Managing Machine Learning Complexity with Advanced Version Control Techniques

Koushik Balaji Venkatesan
Independent Researcher
Seattle, WA

ABSTRACT
Managing the complexity of machine learning workflows is a significant challenge, as these projects often involve not just code but also large datasets, model maintenance, and extensive experimentation. While traditional version control tools like Git are effective for software development, they do not fully accommodate the unique requirements of ML workflows, such as tracking multiple dataset versions, managing evolving models, and maintaining experiment histories. Specific utilities and frameworks have been developed to address these challenges, and this paper explores some of these available tools in detail. Incorporating structured workflows and best practices for managing artifacts helps ML practitioners improve reproducibility, scalability, and collaboration across teams.

Furthermore, these tools can be leveraged as part of an end-to-end ML pipeline combined with CI/CD practices to facilitate tasks such as data preprocessing, model training, and deployment solutions. Through a hands-on case study of a retail recommendation system, this paper demonstrates how these techniques effectively tackle real-world challenges, including handling dynamic datasets, optimizing iterative experimentation, and maintaining model integrity. Finally, the paper explores emerging trends such as automation and sustainability in ML workflows, highlighting how integrating these strategies can enhance scalability and enable teams to build more efficient and production-ready ML systems.

General Terms
Algorithms, Machine Learning, Software Engineering, Data Management, Reproducibility

Keywords
Version Control, Machine Learning Engineering, Data Version Control (DVC), Experiment Tracking, Model Reproducibility

1. INTRODUCTION
ML has made a huge impact by bringing a data-driven solution to the complex problems at hand. However, developing and deploying an ML system brings in its unique set of challenges [1]. Unlike traditional software engineering, ML workflows extend beyond code management to handling datasets, training models, and experimentation in iterative manners. This added complexity creates barriers to reproducibility, collaboration, and managing the lifecycle of ML assets as projects evolve from research toward production.
Version control is a basic software engineering practice that is particularly important in organizing and managing changes in ML workflows. Traditional tools, however, are not sufficient for the dynamic and varied needs of ML projects. Large and constantly evolving datasets are hard to track, while models need versioning to manage configurations, weights, and metadata over time. Also, iterative experimentation demands the most stringent documentation of hyperparameters, configurations, and performance metrics. Without a robust version control system, ML projects can quickly degenerate into chaos where collaboration and reproducibility go for a toss.

In this paper, advanced versioning tools like Data Version Control (DVC) [2], MLflow [3], and Weights & Biases (W&B) [4] are discussed, which enhances these versioning practices to make interdependencies with datasets, hyperparameters & model artifacts effective. Utilities such as DVC extend traditional version control to handle datasets and pipelines effortlessly, while MLflow and W&B represent systems that embed features relevant to experiment tracking and model lifecycle management, respectively [5]. These tools introduce structured workflow management into a data scientist's flow of work, enhancing the overall efficiency and reliability of the process.

The paper covers deficiencies in traditional version control for ML, advanced tools for ML workflows, and best practices to be followed while implementing an effective versioning system. With these strategies and tools, ML practitioners will be able to bring further optimization into their workflows, thereby assuring better reproducibility and seamless transitions from experimentation to production-ready systems [6].

2. UNDERSTANDING VERSION CONTROL IN MACHINE LEARNING
Version control is an important aspect in the development of software because it provides a structured method to track changes, collaborate with others, and reproduce results. However, when applied to machine learning workflows, this concept requires solving many unique challenges. Unlike classic software projects, ML workflows have many components that include code, data, models, and experiments; all these have their own peculiar needs regarding tracking and versioning. This section discusses the application of version control to particular aspects of ML and also the challenges it resolves.

2.1 What is Version Control?
Version control, or management, in general, is the term used for the controlled handling of all changes of files, codebases, and generally any kind of artifacts in a structured and regulated fashion. Version control makes every change of code traceable and, therefore, easily recoverable and makes sharing work among people is easier. In ML, versioning goes beyond source code since there are datasets to be considered, model files, and experiment metadata because ML workflow is complex.

2.2 Importance of Version Control in Machine Learning
Version control has proved to be an indispensable aspect in machine learning due to several reasons:

2.2.1 Reproducibility
Reproducibility guarantees that the results can be reliably reproduced with the same dataset, configuration, and model version.

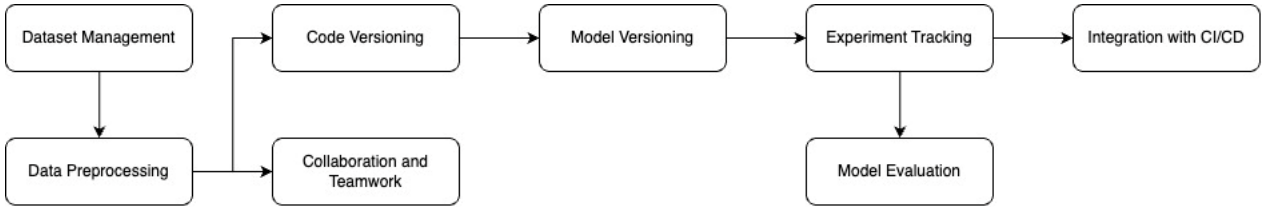

Fig 1: ML Version Control Workflow

2.2.2 Collaboration
Collaboration allows the opportunity for teamwork, where multiple people can work on different parts of a project while having a unified record of changes.

2.2.3 Scalability
Scalability allows for large datasets and multiple model iterations as projects grow in size and complexity.

2.3 Version Control Challenges Unique to Machine Learning

2.3.1 Dataset Versioning
By their nature, machine learning datasets change over time, and this happens frequently, in order to be in sync with the changes made to the models. Such traditional version control tools don't support large files, and at the same time, neither can they track data changes accurately. Tools like DVC let versioning of datasets be saved into some external storage and its references into version control systems like Git [8].

2.3.2 Versioning Models
Trained models get updated rather frequently during both experimentation and deployment. Versioning models involves tracking models' architecture, weights, and their training configurations. Tools like MLflow[2] and Weights & Biases[3] offer functionality to record model versions and compare them such that the best-performing configurations are preserved.

2.3.3 Experiment Tracking
By their very nature, machine learning workflows are iterative and require systematic tracking of hyperparameters, training metrics, and results. Unlike software projects where the final output is often a static application, ML projects are all about continuous experimentation to optimize performance. Tools like MLflow and W&B track these experiments, linking results to specific datasets, configurations, and code versions.

3. CORE COMPONENTS OF VERSION CONTROL IN ML
This section highlights various core components that are involved in ML workflows and some best practices for managing and tracking versions of those components. Version control in machine learning extends beyond traditional software practices and various interconnected components such as code, data, models and experiments should be versioned effectively.

3.1 Versioning Code
Code is the foundation of ML workflows, much like software engineering. But coding in ML encompasses project source code, scripts, notebooks and configurations for data processing. Traditional methods like Git work great for versioning code in ML workflows as well.

3.1.1 Branching Strategies
Always maintain a clean main/master branch for production-ready code. Experiments or bug fixes should be tracked in a separate feature branch forked off of the main branch.

3.1.2 Notebook Versioning
While Git integrates with notebook, widely used tools such as AWS code commit works great with AWS Sagemaker notebooks and offer git-like features to version code for machine learning projects. Converting these notebook logics to scripts and managing them outside with git simplifies collaboration and debugging.

3.1.3 Commit Messages
Ensure that clear and descriptive commit messages are added so that the purpose of any particular change moving through the pipeline can be tracked.

3.2 Versioning Data
Datasets can evolve dynamically in ML workflows due to data collection updates, changes in processing upstream, preprocessing logic changes, or domain shifts. It is critical to track these changes so that the model can be trained and evaluated consistently. Traditional version control tools like Git, don't offer much support when it comes to versioning data.

3.2.1 Data Version Control (DVC)
Tools such as Data Version Control (DVC) track data in files without having to store them in the git repository. It tracks various changes that might have been made to the data like more or less/data than before, changing schemas, or changes in the time that batched data is usually ingested. This approach also provides an efficient mechanism to handle large volumes of data.

3.2.2 Metadata Management
Always record metadata around the data that is being used such as time of ingestion, data source, preprocessing steps and versions. This ensures data lineage and reproducibility and helps with quicker investigations for critical issues.

3.2.3 Immutability
Taking model needs into consideration, after sufficient analysis a dataset is chosen for consumption. This chosen dataset should be locked for writes (immutable) to prevent accidental changes that can affect model training can be prevented.

3.3 Versioning Models
A trained model is one of the most critical components of ML workflows. Versioning the model becomes very important to track changes in model architecture, weights, and configurations over time. This is essential for reproducing results and deploying the most effective model.

3.3.1 Artifact Storage
Consider using tools like MLFlow to store trained models. This also helps store other metadata around the trained model such as parameters and evaluation metrics.

3.3.2 Model Tags
Assigning tags to models based on their purpose, for example, production-ready, experimental, etc. can simplify tracking.

3.3.3 Integration with Pipelines
Versioned models should also be linked to specific datasets and training scripts to provide end-to-end reproducibility.

4. VERSION CONTROL TOOLS FOR MACHINE LEARNING
Specific needs of machine learning workflows have led to a few tools that enhance conventional version control of datasets, models, and experiments. Such tools fix challenges visible within traditional systems like Git, which assist machine learning practitioners manage the complexities of their projects. There are various features and advantages of such increasingly popular version control tools. Over time, wider ML communities have adopted these tools embedding them into machine learning workflows.

4.1 Git
With its strong features to track changes in code, collaborate, and manage development workflows, GIT has emerged as the primary version control tool in use within software development. While absolutely indispensable for versioning scripts and configurations, Git proves to be limited when used with large datasets and model artifacts. Git has the following key features.

4.1.1 Code Management
Git takes care of branching, merging, and collaborative development of ML scripts and notebooks

4.1.2 Integration
Integrates with GitHub, GitLab, and Bitbucket to help teams collaborate and manage remote repositories

4.1.3 Limitations
Git struggles with large files, such as datasets and model weights, necessitating additional tools for effective management.

4.2 Data Version Control (DVC)
DVC is a software tool that manages the version of datasets and pipelines. DVC, when used with Git, is an open-source platform designed to disconnect the actual large data files from a Git repository and to be kept within external storage solutions while maintaining versions in Git. DVC offers a host of important features including:

4.2.1 Data Tracking
DVC allows for the tracking of variations in datasets and data-pipelines without the need for saving huge files in the repository.

4.2.2 Multiple Storage Options
DVC supports multiple storage-types including cloud-based storage (AWS S3 and Google Cloud Storage) on one hand and local storage on the other.

4.2.3 Pipeline Management
Users can define ML pipelines as directed acyclic graphs (DAGs) so that the workflow can be reproducible.

4.2.4 Integration with Git
DVC complements Git, linking data files and pipeline stages to a specific Git commit.

4.2.5 Example use-case
Some example use cases of DVC includes needing to keep track of different versions of the same dataset while preprocessing or to ensure experiments could be reproduced on the exact same dataset.

4.3 MLflow
MLflow is also an open-source platform to manage the complete end-to-end lifecycle of an ML project. Also available for experiment tracking, model versioning, and deployment, it acts as a complete solution for ML workflows. The following are the key aspects of MLflow

4.3.1 Experiment Tracking
Log hyperparameters, metrics, and artifacts for each experiment to make comparisons across runs.

4.3.2 Model registry
Provides a centralized repository to manage model versions with metadata and stage transitions, such as "staging" and "production."

4.3.3 Deployment tools
Supports model deployment into several deployment environments, such as cloud services or REST APIs.

4.3.4 Integration
MLflow integrates with well-known ML frameworks such as TensorFlow [9], PyTorch, and Scikit-learn.

4.3.5 Example use-case
An example of its use case is logging diverse training runs to contrast the performance of hyperparameter configurations and determine the best-fitting model.

4.4 Weights & Biases (W&B)
Another popular experiment tracking and visualization platform that has been on the market for a few years is Weights & Biases. It has the reputation of an intuitive tool that works well with typical ML workflows and focuses on the visualization part a little bit more. It might be convenient for data science teams, where cooperation on projects is key. Some important features are:

4.4.1 Experiment Tracking
Logs training runs, hyperparameters, and metrics support in-depth comparisons through easy-to-use interactive dashboards.

4.4.2 Visualization
Provides mechanisms to visualize training curves, loss functions, and other metrics right in an easy way.

4.4.3 Artifact Management
Helps with versioning of datasets, models, and other ML assets

4.4.4 Collaboration
Exposes options to share experiments, visualizations, and results between colleagues

4.4.5 Example use-case
Tracking the progress of training runs in real-time and sharing insights with team members to refine the ML model.

4.5 TensorFlow Model Garden
TensorFlow Model Garden is a TensorFlow-provided platform for versioning and sharing pre-trained models in a way that emphasizes ease of use with TensorFlow-based projects and guarantees reproducibility. Some key features include,

4.5.1 Model Repository
A model repository to maintain clear, well-documented, and versioned collection of pre-trained models.

4.5.2 Reproducibility
This makes sure that results can be replicated by users, through the provision of code, datasets, and configuration files.

4.5.3 Integration with TensorFlow
TensorFlow Model Garden is developed for full integration with TensorFlow platform and provides seamless integration.

4.5.4 Example use-case
Using a pre-trained model from TensorFlow Model Garden as a starting point for transfer learning while keeping track of custom modifications.

4.6 Comparison and Choosing the Right Tool
Choosing the appropriate option for version control depends on various circumstances and requirements such as,

4.6.1 Code and Basic Versioning
Git should be more than sufficient in this case as it provides enough versioning abilities to manage scripts and configuration files.

4.6.2 Dataset Management
DVC seems to stand out as the best choice to manage large data files and complex data pipelines.

4.6.3 Experiment Tracking
MLflow and W&B are ideal for logging and comparing training/experiment runs.

4.6.4 Collaborative Teams
W&B excels in sharing and visualization and stands out as the top choice for this category

4.6.5 TensorFlow Projects
TensorFlow Model Garden provides the best integration with TensorFlow and provides pre-trained model and reproducibility for TensorFlow users.

ML practitioners can build robust and scalable ML pipelines by leveraging the strengths of these tools and integrating them. Therefore, users must be mindful of the pros and cons of all relevant tools before committing to a particular option as it's not always straightforward to change tools after using a given option for an extended period. In the next section, best practices for using these tools and how they can fit into the ML workflow will be explored in detail.

5. OPTIMAL VERSION CONTROL IN MACHINE LEARNING

5.1 Establishing Clear Workflows
Clearly defined workflows create structure and consistency, ensuring that everyone on the team does things the same way. For better branch management and organization, stable and production-ready changes can be merged into the main branch while keeping experimental changes in another branch. Automated solutions such as DVC or CI/CD systems help automate repetitive tasks such as pipeline deployments and version tracking, ensuring human errors are prevented.

5.2 Automated Dataset and Model Tracking
Automated versioning of datasets and models is critical to help manage the complexity associated with ML workflows. Tools such as DVC facilitate dataset tracking by associating data versions to code commits while keeping larger files in scalable storage solutions e.g. cloud buckets. For models, there are tools like MLflow, W&B to make a model versioning easier, to track and track model artifacts with meta data like Hyperparameters, training configurations, performance metrics etc. These practices make it easier to identify the best-performing models and roll back to earlier versions if needed.

5.3 Standardize Naming Conventions
Following naming conventions helps reduce confusion, especially in larger projects where there are multiple users. Give responsive names to your datasets, experiments, and models, e.g., customer_data_v1.csv file for datasets or exp-01-learning-rate-0.01 for experiments. In organizations where there are multiple contributors, teams should come up with standard naming practices that can be followed across the organization. This helps with identifying the purpose and version of files under use, which can be an important data point for debugging.

5.4 Ensure Reproducibility
ML Models need to be reproducible to verify results and debug problems. Using tools like Conda or virtual environments to capture those dependencies allows for consistent execution across machines and environments. Techniques for keeping track of integration between code, datasets, and experiments (like using DVC or Git) help with traceability. Centralized logs or dashboards can document preprocessing, and model configuration steps resulting in allowing other researchers to reproduce experiments.

5.5 Foster Collaboration
Depending on the nature of the ML project, team members play diverse roles including Annotation and data labeling, Data Science, and Model development. Git or other shared repositories act as a centralized code, data, and documentation store. Pull requesting makes it possible to have pull request peer review processes to better validate contribution quality and consistency. Centralized dashboards (e.g. W&B, MLflow) help team members share the results of the experiments they have run, the insights drawn, and the progress made, thus improving transparency and minimizing misalignment.

5.6 Integrate Continuous Integration and Deployment (CI/CD)
CI/CD pipelines increase efficiency by automating the testing workflow of ML workflows, deployment, etc. Automatic data preprocessing, model validation, and deployment processes can help transitions between development and production go smoothly. Tools such as Kubeflow [12], MLflow, etc. are helping in simplifying this part of pipelines and log the performance in production. They also help configure and raise alarms for example, in case of data drift, or performance degradation.

5.7 Plan for Scalability
ML projects often tend to extend beyond original intentions, so scalability becomes a key property. Distributed storage solutions such as HDFS or cloud storage systems enable efficient storage for large datasets. Scalable training platforms like Kubernetes help teams cope with increasingly complex workflows while pulling version control for all artifacts together into a unified system. Scaling ML workflows requires efficient data pipelines that integrate distributed processing frameworks. Inspired by approaches such as MapReduce [11], modern ML pipelines incorporate data versioning, caching, and lineage tracking to manage dataset evolution.

Applying these practices enables teams to build efficient, collaborative, and scalable ML workflows. These best practices would give ML projects the capability to be reproducible and organized while also being upgradable to new research and

production challenges.

6. CASE STUDY: VERSION CONTROL IN A MACHINE LEARNING WORKFLOW

6.1 Scenario Overview

For the case study, let's consider an e-commerce platform that needs to add a recommendation feature to its set of services. This feature will be used by merchants to suggest personalized product recommendations based on customer behavior data such as sales interactions and preferences. The team building this product faces a few challenges. As data becomes larger, it becomes more complex to track and manage various model versions. They also need to keep track of the updates made to data as it becomes larger over time. Managing safe deployments without affecting customers is another challenge. At this point, reproducibility, scalability, and seamless collaboration are critical to ensuring the project's success.

6.2 The Challenges

The team's problems start with data management. They collect customer behavior data, like clicks, purchases, product views, and product details. This data changes frequently as new customers join, old products are discontinued, and new features are added. Tracking these changes and knowing exactly which version of the data was used for training a model proves to be difficult. Data preprocessing steps, like cleaning and feature extraction, add another layer of complexity, often introducing errors that are hard to trace.

Model management is another challenge. The team tries multiple approaches—like collaborative filtering and deep learning-based methods, resulting in a growing list of trained models. Each model comes with unique configurations and performance metrics, but when performance drops in production, identifying the problematic version or rolling back to a previous one becomes a daunting task.

Tracking experiments is becoming harder over time. As multiple teams work on different components of the system, tracking what, why, and how of various model versions is not easily achievable. Also, ensuring that all team members stay informed and are on the same page about different stages such as development, testing, and production, adds another layer of complexity.

6.3 The Proposed Solution

To tackle these challenges, the team decides to take a formal version control approach with its own tools adapted specifically for a machine learning workflow.

6.3.1 Data Versioning with DVC

To manage the dynamic nature of the dataset, the team implements Data Version Control (DVC) to version datasets alongside code. DVC allows the team to store large datasets in external storage (e.g., AWS S3 or Google Cloud Storage) while keeping lightweight metadata in Git.

Each dataset version is linked to a specific Git commit, enabling the team to trace back to the exact dataset used for any experiment [10]. Additionally, DVC pipelines are used to version preprocessing steps, such as data cleaning and feature extraction, ensuring that any changes to the data pipeline are tracked and reproducible. This approach guarantees that team members can replicate experiments using the exact dataset and preprocessing steps, even as the data evolves over time.

6.3.2 Model Versioning with MLflow

For model versioning, the team adopts MLflow to manage the lifecycle of trained models. Each model is logged in MLflow with metadata such as hyperparameters, training configurations, and evaluation metrics. The MLflow Model Registry is used to organize models into stages (e.g., 'staging,' 'production'), allowing the team to track which versions are ready for deployment.

If a model underperforms in production, the team can easily roll back to a previous version by referencing the registry. Additionally, MLflow's integration with CI/CD pipelines ensures that models are automatically validated and deployed, reducing the risk of human error. This structured approach to model versioning ensures that the team can maintain model integrity and quickly respond to performance issues.

6.3.3 Experiment Tracking with Weights & Biases (W&B)

To streamline experimentation, the team integrates Weights & Biases (W&B) into their workflow. W&B logs hyperparameters, training metrics, and performance visualizations for each experiment, providing a centralized dashboard for real-time monitoring. The team uses W&B's interactive dashboards to compare multiple experiments side-by-side, identifying trends and understanding the impact of different hyperparameter configurations.

For example, they can visualize how changes in learning rate or batch size affect model convergence and performance. W&B also facilitates collaboration by allowing team members to share experiment results and insights, ensuring that everyone stays aligned on the progress and direction of the project. This level of transparency and real-time feedback accelerates the experimentation process and helps the team identify the best-performing models more efficiently.

6.3.4 Collaboration and Integration

Git remains the foundation for versioning code and facilitating collaboration within the team. By integrating Git with DVC and MLflow, the team creates a unified system that links datasets, models, and code changes. Continuous Integration (CI) pipelines are set up to automate testing and deployment processes, ensuring that changes to code or data are validated before being merged into the main branch.

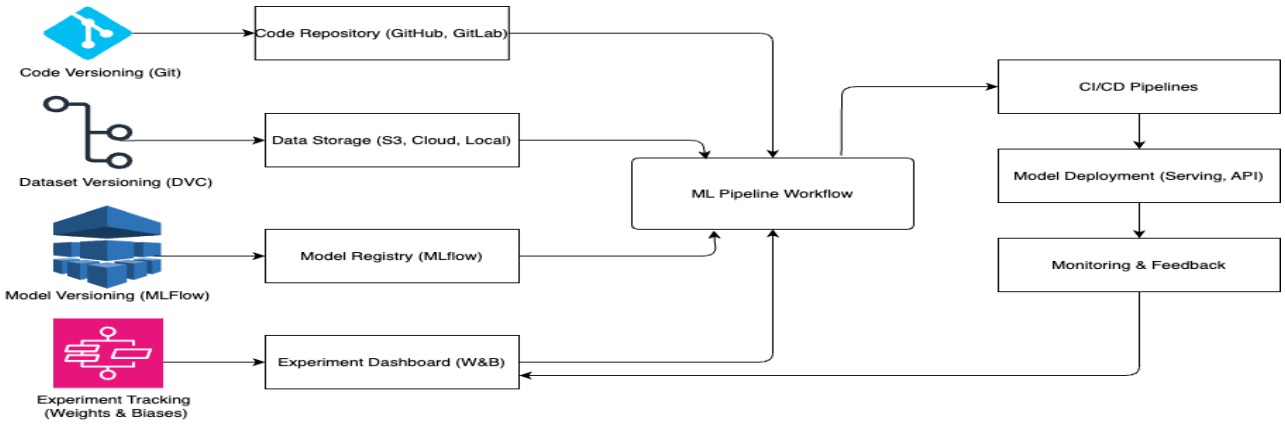

Fig 2: ML Versioning and Experiment Tracking Pipeline

For example, when a new dataset version is pushed, the CI pipeline automatically triggers data validation checks and retrains the model if necessary. This integration not only improves efficiency but also reduces the risk of errors during deployment. Additionally, the team uses pull requests and code reviews to ensure that contributions are thoroughly vetted, maintaining high-quality standards across the project.

To effectively manage dataset updates, model retraining, and deployments, our case study follows the structured version control workflow shown in Figure 2. This pipeline ensures tracking across all ML assets.

6.4 Expected Outcomes

By adopting the proposed version control approach, the team expects to achieve several significant improvements in their machine learning workflow. First, the use of DVC for dataset versioning ensures full reproducibility of experiments, as each dataset version is linked to a specific Git commit. This eliminates discrepancies between training and evaluation datasets, leading to more reliable results. Second, MLflow's model registry streamlines model lifecycle management, allowing the team to version and tag models based on their stage (e.g., 'staging,' 'production'). This makes it easy to identify which versions are ready for deployment and enables quick rollback in case of performance issues.

The integration of Weights & Biases (W&B) further enhances the workflow by providing real-time tracking of hyperparameters, metrics, and training curves. This allows the team to monitor progress and make data-driven decisions during model development. Side-by-side comparisons of experiments help identify the best-performing configurations, reducing the time spent on trial-and-error. Additionally, the combination of Git, DVC, and MLflow fosters seamless collaboration among team members, ensuring that everyone has access to the latest datasets, models, and experiment results.

Finally, the implementation of CI/CD pipelines automates repetitive tasks such as data validation, model training, and deployment. This not only speeds up the development process but also reduces the risk of human error. The proposed approach is designed to scale with the growing complexity of the project, ensuring that the system can handle larger datasets and more complex models as the project evolves. These improvements collectively enhance the team's ability to deliver a robust, scalable, and production-ready recommendation system.

6.5 Lessons Learned

This case study illustrates how complexities in a modern machine learning workflow can be managed with a combination of tools like DVC, MLflow, and W&B and applying good version control practices. By systematically organizing their datasets, models, and experiments, the team tackles their troubles and lays the groundwork for a scalable, reproducible recommendation system. This example highlights the importance of adopting specialized tools in ML engineering, showing that thoughtful planning and the right tools can make even the most complex projects run smoothly.

7. CHALLENGES AND LIMITATIONS

Although versioning tools and practices offer many advantages for the management of ML workflows, they do have some challenges and limitations. Understanding these challenges helps practitioners avoid pitfalls and improve their workflows.

7.1 Scalability Issues

Dealing with scalability, particularly with datasets, is one of the most difficult challenges in ML version control [13]. And, unlike code repositories, ML projects typically contain large datasets and model artifacts that can rapidly exceed local storage or standard Git repositories. Even with the use of tools such as DVC that allow the external storage of large files, the management and file access on distributed environments need solid infrastructure for that, which can lead to high costs, mainly if using cloud solutions.

7.2 Complexity of Integration

While moving a workflow into production, integrating with various tools can be cumbersome and time-consuming. DVC, MLflow, Weights & Biases and other tools to track models often need pre-existing Git and pipelines with a good amount of tweaking to work nicely together. Making sure these tools and environments (local development, cloud platforms, on-premises systems etc.) are compatible can create one more level of complexity.

7.3 Learning Curve

ML version control comes with its unique workflows and terminology that can create a steep learning curve for team members not familiar with it. Examples can be establishing pipelines with DVC or configuring experiment monitoring with MLflow − they need background knowledge − familiar with command-line tools, running the script, or integration processes. It takes time to train and onboard team members to use these tools, which can slow down initial progress.

7.4 Lack of Standardization

While general version control tools can be used for most ML workflows, there is not yet a widely adopted standard for version control of ML itself, despite the recent emergence of

ML-specific version control tools. Common teams had to stitch together several tools touching on different aspects of the workflow (data tracking, model versioning, experiments logging, etc.). Which can cause inconsistency and more difficult to collaborate between the organization or project.

7.5 Cost and Resource Overheads

Version control can become another overhead of working with ML projects at scale. Hosting massive datasets in the cloud, procuring compute for tracking and logging, and needing to pay for premium tools licensing can all inflate the project's budget. In addition, supporting the nuances of pipelines and CI/CD requires engineering resources, which might not always be advantageous for smaller teams or organizations.

7.6 Reproducibility in Dynamic Environments

In the case of dynamic environments where data and requirements change often, reproducibility is still a challenge. Keeping parts of the workflow (code, data, models) in sync takes a lot of work. Small oversights in larger projects can lead to irreproducible experiments or deployment issues. Tackling these challenges involves not only the selection of the right tools but also building good workflows, training team members, and investing in the infrastructure to accommodate the scale and complexity of ML projects.

8. FUTURE DIRECTIONS

ML engineering is a fast-evolving industry, and as ML workflows keep getting increasingly complex over time, practices and tools for version control will need to evolve in parallel. Following are some key emerging trends and future directions likely to emerge within the ML version control landscape.

8.1 Advances in Automation

With ML workflows only growing larger and becoming more complex over time, automation is poised to be a critical underpinning component in making versioning easier to navigate. In the future, tooling themselves are going to embed AI and machine learning capabilities to automate the way of things such as change detection, configuration identification, rollback suggestions, etc. This will increase overhead for managing datasets, models and experiments.

8.2 Standardization of Practices

In the absence of a standard method for ML versioning, community-wide efforts will help move us toward standardization. In reputable organizations and open-source communities, best practices and single frameworks for end-to-end ML workflows would possibly be employed. With standardized pipelines, tools become easier for teams to onboard with and help them to work across organization boundaries.

8.3 Enhanced Collaboration Features

Collaboration tools will emerge for machine learning to help more distributed teams working on machine learning projects work together. Sharing experiments in real time, interactive dashboards and shared cloud-based repositories would become more seamless and frictionless in the team workflow. Enhanced role-based access controls might also help keep things safe and efficient when teams work together.

8.4 Improved Scalability

Future tools will focus more on addressing current limitations in handling large datasets and models. Distributed version control systems and optimized cloud storage solutions will make sure that even the largest ML projects are kept in control. High-performance computing in conjunction with edge computing frameworks enables scale in training and deployment.

8.5 Focus on Sustainability

The increasing awareness of MLE (Machine Learning Engineering) is expected to promote the upcoming tools and practices of version control to become more sustainable. With this will come a need for efficient tracking of data driven decisions and models, along with optimizations leading to energy-efficient computation. Tools might develop functionality to evaluate and reduce the carbon footprint of ML pipelines.

8.6 Integration with Emerging Technologies

As technologies like federated learning and edge computing gain traction, version control systems will need to adapt. For example, managing decentralized datasets and models in federated learning scenarios presents unique challenges. Similarly, tracking and versioning models deployed on edge devices will require new strategies.

8.7 AI-Assisted Version Control

AI will most likely heavily impact version control in ML if it can make predictions to help with frequent workflow problems. It could even suggest which type of model to use based on previous results, identify duplicate experiments, and optimize data cleaning or preparation pipelines, for instance. Such breakthroughs could greatly help with the trial-and-error process of ML development.

8.8 Integration with End-to-End Platforms

Finally, E2E ML platforms are gaining traction and have tightly integrated version control into the entire ML lifecycle. Such platforms are tools like Kubeflow and Vertex AI, and as organizations strive for a more unified approach to their individual systems for any ML workflows, adoption will likely continue to grow.

Instead of manual version tracking, the future of version control in machine learning is more likely to be collaborative, automated, and scalable, allowing teams to take on more complex problems with confidence. Keeping up with these trends allows practitioners to be ready to embrace next-generation tools and workflows, accelerating efficiency, reproducibility, and innovation in ML initiatives.

9. CONCLUSION

Version control has emerged as a critical part of machine learning (ML) workflows — helping this once-nascent field transition from the experimental into production. Different from regular software development where the focus is mainly on the traditional code and its versions, ML projects face extra hurdles due to the need for various artifacts such as datasets, trained models, and experiment configurations. The ML workflow is typically more complex and iterative, which provides additional challenges for reproducibility, scalability and collaboration that require specialized tools and practices to address.

In this paper, shortcomings of the standard version control were discussed along with the need to shift to tools like Data Version Control (DVC), MLflow and W&B for better understandability of model performance with time. Together, these tools plus best practices like standardizing workflows, automating the

tracking of artifacts, and enhancing collaboration form a strong foundation for effectively managing ML projects. However, issues with scalability, integration complexity, and a lack of standardization remain across tools and methodology. However, recent trends are driving ML version control towards automation, standardization, and sustainability. New technologies, such as AI-assisted version control and end-to-end ML platforms, will build on those successes and trend toward better and easier ML workflow management.

Leveraging the strategies and tools covered, ML practitioners can construct well-structured, reproducible, and scalable workflows that enable smooth progression from research to production. Such practices enhance the efficiency and collaboration elements of these properties but also constitute the basis for innovation in MLE and the ability of teams to tackle more complex use cases with the same commitment to reliability

10. REFERENCES

[1] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, and J.-F. Crespo, "Hidden technical debt in machine learning systems," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2015, pp. 2503–2511.

[2] DVC Documentation, "Data Version Control (DVC): Git for Data & Models," Available: https://dvc.org/, Accessed: Jan. 2024.

[3] MLflow Documentation, "MLflow: Open-source platform for machine learning lifecycle," Available: https://mlflow.org/, Accessed: Jan. 2024.

[4] Weights & Biases Documentation, "Experiment tracking for machine learning teams," Available: https://wandb.ai/, Accessed: Jan. 2024.

[5] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, M. Hong, and A. Konwinski, "Accelerating the machine learning lifecycle with MLflow," in *Proc. Conf. Mach. Learn. Syst. (MLSys)*, 2016.

[6] D. Baylor, E. Breck, H. Cheng, N. Fiedel, and N. Polyzotis, "TFX: A TensorFlow-based production-scale machine learning platform," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, 2017.

[7] J. F. Pimentel, L. Murta, V. Braganholo, and J. Freire, "A large-scale study about quality and reproducibility of Jupyter notebooks," in *Proc. 16th Int. Conf. Mining Softw. Repositories (MSR)*, 2019.

[8] GitHub Documentation, "Git: Distributed version control system," Available: https://github.com/, Accessed: Jan. 2024.

[9] TensorFlow Model Garden, "Pre-trained models for reproducibility," Available: https://www.tensorflow.org/lite/models/, Accessed: Jan. 2024.

[10] E. Muškardin and T. Burgstaller, "Active model learning of Git version control system," in *Proc. IEEE Int. Conf. Softw. Test. Verification, and Validation Workshops*, 2024.

[11] M. Zaharia, A. Chen, et al., "Data pipeline in MapReduce," in *Proc. IEEE 9th Int. Conf. e-Science*, 2013.

[12] Kubeflow Documentation, "Machine learning toolkit for Kubernetes," Available: https://www.kubeflow.org/, Accessed: Jan. 2024.

[13] J. F. Pimentel, et al., "Version control challenges in ML," *Int. J. Softw. Eng.*, 2019.