MLOps for TCR Binding Affinity Prediction System

Dominik Felbrich
Technical University Munich
dominik.felbrich@tum.de

Abstract

Machine Learning Operations (MLOps) is a young field that bridges the gap between developing machine learning models and deploying them in real-world systems. It has a broad relevance for academia, industry, and practical applications, each facing unique challenges and opportunities. This paper reviews the role of MLOps in industry and academia while providing insights from a hands-on practical project in computational biology (CompBio) to demonstrate the advantages and challenges of MLOps. While academia is outstanding in providing new research, deployment practices are often overlooked, which are emphasized in industry. In industry, other challenges arise, such as data acquisition, data drift, and scaling. By providing a project, this paper highlights the potential of MLOps, which can lead to more effective processes.

1 Introduction

"Your scientists were so preoccupied with whether or not they could, they didn't stop to think how to make it deployable." – Jurassic Park, probably.

Big data has become an integral aspect of our daily lives. Individuals produce data through social media posts, wearable devices, online shopping, and smart home appliances. Organizations capture and analyze data on customer behavior, product performance, and operational efficiency. One way of tackling the analysis of large amounts of data is through machine learning, which can uncover hidden patterns, predict future trends, and ultimately transform raw data into actionable knowledge. Machine learning assists with automated image analysis in biomedical fields and climate modeling in environmental sciences. Building successful machine learning solutions involves two distinct yet interconnected phases: Development, where data scientists explore data, engineer features, and train models. Moreover, deployment, where DevOps, MLOps, and data engineers integrate those models into production systems, ensure they scale, and monitor their performance [Symeonidis et al., 2022]. This process requires close collaboration among various professionals to ensure that models not only work well in research settings but also remain stable, scalable, and maintainable in real-world environments. A few problems arise when focusing on machine learning model development when trying to deploy such models as a machine learning system. The main issue is the lack of reproducibility if the entire process is not well-defined, including data preprocessing, models, hyperparameters, and environments. Manually deploying the model is a time-consuming process that is prone to errors. Furthermore, a deployed model can experience performance degradation over time due to shifts in the data, which is expected in real-world scenarios. If data scientists and DevOps engineers do not collaborate, workflows may become fragmented. Additionally, inadequate monitoring can create further issues. Without monitoring, changes in model performance, dataset updates, or deployment errors may go unnoticed. An overview of the roles can be found in figure 1. MLOps is a relatively new established field in this ecosystem, appearing around 2018 and 2019 [Treveil et al., 2020]. It streamlines the experimental character of machine learning creation processes, such as data gathering, data preparation, model training, model validation, deployment, and monitoring. Especially if there is an end user, MLOps requires the full bandwidth of development and deployment, and it connects the teams to set up a working machine learning

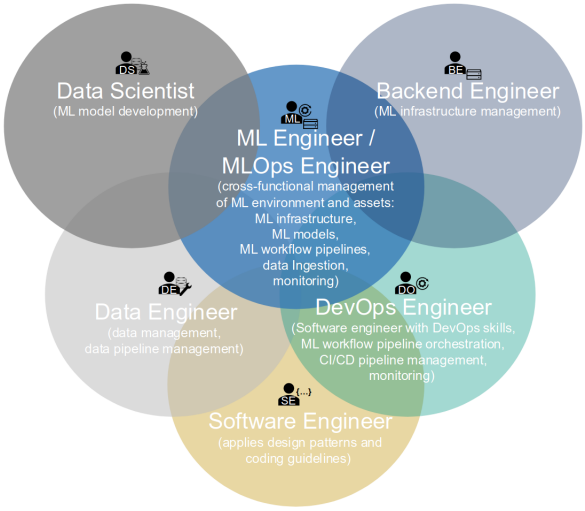

Figure 1: Visualization of the different roles in MLOps. Image from Kreuzberger et al. [2022].

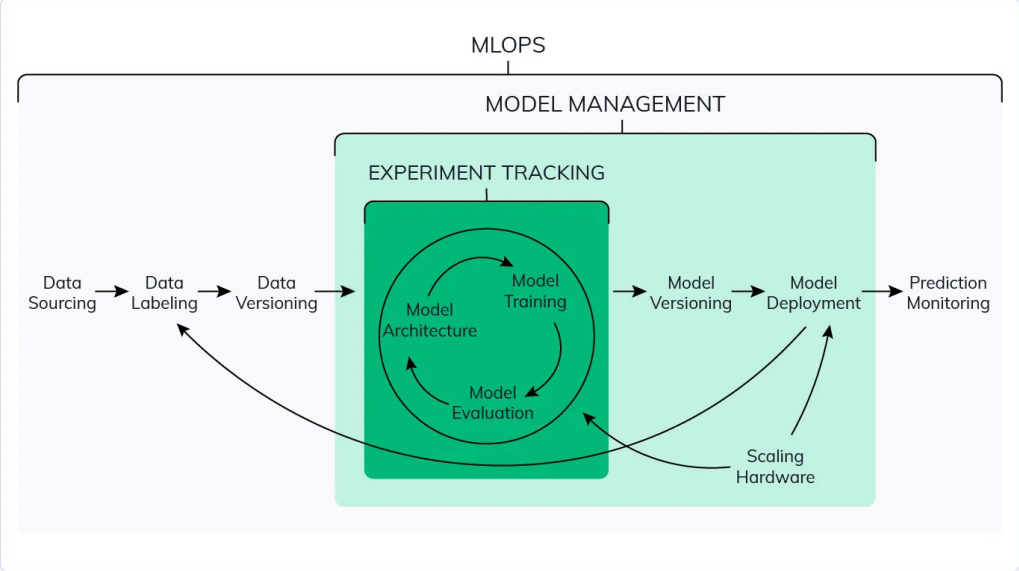

Figure 2: Visualization of the Machine Learning lifecycle with MLOps. Image from Canuma [2024].

system. As pointed out by Sculley et al. [2015], such systems can build up technical debt. Therefore, machine learning systems require constant maintenance. With MLOps, individuals and companies can be more efficient and thus save time and money [Canuma, 2024]. A visualization of the MLOps lifecycle can be seen in figure 2.

In this work, we will look at what MLOps is in section 2 with an overview of the MLOps components of machine learning, and DevOps. Then, in section 3, we will review MLOps in the industry with its applications and challenges. We then transition to the adaptation to academia in section 4, also presenting challenges related to academia. After that, we present a CI/CD pipeline implementation of a project that classifies T-cell reception of epitopes in section 5. Finally, we conclude in section 6.

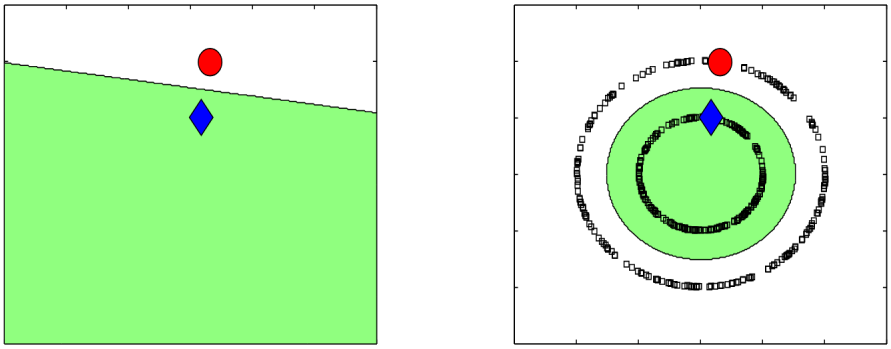

Figure 3: Visualization of the advantage of semi-supervised learning over supervised learning. Only supervised learning was done on the left graph, while semi-supervised learning was performed on the right. Image from Belkin et al. [2006].

2 Background on MLOps

MLOps is a combination of machine learning and DevOps and can thus be also named machine learning operations [Symeonidis et al., 2022]. Machine learning (ML) is not to be confused with artificial intelligence (AI) or deep learning (DL).

2.1 Artificial Intelligence, Machine Learning and Deep Learning

Artificial intelligence has no precise definition, but the one from Raphael fits it well: "Artificial intelligence is the science of making machines do things that would require intelligence if done by a man".

Machine learning is a subset of artificial intelligence. Its focus is on "learning" how to perform tasks, usually relying on data to do so. This includes algorithms such as linear regression, k-nearest neighbors, decision trees, and support vector machines. Learning is typically realized with one of the following types: In supervised learning, all data is labeled, e.g., an image of a cat is labeled as a cat. It is the most common type of machine learning. With unsupervised learning, data is not labeled, but one still tries to extract information from that, e.g., clustering. Semi-supervised learning combines these two, and only a tiny part of data is labeled [Campesato, 2020]. Figure 3 shows how it can be better than supervised learning in specific scenarios. The fourth one is reinforcement learning. Learning is done through trial and error, whereas some actions are awarded, and some are punished with the goal of maximizing the reward.

Deep Learning is a subset of machine learning. It refers to neural networks, which consist of neurons. These neurons are organized in multiple successive layers [Chassagnon et al., 2019]. A famous example of neural networks is generative pre-trained transformers (GPT), which utilize an attention mechanism.

2.2 DevOps

DevOps encompasses practices designed to shorten the time it takes to move a system change from development to regular production while maintaining high standards of quality. There are four relevant DevOps benefits. Firstly, it leads to a higher quality of deployments because the development and operation processes are chained together. This avoids taking shortcuts, leading to errors in ad hoc deployments. The infrastructure code used for the DevOps pipeline is developed using the same practices as the application code. Secondly, requirements for the deployment process are considered from the beginning of a project. The application has to be designed such that all necessary steps can be automated and do not require manual work. This includes logging and monitoring as well as automated tests. Thirdly, the development team is included in the incident handling, leading to faster fixing of issues. Usually, there are separate teams for the development and operations of larger projects. Because of the connected workflow, the development team can handle issues with the application themselves. Fourthly, continuous deployment shortens the time between a developer's

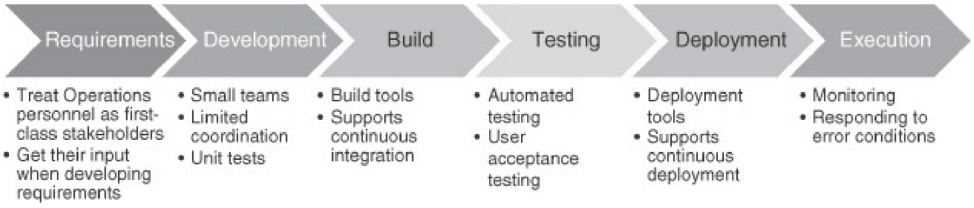

Figure 4: Representation of the DevOps cycle. Image from Bass et al. [2015].

commit to the repository and the code being deployed. This is because the deployment is executed on commit [Bass et al., 2015]. An overview of the DevOps process is visualized in figure 4.

2.3 MLOps: Combining ML and Ops

Depending on the project, an MLOps pipeline may cover various stages of the machine learning lifecycle. However, it typically ends with some form of monitoring and maintenance of the deployed model. In some cases, certain specialized workflows might skip or streamline this step.

The earliest starting point is the data acquisition. In case it needs to be updated frequently as part of the pipeline, a legal data source needs to be available to process it automatically. The source has to be accurate and reliable. After sourcing the data, it has to be preprocessed. This includes removing unwanted parts, de-noising them, and bringing them into a format that the model can process. If multiple data sources are used, labeling the data might be necessary. One might want to keep older datasets as well, for example, if the data source only contains new data or for reproducibility, which requires versioning the data [Treveil et al., 2020].

When the data is ready, the next step is the model experimentation. It includes the selection of an architecture for the model, model training, model evaluation, and hyperparameter tuning. This is the most intensive step in terms of computing power. An improvement can be scaling the hardware accordingly during this phase. Many factors can increase the time it takes to determine the best model, such as the number of model architectures, model size, dataset size, and amount and ranges of hyperparameters. Not to be underestimated is setting up a good tracking of all the parameters used for a training iteration in order to be able to reproduce results [Treveil et al., 2020].

After experimenting and identifying a suitable deployment candidate, the next step is to execute the deployment. In some projects, this may be as simple as copying data artifacts, while in others, it can involve transferring code, dependencies, and data artifacts across multiple networks. Once the model is successfully deployed, establishing monitoring is crucial to ensure consistent performance over time. Monitoring extends beyond accuracy or error rates, also encompassing inference speed and resource usage (for example, memory and GPU consumption) [Treveil et al., 2020].

3 MLOps in Industry

MLOps is already being applied in all industries that utilize machine learning. Otherwise, a productive environment would not be sustainable. In the following, we will list examples of industry fields where MLOps is relevant according to Solve [2023], Oladele [2024a].

1. Finance and Banking

   In the financial sector, MLOps is used for applications such as fraud detection, risk assessment, and algorithmic trading. There is a constant flow of data from markets and banking transactions. Alone in the eurozone, approximately 370 million non-cash payments were made per day in the first half of 2023 [European Central Bank, 2024]. Because frauds are constantly changing, there is a concept drift that must be taken into consideration. For example, JP Morgan is one company actively using MLOps [Morgan, 2024].

2. Manufacturing

   In manufacturing, the applications of MLOps are primarily for quality control and predictive maintenance. Products such as food, medicine, or printed circuit boards can be inspected for

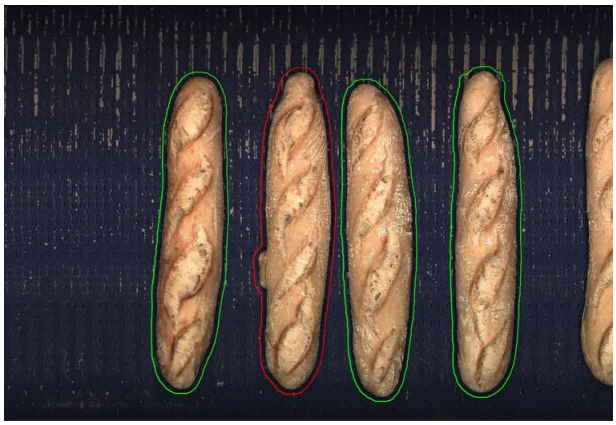

Figure 5: This picture shows an in-production use of anomaly image detection. Image from MVTec [2024].

defects. Siemens uses MLOps to manage predictive maintenance models, where downloading, tracking, and deployment of models is automated via their software [Siemens, 2024]. Another example where anomaly detection in images is used in a production system is shown in figure 5.

For example, a baguette factory can produce 4,000 baguettes per hour [Bell Publishing, 2018]. This means that only less than a second is available per baguette to decide if it is of good quality. Such a system needs to be able to compensate for data drift, such as changes in size or shape.

3. Retail and E-commerce

   Retail leverages MLOps for recommendations and efficient inventory management. One of the largest systems is the Amazon recommendation system. It uses data such as popular items, prices, and coupons and tracks search, browse, and buy history to recommend other items that one hopefully finds interesting [Levine, 2024].

4. Energy and Utilities

   MLOps is applied to forecast power consumption and equipment failure. The power grid needs to be as stable as possible. Therefore, adapting energy supplies and reducing defects in plants is essential. For example, Shell uses MLOps to build models that optimize their LNG production by calculating efficient settings [Gutierrez, 2021].

5. Transportation and Automotive

   MLOps is essential in autonomous driving, route optimization, and predictive maintenance for fleet management. In autonomous driving, MLOps ensures that models are updated with new driving data to improve safety and performance. A commonly known self-driving system is that of Tesla. They take their cars as data sources to train and improve their model, which is then deployed back on the cars [Bhargava and Desai, 2021].

6. Media and Entertainment

   In the entertainment industry, MLOps is employed in content recommendation systems, ad targeting, and copyright protection. For example, Netflix has developed a machine learning platform for their needs, such as personalized recommendations [Netflix, 2024].

7. Healthcare and Biology

   Utilized in healthcare and biology, MLOps is applied to predictive diagnostics, medical imaging, and drug discovery. For instance, Ortho Baltic, an orthopedic device manufacturer, produces patient-specific implants. They use 3D reconstructed anatomical models from a CT scan. This process was automated with an MLOps pipeline, which reduces the time it takes for reconstruction from three hours to a minute [Berkmanas, 2024].

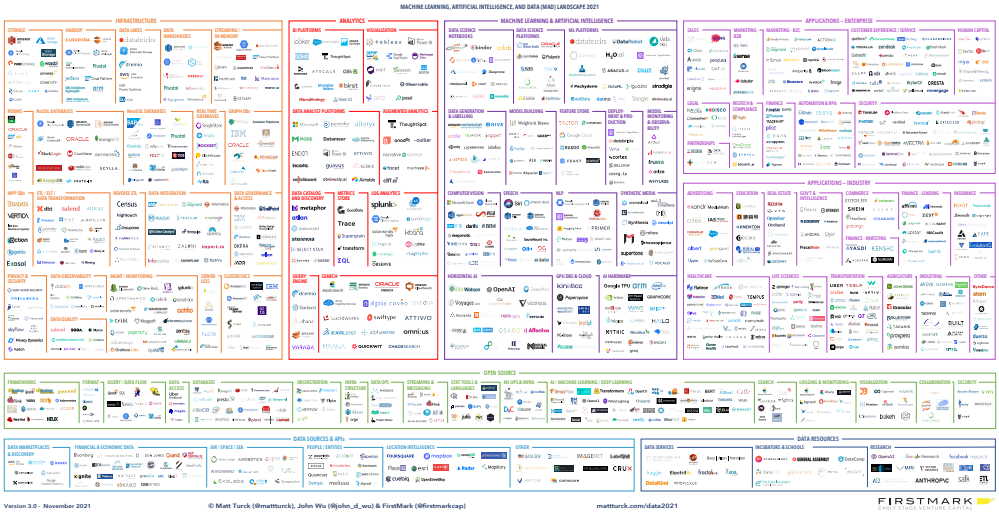

Figure 6: The machine learning, artificial intelligence, and data landscape. Image from Turck [2021].

3.1 Challenges in Industry

"MLOps is in a wild state today with the tooling landscape offering more rare breeds than an Amazonian rainforest." – Morris [2022].

Although there are working examples of MLOps being used in the industry, it is challenging to roll out such a process or a working pipeline fixing all issues. The first issue starts with the business case and what tools to use. Many tools originated in niches and then began to spread into other areas over time. This causes no clear separation of all tools into well-defined areas [Morris, 2022]. For example, Huggingface became well known for its transformer model. It expanded into providing datasets and code for training its models. Even for the choice of the machine learning framework, we have many possibilities: TensorFlow, PyTorch, Keras, MXNet, and scikit-learn, to name a few. Deciding on a framework leads to changes in handling input and output to and from it. This can cause significant changes when swapping the framework, so it must be well thought out. An overview of available applications can be found in figure 6.

Another challenge that comes with automatic retraining is when to trigger that. Data and models may change, and training can be computationally expensive or take a long time, so the trigger must be chosen wisely. Common triggers are upon update, after a specific time interval, e.g., every day, monitoring thresholds or manual triggers, e.g., from customers [Faubel et al., 2023]. Examples of what can happen are data drift and concept drift. With X the input and Y the output, data drift describes a change in the distribution $\mathbb{P}(X)$ and concept drift of the distribution $\mathbb{P}(Y|X)$. Data drift occurs when the input data in production changes and is different from the training data distribution. Concept drift changes the relationship between input data and their targets. Both scenarios are very likely to occur in a real-world scenario and can happen simultaneously [Evidently AI Team, 2024a,b]. Figure 7 provides an example of data and concept drift.

Talking about computation power, we have to decide which hardware to use. If we use our own hardware, we can use it efficiently, which reduces costs but also requires more time. On the other hand, we can buy a lot of hardware at a higher cost, which is then only sometimes utilized. Another way is going into the cloud. It allows us to scale efficiently according to our needs, e.g., temporarily buying a lot more computing power. However, we depend on one provider and their tools, as well as their invoices [Faubel et al., 2023].

When it comes to data, the acquisition might be a challenge. It can require changes in software and hardware, which, in the worst case, requires exchanging machines, infrastructure, or manufacturing lines. In case this data contains sensitive information, it needs to be appropriately protected [Faubel et al., 2023]. For example, the USA Health Insurance Portability and Accountability Act (HIPAA) protects individuals' medical information. In the EU, the General Data Protection Regulation (GDPR) guards the data of its citizens. Usually, the consent of the individuals is necessary to process their

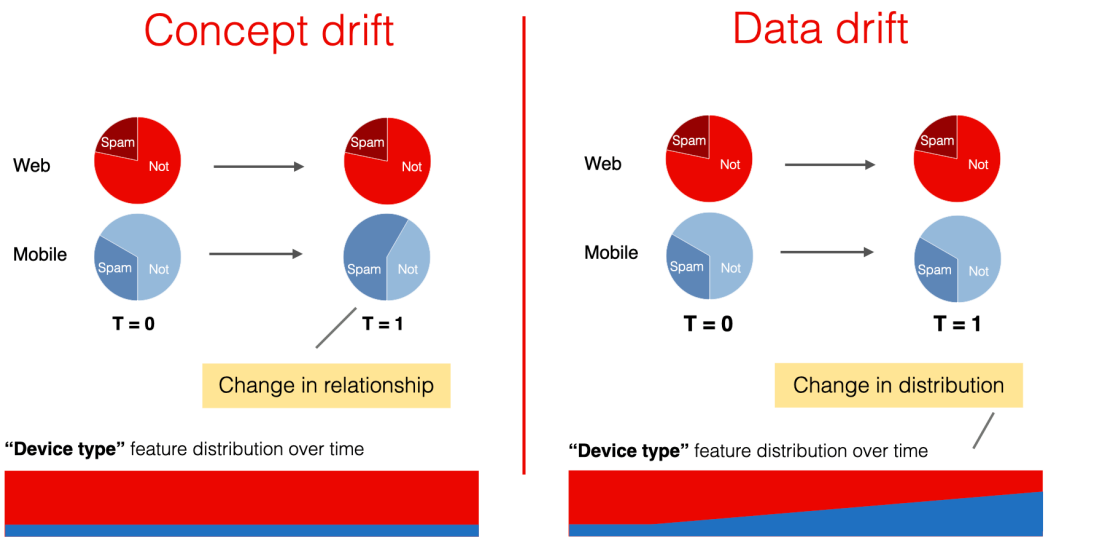

Figure 7: An illustration of data drift and concept drift for mail spam detection. For concept drift, the distribution of web and mobile remains similar, however the feature relationship of spam and not spam changes in time. For data drift, this relation remains similar while the distribution of web and mobile changes over time. Image from Evidently AI Team [2024b].

data. This is especially relevant when utilizing a cloud provider. The CLOUD Act is a privacy issue for cloud providers based in the USA. It forces companies in the USA to provide law enforcement data from the company's servers even when they are outside of the USA [Rutherford, 2019]. This means that when using cloud services from US-based companies, we can never be sure that our data will not be copied. That causes issues with EU data protection laws, which need to be respected, especially when working with private customers.

4 Translating MLOps from Industry to Academia

"Academics push technological frontiers, from artificial intelligence to deep learning, without considering how they will be applied. Manufacturers want to know what types of data to sample, which sensors to use, and where along the production line to install them." – Kusiak [2017]

The difference between industry and academia stems from the objectives. Academic research focuses on exploration and publication rather than creating a stable pipeline. The engineering around machine learning should be paid more attention. An example of such differences can be found in figure 8.

As a result, there are some points in academia that MLOps can improve on. This results in better reproducibility and transparency of the results. In academic research, the machine learning model or its hyperparameters might be specified incorrectly or under-specified. This includes the data used, such as its split or preprocessing. However, there can also be subtle differences, such as using a CPU vs. GPU or using PyTorch vs. TensorFlow. Result metrics are often reported selectively, highlighting the strength of the research. However, in MLOps, we want to pick the best model, therefore considering more than one metric. Not all publications come with the code or data they used. One reason can be the time pressure in publishing, leading to bad code quality. By establishing MLOps, researchers are forced to write better code with automations. Of course, this does not fix issues with not enough time. Maybe in the future, it is possible to establish a culture of quality over quantity in machine learning research with the help of MLOps [Semmelrock et al., 2024].

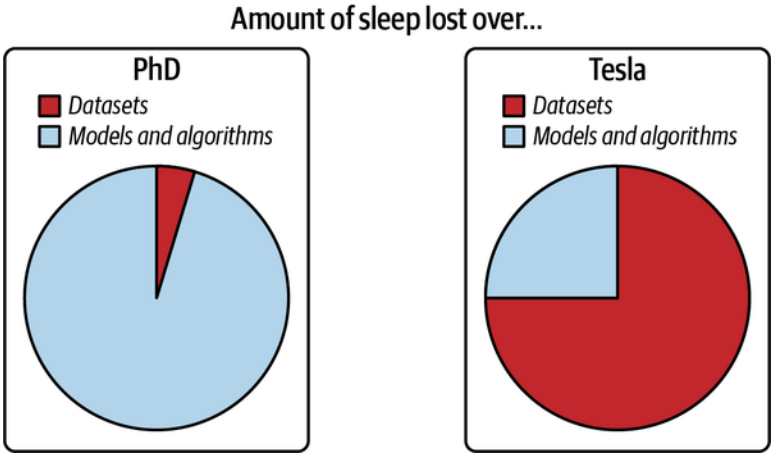

Figure 8: Humorous depiction of the difference in data handling between academia and industry. Image from Huyen [2022] as an adaption from Andrej Karpathy.

4.1 Challenges in Academia

Academia faces different challenges compared to the industry because of the other focus, as presented in this chapter. The first challenge is the limited resources, for example, the lack of scalable computing infrastructure. When infrastructure is available for use, it is usually monolithic, which has the disadvantage of being only scalable if more hardware is bought. Choosing cloud infrastructure makes it easier to scale accordingly. However, we must consider costs and commit to the provider's toolset. We would lose control over the technology stack, which might be relevant for the research. Additionally, the possibilities supported by the provider might not meet the research requirements [Eken et al., 2024].

Due to the rapid growth and evolution of MLOps practices, MLOps is complex and diverse. This results in many solutions and tools for different applications of MLOps. There exists no general standard, and in academia, there cannot exist a "company"-wide standard. Thus, a high diversity of tools is brought along when working in a team for a machine learning or MLOps research project. This can be challenging when trying to collaborate on the same code. Alternatively, the team has to invest time in getting used to the same tool set [Eken et al., 2024].

Another challenge of educational gaps may arise from the novelty of MLOps. The origin can be in the self-learning of different knowledge bases or unfamiliarity of the field, resulting in different focus areas within MLOps.

The biggest challenge in academia arises from "publish or perish", which means outputting short-term research as fast as possible. After completing a research project, there is no incentive to keep the project running or provide maintenance for it.[1] This reduces the value of implementing an MLOps pipeline or fixing issues with it and can lead to a project becoming outdated. Additionally, there is no incentive to have automated monitoring. While manual monitoring is demanding and requires time, it is quicker in setting it up and sufficient for short research projects [Ghosh, 2024].

A challenge similar to those in the industry is data and concept drift. When working with theoretical scenarios using fixed datasets, models or hyperparameters, the results may not be reproducible in the constantly changing real world.

5 MLOps System Application with T-cell Reception of Epitopes

We implement an application in the field of computational biology to illustrate the concepts of MLOps. Firstly, in section 5.1, we explain the machine learning experiment we use here to build an MLOps pipeline around it. Secondly, we then elaborate on this pipeline in section 5.2. We also discuss findings during the implementation.

[1] An exception would be if research continues based on the previous project.

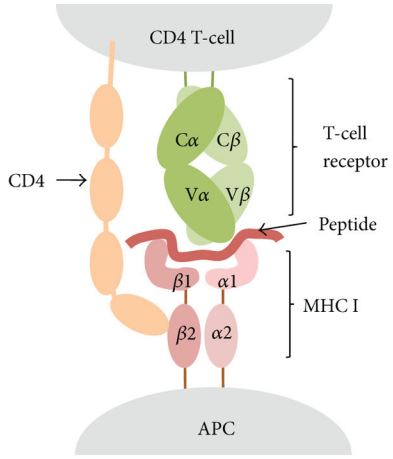

Figure 9: Visualization of the T-cell epitope recognition. Image from Sanchez-Trincado et al. [2017].

Table 1: The viral dataset we use. It consists of ten columns. The variable (V) and joining (J) segments combine and form the antigen-binding region of the TCR, including the CDR3_alpha and CDR3_beta regions. When the TCR recognizes an epitope presented from the MHC, the label is 1; 0 otherwise. The clone_id describes a group of identical or nearly identical receptors. We also provide two examples from the data.

CDR3_alpha	CDR3_beta	V_alpha	V_beta	J_alpha
CAVNAPTGTASKLTF	CASSMRSAVEQYF	TRAV8-1*01	TRBV19*01	TRAJ44*01
CATEDNDMRF	CASSFSDTQYF	TRAV17*01	TRBV5-4*01	TRAJ43*01
J_beta	Epitope	clone_id	MHC	Label
TRBJ2-7*01	GILGFVFTL	25.0	HLA-A*02:01	1
TRBJ2-3*01	GILGFVFTL	39.0	HLA-A*02:01	0

5.1 Machine Learning Task

This section describes the machine learning task of T-cell receptor epitope prediction. T-cell epitope prediction aims to identify the shortest peptide fragments from antigens that can activate CD4 or CD8 T-cells. Generated from larger antigenic proteins, T-cell epitopes are specific peptide sequences. These peptides play a crucial role in T-cell activation, as they are recognized by the T-cell receptor (TCR) when presented by major histocompatibility complex (MHC) molecules on the surface of antigen-presenting cells (APCs) [Sanchez-Trincado et al., 2017]. TCR specificity is essential for developing immunotherapies and understanding the efficiency of vaccines [Drost et al., 2024]. A simplified version of the biological process can be found in figure 9.

For the implementation, we use the data as provided by `https://github.com/SchubertLab/benchmark_TCRprediction`. This repository contains the code to recreate the study from Drost et al. [2024]. To set up the repository with its dependencies, it is vital to note that Python versions 3.8 to 3.12 do not work because of dependency issues. For us, it worked using Python 3.7. We use the viral dataset from Kocher et al. [2024], Adams et al. [2023]. The repository is set up as a benchmark, and we extract the benchmark data to transform it into a dataset. An overview of the dataset structure and example data can be found in table 1.

All of the following is implemented using PyTorch.[2] The dataset needs to be preprocessed in order for a neural network model to be able to process it because we cannot input letters. We convert all letters by analyzing which ones are in the dataset and create a bijective mapping $f : L \to (0, 1]$, where L are the unique characters. The characters are sorted by their Unicode value and then assigned equidistant values between 0 and 1, whereas 0 is not assigned, but 1 is. We replace the letters with the numeric values saved in a list. The 0 is required for the padding, which is done after the mapping.

[2] `https://pytorch.org`

We do the padding in three groups. Groups are (CDR3_alpha and CDR3_beta) – (Epitope) – and (V_alpha, V_beta, J_alpha, J_beta and MHC). For each of these groups, all the lists are padded with zeros until they reach the length of the longest entry in its group. Then we normalize the clone_id to values in $[0, 1]$. As a last step, all of these lists are merged into a single list, and the labels are separated from the input values.

After preparing the dataset, we split it into a training, validation, and test set. We use 10% each for validation and testing. The remaining approximate 80% are used for training. To do that, we use the random_split[3] function and set the seed of PyTorch to 42. An issue that comes with the dataset is class imbalance. We have a lot more negative samples (with label 0) than positive samples (with label 1). Doing naive training would lead to a model preferring to predict negative samples while still achieving a high accuracy with that. Thus we utilize the WeightedRandomSampler[4] from PyTorch. Weights are applied such that they sum up to 1, and the distribution of both classes is equal after sampling.

To demonstrate that the training is working without utilizing MLOps, we provide an example configuration here. For the loss function, we use BCEWithLogitsLoss[5], where this function is a combination of binary cross entropy with a sigmoid layer. The sigmoid layer is required because our model outputs logits that are not necessary in $[0, 1]$, and the combination with the loss is numerically more stable. We use Adam[6] as the optimizer with a learning rate of 0.01. We train for 15 epochs on a GPU and use a batch size of 2048.

As the model, we use a simple, medium-sized, feed-forward neural network. It consists of blocks, each of which contains a linear layer, a batch norm layer, and an exponential linear layer (ELU) as the nonlinearity. In this explicit example, we have twelve blocks, each decreasing the output dimensions of the features by ten. The only exception is the last block, which only consists of a linear layer.

With this setup, we achieve an overall accuracy of 87.47%. Specifically, the accuracy of labels predicted as 1 (positive) is 29.17%. Of course, as already mentioned, these results are only of an experimental nature and are not suitable for applying them in an MLOps scenario. In such a scenario, our focus is not only on the best results but also on reproducibility and robustness. There is no guarantee that little changes in the setup of the experiment will lead to similar outcomes.

5.2 MLOps Implementation

This section elaborates on our MLOps setup, which builds upon the machine learning task described in section 5.1. Figure 10 shows the maturity levels that Microsoft employs. We aim for a level four demonstration, touching level five. An overview of our pipeline implementation can be found in figure 11, which we explain in depth now. The pipeline is executed with Gitlab CI/CD and Gitlab runners, which all run on the same server. In a ".gitlab-ci.yml"-file, the commands are stated, which are then executed by the runner. For our setup, we have a Docker runner, where code runs within a Docker container, and a shell runner, where code runs directly on the machine.

5.2.1 Pipeline Implementation

Starting with the dataset, it is processed flexibly for changes. We download the benchmark repository and move it so that its code can be called from our code. Due to the framework we use to receive the dataset, the data does not change over time. We save the test dataset and the functions applied during the preprocessing as a PyTorch pt-file and in a JSON file to use for the production server.[7]

After preparing the dataset, we continue with the hyperoptimization loop. Hyperoptimization is necessary because we have hyperparameters with different ranges or selections that might improve the model's performance. Because this can only be found experimentally by training, the goal is to find the best combination with the fewest attempts possible. We use the Python package hyperopt[8] for that. Hyperopt chooses hyperparameters from their given definitions and runs a training loop until

[3] `https://pytorch.org/docs/stable/data.html#torch.utils.data.random_split`

[4] `https://pytorch.org/docs/stable/data.html#torch.utils.data.WeightedRandomSampler`

[5] `https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html`

[6] `https://pytorch.org/docs/stable/generated/torch.optim.Adam.html`

[7] We provide further details on what we do with the test dataset and the functions later.

[8] `https://hyperopt.github.io/hyperopt/`

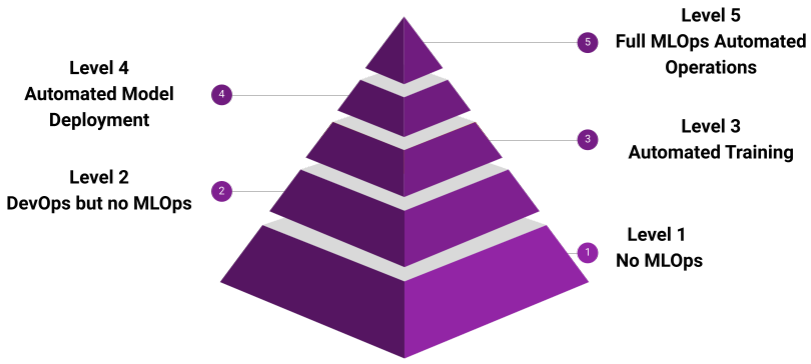

Figure 10: MLOps maturity levels according to Microsoft. Image from Symeonidis et al. [2022].

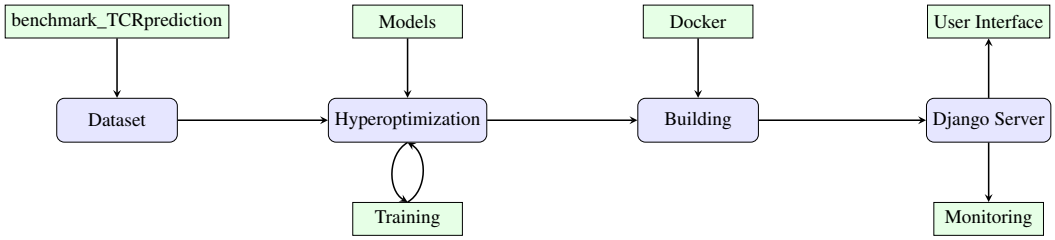

Figure 11: The pipeline we implement in this project. We use the TCR prediction benchmark data and preprocess it to a dataset. This dataset is then used in the hyperoptimization with some pre-defined models. The optimization starts multiple training iterations and tries to find the best combinations. Those are passed onto the building stage, which creates a Django server in a Docker container. This server provides the user interface and monitoring.

the maximum number of evaluations is reached. Typical choices for optimizable hyperparameters are the model, loss function, optimizer, learning rate, and number of epochs. Here we try, in addition to the medium-sized neural network, as described in section 5.1, a large-sized neural network that has double the amount of blocks as the medium-sized neural network and a small neural network, which only consists of three linear layers, separated by two ReLU layers. We optimize the learning rate uniformly in the interval $[0.001, 0.1]$. The number of epochs is in the range between 5 and 25.

To enable hyperopt to compare training loops, we return a loss value for each training. This loss value is a weighted sum of different metrics. The overall accuracy for the test dataset receives a weight of one. We also calculate the relative amount of positive classifications compared to the occurrences in the test dataset and the accuracy of the positive labels. The former is given a weighting of 5 and the latter one of 10. These weights make sure to reward trainings that can handle the class imbalance. We do not use the loss from the loss function of the training. The reason is that the loss is not meaningful to compare when using different models. After being done with the hyperoptimization, the top three models, together with their statistics, are passed on to be accessible later for the server.

Following the hyperoptimization is the building stage. We use Docker to build a Docker container. This container contains all necessary files from the project and builds a runnable Django server.

Once the container is built, it is pushed to the local Docker registry and executed. The main features of the web server are the testing and logging sites.

First, we explain how the user interface works. When the Django server is initialized, it analyses available models and its training results. It chooses the model with the best loss as responsible for the user queries and caches this model. This allows for fast responses to user input. An impression of how the interface looks and displays its results can be found in figure 12 and figure 13.

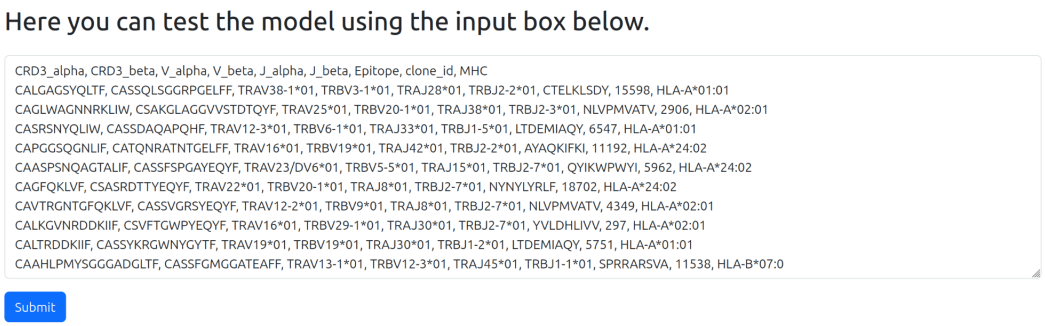

Figure 12: Web interface where data can be input into the model.

Figure 13: Web interface with the results of the input.

Here, we use the test dataset and its preprocessing functions for two reasons. The first one is to be able to parse the user input. A user provides human-readable data similar to the original dataset. In order to be able to input it into the model, we use the preprocessing functions to transform the data. The second one provides example data to the user, such as the one that can be seen in figure 12. We sample ten random entries from the dataset. Because the saved dataset is in tensor form, we reverse the preprocessing functions and apply them to the tensors to provide the user with human-readable example input.

We provide an overview of the saved models. That includes information about when the training was started, whether the model is currently used, if the model file is available, and if the training results are available as a JSON file. Additionally, we log the queries, their output, when the query was submitted, and if it was valid. An example of how the according web pages look can be found in figure 14 and figure 15.

5.2.2 Outlook on Further Features

We set up the code and features in a basic form. In a real production scenario, this would be extended. Extending the Django server would be easy to implement with the current setup. For example, we could include graphs or an automated alarm system that sends notifications via email if specific values are too far off as monitoring. The monitoring could also be more specific with its parameters. Currently, only the indices of choices are displayed, for example, for the model, the loss function, and the optimizer. With the user input, we could check if our current and future models still produce

Time	Active	Model Available	JSON
2024-12-14 13:58:14	✓	✓	{ "loss": -6.313549995422363, "avg_test_loss": 0.0007938563756953428, "avg_test_accuracy": 0.921612560749054, "test_positive_count": 70, "relative_test_positive_count": 0.07838745800671892, "avg_test_positive_accuracy": 0.5, "parameters": { "learning_rate": 0.018880959128307565, "loss_function": 0, "model": 2, "num_epochs": 10.0, "optimizer": 0 } }
2024-12-14 13:58:14	✗	✓	{ "loss": -6.079507350921631, "avg_test_loss": 0.0004665023602403425, "avg_test_accuracy": 0.9171332716941833, "test_positive_count": 76, "relative_test_positive_count": 0.0851063829787234, "avg_test_positive_accuracy": 0.47368421052631576, "parameters": { "learning_rate": 0.02799664336020661, "loss_function": 0, "model": 2, "num_epochs": 6.0, "optimizer": 0 } }

Figure 14: Overview of the models. It shows when the training was performed, if the model is currently used, whether it is available, and the JSON with the training results if available.

Time	Valid
Jan. 28, 2025, 10:43 a.m.	✗

```
Simulation of wrong input
```

```
Error in line 1: must contain 9 comma-separated values.
```

Time	Valid
Jan. 28, 2025, 10:43 a.m.	✓

```
CRD3_alpha, CRD3_beta, V_alpha, V_beta, J_alpha, J_beta, Epitope, clone_id, MHC
CAVSDRPTPLVF, CASSFVLAGGWNTQYF, TRAV8-4*01, TRBV12-4*01, TRAJ29*01, TRBJ2-3*01, TYGPVFMCL, 5479, HLA-A*24:02
CVVITGNQFYF, CASSEGGLGYEQYF, TRAV12-1*01, TRBV2*01, TRAJ49*01, TRBJ2-7*01, LTDEMIAQY, 5383, HLA-A*01:01
CAVRDGTGANNLFF, CATEVTSTDTQYF, TRAV3*01, TRBV19*01, TRAJ36*01, TRBJ2-3*01, RLQSLQTYV, 81, HLA-A*02:01
CAGISDSWGKLQF, CSARAFWDWDDIQYF, TRAV35*01, TRBV20-1*01, TRAJ24*01, TRBJ2-4*01, FPQSAPHGV, 847, HLA-B*07:02
CVVSDNYGQNFVF, CASSQGQGPYEQYF, TRAV12-1*01, TRBV4-1*01, TRAJ26*01, TRBJ2-7*01, YVLDHLIVV, 15842, HLA-A*02:01
CAFFPMDTGRRALTF, CASSVANRATGDEQFF, TRAV24*01, TRBV9*01, TRAJ5*01, TRBJ2-1*01, KCYGVSPTK, 3480, HLA-A*03:01
CAATPFYGGSQGNLIF, CASSPEIGQYF, TRAV13-1*01, TRBV7-9*01, TRAJ42*01, TRBJ2-3*01, GILGFVFTL, 691, HLA-A*02:01
CAVNRGGSNYKLTF, CSVGTGWPYEQYF, TRAV12-2*01, TRBV29-1*01, TRAJ53*01, TRBJ2-7*01, QYIKWPWYI, 8172, HLA-A*24:02
CALSDANRDDKIIF, CAVGQGLYNEQFF, TRAV9-2*01, TRBV10-3*01, TRAJ30*01, TRBJ2-1*01, GILGFVFTL, 6306, HLA-A*02:01
CALSEPNYGGSQGNLIF, CASTRFSTSGRETQYF, TRAV19*01, TRBV6-5*01, TRAJ42*01, TRBJ2-5*01, NYNYLYRLF, 7036, HLA-A*24:0
```

```
False
False
False
False
False
False
False
False
False
False
```

Figure 15: Overview of the queries. It shows when a query was executed, if its input produced valid output, and what the input and output were.

reasonable results. The user interface could also provide the ability to use other models or compare the results with the test dataset.

Additional improvements can be made for the hyperoptimization by providing more options. As of right now, there is only one optimizer and one loss function possible. More complex models could also be tested, e.g., adaptions from research papers.

Regarding code quality, a script to create a testing environment could be added. Right now, the project is optimized to run on the production system. In order to receive a working local setup, things such as virtual environments and file links have to be created. Furthermore, more documentation could be preferable to allow for a quicker start on our production system.

5.2.3 Challenges of the Implementation

While implementing the MLOps pipeline, we faced several challenges, which we want to mention here. The first few challenges occurred with the benchmark framework. Because it is not available as a package, we have to work with the source code, which needs to be moved after downloading in order to use it. There were also some dependencies missing. Additionally, the latest working Python version with that benchmark is 3.7. In order to use a supported Django server, at least Python version 3.8 is required. Thus, we need two separate virtual environments.

With the utilization of Docker combined with the need for two virtual environments, the required disk space is relatively high, averaging around 40GB during the build phase. Also, Docker does not delete old containers on its own. That needs to be triggered manually or after the build stage. Otherwise, even more disk space is used for every pipeline.

Additionally, starting from scratch has a significant overhead from setting up the server. This includes setting up a domain, having the ability to send emails that can be delivered to providers such as Gmail or GMX, installing Gitlab, setting up a reverse proxy using nginx, and managing SSL certificates.

6 Conclusion

This paper aimed to explore the adoption and implementation of MLOps in industry and academia, complemented by insights from a hands-on practical project. We looked at how MLOps is defined and which groups of people are involved. Then, we investigated applications and challenges in the industry when utilizing MLOps. Following that, we studied how this translates to academia and what challenges occur there. We then set up our project on what MLOps in academia can look like.

Ultimately, MLOps is a combination of machine learning and DevOps. Machine learning is part of artificial intelligence and a subset of deep learning. DevOps has been around since 2007, and MLOps is the specialization of it. For many teams within an organization, MLOps is relevant. This includes data scientists, machine learning engineers, data engineers, and, in the broader framework, software engineers and DevOps engineers. MLOps takes a comprehensive approach by considering the entire machine learning system rather than just the development aspect. While development typically aims to achieve good performance metrics, like accuracy, the MLOps perspective emphasizes the importance of deploying and maintaining machine learning models in a production environment. Important practices are scalability, reproducibility, and reliability. Ideally, in an MLOps system, the pipeline is automated. This starts with acquiring the data, preprocessing, and archiving for comparison and reproduction. It is followed by the training loop for one or multiple models. A model is usually trained multiple times with different hyperparameters in order to find the best one. That result is then deployed. After the deployment, the models are continuously monitored to avoid performance decreases.

In the industry, MLOps is relevant in all branches that already utilize machine learning, such as finance, manufacturing, retail, energy, transportation, media, and healthcare. We can use an uncountable amount of tools to build an MLOps pipeline. Thus, the goal of a project must be apparent from the beginning. We must also consider that acquiring and labeling data is one of the most complicated steps. Automation of this is a challenge that does not favor the MLOps philosophy.

In academia, we focus on discovery and exploration, which leads to challenges different from those in industry. The most significant challenges lie in time constraints and different knowledge bases.

However, resources differ between researchers. While some can work on high-performance clusters, others must work with their hardware.

In our project, we worked on an example in the field of computational biology. The dataset states whether immune cells can identify epitopes from viruses. We implement an MLOps pipeline using PyTorch, hyperopt, GitLab CI/CD, Docker, and Django. The pipeline automatically obtains the data, performs training and optimization of hyperparameters, and builds a Django server Docker container, which serves as a web interface and monitoring.

This project suggests that academia could benefit from the automation of MLOps. Once a framework is set, automation saves time and improves code quality. Additionally, it is closer to industry requirements, leading to a better adaption. While this project provides valuable insights, it is limited to the underlying dataset in the area of computational biology and its models. Other fields using data such as images or text or different models such as transformers would require heavier changes to the code base.

Future work could explore how to handle such different types of data and models. Additional work could be improving the end of the pipeline with improved monitoring and applications for users. By analyzing insights from diverse perspectives, this paper emphasizes the potential of MLOps, the future of machine learning development and deployment in academia, and closer collaboration with industry.

References

Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis, and George A. Papakostas. MLOps – definitions, tools and challenges. *CoRR*, abs/2201.00162, 2022. URL https://arxiv.org/abs/2201.00162.

Mark Treveil, Nicolas Omont, Clément Stenac, Kenji Lefevre, Du Phan, Joachim Zentici, Adrien Lavoillotte, Makoto Miyazaki, and Lynn Heidmann. *Introducing MLOps*. O'Reilly Media, Inc., 2020. ISBN 978-1-49208-329-0. URL https://learning.oreilly.com/library/view/introducing-mlops/9781492083283/.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. 28, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/86df7dcfd896fcaf2674f757a2463eba-Paper.pdf.

Prince Canuma. MLOps: What it is, why it matters, and how to implement it, 2024. URL https://neptune.ai/blog/mlops.

Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (MLOps): Overview, definition, and architecture, 2022. URL https://arxiv.org/abs/2205.02302.

Oswald Campesato. *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning and Information, 2020. ISBN 978-1-68392-467-8.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(85):2399–2434, 2006. URL http://jmlr.org/papers/v7/belkin06a.html.

Guillaume Chassagnon, Maria Vakalopolou, Nikos Paragios, and Marie-Pierre Revel. Deep learning: definition and perspectives for thoracic imaging. pages 2021–2030, 2019. URL https://doi.org/10.1007/s00330-019-06564-3.

Len Bass, Ingo Weber, and Liming Zhu. *DevOps: A Software Architect's Perspective (SEI Series in Software Engineering)*. Addison-Wesley, 2015. ISBN 978-0-13404-984-7. URL https://learning.oreilly.com/library/view/devops-a-software/9780134049885/.

Sigma Solve. Industry-specific MLOps use cases: Revolutionize AI deployment, 2023. URL https://medium.com/@sigmasolveusa/industry-specific-mlops-use-cases-revolutionize-ai-deployment-07f99b15c2a9.

Stephen Oladele. How these 8 companies implement MLOps: In-depth guide, 2024a. URL https://neptune.ai/blog/how-these-8-companies-implement-mlops.

European Central Bank. Payments statistics: first half of 2023, 2024. URL https://www.ecb.europa.eu/press/stats/paysec/html/ecb.pis2023~b28d791ed8.en.html.

JP Morgan. Machine learning center of excellence, 2024. URL https://www.jpmorgan.com/technology/applied-ai-and-ml/machine-learning.

Siemens. Ai model manager, 2024. URL https://www.siemens.com/global/en/products/automation/topic-areas/artificial-intelligence-in-industry/industrial-ai-enabled-operations/ai-model-manager.html.

MVTec. Highest level of quality control thanks to HALCON's deep learning, 2024. URL https://www.mvtec.com/application-areas/success-stories/highest-level-of-quality-control-thanks-to-halcons-deep-learning.

Bell Publishing. Mecatherm unveils turnkey solution for industrial baguette production, 2018. URL https://www.confectioneryproduction.com/news/19634/mecatherm-unveils-turnkey-solution-for-industrial-baguette-production-in-africa/.

Ivy Levine. How amazon is using generative ai to improve product recommendations and descriptions, 2024. URL https://www.aboutamazon.com/news/retail/amazon-generative-ai-product-search-results-and-descriptions.

Carlo Gutierrez. Shell builds 10,000 AI models on kubernetes in less than a day, 2021. URL https://www.altoros.com/blog/shell-builds-10000-ai-models-on-kubernetes-in-less-than-a-day/.

Rishabh Bhargava and Nihit Desai. Issue 15: AI for self-driving at Tesla. HuggingFace meets AWS. embedding stores. ML and databases., 2021. URL https://mlopsroundup.substack.com/p/issue-15-ai-for-self-driving-at-tesla.

Netflix. Machine learning platform, 2024. URL https://research.netflix.com/research-area/machine-learning-platform.

Ramūnas Berkmanas. MLOps for healthcare: The road to responsible AI in medicine, 2024. URL https://easyflow.tech/mlops-for-healthcare/.

Morris. MLOps is a mess but that's to be expected, 2022. URL https://mlops.community/mlops-is-a-mess-but-thats-to-be-expected/.

Matt Turck. Red hot: The 2021 machine learning, AI and data (MAD) landscape, 2021. URL https://mattturck.com/data2021/.

Leonhard Faubel, Klaus Schmid, and Holger Eichelberger. Mlops challenges in industry 4.0. *SN Computer Science*, 4(6):828, Oct 2023. ISSN 2661-8907. doi: 10.1007/s42979-023-02282-2. URL https://doi.org/10.1007/s42979-023-02282-2.

Evidently AI Team. What is data drift in ML, and how to detect and handle it, 2024a. URL https://www.evidentlyai.com/ml-in-production/data-drift.

Evidently AI Team. What is concept drift in ML, and how to detect and address it, 2024b. URL https://www.evidentlyai.com/ml-in-production/concept-drift.

Miranda Rutherford. The CLOUD act. *Berkeley Technology Law Journal*, 34:1177–1204, 2019.

Andrew Kusiak. Smart manufacturing must embrace big data. *Nature*, 544(7648):23–25, Apr 2017. ISSN 1476-4687. doi: 10.1038/544023a. URL https://doi.org/10.1038/544023a.

Chip Huyen. *Designing Machine Learning Systems*. O'Reilly Media, Inc., 2022. ISBN 978-1-09810-796-3. URL https://learning.oreilly.com/library/view/designing-machine-learning/9781098107956/.

Harald Semmelrock, Tony Ross-Hellauer, Simone Kopeinik, Dieter Theiler, Armin Haberl, Stefan Thalmann, and Dominik Kowald. Reproducibility in machine learning-based research: Overview, barriers and drivers. 2024. URL https://arxiv.org/abs/2406.14325.

Beyza Eken, Samodha Pallewatta, Nguyen Khoi Tran, Ayse Tosun, and Muhammad Ali Babar. A multivocal review of mlops practices, challenges and open issues. 2024. URL https://arxiv.org/abs/2406.09737.

Samadrita Ghosh. MLOps challenges and how to face them, 2024. URL https://neptune.ai/blog/mlops-challenges-and-how-to-face-them.

Jose Sanchez-Trincado, Marta Gomez-Perosanz, and Pedro Reche. Fundamentals and methods for t- and b-cell epitope prediction. *Journal of Immunology Research*, 2017:1–14, 12 2017. doi: 10.1155/2017/2680160.

Felix Drost, Anna Chernysheva, Mahmoud Albahah, Katharina Kocher, Kilian Schober, and Benjamin Schubert. Benchmarking of t-cell receptor - epitope predictors with epytope-tcr. *bioRxiv*, 2024. doi: 10.1101/2024.11.06.622261. URL https://www.biorxiv.org/content/early/2024/11/08/2024.11.06.622261.

Katharina Kocher, Felix Drost, Abel Mekonnen Tesfaye, Carolin Moosmann, Christine Schuelein, Myriam Grotz, Elvira D'Ippolito, Frederik Graw, Bernd Spriewald, Dirk H Busch, et al. Quality of vaccination-induced t cell responses is conveyed by polyclonality and high, but not maximum, antigen receptor avidity. *bioRxiv*, pages 2024–10, 2024.

Bruce A Adams, Payam Shahi, Daniel Reyes, Shamoni Maheshwari, Nima Mousavi, Sreenath Krishnan, Nandhini Ramen, FuNien Tsai, Poornasree Kumar, Peter Finnegan, et al. An integrated reagent and multimodal analysis workflow to enrich and characterize peptide-specific cd8+ t cells. *The Journal of Immunology*, 210 (1_Supplement):249–17, 2023.

Andrew P. McMahon and Adi Polak. *Machine learning engineering with Python: manage the lifecycle of machine learning models using MLOps with practical examples*. Packt Publishing, 2023. ISBN 978-1-83763-196-4. URL https://portal-igpublish-com.eaccess.tum.edu/iglibrary/search/PACKT0006868.html.

Stephen Oladele. How to learn MLOps in 2024 [courses, books, and other resources], 2024b. URL https://neptune.ai/blog/how-to-learn-mlops.