JUWELS Booster – A Supercomputer for Large-Scale AI Research

Stefan Kesselheim[1⋆], Andreas Herten[1⋆], Kai Krajsek[1⋆], Jan Ebert[1⋆], Jenia Jitsev[1⋆], Mehdi Cherti[1⋆], Michael Langguth[1⋆], Bing Gong[1⋆], Scarlet Stadtler[1⋆], Amirpasha Mozaffari[1⋆], Gabriele Cavallaro[1⋆], Rocco Sedona[1,2⋆], Alexander Schug[1,3⋆], Alexandre Strube[1], Roshni Kamath[1], Martin G. Schultz[1], Morris Riedel[1,2], Thomas Lippert[1]

[1] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Germany, contact <n>.<surname>@fz-juelich.de
[2] School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
[3] University of Duisburg-Essen, Germany

Abstract. In this article, we present JUWELS Booster, a recently commissioned high-performance computing system at the Jülich Supercomputing Center. With its system architecture, most importantly its large number of powerful Graphics Processing Units (GPUs) and its fast interconnect via InfiniBand, it is an ideal machine for large-scale Artificial Intelligence (AI) research and applications. We detail its system architecture, parallel, distributed model training, and benchmarks indicating its outstanding performance. We exemplify its potential for research application by presenting large-scale AI research highlights from various scientific fields that require such a facility.

Keywords: Artificial Intelligence · Deep Learning · GPU · High-Performance Computing · Machine Learning · Jülich Supercomputing Centre.

1 Introduction

In recent years, deep learning methods brought up radical transformation across various disciplines of technology and science, sparking interest in the previously academic field of Artificial Intelligence (AI), and extending it far beyond academia. Progress is notified and valued in mainstream and business media and widely discussed. Without disparaging many other ingenious contributions, it can be said that a single paper started the boom: in 2012, Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton had shown that a deep neural network outperforms the state-of-the-art in image classification in the ImageNet competition, by a large margin [34]. Since then, interest has skyrocketed, and the number of publications on AI has grown exponentially [70].

A vital ingredient of this success was the availability of computational resources for AlexNet: two NVIDIA GeForce GTX 580 graphics processing units

⋆ equal contribution

(GPUs). Since then, not only the cumulated amount of compute has increased tremendously, but also the computational resources needed by AI models. Amodei et. al observed a 300 000-fold increase in six years of the computational effort required for the largest models [6]. This trend extends over all fields of machine learning (ML), ranging from Computer Vision (CV), over Natural Language Processing (NLP) to Reinforcement learning (RL).

A prominent example with a lot of media echo is the GPT-3 model, a transformer-based NLP architecture consisting of 175 billion parameters trained on a corpus of 45 TB of text [11]. The impressive NLP model is capable of performing "in-context learning", i.e. it can pick up novel tasks at inference time without re-training by just providing an input that verbally describes the task or gives the context of the desired outcome.

In the CV field, the BigTransfer (BiT) work follows similar lines. A very large model is pre-trained on an extremely large dataset: up to 300 million images, allowing for fine-tuning the model in an extremely data-efficient way for downstream tasks [33]. This idea is further detailed in Section 3.1. Self-supervised approaches make it possible to also use data that has not been manually annotated for pre-training [13]. Such approaches heavily rely on large models and extensive training procedures but have the potential to reach learning from only a few examples. An even greater source of visual data enclosing also the temporal dimension are videos as shown e.g. in Ref. [44].

While the previous examples involve the private sector, also in the academic world, large scale training have been performed. Kurth et. al trained an ML-based large scale climate model on the Piz Daint and Summit supercomputers, reaching 1 EFlop/s peak in FP16 [35]. Other typical situations in science are inverse problems, in which an accurate forward simulation model is available, but the practical task is the inference of parameters. In an example of Laanait et al., electron densities were inferred from diffraction patterns by an ML model, a computation that involved a training data set of 500 TB and up 27 600 GPUs on Summit [36]. A further example is the evolutionary search of network architectures in cancer research [46]. Here, the objective was to find an architecture that could analyze microscopic tissue slides to support cancer diagnosis and that is suitable for usage in a desktop PC application, requiring an extensive search for the optimal model.

Reproducibility is an indispensable factor for AI research [59]. While many academic papers are accompanied with dedicated repositories, reproducing the largest models from the industry sector are especially difficult, for neither of the examples GPT-3, AlphaZero, AlphaFold or DALL-E the full source code or trained models have been released. Initiatives dedicated to reproducing such results can make very important contributions to the field, yet require considerable computational resources [4].

To support these trends, a new supercomputer has been installed at the Jülich Supercomputing Centre (JSC) for Forschungszentrum Jülich. The machine is installed as a booster module for the modular supercomputer JUWELS (Jülich

[4] See e.g. `https://github.com/EleutherAI/the-pile`

Wizard for European Leadership Science) and referred to as JUWELS Booster. Almost 4000 high-performance GPUs render it both one of the fastest and most energy efficient computers in the world, currently ranked no. 7 in the Top500 list and no. 3 in the Green500 list, as well as the fastest supercomputer in Europe. It is intended as a machine that not only supports AI research, but can act as a substrate for a blooming AI community.

In this paper, we will explain the rationale behind the system design, indicate its technological capabilities, and demonstrate its potential by showing several research highlights from Forschungszentrum Jülich. This paper is structured as follows. After the introduction in Section 1, we present JUWELS' ecosystem in Section 2. Here, we discuss its architecture, parallel model training techniques and ML benchmark results. In Section 3, we present four examples of large scale AI applications. This includes large scale transfer learning, deep-learning driven weather forecast, multispectral remote sensing image classification, and RNA structure analysis. We conclude with a summary and an outlook.

2 JUWELS Booster System

2.1 JSC Supercomputer Ecosystem

The Jülich Supercomputing Centre operates different supercomputers of various sizes. The two largest ones are JURECA and JUWELS. JUWELS is a *Gauss Centre for Supercomputing* Tier 0/1 machine [30], currently consisting of two modules: JUWELS Cluster and JUWELS Booster. While the JUWELS Cluster module provides general-purpose computational resources with more than 2300 compute nodes based on Intel Skylake CPUs, JUWELS Booster is the system's highly scalable module, leveraging GPUs to provide computing performance. Both modules are combined through their network fabric and file system and can be used together, by heterogeneous jobs, through a tight integration via the workload manager.

2.2 JUWELS Booster

Commissioned in 2020, JUWELS Booster features the latest GPU, CPU, and network technology available. 936 compute nodes host four GPUs each, providing access to 3744 GPUs. The installed GPUs are NVIDIA A100 Tensor Core GPUs (40 GB), providing 19.5 TFLOP/s of $\mathrm{FP64_{TC}}$ computing performance each. The GPUs are hosted by AMD EPYC 7402 CPUs with 2×24 cores (SMT-2) per node. Each node is equipped with 512 GB of RAM.

The network of JUWELS Booster is based on Mellanox HDR200 InfiniBand, with four Mellanox ConnectX 6 devices per node, each providing 200 Gbit/s of bandwidth per direction. The network is designed as a DragonFly+ network; the nodes are aligned in sets of 48 in a local switch group (*cell*). While nodes in a cell are tightly connected as a full fat-tree with two levels of switches, each cell connects to the other cell with 10 links. The resulting total bi-section bandwidth

is 400 Tbit/s between the cells. Dedicated network links are available to connect to JUWELS Cluster. These links also provide access to a highly-parallel, flash-based file system with 1400 GB/s peak bandwidth. The storage cluster, JUST, can be reached with a peak of 400 GB/s bandwidth via gateway nodes.

The NVIDIA A100 GPUs installed into JUWELS Booster provide different computing performance depending on the precision used. Within the 400 W TDP, the following peak performance is available: 9.7 TFLOP/s (FP64), 19.5 TFLOP/s FP64_{TC} and FP32, 78 TFLOP/s FP16, 156 TFLOP/s TF32_{TC}, 312 TFLOP/s FP16_{TC}, where *TC* denotes the usage of Tensor Cores. With respect to the FP64 Tensor Cores, an excellent peak efficiency of 48.75 GFLOP/(s W) can be reached. Indeed, JUWELS Booster ranks highest in the Green500 list of November 2020 as the most energy-efficient supercomputer within the first 100 places of the Top500 with 25 GFLOP/(s W).

The software stack on JUWELS is managed via EasyBuild and accessed via environment modules, enabling finely-tuned, homogeneous software environments across the modules of the system. In addition, containers are supported via Singularity, either by using pre-made containers from registries, or by building containers directly from the system through a dedicated container build service.

2.3 Distributed Model Training on JUWELS Booster

The execution of efficient AI algorithms on JUWELS Booster requires different levels of parallelism: At the level of basic numerical operations, at the level of parallelization of deep learning models, and at the level in the training procedure of the deep learning models.

Modern deep learning models transform n-dimensional tensors by applying element-wise operations, e.g. activation functions, convolution operations, or matrix multiplication in fully connected layers. While element-wise operations are embarrassingly parallelizable, convolution operations and matrix multiplication require special, sophisticated parallelization strategies. The development of parallel matrix operations and convolutions that can themselves be reduced to matrix operations is mature and results in highly optimized libraries such as MKL [1], cuBLAS [2], and cuDNN [15]. JUWELS Booster's GPUs contain Tensor Cores, hardware that specializes in matrix multiplications in the context of AI. In conjunction with cuBLAS, it is possible to apply efficient convolution operations and matrix multiplications in GPUs. Depending on the dimension of the input tensor and its data type, the most efficient algorithm is automatically selected by the cuBLAS library. While it is not necessary to optimize these operations manually, AI applications can be tuned by choosing optimal input dimensions and data types, i.e. Tensor Cores work most efficiently when the data dimension is divisible by a certain number depending on the data type. Reducing the precision of the data type, for example from FP32 to FP16, can also lead to a significant speed-up. The deep learning frameworks installed on JUWELS Booster (TensorFlow [4] and PyTorch [45]) provide modules (e.g. `torch.cuda.amp` in PyTorch) to automatically reduce precision for operations where precision is a minor concern. While operator-level parallelization is performed on a single GPU, model-

level parallelization enables usage of all 3744 GPUs of the JUWELS Booster system. A common method for parallelizing the training of neural networks is data parallelism. This involves replicating the model on several GPUs, each of which is trained with a different batch of the training data. Apart from batch normalization [29], all operations in modern deep learning models can be performed independently on the different computing devices. After calculating the gradient of the model parameters, the gradients are averaged over the different replicants, which effectively gives the same result as training a model on a large batch – the combination of all distributed data batches. The averaging of the gradients as well as the batch normalization operation require collective communication across different GPUs, and can become a bottleneck when scaling the training process. To overcome these problems, one usually changes the training procedure in a distributed environment. For example for batch normalization, one computes the batch statistics only on a subset of the parallel training batches involved, which has even been shown to be advantageous over computing over exceptionally large total batches [24]. Collective communication can be accelerated by compressing the gradients before averaging [21,64,5]. The JUWELS Booster software stack provides Horovod [55] as a data-parallel framework with NCCL[5] as a communication framework that works with the supported deep learning frameworks TensorFlow and PyTorch, and comes with built-in FP16 gradient compression. Horovod is applied for the parallel training results shown in this paper. JSC also supports HeAT [26], a general distributed tensor framework for high-performance data analytics. The communication framework of HeAT provides the full MPI [42] functionality and shows a significantly speedup to the comparable library Dask [50]. Large deep learning models may not fit on a single computational device, requiring an extension of the purely data-parallel approach to model parallelism [43] or pipelining [20], which in turn requires automatic differentiation (AD) across different computational devices. AD is a critical feature of modern deep learning libraries, but most libraries[6] support AD only on a single node, and no deep learning library supports AD with MPI requiring extensions. JSC supports DeepSpeed [49], a cutting-edge deep learning optimisation library that supports data and model parallelisation as well as pipelining schemes or all in combination by a lightweight wrapper around PyTorch. In addition JSC supports MPI4Torch[7] allowing to write PyTorch code directly in distributed environments. The last level of parallelization considers the parallelization of the training procedure, which includes ensemble learning [38], model averaging [71], but also hyperparameter search [40], as well as model architecture search [39]. JSC supports training parallelization with the framework L2L [60].

[5] https://docs.nvidia.com/deeplearning/nccl/index.html

[6] PyTorch allows AD for distributing tensors across computational devices based on the remote procedure call (RPC) protocol [9]. However, the RPC framework does not compete with communication frameworks like NCCL or MPI with respect to performance.

[7] https://github.com/helmholtz-analytics/mpi4torch

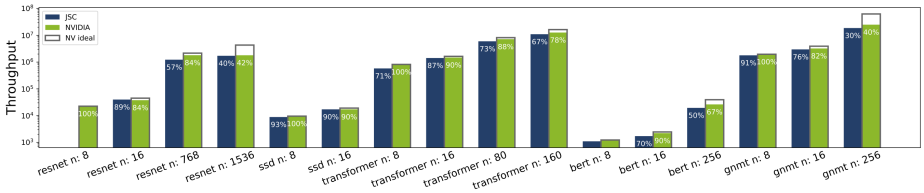


Fig. 1: Benchmark results of a subset of MLPerf training v0.7 runs for different tasks and different number of GPUs (n). Our results shown in blue, NVIDIA's in green. Empty bars indicate the throughput for ideal scaling conditions on NVIDIA's results. For each run the efficiency normalized by NVIDIA's single-node result is indicated in percent.

2.4 Benchmark Results

We investigate the capability of JUWELS Booster to achieve nominal performance under practical machine learning conditions by running the MLPerf training benchmark following the v0.7 submission conditions [41]. As our machine shares key features with NVIDIA's Selene machine, we also run their submission code. To enable comparison, the number of nodes is doubled, as Selene features eight GPUs per node over JUWELS Booster's four. Containers are updated as well. The MLPerf training benchmark measures the time-to-accuracy. This was carefully optimized by NVIDIA by hyperparameter optimization. As we only re-run their benchmarks and do not optimize the hyperparameters, we instead report the throughput in images/second for the tasks *resnet* and *ssd*, words per second for the tasks *transformer* and *gnmt*, and in sequences per second for "bert". As indicated in Fig. 1, we are able to closely reproduce NVIDIA's results. Details of our implementation are published on our GitLab server[8].

3 Large Scale AI Research at JSC

3.1 Large-Scale Deep Learning for Efficient Cross-Domain Transfer

Transfer learning was successfully employed already at the very rise of deep neural networks. Early architectures like AlexNet, OverFeat, or VGG-16 were pre-trained on ImageNet-1k, a large natural image dataset which serves as a gold standard in the visual image understanding community [19,51]. Importantly, after fine-tuning on various other target datasets, pre-trained models showed better performance compared to models just trained on target datasets from scratch [47]. The combination of pre-training on a large generic dataset and transferring in an efficient way the pre-trained model to a specific target dataset, often much smaller in size than the one used during pre-training, has since proven

[8] `https://gitlab.version.fz-juelich.de/kesselheim1/mlperf_juwelsbooster`

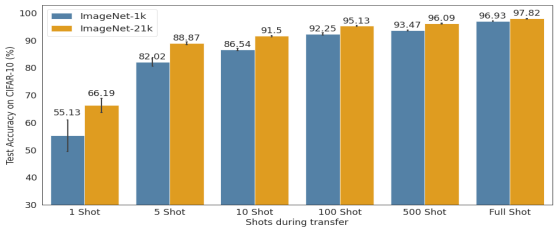


Fig. 2: Fine-tuning results of ResNet-152x4 on CIFAR-10 using pipeline of [33]. The model is pre-trained either on ImageNet-1k or ImageNet-21k, and we show few-shot (e.g. 1-shot means we use 1 example per class, thus a total of 10 examples for CIFAR-10) results as well as full fine-tuning results (the whole training set is used for fine-tuning).

	Precision	Recall	F1-score
COVID-19	0.88	0.84	0.86
Normal	0.96	0.92	0.94
Pneumonia	0.87	0.93	0.90

Table 1: Fine-tuning results of ResNet-152x4 on the COVIDx dataset (we use the COVIDx V7A version) from [65]. In this setup, we pre-train on ImageNet-1k and follow the fine-tuning pipeline of [33].

itself as a viable strategy to create powerful models also for those scenarios where data is scarce [33].

Recently, strong evidence was obtained that increasing the model size while at the same time increasing the amount of data and compute for pre-training results in very large models that have even stronger generalization and transfer capabilities [8,32,10,33,27]. Pre-training of large transferable models requires larger datasets, like for instance ImageNet-21k [19], and is computationally very expensive. Therefore, to obtain such large models and benefit from their improved transferability and generalization, machines like JUWELS Booster are required to perform distributed training efficiently across multiple nodes using a large amount of GPUs.

Here, we demonstrate the merits of such a large-scale distributed pre-training for transfer on different target datasets following [33]. We show that pre-training on ImageNet-21k ($\approx$ 10 times larger than the standard ImageNet-1k) provides a clear performance benefit in terms of accuracy achieved when transferring and fine-tuning the pre-trained model to smaller natural image datasets like CIFAR-10. Especially in the very low data regime using few-shot transfer, where only few examples per class are shown, the benefit of pre-training on large ImageNet-21k is striking (see Figure 2). JUWELS Booster shows good scaling behavior during distributed training across a vast number of compute nodes using Horovod (Fig. 1), which allows us to execute large-scale pre-training procedures in a fraction of the time for single-node training and shorten experimental cycles. Still, a full pre-training of a large ResNet-152x4 network on ImageNet-21k for 90 epochs takes ca. 81 hours when using 256 GPUs. Distributed training also performs without loss of accuracy [56] when compared to single-node training.

Following the goals of our initiative for large-scale transfer learning applied to medical imaging for COVID-19 diagnostics (COVIDNetX[9]), we also envisage application of large-scale generic model pre-training, for efficient and robust transfer to specific domains, like medical images. Motivated by the urgency to provide robust and widely available tools for predictive diagnostics of a patient's current and future state from medical imaging data with regard to a SARS-CoV-2 infection and its disease course [66,58], we have chosen a small publicly available dataset containing X-ray lung images of COVID-19, non-COVID, and healthy control patients (COVIDx) as a use case example for transfer [65,16]. First preliminary results for transfer performance are available for the large ResNet-152x4 model pre-trained on ImageNet-1k (see Table 1). Further investigations are necessary to quantify the benefits of pre-training on much larger datasets, like ImageNet-21k, when attempting to transfer to such small, domain-specific datasets like COVIDx (see e.g. [14] for a follow-up study).

In summary, we have prepared the grounds for advanced transfer learning techniques, which should allow us to efficiently transfer models pre-trained on large amounts of generic data, in a highly-performant, distributed manner, to various specific target datasets of much smaller size, with low computational cost for transfer [33]. Transfer efficiency also paves the road for energy efficient deep learning that is both data-sample and compute efficient, requiring only a small energy budget for each transfer. As follow-up work, we aim to demonstrate that the postulated benefits of better generalization and transfer attributed to large-scale models [8,32,10,33,27] hold across a very broad range of specialized domains and conditions used as transfer targets.

3.2 Deep Learning-Driven Weather Forecast

Weather can have an enormous effect on human lives. Improving weather forecasting can minimize the adverse effects of extreme weather and assist in planning economic activities. Numerical weather prediction (NWP) models are among the earliest and most demanding applications for supercomputers [7]. Recently, modern deep learning (DL) approaches are considered to potentially play an important role in every step of the NWP workflow, from data assimilation, to replacing numerical model parameterizations, to statistical output post-processing [7,48]. Because of the huge size of meteorological datasets from observations and numerical simulations, high-throughput supercomputing systems and parallel DL approaches are necessary.

In this study, we explore the use of state-of-the-art video prediction methods to forecast meteorological variables utilizing the global ERA5 reanalysis dataset [28]. The goal is to forecast the 2-metre temperature over Europe for 12 hours in a data-driven way. The geographic domain consists of 56×92 grid points in meridional and zonal direction. Input variables are the 2-metre temperature, cloud cover, and 850 hPa temperatures of the preceding 12 hours. The first DL model selected is the convLSTM architecture [57]. Input and output tensors

[9] `https://tinyurl.com/CovidNetXHelmholtz`

of our convLSTM model have a dimension of $12 \times 56 \times 92 \times 3$ each. The model has 429 251 parameters and was trained on 11 years of ERA5 reanalysis data in hourly resolution. The total volume of pre-processed data in TFRecords format amounts to 153 GB. An example of a 12-hour forecast is shown in Figure 3.

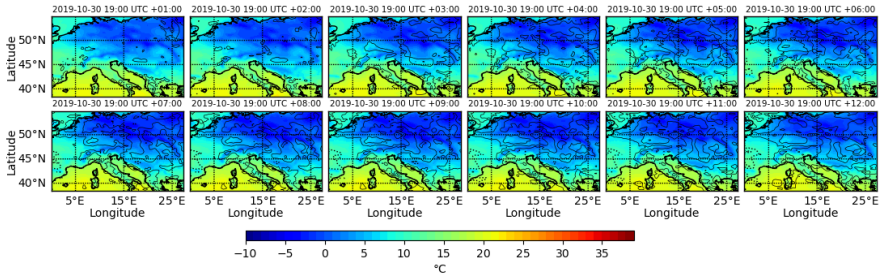


Fig. 3: Example of a 2-metre temperature (°C) prediction with convLSTM (see text for details). Solid (dotted) contours denote positive (negative) temperature differences between forecast and ground truth with an interval of 1 K (starting from 0.5 K).

Training on a single A100 GPU takes about 50 min/epoch. Given that a typical DL experiment typically requires up to 100 epochs to converge, such training times are prohibitive for any serious application, so training is parallelized using Horovod. Figure 4 shows that the model training achieves 90 % scaling efficiency in terms of time comparing 1 GPU against 16 GPUs for 10 epochs. However, it is observed that time variances for all iterations increase significantly beyond 32 GPUs. This could be caused by data loading inefficiency, communication, or lack of enough GPU utilization. These issues are currently being investigated. After solving these issues, we plan to test more complex video prediction models (e.g. the stochastic adversarial video prediction architecture [37]) to improve prediction accuracy and push JUWELS Booster to its limits.

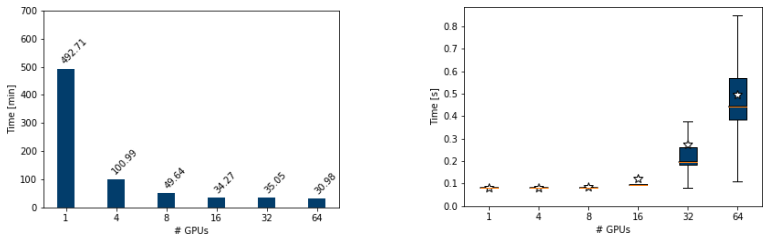
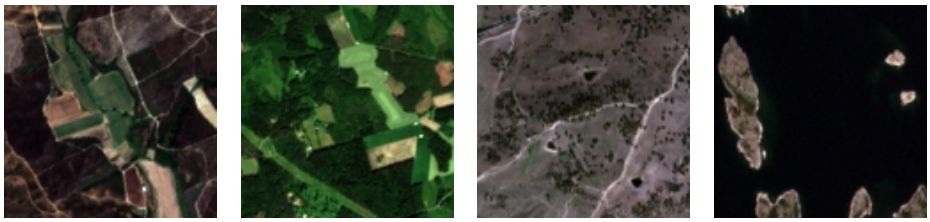


Fig. 4: Total training time in minutes (left) and box whisker plot of iteration time in seconds (right). The star in the box whisker plot denotes the averaged iteration time, while its median is highlighted by an orange line.

3.3 Multispectral Remote Sensing Image Classification

Among other application areas, spaceborne Remote Sensing (RS) can detect and observe the characteristics of the Earth's surface by measuring its reflected and emitted radiation at a distance. Applications from RS use multispectral RS images to classify different physical features that occupy the surface of the Earth (i.e. land-cover classes), as well as to describe the use of the land surface by humans (i.e. land-use classes) [12]. The RS community has started to produce several labeled RS datasets with large spatial coverage and variety of classes (e.g. BigEarthNet [61], SEN12MS [52], etc.). These datasets provide a high number of reliably labeled samples that can be used to train deep neural networks with supervised learning.

BigEarthNet-S2[10] is a large RS archive consisting of 590 326 patches (an example is shown in Fig. 5) extracted from tiles acquired by the Sentinel-2 satellites [62]. We train the models on the version of the dataset with 19 labels. Each patch can be associated with multiple labels, making this a multi-label classification problem. For the experiments we use the following spectral bands: 3 RGB bands and band 8 at 10 m resolution, bands 5, 6, 7, 8a, 11, and 12 at 20 m resolution, and bands 1 and 9 at 60 m resolution. The bands at lower resolution are upsampled to 10 m resolution using bilinear interpolation.



(a) (b) (c) (d)

Fig. 5: Example of patches extracted from Sentinel-2 tiles [62]. The corresponding classes are: (**a**) "Permanent crops", "Broad-leaved forest", "Transitional woodland/shrub", (**b**) "Land principally occupied by agriculture, with significant areas of natural vegetation", "Broad-leaved forest", "Mixed forest", (**c**) "Pastures", "Agro-forestry areas", (**d**) "Mixed forest", "Marine waters".

A multispectral ResNet-152 is trained from scratch on the training subset (60 % of the entire dataset) and evaluated on the test subset (20 % of the dataset). We run the experiments with the NovoGrad optimizer. The values of the learning rate and weight decay follow the choices of [23]. Data augmentations with random flips, rotation of the patches, and mix-up are applied to reduce overfitting. The experiments are carried out for 100 epochs, on 1, 4, 16, and 64 nodes (4, 16, 64 and, 256 GPUs respectively). Horovod is employed to distribute the training

[10] http://bigearth.net/

on multiple compute nodes and GPUs [55]. The batch size per GPU selected for the experiments is 16, with a global batch size that ranges from 64 for 1-node to 4096 for the 64-node configurations respectively.

The results of the classification are evaluated using the macro F1 score, which remains stable among the experiments (0.73), and is in line with [62]. Using a data-parallel approach allows us to scale up a DL model on a large remote sensing dataset and consequently cut the training time down significantly, with an 80 % efficiency comparing 1 node (ca. 2550 s per epoch) against 64 nodes (ca. 50 s per epoch). Various research questions remain open, and one salient point is understanding the feasibility of training with larger global batch sizes, which are known to pose optimization difficulties [25]. A comparison between different training strategies, such as the choice of the learning rate and optimizer, is also in the future plans of the authors. More effort is also needed to enhance the pre-processing and data loading pipeline to feed the model and possibly increase efficiency when using a large number of nodes.

3.4 RNA Structure with ML

On a fundamental level, all life as we know it is orchestrated by interactions between biomolecules, such as proteins, DNA, and RNA. Due to a direct structure-function relationship, it is important to structurally resolve biomolecules, even though this is experimentally challenging. A complementary approach is using statistical tools to mine the rich existing protein databases. Physics-based co-evolutionary models such as direct coupling analysis (DCA) [67,53,68] have in the last decade lead to the prediction of protein structures and complexes with astonishing accuracy [18,63] by a combination of bioinformatics tools with molecular simulations. Considering the aforementioned large, public databases with 100 000+ biomolecular structures and even larger sequence databases, the combination of these methods with machine learning (ML) approaches appears natural and has further improved their accuracy [54].

Another class of biomolecules, Ribonucleic acids (RNA), is critical for biological activities such as coding, regulation and expressions of genes. This critical importance also leads to RNA's application in pharmacology, as some of the most effective drugs in the current COVID epidemic are RNA-based. As for proteins, RNA function is closely related to its three-dimensional structure. It is, however, more challenging to gain structural information on RNA, as reflected by the small number of RNA 3-D structures in databases with many crucial RNA being still structurally unresolved – akin to dark matter of the biomolecular universe [3]. Unfortunately, while DCA still works well on RNA [22,17,3], ML methods so successful for proteins cannot be easily applied for RNA structure prediction, given that existing databases are considerably smaller [31]. Still, even the small amount of existing data can be used to significantly improve prediction of RNA by shallow neural networks by over 70 % using simple convolutional neural networks [69]. Future work will therefore focus on using the massive computing resources of JUWELS to enhance these simple CNNs by, e.g. using knowledge gained from proteins by transfer learning approaches. Similarly,

we plan to run complex molecular dynamics simulations (MD) on RNA, to gain more structural insight. As these simulations needs to simulate all atoms of the RNA and the surrounding solvent, they can quickly require $\mathcal{O}(10)+$ million core hours on highly parallel systems.

4 Summary and Outlook

In this paper we have demonstrated the hardware configuration of JUWELS Booster, currently not only the fastest supercomputer in Europe but also the most energy-efficient large-scale machine in the world. We have elucidated the key parallelization techniques that unlock this computational power for machine learning, and have shown that JUWELS Booster reaches its potential also in practical machine learning settings.

We have indicated how the availability of such a machine can foster developments in various research fields. We have shown that large-scale pre-training can render CV applications much more data-efficient, including medical imaging for COVID-19 detection. We demonstrated that DL video prediction methods are an option for computing weather forecasts. We have seen that large-scale data parallel training enables us to train models on large corpora of multispectral satellite images and the ML-based structure predictions can help us understanding the role of RNA for biology.

All applications not only rely on raw computational power, but also on a system design and infrastructure that allows for fast parallel implementations, and a software ecosystem that gives users the possibility for rapid development, employing groundbreaking methods. The Jülich Supercomputing Centre, hosting JUWELS Booster, brings all these ingredients together, creating a state-of-the art AI landscape.

Acknowledgements

This work was funded by Helmholtz Association's Initiative and Networking Fund under project number ZT-I-0003 and HelmholtzAI computing resources (HAICORE) Funding has been obtained through grants ERC-2017-ADG 787576 (IntelliAQ) and BMBF 01 IS 18O47A (DeepRain). This work was performed in the CoE RAISE and DEEP-EST projects receiving funding from EU's Horizon 2020 Research and Innovation Framework Programme under the grant agreement no. 951733 and no. 754304 respectively. We thank ECMWF for providing ERA-5 data. The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this work by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputers JUWELS, JUWELS Booster at Jülich Supercomputing Centre (JSC) and we acknowledge computing resources from the Helmholtz Data Federation. Further computing time was provided on supercomputer JUSUF in frame of offer for epidemiology research on COVID-19 by JSC.

References

1. Intel Math Kernel Library. Reference Manual. Intel Corporation (2009)
2. NVIDIA CUBLAS Library Documentation [accessed at 2021-04-14]. (2017), https://docs.nvidia.com/cuda/cublas/
3. Shedding light on the dark matter of the biomolecular structural universe: Progress in rna 3d structure prediction. Methods **162-163**, 68 – 73 (2019). https://doi.org/https://doi.org/10.1016/j.ymeth.2019.04.012
4. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), http://tensorflow.org/, software available from tensorflow.org
5. Agarwal, S., Wang, H., Venkataraman, S., Papailiopoulos, D.: On the utility of gradient compression in distributed training systems. ArXiv **abs/2103.00543** (2021)
6. Amodei, D., Hernandez, D., Sastry, G., Clark, J., Brockman, G., Sutskever, I.: Ai and compute. Tech. rep., OpenAI Blog (2018)
7. Bauer, P., Thorpe, A., Brunet, G.: Nature . https://doi.org/10.1038/nature14956
8. Belkin, M., Hsu, D., Ma, S., Mandal, S.: Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences of the United States of America **116**, 15849–15854 (Aug 2019). https://doi.org/10.1073/pnas.1903070116
9. Birrell, A.D., Nelson, B.J.: Implementing Remote Procedure Calls. ACM Transactions on Computer Systems **2**(1), 39–59 (1984). https://doi.org/10.1145/2080.357392, https://doi.org/10.1145/2080.357392
10. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020)
11. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
12. Canty, M.: Image Analysis, Classification and Change Detection in Remote Sensing: With Algorithms for ENVI/IDL and Python, Third Edition. Taylor & Francis (2014), ISBN: 9781466570375
13. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.: Big self-supervised models are strong semi-supervised learners. arXiv preprint arXiv:2006.10029 (2020)
14. Cherti, M., Jitsev, J.: Effect of large-scale pre-training on full and few-shot transfer learning for natural and medical images. arXiv preprint arXiv:2106.00116 (2021)
15. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., Shelhamer, E.: cudnn: Efficient primitives for deep learning (2014)
16. Cohen, J.P., Morrison, P., Dao, L., Roth, K., Duong, T.Q., Ghassemi, M.: Covid-19 image data collection: Prospective predictions are the future. Journal of Machine Learning for Biomedical Imaging (2020)

17. Cuturello, F., Tiana, G., Bussi, G.: Assessing the accuracy of direct-coupling analysis for rna contact prediction (2020). https://doi.org/10.1261/rna.074179.119
18. Dago, A.E., Schug, A., Procaccini, A., Hoch, J.A., Weigt, M., Szurmant, H.: Structural basis of histidine kinase autophosphorylation deduced by integrating genomics, molecular dynamics, and mutagenesis. Proceedings of the National Academy of Sciences **109**(26), E1733–E1742 (2012)
19. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: Imagenet: A large-scale hierarchical image database. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. pp. 248–255 (Jun 2009). https://doi.org/10.1109/CVPR.2009.5206848
20. Deng, L., Yu, D., Platt, J.: Scalable stacking and learning for building deep architectures. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2133–2136 (2012). https://doi.org/10.1109/ICASSP.2012.6288333
21. Dettmers, T.: 8-bit approximations for parallelism in deep learning (2015), `http://arxiv.org/abs/1511.04561`, cite arxiv:1511.04561
22. De Leonardis, E., Lutz, B., Ratz, S., Cocco, S., Monasson, R., Schug, A., Weigt, M.: Direct-Coupling Analysis of nucleotide coevolution facilitates RNA secondary and tertiary structure prediction. Nucleic Acids Research **43**(21), 10444–10455 (09 2015). https://doi.org/10.1093/nar/gkv932
23. Ginsburg, B., Castonguay, P., Hrinchuk, O., Kuchaiev, O., Lavrukhin, V., Leary, R., Li, J., Nguyen, H., Zhang, Y., Cohen, J.M.: Stochastic gradient methods with layer-wise adaptive moments for training of deep networks (2020)
24. Goyal, P., Dollár, P., Girshick, R.B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR **abs/1706.02677** (2017), `http://arxiv.org/abs/1706.02677`
25. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour (2018)
26. Götz, M., Debus, C., Coquelin, D., Krajsek, K., Comito, C., Knechtges, P., Hagemeier, B., Tarnawa, M., Hanselmann, S., Siggel, M., Basermann, A., Streit, A.: HeAT – a Distributed and GPU-accelerated Tensor Framework for Data Analytics. In: Proceedings of the 19th IEEE International Conference on Big Data. pp. 276–288. IEEE (December 2020)
27. Hernandez, D., Kaplan, J., Henighan, T., McCandlish, S.: Scaling laws for transfer. arXiv preprint arXiv:2102.01293 (2021)
28. Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al.: The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society **146**(730), 1999–2049 (2020). https://doi.org/https://doi.org/10.1002/qj.3803
29. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (07–09 Jul 2015), `http://proceedings.mlr.press/v37/ioffe15.html`
30. Jülich Supercomputing Centre: JUWELS: Modular Tier-0/1 Supercomputer at the Jülich Supercomputing Centre. Journal of large-scale research facilities **5**(A171) (2019), `http://dx.doi.org/10.17815/jlsrf-5-171`
31. Kalvari, I., Argasinska, J., Quinones-Olvera, N., Nawrocki, E.P., Rivas, E., Eddy, S.R., Bateman, A., Finn, R.D., Petrov, A.I.: Rfam 13.0: shifting to a genome-centric

resource for non-coding RNA families. Nucleic Acids Research **46**(D1), D335–D342 (11 2017). https://doi.org/10.1093/nar/gkx1038
32. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020)
33. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 491–507. Springer International Publishing, Cham (2020)
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
35. Kurth, T., Treichler, S., Romero, J., Mudigonda, M., Luehr, N., Phillips, E., Mahesh, A., Matheson, M., Deslippe, J., Fatica, M., et al.: Exascale deep learning for climate analytics. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 649–660. IEEE (2018)
36. Laanait, N., Romero, J., Yin, J., Young, M.T., Treichler, S., Starchenko, V., Borisevich, A., Sergeev, A., Matheson, M.: Exascale deep learning for scientific inverse problems. arXiv preprint arXiv:1909.11150 (2019)
37. Lee, A.X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., Levine, S.: Stochastic adversarial video prediction. arXiv preprint arXiv:1804.01523 (2018)
38. Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D.J., Batra, D.: Why M heads are better than one: Training a diverse ensemble of deep networks. CoRR **abs/1511.06314** (2015), `http://arxiv.org/abs/1511.06314`
39. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical Representations for Efficient Architecture Search. arXiv e-prints arXiv:1711.00436 (Nov 2017)
40. Lorenzo, P.R., Nalepa, J., Ramos, L., Ranilla, J.: Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. Proceedings of the Genetic and Evolutionary Computation Conference Companion (2017)
41. Mattson, P., Reddi, V.J., Cheng, C., Coleman, C., Diamos, G., Kanter, D., Micikevicius, P., Patterson, D., Schmuelling, G., Tang, H., et al.: Mlperf: An industry standard benchmark suite for machine learning performance. IEEE Micro **40**(2), 8–16 (2020)
42. Message Passing Interface Forum: MPI: A Message-Passing Interface Standard, Version 3.1. High Performance Computing Center Stuttgart (HLRS) (2015), `https://fs.hlrs.de/projects/par/mpi//mpi31/`
43. Muller, U.A., Gunzinger, A.: Neural net simulation on parallel computers. In: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94). vol. 6, pp. 3961–3966 vol.6 (1994). https://doi.org/10.1109/ICNN.1994.374845
44. Orhan, E., Gupta, V., Lake, B.M.: Self-supervised learning through the eyes of a child. Advances in Neural Information Processing Systems **33** (2020)
45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`

46. Patton, R.M., Johnston, J.T., Young, S.R., Schuman, C.D., Potok, T.E., Rose, D.C., Lim, S.H., Chae, J., Hou, L., Abousamra, S., et al.: Exascale deep learning to accelerate cancer research. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 1488–1496. IEEE (2019)
47. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: An astounding baseline for recognition. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops. pp. 512–519 (Jun 2014). https://doi.org/10.1109/CVPRW.2014.131
48. Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., Prabhat: Deep learning and process understanding for data-driven Earth system science. Nature (2019). https://doi.org/10.1038/s41586-019-0912-1
49. Ren, J., Rajbhandari, S., Aminabadi, R.Y., Ruwase, O., Yang, S., Zhang, M., Li, D., He, Y.: Zero-offload: Democratizing billion-scale model training (2021)
50. Rocklin, M.: Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In: Huff, K., Bergstra, J. (eds.) Proceedings of the 14th Python in Science Conference (SciPy 2015). pp. 130–136 (2015)
51. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
52. Schmitt, M., Hughes, L.: Sen12ms
53. Schug, A., Weigt, M., Onuchic, J.N., Hwa, T., Szurmant, H.: High-resolution protein complexes from integrating genomic information with molecular simulation. Proceedings of the National Academy of Sciences **106**(52), 22124–22129 (2009)
54. Senior, A.W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A.W.R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D.T., Silver, D., Kavukcuoglu, K., Hassabis, D.: Improved protein structure prediction using potentials from deep learning. Nature **577**(7792), 706–710 (Jan 2020). https://doi.org/10.1038/s41586-019-1923-7
55. Sergeev, A., Balso, M.D.: Horovod: Fast and Easy Distributed Deep Learning in TensorFlow. arXiv preprint arXiv:1802.05799 (2018)
56. Shallue, C.J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., Dahl, G.E.: Measuring the effects of data parallelism on neural network training. Journal of Machine Learning Research **20**, 1–49 (2019)
57. Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems (2015)
58. Sriram, A., Muckley, M., Sinha, K., Shamout, F., Pineau, J., Geras, K.J., Azour, L., Aphinyanaphongs, Y., Yakubova, N., Moore, W.: Covid-19 deterioration prediction via self-supervised representation learning and multi-image prediction. arXiv preprint arXiv:2101.04909 (2021)
59. Stodden, V., McNutt, M., Bailey, D.H., Deelman, E., Gil, Y., Hanson, B., Heroux, M.A., Ioannidis, J.P., Taufer, M.: Enhancing reproducibility for computational methods. Science **354**(6317), 1240–1241 (2016)
60. Subramoney, A., Diaz-Pier, S., Rao, A., Scherr, F., Salaj, D., Bohnstingl, T., Jordan, J., Kopp, N., Hackhofer, D., Stekovic, S.: Igitugraz/l2l: v1.0.0-beta (mar 2019). https://doi.org/10.5281/zenodo.2590760
61. Sumbul, G., Charfuelan, M., Demir, B., Markl, V.: Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS) (2019). https://doi.org/10.1109/igarss.2019.8900532

62. Sumbul, G., Kang, J., Kreuziger, T., Marcelino, F., Costa, H., et al.: BigEarthNet Dataset with A New Class-Nomenclature for Remote Sensing Image Understanding (2020), http://arxiv.org/abs/2001.06372
63. Uguzzoni, G., Lovis, S.J., Oteri, F., Schug, A., Szurmant, H., Weigt, M.: Large-scale identification of coevolution signals across homo-oligomeric protein interfaces by direct coupling analysis. Proceedings of the National Academy of Sciences **114**(13), E2662–E2671 (2017)
64. Vogels, T., Karimireddy, S.P., Jaggi, M.: Powersgd: Practical low-rank gradient compression for distributed optimization. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/d9fbed9da256e344c1fa46bb46c34c5f-Paper.pdf
65. Wang, L., Lin, Z.Q., Wong, A.: Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. Scientific reports **10**, 19549 (Nov 2020). https://doi.org/10.1038/s41598-020-76550-z
66. Wehbe, R.M., Sheng, J., Dutta, S., Chai, S., Dravid, A., Barutcu, S., Wu, Y., Cantrell, D.R., Xiao, N., Allen, B.D., MacNealy, G.A., Savas, H., Agrawal, R., Parekh, N., Katsaggelos, A.K.: Deepcovid-xr: An artificial intelligence algorithm to detect covid-19 on chest radiographs trained and tested on a large u.s. clinical data set. Radiology **299**, E167–E176 (Apr 2021). https://doi.org/10.1148/radiol.2020203511
67. Weigt, M., White, R.A., Szurmant, H., Hoch, J.A., Hwa, T.: Identification of direct residue contacts in protein–protein interaction by message passing. Proceedings of the National Academy of Sciences **106**(1), 67–72 (2009)
68. Zerihun, M.B., Pucci, F., Peter, E.K., Schug, A.: pydca v1. 0: a comprehensive software for direct coupling analysis of rna and protein sequences. Bioinformatics **36**(7), 2264–2265 (2020)
69. Zerihun, M.B., Pucci, F., Schug, A.: Coconet: Boosting rna contact prediction by convolutional neural networks. bioRxiv (2020)
70. Zhang, D., Mishra, S., Brynjolfsson, E., Etchemendy, J., Ganguli, D., Grosz, B., Lyons, T., Manyika, J., Niebles, J.C., Sellitto, M., Shoham, Y., Clark, J., Perrault, R.: The ai index 2021 annual report. Tech. rep., AI Index Steering Committee, Human-Centered AI Institute, Stanford University, Stanford, CA (2021)
71. Zhang, S., Choromanska, A.E., LeCun, Y.: Deep learning with elastic averaging sgd. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 28. Curran Associates, Inc. (2015), https://proceedings.neurips.cc/paper/2015/file/d18f655c3fce66ca401d5f38b48c89af-Paper.pdf