Deep Learning – The Big Picture

C. Heumann, M. Aßenmacher, D. Schalk

April 17, 2020

Brief History of Deep Learning

Deep Learning Timeline - 1

1943

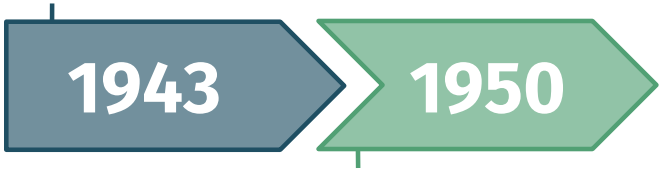
Warren S. McCulloch and **Walter Pitts** tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together.

1943

Deep Learning Timeline - 1

1943

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



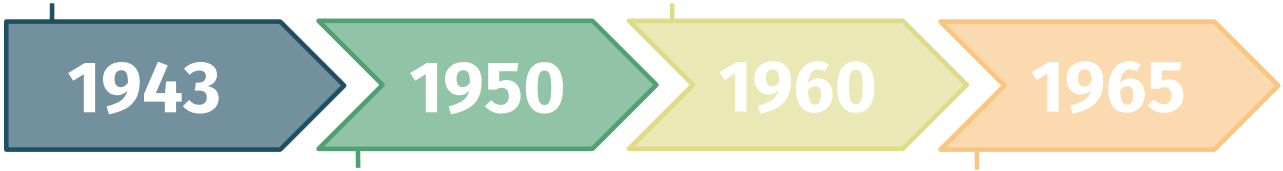
1950

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.

Deep Learning Timeline - 1



1943

Warren S. McCulloch and **Walter Pitts** 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
Warren S. McCulloch and **Walter Pitts** 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.

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.

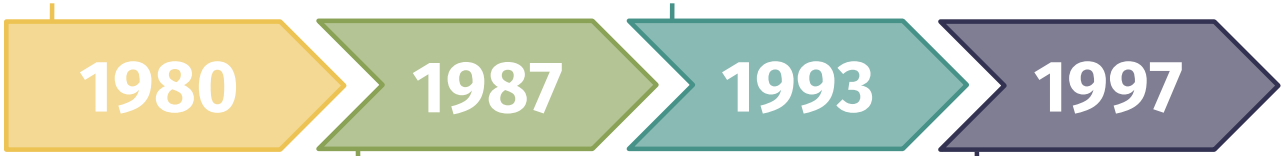
Deep Learning Timeline - 2

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.



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.

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.

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.

Deep Learning Timeline - 3

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2013

Deep Learning Timeline - 3

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework.

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2013 2014

2014 - Attention!

The concept of Attention is introduced by **Bahdanau et al.** as a complement to state-of-the-art neural network architectures.

It is (at first) just used in conjunction with RNNs for machine translation, but will kick-start a completely new path of Deep learning research in NLP in the years to come.

Deep Learning Timeline - 3

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2016 - TPUs

Google developed a processing unit exceptionally suited for deep learning applications.

Its architecture is optimized for performing large matrix multiplications and perfectly integrated to their deep learning library *Tensorflow*.

2014 - Attention!

The concept of Attention is introduced by **Bahdanau et al.** as a complement to state-of-the-art neural network architectures.

It is (at first) just used in conjunction with RNNs for machine translation, but will kick-start a completely new path of Deep learning research in NLP in the years to come.

Deep Learning Timeline - 3

2013 - word2vec

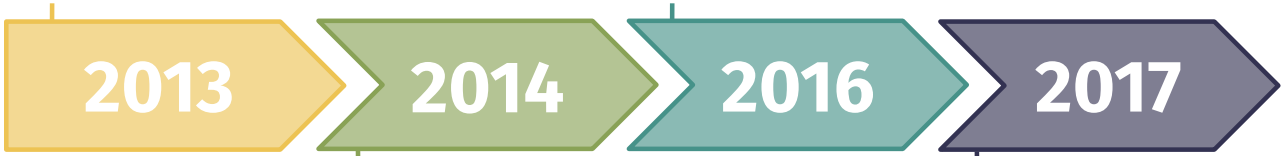
Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2016 - TPUs

Google developed a processing unit exceptionally suited for deep learning applications.

Its architecture is optimized for performing large matrix multiplications and perfectly integrated to their deep learning library *Tensorflow*.



2014 - Attention!

The concept of Attention is introduced by **Bahdanau et al.** as a complement to state-of-the-art neural network architectures.

It is (at first) just used in conjunction with RNNs for machine translation, but will kick-start a completely new path of Deep learning research in NLP in the years to come.

2017 - Attention is all you need!

With their *Transformer* architecture **Vaswani et al.** extend the concept of Attention as an additional building block of neural networks to a purely attention-based architecture.

These developments gave rise to faster (i.e. better parallelizable), bigger (up to billions of parameters) and better models for NLP.

Deep Learning Timeline - 3

2013 - word2vec

Tomas Mikolov et al. publish four papers on vector representations of words constituting the *word2vec* framework

This received very much attention as it revolutionized the way words were encoded for deep learning models in the field of NLP.

2016 - TPUs

Google developed a processing unit exceptionally suited for deep learning applications.

Its architecture is optimized for performing large matrix multiplications and perfectly integrated to their deep learning library *Tensorflow*.

2018 - BERT

BERT (**Devlin et al.**) is a bidirectional contextual embedding model ringing in the new era of transfer learning in NLP.

BERT (and its successors) rely on pre-training on huge unlabelled corpora of text and can afterwards be fine-tuned on many different NLP related tasks.

2014 - Attention!

The concept of Attention is introduced by **Bahdanau et al.** as a complement to state-of-the-art neural network architectures.

It is (at first) just used in conjunction with RNNs for machine translation, but will kick-start a completely new path of Deep learning research in NLP in the years to come.

2017 - Attention is all you need!

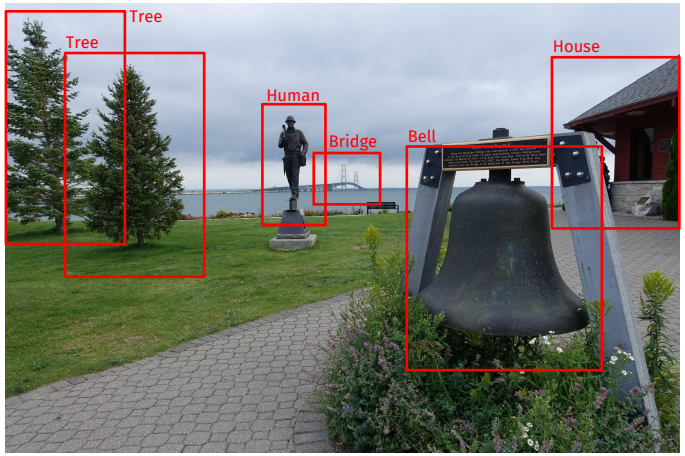
With their *Transformer* architecture **Vaswani et al.** extend the concept of Attention as an additional building block of neural networks to a purely attention-based architecture.

These developments gave rise to faster (i.e. better parallelizable), bigger (up to billions of parameters) and better models for NLP.

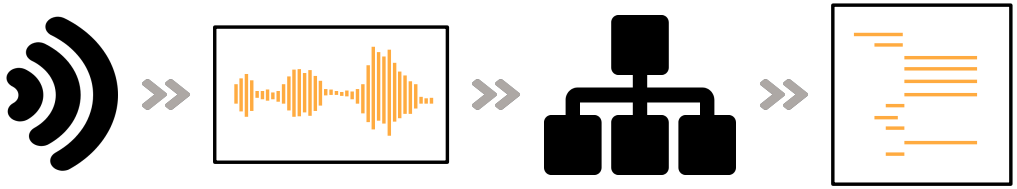
What is Possible With Deep Learning?

Deep Learning Applications

- Supervised learning tasks
- Object Recognition – Seeing:



- Speech Recognition and NLP – Hearing and understanding text:



Deep Learning Applications

- And know we try to teach them creativity:
  - **Music and Text Generation**

    Example of generated text trained on Trump tweets:

    "*The Fake News Media is a great people of the president was a great people of the many people who would be a great people of the president was a big crowd of the statement of the media is a great people of the people of the statement of the people . . .*"

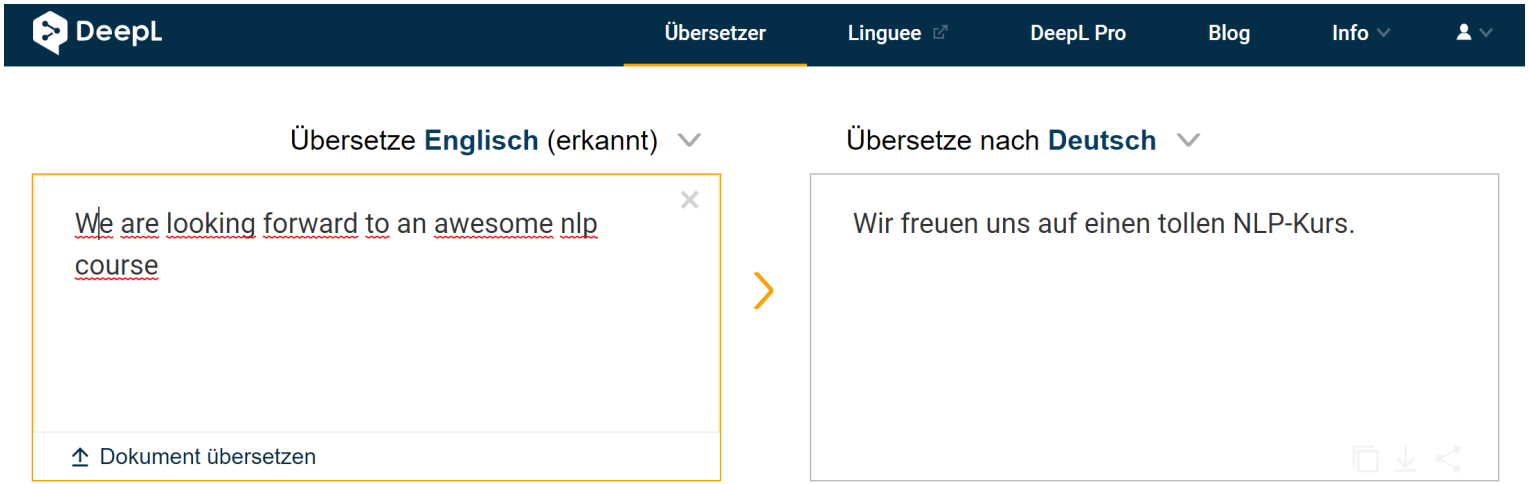
    . . . could be better.
  - **Neural Style Transfer**

  - . . .

Some remarks on NLP

- The world's largest companies are investing heavily in NLP: Google, Facebook, Amazon, Apple, Microsoft, ...
- But still: a lot of (yet) unsolved problems and challenges
- Innovations are more and more open-sourced
- There's a variety of different applications of NLP in everyday life you might not be aware of
- Once you see them, it soon becomes obvious

Neural Machine Translation

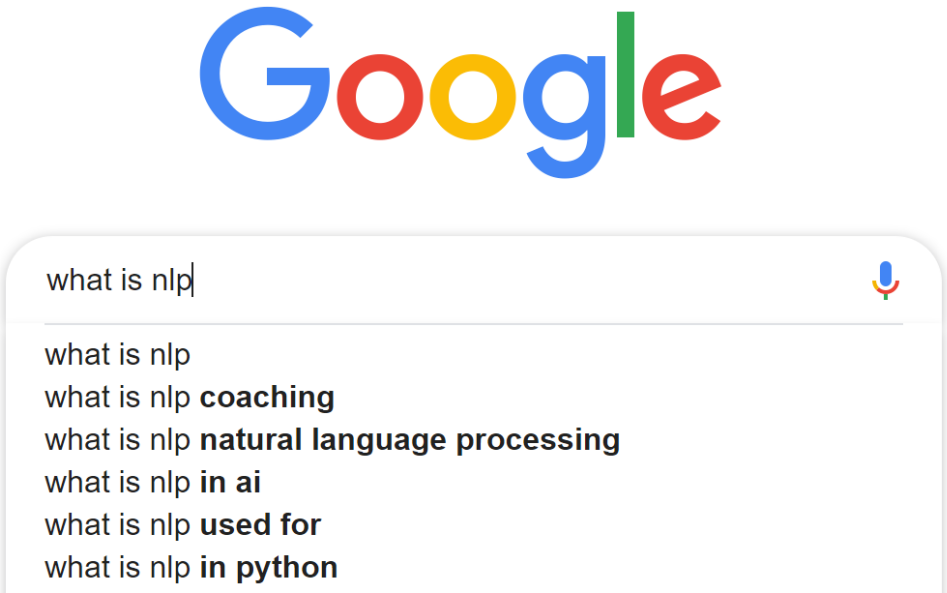


- Two different NLP tasks are hiding in this picture..
- **Underlying task**: Text generation

A brief History of Machine Translation

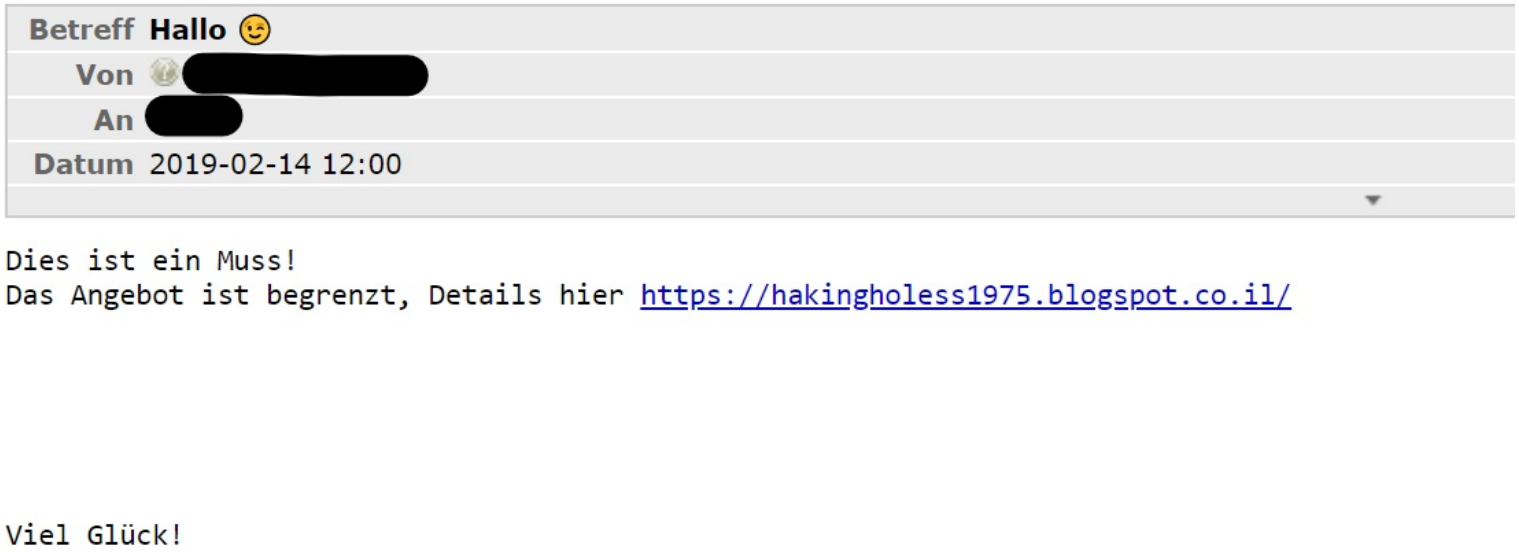
- **Rule-Based Machine Translation** (50s – 80s)
  - Dictionaries + Grammatical Rules
- **Example-Based Machine Translation** (80s – 90s)
  - First suggested by Makoto Nagao (1984)
  - Based on bilingual text corpora
- **Statistical Machine Translation** (90s – 10s)
  - Driven by IBM guys
- **Neural Machine Translation** (last few years)
  - Based on neural networks (LSTMs, Transformers)

Auto Completion



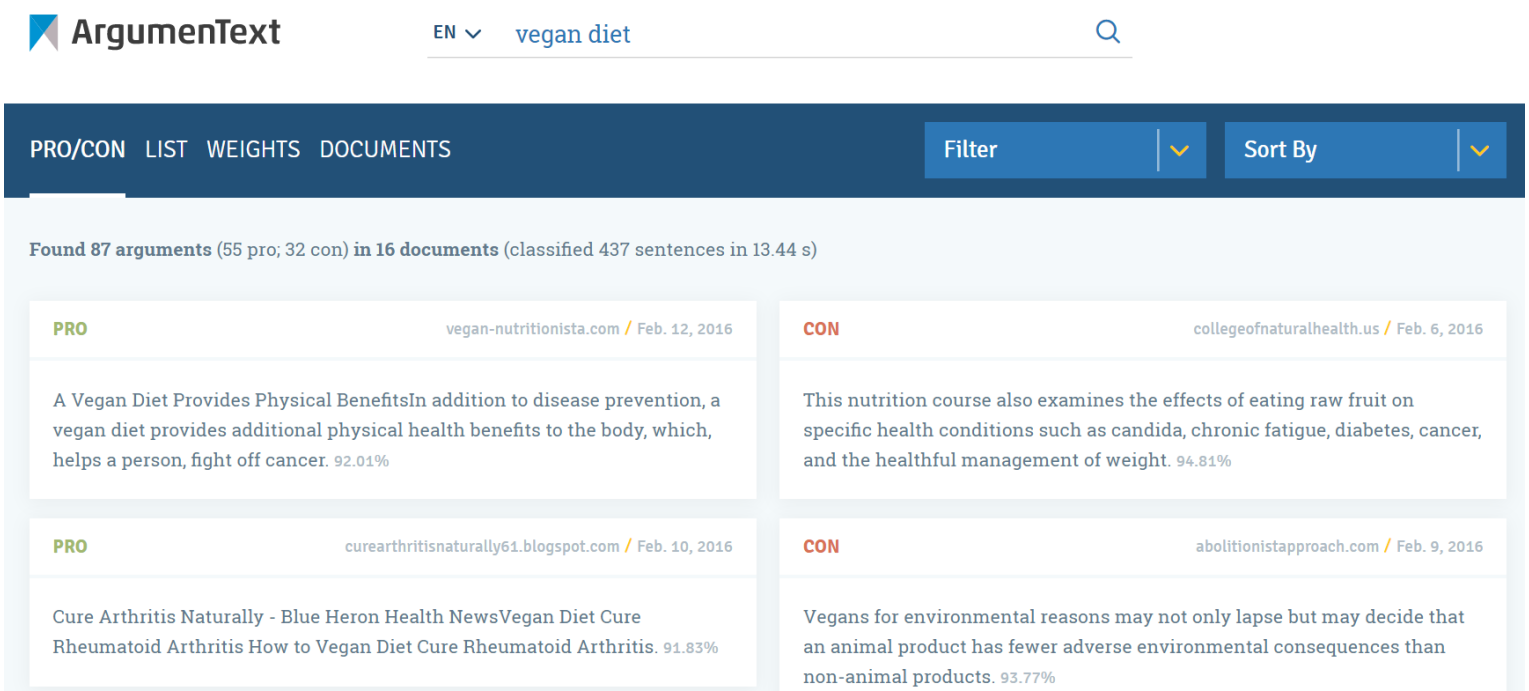
- You might also know this from your smartphone keyboard
- **Underlying task**: Predict the next word given the previous words (+ individual browsing history)

Spam filters



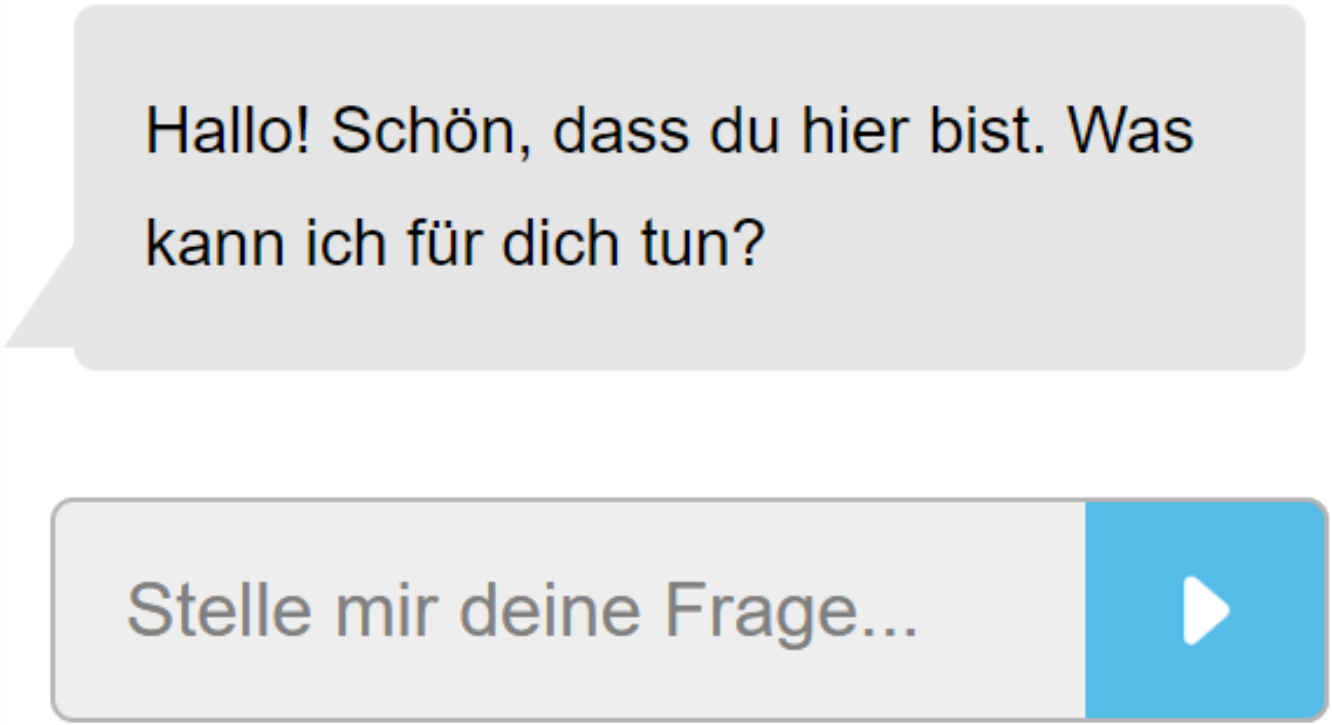
- Indeed, spam filters are often surprisingly simple systems
- **Underlying task**: Document classification (Naive Bayes or Logistic Regression)

Argument Mining



- Detection and ranking of arguments pro/contra a certain topic
- **Underlying task**: Document classification and/or Sequence tagging

Chatbots



- We experience chatbots on a growing number of websites
- **Underlying task**: Natural language understanding / Text generation

Information Retrieval

Alexa, what is the capital of Australia?

- **Underlying task**: Question Answering / Information extraction

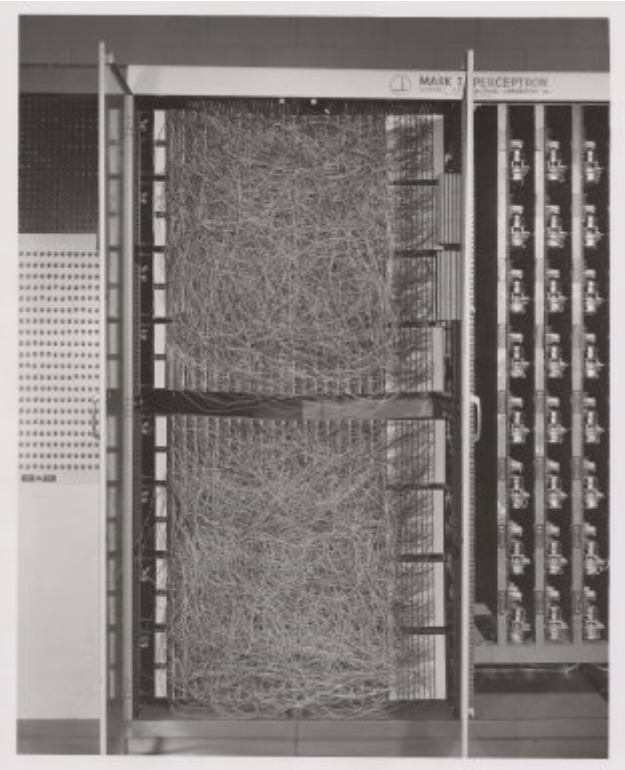
(Some) Further examples

- **Sentiment analysis / Text classification**
  - Determine, whether a document is positive/neutral/negative
- **Text Generation**
  - Example: *OpenAI GPT2*
- **Named Entity Recognition**
  - Search for certain entities (e.g. person, location, etc.)
- **Image Captioning**
  - Hybrid task (Computer Vision + NLP)
- **Dependency Parsing**
  - Determine the grammatical structure of text
- **Part-of-speech tagging**
  - Tagging words according to their role in a text

How Deep Learning Works

The Perceptron

- The perceptron was invented by Frank Rosenblatt 1957.

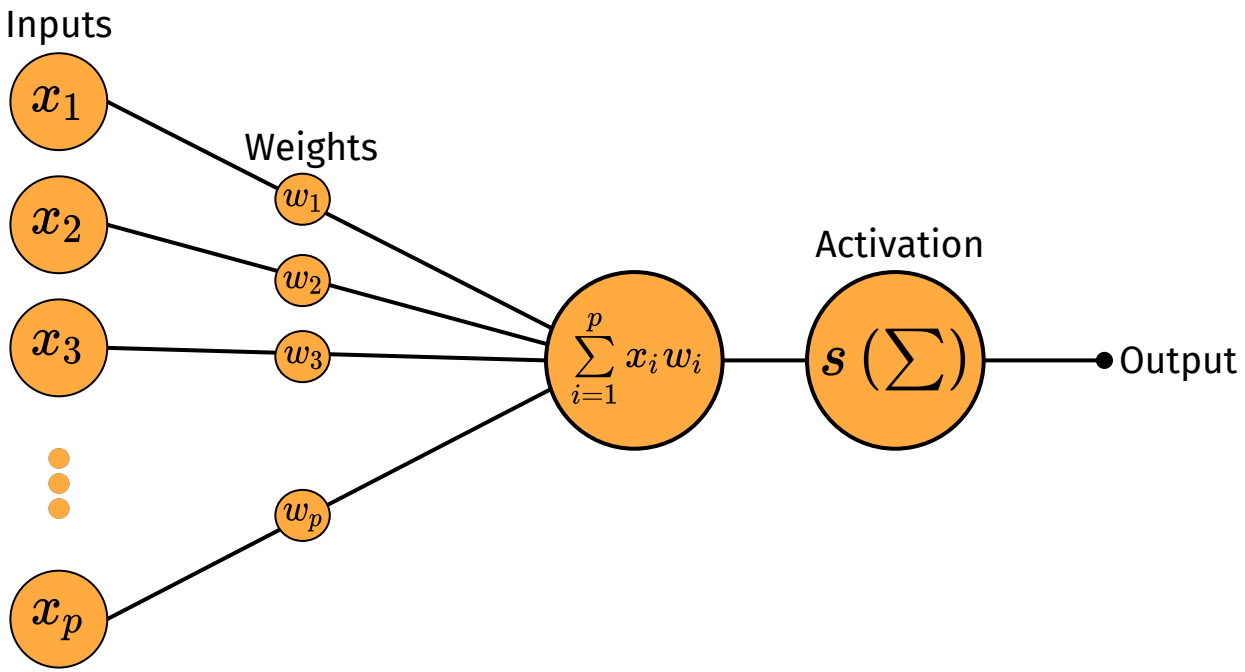


The Mark I Perceptron

- It is the basic computational unit for neural networks.

Singlelayer Perceptron

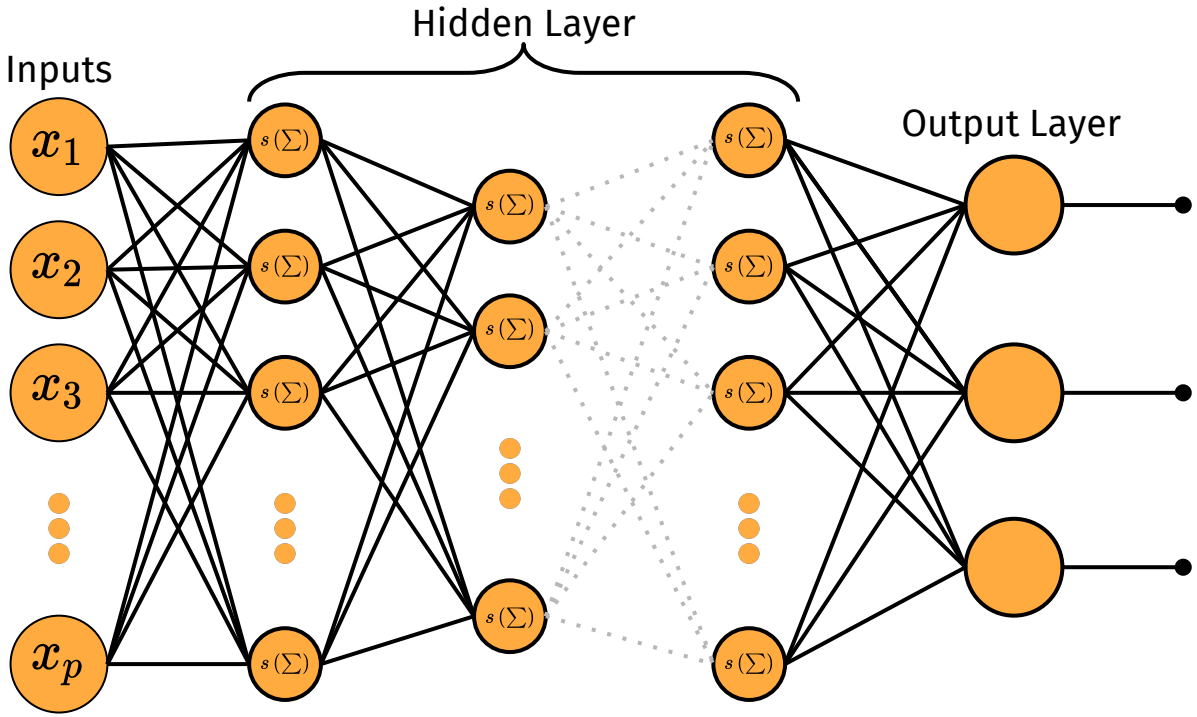
- Weighted sum of input values transformed by an activation function s
- If s is the sigmoid function $(1 + \exp \sum)^{-1}$, then the perceptron does exactly the same as the logistic regression



Multilayer Perceptron

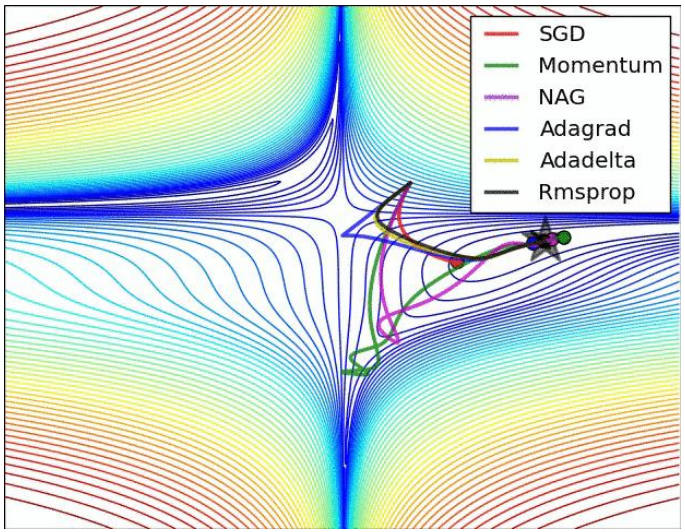
- Stacking of multiple perceptrons
- Corresponds to stacking GLM models
- Number of parameter grows very fast

  $\rightarrow$ Optimization becomes more difficult



Optimizer

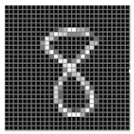
- Having that much parameter/weights to find, standard optimizer like Gradient Descent may fail
- Therefore, much effort was spend to get faster optimizer like adam, rmsprop, etc.:

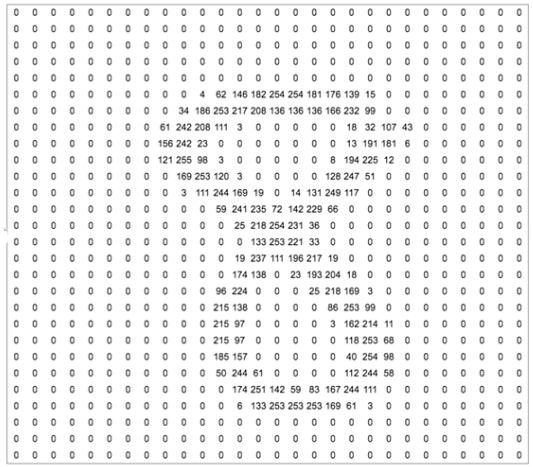


Source: Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Input Source: Images

- Deep Learning is well known for handling more complex data structures like images, text, voice recordings etc.
- When using images as input, one image becomes one observation:

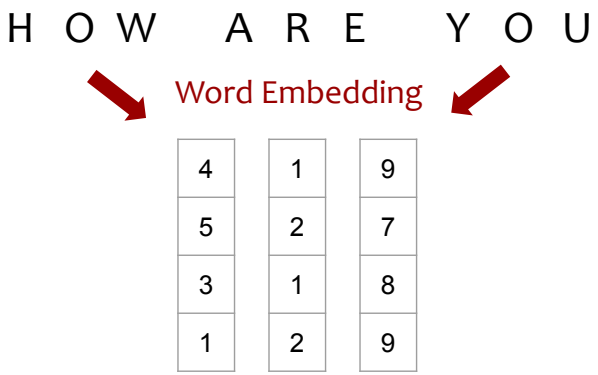




- But, how to analyse images? We have to make use of the spatial structure, but how?

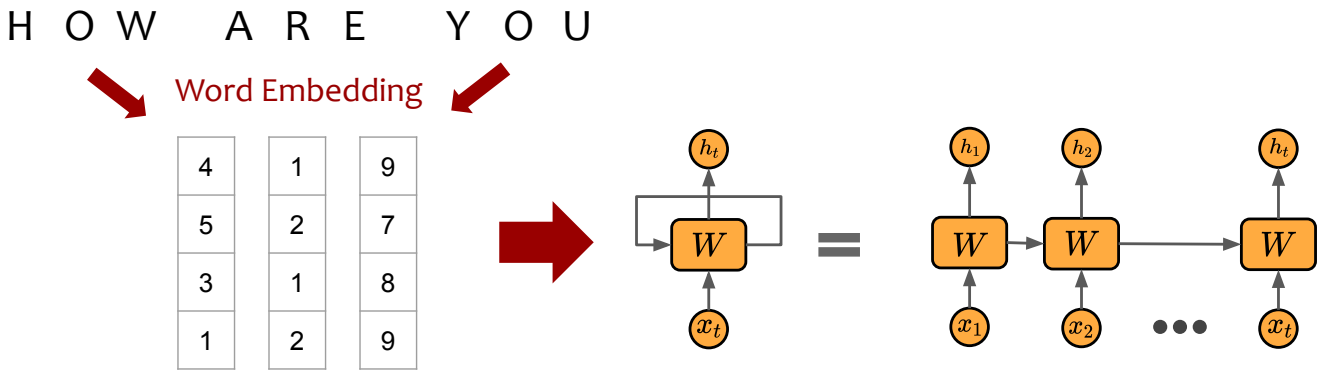
Input Source: Text

- Another typical data format are texts:



Input Source: Text

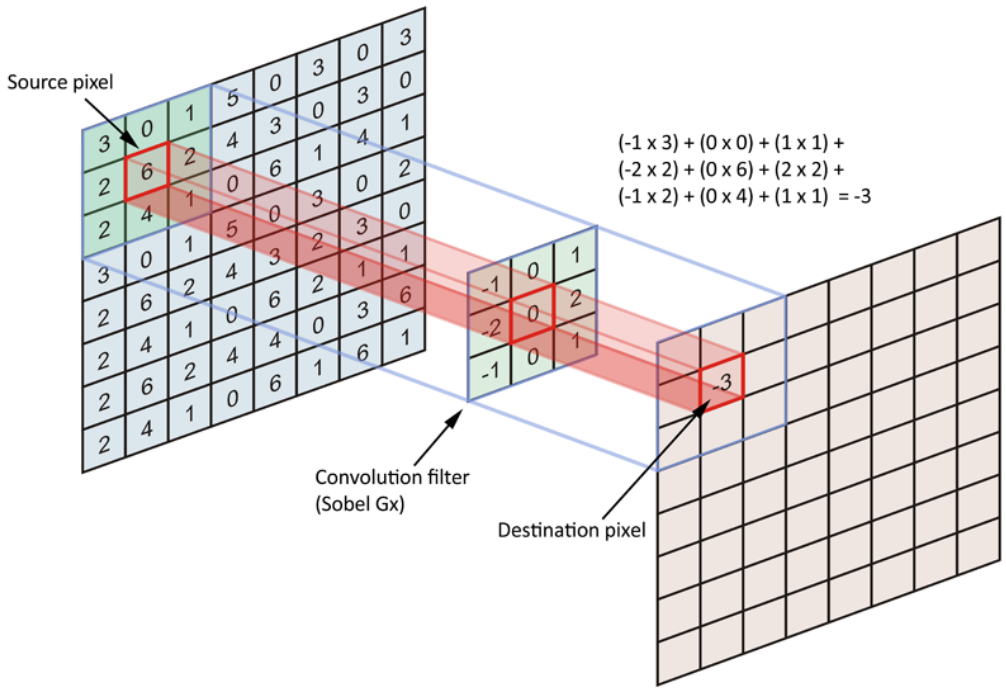
- Another typical data format are texts:



- But, how to analyse text sources? We also have to consider the recurrent dependencies of text, but how?

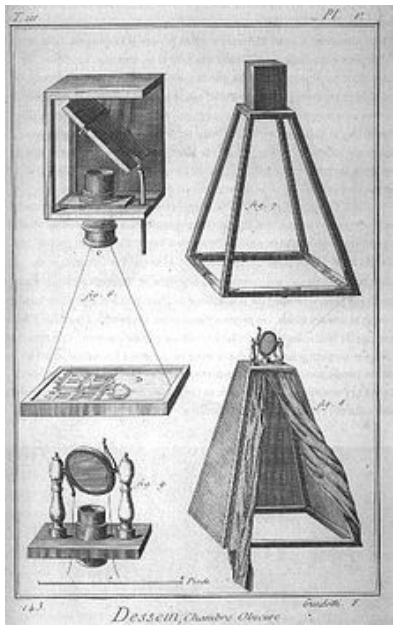
Convolution - As Operation

- Extraction of features of the input data:



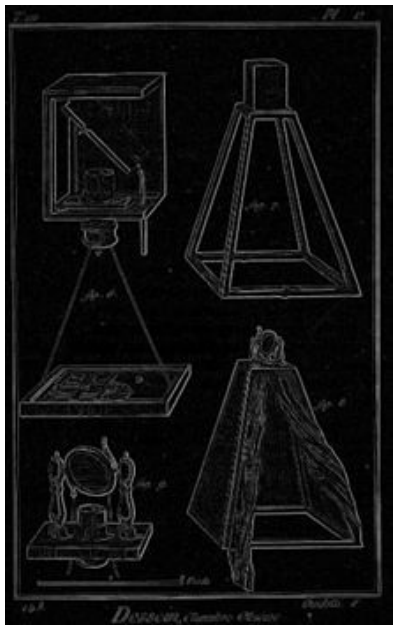
Convolution - As Operation

Input Image



G_x

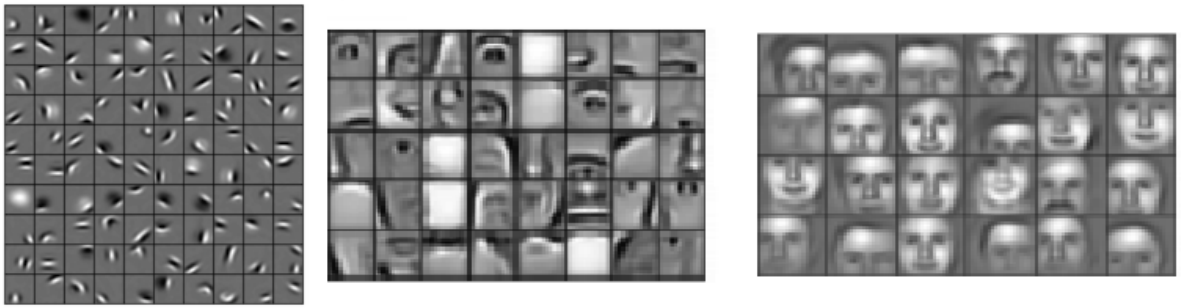
Generated Image/Feature



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

Convolution - In Neural Networks

- Filters, like the Sobel filter, are obtained by thinking long and hard about what should be obtained from a filter (like edges).
- In Deep Learning these filters are learned during the training and automatically generate new features:



Source: Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. Communications of the ACM, 54(10), 95-103. *"Each feature can be thought of as a filter, which filters the input image for that feature (a nose). If the feature is found, the responsible unit or units generate large activations, which can be picked up by the later classifier stages as a good indicator that the class is present."*

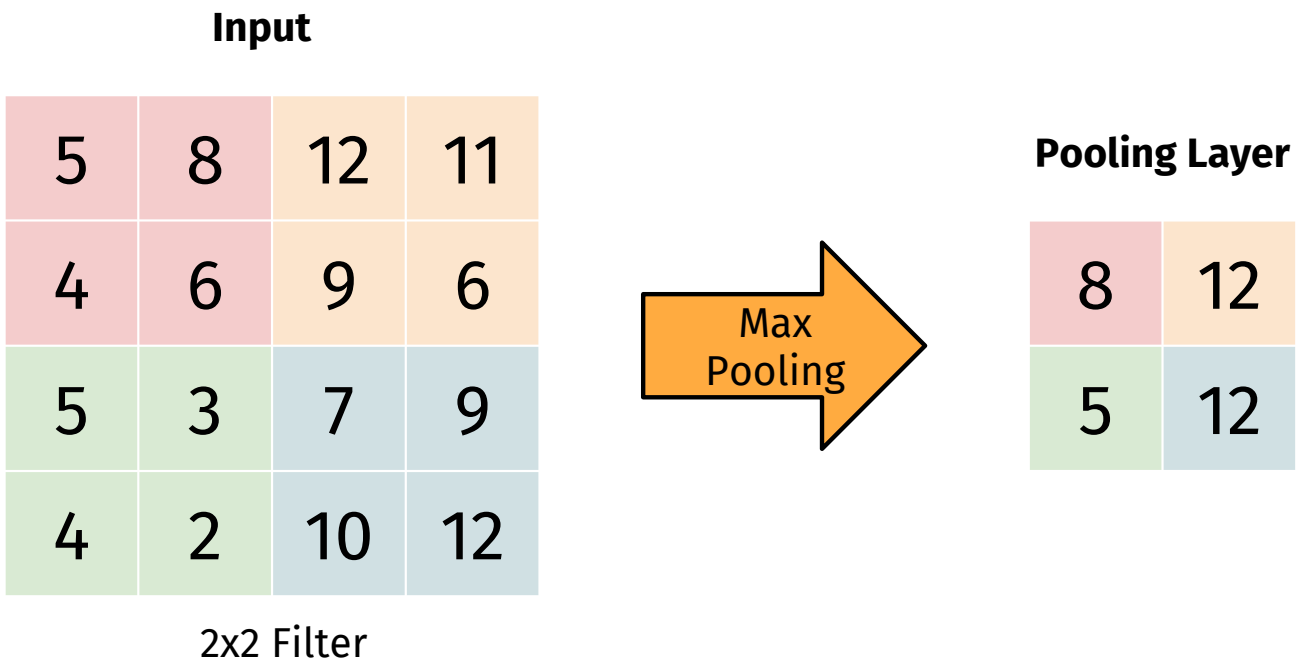
Pooling

Down-sampling of images:

→ Less weights to fit

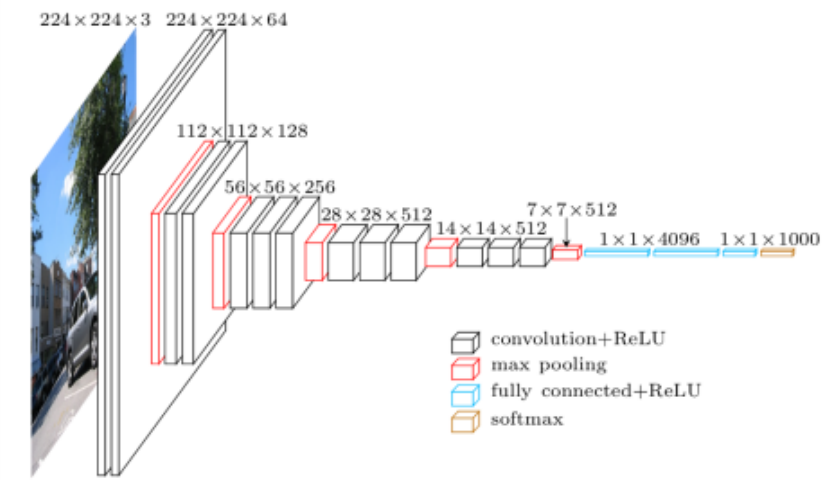
→ Smaller net size (less memory usage, faster fitting process)

→ Reduces overfitting



Lets Get Deep

- The power of Deep Neural Nets comes from chaining 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

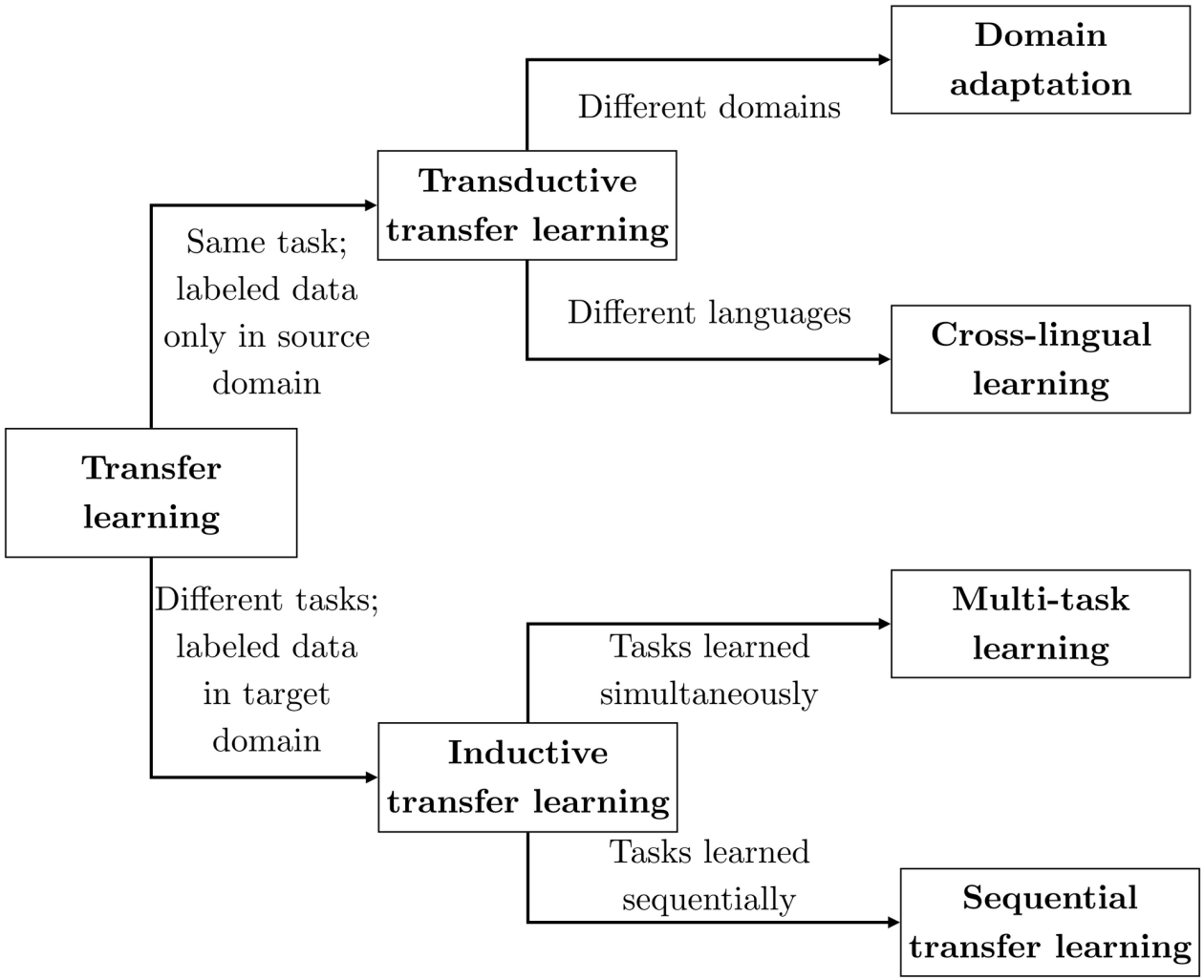
Model	Size	Parameters	Depth
Xception	88 MB	22,910,480	126
VGG16	528 MB	138,357,544	23
VGG19	549 MB	143,667,240	26
ResNet50	99 MB	25,636,712	168
InceptionV3	92 MB	23,851,784	159
InceptionResNetV2	215 MB	55,873,736	572
MobileNet	16 MB	4,253,864	88
MobileNetV2	14 MB	3,538,984	88
DenseNet121	33 MB	8,062,504	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

A primer on Transfer Learning

- What are "*Pre Trained models*"?
  - Knowledge gained from solving one task can be potentially transfered to other tasks
  - Pre Training on a relatively general data set (of images)
  - Fine Tuning on a task specific data set at hand
  - *For images:*
    - Pre Trained model has already learned something about shapes and edges
    - Potentially also already about more complex concepts
  - *For text:*
    - Pre Trained model has already some knowledge of syntax & semantic

A primer on Transfer Learning



Source: https://ruder.io/thesis/index.html

A primer on Transfer Learning

- Good points to start:
  - https://ruder.io/state-of-transfer-learning-in-nlp/
  - Many tutorials, etc. available online: https://lmgtfy.com/?q=transfer+learning+nlp

Challenges in Deep Learning

Challenges in Deep Learning

- **Finding a Good Architecture**

  Having that many possibilities of combining hidden layer, optimizer, and activation functions we run into the problem of finding a good architecture.

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

  → Architecture search

- **Expensive Training**

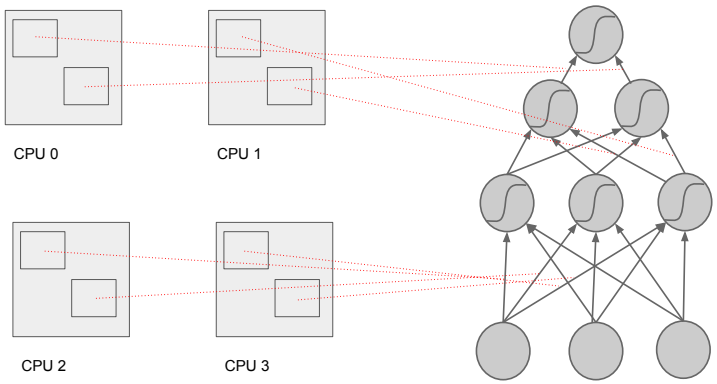
  Training of DNNs require billions of simple operations, hence training **one** DNN might take weeks.

  → Use GPU server for serious applications. Why? See next slides.

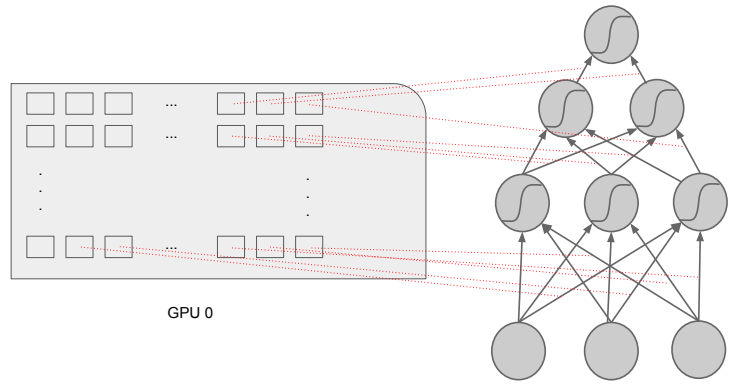
Tech Stack

Hardware

- 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.

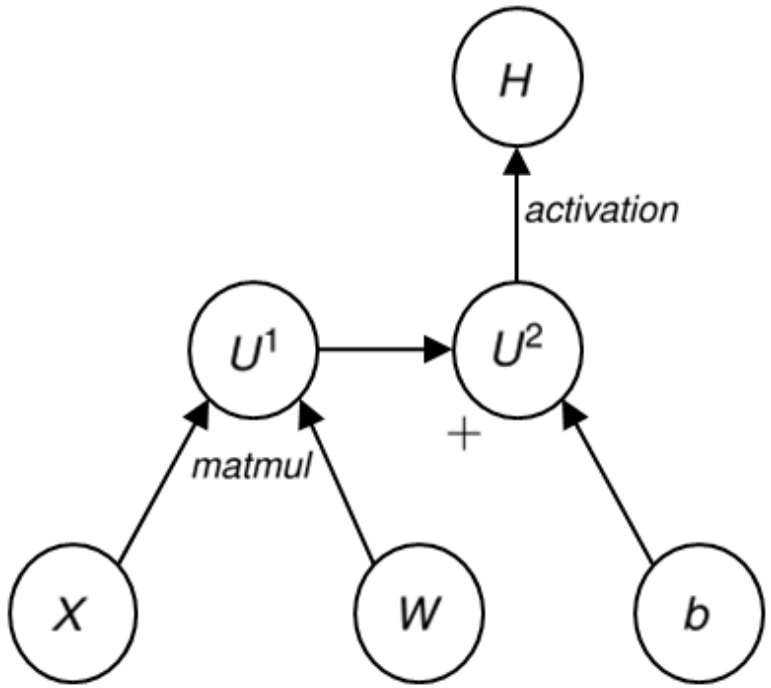


A single GPU can do thousands of simple parallel computations.

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 $f(XW + b)$.

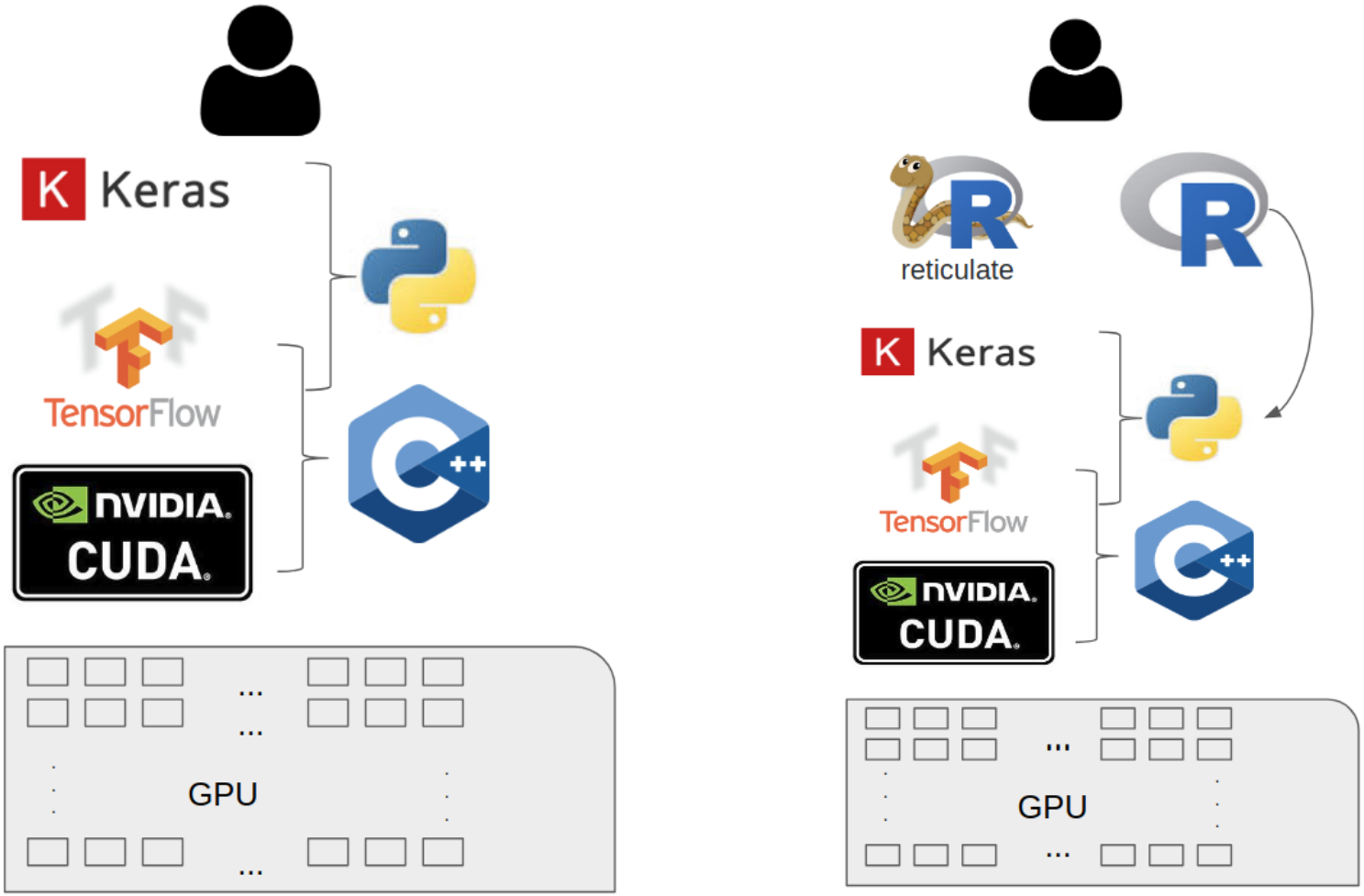
TensorFlow

- 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

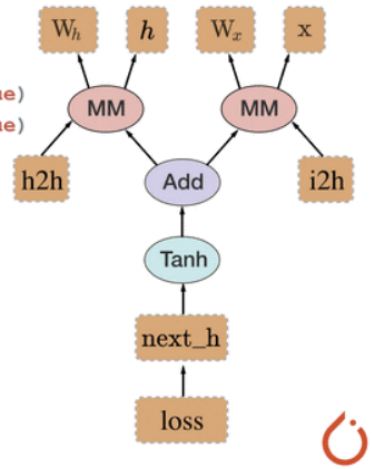
- Open-source framework developed by facebook.
- Reimplementation of Torch.
- Only support for Python.
- Widely used and well documented.

PyTorch

- PyTorch uses dynamic computational graphs. Instead of define and run the graph, the graph is defined by run:

A graph is created on the fly

```
W_h = torch.randn(20, 20, requires_grad=True)
W_x = torch.randn(20, 10, requires_grad=True)
x = torch.randn(1, 10)
prev_h = torch.randn(1, 20)

h2h = torch.mm(W_h, prev_h.t())
i2h = torch.mm(W_x, x.t())
next_h = h2h + i2h
next_h = next_h.tanh()

loss = next_h.sum()
```



Source: https://github.com/pytorch/pytorch

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`.
- The GPU version requires `CUDA` and `cuDNN`. Installation is then done by

```
pip install --upgrade tensorflow-gpu
```

Getting Started with Keras - Overview

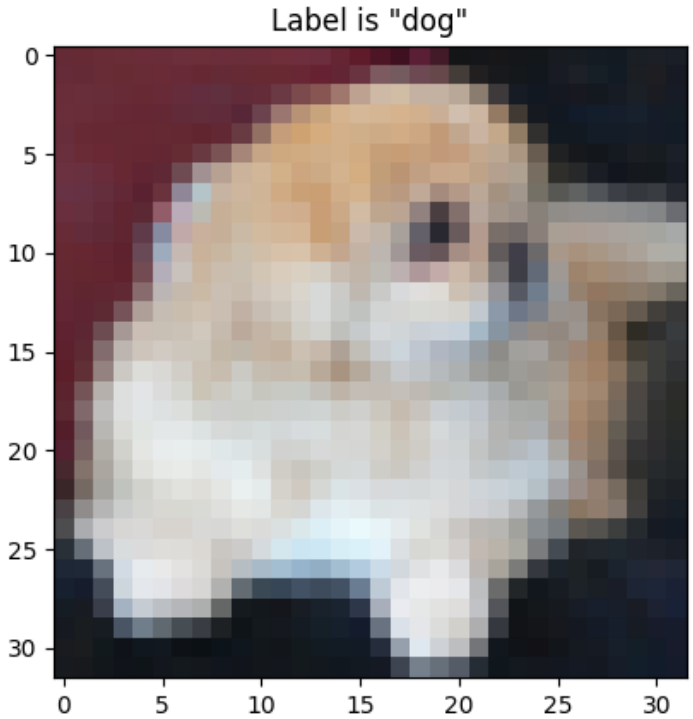
- Instead of introducing theory, we want to get into the topic by applying it.
- We using examples similar to the book **Deep Learning with Python** that are prepared as **notebooks**.
- **But**: When using something new, e.g. a convolution layer or optimizer, try to understand what it does and why it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

```
python      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=20, batch_size=128)

network.evaluate(test_images, test_labels)
## [3.585078500747681, 0.4522]
```

Getting Started with Keras - DNN 4

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:

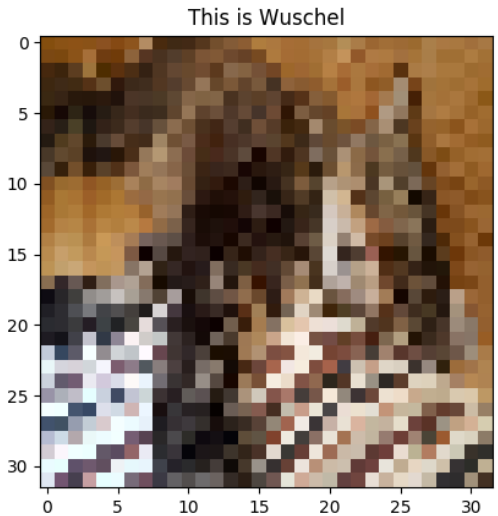
Original Image of Wuschel
(a cat)

Reshape to
32 x 32 Pixel

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=20, batch_size=128)

model.evaluate(test_images, test_labels)
## [1.1461146575927734, 0.6175]
```

Getting Started with Keras - Transfer Learning 5

```
wuschel_pred = model.predict(np.array([img]))
wuschel_pred
## array([[1.4046730e-03, 1.0323318e-03, 1.2766185e-01,
##         4.6453097e-01, 5.1652599e-02, 5.0368346e-02,
##         2.9299492e-01, 1.0088098e-02, 2.4648944e-05,
##         2.4157017e-04]], dtype=float32)

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

What Next?

Improving Performance:

- Choosing the number of epochs using early stopping
- Data augmentation (increasing size of training data by rotating, scaling, flipping, . . .)

More Difficult Tasks:

- RNN, LSTM, GAN, Reinforcement Learning
- Natural Language Processing