


DEEP LEARNING WITH PYTHON

PROF. DR. – ING. MORRIS RIEDEL, UNIVERSITY OF ICELAND / JUELICH SUPERCOMPUTING CENTRE
HEAD OF CROSS-SECTIONAL TEAM DEEP LEARNING & RDA CO-CHAIR INTEREST GROUP BIG DATA
18TH APRIL – 8TH JLESC WORKSHOP, BARCELONA, SPAIN

Joint Laboratory for Extreme-Scale Computing

HELMHOLTZ
RESEARCH FOR GRAND CHALLENGES

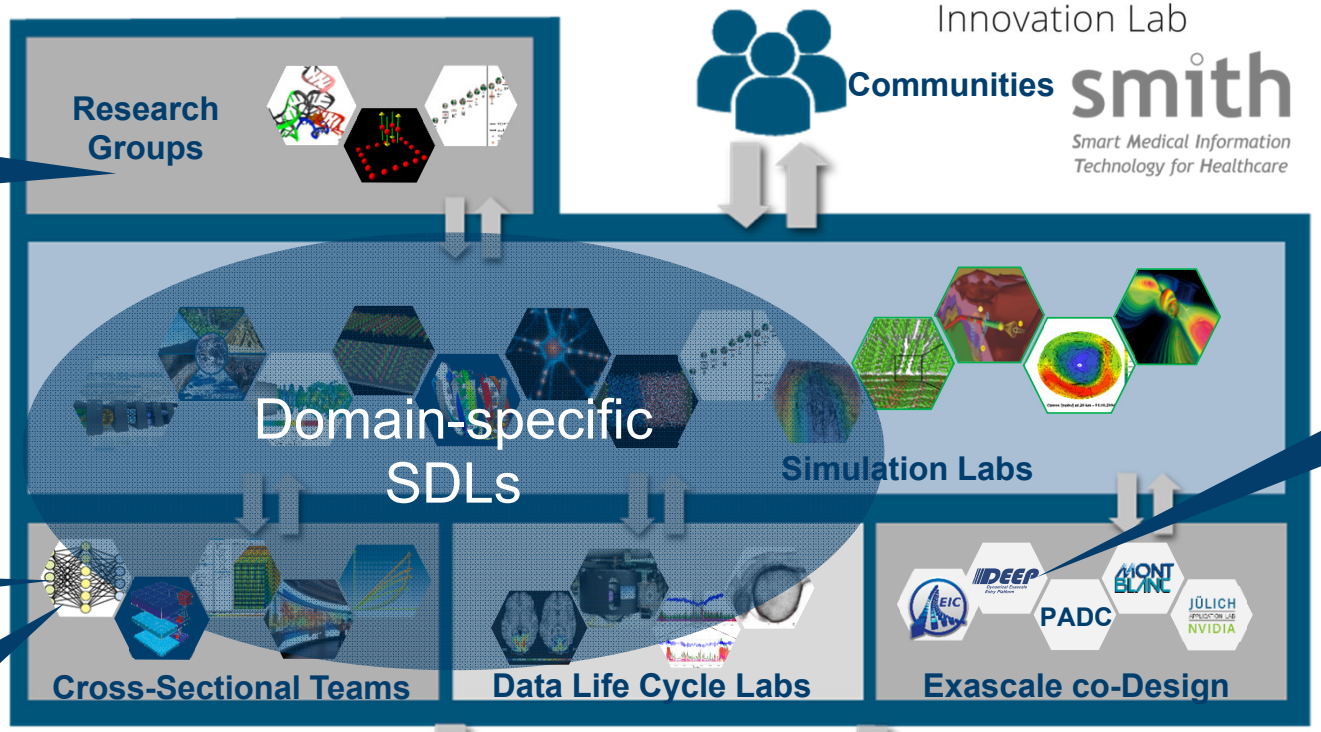
DEEP Projects

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

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

JUELICH SUPERCOMPUTING CENTRE & DEEP LEARNING

Convergence: Simulation and Data Labs (SDL)



Research Group High Productivity Data Processing

Smart Data Innovation Lab

SOCCERWATCH BETA

smith
Smart Medical Information Technology for Healthcare

Landsvirkjun
National Power Company of Iceland

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

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

DEEP-EST EU PROJECT

Cross-Sectional Team Deep Learning

DEEP Projects

Increasing number of Deep Learning Applications in HPC Computing Time Grants

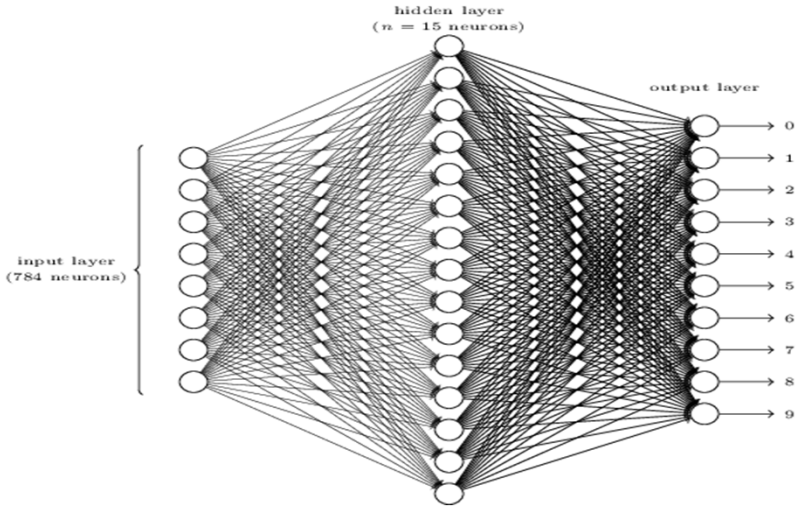
HPC Systems JURECA & JUQUEEN

Modular Supercomputer JUWELS

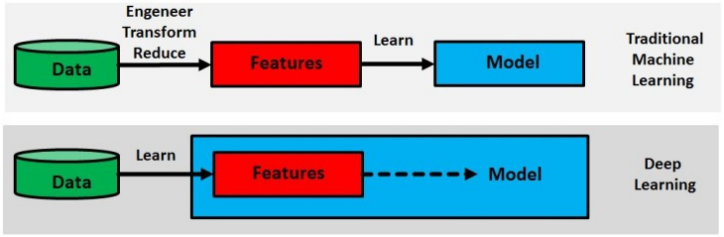
DEEP LEARNING 101

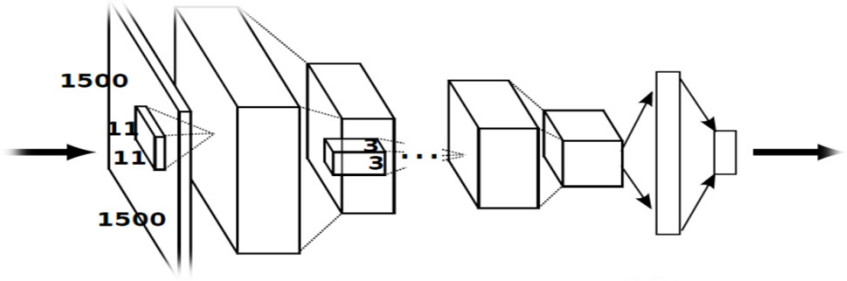
Short Overview & Role of Cross-Sectional Team Deep Learning at Juelich Supercomputing Centre (JSC)

- Innovative & disruptive approach



[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University





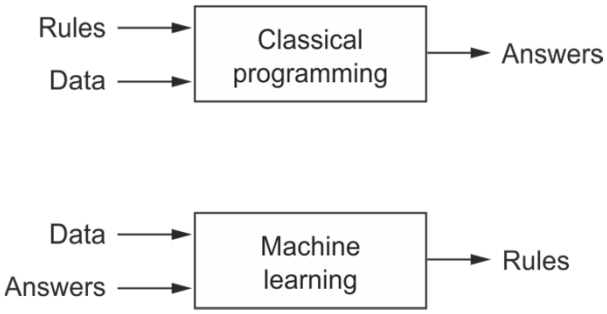
- **Provide deep learning tools that work with JSC HPC machines (e.g. Python/Keras/Tensorflow)**
- **Advance deep learning applications and research on HPC prototypes (e.g. DEEP-EST)**
- **Engage with industry (industrial relations team) & support SMEs (e.g. Soccerwatch)**
- **Offer tutorials & application enabling support for commercial & scientific users (e.g. YouTube)**

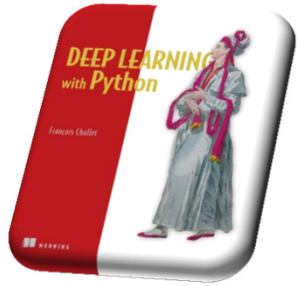
PYTHON

SIMPLE AND FLEXIBLE PROGRAMMING LANGUAGE

[2] Webpage Python

- Selected Benefits
  - Work with many students reveal: qucky & easy to learn
  - Is an interpreted powerful programming language
  - Has Efficient high-level data structures
  - Provides a simple but effective approach to object-oriented programming
  - Great libraries & community support (e.g. numpy)





[3] F. Chollet, 'Deep Learning with Python' Book

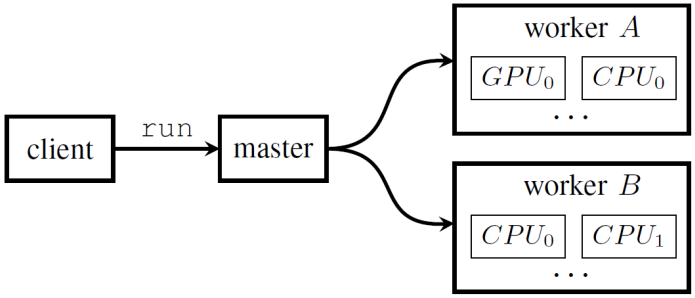
(this talk is not about the book)

- **Python is an ideal language for fast scripting and rapid application development that in turn makes it interesting for the machine learning modeling process**
- **The machine learning modeling process in general and the deep learning modeling process in particular requires iterative and highly flexible approaches**
- **E.g. network topology prototyping, hyper-parameter tuning, etc.**

DEEP LEARNING

Programming with TensorFlow

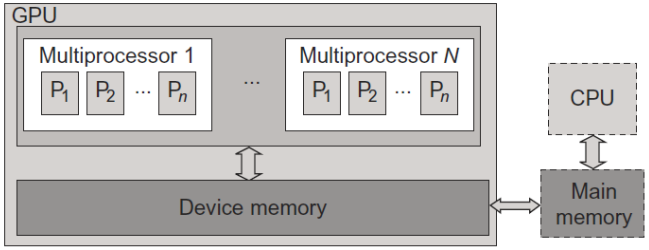
- **Tensorflow is an open source library for deep learning models using a flow graph approach**
- **Tensorflow nodes model mathematical operations and graph edges between the nodes are so-called tensors (also known as multi-dimensional arrays)**
- **The Tensorflow tool supports the use of CPUs and GPUs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



[5] A Tour of Tensorflow

[4] Tensorflow Deep Learning Framework

(Lessons learned: Installing TensorFlow with right versions of CUDA, Python, and other dependencies on a specific system is NOT a trivial task)



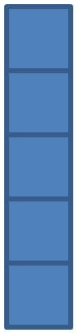
[6] Distributed & Cloud Computing Book

[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

DEEP LEARNING

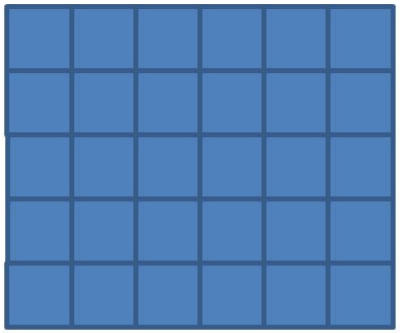
Programming with TensorFlow – What is a Tensor?

- A Tensor is nothing else than a multi-dimensional array often used in scientific & engineering environments
- Tensors are best understood when comparing it with vectors or matrices and their dimensions
- Those tensors 'flow' through the deep learning network during the optimization / learning & inference process



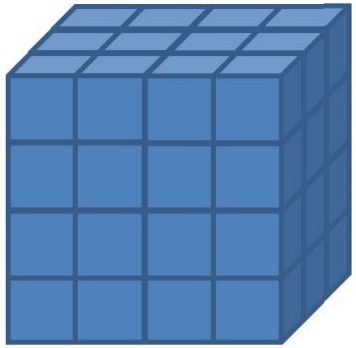
(one dimensional tensor)

(vector of dimension [5])



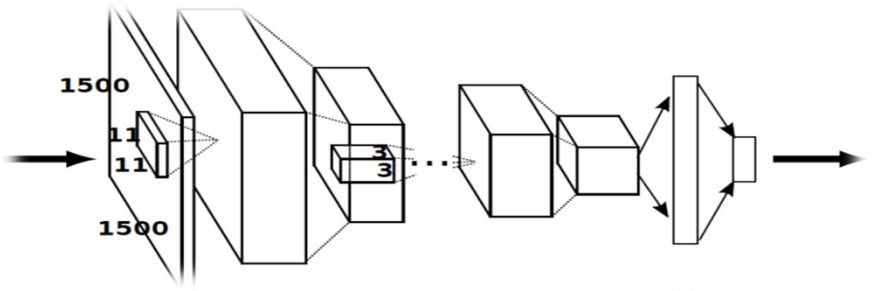
(two dimensional tensor)

(matrix of dimensions [5,6])



(three dimensional tensor)

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



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

[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

DEEP LEARNING

Programming with TensorFlow & Keras

- 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

```
keras.layers.Dense(units,
                   activation=None,
                   use_bias=True,
                   kernel_initializer='glorot_uniform',
                   bias_initializer='zeros',
                   kernel_regularizer=None,
                   bias_regularizer=None,
                   activity_regularizer=None,
                   kernel_constraint=None,
                   bias_constraint=None)


keras.optimizers.SGD(lr=0.01,
                     momentum=0.0,
                     decay=0.0,
                     nesterov=False)
```

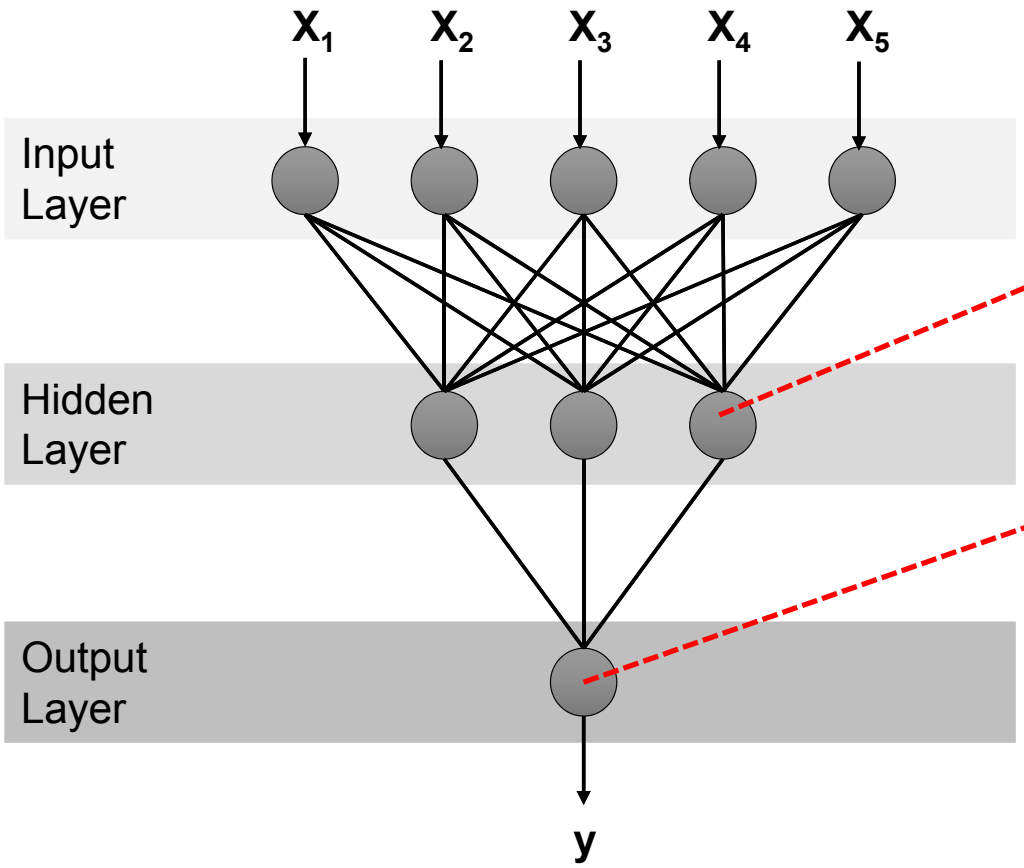
- Tool Keras supports inherently the creation of artificial neural networks using Dense layers and optimizers (e.g. SGD)
- Includes regularization (e.g. weight decay) or momentum

[8] Keras Python Deep Learning Library

[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

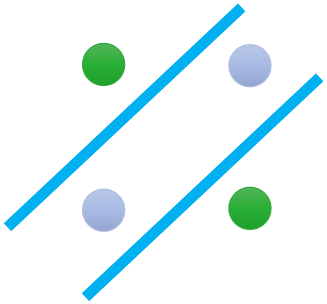
DEEP LEARNING – SIMPLE MNIST EXAMPLE

Creating an Artificial Neural Network (ANN) using Python Scripts



(28*28 pixels, 60000 training samples ~47MB, 10000 test samples ~7.8 MB)

- Think each hidden node as a 'simple perceptron' that each creates one hyperplane



- Think the output node simply combines the results of all the perceptrons to yield the 'decision boundary' above

- Feed-forward neural network: nodes in one layer are connected only to the nodes in the next layer (i.e. 'a constraint of network construction')

DEEP LEARNING – PYTHON SCRIPT PART 1

ANN Parameters & Data Normalization in Python Script

```
import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.utils import np_utils
```

```
# parameters
NB_CLASSES = 10
NB_EPOCH = 200
BATCH_SIZE = 128
VERBOSE = 1
N_HIDDEN = 128
OPTIMIZER = 'SGD'
VALIDATION_SPLIT = 0.2
```

- **NB_CLASSES: 10 Class Problem**
- **NB_EPOCH: number of times the model is exposed to the training set – at each iteration the optimizer adjusts the weights so that the objective function is minimized**
- **BATCH_SIZE: number of training instances taken into account before the optimizer performs a weight update**
- **OPTIMIZER: Stochastic Gradient Descent ('SGD') – only one training sample/iteration**

```
# dataset 28 x 28 pixels = 784 reshaped
(X_train, y_train), (X_test, y_test) = mnist.load_data()
RESHAPED = 784
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalization
X_train /= 255
X_test /= 255

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

- **Data load shuffled between training and testing set**
- **Data preparation, e.g. X_train is 60000 samples / rows of 28 x 28 pixel values that are reshaped in 60000 x 784 including type specification (i.e. float32)**
- **Data normalization: divide by 255 – the max intensity value to obtain values in range [0,1]**

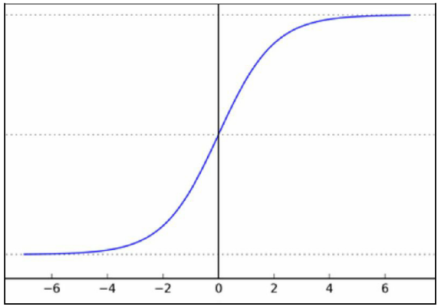
DEEP LEARNING – PYTHON SCRIPT PART 2

Dynamic & Flexible Modeling of the ANN in Python Script

- **The Sequential() Keras model is a linear pipeline (aka 'a stack') of various neural network layers including Activation functions of different types (e.g. softmax)**
- **Dense() represents a fully connected layer used in ANNs that means that each neuron in a layer is connected to all neurons located in the previous layer**
- **The non-linear Activation function 'softmax' represents a generalization of the sigmoid function – it squashes an n-dimensional vector of arbitrary real values into a n-dimenensional vector of real values in the range of 0 and 1 – here it aggregates 10 answers provided by the Dense layer with 10 neurons**

```
# 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 ANN model
model = Sequential()
model.add(Dense(NB_CLASSES, input_shape=(RESHAPED,)))
model.add(Activation('softmax'))
model.summary()

# 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])
```

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(\mathbf{x}_i)}{\sum_j \exp(\mathbf{x}_j)}$$



- **Loss function is a multiclass logarithmic loss: target is *ti,j* and prediction is *pi,j***

$$L_i = -\Sigma_j t_{i,j} \log(p_{i,j})$$

DEEP LEARNING – RUNNING THE PYTHON SCRIPT

Using a Job Script for High Performance Computing (HPC) Machine with the Python Script & ANN Model

```
#!/bin/bash
#PBS -l nodes=1:ppn=all
#PBS -l walltime=1:0:0
#PBS -N KERAS_MNIST_ANN_HIDDEN

module load TensorFlow/1.4.0-intel-2017b-Python-3.6.3
module load Keras/2.1.1-intel-2017b-Python-3.6.3

# make sure Keras is using TensorFlow as backend
export KERAS_BACKEND=tensorflow

export WORKDIR=$VSC_SCRATCH/${PBS_JOBNAME}_${PBS_JOBID}
mkdir -p $WORKDIR
cd $WORKDIR

export OMP_NUM_THREADS=1
python $PBS_O_WORKDIR/KERAS_MNIST_ANN_HIDDEN.py

echo "Working directory was $WORKDIR"
```

```
[vsc42544@gligar03 deeplearning]$ more KERAS_MNIST_ANN_HIDDEN.o1179466
60000 train samples
10000 test samples

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
activation_1 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_2 (Activation)    (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
Train on 48000 samples, validate on 12000 samples
Epoch 1/200
```

```
[vsc42544@gligar03 deeplearning]$ tail KERAS_MNIST_ANN_HIDDEN.o1179466

   32/10000 [..............................] - ETA: 0s
 2272/10000 [=====>........................] - ETA: 0s
 4544/10000 [============>.................] - ETA: 0s
 6784/10000 [====================>.........] - ETA: 0s
 9088/10000 [============================>...] - ETA: 0s
10000/10000 [==============================] - 0s 22us/step
Test score: 0.0772481116249
Test accuracy: 0.9773
Working directory was /user/scratch/gent/vsc425/vsc42544/KERAS_MNIST_ANN_HIDDEN_1179466.master19.golett.gent.vsc
```

```
.] - ETA: 4:29 - loss: 2.3122 - acc: 0.1094
.] - ETA: 16s - loss: 2.2732 - acc: 0.1085
.] - ETA: 7s - loss: 2.2178 - acc: 0.1721
.] - ETA: 4s - loss: 2.1676 - acc: 0.2515
```

[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

DEEP LEARNING – CHANGING THE PYTHON SCRIPT

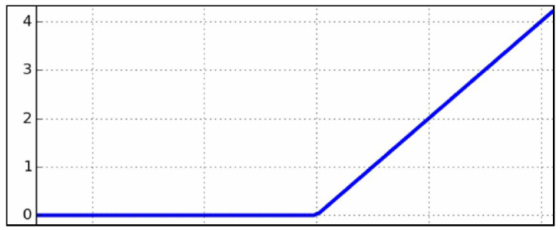
Dynamic & Flexible Modeling of the ANN adding Hidden Layers in Python Script

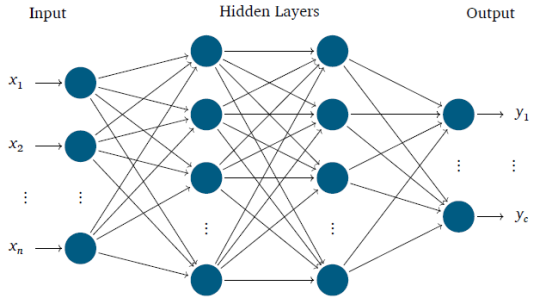
- **A hidden layer in an ANN can be represented by a fully connected Dense layer in Keras by just specifying the number of hidden neurons in the hidden layer**

- **The non-linear Activation function 'relu' represents a so-called Rectified Linear Unit (ReLU) that only recently became very popular because it generates good experimental results in ANNs and more recent deep learning models – it just returns 0 for negative values and grows linearly for only positive values**

```
# 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)

# ANN model with hidden layers
model = Sequential()
model.add(Dense(N_HIDDEN, input_shape=(RESHAPED,)))
model.add(Activation('relu'))
model.add(Dense(N_HIDDEN))
model.add(Activation('relu'))
model.add(Dense(NB_CLASSES))
model.add(Activation('softmax'))
model.summary()

# 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])
```

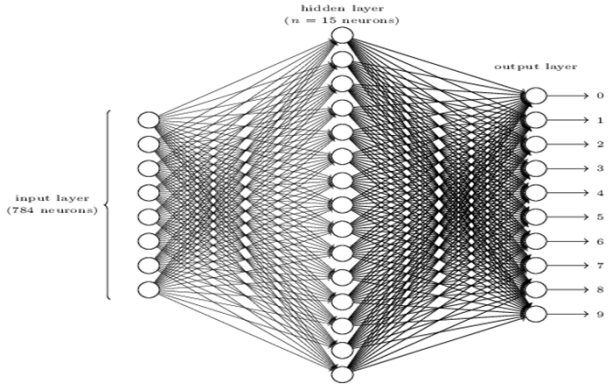
$$f(x) = \max(0, x)$$



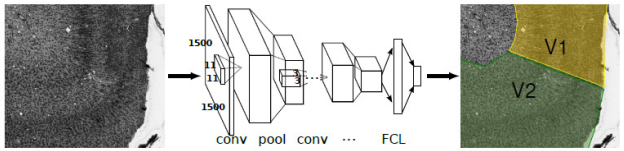


DEEP LEARNING – ARCHITECTURES

Each of the Architectures provide Unique Characteristica (e.g. ‘smart layers‘)



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



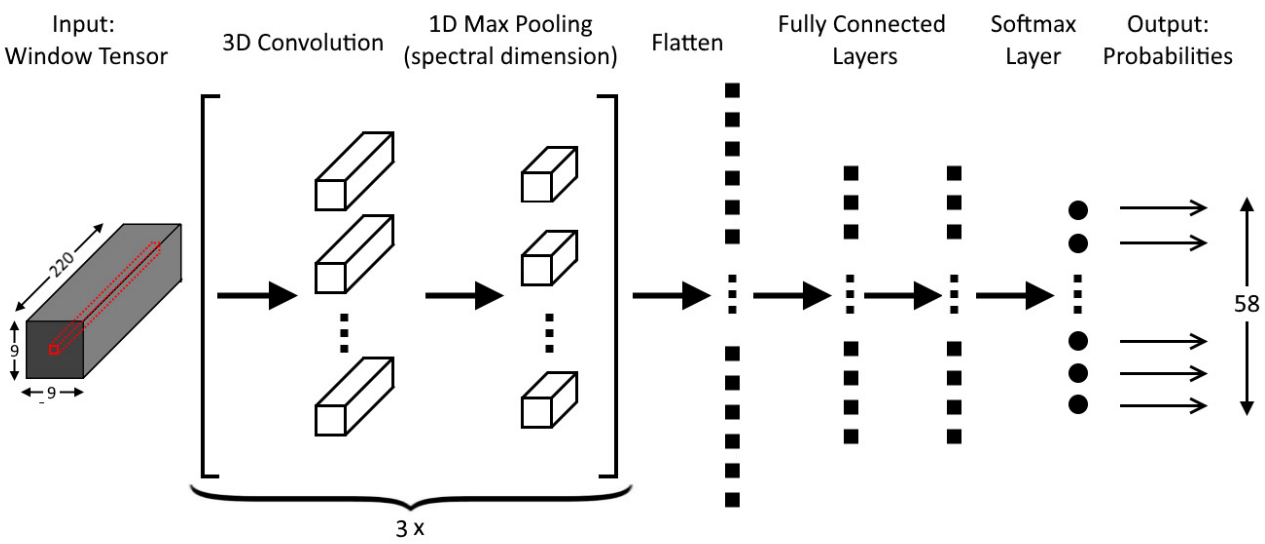
- Deep Belief Network (DBN)
  - Composed of mult iple layers of variables; only connections between layers
- Recurrent Neural Network (RNN)
  - ‘ANN‘ but connections form a directed cycle; state and temporal behaviour

- **Deep Learning architectures can be classified into Deep Neural Networks, Convolutional Neural Networks, Deep Belief Networks, and Recurrent Neural Networks all with unique characteristica**
- **Deep Learning needs ‘big data‘ to work well & for high accuracy – works not well on sparse data**

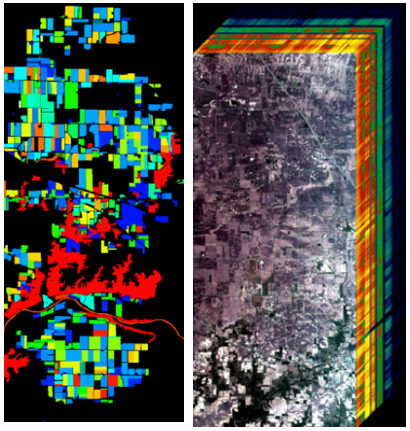
DEEP LEARNING – EXPERIMENTING WITH TOPOLOGIES

Programming with Python & TensorFlow & Keras – Supervised Classification Example – Network Topology

- Classify pixels in a hyperspectral remote sensing image having groundtruth/labels available
- Created CNN architecture for a specific hyperspectral land cover type classification problem
- Used dataset of Indian Pines (compared to other approaches) using all labelled pixels/classes
- Performed no manual feature engineering to obtain good results (aka accuracy)



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

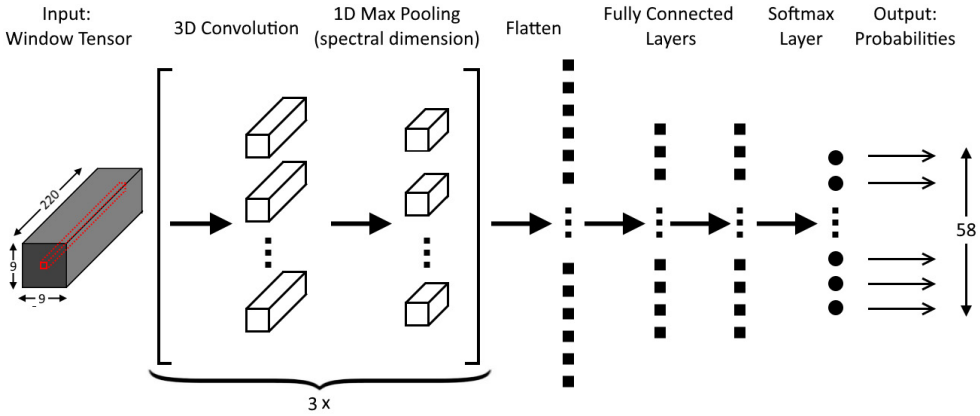


[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

DEEP LEARNING – EXPERIMENTING WITH TOPOLOGIES

Programming with Python & TensorFlow & Keras – Supervised Classification Example – Network Topology

```
# standard model that achieved 85% accuracy on test in best of 5
# The pool_size is adapted to 220 channels
# The first two dimensions of kernel_size are adapted to 9x9 window tensors
def build_model_standard(input_shape, activation, num_classes):
    model = Sequential()
    model.add(Conv3D(48, kernel_size=(3, 3, 5), activation=activation, input_shape=input_shape))
    model.add(MaxPooling3D(pool_size=(1, 1, 3)))
    model.add(ZeroPadding3D((0, 0, 2), data_format=None))
    model.add(Conv3D(32, kernel_size=(3, 3, 5), activation=activation))
    model.add(MaxPooling3D(pool_size=(1, 1, 3)))
    model.add(ZeroPadding3D((0, 0, 2), data_format=None))
    model.add(Conv3D(32, kernel_size=(3, 3, 5), activation=activation))
    model.add(MaxPooling3D(pool_size=(1, 1, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation=activation))
    model.add(Dense(128, activation=activation))
    model.add(Dense(num_classes, activation='softmax'))
    return model
```

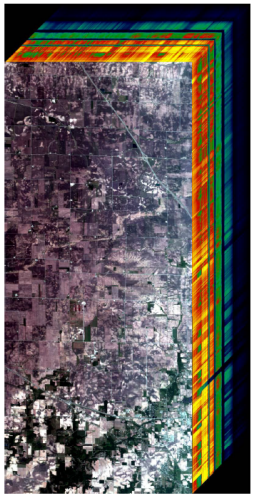


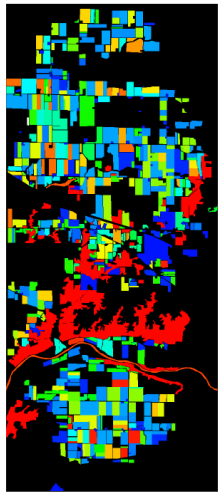
[10] J. Lange, G. Cavallaro, M. Riedel, et al. , IGARSS 2018

[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

DEEP LEARNING HYPERPARAMETER TUNING

Programming with Python & TensorFlow & Keras – Supervised Classification Example – Network Topology

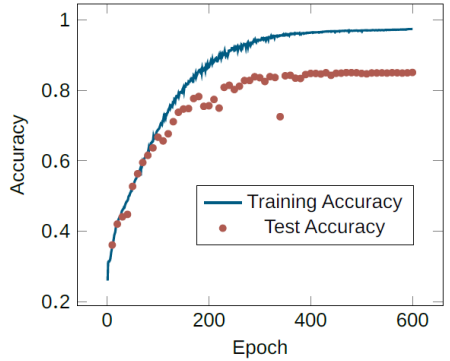


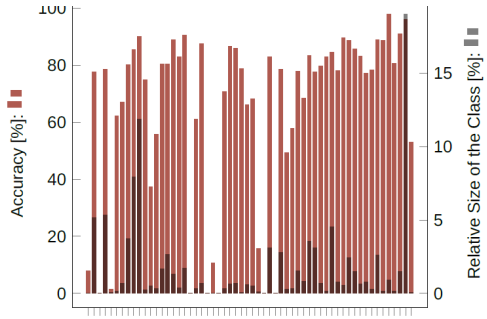


Blue: correctly classified
Red: incorrectly classified

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

Feature	Representation / Value
Conv. Layer Filters	48, 32, 32
Conv. Layer Filter size	(3,3,5), (3,3,5), (3,3,5)
Dense Layer Neurons	128, 128
Optimizer	SGD
Loss Function	mean squared error
Activation Functions	ReLU
Training Epochs	600
Batch Size	50
Learning Rate	1
Learning Rate Decay	5×10^{-6}



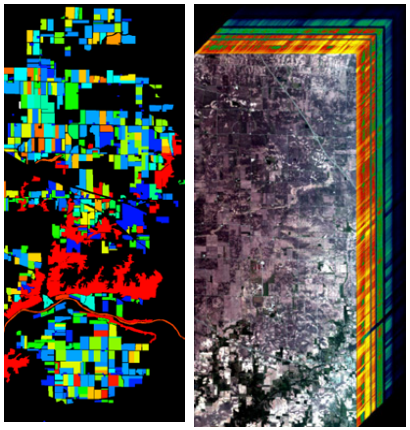


- **Using Python with TensorFlow & Keras easily enables changes in hyper-parameter tuning**
- **Various runs on different topologies add up to computational demand of GPUs**
- **Need for HPC machines with good GPUs and good deep learning software stacks required**

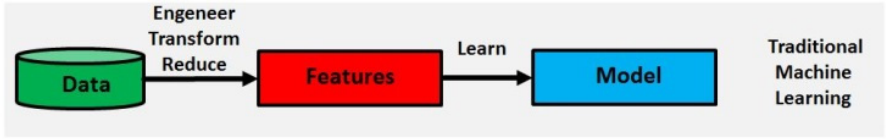
[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University

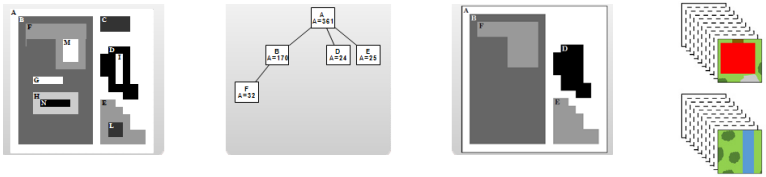
COMPARE TRADITIONAL MACHINE LEARNING

Supervised Classification Example – Remote Sensing Dataset & Results

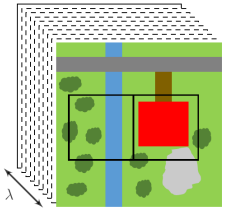


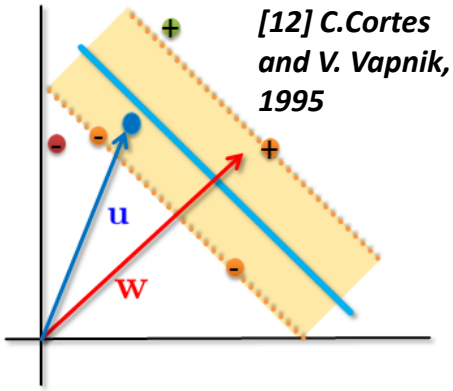
- Traditional Methods
  - Support Vector Machine (SVM)
  - 52 classes of different land cover, 6 discarded, mixed pixels, rare groundtruth
  - Substantial manual feature engineering,e.g. Self Dual Attribute Profile (SDAP)
  - 10-fold cross-validation
  - Achieved 77,02 % accuracy



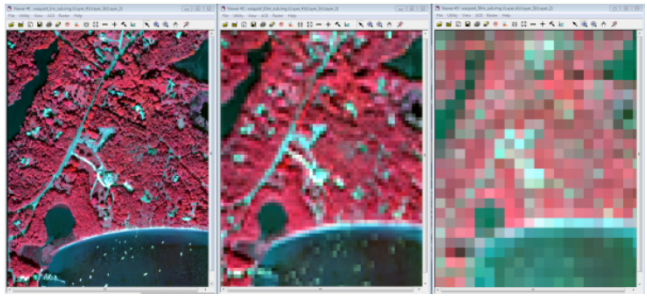


[11] G. Cavallaro et al.

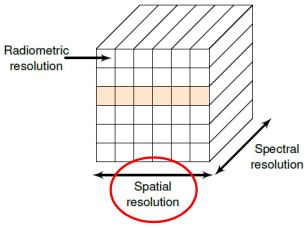




[12] C.Cortes and V. Vapnik, 1995



1m *10m* *30m*

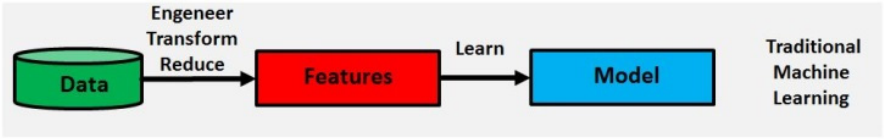


[13] G. Cavallaro and M. Riedel, et al. , 2015

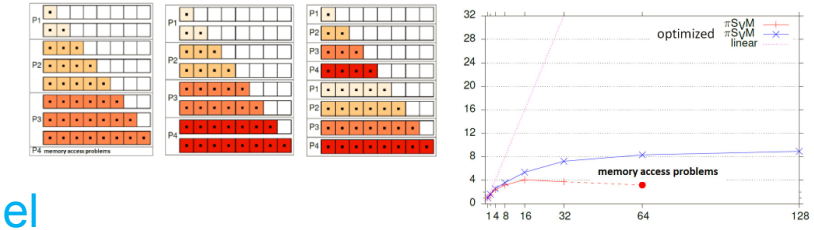
COMPARE TRADITIONAL MACHINE LEARNING

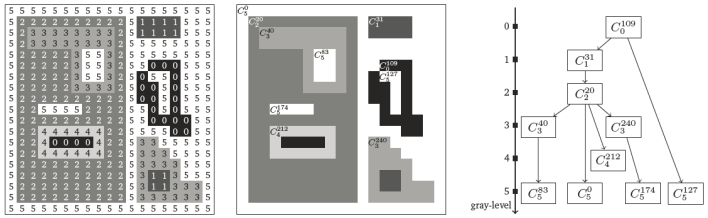
Supervised Classification Example – Speed-up Cross-Validation & MPI C Code

[16] piSVM Website, 2011/2014 code



- Traditional Methods
  - Support Vector Machine (SVM)
  - pISVM C code that can be improved (taken from ML experts – not parallel experts, tuned @ JSC)
  - Message Passing Interface (MPI)





[9] M. Goetz and M. Riedel, et al., 2018

- Feature Engineering
  - Working also on parallel feature engineering using tree-based approach (MPI/OpenMP C)

Scenario 'pre-processed data', 10xCV **serial**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	48.90 (18.81)	65.01 (19.57)	73.21 (20.11)	75.55 (22.53)	74.42 (21.21)
4	57.53 (16.82)	70.74 (13.94)	75.94 (13.53)	76.04 (14.04)	74.06 (15.55)
8	64.18 (18.30)	74.45 (15.04)	**77.00** (14.41)	75.78 (14.65)	74.58 (14.92)
16	68.37 (23.21)	76.20 (21.88)	76.51 (20.69)	75.32 (19.60)	74.72 (19.66)
32	70.17 (34.45)	75.48 (34.76)	74.88 (34.05)	74.08 (34.03)	73.84 (38.78)

Scenario 'pre-processed data', 10xCV **parallel**: accuracy (min)

γ/C	1	10	100	1000	10 000
2	75.26 (1.02)	65.12 (1.03)	73.18 (1.33)	75.76 (2.35)	74.53 (4.40)
4	57.60 (1.03)	70.88 (1.02)	75.87 (1.03)	76.01 (1.33)	74.06 (2.35)
8	64.17 (1.02)	74.52 (1.03)	**77.02 (1.02)**	75.79 (1.04)	74.42 (1.34)
16	68.57 (1.33)	76.07 (1.33)	76.40 (1.34)	75.26 (1.05)	74.53 (1.34)
32	70.21 (1.33)	75.38 (1.34)	74.69 (1.34)	73.91 (1.47)	73.73 (1.33)

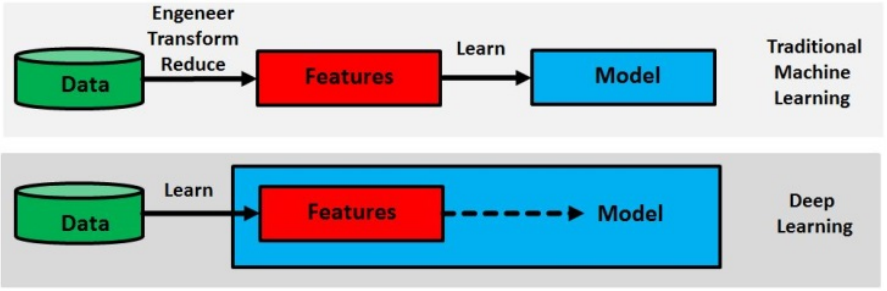
First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

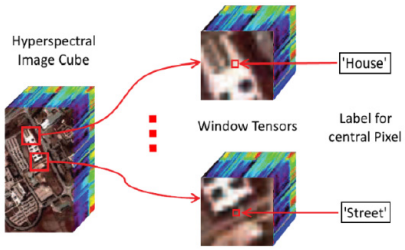
[13] G. Cavallaro and M. Riedel, et al. , 2015

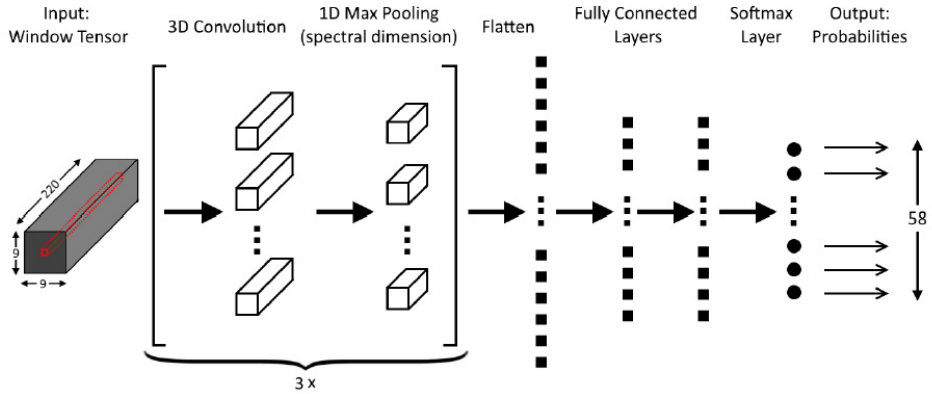
SELECTED COMPARISONS

Supervised Classification Example – Remote Sensing Dataset & Results

- Traditional Methods
  - C MPI-based Support Vector Machine (SVM)
  - Substantial manual feature engineering
  - 10-fold cross-validation for model selection
  - Achieved 77,02 % accuracy
- Convolutional Neural Networks (CNNs)
  - Python/TensorFlow/Keras
  - Automated feature learning
  - Achieved 84,40 % accuracy on all 58 classes







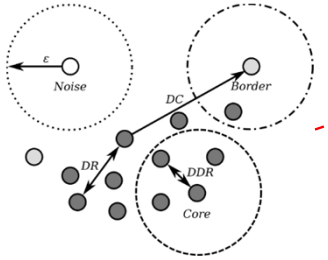
[10] J. Lange, G. Cavallaro, M. Riedel, et al. , 2018

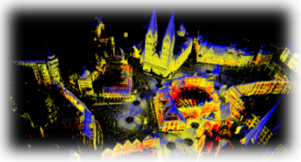
- **SVM + Feature Engineering (~3 years) vs. CNN architecture setup (~1 month)**

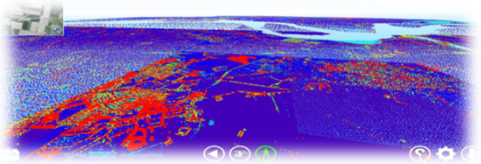
DEEP SERIES OF PROJECTS

EU Projects Driven by Co-Design of HPC Applications

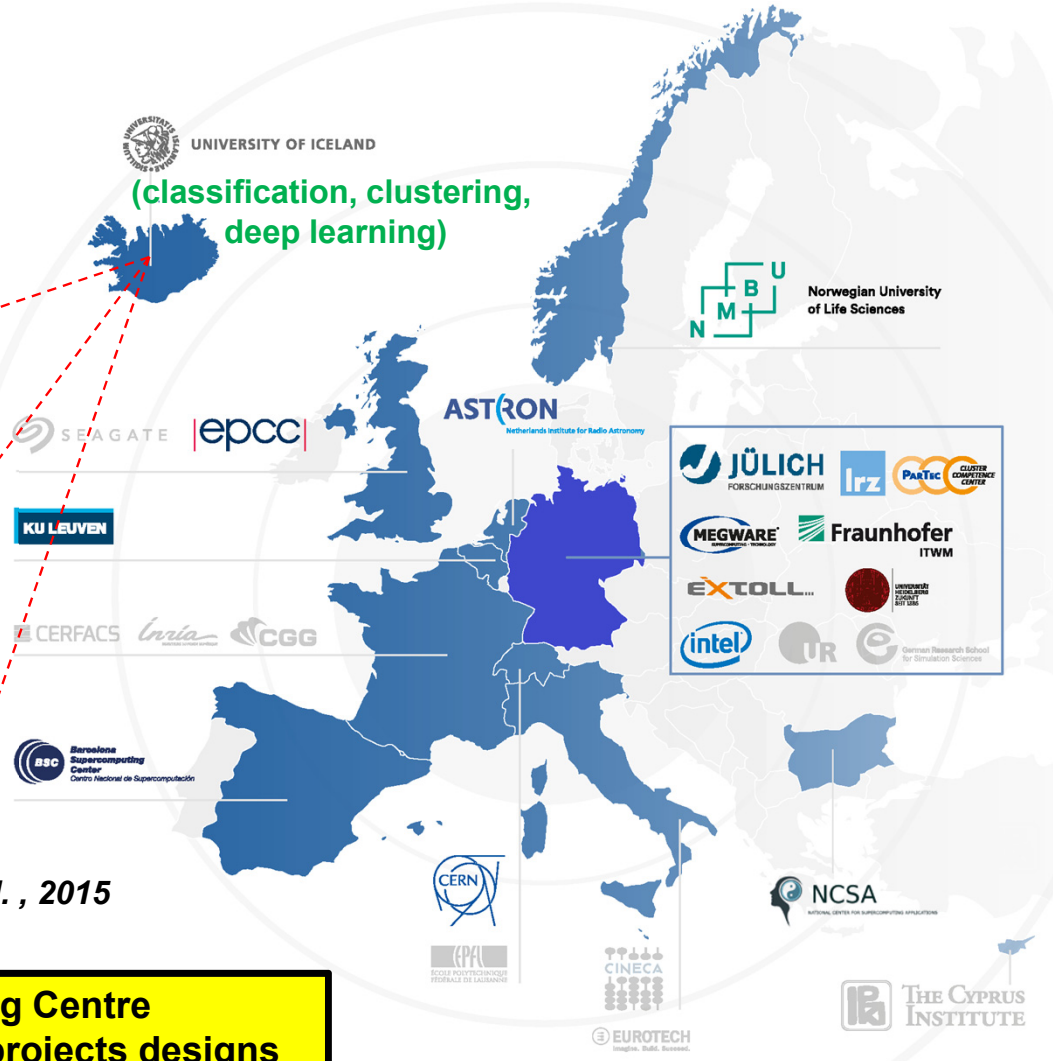
- 3 EU Exascale projects
  - DEEP
  - DEEP-ER
  - DEEP-EST
- 27 partners
  - Coordinated by JSC
- EU-funding: 30 M€
  - JSC-part > 5,3 M€
- Nov 2011 – Jun 2020







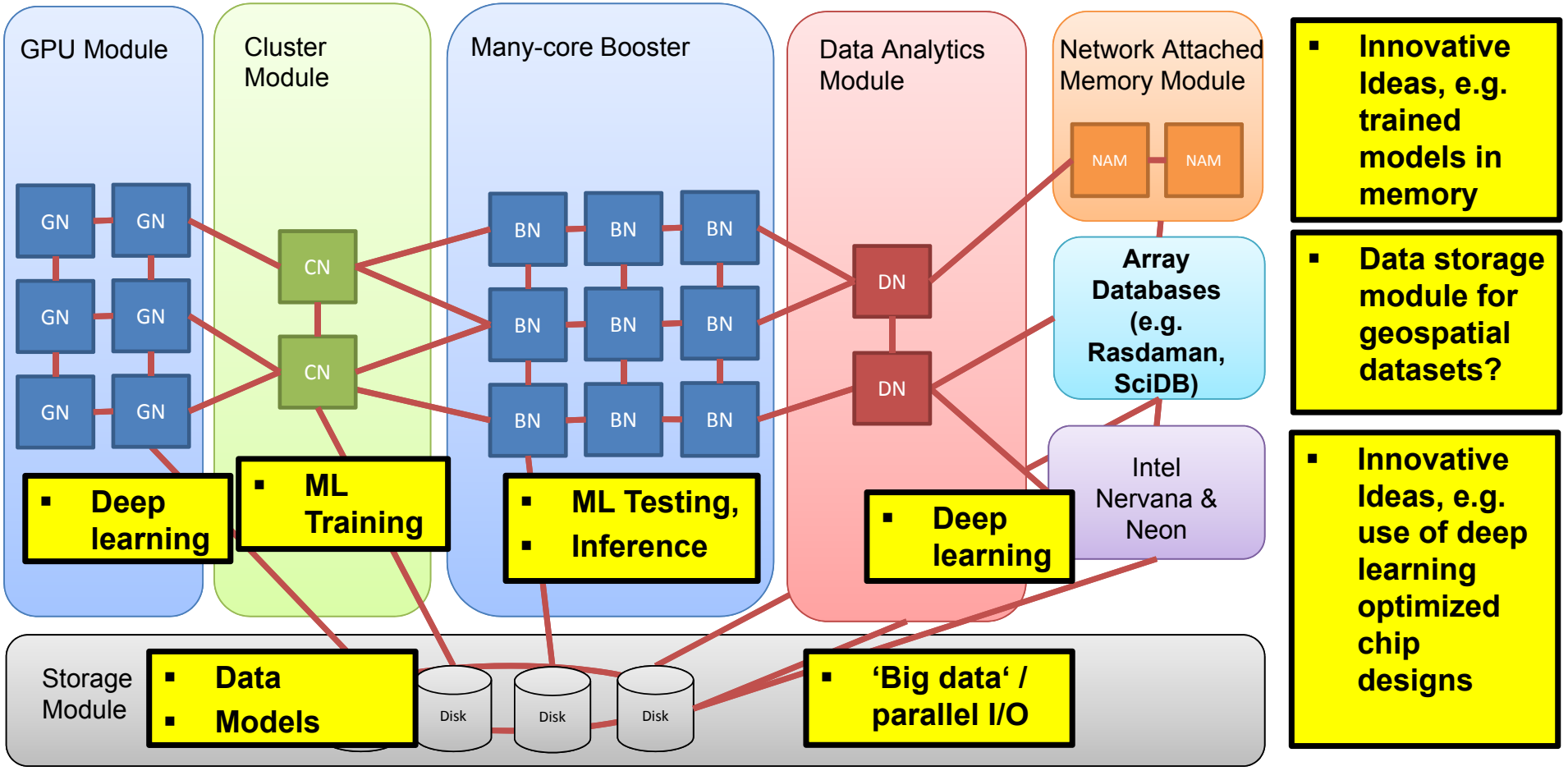
[14] M. Goetz & M. Riedel, et al. , 2015



- **Juelich Supercomputing Centre implements the DEEP projects designs in its production infrastructure**

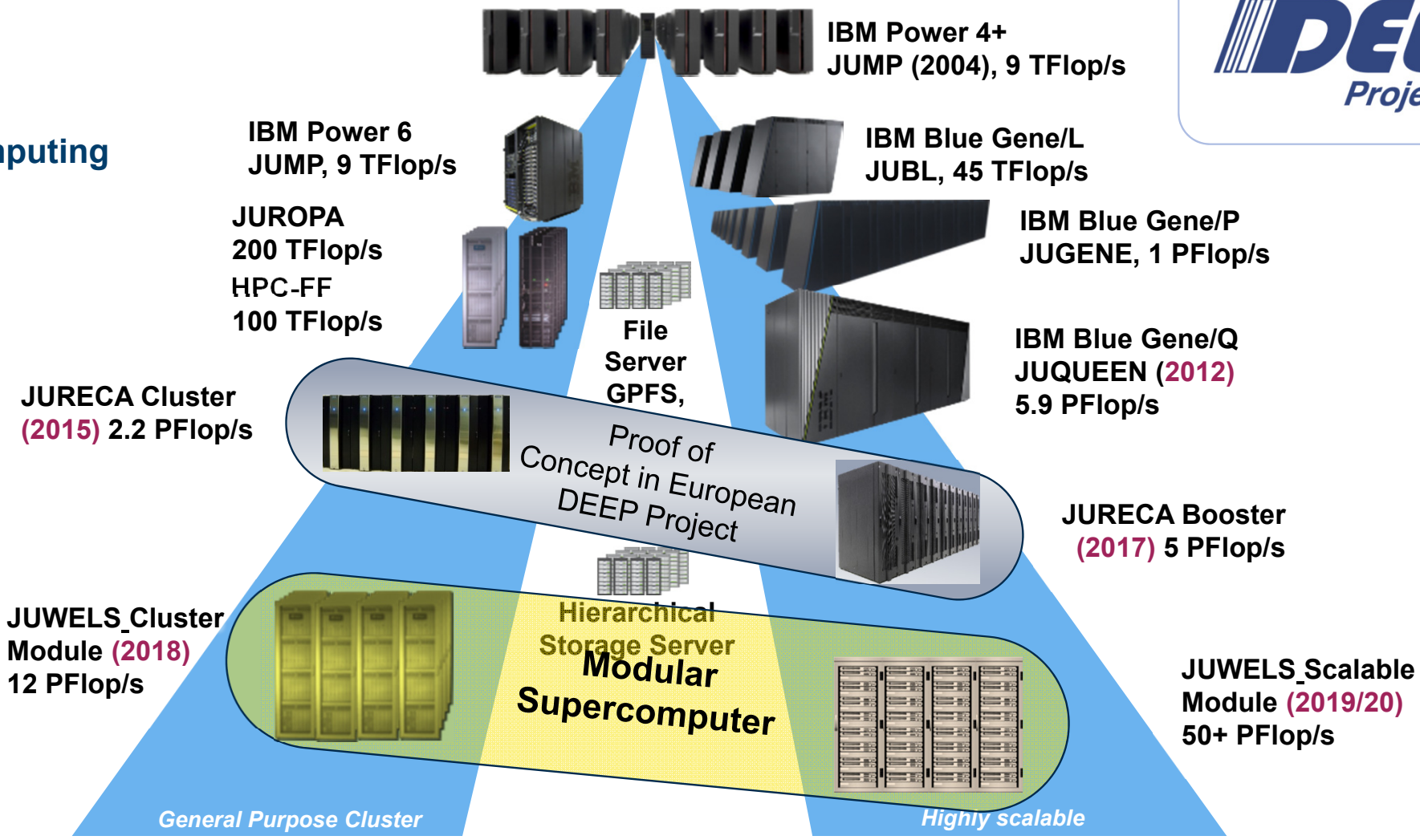
JSC – MODULAR SUPERCOMPUTING ARCHITECTURE

Roadmap



JSC

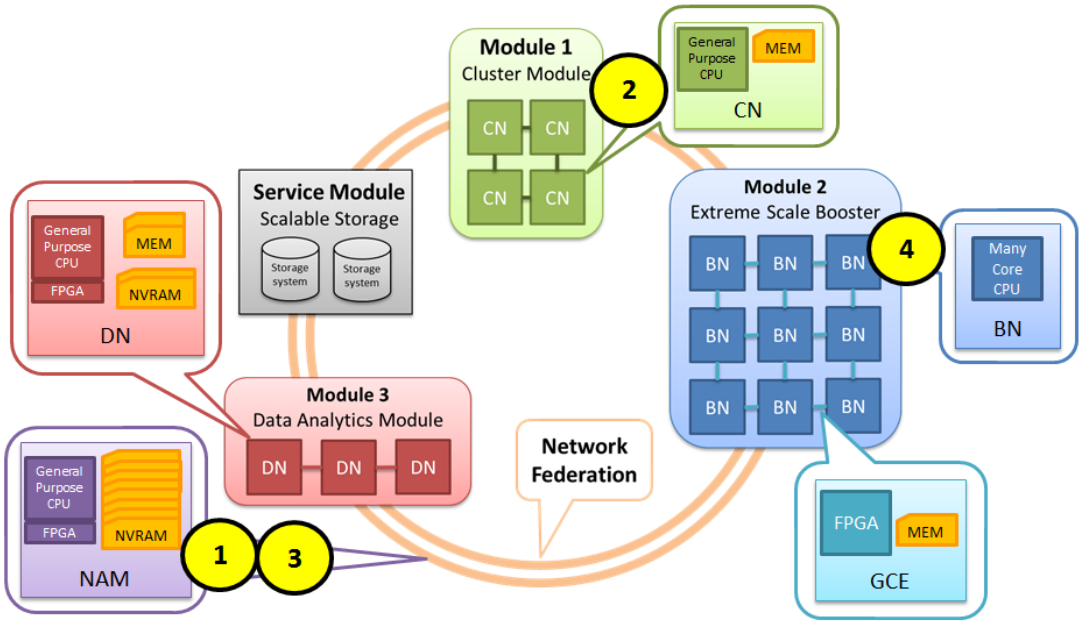
Juelich Supercomputing Centre



TRADITIONAL MACHINE LEARNING

Supervised Classification – Modular Supercomputing Architecture

- (1) The training dataset and testing dataset of the remote sensing application is used many times in the process and make sense to put into the DEEP-EST Network Attached Memory (NAM) module
- (2) Training with piSVM in order to generate a model requires powerful CPUs with good interconnection for the inherent optimization process and thus can take advantage of the DEEP-EST CLUSTER module (use of training dataset, requires piSVM parameters for kernel and cost)
- (3) Instead of dropping the trained SVM model (i.e. file with support vectors) to disk it makes sense to put this model into the DEEP-EST NAM module
- (4) Testing with piSVM in order to evaluate the model accuracy requires not powerful CPUs and not a good interconnection but scales perfectly (i.e. nicely parallel) and thus can take advantage of the BOOSTER module (use of testing dataset & model file residing in NAM), prediction & inference using models is largely usable on the BOOSTER module too
- (5) If accuracy too low back to (2) to change parameters

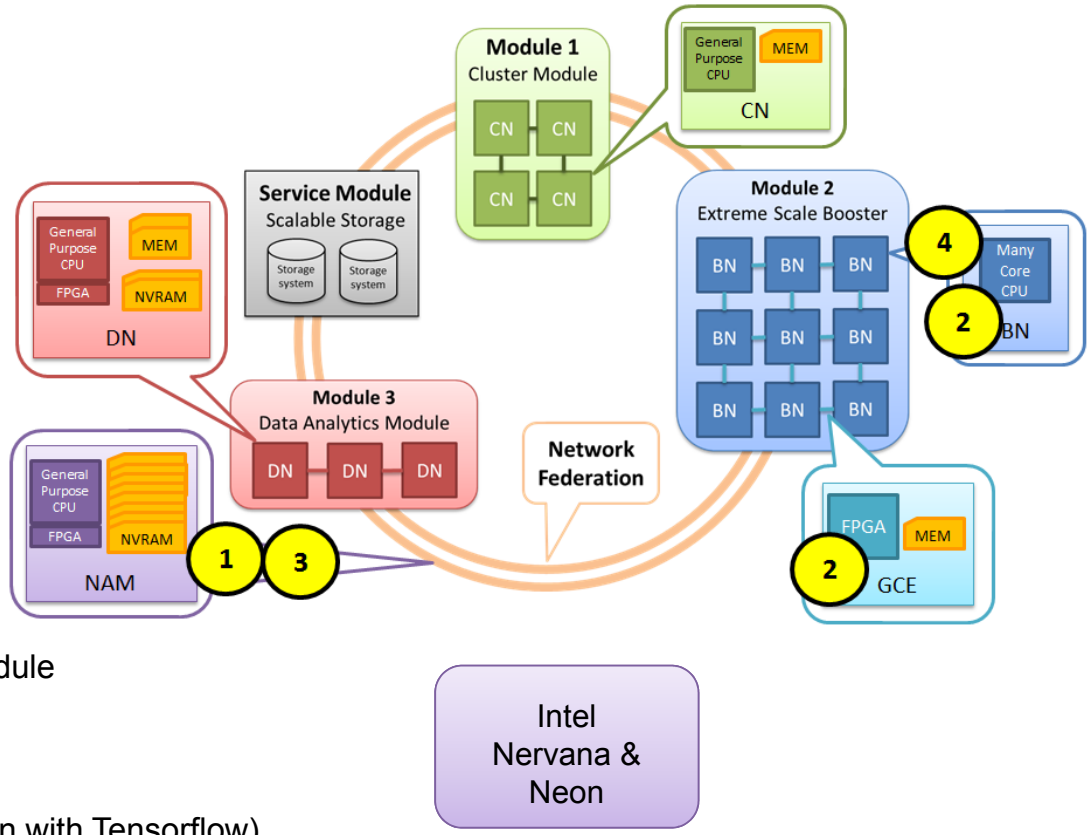


[15] E. Erlingsson, G. Cavallaro, M. Riedel, et al. , 2018

DEEP LEARNING (WORK IN PROGRESS)

Supervised Classification – CNN Design & Setup

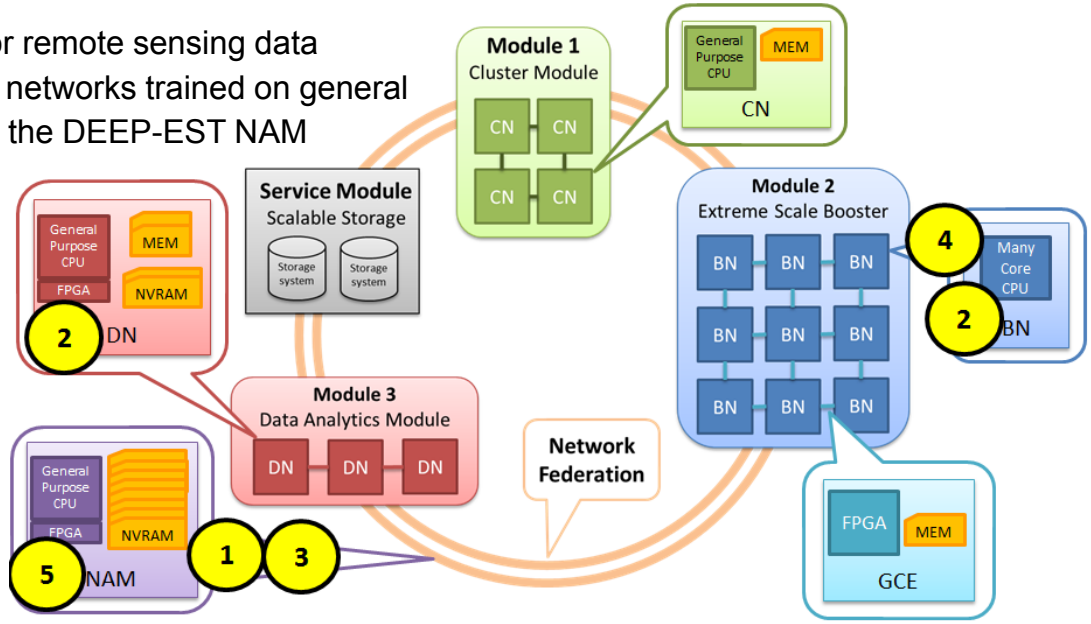
- (1) The training dataset and testing dataset of the remote sensing application is used many times in the process and make sense to put into the DEEP-EST Network Attached Memory (NAM) module
- (2) Training with CNNs in Tensorflow works best fore many-core CPUs for the inherent optimization process based on Stochastic Gradient Descent (SGD) MPI collective available in the DEEP-EST Global Collective Environment (GCE) and thus can take advantage of the DEEP-EST BOOSTER module (use of training dataset, requires CNN architectural design parameters)
- (3) Trained models of selected architectural CNN setups need to be compared and thus can be put in the DEEP-EST NAM module
- (4) Testing with Tensorflow in order to evaluate the model accuracy works also quite well for many-core architectures and scales perfectly (i.e. nicely parallel) and thus can take advantage of the BOOSTER module (use of testing dataset & CNN models residing in NAM)
- (5) If accuracy too low back to (2) to change parameters
- (Upcoming: potentially exploring the use of Intel Nervana Chips & Neon with Tensorflow)



DEEP LEARNING

Supervised Classification – Transfer Learning

- (1) Studies have shown that Transfer Learning works well especially for remote sensing data without groundtruth or labelled data (i.e. unsupervised) and pre-trained networks trained on general images like ImageNet (e.g. like Overfeat) are available and are put into the DEEP-EST NAM module to be re-used for unsupervised deep learning CNN training
- (2) Based on pre-trained features another CNN architectural setup is trained with the real remote sensing data whereby the DEEP-EST DATA ANALYTICS module is an interesting approach since the FPGA might be used to compute the transformation from pre-trained features as suitable inputs to the real training process of the CNN based on remote sensing data
- (3) Trained models of selected architectural CNN setups that have been used with pre-trained features need to be compared and thus can be put in the DEEP-EST NAM module



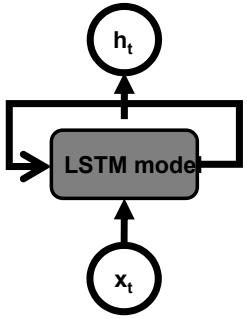
- (4) Testing with Tensorflow in order to evaluate the model accuracy works also quite well for many-core architectures and scales perfectly (i.e. nicely parallel) and thus can take advantage of the BOOSTER module (use of testing dataset & CNN models residing in NAM)
- (5) Testing results are written back to the DEEP-EST NAM per CNN architectural design, the FPGA in the NAM can compute the best obtained accuracy for all the different setups
- (6) If accuracy is too low consider to move back to step (1) to change the pre-trained network or step (2) to create a better CNN architectural

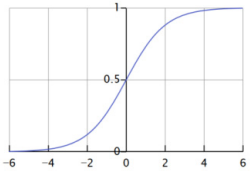
OTHER MODELS – LSTM FOR TIME SERIES ANALYSIS

Python with TensorFlow/Keras support easy development of LSTM network

- Long Short Term Memory (LSTM) networks are a special kind of Recurrent Neural Networks (RNNs)
- LSTMs learn long-term dependencies in data by remembering information for long periods of time
- The LSTM chain structure consists of four neural network layers interacting in a specific way

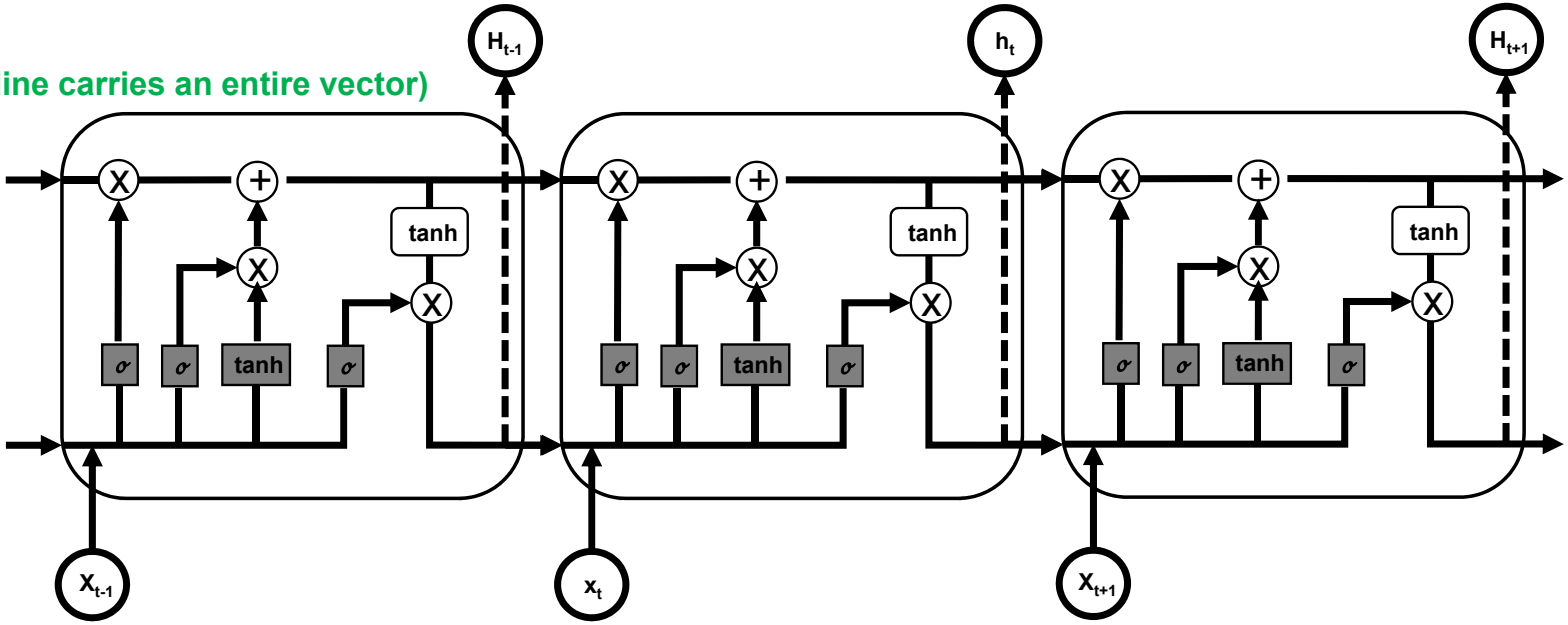
[1] M. Riedel, Invited YouTube Tutorial on Deep Learning, Ghent University





(uses sigmoid σ)

(each line carries an entire vector)



OTHER MODELS – LSTM EXAMPLE

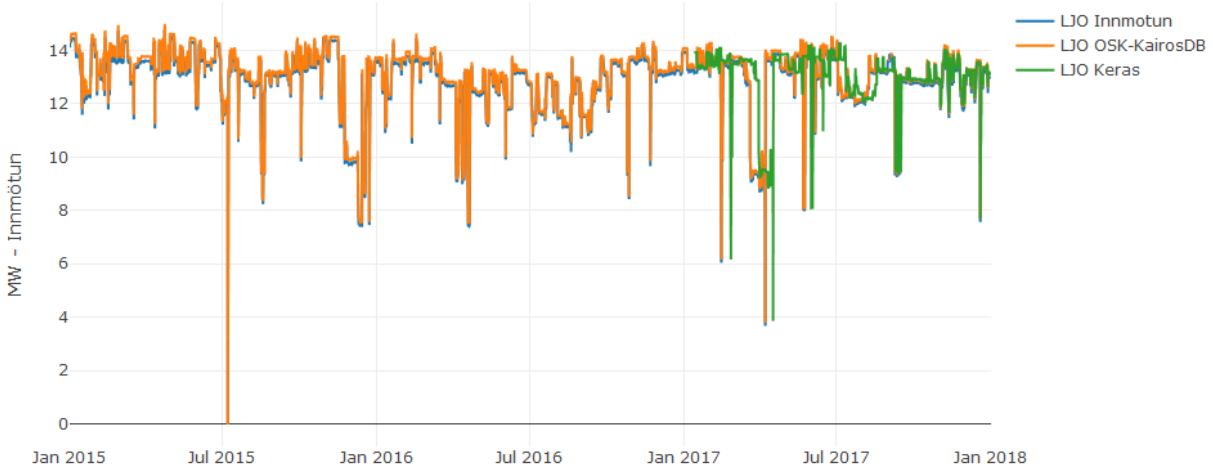
Python with TensorFlow/Keras & LSTM Models

Landsvirkjun
National Power Company of Iceland

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
# design network
model = Sequential()
model.add(LSTM(
    units=config['units'],
    input_shape=(train_X.shape[1], train_X.shape[2])
))
model.add(Dense(1, activation=config['activation']))

model.compile(loss=config['loss'], optimizer=config['optimizer'])

# fit network
print("Fitting model..")

history = model.fit(
    train_X,
    train_y,
    epochs=config['epochs'],
    batch_size=config['batchsize'],
    validation_data=(test_X, test_y),
    verbose=2,
    shuffle=config['shuffle']
)
```

- **Prototyping sequence networks with LSTM models is easy using Python with Tensorflow and Keras library**
- **LSTM models work quite well to predict power but needs to be trained and tuned for different power stations**
- **Observing that some peaks can not be ‘learned‘**



DEEP LEARNING & GPU PARALLELIZATION

Simple Image Benchmark on JURECA JSC HPC System (75 x 2 NVIDIA Tesla K80/node – dual GPU design)

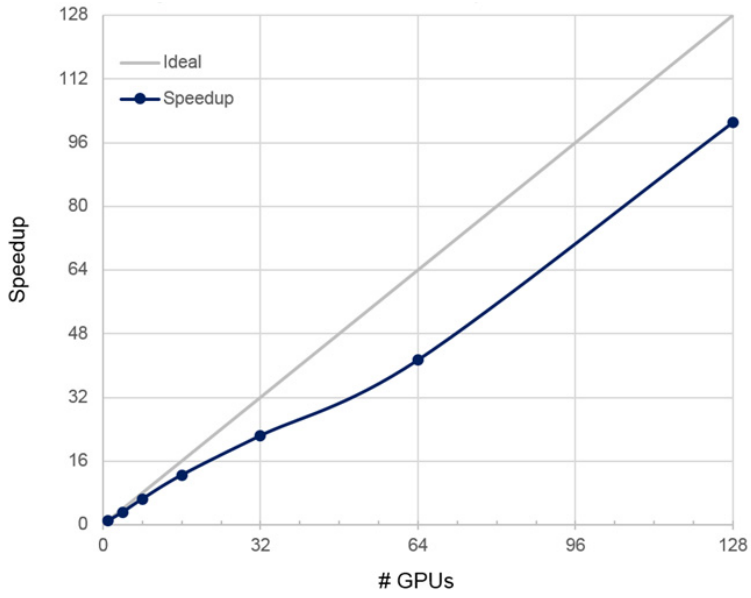
[19] JURECA HPC System

- Setup
  - TensorFlow 1.4
  - Python 2.7
  - CUDA 8
  - cuDNN 6
  - Horovod 0.11.2
  - MVAPICH-2.2-GDR
  - 'Simple' 1.2 million images with 224 x 224 pixels

#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

[20] A. Sergeev, M. Del Balso,'Horovod', 2018

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

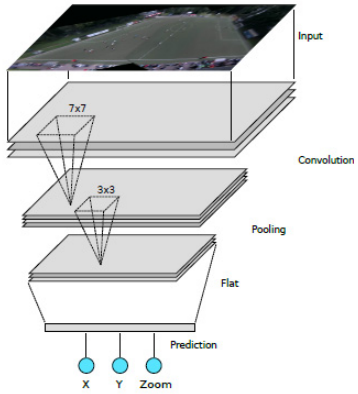


- **Open source tool Horovod enables distributed deep learning using Python with TensorFlow (and Keras)**
- **Machine & Deep Learning: speed-up is just secondary goal after primary goal accuracy**
- **Speed-up & parallelization nice to have for faster hyperparameter tuning, model creation, and inference**
- **Third goal is avoiding much feature engineering through 'feature learning' of deep learning networks**

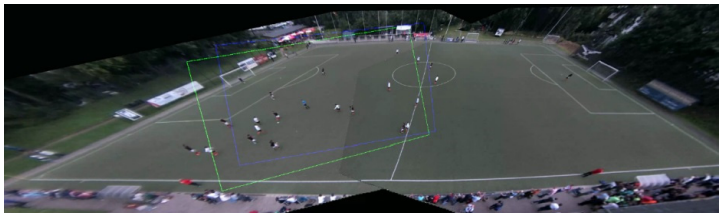
DEEP LEARNING – OTHER ACTIVITIES @ JSC

Selected Projects related to Deep Learning

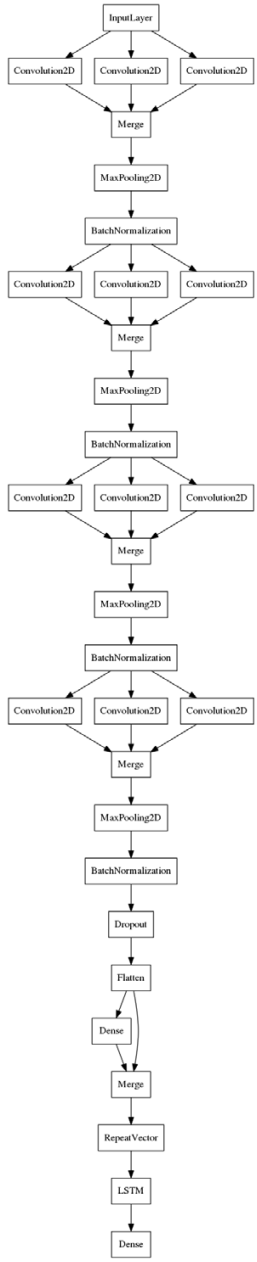
- SoccerWatch.TV
  - SME created/joined by a ‘exit-ing‘ PHD Student
  - Besides upper leagues: 80k matches/week
  - Recording too expensive (amateurs) with camera man needed
  - Approach: Find X,Y center and zoom on panorama camera using Deep Learning
- SMITH
  - Virtual Patient models with Markov Chains Monte Carlo (MCMC) mapped to Deep Learning networks



[17] SoccerWatch.TV Web page



[18] SMITH Web page



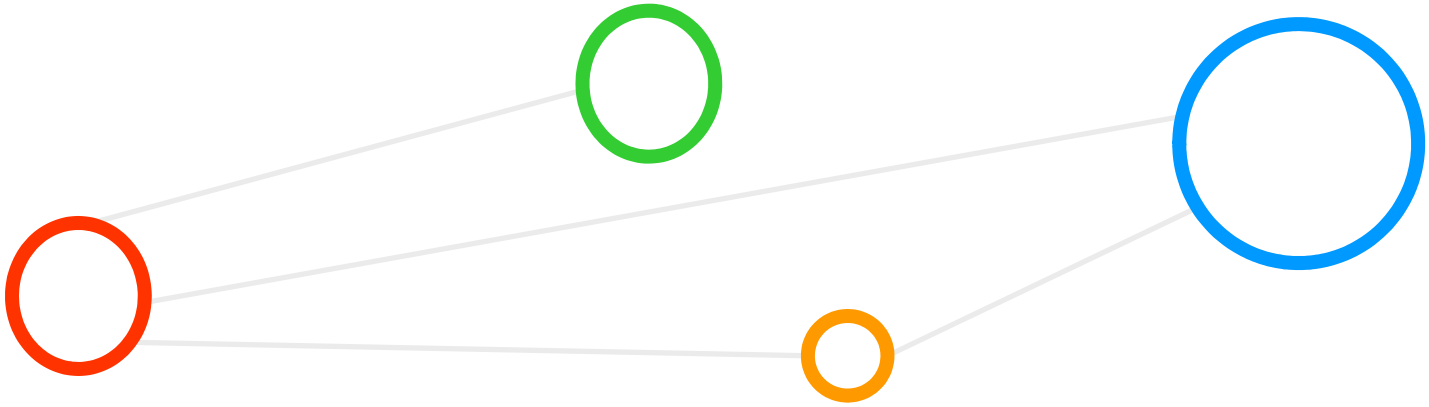
SUMMARY

- Mindset
  - Think traditional machine learning still relevant for deep learning
  - Using interpreted languages like Python enable easy & flexible parameter-tuning
  - Selected new approaches with specific deep learning per problem (CNN, LSTM, etc.)

- Skillset
  - Basic knowledge of machine learning required for deep learning
  - Faster experimentation and use of various topologies and architectures through Python
  - Validation (i.e. model selection) and regularization still valid(!)

- Toolset
  - Parallel versions of traditional machine learning methods exist (piSVM, HPDBSCAN)
  - Python with Tensorflow & Keras just one example & good performance (good install!)
  - Explore technology trends, e.g. specific chips for deep learning, NAM, etc.

REFERENCES



REFERENCES (1)

- [1] M. Riedel, 'Deep Learning using a Convolutional Neural Network', Ghent University, Invited YouTube Tutorial, Online: https://www.youtube.com/watch?v=gOL1_YIosYk&list=PLrmNhuZo9sgZUdaZ-f6OHK2yFW1kTS2qF
- [2] Official Web Page for Python Programming Language, Online: https://www.python.org/
- [3] François Chollet 'Deep Learning with Python', Book, ISBN 9781617294433, 384 pages, 2017, Online: https://www.manning.com/books/deep-learning-with-python
- [4] Tensorflow Deep Learning Framework, Online: https://www.tensorflow.org/
- [5] A Tour of Tensorflow, Online: https://arxiv.org/pdf/1610.01178.pdf
- [6] 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
- [7] Big Data Tips, 'What is a Tensor?', Online: http://www.big-data.tips/what-is-a-tensor
- [8] Keras Python Deep Learning Library, Online: https://keras.io/

REFERENCES (2)

- [9] M. Goetz, G. Cavallaro, T. Geraud, M. Book, M. Riedel, 'Parallel Computation of Component Trees on Distributed Memory Machines', Journal of Transactions on Parallel and Distributed Systems, 2018, to appear
- [10] J. Lange, G. Cavallaro, M. Goetz, E. Erlingsson, M. Riedel, 'The Influence of Sampling Methods on Pixel-Wise Hyperspectral Image Classification with 3D Convolutional Neural Networks', Proceedings of the IGARSS 2018 Conference, to appear
- [11] G. Cavallaro, N. Falco, M. Dalla Mura and J. A. Benediktsson, "Automatic Attribute Profiles," in IEEE Transactions on Image Processing, vol. 26, no. 4, pp. 1859-1872, April 2017, Online: http://ieeexplore.ieee.org/document/7842555/
- [12] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20(3), pp. 273–297, 1995
- [13] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., 'On Understanding Big Data Impacts in Remotely Sensed Image Classification using Support Vector Machine Methods', IEEE Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015, DOI: 10.1109/JSTARS.2015.2458855
- [14] M. Goetz, C. Bodenstein, M. Riedel, 'HPDBSCAN – Highly Parallel DBSCAN', in proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis (SC2015), Machine Learning in HPC Environments (MLHPC) Workshop, 2015, Online: http://dx.doi.org/10.1145/2834892.2834894
- [15] E. Erlingsson, G. Cavallaro, M. Riedel, H. Neukirchen, 'Scaling Support Vector Machines Towards Exascale Computing for Classification of Large-Scale High-Resolution Remote Sensing Images', Proceedings of the IGARSS 2018 Conference, to appear
- [16] Original (not JSC tuned) piSVM tool, Online: http://pisvm.sourceforge.net/

REFERENCES (3)

- [17] SoccerWatch.TV,
  Online: https://soccerwatch.tv/
- [18] Smart Medical Information Technology for HealthCare (SMITH),
  Online: http://www.smith.care/
- [19] JURECA HPC System @ Juelich Supercomputing Centre (JSC),
  Online: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/Configuration/Configuration_node.html;jsessionid=CF5F5276AE7CFF0B185DCA048A2D0010
- [20] A. Sergeev, M. Del Balso,'Horovod: fast and easy distributed deep learning in TensorFlow', 2018
  Online: https://arxiv.org/abs/1802.05799

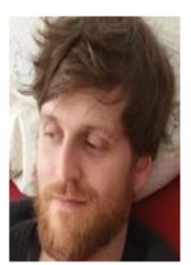
ACKNOWLEDGEMENTS

Previous & current members of the High Productivity Data Processing Research Group

PD Dr.
G. Cavallaro

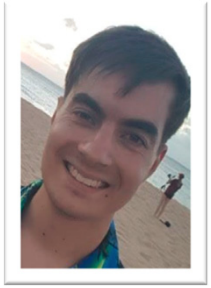
Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon

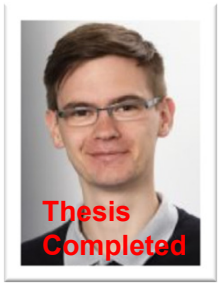


PhD Student
E. Erlingsson

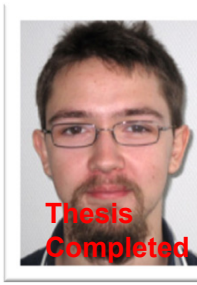
PhD Student
S. Bakarat



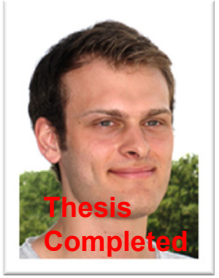
MSc Student
G.S. Guðmundsson
(Landsverkjun)



Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now Soccerwatch.tv)

THANKS

Talk shortly available under www.morrisriedel.de