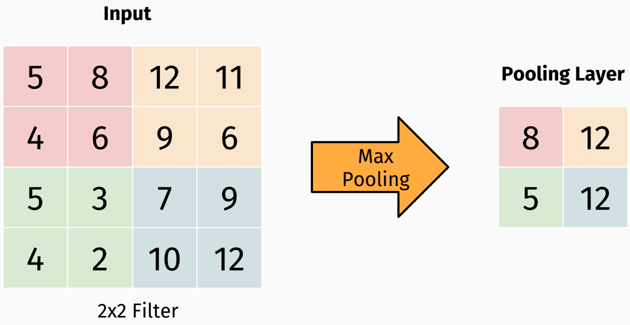
Generated Image/Feature



Note: The recognition of edges and corners requires a multiple application of the operator.

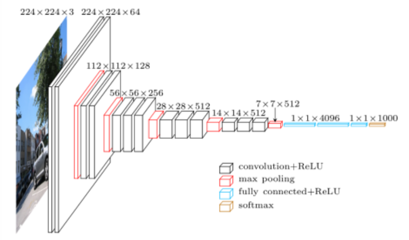
Pooling

- Down-sampling of images
- Reduces overfitting, memory usage, and therefore speeds up the fitting process



Lets Get Deep

- The secret of Deep Learning is the chaining of hidden layer such as convolution layers, pooling layers, and so on
- This deep structure allows the network to create powerful features and explore complex structures within the data
- VGG16 architecture:



Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Pre Trained Models

Model	Size	Parameters	Depth
Xception	88 MB	22,910,480	126
VGG16	528 MB	138,357,544	23
VGG19	549 MB	143,667,240	26
ResNet50	99 MB	25,636,712	168
InceptionV3	92 MB	23,851,784	159
InceptionResNetV2	215 MB	55,873,736	572
MobileNet	16 MB	4,253,864	88
MobileNetV2	14 MB	3,538,984	88
DenseNet121	33 MB	8,062,504	121
DenseNet169	57 MB	14,307,880	169
DenseNet201	80 MB	20,242,984	201
NASNetMobile	23 MB	5,326,716	-
NASNetLarge	343 MB	88,949,818	-

Source: Keras Documentation

Challenges in Deep Learning

- **Architecture Search**
  Having that much possibilities of combining hidden layer, optimizer, and activation functions we run into the problem of finding a good architecture.

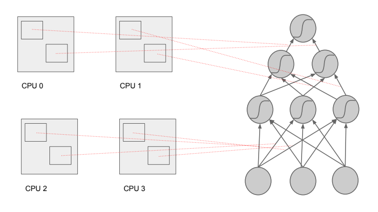
  $\rightarrow$ Transfer learning, use already trained models, adjust them to your data situation, and train (a subset of) the weights.

- **Expensive Training**
  Training of DNNs require billions of matrix multiplications, hence training **one** DNN might take weeks.

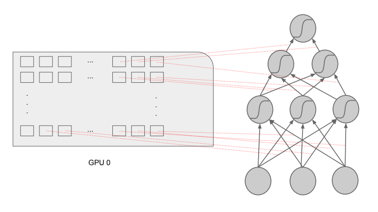
  $\rightarrow$ Use GPU server for serious applications. Why? See next slides.

About Implementations

- Deep Neural Networks require special hardware to be trained efficiently.
- The training is done using **G**raphics **P**rocessing **U**nits and a special programming language called **CUDA**.
- Training on standard CPUs takes a very long long time and gets infeasible for anything but toy examples.



Each CPU can do 2 – 8 parallel computations.

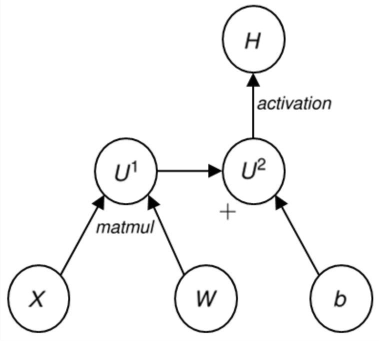


A single GPU can do thousands of simple parallel computations.

Software

CUDA is a very **low level** programming language and thus writing code for Deep Learning requires a lot of work. Software projects, like TensorFlow and abstract CUDA provide additional functionality.

The basic concept of calculations in deep neural networks is a *computational graph*, which describes the dependency structure of the network.



Computational graph for $f(XW + b)$.

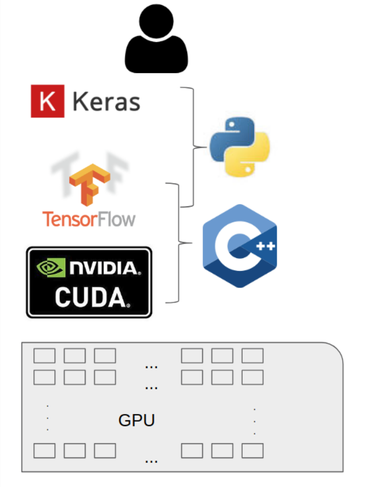
TensorFlow

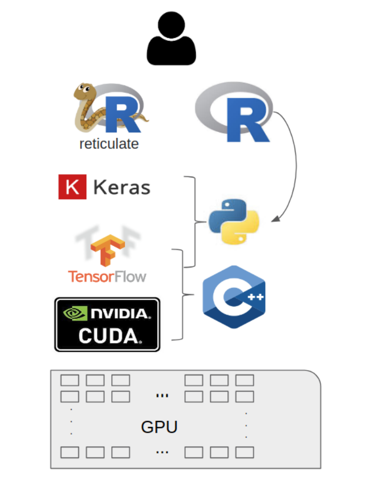
- Open-source framework developed by google.
- Rather low-level and aimed to directly work with computational graphs.
- Mainly support for Python (R support only via *reticulate*).
- Widely used and well documented.

Keras

- Open-source high-level API for Deep Learning.
- Can run on top of **TensorFlow**, CNTK or Theano.
- Mainly support for Python (R support only via **reticulate**).
- Widely used and well documented.

Keras in R





- Deep learning in R is the same as in Python.
- Communication from R to Python via **reticulate**.

Keras in R

- Syntax is (almost) identical.
- Same functionality.
- Same speed (slight overhead for communication between R and Python).
- More difficult to debug.

Useful resources:

https://keras.rstudio.com/

https://rstudio.github.io/reticulate/articles/introduction.html

PyTorch

- Open-source framework developed by facebook.
- Reimplementation of Torch.
- Only support for Python.
- Widely used and well documented.

Fast.ai

fast.ai

Making neural nets uncool again

- Open-source high-level API build on top of PyTorch.
- Still in alpha version.
- Only support for Python.
- Initially developed for the *Practical Deep Learning for Coders* online course.

mxnet

- Open-source framework in the Apache foundation.
- Scalable, allow easy training on multiple GPUs in parallel.
- Supports multiple languages (C++, Python, R, Julia, Matlab, JavaScript, Go, Scala, Pearl).
- Not as widely used as other frameworks.

Where to Start in the DL Jungle

Getting Started with Keras - Installation

Install keras using `pip` (or pip3 for python3) from the command line:

```
# python
pip install --upgrade tensorflow
pip install keras

# python3
pip3 install --upgrade tensorflow
pip3 install keras
```

On linux you may need to run the commands as `sudo`.

Getting Started with Keras - Overview

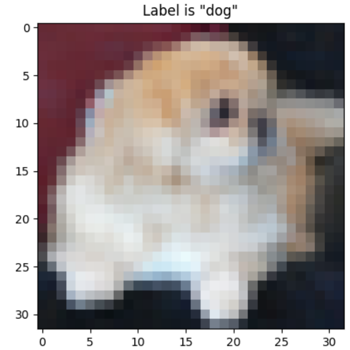
- Instead of introducing theory, we want to get into the topic by applying it.
- We using examples similar to the book **Deep Learning with Python** that are prepared as **notebooks**.
- **But**: When using something new, e.g. a convolution layer or optimizer, try to understand what it does and why it might be beneficial!
- First of all, we have to load keras and import models and layers:

```
import keras # equal to Rs library command
from keras import models
from keras import layers
```

Gettins Started with Keras - Example Data 1

```
import numpy as np
from keras.datasets import cifar10 # Our dataset
# See https://www.cs.toronto.edu/~kriz/cifar.html

(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
X_train.shape
## (50000, 32, 32, 3)

categories = ["airplane", "automobile", "bird", "cat",
              "deer", "dog", "frog", "horse", "ship",
              "truck"]

import matplotlib.pyplot as plt

category = categories[np.asscalar(Y_train[2305])]
plt.title('Label is \"{label}\"'.format(label=category))
plt.imshow(X_train[2305], cmap='gray')
plt.show()
```

Getting Started with Keras - Example Data 2



Getting Started with Keras - Example Data 3

The images are represented as rgb colors, each of the 32×32 pixels is represented as three numbers between 0 and 255. For the training we want to use values between 0 and 1:

```
train_images = X_train.astype('float32') / 255
test_images = X_test.astype('float32') / 255
```

Additionally, we have to convert the labels to a categorical data type:

```
from keras.utils import to_categorical

train_labels = to_categorical(Y_train)
test_labels = to_categorical(Y_test)
```

Getting Started with Keras - DNN 1

```
network = models.Sequential()

network.add(layers.Conv2D(64, (3, 3),
            input_shape=(32, 32, 3), padding='same',))
# now: model.output_shape == (None, 64, 32, 32)

network.add(layers.Flatten())

# Add fully connected hidden layer:
network.add(layers.Dense(units=128,
                         activation='relu'))

# Add output layer which maps each category to a neuron:
network.add(layers.Dense(10, activation='softmax'))

# Make the network ready for training:
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Getting Started with Keras - DNN 2

```
network.summary()
## ________________________________________________________________
## Layer (type)                 Output Shape              Param #
## ================================================================
## conv2d_2 (Conv2D)            (None, 32, 32, 64)        1792
## ________________________________________________________________
## flatten_2 (Flatten)          (None, 65536)             0
## ________________________________________________________________
## dense_3 (Dense)              (None, 128)               8388736
## ________________________________________________________________
## dense_4 (Dense)              (None, 10)                1290
## ================================================================
## Total params: 8,391,818
## Trainable params: 8,391,818
## Non-trainable params: 0
## ________________________________________________________________
##
```

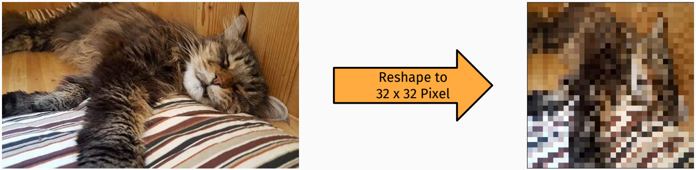
Getting Started with Keras - DNN 3

```
# Train network:
network.fit(train_images, train_labels, epochs=5, batch_size=128)

network.evaluate(test_images, test_labels)
## [1.5289124059677124, 0.4727]
```

Now we try to predict the class of a new image. Therefore, we have to rescale the original image to the same input shape of the training images:

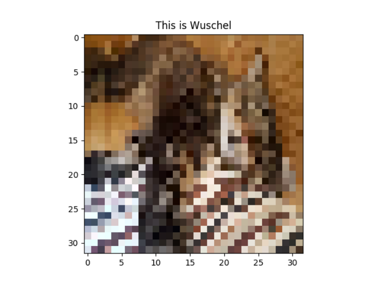
Original Image of Wuschel
(a cat)

Rescaled Image of Wuschel
(still a cat, but very blurred)



Getting Started with Keras - DNN 5

```
from keras.preprocessing.image import load_img, img_to_array
img = load_img('wuschel.jpg', target_size=(32, 32))
img = img_to_array(img).astype('float32') / 255

plt.title('This is Wuschel')
plt.imshow(img, cmap='gray')
plt.show()
```



Getting Started with Keras - DNN 6

```
wuschel_pred = network.predict(np.array([img]))
wuschel_pred
## array([[0.01510279, 0.03711035, 0.04978644, 0.25755176,
##         0.03696484, 0.23572417, 0.00041433, 0.3290789 ,
##         0.00335977, 0.03490658]], dtype=float32)

categories[wuschel_pred.argmax()]
## 'horse'
```

Getting Started with Keras - Transfer Learning 1

```
from keras.applications import VGG16

conv_base = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(32, 32, 3))

from keras import optimizers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

conv_base.trainable = False

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

Getting Started with Keras - Transfer Learning 2

```
model.summary()
## ________________________________________________________________
## Layer (type)                  Output Shape              Param #
## ================================================================
## vgg16 (Model)                 (None, 1, 1, 512)         14714688
## ________________________________________________________________
## flatten_3 (Flatten)           (None, 512)               0
## ________________________________________________________________
## dense_5 (Dense)               (None, 256)               131328
## ________________________________________________________________
## dense_6 (Dense)               (None, 10)                2570
## ================================================================
## Total params: 14,848,586
## Trainable params: 133,898
## Non-trainable params: 14,714,688
## ________________________________________________________________
##
```

Getting Started with Keras - Transfer Learning 3

```
# Train network:
model.fit(train_images, train_labels, epochs=10, batch_size=512)

model.evaluate(test_images, test_labels)
## [1.0980556041717529, 0.615]
```

Getting Started with Keras - Transfer Learning 5

```
wuschel_pred = model.predict(np.array([img]))
wuschel_pred
## array([[9.74539085e-04, 5.73369092e-04, 1.16892666e-01,
##         4.91573721e-01, 1.43246204e-01, 5.08558452e-02,
##         1.90994486e-01, 3.90014239e-03, 1.58492418e-04,
##         8.30588979e-04]], dtype=float32)


categories[wuschel_pred.argmax()]
## 'cat'
```