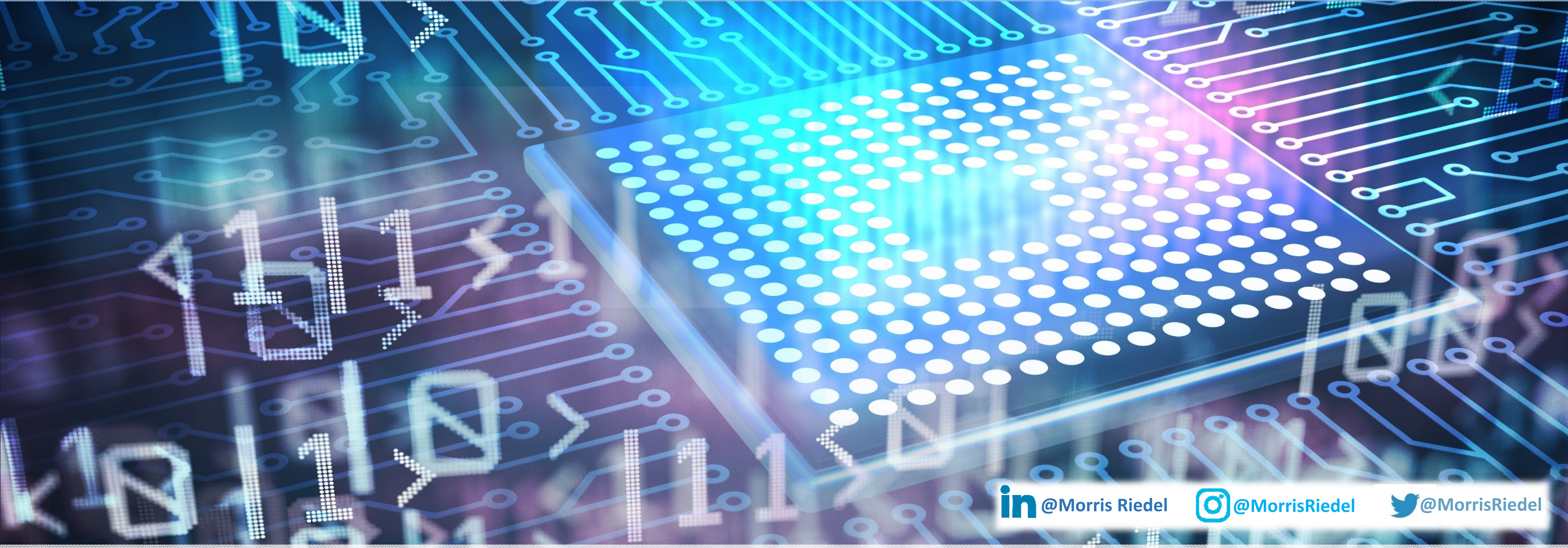
Demystifying Quantum Computing



Prof. Dr. - Ing. Morris Riedel

Head of Research Group 'High Productivity Data Processing'
Juelich Supercomputing Centre, Germany

Associated Professor
School of Engineering and Natural Sciences, University of Iceland

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

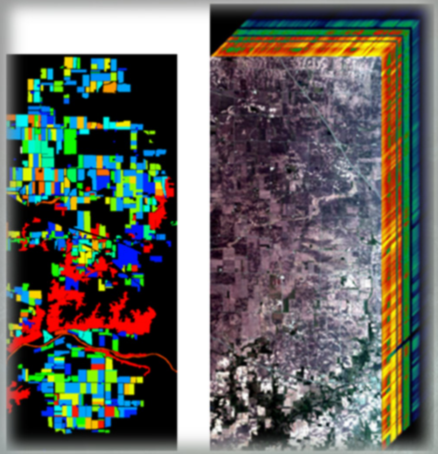
HELMHOLTZ AI

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

DEEP Projects

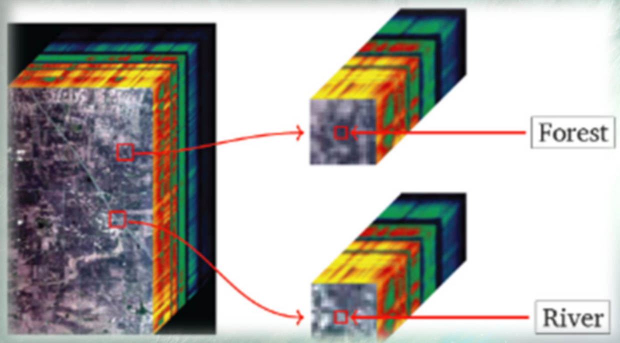
Motivational Practical Application

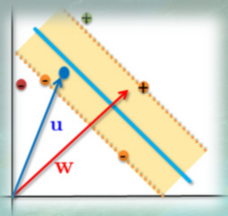
Remote Sensing using Machine Learning



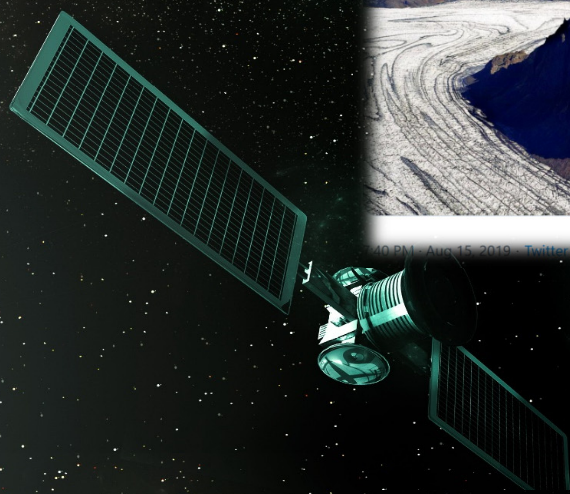
Training based on labeled data of land cover

Testing based on unseen data



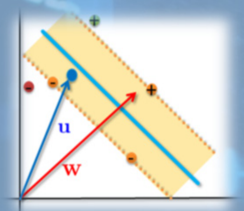


Using Support Vector Machines Machine Learning Models with High Performance Computing



[7] G. Cavallaro & M. Riedel et al.

Enabling Parallel & Scalable Machine & Deep Learning with High Performance Computing



Support Vector Machine Example using multi-core CPUs

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)

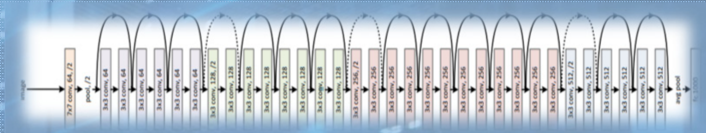
Scenario 'pre-processed data', 10xCV **parallel**: accuracy (min)

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)

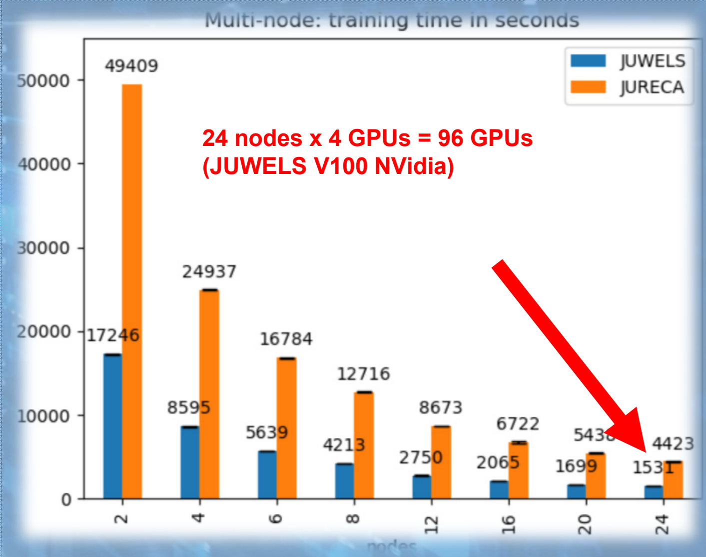
First Result: best parameter set from 14.41 min to 1.02 min
Second Result: all parameter sets from ~9 hours to ~35 min

[7] G. Cavallaro & M. Riedel et al.

Training Machine & Deep Learning Models often requires to solve a complex optimization problem



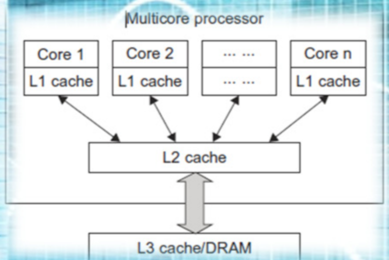
Deep Learning Example using Distributed Training on many-core GPUs via Horovod Framework & TensorFlow



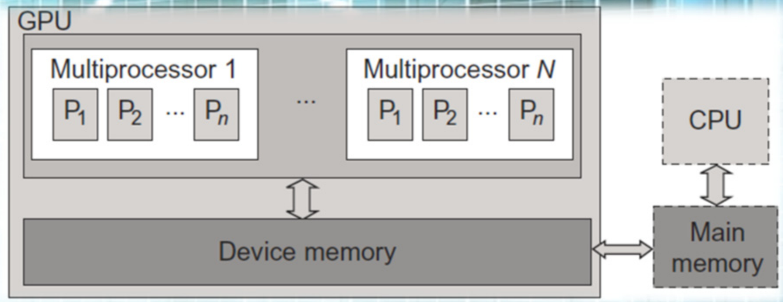
[8] R. Sedona, G. Cavallaro & M. Riedel et al.

High Performance Computing (HPC) uses 'Bits'

Information as '0' or '1' – Classical Computing Model

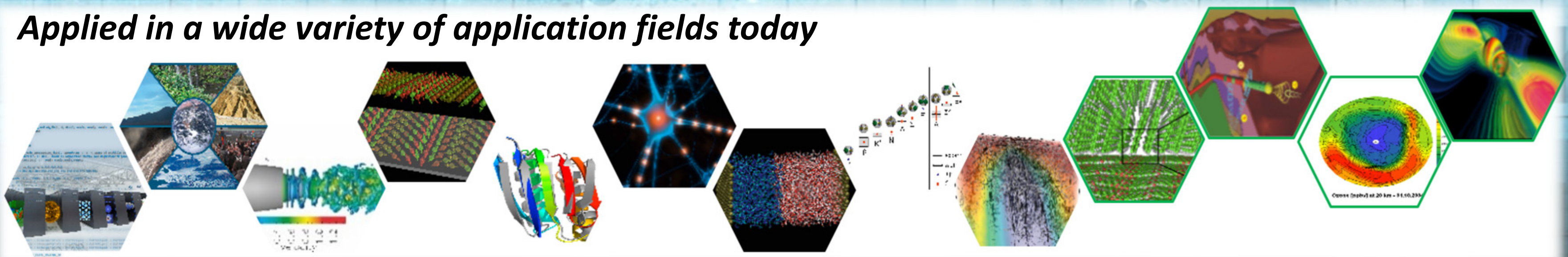


multi-core: high single thread performance



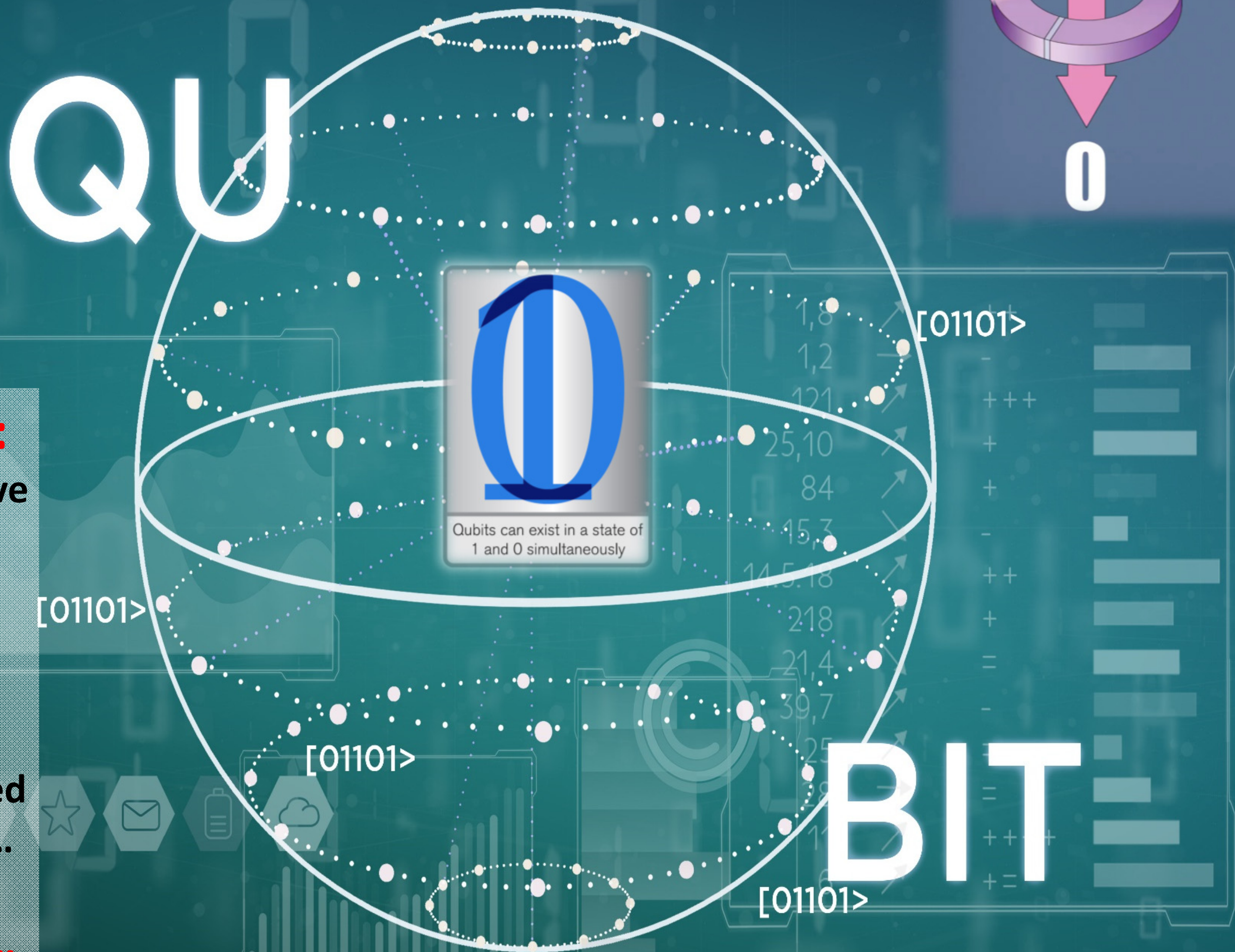
massive scalability through many-core moderate performance

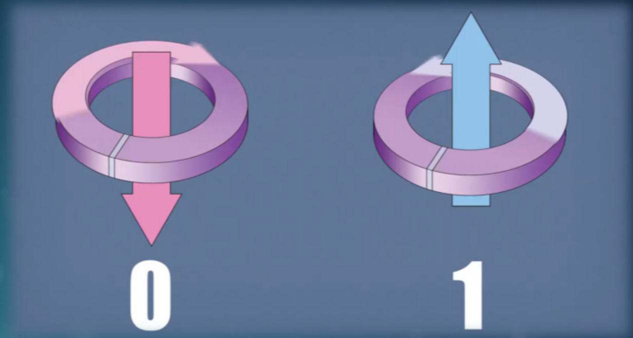
Applied in a wide variety of application fields today



Quantum Computers use 'Qubits'

Information as '0' or '1' or both simultaneously [01101>





Known as 'Superposition':

Origin in strange & counter-intuitive world of quantum mechanics

Manipulate many combinations of quantum bits at the same time

Example: with 1000 qubits, selected quantum computers can evaluate…

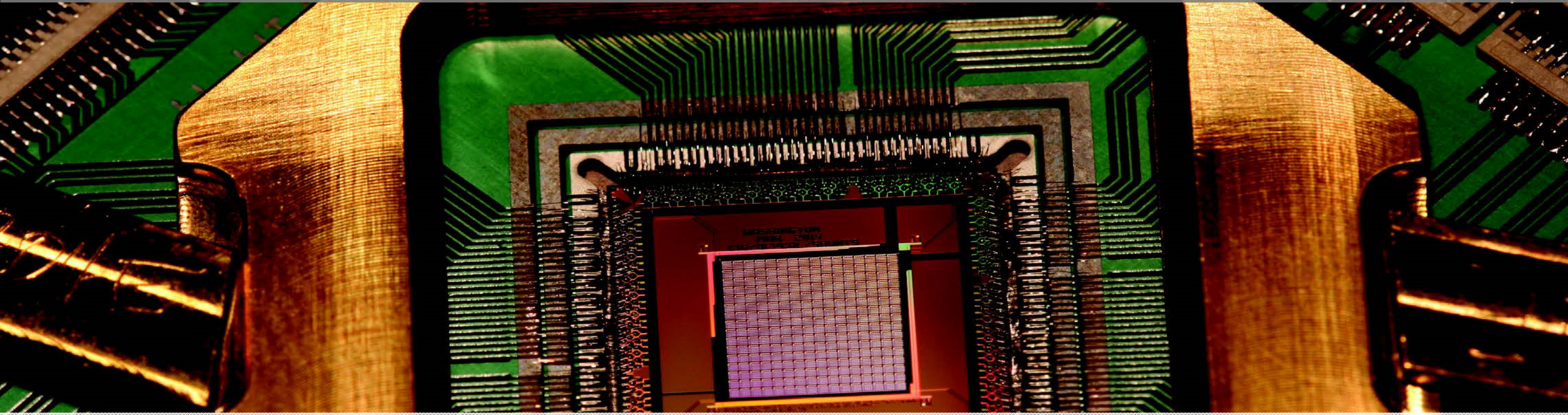
… 2^{1000} possible solutions …

… at the same time!

[1] D-Wave Report *[2] D-Wave YouTube*

Linking many Qubits together with Couplers

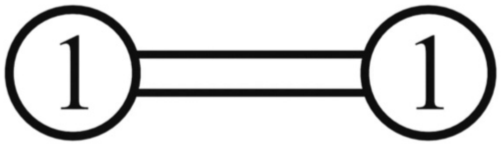
Enables Qubits to influence each other



Multi-qubit processor connects qubits to exchange information

Couplers define how qubits influence each other

Options: Qubits together in same state or neighbouring qubits in opposite state



Fabric of programmable quantum device =

many qubits + couplers + control circuitry to manage magnetic fields

[1] D-Wave Report [2] D-Wave YouTube

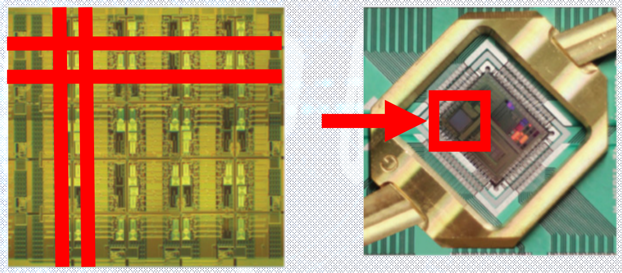
Coupling Qubits enables Entanglement

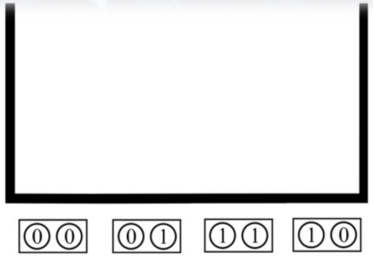
Known as 'Entanglement':

Quantum physics phenomena

Two qubits 'entangled' means they are considered as a single object (but with 4 states)

Beside 'superposition', the real power of quantum computing is using the quantum effects of entanglement & quantum tunneling to manipulate qubits simultanously





[1] D-Wave Report [2] D-Wave YouTube

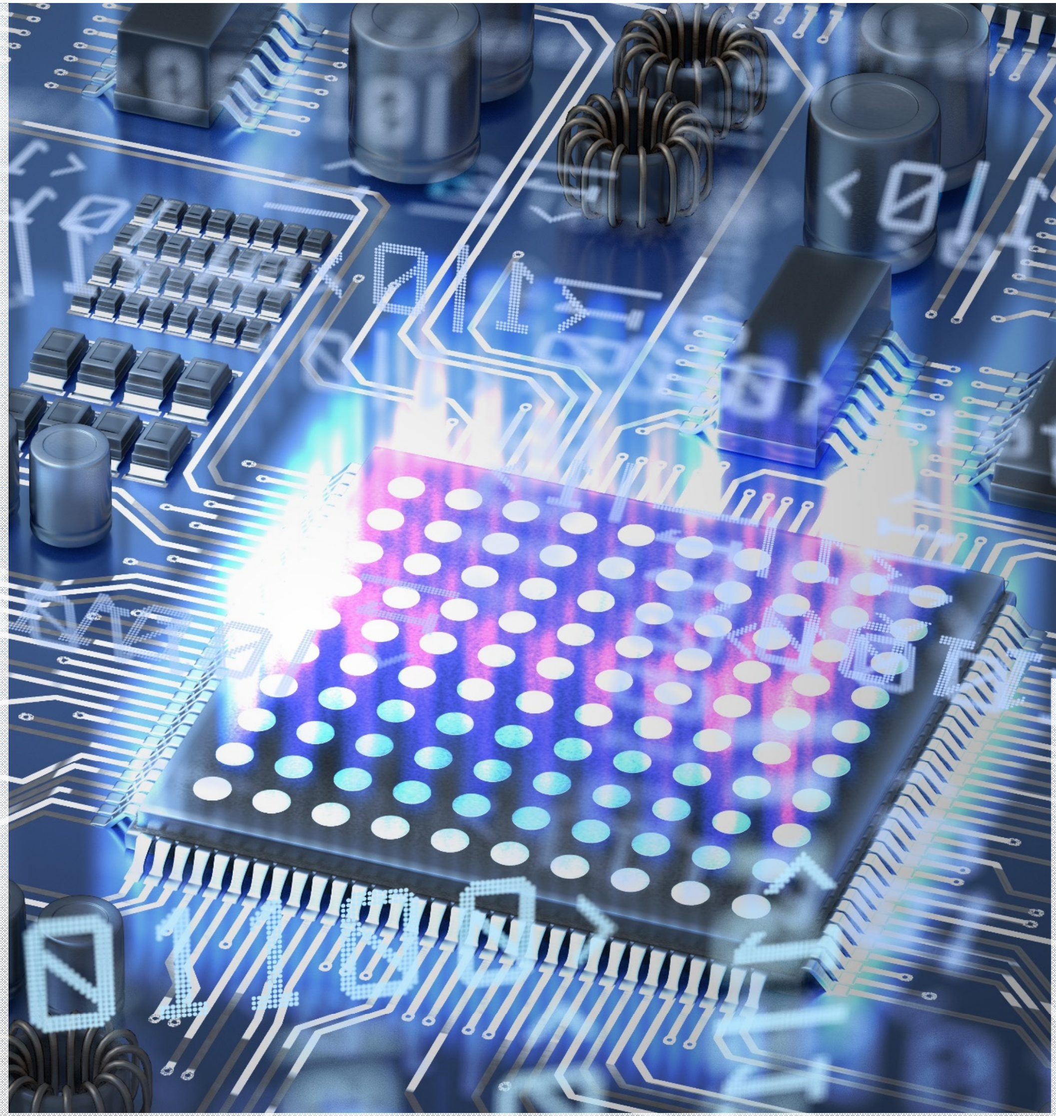
Challenge: Use quantum effects in computation requires processor in extreme isolated environments

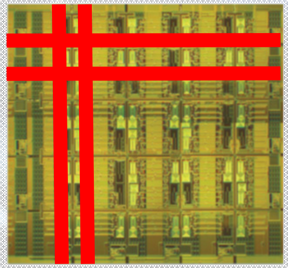
Internal computing environment needs a temperature **close to absolute zero**

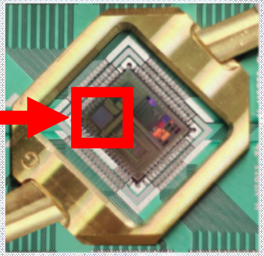
Enables an **'environment'** that is isolated from external magnetic fields, vibration, and external radio frequency signals

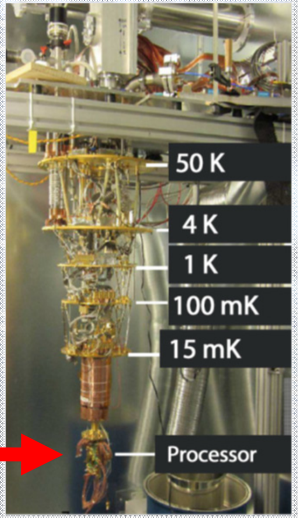
Only within this a quantum processor **'behaves quantum mechanically'**

The lower the temperate, the better the performance of the processor



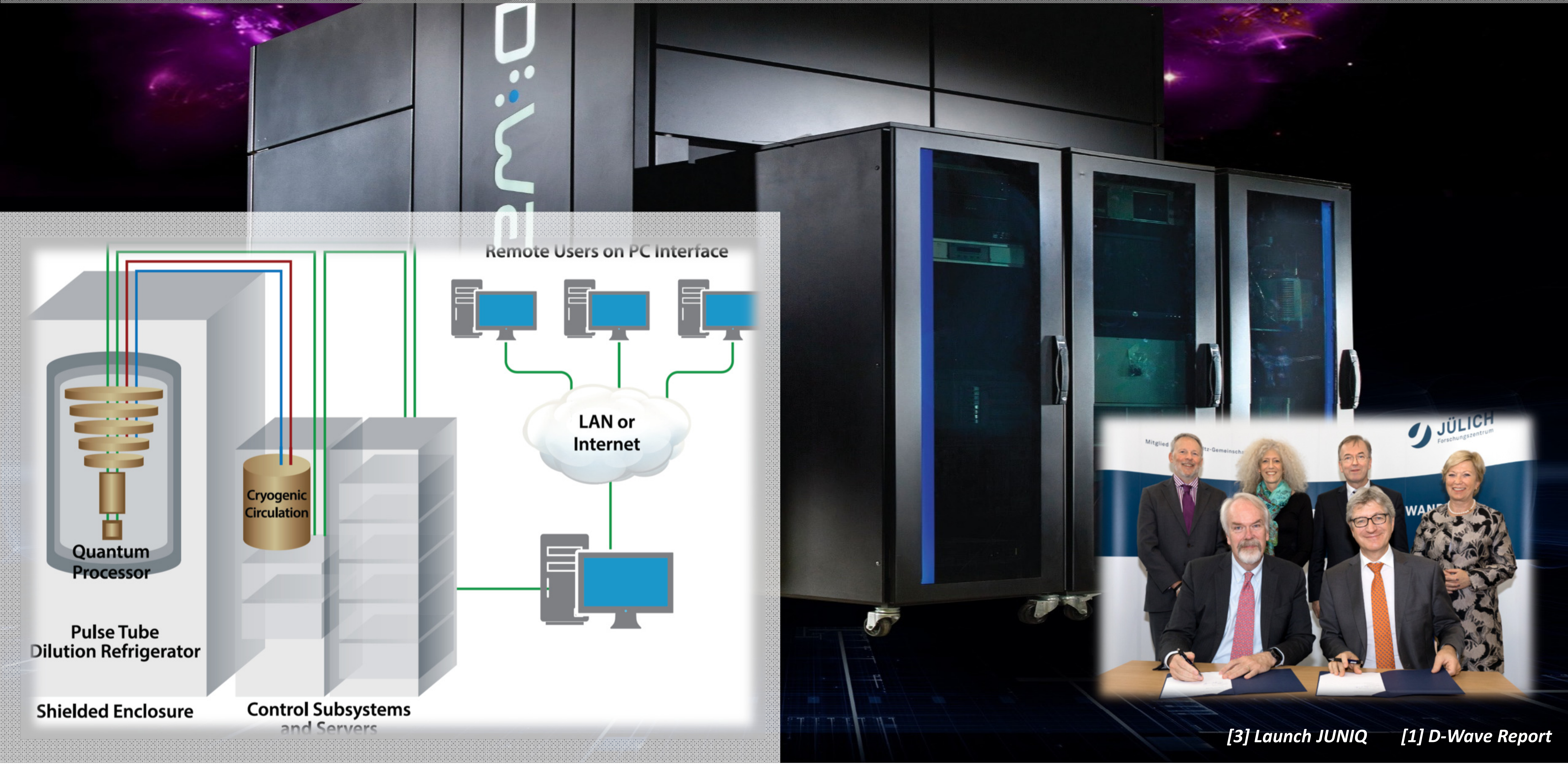






Quantum Computer Internal Environment

This Type of Quantum Computer Implements a Quantum Annealing Algorithm



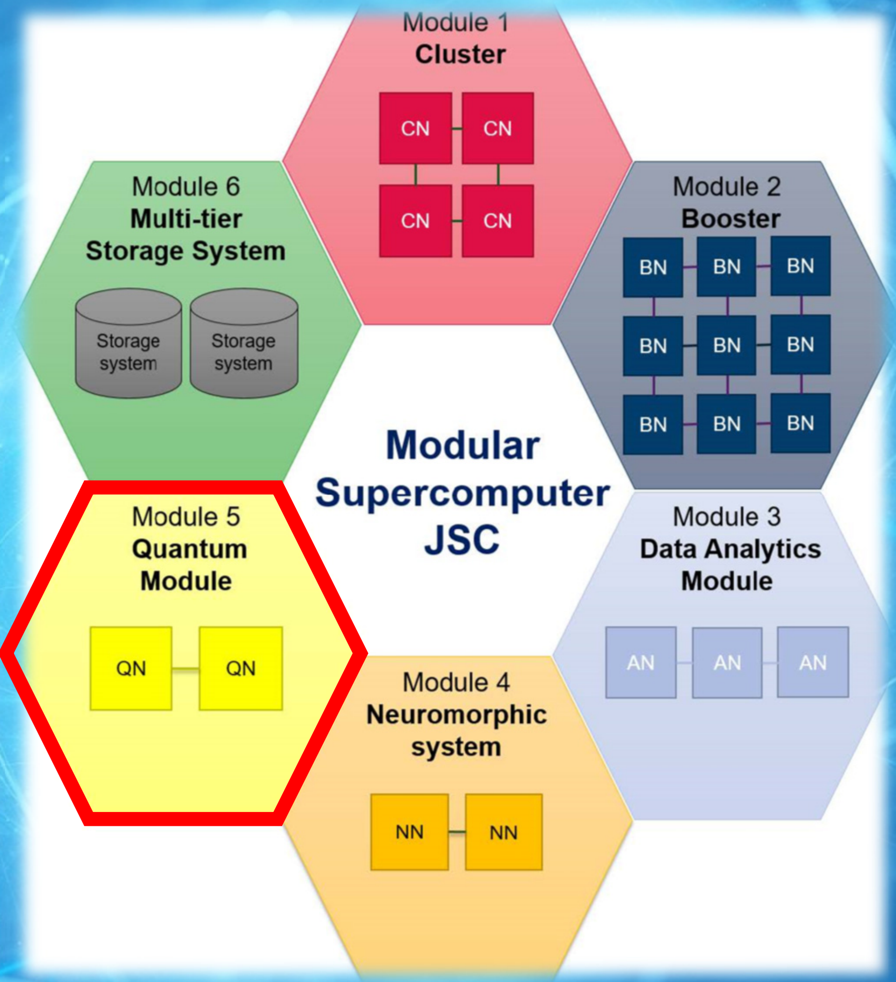
[3] Launch JUNIQ *[1] D-Wave Report*

JUNIQ – Juelich Unified Infrastructure for Quantum Computing

Building a European quantum computer user facility

Cloud access to simulators via a QC-PaaS
Jülich QC Simulator
JUQCS (< 49 qubits)
ATOS-QLM30 (< 31 qubits)

Cloud access to QC hardware via a QC-PaaS
(Non -) commercial gate-based QCs
European experimental qubit devices of any type, IBM, Google, RigettiComputing,..
Annealers
D-Wave quantum annealer
Fujitsu digital annealer
European experimental annealers



JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

[3] Launch JUNIQ *[4] DEEP Projects*

Many different Approaches exist for

Quantum Computing

Quantum Annealing Approach

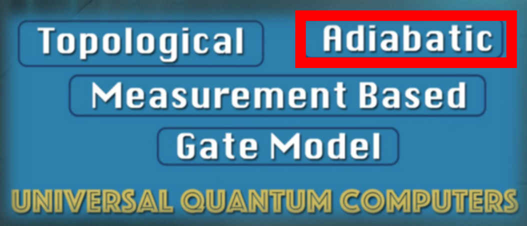
Uses intrinsic effects of Quantum Physics (QP) to help in optimization problems for real applications (but no evolution control)

Example: Quantum annealing algorithm solves problems by searching for the global minimum of a function ('hard in computing')

Setup a problem, then natural evolution of quantum states, and finally configuration at the end of evolution is one/some answer

Fundamentally different from classical computing built on logical operations

key difference



related, because of energy minimization

Gate-Model Approach

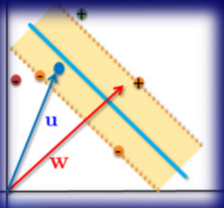
Much more ambitious to control and manipulate the evolution of quantum states over time

More difficult as quantum systems hard to work with

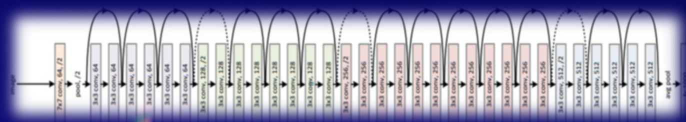
But enables to solve bigger problems, ~ 10 qubits only

Hard to let qubits working together coherently today

Relevance for Machine & Deep Learning – Solving Optimization Problems Fast



Training Machine & Deep Learning Models often requires to solve a complex optimization problem



'Minimum of a function'

Try to find the lowest point on a landscape of peaks and valleys

Every possible solution is mapped to coordinates on the landscape

Altitude of the landscape is 'energy' or 'cost' of solution per point

Goal: Find the lowest (global) point on the map & read the coordinates (i.e., optimal solution to many problems in science & engineering)

Relevance for Machine & Deep Learning – Key Difference to Classical Computing

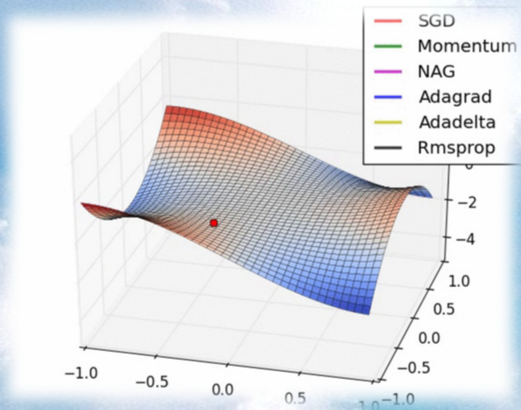
Properties of Quantum Physics known as 'Quantum Tunneling'

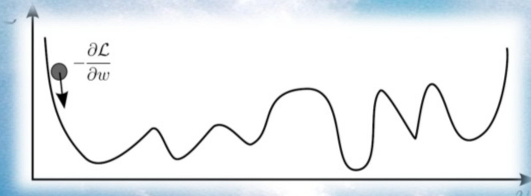
Explore landscape in ways that have never before been possible Like a layer of water that covers the entire landscape

Water as analogy for the probability that a given solution is returned

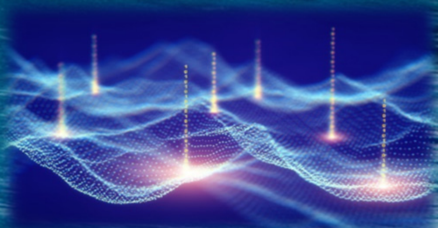
→ As well as running over the surface, water can tunnel trough the mountains as it looks for the lowest valley

Quantum Computations: 'water' or probability is pooled around lowest valleys – the more water there → the higher the probability of solution being returned as solution





$$\Delta w_{ji} = -\eta \frac{\partial \mathcal{L}}{\partial w_{ji}}$$



Classical Computer: A single travelor exploring the surface of a landscapeone point of a time (e.g., stochastic gradient descent)

[9] Optimizers *[1] D-Wave Report*

Quantum Computers use ‘Qubits‘

Quantum Annealing after Superposition: 0 or 1 [01101>

Known as ‘Collapse’:

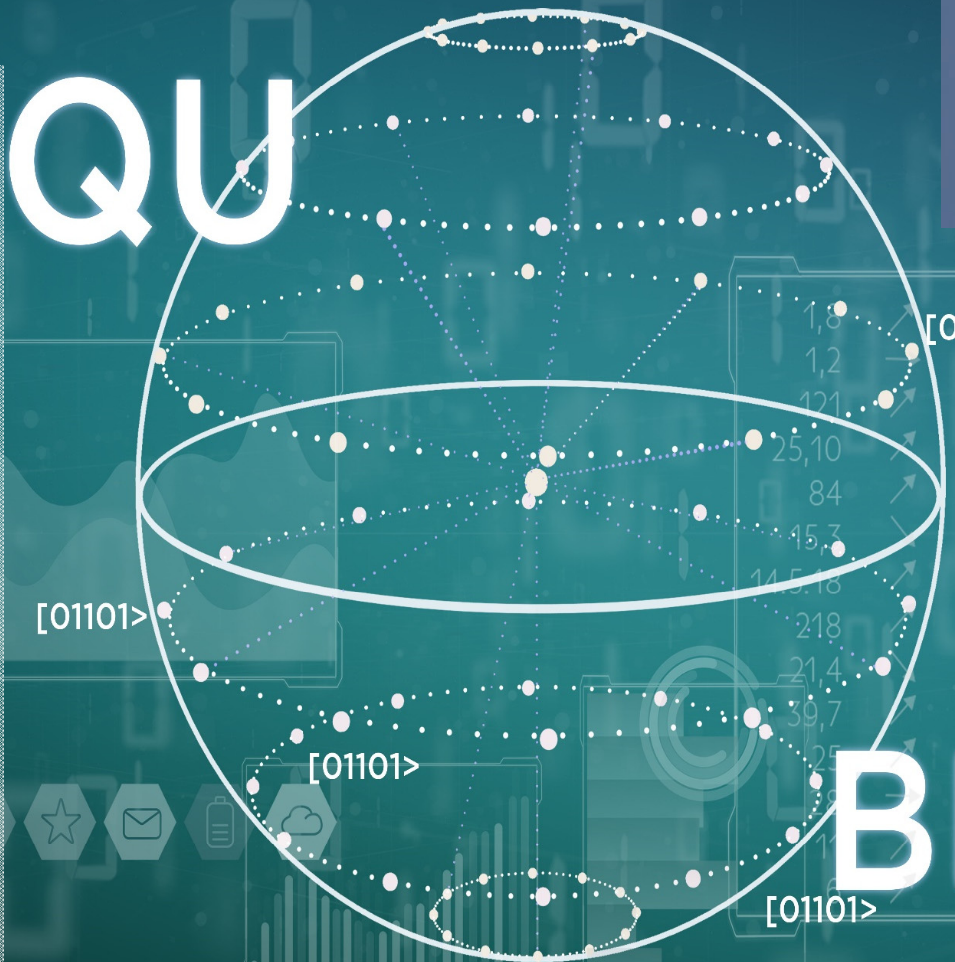
Quantum processor is built from a lattice of loops of metal niobium (each is one qubit when cooled)

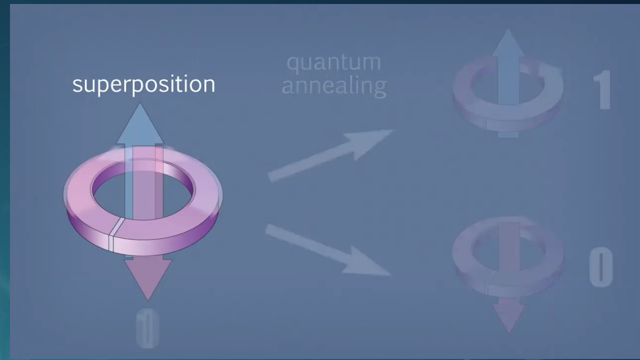
Niobium cooled down is a superconductor that exhibits quantum mechanical effects

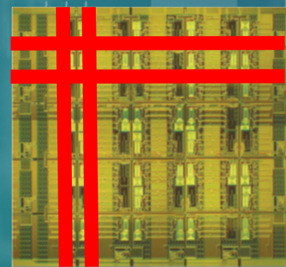
Circulating current clockwise or counter-clockwise, the qubit emits a magnetic field pointing up/down (i.e., logical 0 or 1)

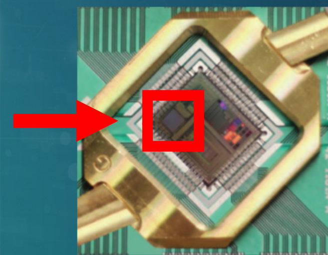
Quantum annealing: current flows clockwise & counter-clockwise simultaneously (‘superposition’)

Collapse at end of quantum annealing: qubit either a 0 or 1









[2] D-Wave YouTube

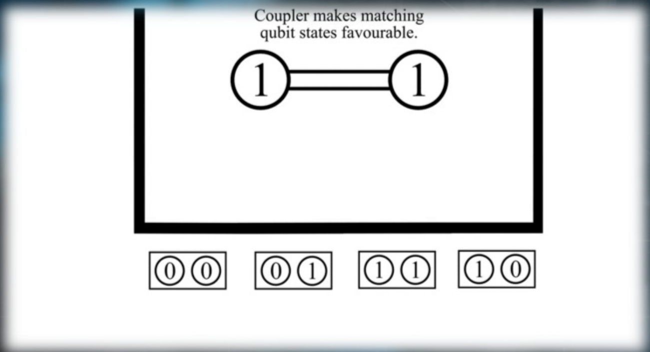
Programming #1 Control Probability of ‘Qubits‘ becoming 0 or 1 State

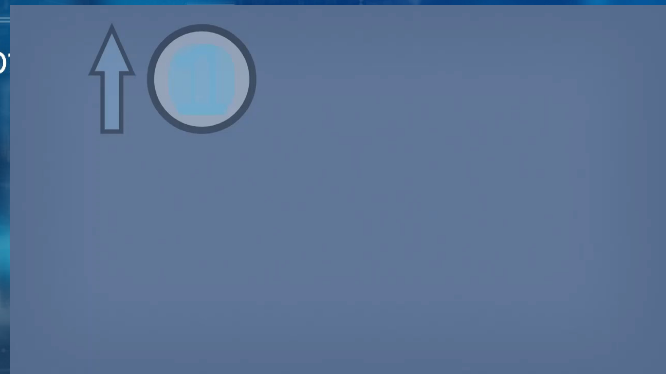
Energy Landscape with Quantum Annealing & Applying an external Magnetic Field called ‘Bias’

[2] D-Wave YouTube

Programming #2 Getting more control via Couplers & Entanglements

Users choose values for Biases of Qubits & Couplers (direction & strength) & Using 'Tunneling'





Disruptive Programming Approach

- ✓ **Each qubit can have a Bias applied to it**
- ✓ **Qubits can interact via the couplers**
- ✓ **Define a whole set of Biases & a whole set of couplings → defines an energy landscape**

Program a quantum computer based on quantum annealing to solve & find the minimum energy of energy landscape

[2] D-Wave YouTube

Quantum Computing Programming for Remote Sensing Application

Using a Software Development Kit (SDK) 'Ocean' – Installation & Configuration

D:Wave
The Quantum Computing Company

0. Setting Up the Access to the D-Wave 2000Q quantum computer

- Make a free account to run on the D-Wave through Leap
- Install Ocean Software with pip install dwave-ocean-sdk
- Configuring the D-Wave System as a Solver with dwave config create

1. Data Preparation

1.1 Load of the Python Modules

```
In [ ]: from utils import * # It contains functions for threat the data (I/O, encoding/decoding) and metrics for evaluations
```

1.2 Select the Dataset

In this notebook we consider the datasets of HyperLabelMe (i.e., a benchmark system for remote sensing image classification).

- It contains 43 image datasets, both multi- and hyperspectral
- For each one, training pairs (spectra and their labels) and test spectra are provided
- The test labels are not given. The predicted labels needs to be uploaded in HyperLabelMe which will return the accuracy

```
In [ ]: # Load the data
        # Ima40.txt can be downloaded after registering at [1]
        id_dataset='Im40'
        [X_train, Y_train, X_test]=dataread('input_datasets/hyperlabelme/'+id_dataset+'.txt')
```

16	Naples99	Landsat	200	200	7	2
17	Nayak - Middle Fork	Quickbird	1659	1331	4	10
18	Taku	Quickbird	1547	1675	4	3
19	Mulchatna	Quickbird	857	971	4	4
20	Kweth	Quickbird	367	362	4	4
21	Kol	Quickbird	1617	1660	4	5
22	Kitlope	Quickbird	1066	883	4	3
23	UCIsatimage	Landsat MSS	6435	36	1	6
24	Indian Pines (full)	AVIRIS	2166	614	220	58
25	Naples99 (full)	Landsat	400	400	7	2
26	Mexico	Landsat	360	512	2	2
27	Barrax	MERIS	1247	1153	13	2
28	France	MERIS	2399	2241	13	2
29	Abracos	MERIS	321	490	15	2
30	Ascension Island	MERIS	321	490	15	2
31	Azores	MERIS	321	493	15	2
32	Barcelona	MERIS	321	493	15	2
33	Capo Verde	MERIS	321	492	15	2
34	Longyearbyen	MERIS	321	493	15	2
35	Mongu	MERIS	321	489	15	2
36	Ouagadougou	MERIS	209	492	15	2
37	Zone13N	QuickBird	512	512	4	10
38	Zurich	Quickbird	828	889	4	7
39	Rome95	Landsat	200	200	7	2
40	Rome99	Landsat	200	200	7	2

Selecting 'simple' two class classification

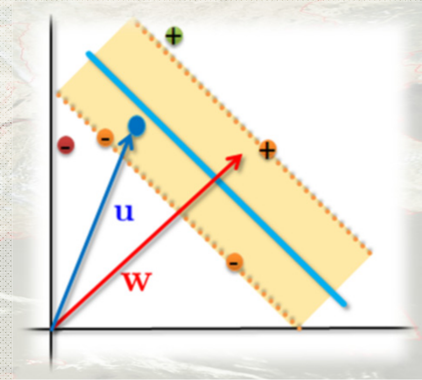
jupyter

[6] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen and M. Riedel

Quantum Computing Programming for Remote Sensing Application

Using a Software Development Kit (SDK) ‘Ocean‘ – Calibrating Quantum SVM

```
In [ ]: from sklearn.model_selection import train_test_split
        from sklearn import preprocessing

        # 10-fold Monte Carlo (or split-and-shuffle) cross-validation
        fold=10

        for i in range(0,fold):
            X_train_cal, X_val_cal, Y_train_cal, Y_val_cal = train_test_split(X_train,Y_train, test_size=0.94, random_state=i)

            # Pre-processing
            X_train_cal = preprocessing.scale(X_train_cal)
            X_val_cal = preprocessing.scale(X_val_cal)

            # Write the data
            write_samples(X_train_cal, Y_train_cal,'input_datasets/calibration/'+id_dataset+'/'+id_dataset+'calibtrain'+str(i))
            write_samples(X_val_cal, Y_val_cal,'input_datasets/calibration/'+id_dataset+'/'+id_dataset+'calibval'+str(i))

        print('Each training set includes '+str(X_train_cal.shape[0])+ ' samples')
        print('Each validation set includes '+str(X_val_cal.shape[0])+ ' samples')
```



$$E = \sum_{i \le j} a_i Q_{ij} a_j ,$$

[5] D. Willsch, M. Willsch, and K. Michielsen

```
In [ ]: from quantum_SVM import *

        # Hyperparameters
        B=[2,3,5,10]
        K=[2,3]
        xi=[0,1,5]
        gamma=[-1,0.125,0.25,0.5,1,2,4,8]
        n_experiments=len(B)*len(K)*len(xi)*len(gamma)

        hyperparameters=np.zeros([n_experiments,4], dtype=float)

        path_data_key='input_datasets/calibration/'+id_dataset+'/'
        data_key = id_dataset+'calibtrain'
        path_out='outputs/calibration/'+id_dataset+'/'

        trainacc=np.zeros([fold], dtype=float)
        trainauroc=np.zeros([fold], dtype=float)
        trainauprc=np.zeros([fold], dtype=float)
```

Previous work on Quantum SVM

[6] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen and M. Riedel

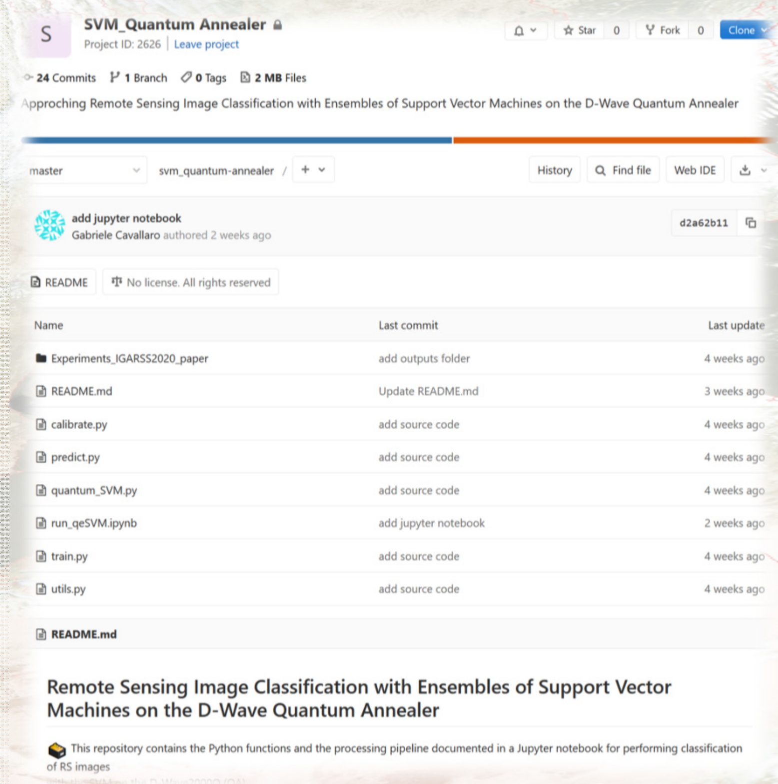
[2] D-Wave YouTube

Quantum Computing Programming for Remote Sensing Application

Using a Software Development Kit (SDK) 'Ocean' – Train & Test Quantum SVM

```
In [ ]: from quantum_SVM import *
        import numpy as np
        from utils import *
        from sklearn.model_selection import KFold
        from sklearn import preprocessing

        # Write the data
        experiments=1
        slice=40 # Number of samples to use for the training
        fold=int(len(X_train)/40)

        print(fold)

        for i in range(0,experiments):
            cv = KFold(n_splits=fold, random_state=i, shuffle=True)
            count=0
            for test_index, train_index in cv.split(X_train):
                #print("Train Index: ", len(train_index), "\n")

                X_train_slice, y_train_slice = X_train[train_index], Y_train[train_index]
                X_train_slice = preprocessing.scale(X_train_slice)

                X_test_slice, y_test_slice = X_train[test_index], Y_train[test_index]
                X_test_slice = preprocessing.scale(X_test_slice)
```

```
In [ ]: from quantum_SVM import *
        from sklearn import preprocessing

        # Pre-processing the test spectra
        X_test = preprocessing.scale(X_test)

        path_data_key='input_datasets/train/'+id_dataset+'/'
        data_key = id_dataset+'calibtrain'
        path_train_out='outputs/train/'+id_dataset+'/'
        path_test_out='outputs/test/'+id_dataset+'/'

        path_files=np.load(path_train_out+'trained_SVMs.npy')

        experiments=1
        slices=10
        scores=[]

        for j in range(0,experiments):
            for i in range(0,slices):
                scores.append(predict(path_data_key,path_files[i],X_test))


        avg_scores=np.zeros((scores[0].shape[0]))
        Y_predicted=np.zeros((scores[0].shape[0]),int)
```

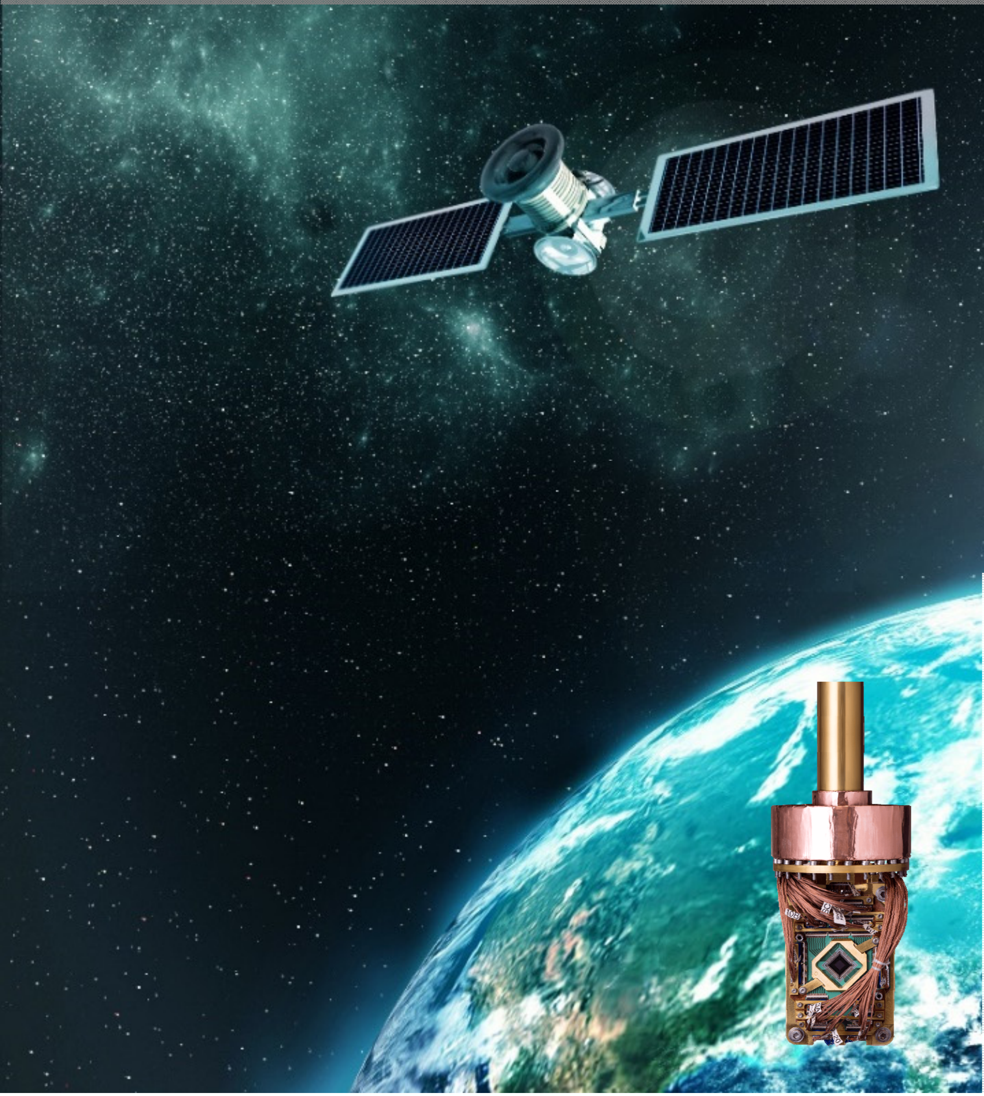


Github access to Jupyter Notebook public when/if paper accepted

[6] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen and M. Riedel

Summary General Approach with Quantum Annealing Works for Applications

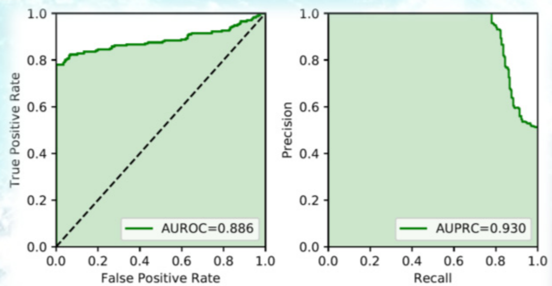
Challenge remains: Only small datasets today & Simple Classification Problems



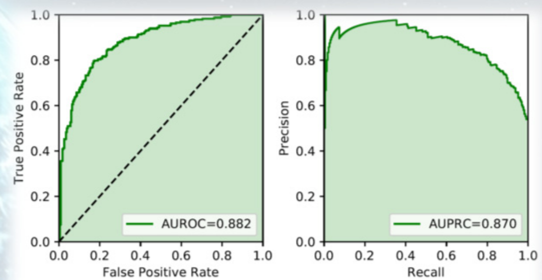
[6] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen and M. Riedel

ID	Sensor	Data points	Train Samples	Classes
Im16	Landsat	200×200×7	500	2
Im40	Landsat	200×200×7	500	2

ID dataset	OA	Kappa	z-score
16	94.26	0.89	134.71
40	78.90	0.57	47.21



(a) Dataset: Im16.



(b) Dataset: Im40.

Demystifying – Comparisons & Summary

TOPICS / FEATURES	Traditional HPC
Works for general-purpose Computing	✓
Ready for large datasets ('Big Data')	✓
Programming models broadly applied	✓
Fast Convergence & Time to Solution	✗
Smart Energy – Same Example 2500 kW	✗

TOPICS / FEATURES	Quantum Annealer
Works for general-purpose Computing	✗
Ready for large datasets ('Big Data')	✗
Programming models broadly applied	✗
Fast Convergence & Time to Solution	✓
Smart Energy – Same Example only 25 kW	✓

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

More Information & Details

UT|messan.

Visit us here @ UTMessan
Saturday @ Silfurberg A

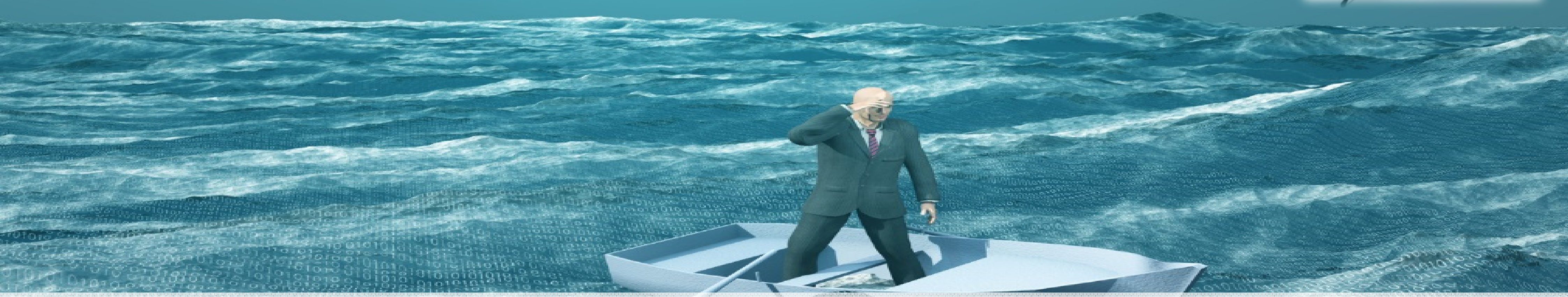
Relevant Yearly Courses at Háskóli Íslands

High Performance Computing (HPC)
Multi-Core; Many-Core & GPUs, Parallel Programming, Quantum Computing

Cloud Computing & Big Data
Deep & Machine Learning, High Throughput Computing, Apache Spark/Hadoop

Thanks & Talk available on www.morrisriedel.de/talks

Selected References



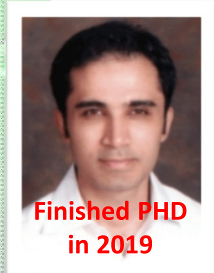
- [1] The D-Wave 2X Quantum Computer – Technology Overview, Online: https://www.dwavesys.com/sites/default/files/D-Wave%202X%20Tech%20Collateral_0915F.pdf
- [2] D-Wave Systems YouTube Channel, Online: https://www.youtube.com/user/dwavesystems
- [3] Launch of JUNIQ Press Release FZJ, Online: https://www.fz-juelich.de/SharedDocs/Pressemitteilungen/UK/EN/2019/2019-10-25-juniq.html
- [4] DEEP Projects, Online: https://www.deep-projects.eu/
- [5] D. Willsch, M. Willsch, H. De Raedt and K. Michielsen, "Support Vector Machines on the D-Wave Quantum Annealer'' 2019, Online: http://dx.doi.org/10.1016/j.cpc.2019.107006
- [6] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen and M. Riedel, "Approching Remote Sensing Image Classification with Ensembles of Support Vector Machines on the D-Wave Quantum Annealer" in the IEEE International Geoscience and Remote Sensing Symposium, 2020 (submitted)
- [7] 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, Online: https://www.researchgate.net/publication/282524415_On_Understanding_Big_Data_Impacts_in_Remotely_Sensed_Image_Classification_Using_Support_Vector_Machine_Methods
- [8] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, J.A. Benediktsson, 'Remote Sensing Big Data Classification with High Performance Distributed Deep Learning', MDPI Journal of Remote Sensing, to appear, online: https://www.researchgate.net/publication/338077024_Remote_Sensing_Big_Data_Classification_with_High_Performance_Distributed_Deep_Learning
- [9] Optimization Problems and Optimizers Visualization, Online: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Acknowledgements

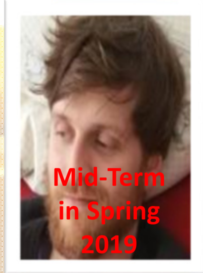
Former & Current Members of Joint Research Group

PD Dr.
G. Cavallaro

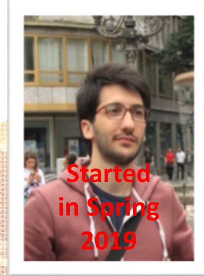
Senior PhD
Student A.S. Memon



Senior PhD
Student M.S. Memon



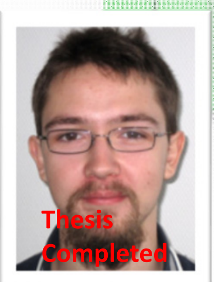
PhD Student
E. Erlingsson



PhD Student
S. Bakarat

PhD Student
R. Sedona

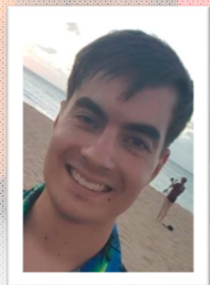
Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen
(now other division)

MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)



MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)

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

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

HELMHOLTZ AI

DEEP Projects