

Supporting Software Engineering Practices in the Development of Data-Intensive HPC Applications with the JUML Framework



Prof. Dr.- Ing. Morris Riedel et al.

Head of Cross-Sectional Team Deep Learning

*2017 Int. Workshop on Software Engineering for High
Performance Computing in Computational and Data-Enabled
Science and Engineering @ Supercomputing 2017
12th November 2017, Denver, USA*



Jülich Supercomputing Centre

Cross-Sectional Team Deep Learning

**High Productivity Data Processing
(HPDP) Research Group**



**UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES**

FACULTY OF INDUSTRIAL ENGINEERING,
MECHANICAL ENGINEERING AND COMPUTER SCIENCE

Selected Software Engineering Challenges in HPC

Software is a 'side-effect' of scientific studies

- Science domain developers driven by **PHD student funds**
- Key goal is to **create high impact papers**, less on software
- Partially **unknown requirements** at software design time



[10] J. Segal and C. Morris,
'Developing Scientific
Software', *IEEE Software*
Vol 25(4), pp. 18-20, 2008

Different from traditional software development

- Rapid changes of underlying **complex hardware systems**
- Inherent **parallelism design and concurrency** effects
- Skills of **software engineering rarely applied** in scientific domains

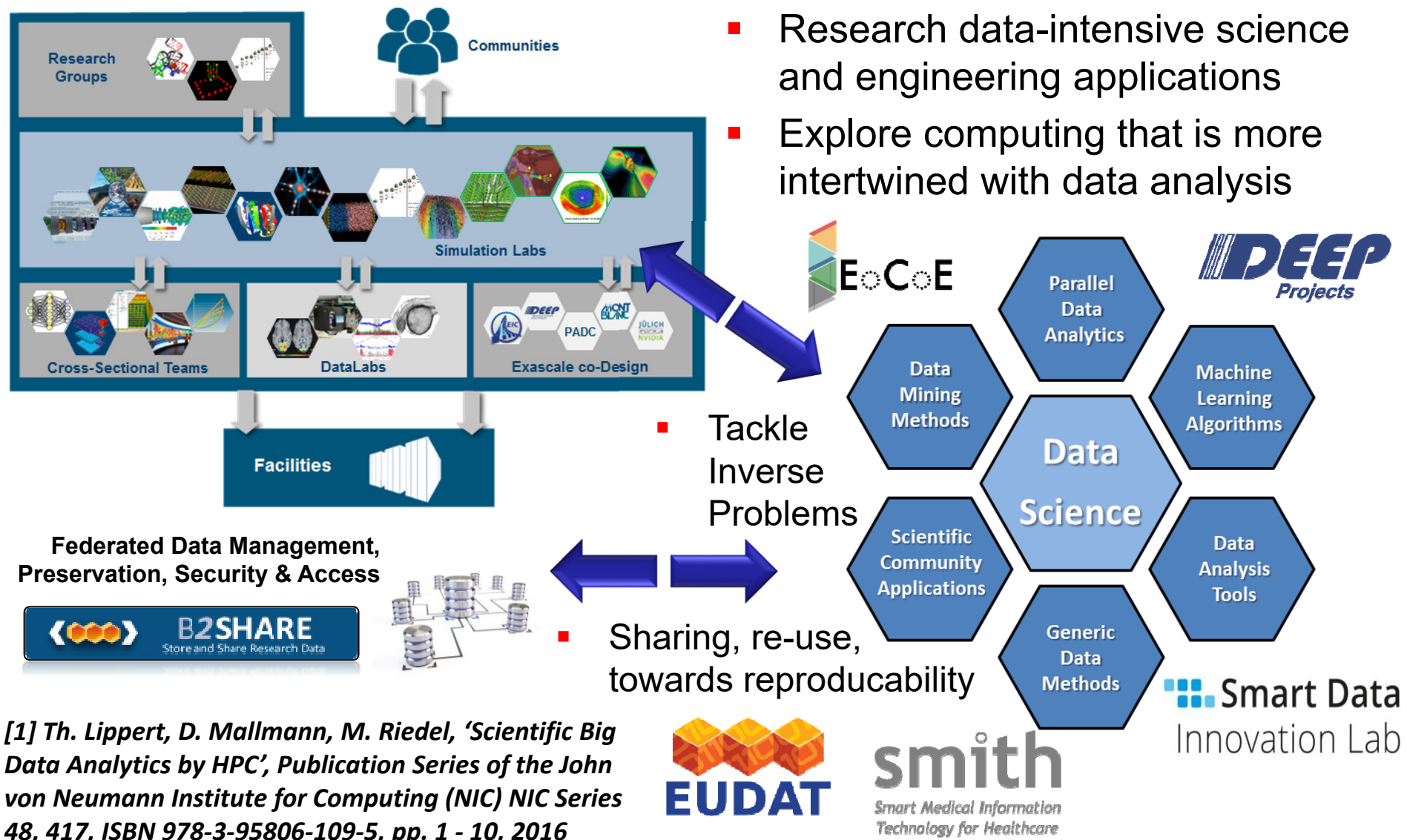


Establishing long-term tools & re-usability problems

- Different workflows and **software lifecycle driven by studies**
- Once PHD finished and moves, **lack of expert knowledge**
- Cost of **maintenance for software rarely in research grants**



Context Juelich Supercomputing Centre – Germany



Juelich Machine Learning Library (JUML) – Key Facts

Machine learning library for High Performance Computing (HPC)

- Leverages HPC key technologies: **parallel I/O** & **fast interconnects**
- Implements common **machine learning algorithms**: e.g. Gaussian Naive Bayes (GNB), K-Means, Artificial Neural Networks (ANN), etc.
- Includes a set of help functions, e.g. **data normalization**, **distributed sorting**
- Enables **usability** for non-HPC experts via **high-level APIs** for C++/Python
- Abstracts of technical complexities by simply **choosing CPUs or GPUs**
- Designed using **software engineering principles**, modularity, structured tests
- Available as **open source**


(extensions of gtest baseline test framework for different computational backends & test fixtures)

```
1 // original fixture class of Google Test
2 class FIXTURE_TEST {
3     FIXTURE_TEST() {
4         // setup code here
5     }
6 }
7
8 // forward definition inheriting from the fixture
9 #define INTECEPTOR_FORWARD_DEFINITION(FIXTURE) \
10 class FIXTURE##_Interceptor : public FIXTURE { \
11     protected: \
12     void test(); \
13 };
14
15 // per-backend test generator
16 #define TEST_BACKEND_F(FIXTURE, BACKEND) \
17 TEST_F(FIXTURE##_Interceptor) { \
18     // set backend
19     jum1::setBackend(BACKEND); \
20     // call the test case
21     this->test(); \
22 }
23
24 // definition of the main macro for all backends
25 #define TEST_ALL_F(FIXTURE) \
26 TEST_INTERCEPTOR_FORWARD_DEFINITION(FIXTURE) \
27 TEST_BACKEND_F(FIXTURE, jum1::Backend::CPU) \
28 #ifdef OTHER_BACKEND \
29 TEST_BACKEND_F(FIXTURE, OTHER_BACKEND) \
30 #endif \
31 void FIXTURE##_Interceptor::test
32
33
34 TEST_ALL_F(FIXTURE_TEST, FEATURE) {
35     // test code here
36 }
```

[6] JUML Open Source GitHub Repository

JUML – Software Engineering Perspective

Re-use of software libraries

- Arrayfire, MPI, HDF5
- cuda, opencl, python, numpy, mpi4py, swig (for python bindings), gtest
- Offers proper **building environment**, via **cmake**, i.e. CMakeLists.txt 

```
mkdir build && cd build && cmake .. && make
```

- Includes proper **source code documentation** via **doxygen** format

```
make doc
```

[8] Doxygen format

- Enables **test-driven developments** via **ctest** distributed test runner (**cmake**)

```
make vtest
```

```
ctest -R <TEST_NAME>
```

[7] Cmake Build, Test, and Packaging Tool,

```
#Project Name
PROJECT(JUML)

# Set minimum CMAKE version#
CMAKE_MINIMUM_REQUIRED(VERSION 2.8.11)

# ADD MPI
FIND_PACKAGE(MPI REQUIRED)
IF (MPI_FOUND)
    SET(CMAKE_C_COMPILER ${MPI_C_COMPILER})
    SET(CMAKE_CXX_COMPILER ${MPI_CXX_COMPILER})
ENDIF (MPI_FOUND)

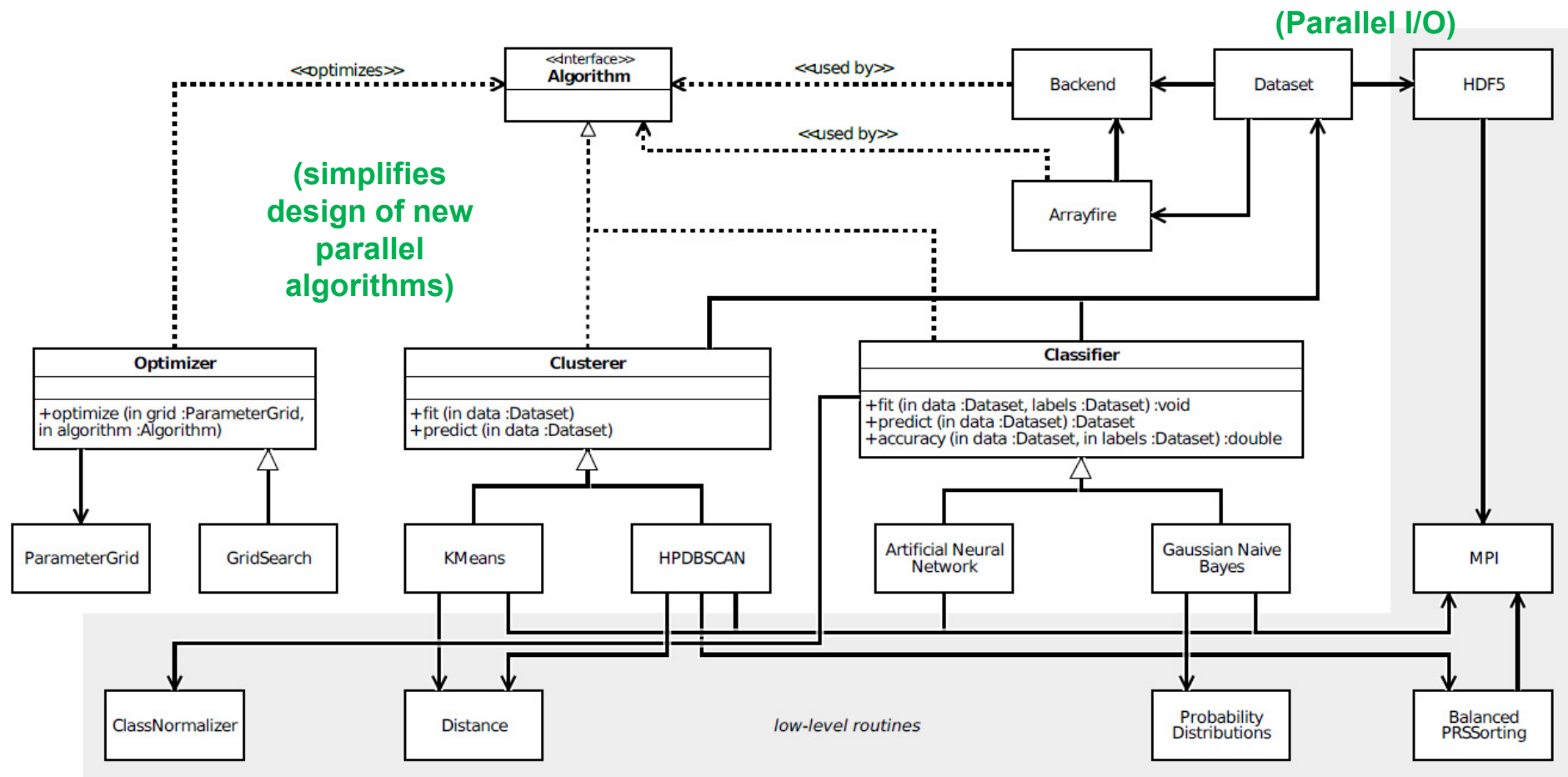
# Set Compiler
SET(CMAKE_C_COMPILER "mpicc")
SET(CMAKE_CXX_COMPILER "mpicxx")

# COMPILER FLAGS
SET(CMAKE_CXX_FLAGS "-std=c++11 -g")
FIND_PACKAGE(Threads REQUIRED)
FIND_PACKAGE(OpenMP REQUIRED)
IF (OPENMP_FOUND)
    SET(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} ${OpenMP_C_FLAGS}")
    SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} ${OpenMP_CXX_FLAGS}")
ENDIF (OPENMP_FOUND)

# ADD ARRAYFIRE
SET(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} "${CMAKE_CURRENT_SOURCE_DIR}/CMakeModules")
FIND_PACKAGE(ArrayFire REQUIRED)
IF (ArrayFire_FOUND)
    INCLUDE_DIRECTORIES(${ArrayFire_INCLUDE_DIRS})
    SET(AF_LIBS ${CMAKE_THREAD_LIBS_INIT} ${ArrayFire_LIBRARIES})
ENDIF (ArrayFire_FOUND)

# ADD HDF5
FIND_PACKAGE(HDF5 REQUIRED)
IF (HDF5_FOUND)
    INCLUDE_DIRECTORIES(${HDF5_INCLUDE_DIRS})
ENDIF (HDF5_FOUND)
```

JUML Architecture – Modularity & Extensibility

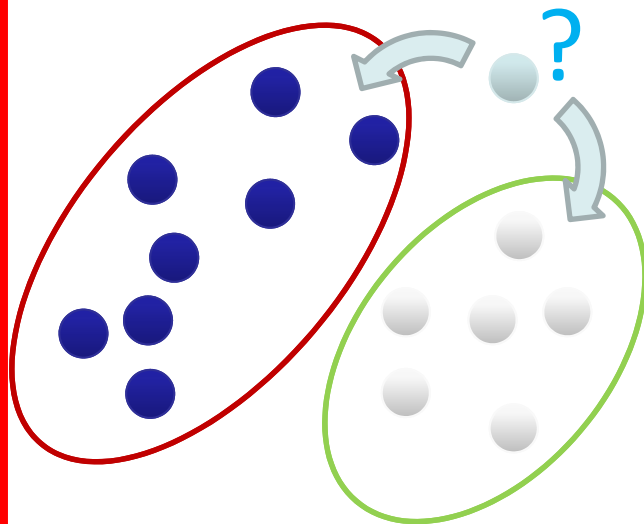


```
1  juml::Dataset data("/home/analysis_data.h5", "samples");
```

[5] M. Goetz & M. Riedel et al., 'Supporting Software Engineering Practices in the Development of Data-Intensive HPC Applications with the JUML Framework', CoDeSe Workshop, Supercomputing 2017, Denver, USA

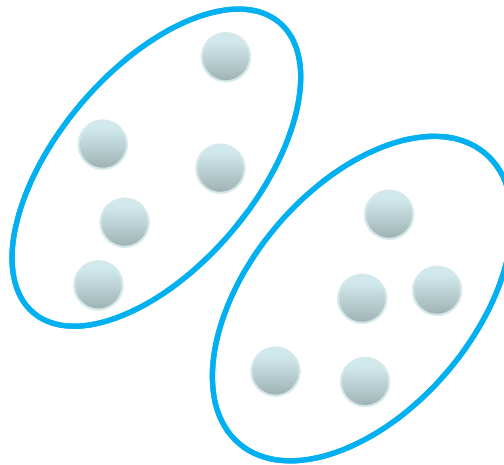
Classification Technique used in JUML Examples

Classification



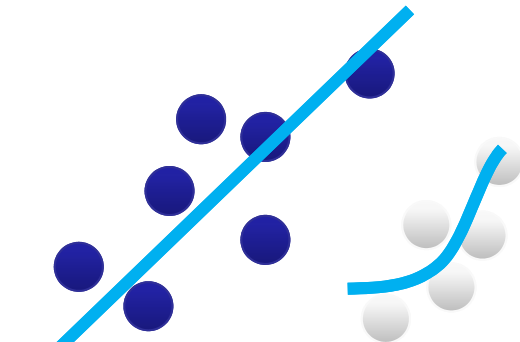
- Groups of data exist
- New data classified to existing groups

Clustering



- No groups of data exist
- Create groups from data close to each other

Regression



- Identify a line with a certain slope describing the data

[9] M. Riedel, 'Machine Learning Tutorial for Supervised Classification using SVMs', Tutorial, Barcelona Supercomputing Centre, 2016

**Parallel & Scalable Machine Learning Tutorial
PRACE PATC Training, Registration – Deadline:
2017-12-31 @ <https://events.prace-ri.eu/event/679/>**

Gaussian Naive Bayes (GNB) – Technique

‘Naive Bayes’ simple approach to prediction in classification

- Applies Bayes’ Theorem → naive assumption of **independence between every pair of features**
- Despite of simplicity works well often in practice

$$p(C_k | \mathbf{x}) = \frac{p(C_k) p(\mathbf{x} | C_k)}{p(\mathbf{x})}$$

(assign instance probabilities)

Gaussian Naive Bayes

- Assumes that the values associated with each class are **distributed according to a Gaussian distribution**
- Segment data by class, then **compute the mean and variance** for each class

$$p(C_k | x_1, \dots, x_n) = \frac{1}{Z} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

$$Z = p(\mathbf{x}) = \sum_k p(C_k) p(\mathbf{x} | C_k)$$

(conditional class distribution over class variable **C**; note **Pi** product notation)

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i | C_k).$$

(decision rule used for classification)

Computational complexity

- Fast training, **no coefficients need to be fitted by optimizations**
- Calculates only $P(\text{each class})$ and $P(\text{each class given different input } \mathbf{x})$

(gaussian naive bayes)

$$p(x = v | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

(**v** = data value, **u_k** = mean, **σ_k** = variance)

JUML Usage Example – GPUs or CPUs

Easy usability & resource specification

[6] JUML Open Source GitHub Repository

- E.g. using C++ API, specify resources: all available GPU nodes

```
1  #include <juml.h>
2  #include <mpi.h>
3
4  int main(void) {
5      juml::GaussianNaiveBayes gnb(
6          juml::Backend::GPU, // local gpu backend
7          MPI_COMM_WORLD // select global node allocation
8      );
9      return 0;
10 }
```

(goal: abstract
from low-level
parallelization
details)

- E.g. using Python API, specify resources: single node CPU

```
1  import juml
2  from mpi4py import MPI
3
4  gnb = juml.GaussianNaiveBayes(
5      juml.Backend.CPU, # local cpu backend
6      MPI.COMM_SELF # global node allocation
7  )
```

(goal: simplify
porting
small-scale
data analysis
code known
from
scikit-learn
or others
to HPC
systems)

Artificial Neural Network (ANN) – Technique

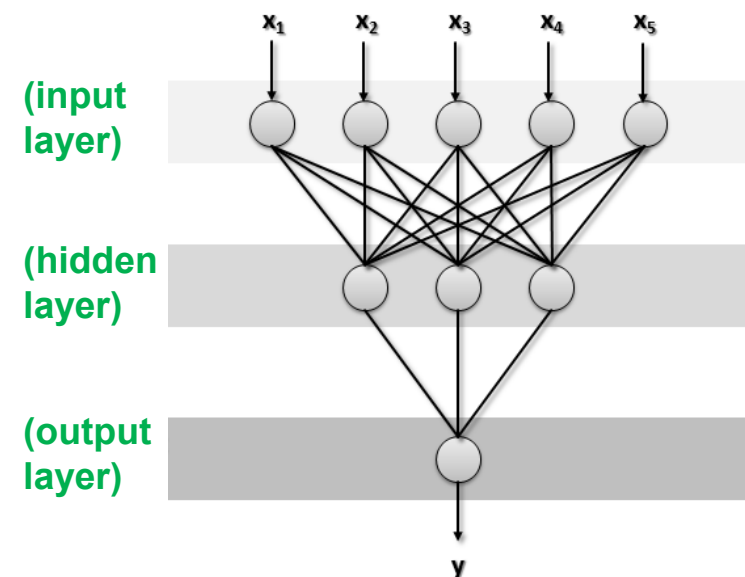
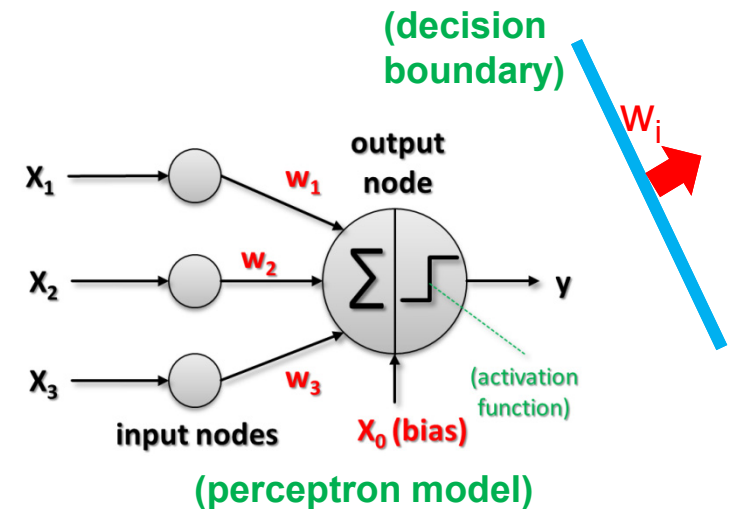
Key Building Block

- Perceptron learning model
- Simplest linear learning model
- Linearity in learned weights w_i
- One decision boundary

ANN – aka Multi-Layer Perceptron

- Enable the modelling of more complex relationships in the datasets
- May contain several intermediary layers
- E.g. 2-4 hidden layers with hidden nodes
- Use of activation function that can produce output values that are nonlinear in their input parameters

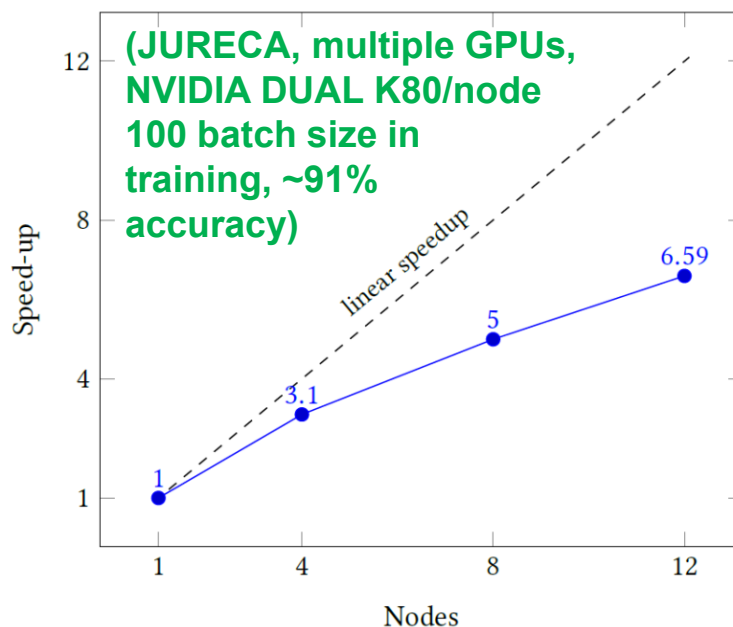
[4] M. Riedel, 'Introduction to Deep Learning with Convolutional Neural Networks & Applications', PRACE 2017 Spring School



Application Example – ANN Classification & Speed-Up

Remote Sensing Application Domain – City of Rome

- E.g. scientific case: automatically generate parts of street maps [today]
- E.g. scientific case: monitor urban planning efforts [over years/decades]
- Input: **labelled data**, hyperspectral images taken from satellites
- Output: classification of land cover type per pixel in **9 different classes**
- Architecture: **image pixels input**, 1000 hidden neurons, 9 output neurons



Buildings	Blocks	Roads
Vegetation	Trees	Bare soil
Ligh Train	Tower	Soil

(geometrical resolution of 1.3m and 55 different frequency bands, feature engineered using a self dual attribute profile)

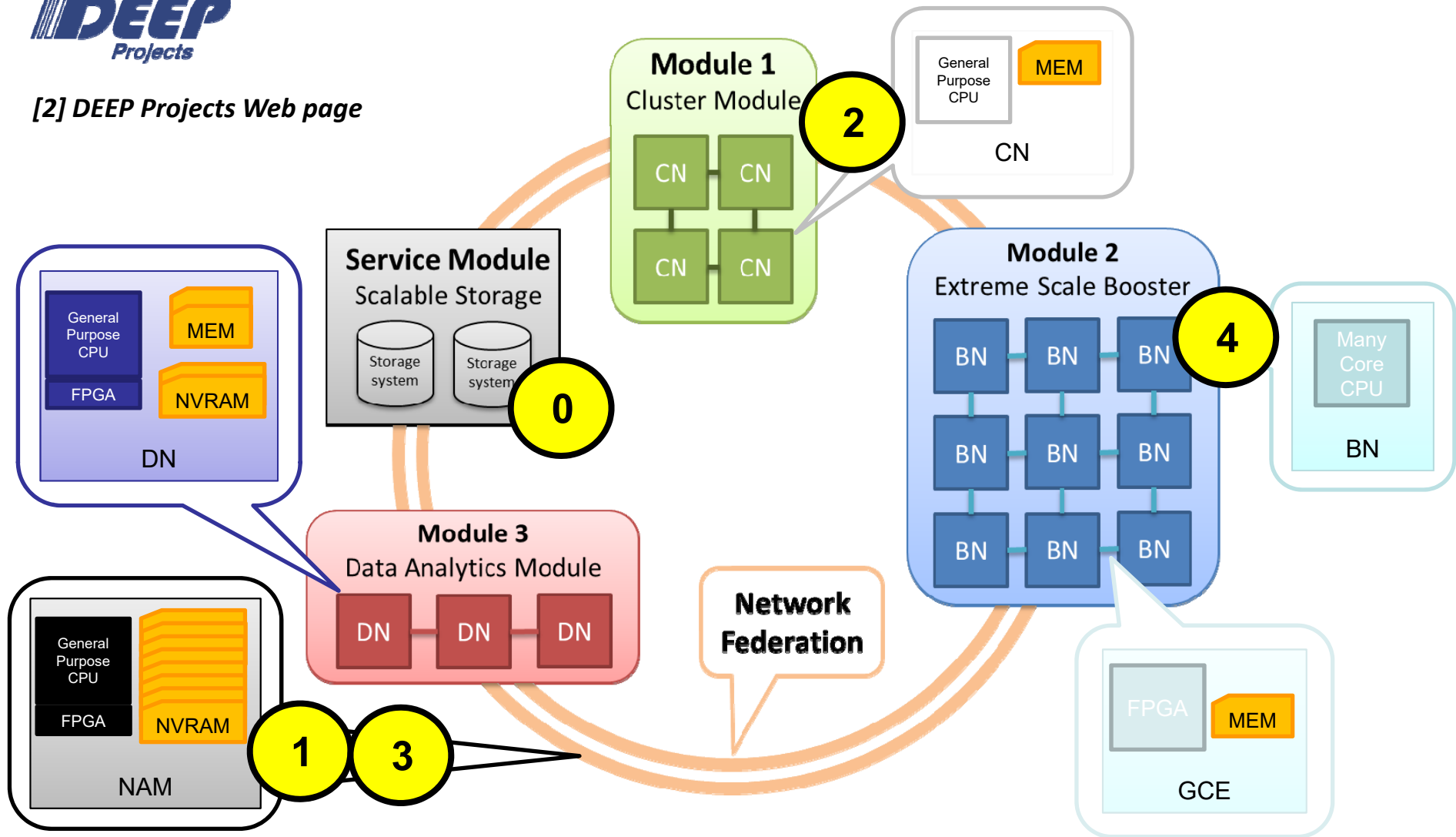
[3] Rome Image dataset



Research Modular Supercomputing Concept & JUML



[2] DEEP Projects Web page



Summary

Scientific Peer Review is essential to progress in the field



- Work in the field needs to be guided & steered by communities
- NIC Scientific Big Data Analytics (SBDA) first step (learn from HPC)
- Enabling reproducibility by uploading runs and datasets (increase quality)



John von Neumann - Institut für Computing



Explore benefits from software engineering in HPC

- Application-domain scientists can focus on science
- Enables long-term maintenance as community effort
- Generic JUMML framework design & re-usable components



Number of 'machine learning et al.' technologies incredible high

- (Less) open source & working versions available, often paper studies
- Evaluating approaches hard: HPC, map-reduce, Spark, SciDB, MaTex, ...
- Increasing number, uptake, and development of deep learning frameworks

Acknowledgments

The DEEP projects DEEP, DEEP-ER and DEEP-EST have received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no ICT-610476 and no ICT-287530 as well as the Horizon2020 funding framework under grand agreement no. 754304.



Members of the High Productivity Data Processing Research Group @ Jülich Supercomputing Centre & University of Iceland

Mohammad Shahbaz Memon

Matthias Richerzhagen

Ahmed Shiraz Memon

Christian Bodenstein

Gabriele Cavallaro

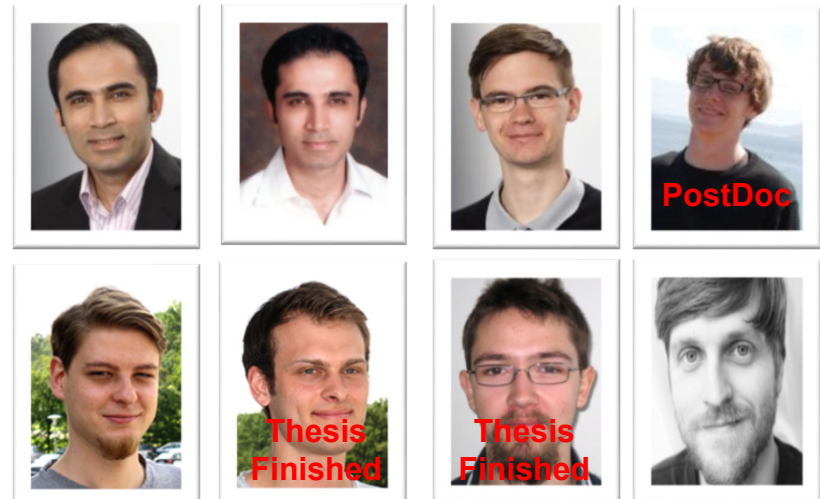
Ernir Erlingsson

Markus Goetz

Philipp Glock

Julius Lange

...



Selected References

- [1] Th. Lippert, D. Mallmann, M. Riedel, 'Scientific Big Data Analytics by HPC', Publication Series of the John von Neumann Institute for Computing (NIC) NIC Series 48, 417, ISBN 978-3-95806-109-5, pp. 1 - 10, 2016
- [2] Deep Projects Web Page,
Online: <http://www.deep-projects.eu>
- [3] Rome Dataset, B2SHARE,
Online: <http://hdl.handle.net/11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9>
- [4] M. Riedel, 'Introduction to Deep Learning with Convolutional Neural Networks & Applications', PRACE 2017 Spring School, YouTube Lecture,
Online: <https://www.youtube.com/watch?v=rqTFN2sxeTg>
- [5] M. Goetz, C. Bodenstein, M. Book, M. Riedel, 'Supporting Software Engineering Practices in the Development of Data-Intensive HPC Applications with the JUML Framework', CoDeSe Workshop, Supercomputing 2017, Denver, USA, DOI: [10.1145/3144763.3144765](https://doi.org/10.1145/3144763.3144765)
- [6] JUML Open Source GitHub Repository,
Online: <https://github.com/FZJ-JSC/JuML>
- [7] Cmake Build, Test, and Packaging Tool,
Online: <https://cmake.org/>
- [8] Doxygen documentation format,
Online: <http://www.stack.nl/~dimitri/doxygen/>
- [9] M. Riedel, 'Machine Learning Tutorial for Supervised Classification using Support Vector Machines', Tutorial, Barcelona Supercomputing Centre, 2016
Online: <https://www.bsc.es/education/training/other-training/machine-learning-tutorial-supervised-classification-using-support-vector-machines>
- [10] J. Segal and C. Morris, 'Developing Scientific Software', IEEE Software Vol 25(4), pp. 18-20, 2008



Appendix: Current Research Questionnaire for Studies



© 2011 Pearson Education, Inc. or its affiliate(s). All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without prior written permission from Pearson Education, Inc.