DOI: 10.58895/ksp/1000162754-12 erschienen in:
Proceedings. 33. Workshop Computational Intelligence - Berlin, 23. – 24. November 2023
DOI: 10.5445/KSP/1000162754 | https://www.ksp.kit.edu/site/books/e/10.5445/KSP/1000162754/

MLOps for Scarce Image Data: A Use Case in Microscopic Image Analysis

Angelo Yamachui Sitcheu[1], Nils Friederich[1,2], Simon Baeuerle[1,3], Oliver Neumann[1], Markus Reischl[1], Ralf Mikut[1]

[1]Institute for Automation and Applied Informatics,
[2]Institute of Biological and Chemical Systems,
Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen
E-Mail: angelo.sitcheu@kit.edu
[3]Robert Bosch GmbH,
Markwiesenstraße 46, 72770 Reutlingen

Abstract

Nowadays, Machine Learning (ML) is experiencing tremendous popularity that has never been seen before. The operationalization of ML models is governed by a set of concepts and methods referred to as Machine Learning Operations (MLOps). Nevertheless, researchers, as well as professionals, often focus more on the automation aspect and neglect the continuous deployment and monitoring aspects of MLOps. As a result, there is a lack of continuous learning through the flow of feedback from production to development, causing unexpected model deterioration over time due to concept drifts, particularly when dealing with scarce data. This work explores the complete application of MLOps in the context of scarce data analysis. The paper proposes a new holistic approach to enhance biomedical image analysis. Our method includes: a fingerprinting process that enables selecting the best models, datasets, and model development strategy relative to the image analysis task at hand; an automated model development stage; and a continuous deployment and monitoring process to ensure continuous learning. For preliminary results, we perform a proof of concept for fingerprinting in microscopic image datasets.

1 Introduction

In the field of image analysis, Machine Learning (ML), particularly its subfield Deep Learning (DL) is being explored to model complex problems such as image registration, classification, segmentation, object detection and tracking [1, 29]. In this context, the goal of ML is to build a model that generalizes across different images for the same analysis task, for example, image segmentation. Therefore, the research community focuses more on developing new methods that achieve better performances and are computationally more efficient [17]. While developing the ML model offline seems easy and cheap, operationalizing the model, which means, deploying the model and maintaining its performance over time, still faces numerous challenges [10]. Unlike standard non ML-based software whose operations include building, testing, deployment and monitoring, ML systems are more complex due to the two new components model and data, and their direct, and generally challenging relationship. These systems are often a source of high technical debt when not operationalized consequently [28]. Similarly to standard non ML-based software whose lifecycle follows the Development and Operations (DevOps) scheme [3], ML systems have their development and operation paradigm known as Machine Learning Operations (MLOps)(see Section 2.1).

Nevertheless, as shown by the multitude of approaches developed [17], it is very challenging to develop and operationalize one single model that generalizes across different images for the same task. Many experts develop new models for the same image analysis task when new input data is available. Due to the time and cost expensiveness of this process, they often focus more on developing the model and neglect the continuous monitoring and deployment aspect [16], resulting in a lack of continuous learning through the flow of feedback from production to development, and therefore to unexpected model deterioration over time [28]. Additionally, the initial training data in this field is often scarcity-prone in quantity and quality due to the tediousness of several tasks such as image acquisition and image annotation requiring skilled and expensive experts. This often leads to a dataset containing less relevant data from the experiments and more noise.

A potential solution is to create a production-oriented machine learning approach that harnesses the full range of available and well-established datasets and models to improve the efficiency of image analysis across multiple tasks. This will be the main research goal of our work. In our paper, MLOps for sparse image analysis will be explored. We propose a long-term vision holistic approach to enhance biomedical image analysis that includes: a fingerprinting process that enables selecting the best models, datasets, and model development strategy relative to the task at hand, therefore making use of available models and datasets; an automated model development stage; and a continuous deployment and monitoring process.

Our work is organized in the following manner. Section 2 explains the fundamental concepts related to our work and provides an overview of other related works, and Section 3 details the proposed approach. In Section 4, the preliminary experiments carried out during the investigations are described. Their results are presented and discussed in Section 5. Section 6 summarizes our investigation and outlines future research directions.

2 Background and Related Work

In this section, we provide fundamental notions around MLOps and other important fields related to our work. We also present state-of-the-art approaches related to ours in general and regarding the different building blocks in particular.

2.1 MLOps

MLOps is a discipline that combines ML with software engineering paradigms such as DevOps and data engineering to enable efficient deployment and operationalization of ML systems [16, 30]. It can be seen to a certain extent as DevOps for ML systems. The key differences and similarities between DevOps and MLOps can be seen in Figure 1. On the one hand, both entail two main concepts Continuous Integration and Continuous Deployment.

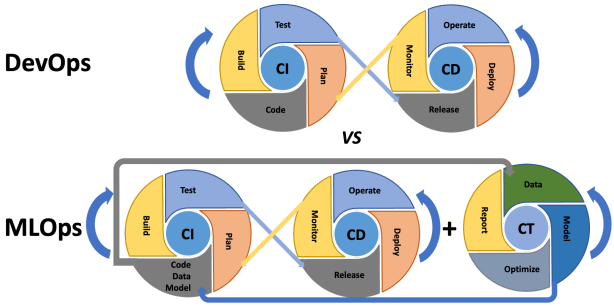

Figure 1: DevOps vs. MLOps [12]. While DevOps entails applying the main concepts of Continuous Integration (CI) and Continuous Deployment (CD) on code only, MLOps has the two new components data and model, which are added to the code component. Additionally, MLOps has a new concept named Continuous Training (CT).

- Continuous Integration (CI): consists in automatically building, testing and validating source code.
- Continuous Deployment (CD): enables frequent release cycles by automatically deploying the software in production. It distinguishes itself from continuous delivery by automatizing the deployment process.

On the other hand, in addition to normal source code, ML systems bring two new components: data and model, which are code-independent and therefore maintained separately from the code. This leads to the creation of pipelines to perform the complete processing. As a result, a novel concept specific to MLOps named Continuous Training (CT) emerges, which involves automatically retraining and serving the models. Furthermore, the CI and CD concepts in MLOps differ from those in DevOps in that CI does not only include integrating new code and components but also new data, models and pipelines, and CD does not deploy a single software package, but a complete ML training and/or serving pipeline. Additionally, continuous monitoring, i.e., automatically monitoring the IT system to detect potential problems such as compliance issues and security threads and address them, is extended in MLOps by monitoring production data and model performance metrics. This enables real-time understanding of model performances [32].

Although MLOps is a relatively new field, important work and progress have been made. While some researchers focus on properly defining MLOps and providing an overview of its concepts and best practices [16, 30, 32], others investigate the challenges faced in ML systems operationalization. Tamburri highlights in [31] the limited attention given to MLOps within academia and the lack of data engineering skills in academia as well as in industry. Renggli et al.[25] and Granlund et al.[7] extend on this and show that one massive problem in MLOps is data management. This is mostly due to the strong dependency between model performance and data quality. The cloud architecture center of Google proposes in [2] how to automate ML workflows. They classify MLOps into three different levels: MLOps Level 0 that refers to classical ML pipelines with no CI, no CD and manually operated workflows totally disconnected from the ML system, MLOps Level 1 in which data validation, model validation, continuous delivery and CT are introduced, and MLOps Level 2 where CI, CT and continuous delivery are fully explored. They apply semi-automated deployment in a pre-production environment and manual deployment in the production environment.

Several MLOps tools have been developed. A non-exhaustive list of tools and their features can be found in [16, 24, 30, 32]. There is no ideal tool, as each tool covers different MLOps aspects. In practice, tools are often combined to achieve maximal efficiency. However, the majority of tools focus on model versioning and tracking and ignore dataset versioning. This impedes the ability to reproduce results and renders it reliant on the coding practices of skilled experts [36]. In industry particularly, due to IT-Software's large and complex nature, the MLOps tools are often diverse and must match a specific established strategy.

Despite advancements in the MLOps domain, there are very few published real-world use cases in which MLOps is clearly designed, explained and applied. One use case found is Oravizio [8], a medical software for evaluating the risks associated with joint replacement surgeries. The researchers had four different risk models from which the best was selected and deployed. One issue they faced during development was related to data management, due to the multitude of data formats they had to process. A second use case is SemML [38] in which the researchers propose a ML-based system that

leverages semantic technologies to enhance industrial condition monitoring for electric resistance welding. Their workflow enables them to reuse and enhance ML pipelines over time based on new input data. A third example is microbeSEG [26], a DL-based tool for instance segmentation of objects. The researchers automate several data management tasks and model building processes. They, however, do not focus on monitoring and deployment.

We found two ongoing works connected to ours. Firstly, Friederich et al.explore in [4] Artificial Intelligence (AI) to encode dynamic processes. They propose a MLOps-based pipeline, in which active learning will be used to predict and record potentially important events during experiments in light microscopy. Secondly, Zárate et al.present K2E [36], a new approach to governing data and models. They investigate MLOps environments for creating, versioning and tracking datasets as well as models. They are still in the conceptualization step and plan to build their platform by following the Infrastructure as Code (IaC) paradigm. To the best of our knowledge, there is no MLOps approach similar to ours, particularly in the field of biomedical image analysis.

2.2 AutoML and Meta-learning

Automated Machine Learning (AutoML) is the process of automating different stages of the ML development cycle. These stages often include data preprocessing, feature engineering, model training and model evaluation. AutoML addresses problems such as Dynamic Algorithm Configuration (DAC), Hyperparameter Optimization (HPO), Combined Algorithms Selection and Hyperparameter Optimization (CASH) and Neural Architecture Search (NAS) [13]. Several researchers work towards building fully automated systems for their applications. For example, Meisenbacher et al.explain in [18] how to achieve AutoML for forecasting applications. They define five levels of automation for designing and operating forecasting models.
Numerous tools have been developed, each addressing specific AutoML subproblems. The developed tools often log metadata such as hyperparameters tried, pipelines configurations set, model evaluation results, learned weights and network architectures. Based on these experiences and other dataset meta-

data, a model is trained with the aim to adapt faster to new tasks [11]. This is meta-learning, also known as learning to learn.

While it is clear that AutoML can be combined with MLOps to enable high level automation in the ML development lifecycle [2, 30], exploring the output of this combination for meta-learning seems not to be investigated. The information acquired during the continuous monitoring stages appears not to be widely researched in combination with AutoML or meta-learning.

2.3 Scarce Image Data

Scarce image data is a massive problem in ML. In the field of image analysis, particularly in biomedicine, the acquisition and annotation of images must be done by highly skilled experts, require a considerable amount of time and resources, and are often error-prone [27]. As a result of these constraints, the amount of image data present is either small in quantity, dominated by noise, or small in quality, very weakly or not labeled. A lot of approaches have been developed to address data scarcity. On the one hand, there are image processing techniques to augment image data such as scaling, rotation and cropping. On the other hand, DL methods such as Generative Adversarial Network (GAN) [6] and Variational Autoencoder (VAE) [14] are used to generate synthetic images. These DL approaches often use Self-Supervised Learning (SSL) to understand better how data points are sampled [22]. We notice however that these solutions often focus more on image classification tasks.

2.4 Image Fingerprinting

Fingerprinting is used in image processing to generate concise and distinct representations of images. It is useful for diverse objectives such as image retrieval, copyright protection, or image similarity analysis. We focus on image fingerprinting to measure the similarity between images and/or datasets.

Ranging from simple pixel distribution methods [21] to DL approaches [15], there are numerous methods to compute similarity between images. Godau and Maier-Hein present in [5] an image fingerprinting approach that consists

of embedding images along with their labels in a fixed-length vector in order to capture semantic similarities in biomedical image datasets. Molina-Moreno et al.[22] build an autoencoder (AE), train this using SSL and obtain a two-dimensional latent space in which the disposal of image datasets displays their similarity. Such similarity measures enable researchers to apply transfer learning for their respective tasks. This is often achieved by selecting suitable pre-trained models and/or datasets for a new task based on the similarity measure obtained. Nevertheless, most state-of-the-art methods compute this similarity on a dataset level or on an image level and do not investigate the computation on an image patch level.

To fill the observed gaps in the context of image analysis, we propose a new approach, which is fully described in the following section.

3 Methodology

This section describes the proposed methodology to address the different problems identified and mentioned in Section 1 and Section 2. We first provide a global description of the system and subsequently delve into its individual building blocks.

3.1 Overview of the proposed approach

Figure 2 shows a conceptual architecture of our method. The main entering point is a scientist bringing a new image analysis task modeled as the triple(I, A, T_a), where I represents the image dataset, A the performance analysis metric, e.g., F1-Score, and T_a the task, e.g., image classification. At a time t, the performance metric A_t can be computed as defined in Equation (1), in which m_t represents the current model, I_t the current image dataset and $Y_{MD,t}$ the metadata relative to I_t. The following stages of our approach attempt to find the best model m^* defined through Equation (2).

$$A_t = m_t(T_a, I_t, Y_{MD,t})_{m \in \mathbb{M}} \tag{1}$$

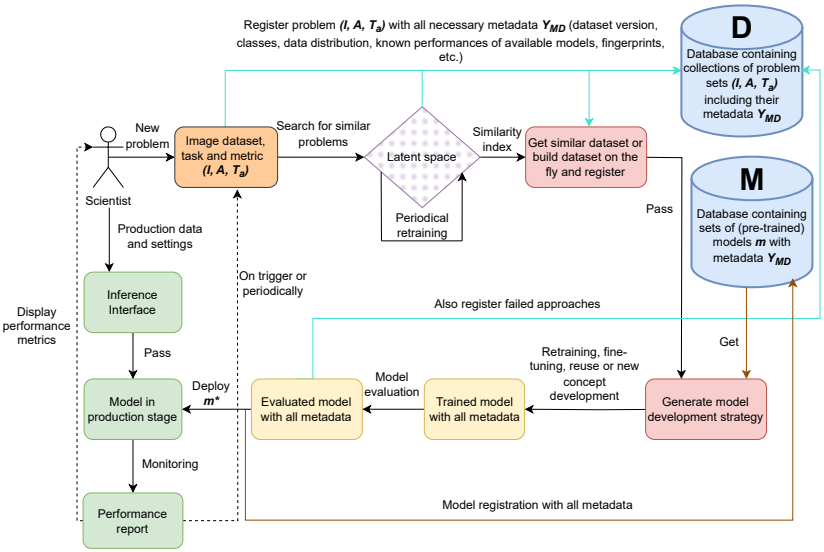

Figure 2: Abstract architecture of the proposed MLOps-based image analysis approach. The turquoise arrows indicate exchanges with the data database $\mathbb{D}$ and the brown arrows exchanges with the model database $\mathbb{M}$. The black arrows model the information flow between different building boxes of the system and the dashed lines feedback from production to development and to the scientist. The orange box represents the input provided by the scientist. The red boxes are the meta-learning system that handles dataset and algorithm selection. The yellow boxes display the AutoML pipeline. The green boxes represent the continuous deployment and monitoring stage for production.

$$m^*(t) = \underset{m \in \mathbb{M}}{\arg\max}\ (A_t) \tag{2}$$

3.2 Image Similarity

At $t = 0$, the newly provided problem is initialized and registered in the image database $\mathbb{D}$. The initial image dataset I_0 along with the task T_a, the performance analysis metric A_0 set to $-\infty$ and other metadata $Y_{MD,0}$ such as dataset version, classes, data distribution, fingerprints, known performances of available models for the same task T_a, etc. build an entry in $\mathbb{D}$. In order to discover images similar to I_0, its fingerprints $f_i(I_0)$ are subsequently computed, saved

and compared to those already present in $\mathbb{D}$ through a latent space embedding built in advance. This embedding will be periodically retrained to verify that the performance meets expectations continuously.

As emphasized in [5, 22], with such fingerprints, deploying new ML models for biomedical applications can be seed up by selecting appropriate pre-training models. Furthermore, this could solve scarcity issues by finding appropriate images for image augmentation on the one hand and for pre-training on the other hand.

3.3 Model Development Strategy

Inspired by early attempts in [35] and [37], the model development strategy aims to leverage the concept of meta-learning for scarce data. Given the tuple (I, A, T_a), the metadata Y_{MD} acquired during the registration phase and the computed fingerprints $f_i(I)$, the goal is to find the top $k(k \geq 1)$ cheapest and efficient model developing approaches. These approaches include both model and dataset and could range from model selection together with potential weights and development strategy, i.e., fine-tuning, retraining, to dataset selection or conception on the fly. Dataset conception in this context involves selecting the closest images to I in $\mathbb{D}$ and using these as training data.

The reason we envision using not only $f_i(I)$ is to minimize error propagation when fingerprinting is inaccurate. In this case, the meta-learner will solely rely on the metadata Y_{MD}.

3.4 Automatic Model Development

This stage is a direct application of the strategy established previously. It performs AutoML according to the strategy defined. Although AutoML is more computationally expensive than normal ML techniques, it is more efficient and faster in producing the best-trained model [2, 20]. It can be combined to MLOps to improve a project's automation level, therefore reducing the pipeline configurations overhead. For example, in a retraining process or new training, it could help solve the HPO problem faster at time t.

All development runs, including failing approaches, will be tracked and recorded in the model database $\mathbb{M}$. They would serve as input data for the meta-learner in the previous stage and may help identify flaws in the data or in the whole system. The best model m^* is sent in the last stage.

3.5 Continuous Deployment and Monitoring

The best model developed m^* is continuously deployed as a service and monitored during production. On the one side, we envision a deployment framework fulfilling fundamental requirements such as independency towards the ML Framework, rapid maintenance, accessibility, and parallel computing. Despite the existence of numerous deployment frameworks, we will focus on Representational State Transfer (REST) frameworks such as DEEPaaS [**?**] and EasyMLServe [23]. On the other side, the performance metric A_t as well as defined metrics by the scientist will be continuously monitored over time and reported. This monitoring will be particularly investigated, as it could help identify potential concept drift and decrease in performances, and act accordingly by triggering the complete framework from the start.

The high information reuse, which will be investigated in this work, will serve the purpose of exploiting available feedback and enabling transfer learning. Our approach will mostly be done using the Python programming language, due to its rich ecosystem of data science libraries and its expansive and highly engaged user community. We also plan to apply containerization with Docker [19] and its microservices, as it guarantees platform independency and enforces reproducibility.

4 Preliminary Experiments

This section describes the current state of the investigation. The experiments performed mainly focus on the first stage of our approach described in Section 3 which consists in building a latent space embedding in which image data can be represented along with their similarities. To this end, we build an autoencoder whose architecture will be depicted in Section 4.1, and evaluate

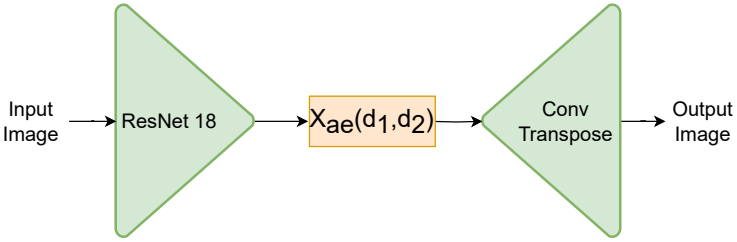

Figure 3: Autoencoder. The encoder is based on the ResNet18 architecture, the latent space representation is a two-dimensional space, and the decoder entails stack transpose convolutional layers.

it on biomedical image datasets presented in Section 4.2. Our implementation can be found in [33].

4.1 Autoencoder

An autoencoder is a special type of neural network that takes an input x, transforms it in a compressed and informative representation x_{ae} and reconstructs the initial input based on x_{ae}. The goal is to find an encoding function $f : x \mapsto x_{ae}$, and a decoding function, $g : x_{ae} \mapsto \hat{x}$ such that the difference between the reconstructed input $\hat{x}$ and the initial input x is minimal. This difference is measured by a loss $L : \min_{f,g} L(x, g(f(x)))$. The goal is to obtain a powerful representation x_{ae} that can be used for various tasks.

The architecture of our autoencoder can be seen in Figure 3. Our encoder is a vanilla (standard) ResNet18 [9] based neural network. The final fully connected layer is replaced by a linear layer mapping the 512 feature channels vector to a two-dimensional vector. Because we are more interested in x_{ae}, we build a simple decoder that entails five stacked convolutional transpose layers. We choose the Mean Squared Error (MSE) as loss function that computes the difference between the original input image and the reconstructed output image, and Adam as optimizer.

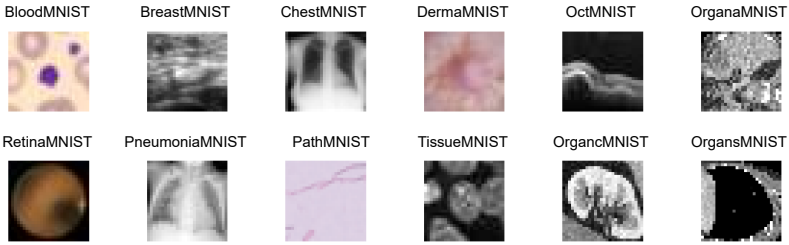

Figure 4: MedMNIST v2 2D datasets

4.2 Datasets

To evaluate our autoencoder, we select all the 2D image datasets of the MedMNIST v2 [34] dataset, a benchmark dataset for 2D and 3D biomedical image classification. The 12 selected datasets can be found in Figure 4. They consist of eight gray-scale image datasets (BreastMNIST, ChestMNIST, OctMNIST, OrganaMNIST, OrgancMNIST, OrgansMNIST, PneumoniaMNIST and TissueMNIST) and four color image datasets (BloodMNIST, DermaMNIST, RetinaMNIST and PathMNIST). The autoencoder is trained, validated and tested on the respective datasets splits. All the images are processed as $3 \times 32 \times 32$ images. For our architecture to be usable on all these datasets, the gray-scale images were loaded and processed as three-channel images.

5 Results and Discussions

We present in this section the results of the experiments performed in the previous section and discuss these.

Figure 5 shows the latent space representation of $N = 10000$ test samples collected from the 12 2D image datasets presented in the previous section. We notice that the color image datasets BloodMNIST, DermaMNIST and PathMNIST build a cluster at the top left. This is most likely due to the close distribution of the pixel values of the respective images. These color image datasets are, however dissimilar to the color image dataset RetinaMNIST, in

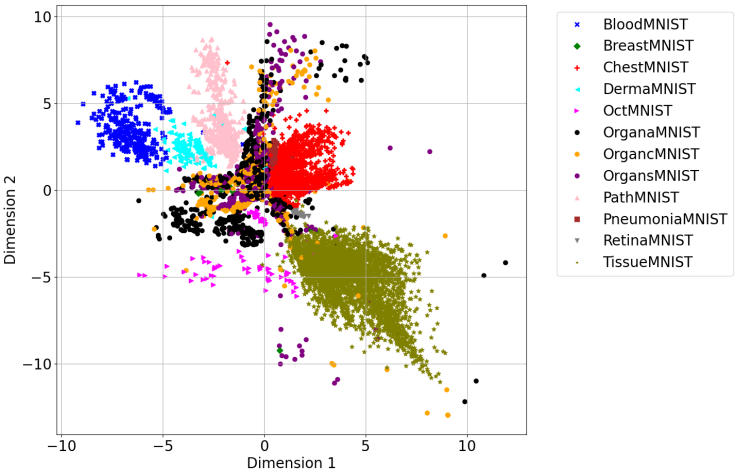

Figure 5: Latent space representation of the MedMNIST v2 2D datasets (test sets)

which the images of the retina all have a black contour. The OrganaMNIST, OrgancMNIST and OrgansMNIST datasets are mixed and almost not differentiable, leading to a non-identification of particular structures in the latent space. This is because all three datasets have images of the same organ taken on different views. In OrganaMNIST, the images are acquired on an axial view, in OrgancMNIST in a coronal, and in OrgansMNIST in a sagittal view. Nevertheless, all three datasets have multiple outliers that may hinder the model in positioning new incoming images.

A better view of the latent space is provided in Figure 6, in which the mean of all embedding vectors of the test images are computed for each dataset. We notice four main clusters. The first cluster entails the three-channel image datasets BloodMNIST, DermaMNIST and PathMNIST, the second cluster in which OrganaMNIST, OrgancMNIST and OrgansMNIST are very close, as well as PneumoniaMNIST and ChestMNIST. The third cluster consists of BreastMNIST and RetinaMNIST, and the final cluster of TissueMNIST and OctMNIST.

This autoencoder shows a promising ability to capture similarities between images. Despite being in the early stages of our investigation, we strongly believe

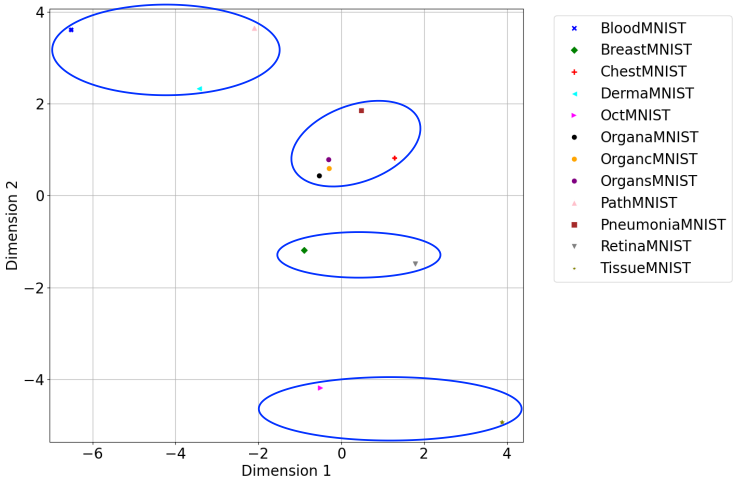

Figure 6: Mean value latent space representation of the MedMNIST v2 2D datasets (test sets)

that this approach to solving the image similarity question is encouraging and could be fine-tuned to identify specific structures in image crops.

6 Conclusion and Future Work

In this paper, we presented a new approach to improve biomedical image analysis. Our approach aimed to apply MLOps to image analysis tasks, particularly when the image dataset is scarce. To achieve this goal, we presented a multi-stage framework that leverages the existence of models and benchmark datasets to solve a given task. The first stage enables us to find image datasets similar to the images of the given task. The second stage applies meta-learning to select the best model development strategy for the given task. This strategy is executed via the third stage of AutoML. The final stage deploys the best-trained model and monitors the model's performance continuously to achieve optimal performance.

The preliminary experiments carried out in this paper mostly focused on the first stage, which consists in computing image fingerprints to identify similar

datasets. We built a ResNet18-based autoencoder and showed that the resulting 2D latent space representation is interpretable enough to find similarities between images. We, however, faced some challenges when the images are all from the same object but taken from different angles and focused only on 2D image datasets.

Our future research will, therefore, focus on extending the image fingerprinting process to 3D image datasets and even to the level of image patches and improving the representation of different artifacts. Understanding the image at this granularity level would enable us to generate data or labels to solve the data scarcity problem. We also plan to investigate the effects of outliers, as they may highly impact the similarity measurements. Finally, we will continue our research on the other stages of the proposed approach, including meta-learning for biomedical image analysis, AutoML and efficient model deployment and monitoring.

Acknowledgements

This project is funded by the Helmholtz Association under the program "Natural, Artificial and Cognitive Information Processing (NACIP)". We would like to thank the Helmholtz Artificial Intelligence Cooperation Unit (Helmholtz AI) for providing the computing infrastructure via the Helmholtz AI COmpute REssources (HAICORE). We also extend our gratitude to all those who provided assistance, even if not explicitly mentioned here.

The authors have accepted responsibility for the entire content of this manuscript and approved its submission. We describe the individual contributions of A. Yamachui Sitcheu (AYS), N. Friederich (NF), S. Baeuerle (SB), O. Neumann (ON), M. Reischl (MR), R. Mikut (RM): Conceptualization: AYS, NF, RM; Methodology: AYS, NF; Software: AYS, SB; Investigation: AYS, NF, SB; Writing – Original Draft: AYS; Writing – Review & Editing: AYS, NF, SB, ON, MR, RM; Supervision: RM; Project administration: RM; Funding Acquisition: MR, RM.

References

[1] H.-P. Chan, R. K. Samala, L. M. Hadjiiski, and C. Zhou. Deep Learning in Medical Image Analysis. *Deep Learning in Medical Image Analysis: Challenges and Applications*, pages 3–21, 2020. Springer.

[2] G. Cloud Architecture Center. MLOps: Continuous delivery and automation pipelines in machine learning. *Google*. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning Last Accessed: 10.08.2023.

[3] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. DevOps. *IEEE Software*, 33:94–100, 05 2016.

[4] N. Friederich, A. Yamachui Sitcheu, O. Neumann, S. Eroğlu-Kayıkçı, R. Prizak, L. Hilbert, and R. Mikut. AI-based automated active learning for discovery of hidden dynamic processes: A use case in light microscopy. In *Proceedings of the 33st Workshop on Computational Intelligence, Berlin*, 2023. Accepted.

[5] P. Godau and L. Maier-Hein. Task Fingerprinting for Meta Learning in Biomedical Image Analysis. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part IV*, page 436–446, Berlin, Heidelberg, 2021. Springer-Verlag.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen. MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases. In *1st IEEE/ACM Workshop on AI Engineering - Software Engineering for AI, WAIN@ICSE 2021, Madrid, Spain, May 30-31, 2021*, pages 82–88. IEEE, 2021.

[8] T. Granlund, V. Stirbu, and T. Mikkonen. Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN Computer Science*, 2(5), 2021.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[10] E. Hechler, M. Oberhofer, and T. Schaeck. *The Operationalization of AI*, pages 115–140. Apress, Berkeley, CA, 2020.

[11] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(9):5149–5169, 2022.

[12] S. Islam. MLOps vs. DevOps: What is the difference? https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/ Last Accessed: 18.09.2023.

[13] S. K. Karmaker ("Santu"), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni. AutoML to Date and Beyond: Challenges and Opportunities. *ACM Comput. Surv.*, 54(8), oct 2021.

[14] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[15] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[16] D. Kreuzberger, N. Kühl, and S. Hirschl. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*, 11:31866–31879, 2023.

[17] M. Maška, V. Ulman, P. Delgado-Rodriguez, E. Gómez-de Mariscal, T. Nečasová, F. A. Guerrero Peña, T. I. Ren, E. M. Meyerowitz,

T. Scherr, K. Löffler, et al. The Cell Tracking Challenge: 10 years of objective benchmarking. *Nature Methods*, pages 1–11, 2023.

[18] S. Meisenbacher, J. Pinter, T. Martin, V. Hagenmeyer, and R. Mikut. Concepts for automated machine learning in smart grid applications. In *Proceedings of the 31st Workshop on Computational Intelligence, Berlin*, pages 25–26, 2021.

[19] D. Merkel. Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.*, 2014(239), mar 2014.

[20] Microsoft. Machine Learning operations maturity model - Azure Architecture Center. *Microsoft Learn*, Last Accessed: 10.09.2023.

[21] H. B. Mitchell. *Image Similarity Measures*, pages 167–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[22] M. Molina-Moreno, M. P. Schilling, M. Reischl, and R. Mikut. Automated Style-Aware Selection of Annotated Pre-Training Databases in Biomedical Imaging. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2023.

[23] O. Neumann, M. Schilling, M. Reischl, and R. Mikut. EasyMLServe: Easy deployment of REST machine learning services. In *Proceedings 32. Workshop Computational Intelligence*, volume 1, page 11, 2022.

[24] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. D. Nucci, F. Palomba, and D. A. Tamburri. A Multivocal Literature Review of MLOps Tools and Features. In G. M. Callicó, R. Hebig, and A. Wortmann, editors, *48th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2022, Maspalomas, Gran Canaria, Spain, 31 August - 2 September 2022*, pages 84–91. IEEE, 2022.

[25] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlas, W. Wu, and C. Zhang. A Data Quality-Driven View of MLOps. *IEEE Data Eng. Bull.*, 44(1):11–23, 2021.

[26] T. Scherr, J. Seiffarth, B. Wollenhaupt, O. Neumann, M. P. Schilling, D. Kohlheyer, H. Scharr, K. Nöh, and R. Mikut. microbeSEG: A deep

learning software tool with omero data management for efficient and accurate cell segmentation. *PLOS ONE*, 17(11):1–14, 11 2022.

[27] M. P. Schilling, S. Schmelzer, L. Klinger, and M. Reischl. KaIDA: a modular tool for assisting image annotation in deep learning. *J. Integr. Bioinform.*, 19(4), 2022.

[28] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, and D. Dennison. Hidden Technical Debt in Machine Learning Systems. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2503–2511, 2015.

[29] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy. A review on deep learning in medical image analysis. *Int. J. Multim. Inf. Retr.*, 11(1):19–38, 2022.

[30] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas. MLOps - Definitions, Tools and Challenges. In *12th IEEE Annual Computing and Communication Workshop and Conference, CCWC 2022, Las Vegas, NV, USA, January 26-29, 2022*, pages 453–460. IEEE, 2022.

[31] D. A. Tamburri. Sustainable MLOps: Trends and Challenges. In *22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2020, Timisoara, Romania, September 1-4, 2020*, pages 17–23. IEEE, 2020.

[32] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, and G. Vessio. MLOps: A taxonomy and a Methodology. *IEEE Access*, 10:63606–63618, 2022.

[33] A. J. Yamachui Sitcheu. Pomlia. GitLab Repository, https://gitlab.kit.edu/angelo.sitcheu/pomlia, 2023.

[34] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni. MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification. *CoRR*, abs/2110.14795, 2021.

[35] P. Yuan, A. Mobiny, J. Jahanipour, X. Li, P. A. Cicalese, B. Roysam, V. M. Patel, M. Dragan, and H. V. Nguyen. Few is enough: Task-augmented active meta-learning for brain cell classification. In A. L. Martel, P. Abolmaesumi, D. Stoyanov, D. Mateus, M. A. Zuluaga, S. K. Zhou, D. Racoceanu, and L. Joskowicz, editors, *Medical Image Computing and Computer Assisted Intervention - MICCAI 2020 - 23rd International Conference, Lima, Peru, October 4-8, 2020, Proceedings, Part I*, volume 12261 of *Lecture Notes in Computer Science*, pages 367–377. Springer, 2020.

[36] G. Zárate, R. Miñón, J. Díaz-de-Arcaya, and A. I. Torre-Bastida. K2E: Building MLOps Environments for Governing Data and Models Catalogues while tracking versions. In *IEEE 19th International Conference on Software Architecture Companion, ICSA Companion 2022, Honolulu, HI, USA, March 12-15, 2022*, pages 206–209. IEEE, 2022.

[37] L. Zhang, X. Wang, D. Yang, T. Sanford, S. A. Harmon, B. Turkbey, B. J. Wood, H. Roth, A. Myronenko, D. Xu, and Z. Xu. Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation. *IEEE Trans. Medical Imaging*, 39(7):2531–2540, 2020.

[38] B. Zhou, Y. Svetashova, A. Gusmao, A. Soylu, G. Cheng, R. Mikut, A. Waaler, and E. Kharlamov. SemML: Facilitating development of ML models for condition monitoring with semantics. *J. Web Semant.*, 71:100664, 2021.

[39] Álvaro López García. DEEPaaS API: a REST API for Machine Learning and Deep Learning models. *Journal of Open Source Software*, 4(42):1517, 2019.