A Computational Workflow for Interdisciplinary Deep Learning Projects utilizing bwHPC Infrastructure

Marcel P. Schilling[1]*, Oliver Neumann[1], Tim Scherr[1], Haijun Cui[2], Anna A. Popova[2], Pavel A. Levkin[2], Markus Götz[3], Markus Reischl[1]
[1]Institute for Automation and Applied Informatics (IAI),
[2]Institute of Biological and Chemical Systems (IBCS),
[3]Steinbuch Centre for Computing (SCC),
Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, Germany.

Abstract

Deep neural networks have the capability to solve complex tasks through accurate function approximation. The process from submitting domain data and defining process requirements to analyzed data consists of multiple steps, disallowing a simplistic straightforward procedure. It follows that one of the core questions is: how does an application development process facilitating interaction between data scientists and domain experts look like? Practically, two connected challenges need to be addressed. Firstly, it requires a solution for handling large amounts of domain-specific data. Secondly, when dealing with complex deep neural networks, it is essential to find a concept of how model training can be designed in an computationally efficient manner. While tailored solutions for addressing these challenges in interdisciplinary deep learning projects exist, a comprehensive and structured approach is missing. Hence, we present a computational workflow to enhance these kinds of projects concerning data handling, integration of cluster computing resources such as bwHPC infrastructure, and development processes. We exemplify our proposal by means of a biomedical image analysis project.

1 Introduction

Deep Neural Networks (DNNs) are often considered for data processing since they are capable to solve complex tasks accurately [9]. Deep Learning (DL) projects are often interdisciplinary since the expertise of both, domain experts (e.g. biologists or energy experts) and data scientists (summarized as actors) are required. The overall goal is to obtain a framework to analyze domain data accurately, smartly, and efficiently. Domain experts have knowledge concerning the data and specify the desired outcome of the data analysis. Data scientists are needed regarding algorithms, computing, or automation.

Nevertheless, there are several challenges impeding the formulated objective. Figure 1 sketches the problem area at the beginning of interdisciplinary DL projects. In general, the development process of DL pipelines is complex, composed of several steps (e.g. problem formulation, labeling, DNN architecture selection/development, model training, hyperparameter optimization, ablation studies, evaluation), and requires the interaction of domain experts and data scientists. For instance, the specified outcome of the data analysis, which is defined by the domain experts, needs to be aligned with algorithms developed by data scientists. Handling large amounts of data is an issue as well. In general, there is no shared data infrastructure between the actors. Consequently, data exchange often relies on individual solutions like external storage devices or e-mail. Closely linked to this and due to the complexity of DNNs, a concept for the best usage of computing infrastructure is of particular interest. Normally, the computing infrastructure of domain experts is not designed for large optimization routines. Local computing devices of data scientists reach their limits as well. Besides, visualizations within DL projects are essential, e.g. for benchmarks, project progress reports, or discussions between actors.

*marcel.schilling@kit.edu

Proceedings of the 7th bwHPC Symposium (ISBN 978-3-948303-29-7)
Article No. 13; pages 69–74

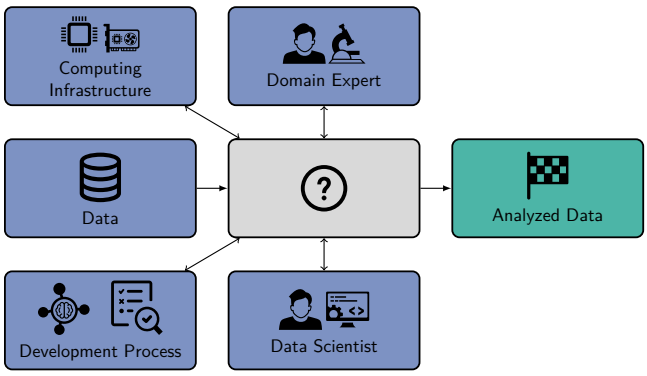


Figure 1: Problem area: A workflow is required considering data handling, computing infrastructure, and the development process of DL processing pipelines in order to analyze data. Thereby, a way of collaboration between domain experts and data scientists needs to be obtained.

There are individual solutions solving parts of the aforementioned problems. However, to our knowledge, a comprehensive workflow focusing on interdisciplinary DL projects is missing.

Our key **contributions** are the following:

- a survey of methods and tools to facilitate interdisciplinary DL projects,
- a comprehensive workflow nested in bwHPC infrastructure serving as a template to enhance interdisciplinary DL projects w.r.t. data handling, integration of cluster computing, or DNN development (e.g. smart logging or hyperparameter search) and to allow focusing more on research questions, and
- a demonstration of our proposal by means of a practical biomedical image processing project.

The **state of the art** is shown in Section 2. Our **workflow** is presented in Section 3. The **results** of application in practical interdisciplinary DL projects are depicted in Section 4. **Conclusions** can be found in Section 5.

2 State of the Art

In the following, related work is presented in the characterized problem clusters: data, computing infrastructure, and development process of DL processing pipelines. To enable a shared storage infrastructure in DL projects, the Large Scale Data Facility (LSDF) [7] can serve as a platform offering large storage capacity and high-speed network connectivity. Consequently, both actors can exchange data conveniently.

During the development process, the domain expert can be supported using tools like a label assistant [14] to obtain training data simplified by various ways of assistance, e.g. pre-labeling or feedback in terms of labeling quality. The wrapper PyTorch Lightning [4] can serve as a template according to the implementation of DNNs since it offers a structured and reusable DL framework. Code versioning and deployment can be performed using the tool Git. Weights & Biases [2] can be utilized concerning logging, hyperparameter tuning, and automated experiment planning. For instance, the framework offers parallelizable state-of-the-art algorithms for hyperparameter optimization like Random Search depicted in [1] inherently.

Considering computing infrastructure, DL training can be boosted via computing clusters as bwHPC infrastructure. Concepts like presented in [3, 6, 8] are needed to make use of High Performance Computing (HPC)/High Throughput Computing (HTC). In general, HTC describes independent computing tasks [6]. In contrast, HPC characterizes dependent computing tasks within a job [6]. Boosting DL training via a single or multiple GPUs by using data parallel training approaches presented in [3, 8] show the potential of HPC in the context of DL. The framework PyTorch Lightning directly provides implementations of data parallel training. However, the main objective of clusters is providing computing devices, the focus is less on natively offering all methods and tools needed for DL projects (e.g. hyperparameter tuning or logging). Furthermore, the hurdle for domain experts is large to use a computing cluster.

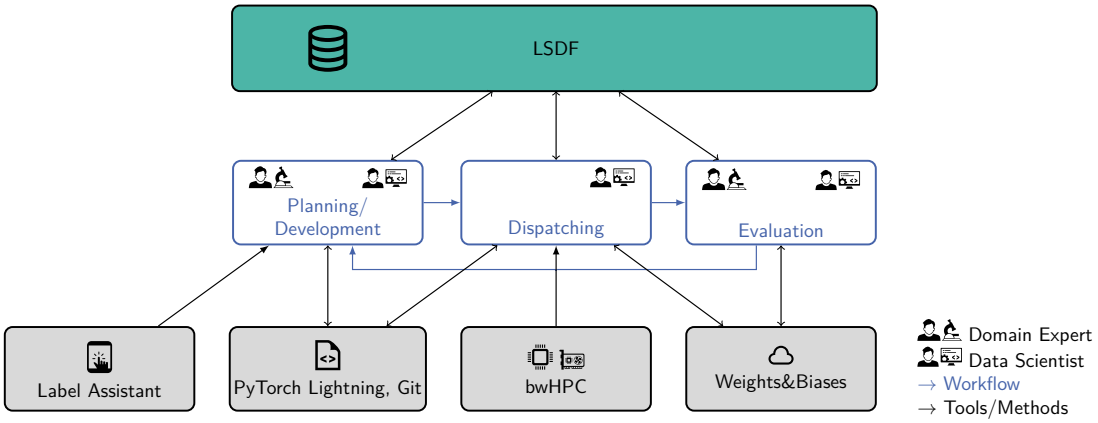


Figure 2: Proposed workflow of DL projects including domain experts and data scientists: The main steps of the proposal are planning/development, dispatching, and evaluation. LSDF serves as the backbone for data handling. bwHPC infrastructure is included in the dispatching step for enhancing computing performance. Tools like Label Assistant, Git, PyTorch Lightning, and Weights&Biases provide support within the workflow.

The **challenges** can be summarized in: (i) a missing definition of a concept, how domain experts and data scientists can interact in DL projects, (ii) a lack of a summary concerning tools and methods which are helpful in this process, and (iii) how a resulting implementation of the workflow connecting the DL development process with bwHPC infrastructure and data handling is realized.

3 Workflow

Our proposed workflow shown in Figure 2 is composed of planning/development, dispatching, and evaluation.

Planning/Development First, the expert can share raw data on LSDF. In the planning/development step, the requirements of the desired processing (known by domain experts) need to be aligned with algorithms (given by data scientists). A label assistant can support the data labeling process. Developing DNNs in PyTorch Lightning decouples research and code engineering by providing a smart DL framework. Research is in the center since many functionalities (e.g. training loop or data loading) are already implemented. Moreover, source code of developed DNNs can be versioned and deployed to bwHPC computing clusters via Git.

Dispatching In order to boost model training and to avoid the need for local infrastructures, we formulate a generic dispatching step, in which the bwHPC cluster computing infrastructure is included. We propose a hybrid computing approach by utilizing the HPC and HTC concepts.

On the HPC branch, training can be accelerated using multiple GPUs in combination with data parallelization techniques. PyTorch Lightning provides Distributed Data Parallel (DDP) and Data Parallel (DP) as parallelization algorithms. Furthermore, the resource management SLURM [15] is supported in PyTorch Lightning.

On the HTC branch, different experiments can be described via Weights&Biases which also offers implementations for hyperparameter search [1]. Using several computation nodes in parallel to boost processing is possible due to an agent system requesting the next parameter setting before starting calculations. Thus, parallelization of different runs can be done very adjustable. The hybrid approach is flexible in two points: scaling towards computing devices (number of GPUs) and subdivision (large studies into smaller experiments). Hence, it is very adaptive concerning the available infrastructure. The LSDF enables the handling of large files resulting from different training setups (e.g. network parameters) without any hurdle.

Evaluation Evaluation is essential during the DL development process. Hence, we propose two approaches to simplify evaluation: the online tool Weights&Biases and the LSDF. Weights&Biases provides a smart logging

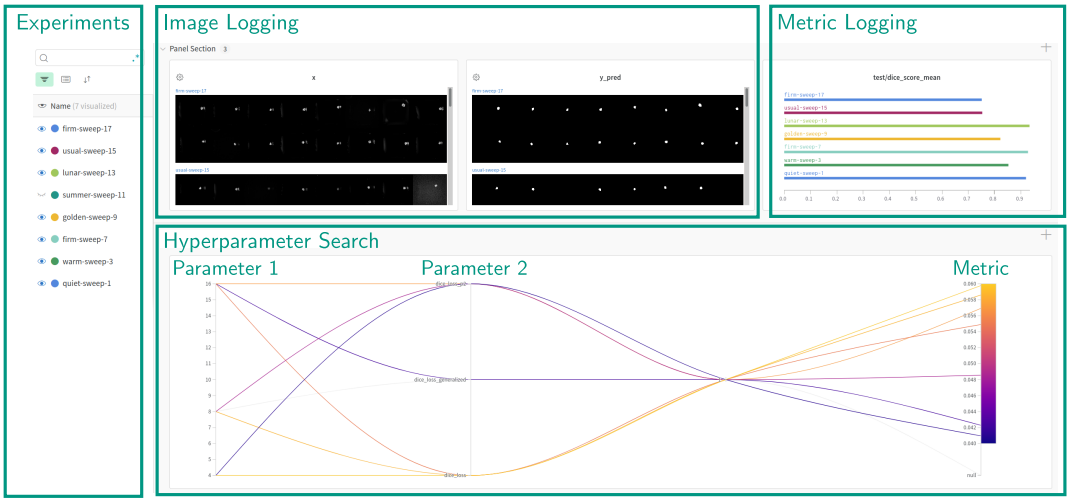


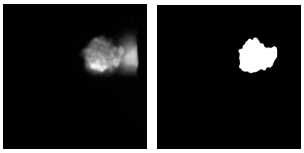
Figure 4: Exemplary logging via Weights&Biases of a hyperparameter search: Corresponding parameter sets and results in form of images or metrics may be displayed to structure and to compare different experiments.

environment, in which results can be inspected directly on a web-based dashboard. No direct connection to bwHPC infrastructure is required during training. Moreover, Weights&Biases allows designing different kinds of plots in the web interface using available logging data. This gains flexibility instead of programming all needed visualizations before starting experiments. Especially, for domain experts, this clears the hurdle of programming own result plots or directly interacting on a computing cluster. Results can be logged utilizing the LSDF in parallel maintaining flexibility. For instance, final results which require larger storage like interpreted images can be stored on the LSDF comfortably. If the accuracy of the best-performing model does not satisfy the requirements, a new iteration starts within the development/planning step. For instance, other architectures can be investigated or more labeled data may be collected. Otherwise, the application stage follows.

Since our workflow is not limited to model development, the application stage can be performed as well. Our proposed workflow is reduced in this case. The best-performing DNN can be loaded from the LSDF in the development/planning step. Dispatching is done via processing data submitted to the LSDF in HTC fashion. In the evaluation step, results are exported to LSDF/Weights&Biases depending on the data or problem.

4 Results

A biomedical image processing project, i.e., a binary segmentation of spheroids in a high-throughput Droplet Microarray (DMA) [12]), serves as example to introduce our workflow (cf. Figure 3). We use a DNN in order to solve the segmentation problem since e.g. the Otsu thresholding approach [10] fails to distinguish between spheroids and imaging reflections at borders.



(a) Sample (b) Mask

Figure 3: DMA Spheroid segmentation task [12]

First, the domain expert submits a small labeled dataset to the LSDF. Labeling is supported by a labeling tool to highlight the requirements of the needed binary image segmentation problem. Using this input, the data scientist integrates the U-Net [13] architecture in the PyTorch Lightning Framework. Since the hyperparameters of the U-Net are initially unknown, we utilize the algorithm Random Search [1] provided by Weights&Biases to obtain the best hyperparameter setting using multiple GPUs. Using the wrapper PyTorch Lightning [4], data parallel training is enabled. Hence, computing is boosted in terms of HTC starting several experiments in parallel and in terms of HPC since data parallel GPU training is used instead of using CPUs.

Logging in terms of evaluation was performed by Weights&Biases in the development stage to obtain the best-performing model. An example of a hyperparameter search is depicted in Figure 4[1]. Different

[1] Grid search is presented as dummy experiment in the figure since it is clearer arranged for visualization purposes.

Table 1: Run-time comparison: Average training time per epoch t_{epoch} and processing time t_{image} for different setups. DDP/DP denote the data parallelism.

Metric	CPU	1 GPU	2 GPUs (DP)	2 GPUs (DDP)	Human†
$\varnothing\ t_{\text{epoch}}$ in s	332.83	7.21	5.88	**4.72**	-
$\varnothing\ t_{\text{image}}$ in s	0.31	**0.003**	**0.003**	**0.003**	180

†Time for segmentation needed by an expert (lower bound obtained via a survey).

representations of logging and results (e.g. images or metric plots) can be utilized through evaluation. The accuracy of the best-performing model already satisfied the requirements of domain experts in the first iteration. The main reason is that binary spheroid segmentation is a well-known and less complex problem, through which no further iterations were needed in the process.

The best model parameters deployed on the LSDF can be used for image processing in the application stage. Analogously, the domain expert submits all the experimental data on the LSDF. The whole high-throughput experiment covers more than 55.000 samples that needs to be processed. Hence, the benefits of the proposal using powerful cluster computing infrastructure pay off in the application stage as well. Final results are stored on the LSDF and therefore can be accessed by the actors directly.

Taking computing performance of our proposal into account, Table 1 opposes a comparison of training time per epoch t_{epoch} and final processing time per image t_{process}.

Comparing a local CPU device of a domain expert (Intel i7 10750H) to GPUs mounted in bwHPC cluster (NVIDIA Tesla V100), the benefits of using powerful cluster hardware during training become clear. Computing with a single GPU accelerates training by more than a factor of 40 ($332.83\text{s} \rightarrow 7.21\text{s}$). Moreover, the DDP algorithm is superior to DP w.r.t. t_{epoch} in the case of using 2 GPUs. Furthermore, using a trained DNN on GPUs can boost processing time t_{image} in the application stage by about 60k ($180\text{s} \rightarrow 0.003\text{s}$) compared to manual inspection. Moreover, this computation can be done without human supervision. The effort for the expert consists only of submitting small labeled datasets and no special hardware is required locally. Hence, including bwHPC infrastructure in the workflow enables powerful computing without the need for local GPU infrastructure. Besides, depending on the number of parallel experiments and available computing resources, HTC or HPC branches can be favored. Since data parallel training does not scale linearly with the number of GPU due to data synchronization/exchange $\left(\text{cf. Table 1, } \varnothing\ \frac{t_{\text{epoch}}(1\text{ GPU})}{2} < \varnothing\ t_{\text{epoch}}(2\text{ GPUs,DP/DDP})\right)$, HTC should be preferred for a large number of experiments. Moreover, it needs to be considered that especially in the case of DDP, training deviates to a single GPU approach since complete batches are distributed to all GPUs. Hence, the effective batch size scales with the number of GPUs due to averaging over all GPUs per optimization step, whereby training can be destabilized following the results in [5].

5 Conclusions

We motivated that the process from submission of data to final processing is composed of different steps, which often impedes a straightforward proceeding in interdisciplinary DL projects. Current issues are data handling, computing strategies, and a complex DNN development process. Especially, the trend of big data and complex DNNs require an integrated and comprehensive solution. We present a workflow that can be used as a template for DL projects linking tools and methods with solutions for flexible HPC/HTC computing and data storage. The workflow shows, how the interaction between domain experts and data scientists can be designed. It can help to focus more on research in following DL projects and reducing overhead concerning infrastructure, tool, or collaboration of different actors in the development process. We demonstrate the benefits of our proposal by a biomedical image segmentation high-throughput experiment. It can be shown that utilizing bwHPC clusters accelerates the development of DNNs, boosts application, and relax processing requirements on expert devices. Results can be directly provided using Weights&Biases or the LSDF without any local processing. Part of further research can be the establishment of a submission system that triggers processing automatically when data is transferred to LSDF. The integration of the more elaborate HPC DASO approach [3] and a corresponding benchmark to the PyTorch Lightning HTC algorithms is an objective for the next investigations. Further, combining our proposal with data version control presented in [11] is pending.

Acknowledgments

The project was funded in the KIT Future Fields project "Screening Platform for Personalized Oncology (SPPO)" and was performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC. This project was partly supported by DFG (Heisenbergprofessur Projektnummer: 406232485, LE 2936/9-1). Furthermore, we thank the Helmholtz Program "Materials Systems Engineering" for the support.

References

[1] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10), 2012.

[2] L. Biewald. Experiment tracking with weights and biases. Accessed: 2021-08-30. Available: https://www.wandb.com, 2020.

[3] D. Coquelin, C. Debus, M. Götz, F. von der Lehr, J. Kahn, M. Siggel, and A. Streit. Accelerating neural network training with distributed asynchronous and selective optimization (DASO). arXiv:2104.05588 [cs.LG], 2021.

[4] W. Falcon. PyTorch lightning. Accessed: 2021-08-04. Available: https://github.com/PyTorchLightning/pytorch-lightning, 2019.

[5] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. arXiv:1706.02677 [cs.CV], 2018.

[6] E. A. Huerta, R. Haas, S. Jha, M. Neubauer, and D. S. Katz. Supporting high-performance and high-throughput computing for experimental science. *Computing and Software for Big Science*, 3(1), 2019.

[7] T. Jejkal, V. Hartmann, R. Stotzka, J. Otte, A. Garcia, J. van Wezel, and A. Streit. LAMBDA – the LSDF execution framework for data intensive applications. In *Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2012.

[8] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala. PyTorch distributed: Experiences on accelerating data parallel training. *Very Large Data Base Endowment*, 13(12), 2020.

[9] N. O' Mahony, J. Walsh, S. Campbell, A. Carvalho, L. Krpalkova, G. Velasco-Hernandez, S. Harapanahalli, and D. Riordan. Deep learning vs. Traditional computer vision. In *Advances in Computer Vision*, 2019.

[10] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 1979.

[11] D. Petrov and I. Shcheklein. Data version control - open-source version control system for machine learning projects. Accessed: 2021-09-21. Available: https://dvc.org/, 2021.

[12] A. A. Popova, T. Tronser, K. Demir, P. Haitz, K. Kuodyte, V. Starkuviene, P. Wajda, and P. A. Levkin. Facile one step formation and screening of tumor spheroids using droplet-microarray platform. *Small*, 15(25), 2019.

[13] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2015.

[14] M. P. Schilling, L. Rettenberger, F. Münke, H. Cui, A. A. Popova, P. A. Levkin, R. Mikut, and M. Reischl. Label assistant: A workflow for assisted data annotation in image segmentation tasks. In *Proceedings - 31. Workshop Computational Intelligence*, pages 211–234, 2021.

[15] A. B. Yoo, M. A. Jette, and M. Grondona. SLURM: Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing*, 2003.