MODULAR SUPERCOMPUTING DESIGN SUPPORTING MACHINE LEARNING APPLICATIONS

Ernir Erlingsson[1], *Gabriele Cavallaro*[2], *Andreas Galonska*[2], *Morris Riedel*[1,2], *Helmut Neukirchen*[1]

[1] School of Engineering and Natural Sciences, University of Iceland, Iceland
[2] Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany

ABSTRACT

The DEEP-EST (DEEP – Extreme Scale Technologies) project designs and creates a Modular Supercomputer Architecture (MSA) whereby each module has different characteristics to serve as blueprint for future exascale systems. The design of these modules is driven by scientific applications from different domains that take advantage of a wide variety of different functionalities and technologies in High-Performance Computing (HPC) systems today. In this context, this paper focuses on machine learning in the remote sensing application domain but uses methods like Support Vector Machines (SVMs) that are also used in life sciences and other scientific fields. One of the challenges in remote sensing is to classify land cover into distinct classes based on multi-spectral or hyper-spectral datasets obtained from airborne and satellite sensors. The paper therefore describes how several of the innovative DEEP-EST modules are co-designed by this particular application and subsequently used in order to not only improve the performance of the application but also the utilization of the next generation of HPC systems. The paper results show that the different phases of the classification technique (i.e. training, model generation and storing, testing, etc.) can be nicely distributed across the various cluster modules and thus leverage unique functionality such as the Network Attached Memory (NAM).

Index Terms— High-Performance Computing (HPC), Exascale Computing, Network Attached Memory (NAM), Support Vector Machines (SVMs), Remote Sensing

1. INTRODUCTION

Traditional High-Performance Computing (HPC) systems have been mainly designed for scientific and engineering applications that often are based on numerical methods using known physical laws. Those class of applications are often summarized under the term 'simulation sciences' but are partly different from purely data-driven applications known as 'data sciences' today. Those latter applications use parallel and scalable machine learning methods, data mining algorithms, or statistical approaches in order to automatically discover information in large datasets (i.e. 'big data'), ideally following a systematic process such as the Cross-Industry Standard Process for Data Mining (CRISP-DM) [1]. As a consequence, current HPC systems design needs to support a much broader range of application requirements in order to enable performance improvements.

Another change in HPC systems design is driven by the ever expanding set of parallel technology hardware that becomes available such as a wide range of accelerator technologies like NVIDIA, General-Purpose computing on Graphics Processing Units (GPGPUs), or innovative memory technologies. While many paper contributions purely focus on the evolution of using particularly GPGPUs and accelerators in general in applications, there is less focus on innovative memory technologies that have very interesting capabilities especially when considering data-driven applications.

This paper, therefore, describes key approaches of using the innovative Network Attached Memory (NAM) [2, 3] from the HPC hardware manufacturer EXTOLL in a real data-driven application that employs machine learning algorithms that are used in a wide variety of applications, such as in the realm of life or earth sciences. The particular machine learning algorithm used to drive the HPC system co-design using the aforementioned NAM are Support Vector Machines (SVMs) [4] in the application field of remote sensing, but SVMs and their usage of the NAM are applicable to other application domains in the same way. The results of this paper have been implemented as part of the European Union's Horizon 2020 exascale project *Dynamical Exascale Entry Platform – Extreme Scale Technologies* (DEEP-EST) [5] and its HPC system prototypes that beside the NAM also offers other innovative modules.

The remainder of the paper is structured as follows. Following this introduction, Section 2 describes the technology background in the realm of the DEEP-EST project such as its various modules while Section 2 also includes a brief introduction to machine learning with a particular focus on the SVM technique. Section 3 describes the application problem domain and its need for parallel and scalable SVM implemen-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No. 754304 DEEP-EST. Correspondence: ere29@hi.is

tations on HPC systems. Afterwards, Section 4 technically describes how the NAM module and our SVM implementation in combination with other innovative DEEP-EST modules is particularly useful to improve application performance and HPC system utilization. The paper concludes with a summary and outline of future work.

2. BACKGROUND

This section provides a short background on machine learning and the various DEEP-EST hardware accelerator modules.

2.1. Machine Learning with Support Vector Machines

Machine learning techniques are used in a process of automatically discovering useful information or specific patterns in large datasets. We thus assume a pattern to be detected in the data and there is no mathematical formula or known physical law (cf. simulation sciences) that can be directly implemented to find those patterns with an analytic solution. Instead, the focus on the dataset in question itself in order to discover the pattern with learning methods such as classification and clustering algorithms or regression methods.

The contributions of this paper are based on supervised learning using classification whereby groups of data already exist with labeled group identity and new data points are classified to those existing groups based on a learned classification model. In more detail, this learning process essentially follows the following three steps: (i) training, (ii) model generation and storing, and (iii) testing. We describe these three rather general classification steps in the context of a concrete model called Support Vector Machines (SVMs) [4], but they are valid for any other classification technique such as Random Forests (RFs), Artificial Neural Networks (ANNs), or, more recently, deep learning networks such as Convolutional Neural Networks (CNNs).

The (i) training process uses training data with a machine learning algorithm is often based on constrained optimization techniques that are solved with quadratic programming and sequential minimal optimization (SMO) in the case of SVMs. These computational demanding optimization techniques do not scale well and thus usually require a CPU with high single thread performance in order to perform proper model training and reach convergence in a reasonable amount of time. The result of step (i) is a model that was trained with a particular setup with respect to training parameters (e.g. regularization parameter C in SVMs) or the use of a kernel like Radial Basis Functions (RBFs).

In step (ii), the trained model from step (i) needs to be generated in a specific model format (i.e. SVM model file) and stored as a file, or transferred in some other way, to reach step (iii). The specific property of the particular step relevant for this paper (ii) is that the SVM model consists of an 'in-sample' property meaning that the support vectors [4] that support the decision surface to separate datasets of different classes is also part of the dataset. Hence, there is a likelihood that the bigger the dataset, the more support vectors are needed; as a consequence, the model file is getting larger, too. Thus dropping the model file to storage after step (i) in step (ii) to be loaded by step (iii) is something we refer to as 'model storage bottleneck' that is later addressed in this paper.

In order to predict new data items in step (iii) based on the model, the model file from step (ii) is again loaded into memory from file in the above case and used for inference. This prediction step is called testing and uses its own dataset or, when model building is finished, represents the way how a model is used in a final model deployment in daily applications. In contrast to step (i), the computational needs for step (iii) and its nature of inference is a process that requires only moderate CPU power. Finally, due to the page restriction of the paper, we can not go into details for another important element of the machine learning process that is validation that in turn is also used in conjunction with regularization techniques to combat overfitting.

2.2. Innovative DEEP-EST Modules

The DEEP-EST project [5] performs research in new HPC system architectures using innovative hardware co-designed by a wide variety of applications. One of the research outcome of the DEEP-EST project is the creation and exploration of the Modular Supercomputer Architecture (MSA) [6] that in turn is adopted by the Jülich Supercomputing Centre (JSC) in Germany for production end users. 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 Nodes (CNs) which consist of multi-core CPUs 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 the Network Federation (FN).

In contrast to the multi-core CM module, the Extreme Scale Booster (ESB) module [6] 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 net-

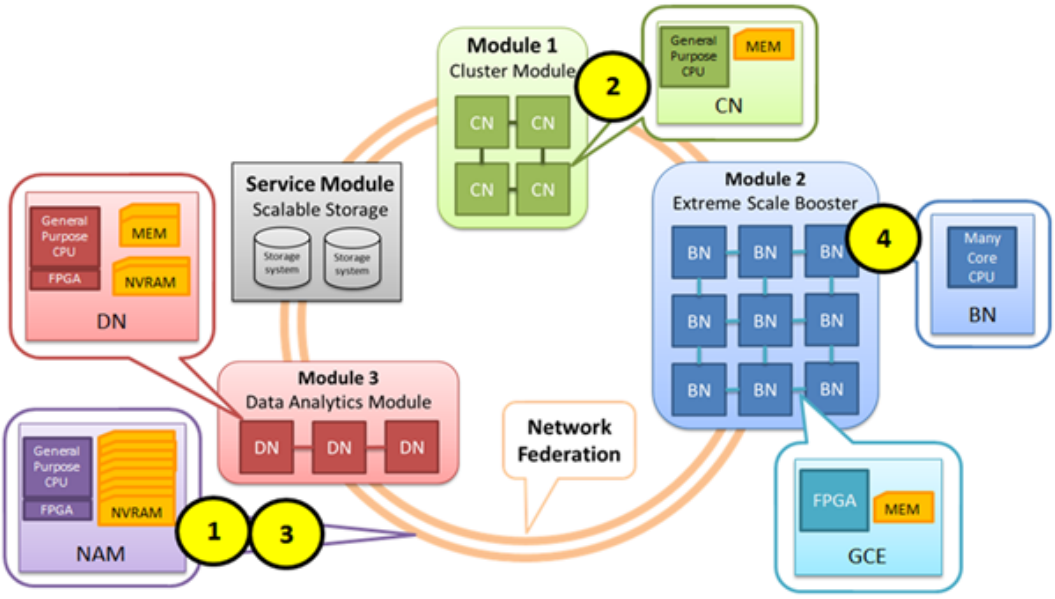

Fig. 1. Utilizing the Modular supercomputer Architecture (MSA) for machine learning algorithms like Support Vector Machines

work 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.

The Data Analytics Module (DAM) offers accelerators like GPGPUs which are particularly useful for deep learning algorithms. 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 NAM module [2, 3] is a special module that is the key focus of this paper and described in more detail in Section 4.

3. PARALLEL AND SCALABLE SVMS IN REMOTE SENSING USING DEEP-EST PLATFORM

This section describes our application and how we map it to the MSA of the DEEP-EST platform.

3.1. Parallel and Scalable Application Demands

The remote sensing application dataset used in this paper is the Indian Pines AVIRIS dataset [7] over an agricultural site composed of agricultural fields with regular geometry (200 spectral bands, 1417x617 pixels, spatial resolution of 20 meter, 52 classes of different land cover). Before the above phases of classification are performed the raw hyper-spectral data is used with feature engineering, which in this case is the Self-Dual Attribute Profile (SDAP) [8].

Progress in sensor technology leads to an ever-increasing amount of data which needs to be classified in order to extract information. This big amount of data requires parallel processing by running parallel and scalable implementations of classification algorithms such as the SVM introduced above on HPC systems. This paper uses our parallel and scalable PiSvM [9] that scales well across a large amount of CPUs and is an improved version of the original πSvM [10] implementation. As shown in Cavallaro et al. [11] this optimized version offers a good classification accuracy while at the same time being more efficient in using the MPI standard for parallel processing in HPC clusters enabling a speed-up for remote sensing applications.

3.2. Parallel Training and Model File Generation

The (i) training phase feeds the pre-processed remote sensing training data into the SVM algorithm which in turn calculates data-space coefficients using a particular set of hyper parameters. These coefficients identify support vectors that can then implicitly be used to classify arbitrary input data (e.g., new acquired hyper-spectral images) via a generated SVM model classifier.

In order to speed-up the (i) training phase outlined above, we use our parallel `pisvm-train` executable that in turn is based on the serial de-facto standard implementation libSVM [12]. As the optimization algorithm implementations that are part of the inherent libSVM are very computational demanding, it makes sense to use the DEEP-EST CM for the (i) training phase. This executable takes as input, alongside the training dataset location, several parameters related to the precise type of SVM to be used as they can also be used for regression. Another parameter for this executable is the type of the kernel function to be used in training and its associated parameters like *gamma* in the case of the RBF kernel.

The implementation of `pisvm-train` is executed on the DEEP-EST CM using a typical scheduler batch script for parallel processing. According to phase (ii) model file generation and storage, the parallel computing job on the CM creates an SVM model on the parallel file system to be used by step (iii).

The PiSvM implementation from [9] uses non-optimized POSIX I/O for file handling. Our porting to the DEEP-EST MSA involves using MPI parallel I/O for for reading the initial input and writing the final output. Passing the intermediate data between the different steps and their executables is performed in a highly optimized way using the NAM as discussed in Section 4.

3.3. Use of the SVM Model File in Parallel Prediction

In the (iii) testing phase, we use the generated model file as input parameter alongside a test dataset location for the `pisvm-predict` executable that is used for prediction. The goal of this phase is to evaluate the SVM classifier's accuracy by comparing its output for a specifically selected testing dataset of input data with a correct classifications known beforehand and not used during the (i) training phase. This rather embarassingly parallel process is highly scalable and thus well suited for the DEEP-EST ESB module.

4. IMPLEMENTATION USING NAM

The preceding Section 3 outlined the standard use of the DEEP-EST modules with POSIX file I/O or potentially MPI parallel I/O for the involved file accesses. However, we observe a limitation in always storing an SVM model as a normal file during the phase (ii) model file generation and storage as performed by the executable `pisvm-train` and reading it again in as normal file in the subsequent phase by the executable `pisvm-predict`. This section describes how the NAM DEEP-EST module can be exploited in order to avoid this 'model storage bottleneck' which can be severe because SVM models tend to be very large.

4.1. Proposed Methodology

We exploit the MSA to enhance the training and testing phase of our PiSvM implementation focusing on the speed-up the NAM offers as an intermediary model storage target between the training and evaluation steps of the two involved PiSvM executables. This is depicted in Fig. 1 with the following description of the steps marked in the figure with circles: (1) The remote sensing training and testing datasets are loaded into the NAM module, this reduces latency and bandwidth restrictions, compared with the standard storage module, when data is accessed; (2) Training a model is computationally expensive due to the inherent convergence process executed by PiSvM, it is therefore best suited for the DEEP-EST CM, which offers the most powerful CPUs as discussed in Section 3.2; (3) The trained model is placed in the NAM module to speed-up access to it during the evaluation step, where the model is evaluated using another dataset; (4) The model is loaded from the NAM and its accuracy is evaluated. This is an embarrassingly parallel operation which is therefore best suited to take advantage of the ESB module that has the highest number of CPUs as discussed in Section 3.3.

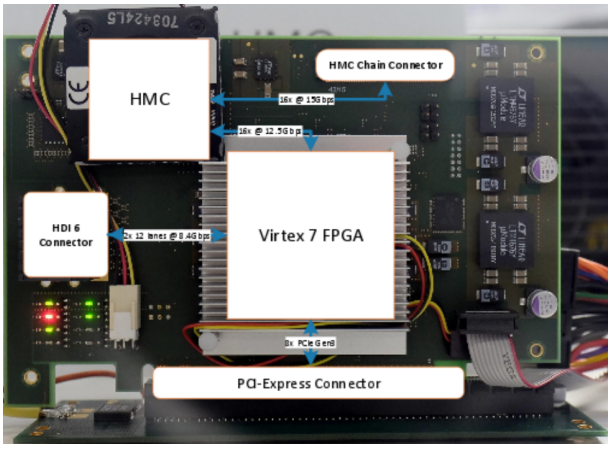

Fig. 2. NAM hardware architecture [3]

4.2. NAM Overview

The Network Attached Memory (NAM) is a device that can be described as a module which exists in the fast fabric interconnection between all other modules within the heterogeneous system. It has non-volatile high-performance RAM which makes it extremely suitable for the role as a "faster than SSD" intermediary storage. Furthermore, it also has an FPGA integrated on its board that applications can utilize to perform near-data processing, independent and parallel to other modules, on data stored in the NAM

Additional important use for the NAM is fast checkpointing [2] to improve the application's fault-tolerance, or model-selection, using inference to organize a set of trained models which can then used for transfer learning [13]. In this paper, however, we focus on its role as a fast storage target used by the PiSvM application for I/O performance gains.

4.3. NAM Architecture

The NAM hardware architecture, as depicted in Fig. 2, consists of a PCI-Express form factor card that is equipped with a Xilinx Virtex7 FPGA and Micron's Hybrid Memory Cube (HMC), which is a very fast DRAM stacked in 3D with integrated memory controllers. Multiple HMCs can be connected together using the HMC chain connector to extend the NAM's fast memory capacity. Finally, the NAM is accessible over the

high-bandwidth and low-latency EXTOLL network[1] via the HDI-6 connector depicted in Fig. 2

4.4. libNAM

The libNAM library [14], written in C, provides the API that developers need to use the NAM for their applications in DEEP-EST. Its API is modeled after the proprietary EXTOLL RMA API and its current implementation is indeed built on top of the same API, effectively limiting its usage to EXTOLL fabric interconnects at the moment. The usage of this API for speeding up PiSvM is described in the following two sections.

4.5. Using NAM for Training

Near the end of its execution, the training step of PiSvM has generated a model which can be used to classify unseen data in the prediction step. The model, however, must first be stored in the NAM where it can subsequently be accessed in the prediction step. This is demonstrated by the code snippet below: first, the NAM is initialized and then the fitting memory allocation is requested from it; thereafter the model is written into the allocation, and finally, a metadata file describing the allocation is produced.

```
#include <stdio.h>
#include "nam_util.h"

void saveModelNAM(Model* trained, char* path) {
    m_size = sizeof(trained);
    nam_initialize();
    nam_alloc = nam_malloc(m_size);
    nam_put(nam_alloc, trained);
    nam_write_metadata(nam_alloc, path);
}
```

Code Snippet 1. Storing the trained model in NAM

Note that `nam_put` in the above Code Snippet 1 stores the actual data whereas `nam_write_metadata` saves the allocation's metadata to a standard HPC file system with the given path using standard POSIX file I/O. The metadata includes the location and size of an allocation which enables other applications to access its underlying data on the NAM, as can be seen in the code snippet in the next section.

4.6. Using NAM for Prediction

A generated model must be evaluated to make inferences about its accuracy and suitability. The evaluation step loads the model from the NAM, as demonstrated by Code Snippet 2: first, the NAM is initialized, followed by the loading of metadata file from the standard HPC file system. The metadata contains the necessary information about the allocation made during the training step which enables the NAM API to load its contents, i.e. the model.

```
#include "nam_util.h"

Model* loadModelNAM(char* path) {}
    nam_initialize();
    nam_alloc = nam_read_metadata(path);
    model = nam_get(nam_alloc);
    return model;
}
```

Code Snippet 2. Model loaded from the NAM and evaluated

Using the same metadata file allows the PiSvM executables `pisvm-train` and `pisvm-predict` to exchange the model data in an extremely efficient way and thus minimizes idle processes by reducing waiting for I/O to complete.

5. SUMMARY AND OUTLOOK

We presented the Dynamical Exascale Entry Platform – Extreme Scale Technologies (DEEP-EST) Modular Supercomputer Architecture (MSA) which is heavily based on hardware acceleration for all parts of an HPC system, i.e. not only computation, but also storage/memory and interconnects. These accelerated hardware modules are intended to serve as blueprints for tailoring future HPC systems towards exascale driven by the needs of the particular applications.

Our application that drives the co-design of the DEEP-EST MSA is parallel and scalable machine learning on huge datasets which includes deep learning [13], clustering [15], and classification. For the latter, we use Support Vector Machines (SVMs) that are mapped on the hardware modules of the MSA as described in this paper: wherever possible, the benefits of these hardware accelerator modules are leveraged: massively parallel manycore processing of embarrassingly parallel processes is performed on the Extreme Scale Booster (ESB), whereas tasks that do not scale well are executed on the high-performance Cluster Module (CM), frequently used data is stored in the Network Attached Memory (NAM) including near-data processing via FPGA. This all is facilitated by a fast fabric interconnection and the Global Collective Engine (GCE) for MPI collective operations.

The focus of this paper is on exploiting the Network Attached Memory (NAM) for storing intermediate results of the different steps of an SVM classification chain, e.g: a model is trained and then its prediction accuracy is evaluated by another executable, i.e. the learned model needs to be stored persistently between runs of different executables. By using the fast NAM hardware and NAM API instead of the comparably slower Scalable Storage Service Module (SSSM) and standard POSIX I/O, a considerable speed-up of I/O is possible. In addition to using the NAM for intermediate data, using MPI parallel I/O for reading the initial input and writing the final output allows further I/O improvement. The potential for

[1] `http://www.extoll.de`

speed-up using the NAM has been identified by tracing the initial PiSvM implementation which gave valuable insights concerning idle hardware due to inefficient I/O and load imbalances that we mitigate with our approach.

The described accelerator modules are currently being developed as part of the DEEP-EST project and are still subject to change. For example, the current NAM prototype will be replaced within the life-span of the DEEP-EST project with an improved device that includes a more powerful FPGA, at least 128 GB of RAM, and much higher bandwidth. Because the hardware and its corresponding API implementation are not yet stable, we can currently not yet provide benchmarking results, but the first preliminary runs of our parallel PiSvM chain on the draft prototype hardware are very promising.

Future work, outside of porting our PiSvM implementation to the final DEEP-EST hardware, includes improving the scalability of parallel SVMs even further: we have started to implement the Cascade SVM approach where a dataset is split and then optimized separately using multiple SVMs; the subresults are finally combined again in a cascade of SVMs until the global optimum is reached [16].

Finally, we have started to investigate an alternative to using SVMs for hyperspectral image classification by applying deep learning with CNNs: first results look promising [17] and we intend to investigate its scalability on the DEEP-EST platform.

6. REFERENCES

[1] C. Shearer, "CRISP-DM Model The New Blueprint for Data Mining," *Journal Data Warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[2] J. Schmidt, *Accelerating Checkpoint/Restart Application Performance in Large-Scale Systems with Network Attached Memory*, Ph.D. thesis, Ruprecht-Karls University Heidelberg, Germany, 2017, DOI: 10.11588/heidok.00023800.

[3] J. Schmidt, "NAM – Network Attached Memory," Doctoral Showcase summary and poster at International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2016, Nov. 2016, http://sc16.supercomputing.org/sc-archive/doctoral_showcase/doc_files/drs106s2-file2.pdf.

[4] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20(3), pp. 273–297, 1995.

[5] DEEP-EST project consortium, "DEEP-EST project," website, 2018, http://deep-est.eu/.

[6] N. Eicker, "Taming Heterogeneity in HPC," Keynote Presentation at SAI Computing Conference 2016, July 2016, https://youtu.be/aM9AkgG5ud4.

[7] M.F. Baumgardner, L.L. Biehl, and D.A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," 2015, https://purr.purdue.edu/publications/1947/1.

[8] G. Cavallaro, N. Falco, M. Dalla Mura, and J.A. Benediktsson, "Automatic Attribute Profiles," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1859–1872, 2017.

[9] M. Richerzhagen, "piSvM," website, 2015, https://github.com/mricherzhagen/pisvm.

[10] D. Brugger, "πSvM," website, 2014, http://pisvm.sourceforge.net.

[11] 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 J. Sel. Topics Appl. Earth Observ.*, vol. 8, no. 10, pp. 4634–4646, 2015.

[12] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[13] E. Erlingsson, G. Cavallaro, M. Riedel, and H. Neukirchen, "Enhancing deep learning towards exascale with the DEEP-EST Modular Supercomputer Architecture," in *5th Exascale Applications and Software Conference (EASC 2018), Edinburgh, Scotland, 17-19 April 2018.*, 2018 (to appear), Abstract and poster.

[14] A. Galonska, "libNAM," website, 2017, https://gitlab.version.fz-juelich.de/galonska1/libNAM.

[15] E. Erlingsson, G. Cavallaro, M. Riedel, and H. Neukirchen, "Scaling DBSCAN towards exascale computing for clustering of big data sets," in *Geophysical Research Abstracts*. 2018 (to appear), Copernicus, Abstract at European Geosciences Union (EGU) General Assembly 2018 session Big data and machine learning in geosciences, Vienna, Austria, 8-13 April 2018.

[16] H.P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel Support Vector Machines: The Cascade SVM," in *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, 2004, pp. 521–528.

[17] J. Lange, G. Cavallaro, M. Götz, E. Erlingsson, and M. Riedel, "The Influence of Sampling Methods on Hyperspectral Image Classification with 3D Convolutional Neural Networks," in *Proc. IGARSS*, 2018 (submitted).