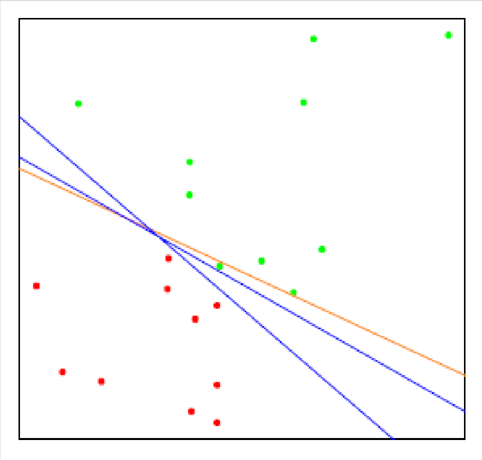


Deep Learning

Using a Convolutional Neural Network

Dr. – Ing. Morris Riedel

Adjunct Associated Professor

School of Engineering and Natural Sciences, University of Iceland

Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 2

Convolutional Neural Networks & Tools

November 30th, 2017

Ghent, Belgium

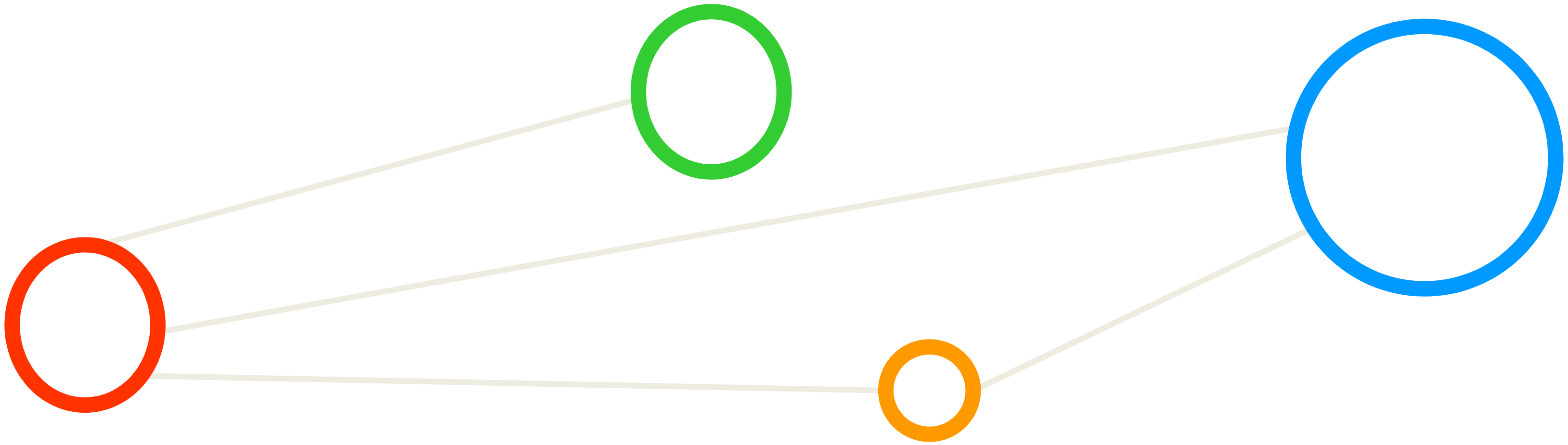
UNIVERSITY OF ICELAND

SCHOOL OF ENGINEERING AND NATURAL SCIENCES

FACULTY OF INDUSTRIAL ENGINEERING,
MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH FORSCHUNGSZENTRUM

Outline



Outline of the Course

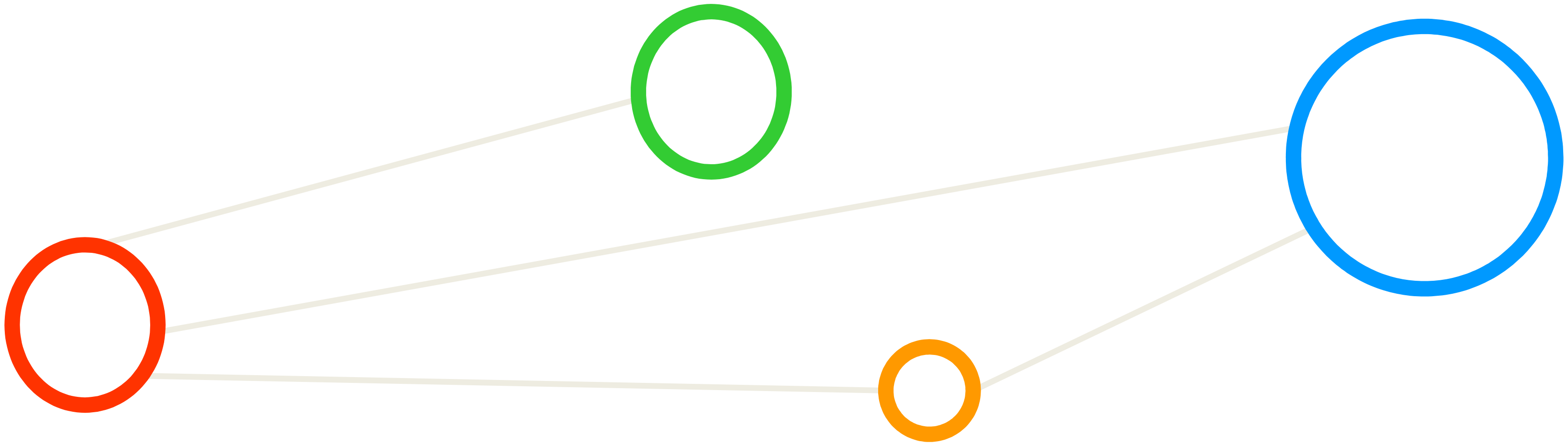
1. Deep Learning Fundamentals & GPGPUs
2. Convolutional Neural Networks & Tools
3. Convolutional Neural Network Applications
4. Convolutional Neural Network Challenges
5. Transfer Learning Technique
6. Other Deep Learning Models & Summary

Outline

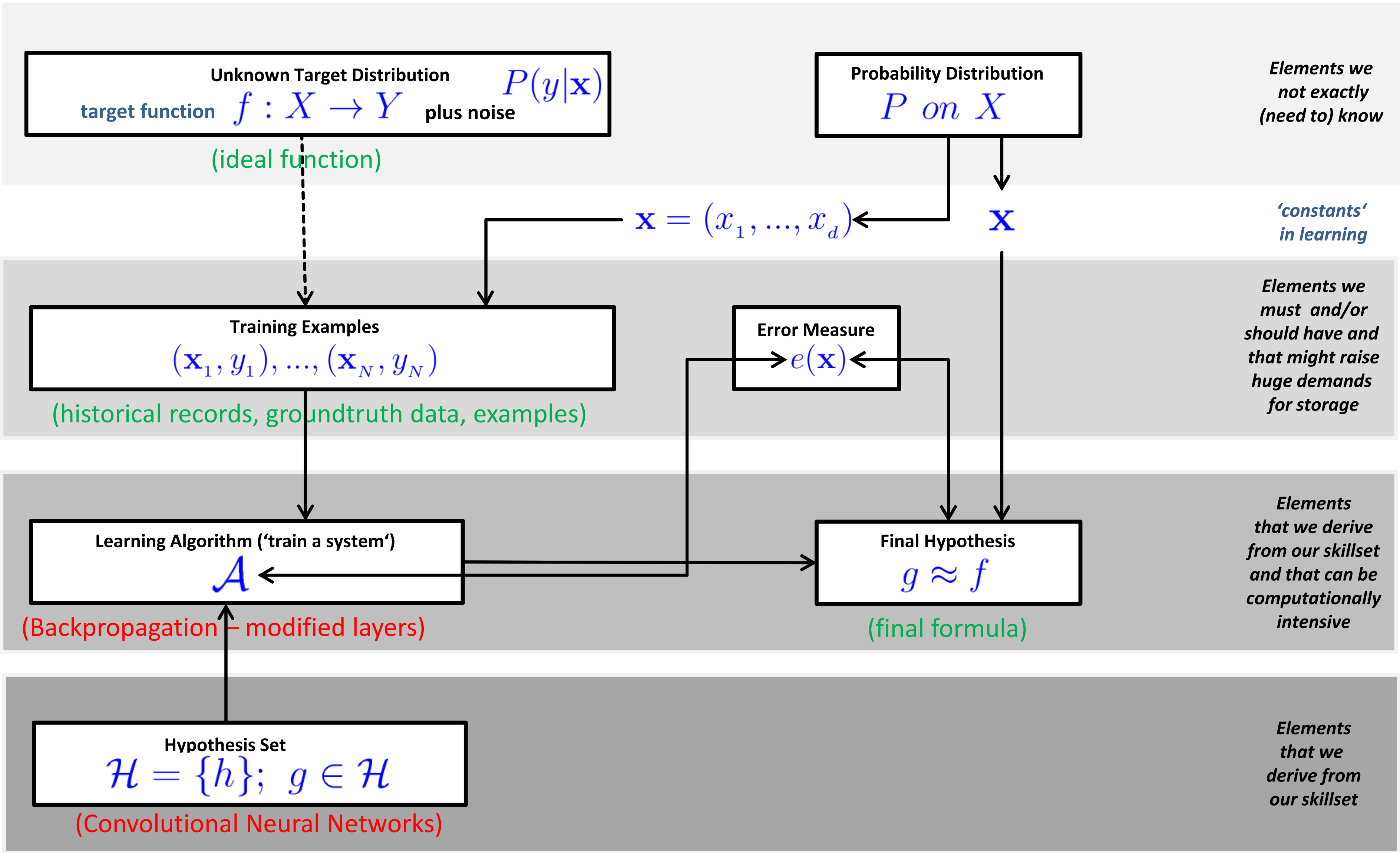
- Convolutional Neural Networks (CNNs)
  - Basic Principles & MNIST Application Example
  - Local Receptive Fields & Sliding
  - Revisit ANN Overfitting & Weight Problem
  - Shared Weights & Feature Maps
  - Advanced Application Examples

- Deep Learning Toolset
  - Timeline of Selected Relevant Tools
  - Low-level Deep Learning Libraries
  - Tensorflow, Caffe and Theano
  - Tensorflow Computational Graph
  - What is a Tensor & Chain Rule

Convolutional Neural Networks (CNNs)



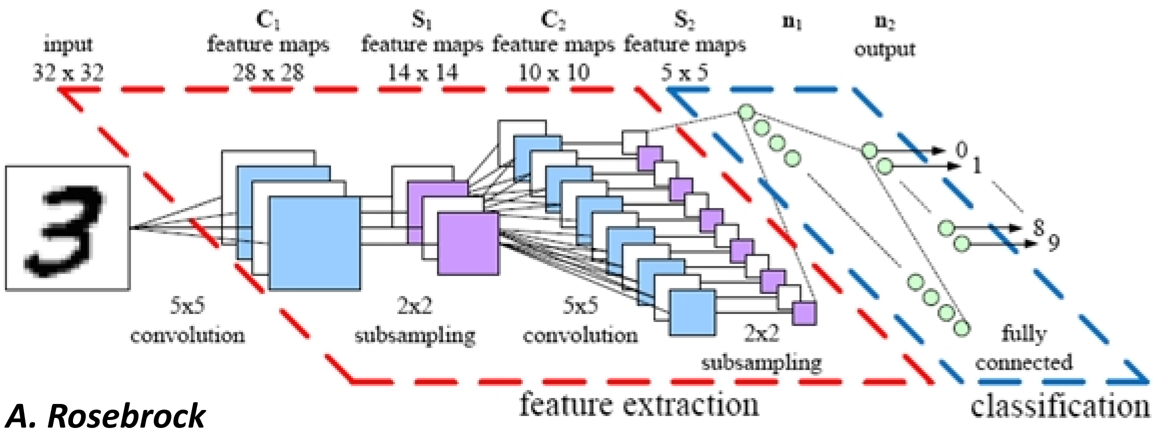
Solution Tools: Convolutional Networks Learning Model



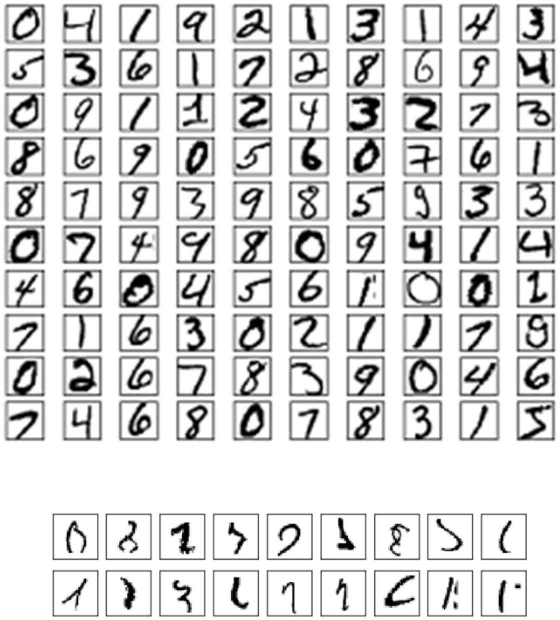
CNNs – Basic Principles

- **Convolutional Neural Networks (CNNs/ConvNets) implement a connectivity pattner between neurons inspired by the animal visual cortex and use several types of layers (convolution, pooling)**
- **CNN key principles are local receptive fields, shared weights, and pooling (or down/sub-sampling)**
- **CNNs are optimized to take advantage of the spatial structure of the data**

- Simple application example
  - MNIST database written characters
  - Use CNN architecture with different layers
  - Goal: automatic classification of characters



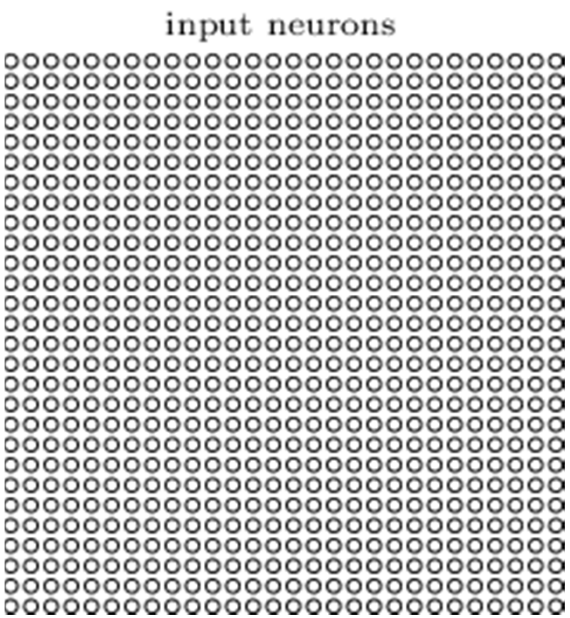
[3] A. Rosebrock



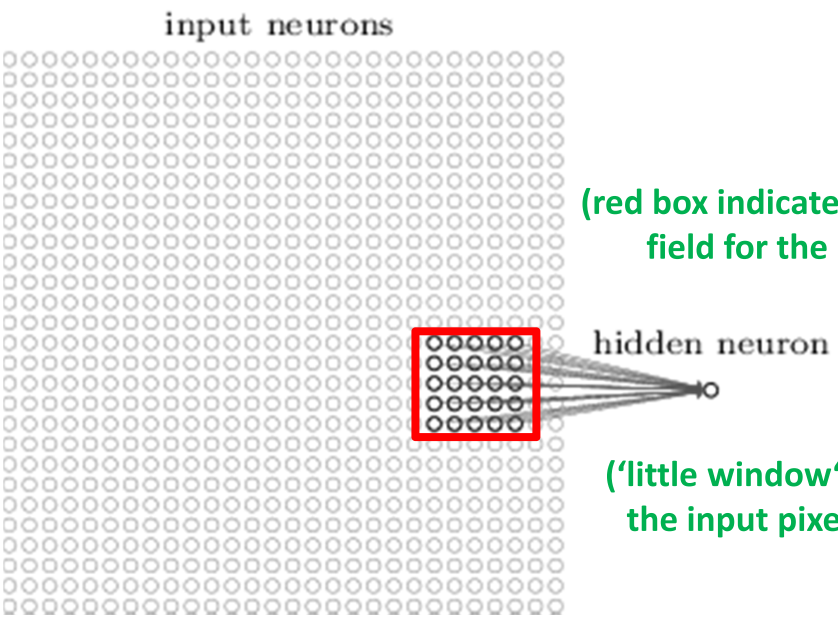
[1] M. Nielsen

CNNs – Principle Local Receptive Fields

- MNIST dataset example
  - 28 * 28 pixels modeled as square of neurons in a convolutional net
  - Values correspond to the 28 * 28 pixel intensities as inputs



(28 * 28 pixel image)



(red box indicate the local receptive field for the hidden neuron)

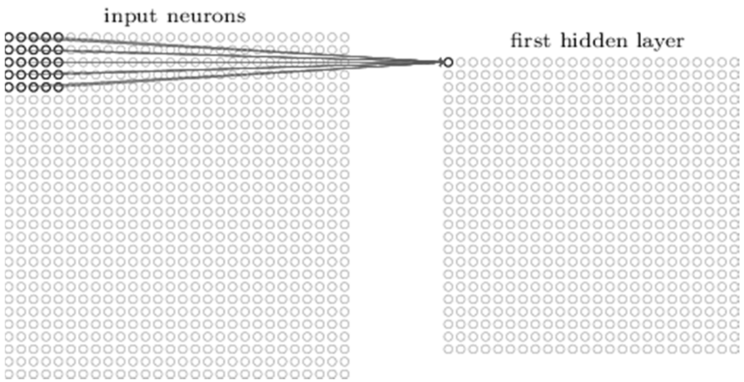
('little window' on the input pixels)

(5 * 5 local connectivity)

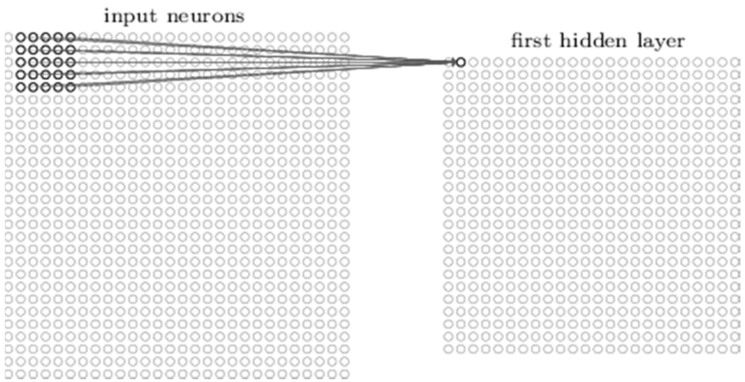
[1] M. Nielsen

CNNs – Principle Local Receptive Fields & Sliding

- MNIST database example
  - Apply stride length = 1
  - Different configurations possible and depends on application goals
  - Creates 'feature map' of 24 * 24 neurons (hidden layer)



(28 * 28 pixel image) (24 * 24 feature map)

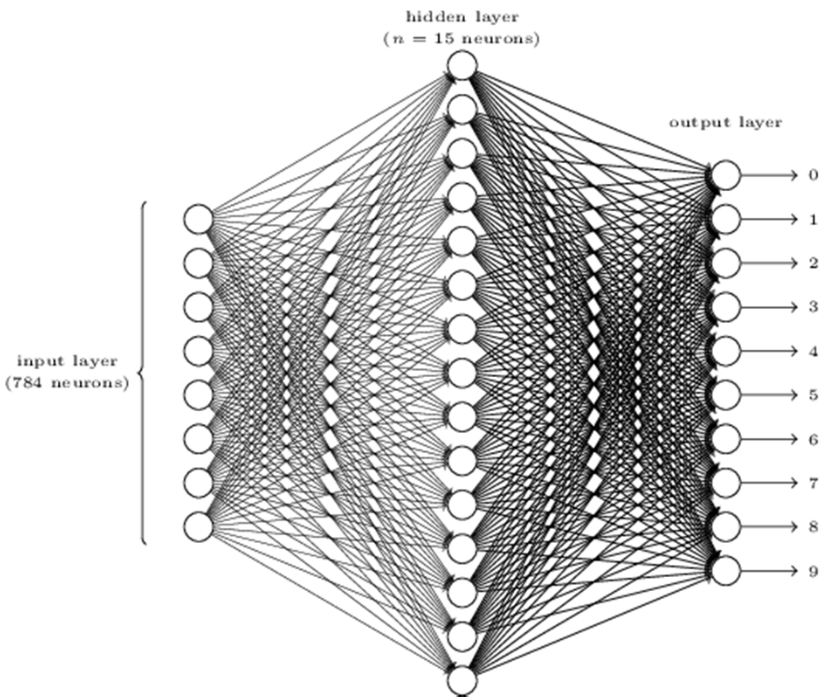


(28 * 28 pixel image) (24 * 24 feature map)

[1] M. Nielsen

CNNs –Example with an ANN with risk of Overfitting

- MNIST database example
  - CNN: e.g. 20 feature maps with 5 * 5 (+bias) = 520 weights to learn
  - Apply ANN that is fully connected between neurons
  - ANN: fully connected first layer with 28 * 28 = 784 input neurons
  - ANN: e.g. 15 hidden neurons with 784 * 15 = 11760 weights to learn



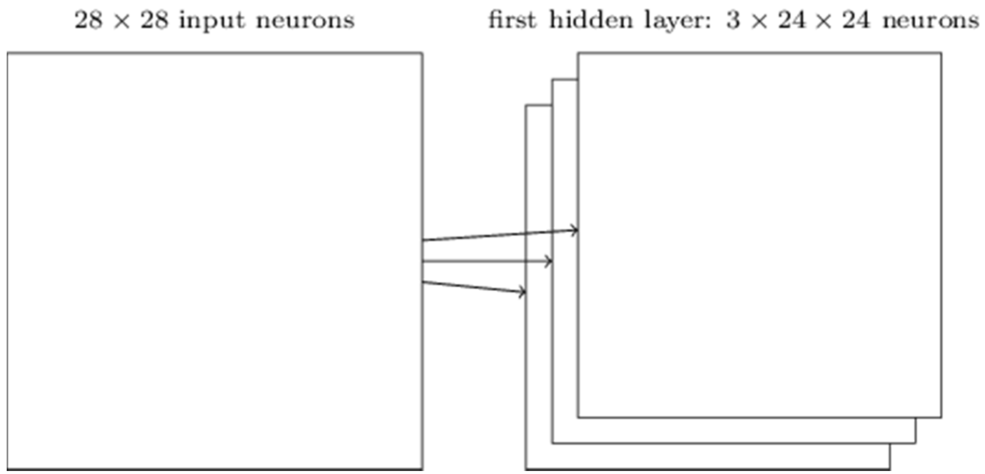
(eventually lead to overfitting and much computing time)

[1] M. Nielsen

Exercises – ANN Example Revisited – Count Parameters

CNNs – Principle Shared Weights & Feature Maps

- Approach
  - CNNs use same shared weights for each of the 24 * 24 hidden neurons
  - Goals: significant reduction of number of parameters (prevent overfitting)
  - Example: 5 * 5 receptive field → 25 shared weights + shared bias
- Feature Map
  - Detects one local feature
  - E.g. 3: each feature map is defined by a set of 5 * 5 shared weights and a single shared bias leading to 24 * 24
  - Goal: The network can now detect 3 different kind of features **(many more in practice)**



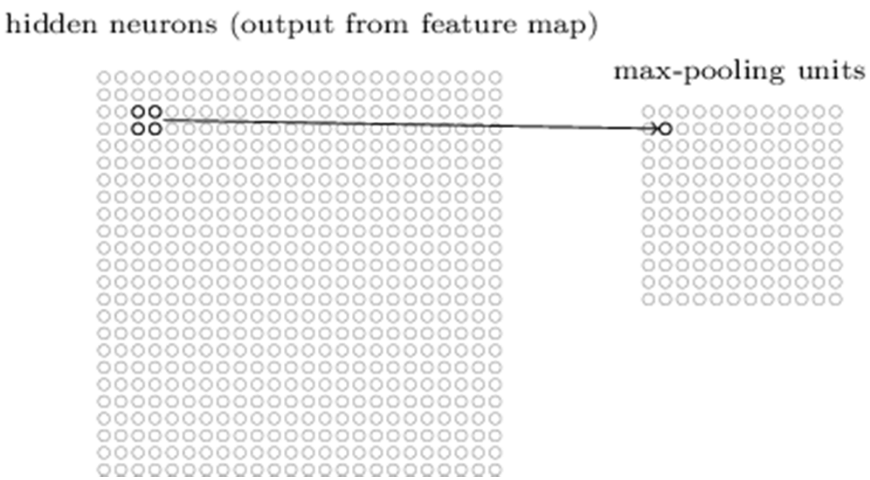
(shared weights are also known to define a kernel or filter)

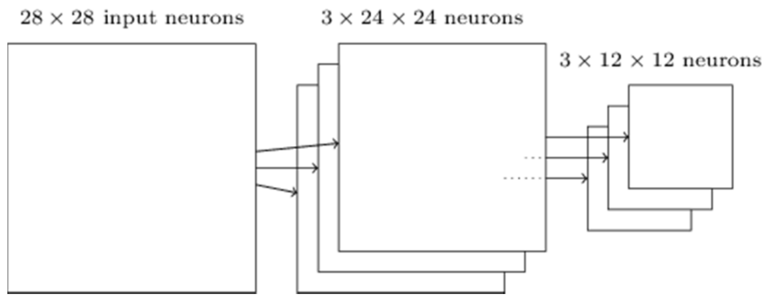
  - Benefit: learned feature being detectable across the entire image

[1] M. Nielsen

CNNs – Principle of Pooling

- ‘Downsampling‘ Approach
  - Usually applied directly after convolutional layers
  - Idea is to simplify the information in the output from the convolution
  - Take each feature map output from the convolutional layer and generate a condensed feature map
  - E.g. Pooling with 2 * 2 neurons using ‘max-pooling‘
  - Max-Pooling outputs the maximum activation in the 2 * 2 region

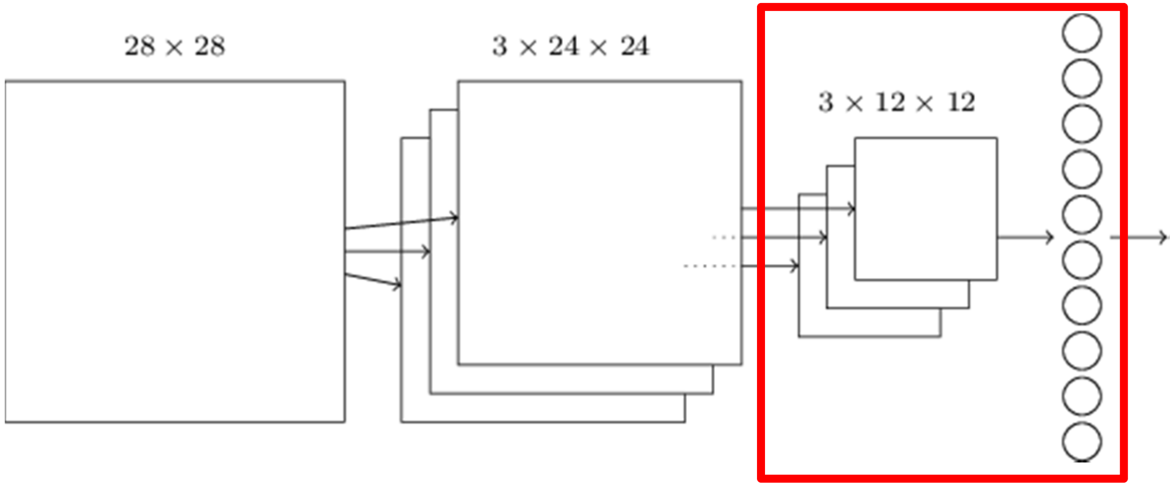




[1] M. Nielsen

CNN – Application Example MNIST

- MNIST database example
  - Full CNN with the addition of output neurons per class of digits
  - Apply ‘fully connected layer‘: layer connects every neuron from the max-pooling outcome layer to every neuron of the 10 out neurons
  - Train with backpropagation algorithm (gradient descent), only small modifications for new layers



  - Approach works, except for some bad training and test examples

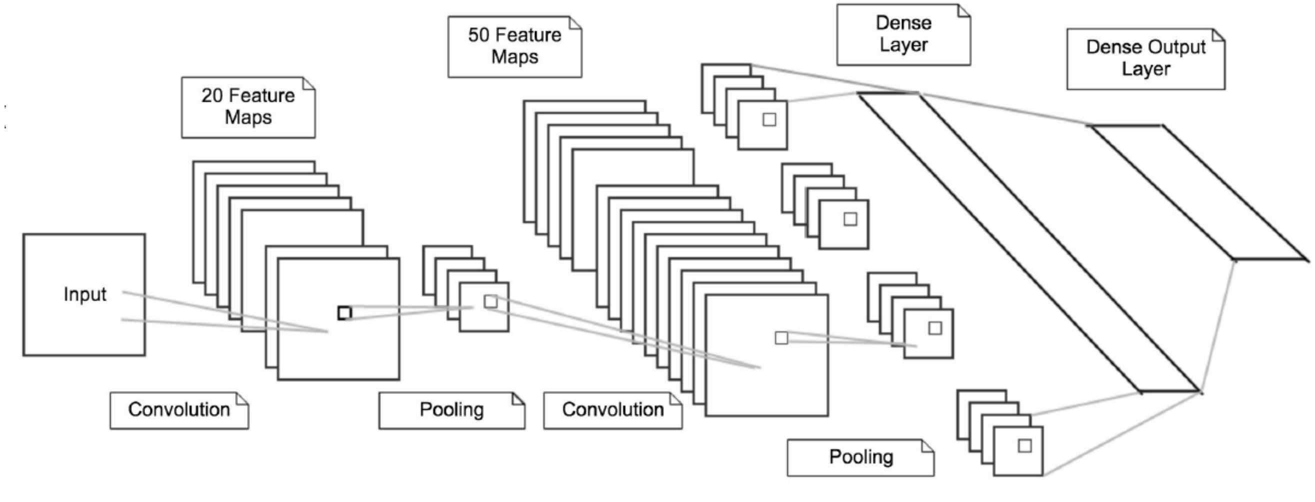
(another indicator that even with cutting edge technology machine learning never achieves 100% performance)

[1] M. Nielsen

Exercises – MNIST Dataset – CNN Model Example

MNIST Dataset – CNN Model

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation, Flatten
from keras.utils import np_utils
from keras import backend as K
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, Adam

# model
class CNN:
  @staticmethod
  def build(input_shape, classes):
    model = Sequential()
    model.add(Convolution2D(20, kernel_size=5, padding="same", input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Convolution2D(50, kernel_size=5, border_mode="same"))
    model.add(Activation("relu"))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation("relu"))
    model.add(Dense(classes))
    model.add(Activation("softmax"
    return model
```



[9] A. Gulli et al.

MNIST Dataset – CNN Python Script

```python
# parameters
NB_CLASSES = 10
NB_EPOCH = 20
BATCH_SIZE = 128
VERBOSE = 1
OPTIMIZER = 'Adam'
VALIDATION_SPLIT = 0.2
IMG_ROWS, IMG_COLS = 28, 28
INPUT_SHAPE = (1, IMG_ROWS, IMG_COLS)

# dataset 28 x 28 pixels
(X_train, y_train), (X_test, y_test) = mnist.load_data()
K.set_image_dim_ordering("th")
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalization
X_train /= 255
X_test /= 255

# input convnet
X_train = X_train[:, np.newaxis, :, :]
X_test = X_test[:, np.newaxis, :, :]

# data output
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

# convert vectors to binary matrices of classes
Y_train = np_utils.to_categorical(y_train, NB_CLASSES)
Y_test = np_utils.to_categorical(y_test, NB_CLASSES)

# Simple CNN model
model = CNN.build(input_shape=INPUT_SHAPE, classes=NB_CLASSES)

# Compilation
model.compile(loss='categorical_crossentropy', optimizer=OPTIMIZER, metrics=['accuracy'])

# Fit the model
history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NB_EPOCH, verbose=VERBOSE, validation_split=VALIDATION_SPLIT)

# evaluation
score = model.evaluate(X_test, Y_test, verbose=VERBOSE)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

- **OPTIMIZER: Adam - advanced optimization technique that includes the concept of a momentum (a certain velocity component) in addition to the acceleration component of Stochastic Gradient Descent (SGD)**
- **Adam computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients**
- **Adam enables faster convergence at the cost of more computation and is currently recommended as the default algorithm to use (or SGD + Nesterov Momentum)**

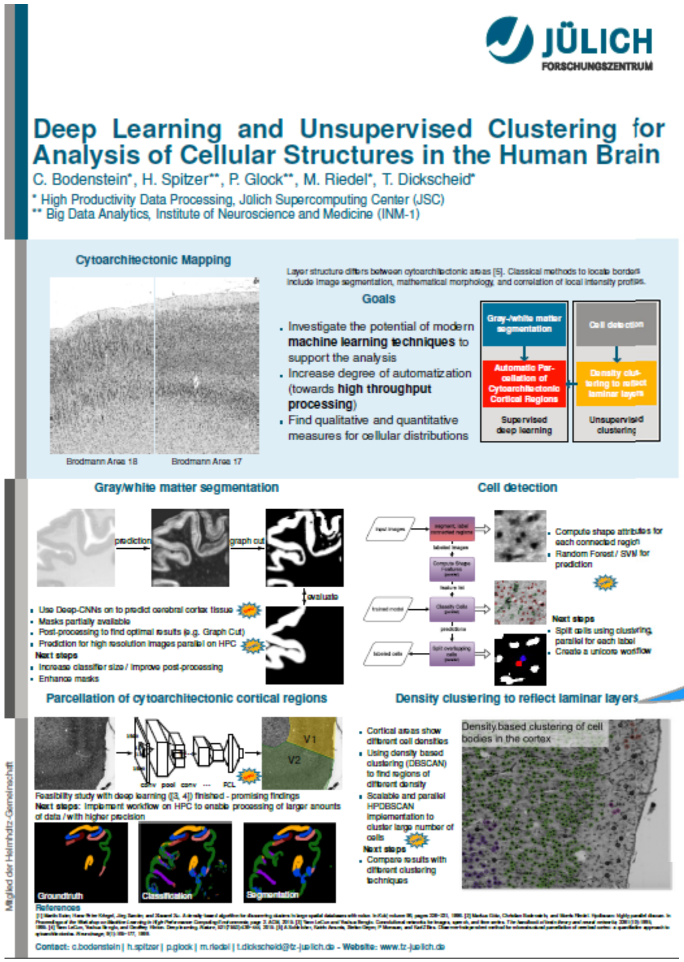
[12] D. Kingma et al., 'Adam: A Method for Stochastic Optimization'

MNIST Dataset – CNN Model – Output

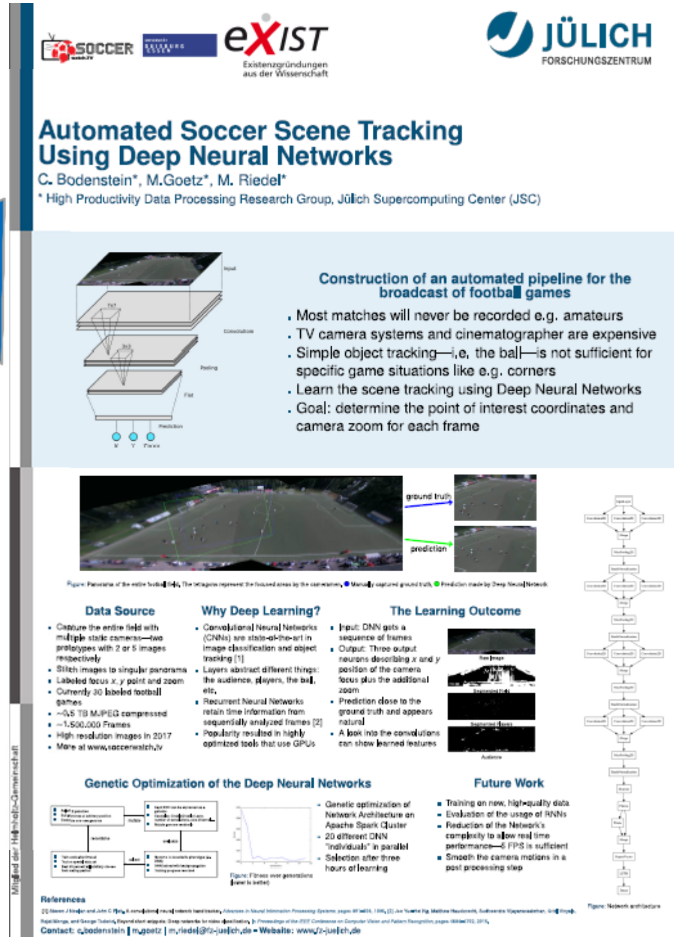
```
[vsc42544@gligar01 deeplearning]$ head KERAS_MNIST_CNN.o1179880
60000 train samples
10000 test samples
Train on 48000 samples, validate on 12000 samples
Epoch 1/20

  128/48000 [..............................] - ETA: 10:06 - loss: 2.2997 - acc: 0.1250
  256/48000 [..............................] - ETA: 7:46 - loss: 2.2578 - acc: 0.1992
  384/48000 [..............................] - ETA: 6:58 - loss: 2.2127 - acc: 0.2083
  512/48000 [..............................] - ETA: 6:35 - loss: 2.1632 - acc: 0.2598
  640/48000 [..............................] - ETA: 6:20 - loss: 2.0934 - acc: 0.3234
```

```
[vsc42544@gligar01 deeplearning]$ tail KERAS_MNIST_CNN.o1179880
 9824/10000 [============================>.] - ETA: 0s
 9856/10000 [============================>.] - ETA: 0s
 9888/10000 [============================>.] - ETA: 0s
 9920/10000 [============================>.] - ETA: 0s
 9952/10000 [============================>.] - ETA: 0s
 9984/10000 [============================>.] - ETA: 0s
10000/10000 [==============================] - 41s 4ms/step
Test score: 0.0483192791523
Test accuracy: 0.99
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_CNN_1179880.master19.golett.gent.vsc
```

Advanced Application Examples & Opportunities

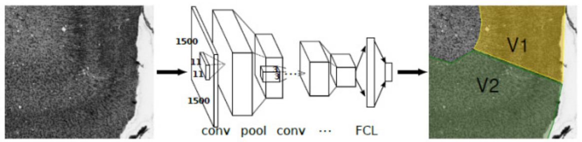


Enabling new Start-ups

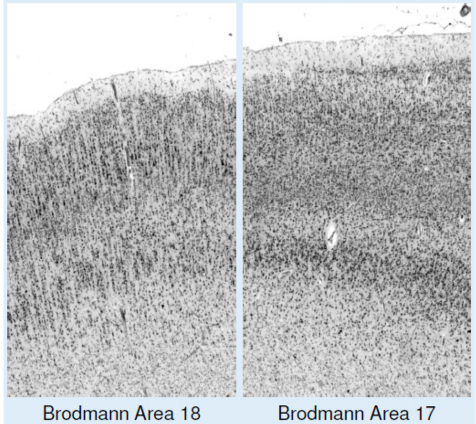


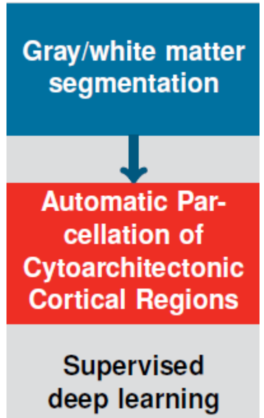
CNN – Neuroscience Application

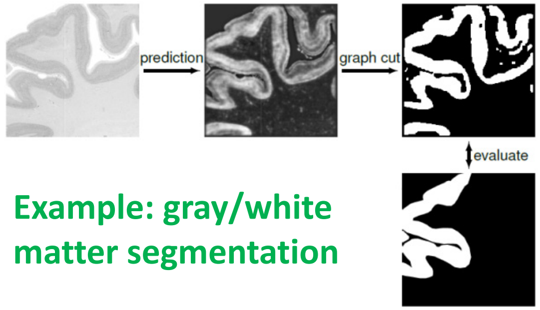
- Goal: Cytoarchitectonic Mapping
  - Layer structure differs between cytoarchitectonic areas
  - Classical methods to locate borders consists of much manual work: e.g. image segmentation, mathematical morphology, etc.
  - Deep Learning: Automate the process of learning 'border features' by providing large quantities of labelled image data
  - However: the structure setup of the deep learning network still requires manual setup (e.g. how many hidden layers, etc.)



Example: Parcellation of cytoarchitectonic cortical regions

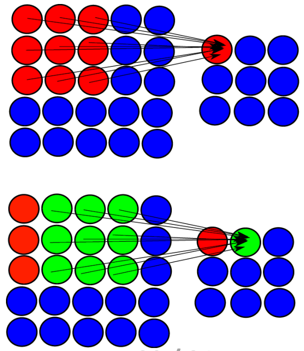






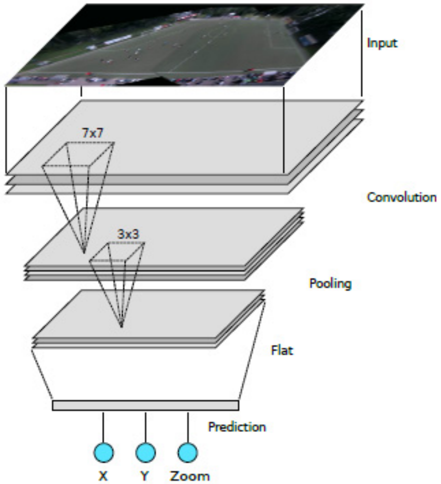
Example: gray/white matter segmentation

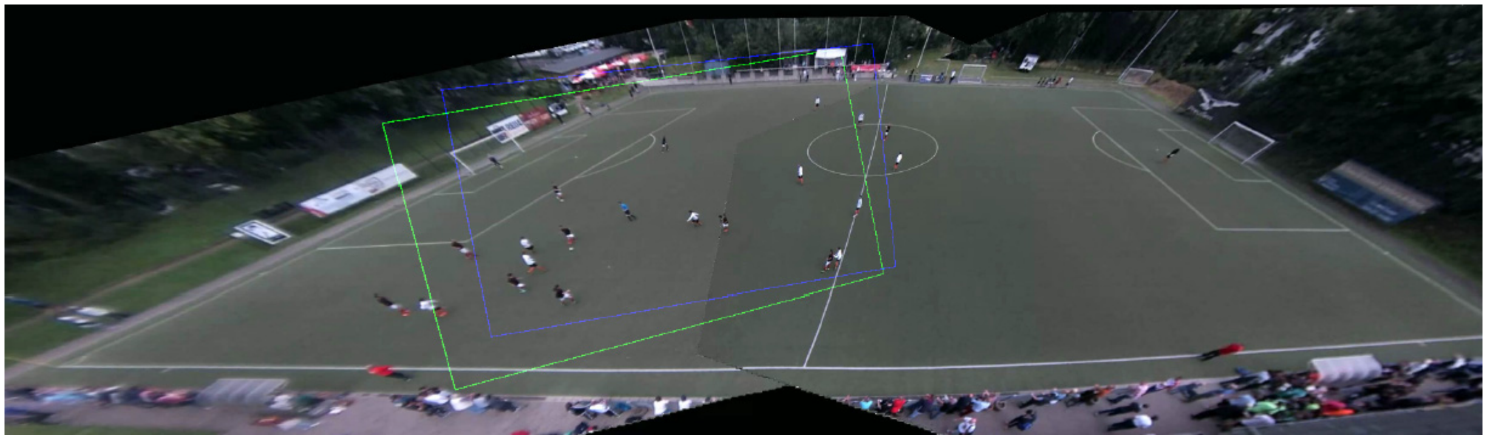
Use Convolution Neural Networks: arbitrary dimension, move 'filter' kernel over input space, take local space into account, much cheaper, less parameters than fully connected (e.g. ANNs)

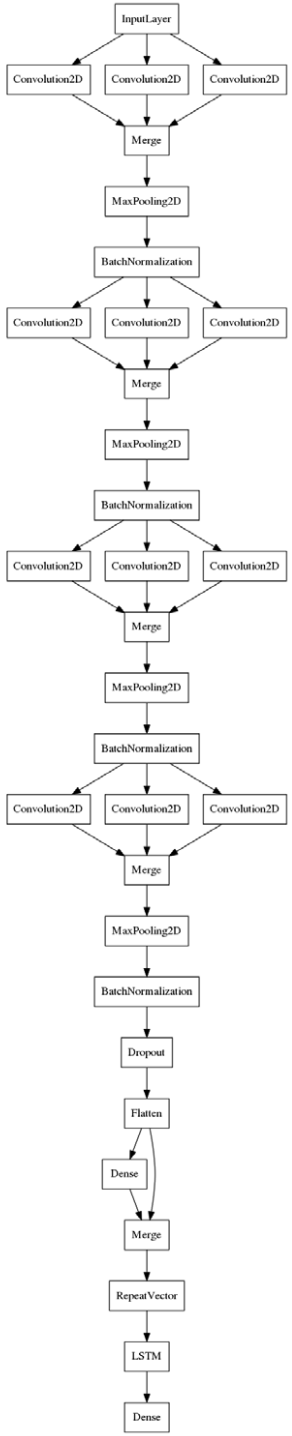


CNN – Soccerwatch.tv Application

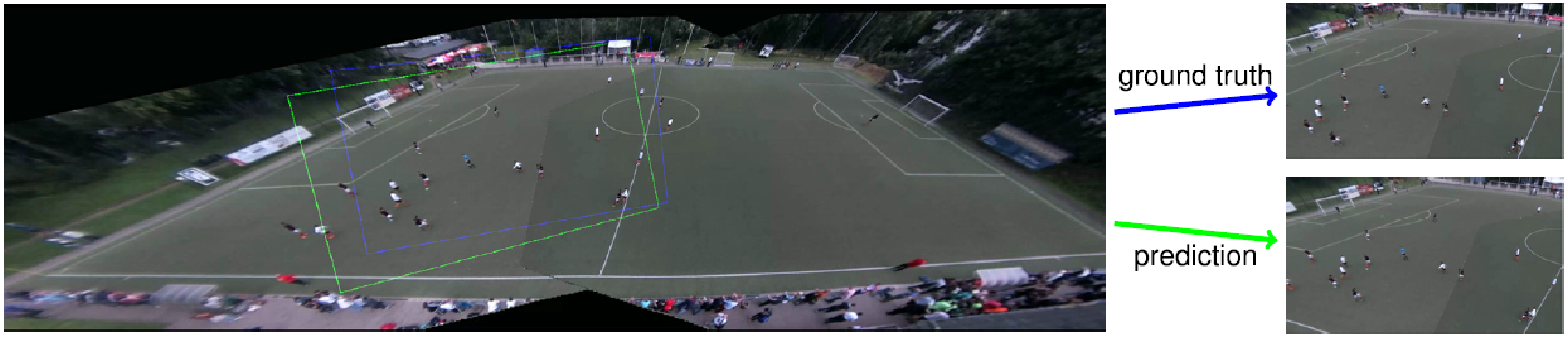
- Goal: Automatic zoom w/o camera man
  - Besides upper leagues:
    80k matches/week
  - Recording too expensive (amateurs)
  - Camera man needed
  - Soccerwatch.tv provides panorama
  - Approach: Find X,Y center and zoom on panorama using Deep Learning

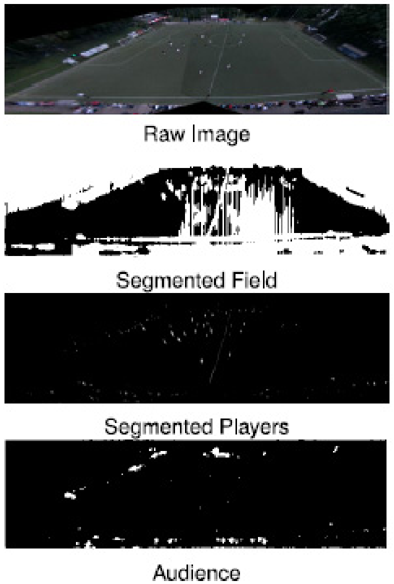






CNN – Soccerwatch.tv Application – Results





(Look into convolutions shows learned features)

UNIVERSITÄT DUISBURG ESSEN

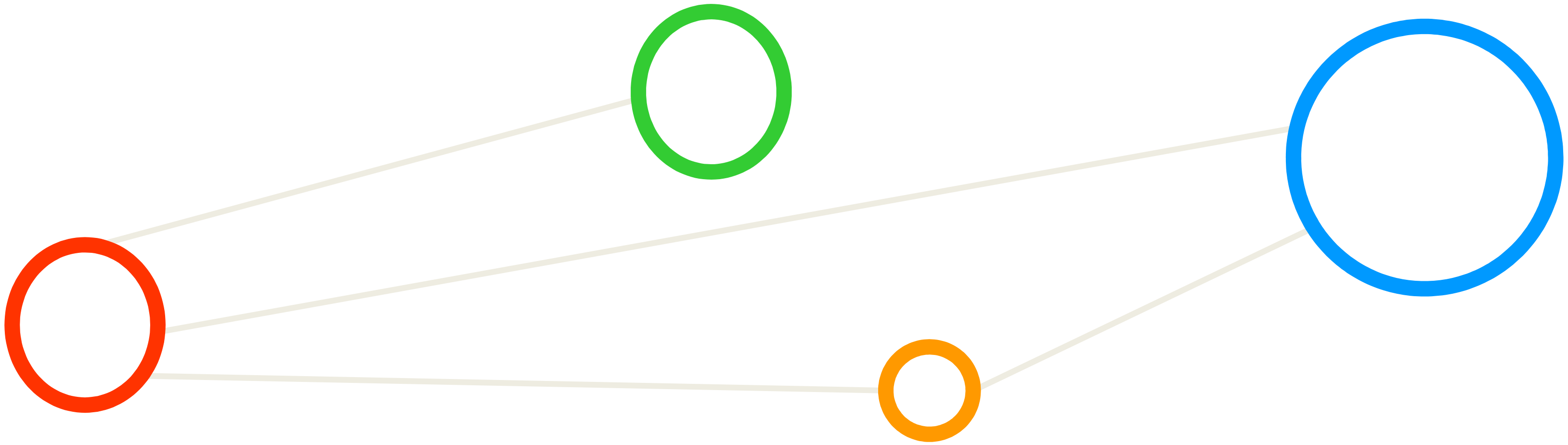
eXIST Existenzgründungen aus der Wissenschaft

[11] Soccerwatch.tv

[Video] CNN Application in Autonomous Driving

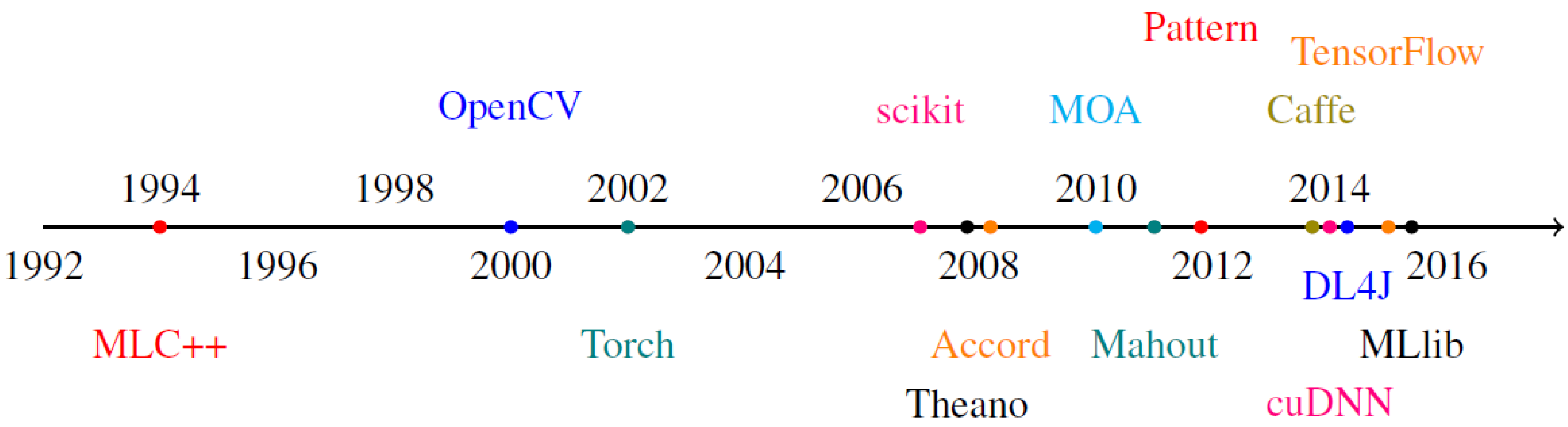
[14] YouTube Video, Speed Sign Recognition

Deep Learning Toolset

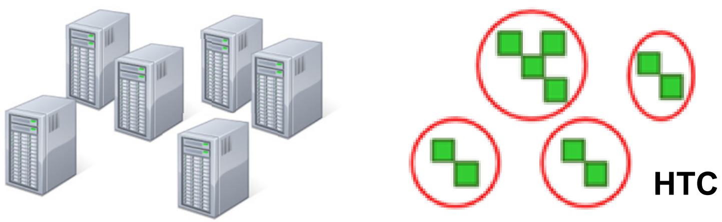


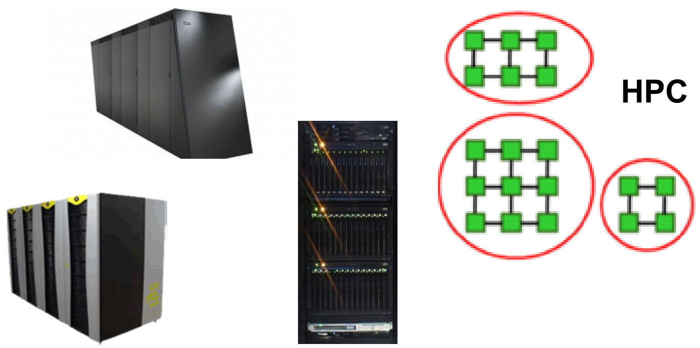
Time Line of Deep Learning and Machine Learning

- Selected Frameworks only



[8] A Tour of Tensorflow





Increasing number of Deep Learning Frameworks

- TensorFlow
  - An open-source software library often used
  - Supported device types are CPU and GPU
- Caffe
  - Deep learning framework made with speed and modularity in mind
  - Switch between CPU and GPU by setting a single flag
  - E.g. train on a GPU machine, then deploy to commodity clusters
- Theano
  - Python library for deep learning with integration of NumPY
  - Transparent use of GPGPUs

[4] Tensorflow

[5] Caffe

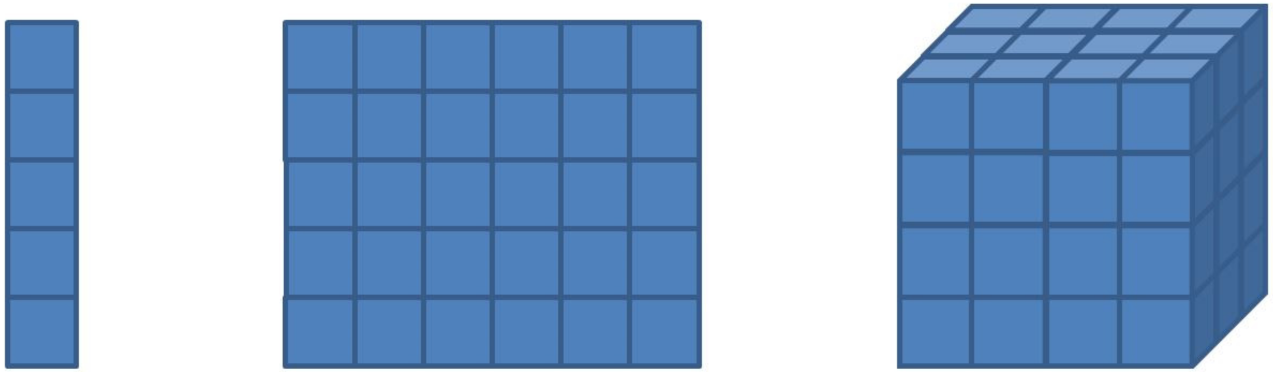
[6] Theano

- **There are a wide variety of deep learning frameworks available that support convolutional neural networks and take advantage of GPGPUs, e.g. TensorFlow, Caffe, Theano**

[7] Deep Learning Framework Comparison

What is a Tensor?

- Meaning
  - Multi-dimensional array used in big data analysis often today
  - Best understood when comparing it with vectors or matrices



(one dimensional tensor) (vector of dimension [5])

(two dimensional tensor) (matrix of dimensions [5,6])

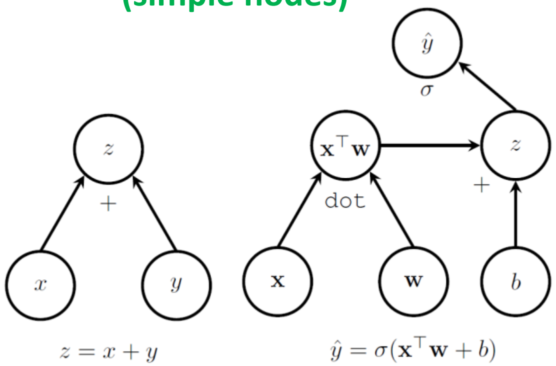
(three dimensional tensor) (tensor of dimension [4,4,3])

[10] Big Data Tips, What is a Tensor?

Tensorflow Computational Graph

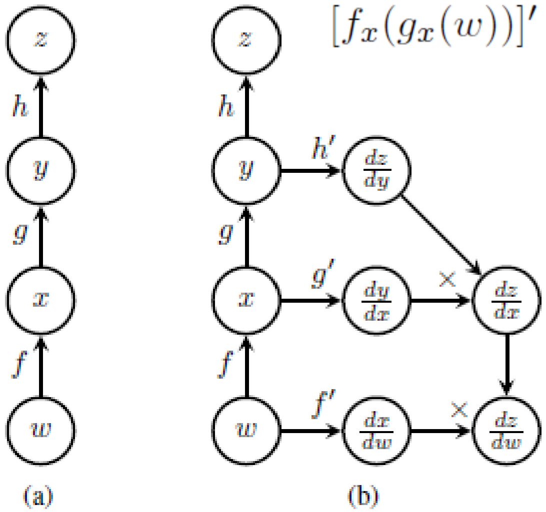
- Keras as a High-Level Framework (on top of Tensorflow)
  - Abstracts from the computational graph and focus on layers
- Machine learning algorithms as computational graph
  - Sometimes also called 'dataflow graph' to emphasize data elements
  - Edges represent data (i.e. often tensors) flowing between nodes
  - Vertices / nodes are operations of various types (i.e. combination or transformation of data flowing through the graph)

(simple nodes)



[8] A Tour of Tensorflow

(adds gradient node for each operation that takes the gradient of the previous link – outer functions – and multiplies with its own gradient)



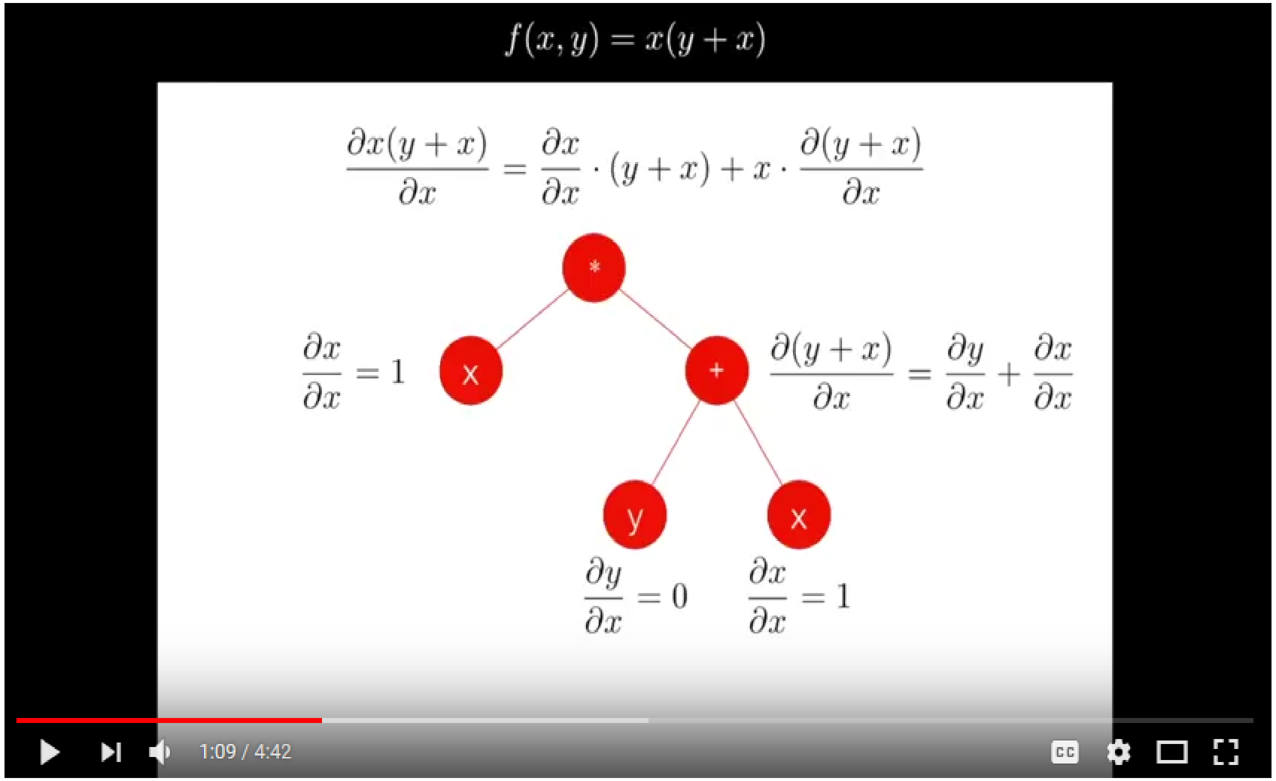
$$[f_x(g_x(w))]' = f'_x(g_x(w)) \cdot g'_x(w)$$

(backpropagation algorithm traverses Tensorflow graph in reverse to compute this chain rule)

[4] Tensorflow

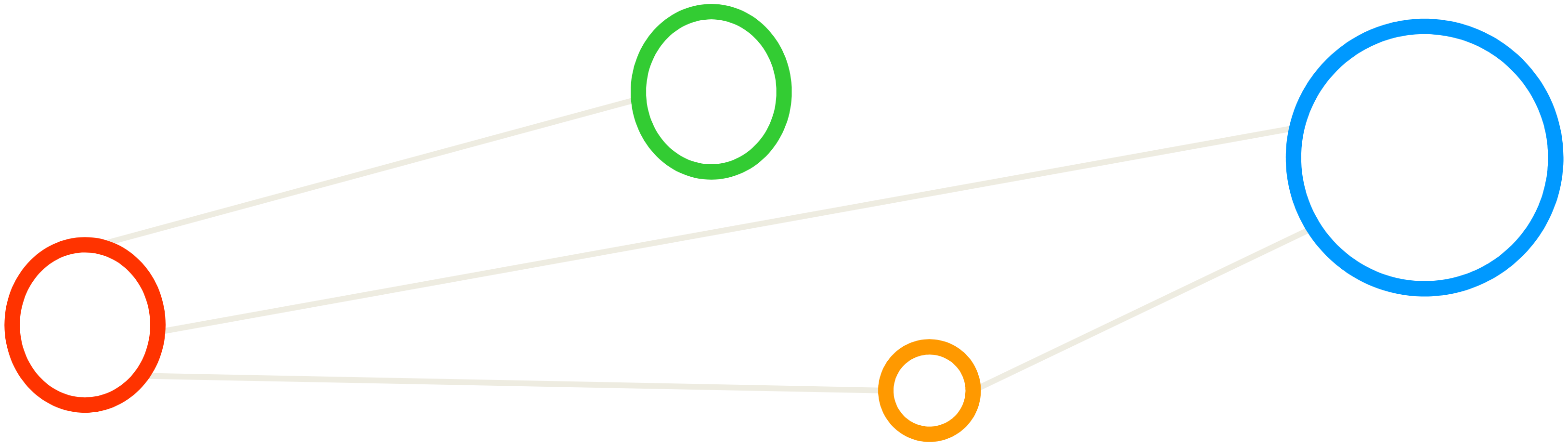
Exercises – MNIST Dataset – CNN Model Check

[Video] Backpropagation in Deep Learning Frameworks



[13] YouTube Video, Chain Rule in Backpropogation

Lecture Bibliography



Lecture Bibliography (1)

- [1] M. Nielsen, 'Neural Networks and Deep Learning',
  Online: http://neuralnetworksanddeeplearning.com/
- [2] Ugent Tier-2 Clusters,
  Online: https://www.vscentrum.be/infrastructure/hardware/hardware-ugent
- [3] A. Rosebrock, 'Get off the deep learning bandwagon and get some perspective', Online:
  http://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/
- [4] Tensorflow,
  Online: https://www.tensorflow.org/
- [5] Cafe Deep Learning Framework,
  Online: http://caffe.berkeleyvision.org/
- [6] Theono Deep Learning Framework,
  Online: http://deeplearning.net/software/theano/
- [7] Deep Learning Framework Comparisons,
  Online: http://neuralnetworksanddeeplearning.com/chap6.html
- [8] A Tour of Tensorflow,
  Online: https://arxiv.org/pdf/1610.01178.pdf
- [9] A. Gulli and S. Pal, 'Deep Learning with Keras' Book, ISBN-13 9781787128422, 318 pages,
  Online: https://www.packtpub.com/big-data-and-business-intelligence/deep-learning-keras
- [10] Big Data Tips, 'What is a Tensor?',
  Online: http://www.big-data.tips/what-is-a-tensor
- [11] Soccerwatch.tv,
  Online: http://www.soccerwatch.tv

Lecture Bibliography (2)

- [12] D. Kingma and Jimmy Ba, ‘Adam: A Method for Stochastic Optimization‘,
  Online: https://arxiv.org/abs/1412.6980
- [13] YouTube Video, ‘Simple explanation of how backpropagation works in deep learning libraries‘,
  Online: https://www.youtube.com/watch?v=zhKWBye_RgE
- [14] YouTube Video, ‘Speed Sign Recognition by Convolutional Neural Networks‘,
  Online: https://www.youtube.com/watch?v=kkha3sPoU70


services
measurement
within
fusion
new
funding
Policy-based
concepts
forms
cross-disciplinary
resources
device
Analysis
Centre
climate
Computer
expertise
year
like
Computational
Test
E.g
Computing
disciplines
results
Approaches
construction
Fusion
Enable
Cross-Disciplinary
EU
methods
High
big
nodes
PB
community
Environment
Modelling
different
Science
centers
Key
scientific
important
storage
basis
Infrastructure
files
etc
DKRZ
Use
model
international
SMAQ
engineering
References
formats
simulations
ENES
technologies
physical
brain
increasing
via
EUDAT
processing
IPCC
Common
TB
Big
Project
Data
MapReduce
create
Methods
computing
Using
European
databases
codes
GB
Supercomputing
Work
images
analysis
DLCL
stored
Resources
analyze
Large
Summary
data-intensive
Service
Scientific
levels
often
use
Level
Research
Matters
Concrete
external
performance
JSC
computational
research
e.g
science
HPC
Juelich
datasets
acquisition
large-scale
system
compute
Landscape
System
Climate
modelling
AR5
Hardware
distributed
cores
access
hundreds
Nutshell
Services
management
Technologies
Earth
Structure
current
network
DLCLs
Understanding
structures
directory
project
General
Germany
Simulation
Processing
Provide
NASA
Energy
systems
day
Health