DEEP Projects

HPC for Big Remote Sensing Data Analytics (Convolutional Neural Networks)

Dr. Gabriele Cavallaro
Postdoctoral Researcher
High Productivity Data Processing Group
Jülich Supercomputing Centre

Ernir Erlingsson
PhD student
University of Iceland

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

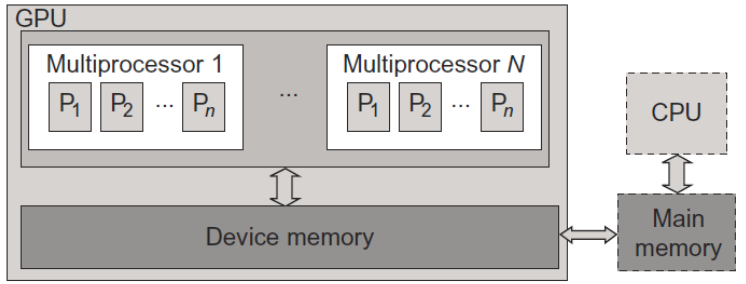
Outline

- HPC Developments
  - Advances in Computation
  - Deep Learning – The Tools of the Trade
- Convolutional Neural Networks
  - Deep Learning Revisited
  - Scaling Deep Learning with HPC
- Practicals

HPC Developments

Advances in Computation

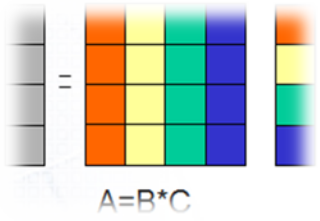
- The graphics Processing Unit (GPU) is repurposed as General-Purpose GPUs (GPGPUs) and used for computing.
- Slower than CPUs but more than makes up for it with sheer volume, i.e. consists of *very many* simple cores
  - High throughput computing-oriented architecture
  - Use massive parallelism by executing a lot of concurrent threads
  - Handle an ever increasing amount of multiple instruction threads
  - CPUs instead typically execute a single long thread as fast as possible
- Simplicity leads to leass power consumption
- Many-core GPUs are already used in large clusters and within massively parallel supercomputers

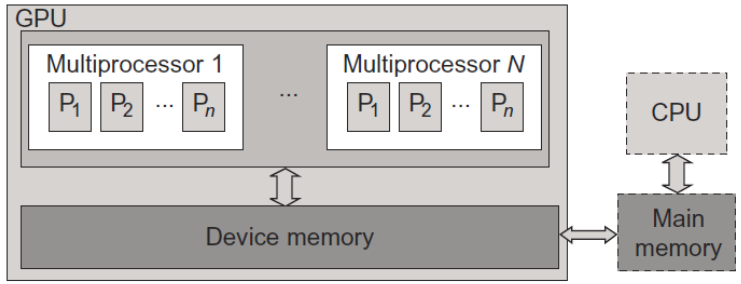


[1] Distributed & Cloud Computing Book

Advances in Computation: GPGPU Acceleration

- GPUs accelerate computing thru massive parallelism, with thousands of threads.
- GPUs are designed to compute a large number of floating point operations in parallel
- GPU accelerator architecture example - NVIDIA card
  - GPUs can have **256 cores** on one single GPU chip (NVIDIA TEGRA X1)
  - Each core can work with **eight threads** of instructions
  - GPU is able to concurrently execute 256 * 8 = **2048 threads**
  - Interaction and thus major (bandwidth) bottleneck between CPU and GPU is via memory interactions
  - E.g. applications that use matrix – vector multiplication
  - (other well known accelerators & many-core processors are e.g. Intel Xeon Phi → run 'CPU' applications easier)





[1] Distributed & Cloud Computing Book

Advances in Computation: FPGA Acceleration

- Recent introduction of hardware advances with Field Programmable Gate Arrays (FPGA)
- Have the potential to outperform GPUs with less energy requirements. Hardware is faster than software.
- Not enough to only have fast hardware
  - Needs easy code development
  - Library support

[2] Field Programmable Gate Array

Deep Learning – Tools of the Trade

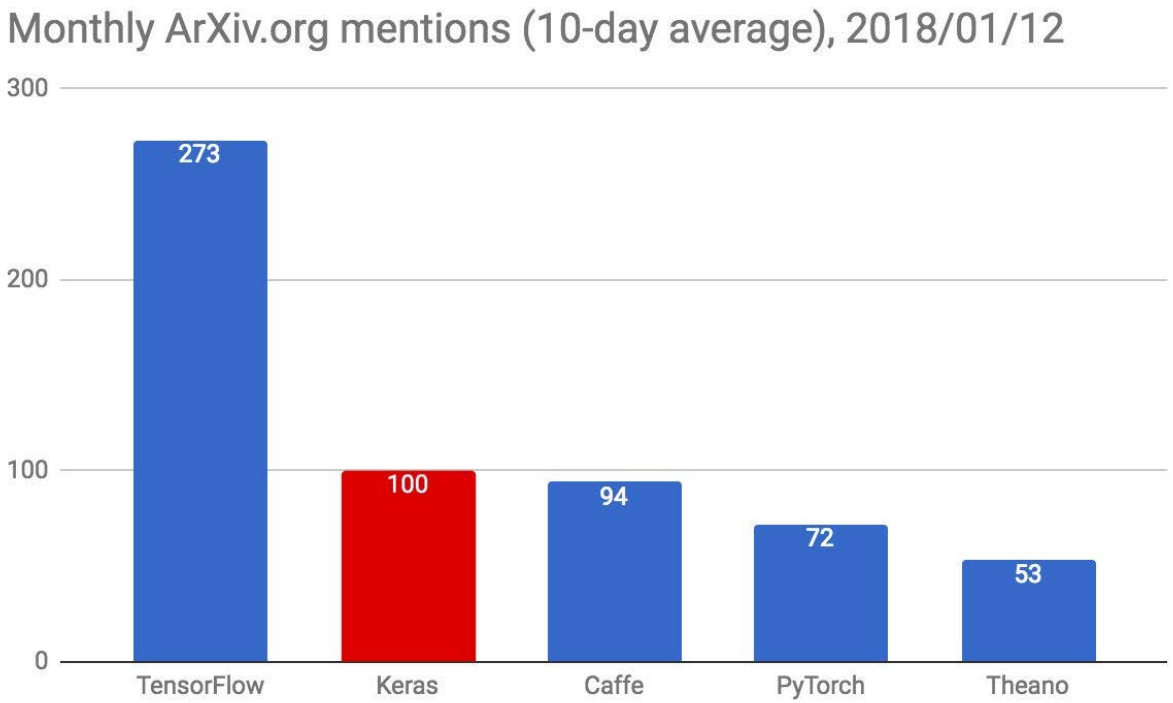
Caffe

theano

Deep Learning – Tools of the Trade

- The top 5 mentions on arXiv.org (01/18), many more exist.
- Free and Open Source (FOSS) frameworks, libraries and extensions.
- Mostly used with Python, a major contributer to its growing popularity.
- Initiated and maintained by a mixture of both academia and industry

Deep Learning – Tools of the Trade

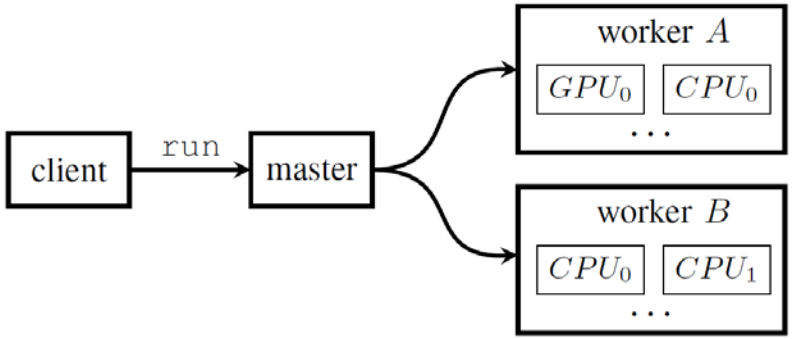


Keras with Tensorflow Backend – GPGPU Support

- Keras is a high-level deep learning library implemented in Python that works on top of existing other rather low-level deep learning frameworks like Tensorflow, CNTK, or Theano
- The key idea behind the Keras tool is to enable faster experimentation with deep networks
- Created deep learning models run seamlessly on CPU and GPU via low-level frameworks

[3] Keras Python Deep Learning Library

[4] Tensorflow Deep Learning Framework



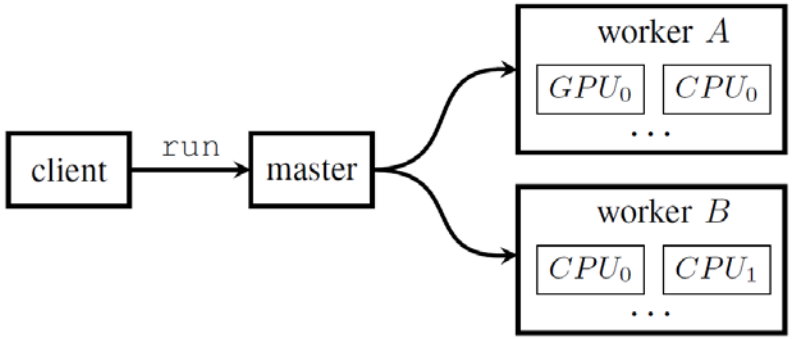
[5] A Tour of Tensorflow

Keras with Tensorflow Backend – GPGPU Support

- Tensorflow is an open source library for deep learning models using a flow graph approach
- Tensorflow nodes model mathematical operations and graph edges between the nodes are so-called tensors (also known as multi-dimensional arrays)
- The Tensorflow tool supports the use of CPUs and GPUs (much more faster than CPUs)
- Tensorflow work with the high-level deep learning tool Keras in order to create models fast

[3] Keras Python Deep Learning Library

[4] Tensorflow Deep Learning Framework

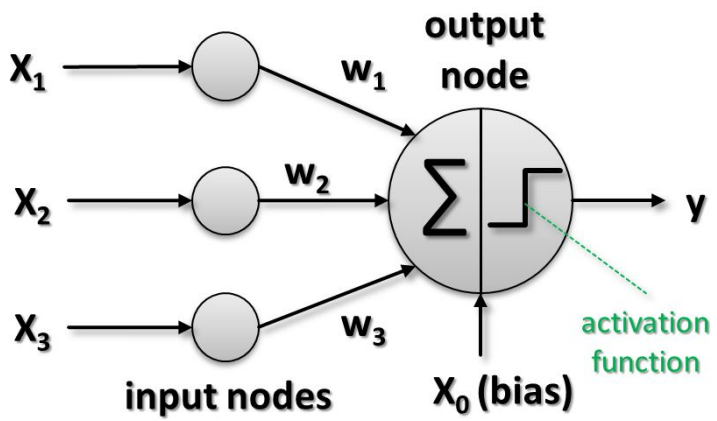


[5] A Tour of Tensorflow

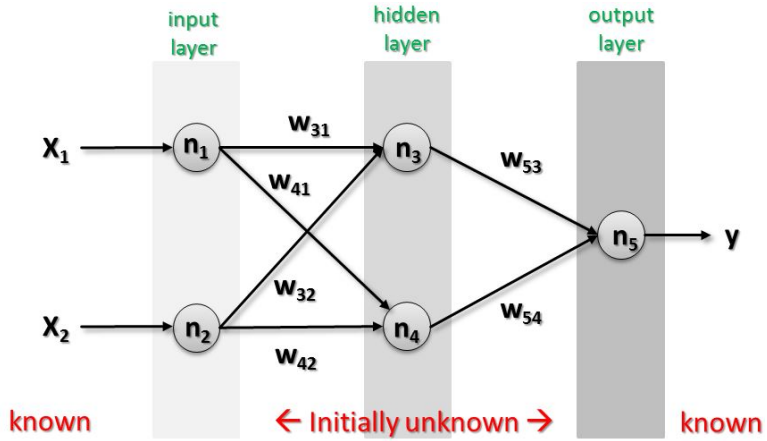
Convolutional Neural Networks

Artificial Neural Network – Feature Engineering & Layers

- Approach: Prepare data before
  - Classical Machine Learning
  - **Feature engineering**
  - Dimensionality reduction techniques
  - Low number of layers (many layers computationally infeasible in the past)
  - Very succesful for speech recognitition (‘state-of-the-art in your phone‘)



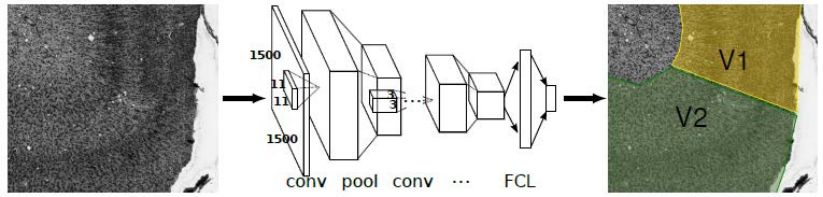
(Perceptron model: designed after human brain neuron)



(Artificial neural network two layer feed – forward)

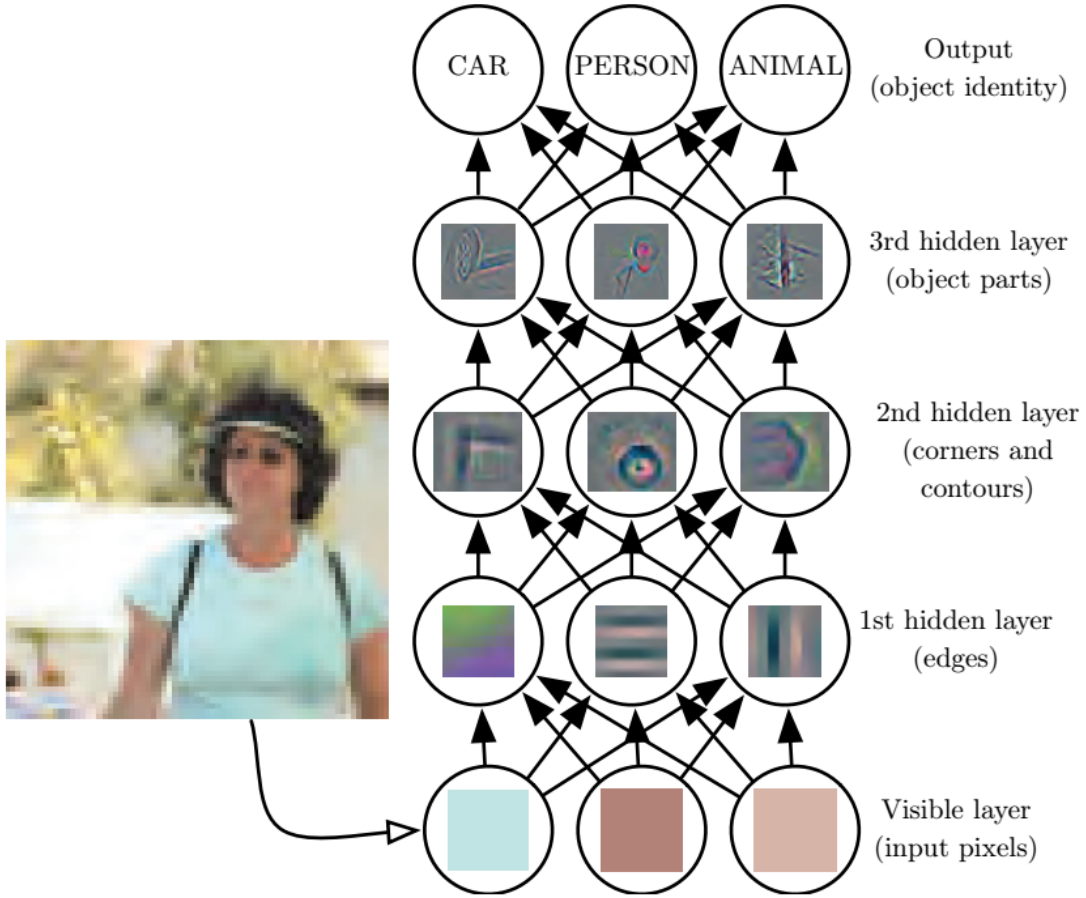
Deep Learning Architectures

- Deep Neural Network (DNN)
  - ‘Shallow ANN‘ approach with many hidden layers between input/output
- Convolutional Neural Network (CNN, sometimes ConvNet)
  - Connectivity pattern between neurons inspired by the visual cortex



- Recurrent Neural Network (RNN)
  - ‘ANN‘ but connections form a directed cycle; state and temporal behaviour
- Deep Reinforcement Learning (DRN)
  - Algorithms driven by positive and negative rewards

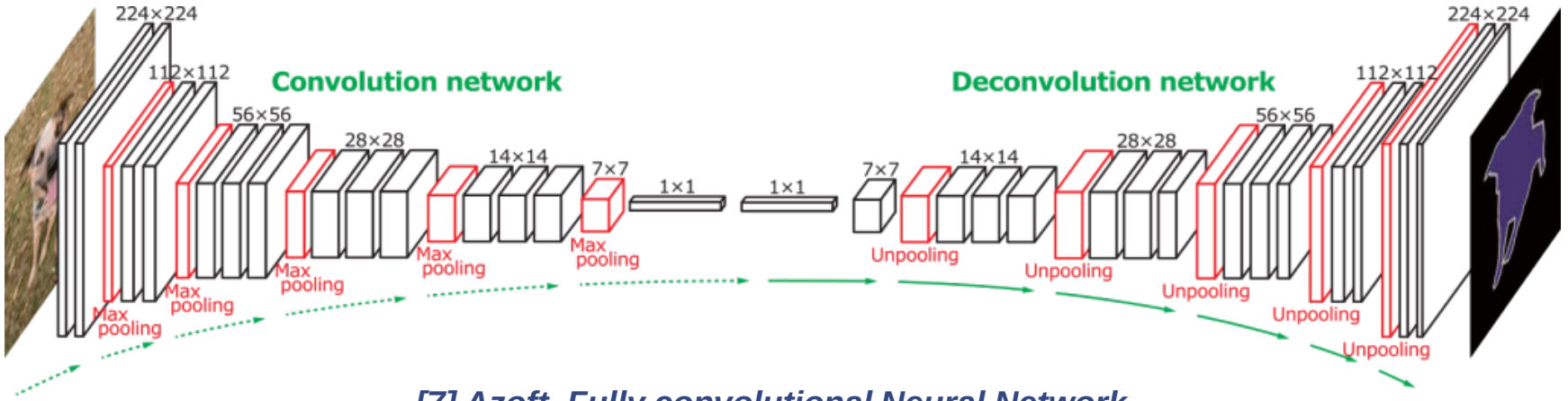
Deep Neural Networks (DNNs)



[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville 'Deep Learning'

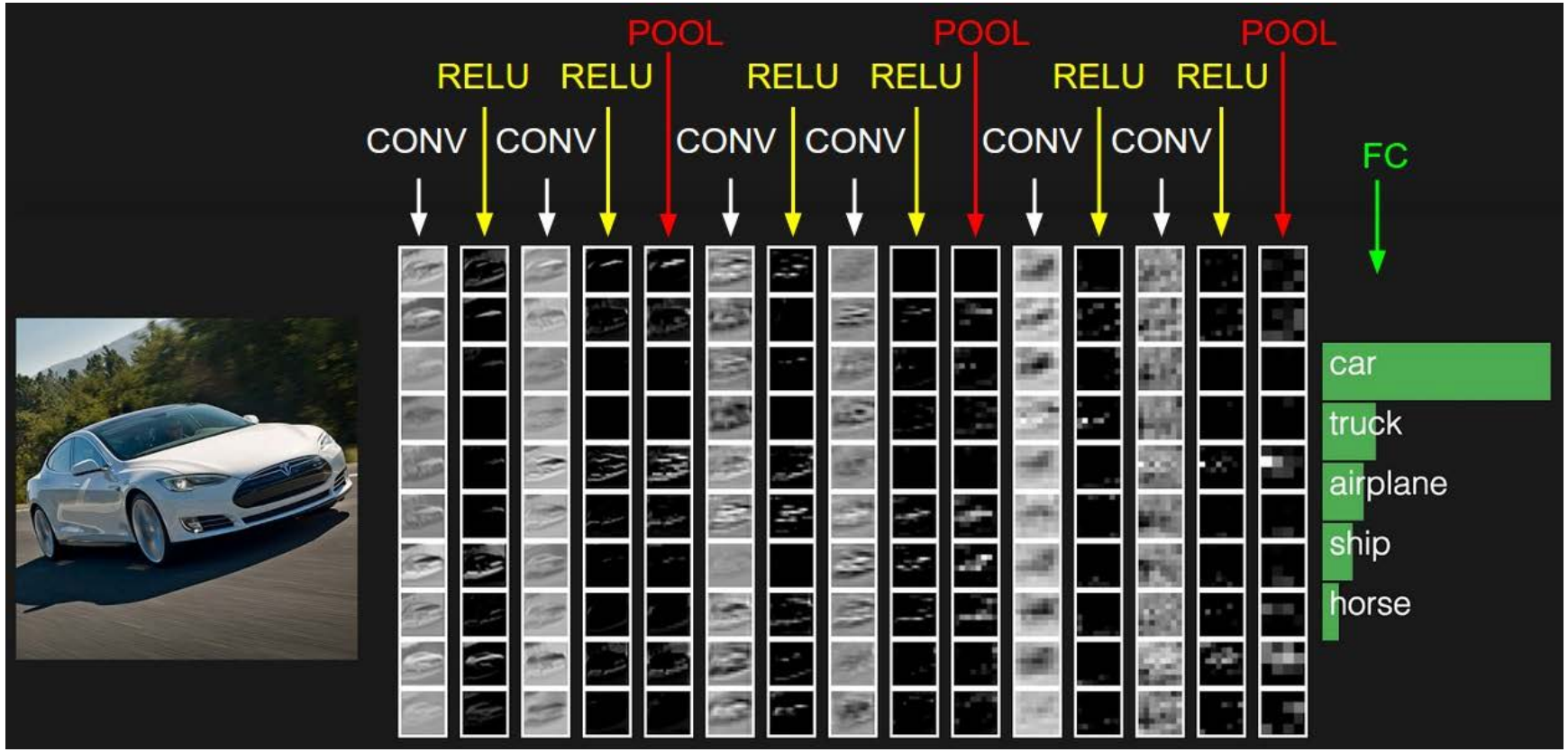
Convolutinal Neural Networks

- Inspired by connectivity patterns between neurons in the animal visual cortex.
- Usually built with three types of layers:
  - Convolution layers using kernels
  - Pooling layers (downsampling)
  - Fully connected layers (classification vote)



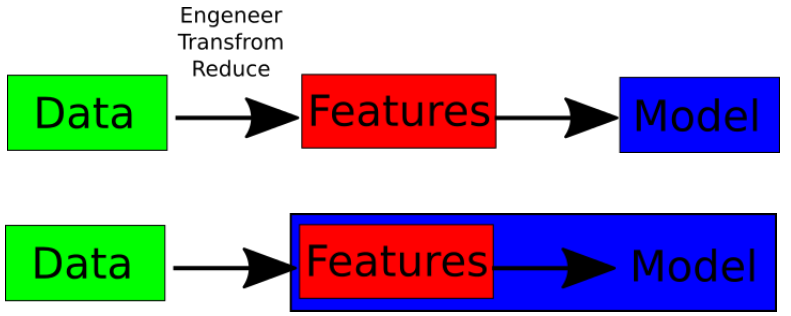
[7] Azoft, Fully convolutional Neural Network

CNNs – Putting it all together



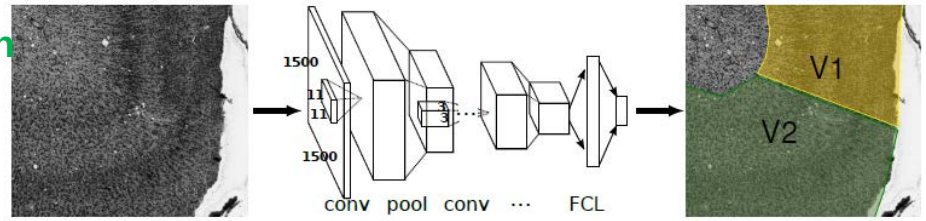
***[8]* Convolutional Neural Networks (CNNs / ConvNets)**

Feature Learning & More Smart Layers



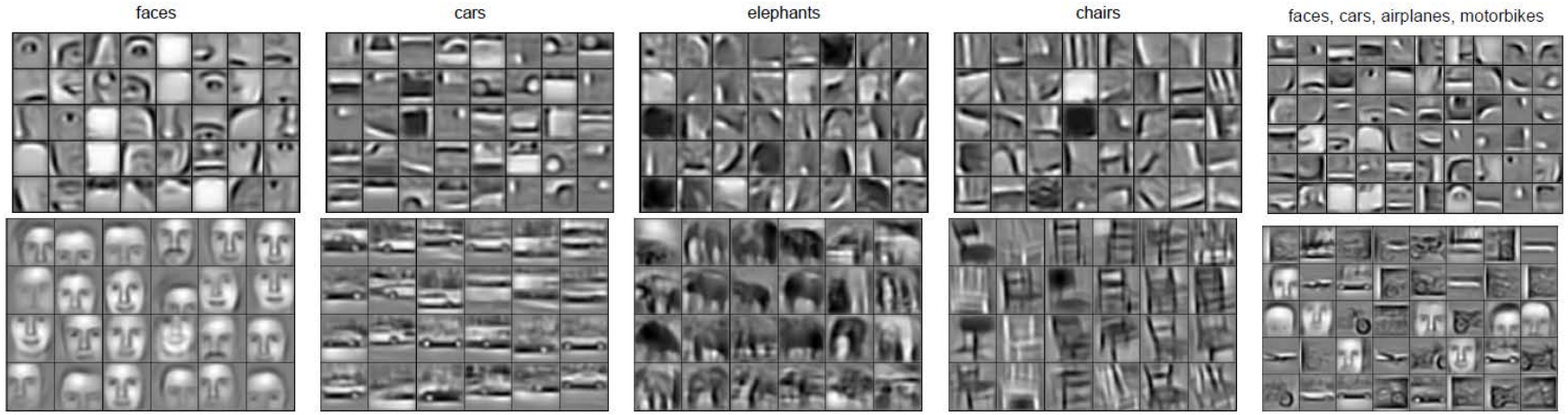
- Approach: Learn Features
  - Classical Machine Learning
  - (**Powerful computing evolved**)
  - Deep (**Feature**) Learning
  - Very succesful for **image recognition** and other emerging areas
  - Assumption: data was generated by the interactions of many different factors on different levels **(i.e. form a hierarchical representation)**
  - Organize factors into multiple levels, corresponding to different levels of abstraction or composition(i.e. **first layers do some kind of filtering**)
  - Challenge: **Different learning architectures**: varying numbers of layers, layer sizes & types used to provide different amounts of abstraction

(Example: Parcellation of cytoarchitectonic cortical regions in the human brain)



Deep Learning – Feature Learning Benefits

- Traditional machine learning applied feature engineering before modeling
- Feature engineering requires expert knowledge, is time-consuming and a often long manual process, requires often 90% of the time in applications, and is sometimes even problem-specific
- Deep Learning enables feature learning promising a massive time advancement

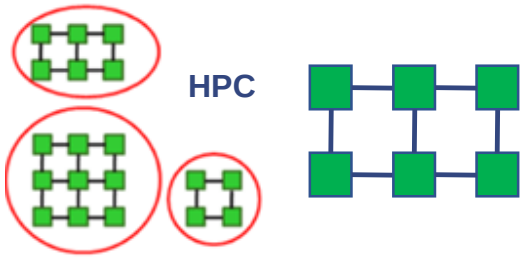


[9] H. Lee et al.

JURECA HPC System at JSC – GPGPUs

- Characteristics
  - Login nodes with 256 GB memory per node
  - 45,216 CPU cores
  - 1.8 (CPU) + 0.44 (GPU) Petaflop/s peak performance
  - Two Intel Xeon E5-2680 v3 Haswell
    CPUs per node: 2 x 12 cores, 2.5 GhZ
  - **75 compute nodes equipped with two NVIDIA K80 GPUs (2 x 4992 CUDA cores)**
- Architecture & Network
  - Based on T-Platforms V-class server architecture
  - Mellanox EDR InfiniBand high-speed
    network with non-blocking fat tree topology
  - 100 GiB per second storage connection to JUST

[10] JURECA HPC System



Deep Learning – Scaling Example on JURECA HPC System

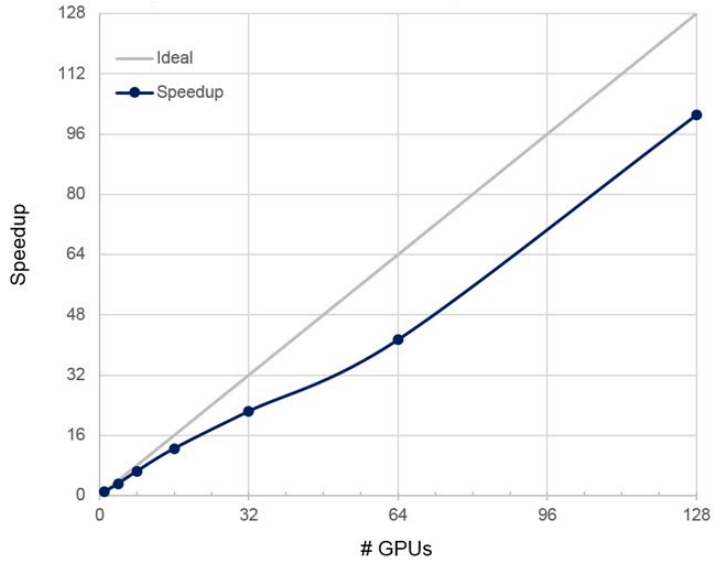
- Open source tool Horovod, which uses MPI, enables distributed deep learning with TensorFlow / Keras
- Machine & Deep Learning: speed-up is just secondary goal after 1st goal accuracy
- Speed-up & parallelization good for faster hyperparameter tuning, training, inference
- Third goal is to avoid much feature engineering through ‘feature learning‘

- Simple Image Benchmark on JURECA
  - 1.2 mil. images with 224 x 224 pixels

[11] A. Sergeev, M. Del Balso,’Horovod’, 2018

#GPUs	images/s	speedup	Performance per GPU [images/s]
1	55	1.0	55
4	178	3.2	44.5
8	357	6.5	44.63
16	689	12.5	43.06
32	1230	22.4	38.44
64	2276	41.4	35.56
128	5562	101.1	43.45

(absolute number of images per second and relative speedup normalized to 1 GPU are given)



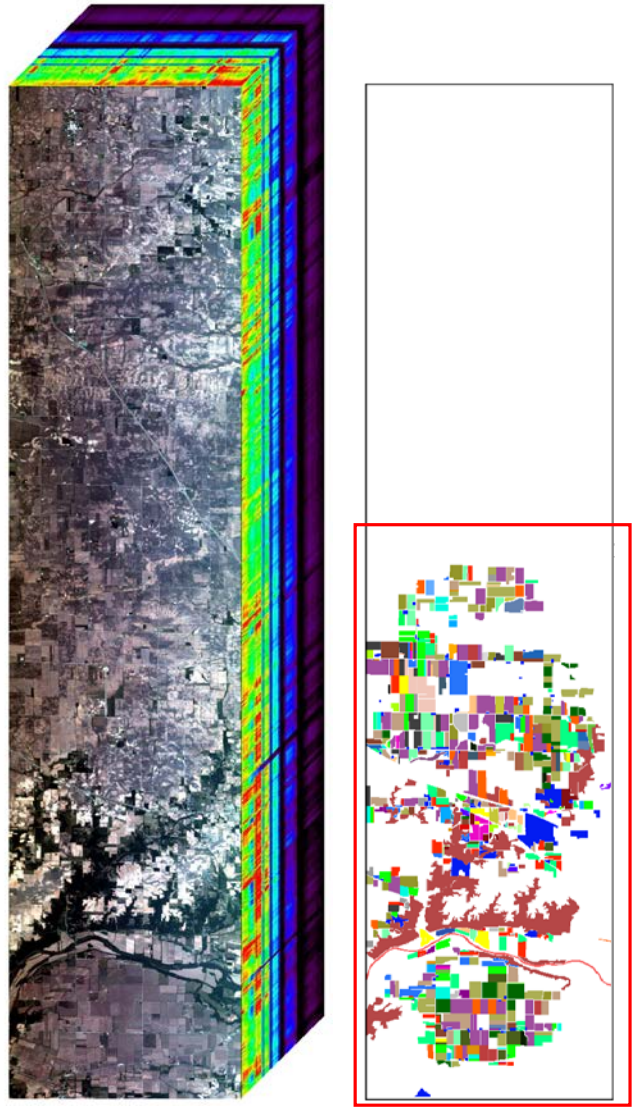
(setup: TensorFlow 1.4, Python 2.7, CUDA 8, cuDNN 6, Horovod 0.11.2, MVAPICH-2.2-GDR)

Practicals

Indian Pines Dataset – Revisited

- Crop of 1417 x 617 pixels (~600MB)
- **No bands and classes** were **discarded**
  - 220 bands and 58 classes

BareSoil (57)	Corn-MinTill (1049)	Hay (1128)	Soybeans-NS (1110)	Soybeans-NaTill (2157)
Buildings (17195)	Corn-MinTill-EW (5629)	Hay (pre.) (2185)	Soybeans-CleanTill (5074)	Soybeans-NaTill-EW (2533)
Concrete/Asphalt (69)	Corn-MinTill-NS (8862)	Hay-Alfalfa (2258)	Soybeans-CleanTill (pre.) (2726)	Soybeans-NaTill-NS (929)
Corn (17783)	Corn-Natill (4381)	Lake (224)	Soybeans-CleanTill-EW (11802)	Soybeans-NaTill-Drilled (8731)
Corn (presumed) (158)	Corn-Natill-EW (1206)	NotCropped (1940)	Soybeans-CleanTill-NS (10387)	Swampy Area (583)
Corn-EW (514)	Corn-Natill-NS (5685)	Oats (1742)	Soybeans-CleanTill-Drilled (2242)	River (3110)
Corn-NS (2356)	Fescue (114)	Oats (pre.) (335)	Soybeans-CleanTill-Weedy (543)	Trees (pre.) (580)
Corn-CleanTill (12404)	Grass (1147)	Orchard (39)	Soybeans-MinTill (15118)	Wheat (4979)
Corn-CleanTill-EW (26486)	Grass/Trees (2331)	Pasture (10386)	Soybeans-Drilled (2667)	Woods (63562)
Corn-CleanTill-NS (39678)	Grass/Pasture-mowed (19)	Pond (102)	Soybeans-MinTill (1832)	Unknown (144)
Corn-CleanTill-NS-Irrigated (800)	Grass/Pasture (73)	Soybeans (9391)	Soybeans-MinTill-Drilled (8098)	
Corn-CleanTill-NS (pre.) (1728)	Grass-runway (37)	Soybeans (pre.) (894)	Soybeans-MinTill-NS (4953)	



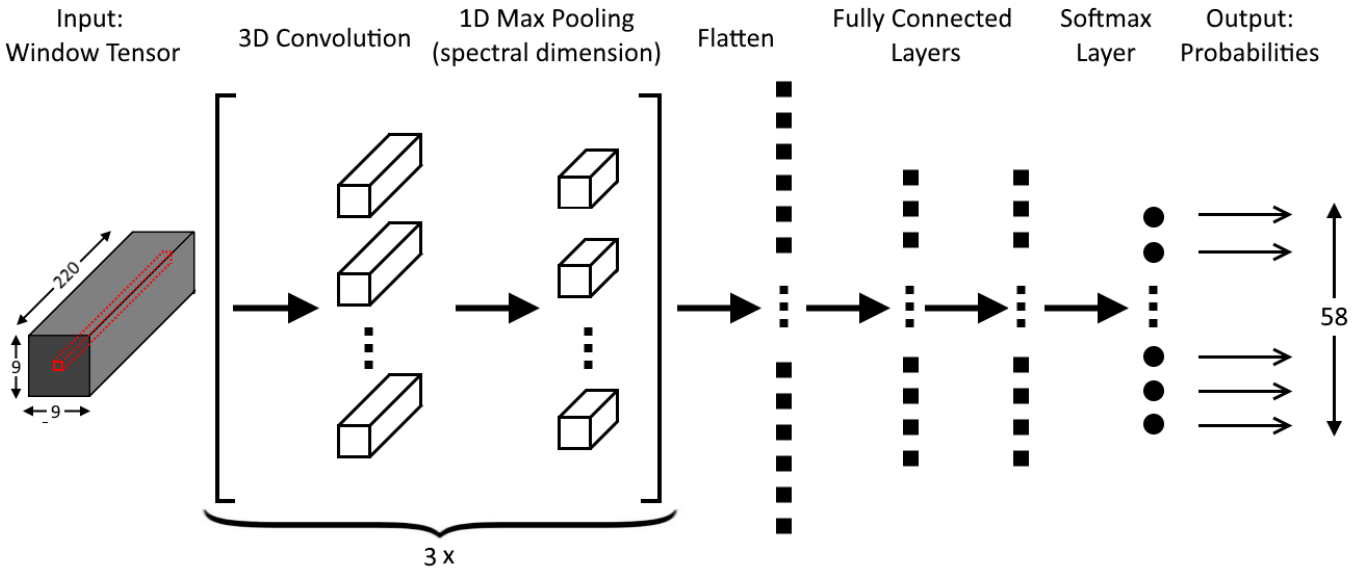
[12] J. Lange, G. Cavallaro, M. Riedel, et al. , IGARSS 2018

Experimental Setup - 3D Convolutional Neural Network

- Designed to perform pixelwise classification of hyperspectral images.
- Input: spatial-spectral tensors of size [w,w,c]
  - w window size, c number of spectral bands
- Exploits spectral information and spatial correlation between neighboring pixels
  - Predict the center pixel

Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	$(3,3,5)$, $(3,3,5)$, $(3,3,5)$
Pooling size	$(1,1,3)$, $(1,1,3)$, $(1,1,2)$
Dense Layer Neurons	128, 128
Activation Functions	rectified linear unit (ReLU)
Loss Function	mean-squared error (MSE)
Optimization	stochastic gradient descent (SGD)
Training Epochs	600
Batch Size	50
Learning Rate	1.0
Learning Rate Decay	5×10^{-6}

(583962 trainable parameters)



[12] J. Lange, G. Cavallaro, M. Riedel, et al. , IGARSS 2018

Python Code Specifications

- Software packages used for the realization of the CNNs
  - Python/3.6.5
  - Keras 2.2.0 on top of the TensorFlow 1.8.0 compute backend
- Source code repository: *https://github.com/Markus-Goetz/cluster-sampling*

Paper Code: TU3.R12.3 **Paper Number:** 2255
Title: THE INFLUENCE OF SAMPLING METHODS ON PIXEL-WISE HYPERSPECTRAL IMAGE CLASSIFICATION WITH 3D CONVOLUTIONAL NEURAL NETWORKS
Topic: Invited Sessions: Deep Learning Methods for Multispectral Image Analysis
Session: TU3.R12: Deep Learning Methods for Multispectral Image Analysis I
Time: Tuesday, July 24, 14:10 - 15:50
Authors: Julius Lange, Gabriele Cavallaro, Markus Götz, Ernir Erlingsson, Morris Riedel

First Step – Navigate to the Folder 3dcnn

- **$ cd ~/igarss_tutorial/3dcnn**
- **$ ls**

```
[train053@jrl06 ~]$ cd ~/igarss_tutorial/3dcnn/
[train053@jrl06 3dcnn]$ ls
indian_pines.hdf5  submit_train_3dcnn.sh
[train053@jrl06 3dcnn]$
```

Second Step – Training and Test Set Generation

- **$ module restore igarss_tutorial**
- **$ python /homea/hpclab/train001/tools/3dcnn/data_generator.py -f 0.1 -s random -c 1 ~/igarss_tutorial/3dcnn/indian_pines.hdf5**

```
[train053@jrl03 3dcnn]$ module restore igarss_tutorial
Restoring modules from user's igarss_tutorial
        Preparing the environment for use of requested stage ( Devel-2018a ).

[train053@jrl03 3dcnn]$ python /homea/hpclab/train001/tools/3dcnn/data_generator.py -f 0
.1 -s random -c 1 ~/igarss_tutorial/3dcnn/indian_pines.hdf5
___random___
        Generating fraction: 0.1, 1/1...
                Reading data source...                  [Done, took: 8.13s]
                Creating selection mask...              [Done, took: 0.06s]
                Calculate normalization coefficients... [Done, took: 0.11s]
                Saving data file ...                    [Done, took: 0.04s]


[train053@jrl03 3dcnn]$ ls
indian_pines.hdf5  indian_pines_random_0.1_0.hdf5  submit_train_3dcnn.sh
```

-f [0.1, 0.3, 0.6, 0.9] : train set fraction
-s [random, size, stddev] : sampling mode
-c : number of data sets to be generated

Third Step: Submit Job Script for Training

```
#!/bin/bash -x

#SBATCH--job-name=train_3dcnn
#SBATCH--output=train_3dcnn_out.%j
#SBATCH--error=train_3dcnn_err.%j
#SBATCH--mail-user=your_email
#SBATCH--mail-type=ALL

#SBATCH--partition=gpus
#SBATCH--gres=gpu:1
#SBATCH--time=06:00:00
#SBATCH--nodes=1
#SBATCH--ntasks=1
#SBATCH--reservation=deep_learning

### load modules
module restore igarss_gpu

### submit
python /homea/hpclab/train001/tools/3dcnn/spectral_cnn.py -g -t -s 0 -w 9 -b 50 -e 400
--model "indian_pines_random_0.1_0_model.h5" --train-history "indian_pines_random_0.1_0_train.csv“
--test-history "indian_pines_random_0.1_0_test.csv" --results "indian_pines_random_0.1_0_results.csv"
indian_pines.hdf5 indian_pines_random_0.1_0.hdf5
```

(ReservationName=igarss-gpu StartTime=2018-07-22T12:45:00 EndTime=2018-07-22T18:15:00)

Outputs

```
[train053@jrl04 3dcnn]$ ls
indian_pines.hdf5                indian_pines_random_0.1_0_model.h5    indian_pines_random_0.1_0_test.csv   submit_train_3dcnn.sh    train_3dnn_err.5813833
indian_pines_random_0.1_0.hdf5   indian_pines_random_0.1_0_results.csv  indian_pines_random_0.1_0_train.csv  train_3dcnn_out.5813833
```

$ vi Indian_pines_random_0.1_0_results.csv

```
OA: 0.8338447116391475
AA: nan
kappa: 0.8200419767971114
F1: 0.5385082040051928
Accuracy for each class:
[ nan 0.   0.84 0.   0.83 0.   0.   0.47 0.84 0.8  0.93 0.7  0.25 0.
 0.65 0.84 0.86 0.   0.76 0.   0.53 0.84 0.   0.   0.   0.28 0.76 0.89
 0.94 0.57 0.62 0.02 0.   0.79 0.   0.63 0.   0.55 0.71 0.81 0.9  0.77
 0.66 0.76 0.86 0.72 0.28 0.81 0.95 0.85 0.78 0.   0.9  0.86 0.99 0.91
 0.89 0.98 0.  ]
F1 score for each class:
[0.   0.   0.82 0.   0.78 0.   0.   0.58 0.76 0.85 0.87 0.8  0.33 0.
 0.7  0.8  0.78 0.   0.78 0.   0.55 0.9  0.   0.   0.   0.41 0.83 0.82
 0.71 0.67 0.63 0.03 0.   0.84 0.   0.71 0.   0.41 0.74 0.81 0.85 0.81
 0.71 0.64 0.85 0.8  0.4  0.86 0.88 0.8  0.81 0.   0.85 0.77 0.99 0.91
 0.92 0.96 0.  ]
Confusion matrix:
[[0.00e+00 0.00e+00 3.00e+01 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  0.00e+00 1.23e+02 6.00e+00 0.00e+00 0.00e+00 0.00e+00 1.00e+00 0.00e+00
  1.00e+00 8.00e+00 0.00e+00 0.00e+00 0.00e+00 3.00e+00 0.00e+00 0.00e+00
  0.00e+00 1.00e+01 0.00e+00 0.00e+00 0.00e+00 5.30e+01 8.00e+00 1.00e+00
  0.00e+00 0.00e+00 0.00e+00 3.00e+00 1.10e+01 0.00e+00 2.00e+00 0.00e+00
  0.00e+00 0.00e+00 6.00e+00 0.00e+00 3.00e+00 1.00e+00 0.00e+00 0.00e+00
  0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
  0.00e+00 0.00e+00 0.00e+00]
```

Bibliography

- [1] K. Hwang, G. C. Fox, J. J. Dongarra, ‘Distributed and Cloud Computing’, Book, Online: http://store.elsevier.com/product.jsp?locale=en_EU&isbn=9780128002049
- [2] Field Programmable Gate Arrays Online: https://en.wikipedia.org/wiki/Field-programmable_gate_array
- [3] Keras Python Deep Learning Library
- [4] Tensorflow Deep Learning Framework, Online: https://www.tensorflow.org/
- [5] A Tour of Tensorflow, Online: https://arxiv.org/pdf/1610.01178.pdf
- [6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, ‘Deep learning’, The MIT Press, 2016
- [7] Fully convolutional neural networks’, Online: https://www.azoft.com/blog/fully-convolutional-neural-networks/
- [8] Convolutional Neural Networks (CNNs / ConvNets) Online: http://cs231n.github.io/convolutional-networks/
- [9] H. Lee et al., ‘Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations’, Proceedings of the 26th annual International Conference on Machine Learning (ICML), ACM, 2009
- [10] JURECA HPC System at JSC Online: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html
- [11] A. Sergeev, M. Del Balso,‘Horovod: fast and easy distributed deep learning in TensorFlow’, 2018 Online: https://arxiv.org/abs/1802.05799
- [12] J. Lange, G. Cavallaro, M. G¨otz, E. Erlingsson, and M. Riedel, “The Influence of Sampling Methods on Pixel-Wise Hyperspectral Image Classification with 3D Convolutional Neural Networks,” in, 2018. *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*

DEEP Projects

The DEEP projects have received funding from the European Union's Seventh Framework Programme (FP7) for research, technological development and demonstration and the Horion2020 (H2020) funding framework under grant agreement no. FP7-ICT-287530 (DEEP), FP7-ICT-610476 (DEEP-ER) and H2020-FETHPC-754304 (DEEP-EST).