Practice and Experience in using Parallel and Scalable Machine Learning with Heterogenous Modular Supercomputing Architectures

Morris Riedel
Department of Computer Science
University of Iceland
Reykjavik, Iceland
morris@hi.is

Rocco Sedona
Jülich Supercomputing Centre
Forschungszentrum Jülich
Jülich, Germany
r.sedona@fz-juelich.de

Chadi Barakat
Jülich Supercomputing Centre
Forschungszentrum Jülich
Jülich, Germany
c.barakat@fz-juelich.de

Petur Einarsson
Department of Computer Science
University of Iceland
Reykjavik, Iceland
peturhelgi@gmail.com

Reza Hassanian
Department of Computer Science
University of Iceland
Reykjavik, Iceland
seh38@hi.is

Gabriele Cavallaro
Jülich Supercomputing Centre
Forschungszentrum Jülich
Jülich, Germany
g.cavallaro@fz-juelich.de

Matthias Book
Department of Computer Science
University of Iceland
Reykjavik, Iceland
book@hi.is

Helmut Neukirchen
Department of Computer Science
University of Iceland
Reykjavik, Iceland
helmut@hi.is

Andreas Lintermann
Jülich Supercomputing Centre
Forschungszentrum Jülich
Jülich, Germany
a.lintermann@fz-juelich.de

***Abstract*—We observe a continuously increased use of Deep Learning (DL) as a specific type of Machine Learning (ML) for data-intensive problems (i.e., 'big data') that requires powerful computing resources with equally increasing performance. Consequently, innovative heterogeneous High-Performance Computing (HPC) systems based on multi-core CPUs and many-core GPUs require an architectural design that addresses end user communities' requirements that take advantage of ML and DL. Still the workloads of end user communities of the simulation sciences (e.g., using numerical methods based on known physical laws) needs to be equally supported in those architectures. This paper offers insights into the Modular Supercomputer Architecture (MSA) developed in the Dynamic Exascale Entry Platform (DEEP) series of projects to address the requirements of both simulation sciences and data-intensive sciences such as High Performance Data Analytics (HPDA). It shares insights into implementing the MSA in the Jülich Supercomputing Centre (JSC) hosting Europe No. 1 Supercomputer Jülich Wizard for European Leadership Science (JUWELS). We augment the technical findings with experience and lessons learned from two application communities case studies (i.e., remote sensing and health sciences) using the MSA with JUWELS and the DEEP systems in practice. Thus, the paper provides details into specific MSA design elements that enable significant performance improvements of ML and DL algorithms. While this paper focuses on MSA-based HPC systems and application experience, we are not losing sight of advances in Cloud Computing (CC) and Quantum Computing (QC) relevant for ML and DL.**

***Index Terms*—High performance computing, cloud computing, quantum computing, machine learning, deep learning, parallel and distributed algorithms, remote sensing, health sciences, modular supercomputer architecture**

This work was performed in the Center of Excellence (CoE) Research on AI- and Simulation-Based Engineering at Exascale (RAISE), the Euro CC, and DEEP-EST projects receiving funding from EU's Horizon 2020 Research and Innovation Framework Programme under the grant agreement no. 951733, no. 951740 and no. 754304 respectively.

I. Introduction

Today, an academically-driven supercomputing centre's (e.g., Juelich Supercomputing Centre[1], Barcelona Supercomputing Centre[2], or Finish IT Center for Science CSC[3]) application portfolio is highly multidisciplinary, raising diverse requirements for a HPC architecture that enables research for a wide variety of end-user communities [1]. Examples include but are not limited to astrophysics, computational biology and biophysics, chemistry, earth and environment, plasma physics, computational soft matter, fluid dynamics, elementary particle physics, computer science and numerical mathematics, condensed matter, and materials science. Not only the research approaches in these communities are diverse, but also the way how they employ scalable algorithms, numerical methods, and parallelisation strategies. Many of these are *'traditional HPC applications'* (i.e., modeling and simulation sciences) that use iterative methods and rely heavily on a small number of

[1] https://www.fz-juelich.de/ias/jsc/EN/Home/home_node.html
[2] https://www.bsc.es/
[3] https://www.csc.fi/en/csc

numerical algorithmic classes that operate on relatively small to moderate-sized data sets and accrue very high numbers of floating-point operations across iterations. But we observe that the complexity (e.g., using CPU in conjunction with GPUs) and memory requirements (e.g., using complex memory hierarchies) of HPC codes of those applications increases, leading to a dissonance with these traditional HPC system workloads.

More recently, new user communities add to the above mentioned diversity in the sense of using the HPC systems with HPDA using ML and DL in conjunction with containers [2] and interactive supercomputing (e.g., via Jupyter[4] notebooks) [3]. Those workloads (e.g., remote sensing or health sciences) are rapidly emerging and require a change in HPC systems architecture. They exhibit less arithmetic intensity and instead require additional classes of parallel and scalable algorithms to work well (e.g., DL networks with interconnections of GPUs to scale to extreme scale). Some end-user communities (e.g., neurosciences and earth sciences) make intertwined use of both traditional simulation sciences-based HPC and HPDA simultaneously, leading to the term *'scientific big data analytics'* [4], [5]. This confluence of HPC and HPDA is also recognized by processor chip vendors such as Intel[5] or NVIDIA[6] and collaboration between academic centers and such industry partners is key to achieve extreme scale.

CC is another computing approach that is highly relevant for ML and DL, making parallel and distributed computing more straightforward to use (e.g., via containers or Jupyter notebooks) than traditional rather complex HPC systems. Remote sensing researchers and health scientists often take advantage of Apache open-source tools with parallel and distributed algorithms (e.g., map-reduce [6] as a specific form of divide and conquer approach) based on Spark [7] or the larger Hadoop ecosystem [8]). Also, inherent in many ML and DL approaches are optimization techniques while many of them are fast solvable by QCs [9] that represent the most disruptive type of computing today. Despite being in its infancy, Quantum Annealer (QA)s are specific forms of QC used by remote sensing and health researchers to search for solutions to optimization problems already [10], [11].

This paper reveals practice and experience using the heterogenous MSA that has been co-designed by 15 applications during the course of the DEEP[7] series of projects. The goal of the MSA is to address traditional HPC and more recent HPDA workloads while we particularly focus in this paper on heterogenous ML and DL workloads. We offer lessons learned implementing our production implementation of the MSA in HPC systems such as the DEEP cluster[8] or Europe No. 1 Supercomputer JUWELS[9]. Although HPC drives the MSA, certain aspects in this paper will also provide information about our recent QC MSA module briefly discussing our early knowledge of using D-Wave Systems[10] quantum annealers for ML. Also, given that service offerings from commercial cloud vendors are relevant for ML and DL research, we provide interoperability links in the MSA context where possible (e.g. using containers). To provide a reasonable focus in this paper, we concentrate on two concrete end-user communities case studies using the MSA with different ML and DL applications from remote sensing and health sciences.

The remainder of the paper is structured as follows. After the scene is set in Section I, Section II introduces the MSA heterogenous overall design. Section III then reveals lessons learned from using this MSA with concrete application case studies from the remote sensing end user communities. Section IV informs about practice and experience in health science applications using the MSA. While related work is reviewed in Section V, our paper ends with some concluding remarks.

II. Heterogenous Modular Supercomputing Architecture

Over the last years, we observe a continuously increasing complexity of computations with various concurrently executed functionalities in HPC and HPDA application workloads. To support these workloads requires heterogeneous hardware architecture designs under the constraints of minimal energy consumption, minimal time to solution, and minimal general systems cost. As shown in Fig. 1, the MSA [1] strives to address these constraints by providing a modular design to break with the tradition of replicating many identical (potentially heterogeneous) compute nodes and integrate the heterogeneous computing resources instead at the system level. That design connects computing modules with different hardware and performance characteristics to create a single heterogeneous system seamlessly integrating a storage module (i.e., multi-tier storage system). While each module is a parallel clustered system (i.e., of potentially large sizes), a high-performance federated network connects the module-specific interconnects.

Our MSA brings substantial benefits for heterogeneous application workloads because each application and its parts can be run on an exactly matching system, improving time to solution and energy use. An MSA implementation is ideal for a supercomputer centre infrastructure such as JSC in Germany (e.g., JUWELS) or CSC in Finland (e.g., EuroHPC LUMI[11]) running heterogeneous application mixes. One of the MSA advantages is the valuable flexibility for system operators, allowing the set of modules and their size to be tailored to the computing centre actual application portfolio. That includes a design approach of gradually integrating also innovative modules with disruptive technologies such as emerging neuromorphic or quantum devices.

The MSA has many benefits resulting from more than a decade of experience gathered at the Juelich Supercomputing

[4] https://jupyter.org/

[5] https://www.intel.com/content/www/us/en/high-performance-computing/high-performance-data-analytics.html

[6] https://www.nvidia.com/en-us/deep-learning-ai/solutions/data-analytics/

[7] https://www.deep-projects.eu/

[8] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST_node.html

[9] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUWELS/JUWELS_node.html

[10] https://www.dwavesys.com/

[11] https://eurohpc-ju.europa.eu/discover-eurohpc

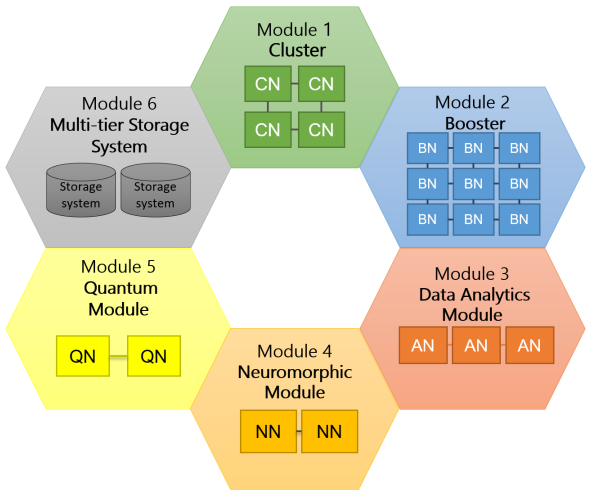

Fig. 1. Heterogenous Modular Supercomputer Architecture

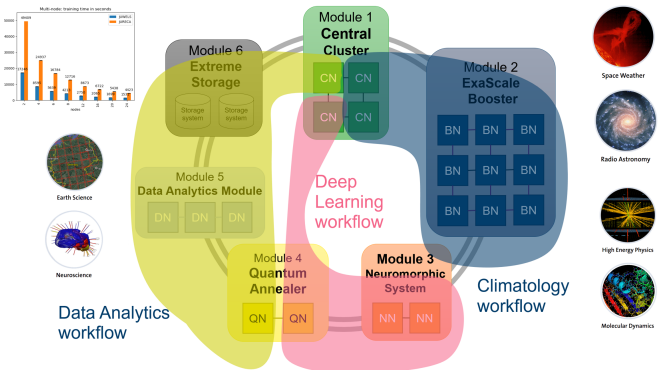

Fig. 2. Scalable and Diverse Application Workload Examples of the MSA

Centre in the co-development, operation, support of diverse user communities, and maintenance of HPC systems. The MSA has successfully shown that this approach to heterogeneous computing enables the most efficient use of computing resources while providing application developers with all necessary tools to take the step from Petascale to emerging Exascale computing [1]. Fig. 2 provides selected examples that show that no single technology could ever satisfactorily fulfil all the requirements of diverse HPC user communities (e.g., earth sciences, neurosciences, space weather, radio astronomy, high energy physics, molecular dynamics). The above benefits become even better understood when considering HPC systems' constraints to become both more user-friendly and more energy-efficient at the same time.

As Fig. 2 also reveals, the MSA enables users to take advantage of HPC systems that best suit their needs. Users with low/medium-scalable codes with high data management benefit from a general purpose cluster (i.e., cluster module). Other users with highly scalable codes and more regular communication patterns benefit from a massively parallel and scalable HPC system (i.e., booster module). However, the third type of users benefits from some characteristics of both of these two architecture elements requiring one single platform well interconnected, as shown in Fig. 2. Those third type of users also use applications that take advantage of the above rather traditional architectures (i.e., general purpose cluster vs. highly scalable booster) in combination with innovative computing architectures such as those specifically designed for data analytics (i.e., large memory), quantum computing, or neuromorphic computing.

A. The Module Heterogenous Characteristics

The MSA is illustrated in Fig. 1 and consists of various modules with different HPC hardware characteristics in order to support different workloads of applications on one overarching HPC system. Each MSA module is tailored to fit the needs of a specific set of computations, storage, or communication tasks with the goal of reaching exascale performance, which is unlikely to be achieved using traditional and rather static non-accelerated HPC system designs with CPUs, storage, memory, and interconnects.

The Cluster Module (CM) offers a module with powerful Cluster Nodess (CNs) which consist of multi-core CPU that offer fast single-thread performance and therefore makes is suitable for applications that are very computationally expensive. It offers a good amount of memory but enables only limited scalability being highly interconnected within the module itself but also to other modules using a high-performance Network Federation (NF) (e.g., EXTOLL[12]).

In contrast to the multi-core CM module, the Extreme Scale Booster (ESB) module [1] is a manycore system for highly scalable application workloads whereby each of the many CPU cores in the system offers only moderate performance. As shown in Fig. 1, the ESB module also includes the Global Collective Engine (GCE) integrated in its network fabric that leverages a Field-Programmable Gate Array (FPGA) in order to speed-up common Message Passing Interface (MPI) collective operations in hardware such as MPI reduce operations. One use case for ML is typically that compute-intensive training can be performed on the CM module while inference and testing (i.e., both less compute-intensive) can be scaled-out on the ESB.

The Data Analytics Module (DAM) offers accelerators like Graphics Processing Unit (GPU) which are particularly useful for deep learning algorithms, but also offers a high amount of memory for Apache Spark and other Hadoop ecosystem tools. Furthermore, the Scalable Storage Service Module (SSSM) offers a high capacity in storage using underlying parallel file system technologies, such as Lustre or the General Parallel File System (GPFS) from IBM at JSC.

Finally, the Network Attached Memory (NAM) module [12] is a special module that is only available as a prototype right now in DEEP. It enables setups for machine learning and sharing datasets over the network instead of duplicate downloads of datasets by individual research group members.

[12] http://www.extoll.de/

B. The MSA Implementation at JSC

To fully exploit and conduct research with our unique MSA approach [1], the JSC implements the MSA approach with different systems such as the DEEP modular supercomputer (see Fig. 3 G) and the JUWELS modular supercomputer (see Fig. 3 H). The DEEP DAM comprises 16 nodes, each with 2 Intel Xeon Cascade Lake CPUs, 1 NVIDIA V100 GPU, 1 Intel STRATIX10 FPGA, and 446 GB of RAM, as well as a total of 2 TB of Non-Volatile Memory (NVM). Hence, with an aggregated 32 TB of NVM, this HPC module design is primarily driven to support big data analytics stacks like Apache Spark[13] (see Fig. 3 R) that require a high amount of memory to work fast. The module also has access to the SSSM module (see Fig. 3 S) of the cluster to support large-scale datasets and keep the local DAM storage available for memory-intensive applications. The specifications of the DAM of the DEEP cluster are presented in Table I.

TABLE I
TECHNICAL SPECIFICATIONS OF THE DEEP DAM

CPU	16 nodes with 2x Intel Xeon Cascade Lake
Hardware Acceleration	16 NVIDIA V100 GPU 16 Intel STRATIX10 FPGA PCIe3
Memory	384 GB DDR4 CPU memory /node 32 GB DDR4 FPGA memory /node 32 GB HBM2 GPU memory /node
Storage	2x 1.5 TB NVMe SSD

The JUWELS supercomputer, currently the fastest supercomputer in Europe and 7th fastest worldwide[14], consist of 2,583 and 940 nodes respectively, totalling 122,768 CPU cores and 224 GPUs in the cluster module, and 45,024 CPU cores and 3,744 GPUs in the booster module.

III. REMOTE SENSING CASE STUDY EXPERIENCES

Earth Observation (EO) programs have an open data policy and provide a massive volume (i.e., *'big data'*) of free multi-sensor datasets for remote sensing community researchers every day. EO systems (e.g., satellites) have advanced in recent decades due to the technological evolution integrated into Remote Sensing (RS) optical and microwave instruments. For example, NASA's Landsat [13] and ESA's Copernicus [14] provide this *'big data'* via high spectral-spatial coverage at high revisiting time, which enables global monitoring of the Earth in a near real-time manner. Their characteristics include volume (increasing scale of acquired/archived data), velocity (rapidly growing data generation rate and real-time processing needs), variety (data obtained from multiple satellites' sensors that have different spectral, spatial, temporal, and radiometric resolutions), veracity (data uncertainty/accuracy), and value (extracted information) [15]. Some RS applications that require HPC resources are *'(near) real-time processing'* in case of earth disasters (see Fig. 3 A), *'exploration of oil reservoirs'* (see Fig. 3 B), and *'earth land cover classification'* (see Fig. 3 C). Our case study focuses on the research and development of successfully operational DL classifiers for *'earth land cover classification'* (see Fig. 3 top right).

[13] https://spark.apache.org/
[14] https://www.top500.org/lists/top500/2020/11/

The use of highly scalable DL tools on parallel HPC systems such as available in the Partnership for Advanced Computing in Europe (PRACE) infrastructure (see Fig. 3 D) is a necessary solution to train DL classifiers in a reasonable amount of time, providing RS researchers with a high-accuracy performance in the application recognition tasks. The same is true for the emerging HPC system landscape currently acquired by the EuroHPC Joint Undertaking such as the LUMI supercomputer in Finland (see Fig. 3 E).

Our RS case study mainly takes advantage of the MSA-based JUWELS system (see Fig. 3) at the JSC in Germany, representing the fastest EU supercomputer with 122,768 CPU cores only in its cluster module (cf. Section II-A H). While JUWELS and multi-core processors (see Fig. 3 U) offer tremendous performance, the particular challenge to exploit this data analysis performance for ML is that those systems require specific parallel and scalable techniques. In other words, using JUWELS cluster module CPUs with Remote Sensing (RS) data effectively requires parallel algorithm implementations opposed to using plain scikit-learn[15], R[16], or different serial algorithms. Parallel ML algorithms are typically programmed using the MPI standard, and OpenMP (see Fig. 3 L) that jointly leverage the power of shared memory and distributed memory via low latency interconnects (e.g., Infiniband[17]) and parallel filesystems (e.g., Lustre[18]).

Given our experience, the availability of open-source parallel and scalable machine learning implementations for the JUWELS cluster module CPUs that go beyond Artificial Neural Network (ANN)s or more recent DL networks (see Fig. 3 O) is still relatively rare. The reason is the complexity of parallel programming of ML and DL codes and thus using HPC with CPUs only can be a challenge when the amount of data is relatively moderate (i.e., DL not always succesful). One example is using a more robust classifier such as a parallel and scalable Support Vector Machine (SVM) open-source package (see Fig. 3 M) that we developed with MPI for CPUs and used to speed up the classification of RS images [16].

A. Selected DL Experiences on MSA-based Systems

The many-core processor approach of the highly scalable JUWELS booster (see Section II-B) with accelerators brings many advancements to both simulation sciences and data sciences, including innovative DL techniques. Using many numerous simpler processors with hundreds to thousands of independent processor cores enabled a high degree of parallel processing that fits very nicely to the demands of DL training whereby lots of matrix-matrix multiplications are performed. Today, hundreds to thousands of accelerators like Nvidia GPUs (see Fig. 3 V) are used in large-scale HPC systems, offering unprecedented processing power for RS data analysis. JUWELS Booster module offers 3744 GPUs of the most

[15] https://scikit-learn.org/stable/
[16] https://www.r-project.org/
[17] https://www.mellanox.com/products/interconnect/infiniband-overview
[18] https://www.lustre.org/

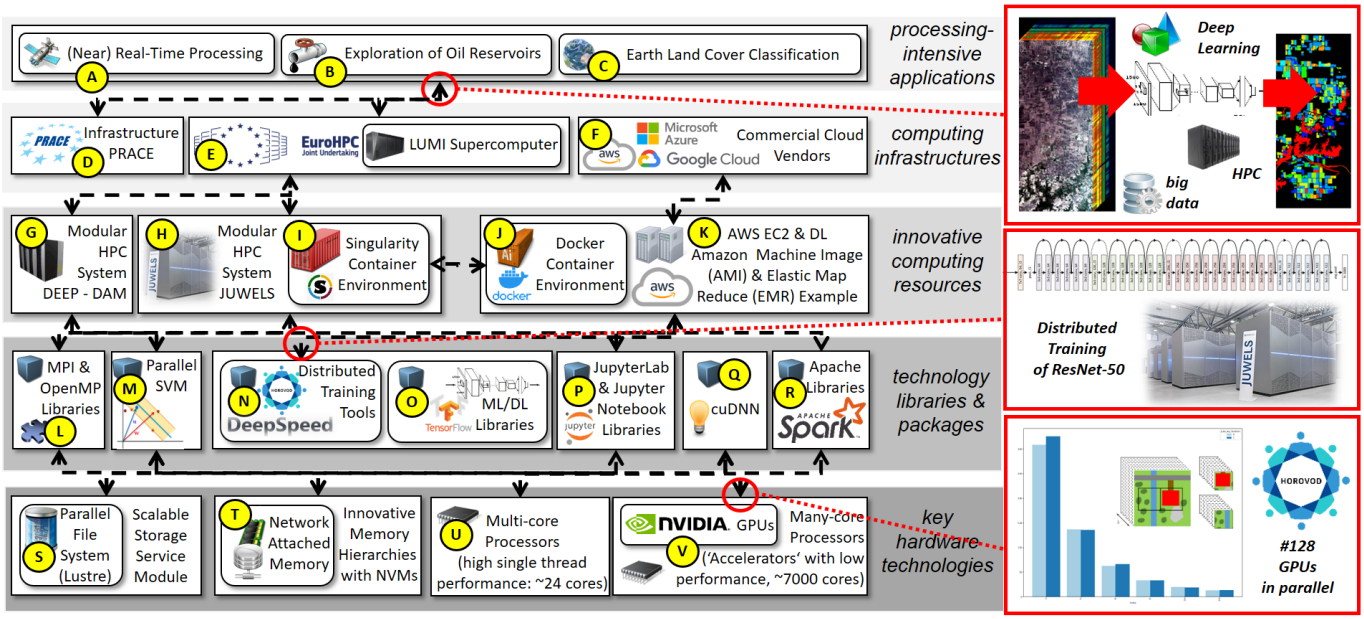

Fig. 3. Remote Sensing applications taking advantage of the MSA ensuring conceptual interoperability with Clouds.

recent innovative type of Nvidia A100 tensor core[19] cards. Our experience on MSA-based systems such as DEEP[20] (see Fig. 3 G), JURECA [21], and JUWELS shows that open-source DL packages such as TensorFlow[22] (now including Keras[23]) or pyTorch[24] are powerful tools for large-scale RS data analysis.

We experienced that it can be quite challenging to have the right versions of python code matching the available DL and ML tools and libraries versions on HPC systems with GPUs given the fast advancements of DL libraries, accelerators, and HPC systems. Our case study further reveals that using HPC systems can significantly speed up DL networks' training through distributed training frameworks, which can exploit a heterogenous HPC cluster's parallel environment such as JUWELS. Using one GPU is usually straighforward to use, but using very many GPUs connected by NVLink or NVSwitches to scale beyond a large-scale HPC node setup can be challenging using distributed DL training tools such as Horovod[25] or, more recently, DeepSpeed[26] (see Fig. 3 N). The DL model's distributed training employs a multi-node data parallelism strategy that minimises the time required to finish full training using multiple GPUs and communicating with MPI to synchronise the learning process. Hence, the fast dedicated network of the MSA-based JUWELS Booster (i.e., Infiniband) is used while the learning from data processing is distributed across multiple nodes.

Our experience in using the MSA-based JUWELS with Horovod with a cutting-edge Residual Network (RESNET-50) [17] DL network indicates a significant speed-up of training time without loosing accuracy [18] (see Fig. 3 middle right). That effect has even more impact because the speed-up enables the deployment of various models to compare their performances in a reasonable amount of time. Thus, our case study performed a high-performance distributed implementation of RESNET-50 type of a deep Convolutional Neural Network (CNN) but tuned for our *'multi-class land cover image classification'* problem. Lessons learned also revealed that RESNET-50 alongside other known DL networks is already part of the library, making it simple for even other user communities to re-use proven deep neural network architectures. Our RS case study uses the BigEarthNet [19] dataset that is conveniently stored in the Scalable Storage Service Module (SSSM) of JUWELS (see Fig. 3 D). Our experimental results attest that distributed DL training can significantly reduce the training time without affecting prediction accuracy (see Fig. 3 bottom right). Our initial case study used 96 GPUs while in later research driven by Sedona et al. [20], we achieved even a better speed-up on JUWELS using 128 interconnected GPUs after having more experience with Horovod.

B. Conceptual Interoperability with Commercial Clouds

Using MSA-based HPC systems with their software portfolio also enables conceptual interoperability with commercial Cloud vendors (). RESNET-50, for example, is available in DL packages available in our HPC module environment (i.e., Keras, TensorFlow, see Fig. 3 O) on JUWELS and DEEP. Those Python scripts from Keras and TensorFlow can be quickly migrated into clouds if needed using the Amazon Web

[19] https://www.nvidia.com/en-us/data-center/a100/
[20] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/DEEP-EST/_node.html
[21] https://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JURECA/JURECA_node.html
[22] https://www.tensorflow.org/
[23] https://keras.io/
[24] https://pytorch.org/
[25] https://horovod.ai/
[26] https://www.deepspeed.ai/

Services (AWS) EC2 combined with the Amazon Machine Images (AMI)[27] that also offer DL images with the same set of DL packages (see Fig. 3 K).

Using container technologies such as Docker[28] (see Fig. 3 J) in Clouds and Singularity[29] on JUWELS (see Fig. 3 I) enables another interoperability layer. In other words, Singularity on JUWELS can work with Docker files[30] available on the DockerHub[31]. Also Docker images are available for DL packages (e.g., TensorFlow in DockerHub[32]) but in practice working with commercial clouds is still challenging when using cutting-edge GPU types (see Fig. 3 v) required for DL because of high costs (e.g., AWS[33] EC2 24 USD per hour rate for V100, i.e., p3.16xlarge). Our RESNET-50 studies mentioned above is using 128 GPUs for many hours, hence, we need to use still the cost-free HPC computational time grants to be feasible. Examples for those HPC grants are provided by e-infrastructures such as PRACE[34] in the EU (e.g., that includes free of charge A100 GPUs in JUWELS) or Extreme Science and Engineering Discovery Environment (XSEDE)[35] in the US. Lessons learned from us reveal that free CC resources of CC vendors have drawbacks like the Google Collaboratory[36] getting just different types of GPUs assigned that make it relatively hard to perform proper speed-up studies not even mentioning the missing possibility to interconnect GPUs for large-scale distributed training of DL networks.

Beside DL packages in containers, also CC vendors offer other relevant software stacks in containers or native that are used by RS researchers with parallel and scalable tools such as Apache Spark (see Fig. 3 R) [8] in the last years. Our experience case study in using Apache Spark in Clouds as described in Haut et al. [7] uses Spark to develop a cloud implementation of a DL network for non-linear RS data compression known as AutoEncoder (AE). Of course, Spark pipelines offer also the possibility to work in conjunction with DL techniques such as recently shown by Lunga et al. [21] for RS datasets. The analysis of larger RS datasets can take advantage of Apache Spark on the large-memory DEEP DAM nodes (cf. Section II-B) using the MLlib implementation that offers also robust classifiers often used [37]. Using the DEEP DAM system then can be combined by using new types of memory hierarchies that go beyond NVM using an innovative NAM [1] (see Fig. 3 T). That also enables another interoperability level with clouds since most CC vendors offer Apache Spark with MLlib as part of their Hadoop ecosystems (e.g., AWS Elastic Map Reduce service, see Fig. 3 K) too.

[27] https://aws.amazon.com/machine-learning/amis/
[28] https://www.docker.com/
[29] https://singularity.lbl.gov/
[30] https://apps.fz-juelich.de/jsc/hps/juwels/container-runtime.html
[31] https://hub.docker.com/
[32] https://hub.docker.com/r/tensorflow/tensorflow
[33] https://aws.amazon.com/
[34] https://prace-ri.eu/
[35] https://www.xsede.org/
[36] https://colab.research.google.com/
[37] https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forest-classifier

Our case studies reveal that CC makes parallel and distributed computing more straightforward to use than traditional HPC systems, such as using the very convenient Jupyter[38] (see Fig. 3 P) toolset that abstracts the complexities of underlying computing systems. But our experience also revealed that the look and feel of using CC services (e.g., starting up Jupyter notebooks or dataset handling) differ between those vendors such as Amazon Web Services (AWS), MS Azure[39], Google Collaboratory or Google Cloud [40]. That makes it hard for non-technical users to work with various commercial clouds at the same time. The use of Jupyter notebooks is also becoming more widespread as shown by Goebbert et al. in [3] for JSC MSA systems[41]. Apart from whole containers, Jupyter notebooks can also be easily migrated into Clouds representing yet another interoperability level. To use Jupyter straightforward with DL packages and Dask [22], but we usually define our own Kernel in the Jupyter environments that work good with our HPC systems.

C. Disruptive Quantum Computing Module

Section II-B already introduced the quite innovative QM of the MSA architecture that is currently being deploying at JSC in Germany under the umbrella of the Juelich UNified Infrastructure for Quantum computing (JUNIQ)[42]. QC use *'Qubits'* that carries information as '0' or '1' or both simultaneously known as *'superposition''* [23] and the advances of ML and QC open possibilities to address new RS data analysis problems. Combining these fields is termed *'Quantum ML'* [24] leveraging QC concepts such as *'superposition'* and *'entanglement'* that make quantum computers much faster than conventional computers for certain computational tasks [23].

QA represents an innovative computing approach used for simple RS data analysis problems to solve certain ML algorithms' optimisation problems [9]. We used a quantum SVM that reveals that on QA modules of MSA-based HPC systems such as a D-Wave system[43] with 2000 qubits enables new approaches for RS research, but are still limited by having only binary classification or the requirement to sub-sample from large quantities of data and using ensemble methods [11]. More recent lessons learned from us revealed that QA evolution bears a lot of potentials since we are already using D-Wave Leap[44] with the QQ Advantage system using 5000 qubits and 35000 couplers. As shown in our module design in Fig. 1, the idea is not to replace HPC system but rather augment them (i.e., like accelerators did with CPUs) for particular computational tasks using QNs (e.g., for specific machine learning optimization problems).

[38] https://jupyter.org/
[39] https://azure.microsoft.com/en-us/
[40] https://cloud.google.com/
[41] https://jupyter-jsc.fz-juelich.de/
[42] https://www.fz-juelich.de/SharedDocs/Pressemitteilungen/UK/EN/2019/2019-10-25-juniq.html
[43] https://www.dwavesys.com/quantum-computing
[44] https://cloud.dwavesys.com/leap/

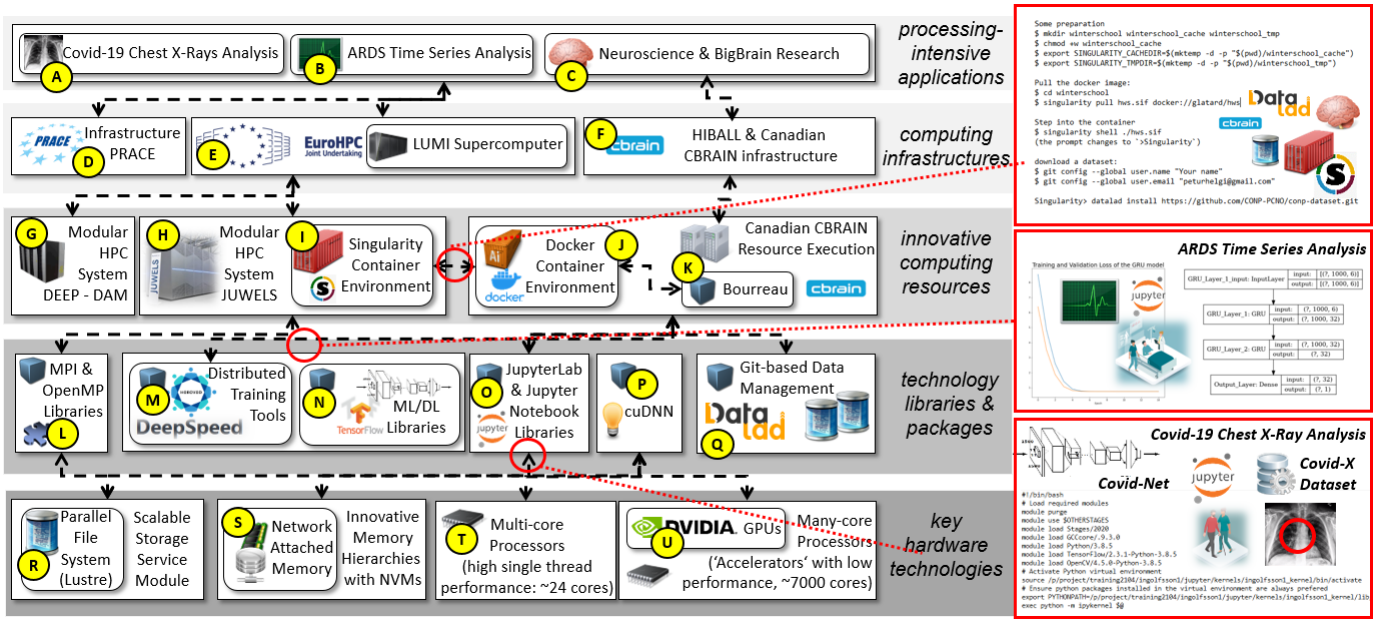

Fig. 4. Health science applications taking advantage of the MSA enabling seamless access for non-technical medical experts.

IV. Health Case Study Experiences

Health sciences are a broad field while our case studies involve medical imaging, time series analyis of medical patients, and computational neuroscience to better understand our human brain. Despite being a relative broad field all of them apply cutting-edge DL techniques on MSA systems while at the same need to be used by not technical users. Details of HPC, containers, CPU vs. GPUs, job scripts, distributed DL training tools, etc. needs to be all at least partly abstracted away so that medical doctors, medical imaging experts, or neuroscientists still are able to work with the tools.

As a consequence and in contrast to the above case study in remote sensing, these health case studies emphasize on the fact that complex MSA-based systems such as DEEP or JUWELS with cutting-edge technologies can still be used by not technical experts. Another notion is that medical data analysis or neuroscience dataset analysis results need to be often seamlessly shared with other experts to form a second opinion. The technology in the context of MSA that addresses the above constraints is our JupyterLab[45] installation at JSC (see Fig. 4 O) as described in [3]. JupyterLab is a Web-based Interactive Development Environment (IDE) for Jupyter notebooks, code, and data. JupyterLab is flexible to support a wide range of workflows in data science, HPC, and ML.

Finally, note that all the datasets we used for our case studies are actually anonymized patient data or post-mortem human brains. Hence, all data analysis is in-line with the General Data Protection Regulation (GDPR)[46].

[45] https://jupyterlab.readthedocs.io/en/stable/

[46] https://www.eu-patient.eu/globalassets/policy/data-protection/data-protection-guide-for-patients-organisations.pdf

A. Covid-19 Chest X-Ray Image Analysis

Our first case study called *'Covid-19 Chest X-Rays Analysis'* (see Fig. 4 B) represents one approach to address the COVID-19 pandemic that continues to have a devastating effect on the health of the global population. A critical step in the fight against COVID-19 is effective screening of infected patients, with one of the key screening approaches being radiology examination using chest radiography. World-wide studies revealed that patients present abnormalities in chest radiography images that are characteristic of those infected with COVID-19 [25].

We initially reused a CNN model available as open-source named COVID-Net [25] that is tailored for the detection of COVID-19 cases from chest X-ray (CXR) images (see Fig. 4 bottom right). Using DEEP and JUWELS HPC systems with distributed trailing tools (see Fig. 4 M) based on MPI (see Fig. 4 L) and DL package TensorFlow (see Fig. 4 N) we have been able to reproduce the results. Given that JUWELS is equipped with A100 GPUs (see Fig. 4 U) with latest cuDNN support (see Fig. 4 P) the inference and training time of the Covid-Net model is significantly faster as with GPUs of the previous generation given its tensor cores.

We used several publicly available datasets of COVIDx [25] that is an open access benchmark dataset initially comprising of 13,975 CXR images across 13,870 patient patient cases. But in the last couple of month this dataset was extended numerous times with new datasets made available that in turn we used again with Covid-Net as well. The SSSM of the MSA systems and its parallel file system Lustre (see Fig. 4 R) provides a powerful storage mechanism to store the COVIDx datasets and its updates.

This module also stores additional data we obtained from

a collaborating Pharma company that we in turn used to validate that Covid-Net is able to generalize well to unseen datasets. The dataset used is available as open datasets as part of the B2DROP services of the European Open Science Cloud (EOSC) [47]. Using the MSA-based systems JUWELS and DEEP seamlessly with Jupyter requires the definition of an own Kernel[48] using the module[49] environment of the MSA HPC systems (see Fig. 4 bottom right). Our experience on using our own Kernels with Jupyter notebooks is very positive while at the same time offering a user interface with notebooks that are user-friendly enough for medical imaging experts.

B. Time Series Data Analysis of ARDS Patients

Our application case study *'ARDS Time Series Analysis'* (see Fig. 4 A) addresses the medical condition Acute Respiratory Distress Syndrome (ARDS) that affects on average 1-2% of mechanically-ventilated (MV) Intensive Care Unit (ICU) patients and has a 40% mortality rate [26], [27]. At present the leading protocol for diagnosing the condition is the Berlin definition that defines onset of ARDS as a prolonged ratio of arterial oxygen potential to fraction of inspired oxygen (P/F ratio) of less than 300 mmHg, and the lower this value is determined to be, the more severe the diagnosis is [28]. Several papers have determined a correlation between early detection of onset of ARDS and survival of the patient, which highlights the need of early detection and treatment of the condition, before onset of sepsis and subsequently multi-organ failure [27], [29], [30]. Hence, the goal of this case study is to develop an algorithmic approach that provides early warning and informs medical staff of mitigating procedures can be a beneficial tool for ICU personnel.

We take advantage of the freely available ICU patient data provided in the Medical Information Mart for Intensive Care - III (MIMIC-III) database, compiled between 2001 and 2012 from admissions to the Beth Israel Deaconess Medical Center in Boston, MA [31]. The procedure thus, is to build and test our models using patient data from the MIMIC-III database, then verify our results using patient data collected from hospital participating in our German Smart Medical Information Technology for Healthcare (SMITH) project consortium[50] with real hospitals, and finally roll out the developed model for implementation in ICU for real-time testing [32]. The data, which consists of many time- series of varying lengths, is noisy and often has many missing values. Noise is often easy to filter out either using passive or unsupervised learning methods, however filling out missing values requires more in-depth knowledge of the data and how the features relate to one another. Which leads us to ask: can we implement modern DL techniques in sequence analysis to predict missing values in medical time series data?

As shown in Fig. 4, our case study uses the Gated Recurrent Units (GRU) model is that is built with two GRU layers with 32 units each, with dropout values of 0.2 and both kernel and recurrent regularization, followed by an output layer (Dense layer of size 1). GRU are belonging to the class of Recurrent Neural Network (RNN) and are thus massively computational expensive thus requiring HPC systems for training. 32 units were chosen for the layers after testing several sizes and tuning for the combination that produced lowest loss value. Loss is calculated using the Mean Absolute Error (MAE) function and the optimisation is performed using the ADAM algorithm with a learning rate of 1e-4. Not many further parameters could be altered as part of this research due to the Keras library not supporting CuDNN for GRU parameters outside the defaults. Fig. 4 shows the model structure and the shape of the tensors at each layer.

We used for early setups the DEEP DAM and then transitioned to the JUWELS MSA-based system and both worked fine for parallel and scalable time-series analysis. The results highlight One-Dimensional CNN as promising method as well as GRUs for predicting missing values in time-series data and that further supports our ongoing activities to develop an early warning system for ARDS. Finally, in this use case, medical experts use a Jupyter notebook to work with the datasets and DL models without being exposed to the underlying complexity of the MSA-based HPC systems.

C. Neuroscience

The application case study *'Neuroscience and BigBrain Research'* (see Fig. 4 C) works on the interoperability of the Canadian CBRAIN[51] infrastructure for neuroscience (see Fig. 4 F) as part of our joint Canadian-German project HIBALL[52]. We enabled interoperability by using container technologies such as Singularity on JUWELS (see Fig. 4 I) and Docker-based environments (see Fig. 4 J) available in the CBRAIN resource execution managed by the Bourreau system (see Fig. 4 K). The seamless use of containers between CBRAIN and JUWELS (see Fig. 4 top right) enables powerful neuroscience workflows across these computing infrastructures that also includes the use of the DataLad[53] tool (see Fig. 4 Q) for managing TB and PB of relevant BigBrain datasets. The user-friendly CBRAIN portal enables the use of the complex MSA-based system JUWELS without knowing the details of the system whereby this is being preconfigured for neuroscience users via Bourreau in conjunction with pre-configured containers.

V. Related Work

Given the innovative approach of the MSA-based HPC design approach (e.g., as in JSC systems like JUWELS or the nordic supercomputer LUMI), there is not much related work specifically when focussing on remote sensing and health sciences in the light of cutting-edge machine learning or deep learning. Instead we survey related work items that address

[47] https://marketplace.eosc-portal.eu/services/b2drop

[48] https://jupyter-jsc.fz-juelich.de/nbviewer/github/FZJ-JSC/jupyter-jsc-notebooks/blob/master/001-Jupyter/Create_JupyterKernel_general.ipynb

[49] https://hpc-wiki.info/hpc/Modules

[50] https://www.smith.care/home-2/

[51] https://mcin.ca/technology/cbrain/

[52] https://bigbrainproject.org/hiball.html

[53] https://www.datalad.org/

HPC, CC, and QC in the context of remote sensing and health sciences. However, we observe a move to more and more heterogenous HPC architectures, including the use of different accelerator technologies (i.e., Nvidia GPUs in JUWELS vs AMD Instinct in LUMI).

Already ten years ago, Lee et al. [33] highlighted in a survey paper for RS a trend in the design of HPC systems for data-intensive problems is to utilize highly heterogeneous computing resources. CC evolved as an evolution of Grid computing to make parallel and distributed computing more straightforward to use than traditional rather complex HPC systems. In this context RS researchers often take advantage of Apache open-source tools with parallel and distributed algorithms (e.g., map-reduce [6] as a specific form of divide and conquer approach) based on Spark [7] or the larger Hadoop ecosystem [8]. Inherent in many ML and DL approaches are optimization techniques while many of them are incredibly fast solvable by QCs [9] that represent the most innovative type of computing today. Despite being in its infancy, QAs are specific forms of QC used by RS researchers [10], [11] to search for solutions to optimization problems already today. Using GPUs in the context of Unmanned Aerial Vehicle (UAV)s is shown in [34]. Other examples of RS approaches of using Spark with CC are distributed parallel algorithms for anomaly detection in hyper-spectral images as shown in [35]. Another recent example of using QA for feature extraction and segmentation is shown by Otgonbaatar et al. in [36].

Le *et al.* trained a gradient boosted tree model using the MIMIC-III database that would provide an early prediction model for ARDS. Their model could accurately detect onset of ARDS, and had a relatively high predictivity of the condition up to 48 hours before onset [37]. Zhou *et al.* developed a joint Convolutional Neural Network (CNN)-RNN model that uses Natural Language Processing (NLP) to extract information from collected patient-physician data. The trained model is expected to provide patients with a recommendation for the appropriate clinic based on their reported symptoms and the researchers reported an accuracy of results of 88.63% [38]. Che *et al.* employed the MIMIC-III database, as well as synthetic data, in the development and testing of a novel RNN-based mortality prediction model. Their GRU-D model is based on the GRUs discussed earlier in this paper, with an added ”decay” mechanism that takes advantage of some of the inherent properties of medical timeseries data (i.e. homeostasis) in order to accomodate missing values. The proposed model performed better than traditional GRU, SVM, and Random Forest approaches [39]. Finally, Punn *et al.* fine-tuned and compared the performance of several currently available deep neural networks in diagnosing COVID-19 from chest X-ray images. The models were tested for binary classification in order to find out whether COVID-19 is detected or not, as well as for multi-class classification where healthy patients the model would distinguish between healthy, COVID-19, and pneumonia patients. Their results highlighted the NASNetLarge-based model as having an overall superior performance than the other proposed models [40].

VI. Conclusions

We conclude that HPC centres and their heterogeneous research domains entail many highly processing-intensive application areas for HPC, CC, and QC systems. The MSA addresses all these applications' needs, representing an energy-efficient system architecture that fits innovative HPC and HPDA workloads of cutting-edge HPC systems such as those emerging from the EuroHPC pre-exascale and exascale tracks (e.g., VEGA, MeluXina, Euro-IT4I, PetaSC, LUMI, Deucalion, Leonardo, etc.). It balances and satisfies the requirements of end-users as well as HPC centre operators. The highly interconnected MSA ensures that current workloads are supported by the Cluster module, the Extreme Scale Booster module, and the Data Analytics module (e.g., health application lessons learned described above). More disruptive workloads with quantum modules and neuromorphic modules can be conveniently integrated too (e.g., remote sensing application lessons learned described above).

Further conclusions about resource management and scheduling are fully supporting the MSA with scheduling heterogeneous workloads to show being able to schedule heterogeneous workloads onto matching combinations of MSA module resources. The two diverse scientific domain areas of remote sensing and health sciences have been used in this paper to demonstate some of these added values of the MSA while more impacts of the MSA can be seen on the DEEP series of projects and the Center of Excellence (CoE) Research on AI- and Simulation-Based Engineering at Exascale (RAISE) Web page[54].

CoE RAISE will leverage the MSA implementation (e.g., in HPC systems such as JUWELS and LUMI) in various areas of engineering applications. Hence, our approaches outlined in this paper are currently applied to those engineering applications outlining our future work in this area briefly. We started working on compute-intensive use cases in the areas of AI for turbulent boundary layers, AI for wind farm layout optimization, AI for data-driven models in reacting flows, smart models for next-generation aircraft engine design, and AI for wetting hydrodynamics. We also started working on data-intensive use cases in the areas of event reconstruction and classification at the CERN HL-LHC, seismic imaging with remote sensing (oil and gas exploration and well maintenance), defect-free metal additive manufacturing, sound engineering. Also these applications will validate the full MSA hardware and software stack with relevant HPC and extreme data workloads and thus demonstrate the benefits of the MSA in the next years.

References

[1] E. Suarez, N. Eicker, and T. Lippert, *Modular Supercomputing Architecture: From Idea to Production*, 1st ed. Imprint CRC Press, 2019, pp. 223–255, contemporary High Performance Computing.

[2] J. Kwon, N. Kim, M. Kang, and J. WonKim, “Design and prototyping of container-enabled cluster for high performance data analytics,” in *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 436–438.

[54] https://www.coe-raise.eu

[3] J. H. Göbbert, T. Kreuzer, A. Grosch, A. Lintermann, and M. Riedel, "Enabling interactive supercomputing at jsc lessons learned," in *International Conference on High Performance Computing*. Springer, 2018, pp. 669–677.

[4] G. Aloisioa, S. Fiorea, I. Foster, and D. Williams, "Scientific big data analytics challenges at large scale," *Proceedings of Big Data and Extreme-scale Computing (BDEC)*, 2013.

[5] T. Lippert, D. Mallmann, and M. Riedel, "Scientific big data analytics by HPC," in *John von Neumann Institute for Computing Symposium, Jülich Supercomputing Center*, 2016.

[6] Q. Zou, G. Li, and W. Yu, "MapReduce Functions to Remote Sensing Distributed Data Processing - Global Vegetation Drought Monitoring as Example," *Journal of Software: Practice and Experience*, vol. 48, no. 7, pp. 1352–1367, 2018.

[7] J. M. Haut, J. A. Gallardo, M. E. Paoletti, G. Cavallaro, J. Plaza, A. Plaza, and M. Riedel, "Cloud deep networks for hyperspectral image analysis," *IEEE transactions on geoscience and remote sensing*, vol. 57, no. 12, pp. 9832–9848, 2019.

[8] I. Chebbi, W. Boulila, N. Mellouli, M. Lamolle, and I. R. Farah, "A comparison of big remote sensing data processing with hadoop mapreduce and spark," in *2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*. IEEE, 2018, pp. 1–4.

[9] M. Henderson, J. Gallina, and M. Brett, "Methods for accelerating geospatial data processing using quantum computers," *Quantum Machine Intelligence*, vol. 3, no. 1, pp. 1–9, 2021.

[10] R. Ayanzadeh, M. Halem, and T. Finin, "An Ensemble Approach for Compressive Sensing with Quantum Annealers," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020, to appear.

[11] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen, and M. Riedel, "Approaching remote sensing image classification with ensembles of support vector machines on the d-wave quantum annealer," in *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2020, pp. 1973–1976.

[12] J. Schmidt, "Accelerating checkpoint/restart application performance in large-scale systems with network attached memory," Ph.D. dissertation, Ruprecht-Karls University Heidelberg, Germany, 2017, dOI: 10.11588/heidok.00023800.

[13] M. A. Wulder, J. G. Masek, W. B. Cohen, T. R. Loveland, and C. E. Woodcock, "Opening the archive: How free data has enabled the science and monitoring promise of landsat," *Remote Sensing of Environment*, vol. 122, pp. 2–10, 2012.

[14] J. Aschbacher, "Esa's earth observation strategy and copernicus," in *Satellite earth observations and their impact on society and policy*. Springer, Singapore, 2017, pp. 81–86.

[15] M. Chi, A. Plaza, J. A. Benediktsson, Z. Sun, J. Shen, and Y. Zhu, "Big data for remote sensing: Challenges and opportunities," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2207–2219, 2016.

[16] G. Cavallaro, M. Riedel, M. Richerzhagen, J. A. Benediktsson, and A. Plaza, "On understanding big data impacts in remotely sensed image classification using support vector machine methods," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 8, no. 10, pp. 4634–4646, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and J. A. Benediktsson, "Remote sensing big data classification with high performance distributed deep learning," *Remote Sensing*, vol. 11, no. 24, p. 3056, 2019.

[19] G. Sumbul, M. Charfuelan, B. Demir, and V. Markl, "Bigearthnet: A large-scale benchmark archive for remote sensing image understanding," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2019, pp. 5901–5904.

[20] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and M. Book, "Scaling up a Multispectral RESNET-50 to 128 GPUs," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2020, to appear.

[21] D. Lunga, J. Gerrand, L. Yang, C. Layton, and R. Stewart, "Apache spark accelerated deep learning inference for large scale satellite image analytics," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 271–283, 2020.

[22] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proceedings of the 14th python in science conference*, vol. 126. Citeseer, 2015.

[23] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[24] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[25] L. Wang, Z. Q. Lin, and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[26] D. G. Ashbaugh, D. B. Bigelow, T. L. Petty, and B. E. Levine, "Acute respiratory distress in adults," *The Lancet*, vol. 290, no. 7511, pp. 319 – 323, 1967, originally published as Volume 2, Issue 7511. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140673667901687

[27] J. Villar, J. Blanco, J. M. Añón, A. Santos-Bouza, L. Blanch *et al.*, "The ALIEN study: incidence and outcome of acute respiratory distress syndrome in the era of lung protective ventilation," *Intensive Care Medicine*, vol. 37, no. 12, pp. 1932–1941, Dec 2011. [Online]. Available: https://doi.org/10.1007/s00134-011-2380-4

[28] The ARDS Definition Task Force, "Acute Respiratory Distress Syndrome: The Berlin Definition of ARDS," *JAMA*, vol. 307, no. 23, pp. 2526–2533, Jun 2012. [Online]. Available: https://doi.org/10.1001/jama.2012.5669

[29] A. Das, P. P. Menon, J. G. Hardman, and D. G. Bates, "Optimization of mechanical ventilator settings for pulmonary disease states," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 6, pp. 1599–1607, 2013.

[30] S. Kushimoto, T. Endo, S. Yamanouchi, T. Sakamoto, H. Ishikura, Y. Kitazawa *et al.*, "Relationship between extravascular lung water and severity categories of acute respiratory distress syndrome by the berlin definition," *Critical Care*, vol. 17, no. R132, 2013.

[31] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, no. 160035, May 2016. [Online]. Available: https://doi.org/10.1038/sdata.2016.35

[32] A. Winter, S. Stäubert, D. Ammon, S. Aiche, O. Beyan, V. Bischoff, P. Daumke, S. Decker, G. Funkat, J. E. Gewehr, M. Riedel *et al.*, "Smart medical information technology for healthcare (smith): data integration based on interoperability standards," *Methods of information in medicine*, vol. 57, no. Suppl 1, p. e92, 2018.

[33] C. A. Lee, S. D. Gasster, A. Plaza, C.-I. Chang, and B. Huang, "Recent developments in high performance computing for remote sensing: A review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 4, no. 3, pp. 508–527, 2011.

[34] R. Wang, X. Xiao, B. Guo, Q. Qin, and R. Chen, "An effective image denoising method for uav images via improved generative adversarial networks," *Sensors*, vol. 18, no. 7, p. 1985, 2018.

[35] Y. Zhang, Z. Wu, J. Sun, Y. Zhang, Y. Zhu, J. Liu, Q. Zang, and A. Plaza, "A distributed parallel algorithm based on low-rank and sparse representation for anomaly detection in hyperspectral images," *Sensors*, vol. 18, no. 11, p. 3627, 2018.

[36] S. Otgonbaatar and M. Datcu, "Quantum annealing approach: Feature extraction and segmentation of synthetic aperture radar image," in *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2020, pp. 3692–3695.

[37] S. Le, E. Pellegrini, A. Green-Saxena, C. Summers, J. Hoffman, J. Calvert, and R. Das, "Supervised machine learning for the early prediction of acute respiratory distress syndrome (ARDS)," *Journal of Critical Care*, vol. 60, pp. 96–102, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0883944120306237

[38] X. Zhou, Y. Li, and W. Liang, "Cnn-rnn based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[39] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[40] N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Covid-19 epidemic analysis using machine learning and deep learning algorithms," *MedRxiv*, 2020.

View publication stats