Visualization Tools for Machine Learning Pipelines: A Review

Valentina Golendukhina*
University of Innsbruck
Innsbruck, Austria

Michael Felderer †
German Aerospace Center (DLR)
Institute for Software Technology
University of Cologne
Cologne, Germany

Lisa Sonnleithner ‡
CDL VaSiCS, LIT CPS Lab
Johannes Kepler University Linz
Linz, Austria

ABSTRACT

The increasing complexity and adoption of machine learning (ML) pipelines has led to a rising demand for effective visualization tools. This paper presents a comprehensive review of existing tools for visualizing data flow in machine learning (ML) pipelines. We highlight the tools' purposes, integration methods, and visualization techniques. We collected and analyzed 22 open-source tools and concepts, analyzing their features and classifying them based on their primary purpose. Our analysis revealed six main purposes of visualization tools: exploration, explanation, visual development, comparison, monitoring, and domain-specific tools. We provide an analysis of their integration methods, from standalone visual interfaces to code-level libraries, as well as a review of various visualization techniques, including Directed Acyclic Graphs (DAGs), pipeline matrices, and annotated visualizations. Our findings highlight the importance of visualization in enhancing the interpretability and efficiency of ML workflows. Moreover, the paper provides key limitations and challenges in current visualization methods to promote future research directions enhancing the usability and functionality of ML pipeline visualization tools.

Index Terms: Human-centered computing—Visualization—Visualization systems and tools; Computing methodologies—Machine Learning

1 INTRODUCTION

Visualization tools play a critical role in enhancing the understandability of machine learning (ML) systems, particularly as these systems grow increasingly complex and handle vast amounts of data. From data preprocessing to model evaluation, each stage of an ML pipeline involves numerous interconnected steps that can be challenging to track and optimize, especially as the size and complexity of datasets and models continue to grow. Using visualization tools is one of the strategies to overcome this challenge. Beyond their utility in the development phase, visualization tools are also valuable for debugging, monitoring, and understanding ML models and data, helping practitioners maintain transparency and control over their workflows [15].

While numerous tools have been developed to address these needs, their functionalities, the stages of the ML pipeline they support, and their target audiences vary considerably. Additionally, each tool is designed to work with specific models, data types, and integration requirements, making tool selection and usage highly dependent on the context of an ML task.

*e-mail: valentina.golendukhina@uibk.ac.at
†e-mail: michael.felderer@dlr.de
‡e-mail: lisa.sonnleithner@jku.at

In this paper, we explore various visualization tools designed specifically to cover the whole ML pipeline. We examine their integration specifications, purposes, and visualization techniques used.

Thus, the purpose of this paper is twofold. First, it aims to help anyone seeking to choose the most appropriate visualization tool for their ML workflows, whether they are practitioners, educators, or researchers. Moreover, it provides insights into different visualization methods for those who aim to implement their visualizations. Finally, it identifies gaps and challenges in current solutions for future tool development. Our study provides the following contributions:

- A list of visualization tools used for data flow visualization of ML pipelines and their classification by purpose of use and integration methods.
- A description of the current common and applied methods of visualization for data flow in ML pipelines.
- An analysis of challenges and limitations of the existing visualization tools.

The remainder of the paper is structured as follows: Section 2 presents relevant background information on ML pipelines and related work. Section 3 describes the research procedure of this study. Section 4 provides the results on the identified visualization tools, integration methods, and their purposes. In Section 5, the main visualization methods are shown and described, whereas Section 6 highlights the challenges and limitations of the visualization tools. Finally, Section 7 concludes the paper.

2 BACKGROUND

This section explains the main concepts and stages of the ML pipeline and provides information on the related work in the field of ML visualization.

2.1 Machine Learning Pipeline

A machine learning (ML) pipeline is a sequence of steps for the development, deployment, and maintenance of ML models. It provides a systematic approach for data ingestion and preparation for further model training and evaluation.

A generalized ML pipeline was described by Amershi et al. [1]. They distinguish nine main stages of an ML pipeline: model requirements, data collection, data cleaning, data labeling, feature engineering, model training, model evaluation, model deployment, and model monitoring. However, depending on the data type and type of the ML model, additional stages can be included in the ML pipeline structure. The complexity varies depending on the project requirements and the domain of application. For instance, simple pipelines may focus on basic data preprocessing and model training, while more advanced pipelines may include complex feature engineering, multiple model comparisons, or ensembling techniques.

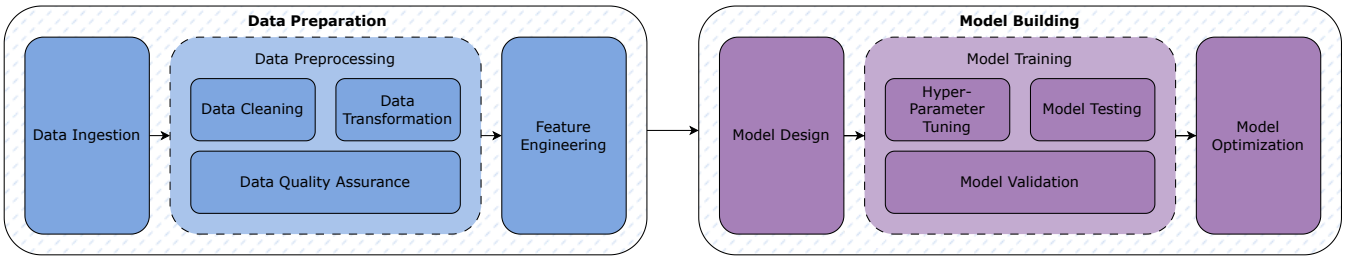

Figure 1: The main stages of a generalized machine learning pipeline. Data-related stages include data ingestion, data preprocessing, and feature engineering. Model-related building contains model design, model training, and model optimization. The pipeline includes the stages that can be visualized and benefit from visualization tools assistance.

Overall, an ML pipeline can be divided into two main stages: data preparation and model building. Each of these stages and several steps within them could benefit from visualization aid. Figure 1 illustrates the steps within an ML pipeline that could be visualized.

2.2 Related Work

The visualization of ML pipelines is a broad area of research, driven by the increased complexity of ML workflows and the need for interpretability and understanding. A lot of work is done to visualize one particular stage of an ML pipeline.

A large number of tools prioritize solely model development [20, 35, 38]. Such tools focus on the interpretability of the training process, especially in Deep Learning (DL) models, where tool support is needed to monitor a plethora of model parameters during the training process.

Several works provide an overview of visualization techniques for ML model visualization. The survey of Hohman et al. [15] addresses visualization for deep learning (DL) and provides an overview of visualization tools for DL training. The survey covers the tools useful for each component of neural networks that could be visualized and visualization techniques specific to certain purposes. Choo and Liu [4] made an overview of the visualization tools for explainable deep learning. Seifert et al. [29] focused on the visualization of DL in computer vision. Chatzimparmpas et al. [3] summarize the findings on ML model visualization by providing a survey of surveys, including the surveys mentioned above.

Other tools focus on data visualization and understanding. Different tools prioritize different types of data, such as tabular [2], highly multidimensional data [8], or unstructured text [32]. Furthermore, several surveys explored data visualization tools and techniques for training data [34, 37].

Although the tools that prioritize specific steps of ML development are undeniably important and highly useful, there are situations where it is crucial to have an overview of the entire pipeline. To the best of our knowledge, there are a lot of tools focusing on the ML pipeline as a whole, but there is no overview of such tools and their classification.

3 Methodology

In this section, the research methodology is defined, detailing the approach taken to conduct the research. First, we defined the research questions we aim to answer. Afterward, we specified the search engines and the search terms used to conduct the research. We then defined the selection criteria. To systematically collect all tools relevant to this study, we followed the multivocal literature review (MLR) guidelines [10] [18]. The full list of the accepted and rejected tools, the selection process, and their evaluation are available on Zenodo[1].

[1] https://zenodo.org/records/14185646

3.1 Research Questions

With this paper, we aim to enhance the understanding of the current state of the art of data flow visualization in machine learning pipelines. Therefore, we derive the following research questions:

RQ1: What are the existing methods and tools for visualizing data flow within machine learning pipelines?

RQ2: Which types of visualization are utilized for machine learning pipelines?

RQ3: What are the limitations and challenges of the tools for visualizing data flow within machine learning pipelines?

RQ4: What are the future research directions in data flow visualization for machine learning pipelines?

For the first research question, the goal is to investigate the existing tools designed for visualizing data flow within machine learning pipelines. This exploration aims to provide an overview, benefiting researchers, practitioners, and others seeking insight into ML visualization resources.

Regarding the second research question, the objective is to determine which types of visualization are utilized for specific stages and contexts within the ML pipeline. Since the ML pipeline involves several distinct steps, understanding where these tools fit and their suitability for each phase is crucial. We consider the following phases: data ingestion, data validation, data preprocessing, model training, and model validation.

The third research question focuses on identifying the limitations, weaknesses, and challenges in current visualization tools and methods. With that, we aim to provide guidance on tool selection. Understanding potential limitations offers insights into what may go wrong or what may not be achievable beforehand.

The last research question aims to explore new ideas and innovations for enhancing data flow visualization in ML pipelines. This can be seen as a follow-up to the preceding question, helping identify areas in need of further exploration within data flow visualization techniques.

3.2 Search Strategy

In our research, we included the usage of multiple search engines. Google and Google Scholar were substantial since they cover a wide range of sources, especially for less formal or grey literature. Additionally, we made use of the ACM Digital Library to focus on more academic or white literature sources.

The search strategy targets resources discussing data flow in machine learning pipelines combined with terms like visualization, graphical representation, monitoring, and dataflow analysis. By using this approach, we created the search string shown in Listing 1 to gather a wide range of information.

```
("visualizing dataflow" OR "dataflow visualization
↪  tools" OR "graphical representation" OR "monitor
↪  dataflow" OR "dataflow analysis")
  AND
("machine learning pipelines" OR "ml pipelines" OR "ml
↪  workflows")
```

Listing 1: The *search string* for the review.

Finally, we planned one iteration of backward snowballing to identify literature that we possibly missed with the search term. This approach ensures that we gather a diverse set of materials, covering different types of information about data flow visualization in machine learning pipelines.

3.3 Selection Criteria

To refine and limit the search results and to identify the most appropriate resources related to the research questions, we defined inclusion and exclusion criteria:

Inclusion Criteria:

- Has to be relevant to the research questions, specifically addressing data flow visualization within the context of machine learning pipelines.
- Where several studies have reported the same results, only the most recent paper will be considered.
- The publication date is within the last 10 years.
- Is written in English.

Exclusion Criteria:

- The paper is not accessible.
- Only the abstract but not the full text is available and accessible.
- The paper is not related to at least one aspect of the research questions.
- Visualization is only a side topic.
- Duplicate or redundant sources.

3.4 Conducting the Review

In total, we retrieved 130 publications from ACM, 170 from Google Scholar, and 100 from Google. We then filtered these publications, based on the previously defined inclusion and exclusion criteria. After checking the titles, we excluded 250 papers. We checked the abstracts of the remaining 150 papers and excluded another 99 papers. We then checked the full text of the remaining 51 papers and included 16 for analysis.

In many cases, the primary focus of these papers was not on visualization methods, or they only referenced relevant GitHub repositories without further discussion. To address this gap, we employed backward snowballing, which led to the identification of an additional 13 relevant sources.

By utilizing Research Rabbit, we identified additional, more specialized papers—such as those on visualization techniques for Natural Language Processing (NLP) that we did not find with the initial search terms. After checking the abstracts and full texts of these papers, this process resulted in the inclusion of four additional sources.

However, after analyzing the functionality of the selected tools, a part of them prioritized one particular ML pipeline stage and not the pipeline as a whole, e.g., some tools focused on data ingestion and monitoring or model training stages. These tools were excluded from the final list.

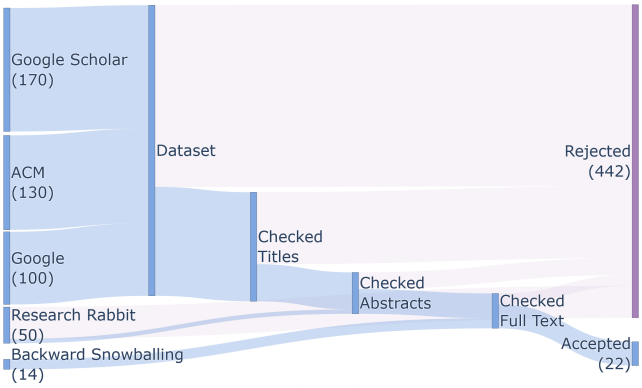

Figure 2: Retrieved sources and included sources in the literature review.

Overall, we incorporated a total of 16 sources from white literature and 6 from grey literature in this review. The overall process leading to these sources is also shown in Figure 2.

To identify the characteristics and purposes of the tools, two researchers independently reviewed each tool and assigned descriptive labels to them. The final set of labels from all tools was analyzed and coded to derive high-level themes representing the purposes. Similarly, different visualization strategies were collected and recorded from each tool to present the various ways of ML pipeline visualization.

4 RQ1: Pipeline Visualization Tools

After selecting and analyzing the available pipeline visualization tools, certain characteristics and specifications were discovered. The tools varied largely in their integration and compatibility requirements. Moreover, they were developed to support various purposes although all of them focused on the visualization of ML pipelines. This section uncovers the different ways visualization tools are integrated with ML development and the different goals of such tools. Table 1 provides an overview of the tools and their integration and purpose specificity. The thematic distribution of the visualization tools by integration specification and purpose is also shown in Figure 3.

4.1 Integration Specifications

Developers have implemented various methods to integrate visualization tools in ML pipelines, each having specific requirements and covering certain user needs. Some tools are implemented as **full systems with a visual interface** that offer comprehensive environments for pipeline visualization [6, 13, 14, 26]. These systems can function in two primary ways. One way requires the usage of ML pipeline *artifacts*, such as pre-existing data processing scripts, models, and metadata, in a specific format to generate visualizations. Such tools may require either that models be developed in a particular ML framework, such as TensorFlow [2], or datasets or metadata to be in specific formats, e.g., CSV or JSON, to generate compatible visualizations.

Another way is by providing end-to-end capabilities that allow users to design, develop, and visualize entire pipelines within the tool itself using the in-built functions and methods [19, 27, 28]. Such tools are known as visual development or no-code software. They are represented by browser-based systems or full stand-alone software. The systems can be running locally or using cloud capabilities. Moreover, already existing software systems provide an option to

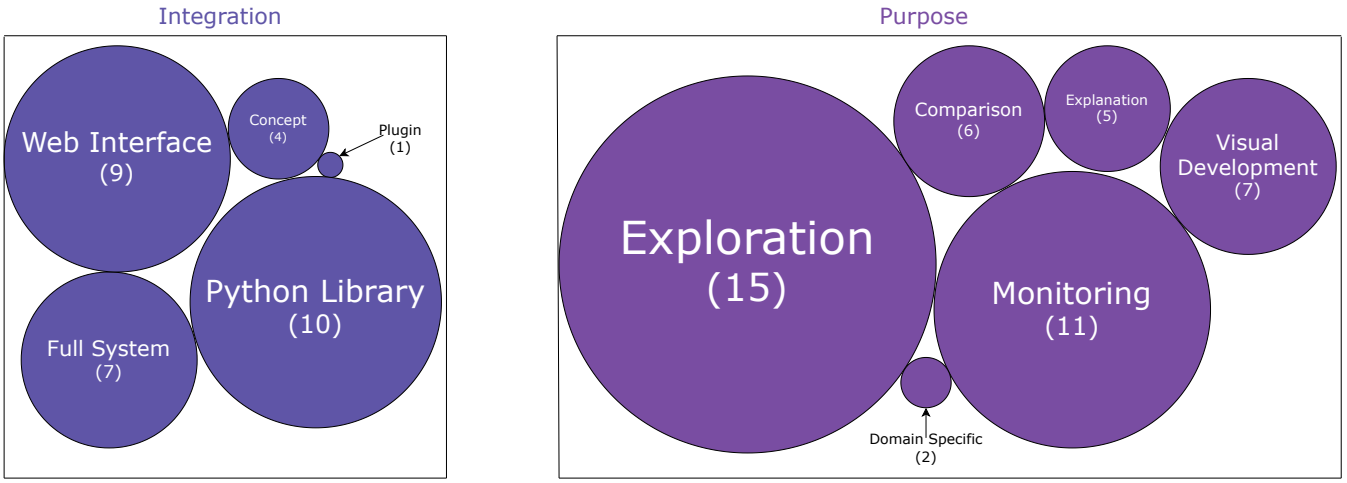

Figure 3: Thematic distribution of the visualization tools by integration specification and purpose. Since one tool might have more than one integration method and purpose, the sum of all topics does not add up to the number of tools described.

implement plugins expanding the capabilities of the original system [31].

Alternatively, many visualization tools are integrated at the **library level**, offering flexibility through packages and APIs that developers can incorporate directly into their code and generate visualizations in the development environments [5, 7, 11, 23]. These library-based tools are typically available in popular programming languages like Python and JavaScript. Given Python's extensive use in the ML community and compatibility with many ML frameworks, Python was the most identified language in the visualization tools.

4.2 Tools Classification

In addition to integration methods, visualization tools for ML pipelines serve specific purposes that further highlight their different development workflows. After screening various visualization tools, we identified six main purposes: exploration, explanation, visual development, comparison, monitoring, and domain-specific tools. Each purpose is described in more detail in the following subsections.

4.2.1 Exploration

One of the most frequently provided purposes of visualization tools for ML pipelines is exploration. Tools focused on exploration enable users to facilitate the understanding of datasets, model structures, and data transformations at various stages of the pipeline. The exploratory function aims to provide an overview of the pipeline and its stages and generates information about the characteristics and quality of the data, the main functions, methods, and libraries, and the sequences of steps in the pipeline. By breaking down the pipeline into sequential stages, from data preprocessing to model training and evaluation, these tools provide a structured view of the entire process. The tools aim to make complex workflows more accessible allowing more efficient knowledge transfer and an increased understanding of the underlying ML pipeline operations.

Different tools providing exploration support vary in their integration methods and supported models. [35] provides a step-by-step visual representation of pipeline construction based on the models created in TensorFlow. [30] is a visualization tool for PyTorch and TensorFlow models visualization, that works in Jupyter Notebook and allows real-time visualization during ML training. [24] provides a Python library for pipeline logging and supports common model structures, such as Paddle, ONNX, and Caffe, providing detailed static and dynamic graphs, i.e., the graphs that indicate the direction of data flow.

4.2.2 Explanation

Explanation is another purpose of visualization tools in ML pipelines, aimed at making complex models and processes understandable to diverse audiences. Tools designed with explanation purposes often include features that generate hints or notes to provide contextual insights about specific elements within an ML pipeline. Such annotations can offer explanations of the functions used regarding their purpose, parameters, and impact on the outputs, which facilitates both learning ML development and understanding a developed pipeline. Additionally, some tools might highlight areas of the pipeline or lines of code where issues such as bias or errors may have a higher likelihood of arising. For example, if certain preprocessing steps or model training functions are susceptible to introducing data imbalance or skew, the tool might flag these steps, prompting users to review and potentially adjust them.

There are several examples of explanation goals in visualization tools. [11] provides explanations and notifications about possible data distribution bugs in the preprocessing pipeline, highlighting potential issues. Similarly, [36] focuses on pipeline functions that could contribute to data biases with the main objective of improving fairness. [31] provides a plugin for TensorBoard for explanations of the functions of ML algorithms.

4.2.3 Visual Development

Visual development, also referred to as no-code or visual ML, is a purpose of visualization tools that enables users to build, modify, and deploy ML pipelines without requiring extensive programming knowledge. These tools provide an intuitive, drag-and-drop interface where users can assemble pipeline components visually, connecting blocks or modules that represent different pipeline stages, such as data preprocessing, model training, and evaluation. Visual ML tools come with pre-configured blocks containing widely used methods and functions for common ML tasks. Visual development also promotes collaboration, as it enables team members across disciplines to participate in model design and understand the overall workflow structure.

Visual development tools provide similar functionalities with predefined blocks that can be used to build an ML pipeline and differ in the levels of granularity and supported methods. [6] focuses on

Table 1: Tools for Machine Learning Pipeline Visualization

	Integration					Purpose					
Author	Web interface	Full system	Python library	Plugin	Concept	Exploration	Explanation	Visual development	Comparison	Monitoring	Domain specific
clearml	+		+			+			+	+	
Du, et al., 2023	+	+						+			
evidently			+			+			+	+	
Françoise, et al., 2021	+							+	+	+	
Grafberger, et al., 2022			+			+	+				
Guo, et al., 2016	+	+						+		+	
Harenslak, et al., 2021	+	+				+				+	
Khan, et al., 2021					+	+				+	
knime		+	+					+			
Liu, et al., 2022					+	+	+				
Ming et al., 2018			+			+					+
Ono, et al., 2020			+			+			+		
Qu, et al., 2019		+				+				+	+
rapidminer			+					+			
Sacha, et al., 2017					+	+		+	+		
Shah, et al., 2019	+		+			+				+	
Spinner, et al., 2019				+		+	+				
van den Elzen, et al., 2023					+	+	+				
visualdl	+		+			+			+	+	
Wongsuphasawat, et al., 2017		+	+			+				+	
Yang, et al., 2020	+		+			+	+			+	
Zhang, et al., 2023	+	+						+			+

collaboration processes and quick and easy application and experimentation. [13] provides an extensive library that can be extended manually and a minimalistic interface.

4.2.4 Comparison

Comparison is a key purpose of some visualization tools, designed to facilitate the side-by-side evaluation of various models, configurations, and pipeline versions within an ML workflow. Comparison-focused tools allow users to visually assess different pipeline elements, such as model performance metrics, data preprocessing techniques, or feature engineering methods, across multiple experiments. For example, users can quickly compare stages and methods contained in each pipeline and the outcome of each model represented by accuracy, precision, recall, or other commonly applied model performance metrics. Such visualizations provide a quick comparison of different model configurations and promote an overview of developed pipelines.

Most of the tools focus on comparing the performance metrics of the pipelines [5, 7, 31]. On the other hand, [23] explores and compares combinations of preprocessing steps, feature transformations, and ML models.

4.2.5 Monitoring

Monitoring is another essential purpose of visualization tools in machine learning pipelines, focused on providing real-time insights into the performance and health of models once deployed. Monitoring tools track various metrics, such as accuracy, precision, data drift, latency, and resource usage, ensuring that models continue to perform consistently within the predefined thresholds. This is particularly important in production environments, where model performance can degrade over time due to changing data patterns or unexpected shifts in input data distributions. These tools often include dashboards or time-series graphs that illustrate the trends of key metrics over time, allowing for quick diagnostics if issues arise. Some advanced monitoring tools also support anomaly detection, flagging unusual patterns that could indicate a need for retraining or pipeline adjustments.

[31] explores the model in runtime and during training providing capabilities for global provenance. [7] examines different performance metrics and can provide a report on pipeline performance.

4.2.6 Domain-Specific Tools

We distinguish domain-specific tools as a separate purpose, even though these tools often align with one or more of the previously described purposes, such as exploration, monitoring, or visual development. Domain-specific tools can represent the whole ML pipeline, but they aim to address the specialized needs of particular fields, dealing with unique data types, workflows, and challenges that general-purpose tools may not fully capture. For example, in genomics, visualization tools focus on representing complex genomic data [26]. They integrate with specialized genomic databases and provide visual interfaces designed to handle the massive datasets common in genomic research. In another instance, a visualization tool provides capabilities for no-code visual development of ML pipelines for cyber attack prevention in edge devices [40]. Domain-specific tools provide targeted solutions for specific ML application fields, however, they are not limited to the examples provided.

5 RQ2: Visualization Methods

When visualizing an entire ML pipeline, several techniques can be employed to represent the flow and relationships or improve understanding of the pipeline. We identified two main techniques to visualize a pipeline including a directed acyclic graph and its extensions and a pipeline matrix. Additionally, different types of labels can be implemented for explanatory purposes.

5.1 Directed Acyclic Graph

Directed Acyclic Graph (DAG) is the most common approach to visualizing a sequence of elements in an ML pipeline [5, 13, 14, 19, 30, 40]. These diagrams use blocks to represent individual stages connected by arrows that indicate the flow of data. The method visually maps the entire process, showing a clear, top-down, or left-to-right progression through each pipeline stage. A DAG of a simplified ML model is shown in Figure 4.

DAGs are especially valuable for ML pipelines because they visually represent the logical flow from data ingestion to preprocessing, feature engineering, model training, and evaluation, making it easy to trace the sequence of operations. Each node in the DAG represents a distinct stage in the pipeline, such as loading data, transforming and cleaning data, or training a model, while edges show the dependencies between these stages. This highlights which stages rely on previous outputs and helps to understand and manage complex dependencies.

DAGs can vary in complexity and can be adapted to represent various pipeline architectures including simple linear flows and more complex branched structures with different models or transformations run in parallel, thus, making it suitable for both visualization of already developed pipelines and no-code development of new pipelines.

Some DAGs enhance their interpretability by incorporating color coding to convey additional information, adding a new dimension to the visualization. Colors can be used to represent various attributes, such as the type of operation, e.g., data transformation, model training, evaluation; the status of each stage, e.g., completed, in progress, failed; or performance metrics like accuracy or processing time.

5.1.1 DAG with Annotations

In some cases, adding annotations to DAGs enhances their interpretability by including detailed information directly on nodes or

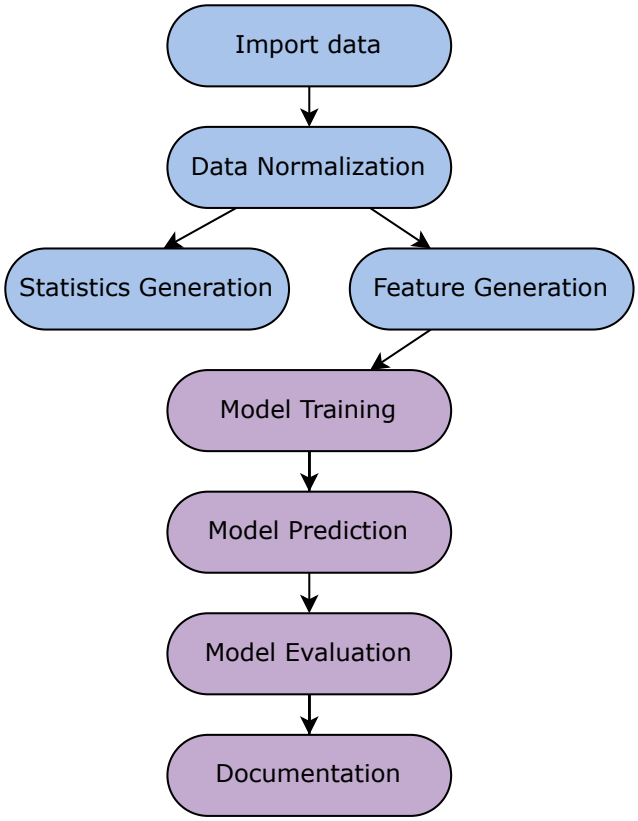

Figure 4: Directed acyclic graph (DAG) representation of a machine learning pipeline. Each node represents a specific stage. The nodes are connected by directed edges indicating the order of execution.

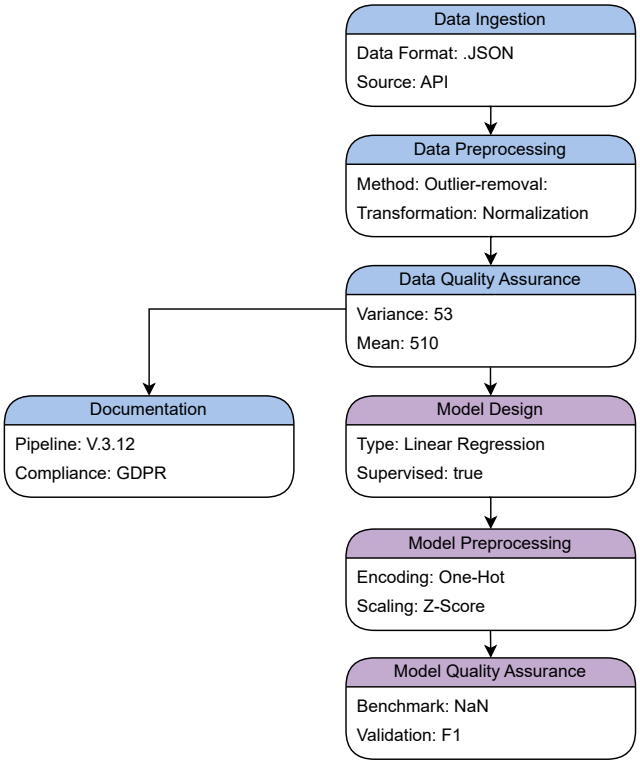

Figure 5: Directed acyclic graph (DAG) with added annotations. The annotations include additional information within each stage of the pipeline.

edges. Annotations can describe the type of transformation or function being performed, highlight key parameters, or provide metric summaries (e.g., accuracy, data quality) [6, 24, 35]. This method enriches the visualization and provides more context. Figure 5 represents an exemplary extension of some steps within an ML pipeline.

Annotations can appear in various forms, such as text labels within or outside a node, or they can be displayed when hovering over specific elements. For instance, a node representing a data preprocessing step might include information about data distribution, mean, and variance values. Likewise, nodes representing the model training stage may provide information about accuracy, loss, or training time.

5.1.2 DAG with Collapsing Elements

To manage the complexity of larger pipelines, DAGs with collapsing elements allow users to expand or collapse sections of the graph [24, 31]. Figure 6 shows an example of a collapsed data preprocessing element.

In a typical ML pipeline, certain processes like data preprocessing or model training can involve multiple steps and sub-tasks. Data preprocessing might include missing values handling, normalization, scaling, and other functions, each of which could be represented as a separate node. However, in larger pipelines with extensive data preprocessing stages, representing each function as an individual node would increase the complexity of the visualization and decrease the initial understanding of the core processes. Thus, these nodes can be grouped into a single collapsible node. Thus, users are provided with a high-level overview with an option to expand this node if they want to inspect the detailed steps involved. This is particularly beneficial when different stakeholders interact with the pipeline and have varying needs for information.

5.2 Pipeline Matrix

The pipeline matrix is a grid-like visualization method where stages of the pipeline are organized in rows and columns [23]. The rows represent different versions of an ML pipeline and might correspond to a specific configuration, experiment, or set of hyperparameters. The columns represent stages or distinct functions of the ML pipeline, depending on the degree of abstraction of such visualization. Moreover, each column may contain specific details about the techniques or parameters applied at that stage.

The matrix design provides several key benefits. The primary advantage of a pipeline matrix is a facilitated comparison across multiple pipelines. Users can easily compare the influence of different preprocessing methods (e.g., missing value handling or normalization techniques) and model types on the performance metrics of the model.

Moreover, the pipeline matrix is highly scalable with the addition of new experiments and configurations. New columns and rows can be added at any time without disrupting the overall structure of the visualization. An example of a pipeline matrix is shown in Figure 7.

5.3 Side Labels with Descriptive Information

Adding side labels or comments to visualizations is an effective method for including context or descriptive information about certain functions or stages. Such labels can be added for two purposes: they either provide additional explanatory information [5, 27, 31, 35] or highlight potential risks or issues [11, 36]. An exemplary setting is shown in Figure 8.

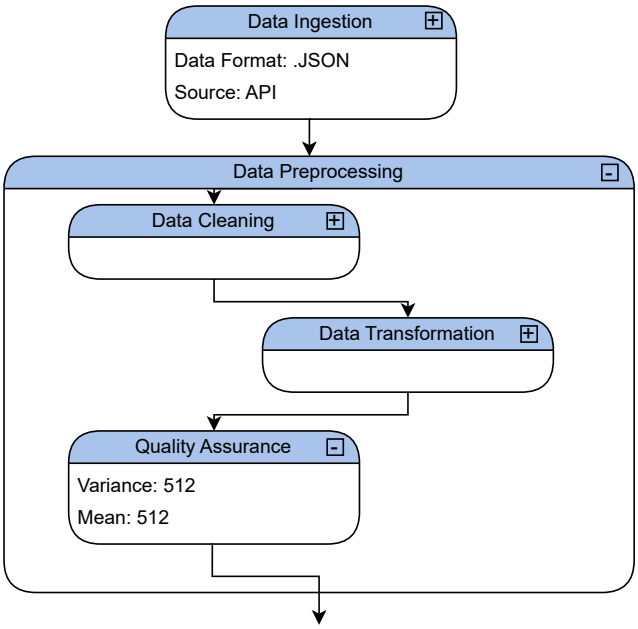

Figure 6: Directed acyclic graph (DAG) with collapsible elements. The functionality allows users to expand or collapse specific stages of the ML pipeline for a more abstract or detailed view.

Blocks with additional information can enrich the ML pipeline. Such descriptions can provide explanatory information about the code blocks or particular functions and include links to external resources such as documentation, related datasets, or code snippets.

Annotations added for notification purposes might indicate where bias is likely to occur, which is especially relevant for training dataset formation and feature balance. Furthermore, annotations or flags can be added to the functions and steps with a high risk of introducing errors, e.g., data transformation steps including merging and joining datasets.

The labels can appear in the sidebar of the coding environment with explanatory information about certain functions or can be implemented within the pipeline visualization highlighting areas requiring additional attention.

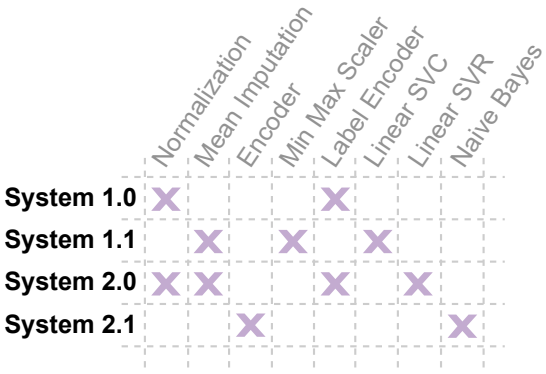

Figure 7: Pipeline matrix visualization of machine learning pipelines. The rows represent different ML pipelines and their versions. The columns correspond to individual steps or functions within each stage.

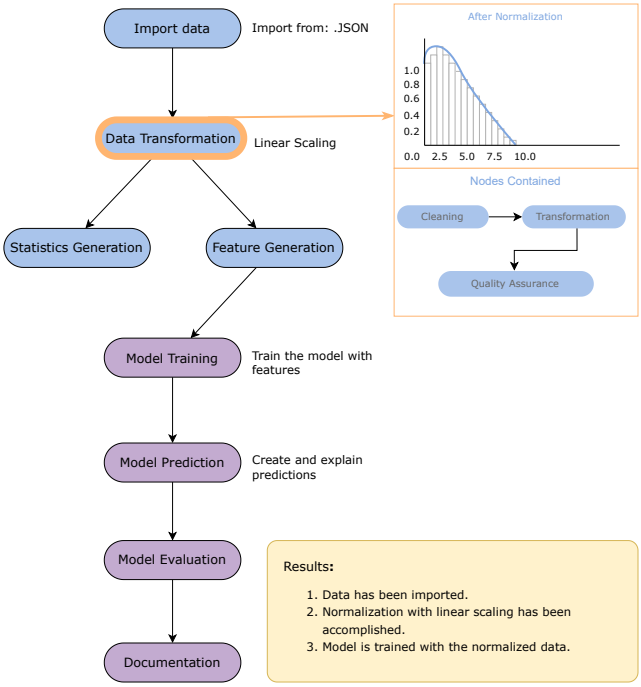

Figure 8: Machine learning pipeline with side labels and explanatory annotations. The visualization provides additional information on data distribution and preprocessing and explanatory comments on each ML stage.

6 Challenges and Future Research Directions

In this section, we discuss the limitations and challenges of current tools and future research directions, answering research questions 3 and 4.

6.1 RQ3: Limitations and Challenges

There were several papers which reported *integration challenges* [9,11,31]. The compatibility of visualization tools is essential, meaning that some tools may only be suitable for specific contexts, such as certain types of neural network architectures (e.g., LSTM / RNN / CNN) or specialized tasks like genome detection. With this, the structure and complexity of machine learning pipelines can vary significantly. Not only the structure plays a major role, but also how the visualization tools can be integrated into existing libraries, settings, and use cases. The provided interfaces (e.g. AutoML, tensorflow) also pose integration challenges. Furthermore combining various visualization methods for different stages of the machine learning pipeline adds another layer of complexity and further compatibility concerns [17].

Machine learning algorithms are primarily designed for computational efficiency and accuracy rather than visualization. This inherent focus can lead to challenges in visualizing data flows efficiently [20–22]. Online analysis, where visualization tools need to keep up with rapidly changing data, can pose significant *efficiency challenges*. Additionally, there may be overhead from repeatedly executing visualization tasks [12], especially when dealing with large datasets or deep models like recurrent neural networks that involve complex graph structures [22].

Visual *scalability* is essential for ensuring clear visualization when handling extensive datasets and complex models. However, as data size and complexity grow, maintaining clarity becomes increasingly difficult [21,23,28]. A simple example is the limitations that may arise with color palettes, as they may not effectively adapt to

large datasets [25]. Additionally, issues with granularity can occur. If the visualization's granularity remains constant regardless of the data size it can lead to difficulties in interpretation when analyzing significant amounts of data.

Similar to visual scalability, also the overall *usability* is crucial for ensuring that visualization tools are accessible and beneficial to a wide range of users. Sources report on the challenge of simplifying the learning process for individuals with varying levels of expertise in machine learning and data analysis. These tools should be designed to meet the needs of both experts and non-experts. However, a limitation of current tools is that they typically offer only one view, which is either designed for experts with extensive configurability or non-experts at a very high level [17, 31]. Additionally, incorporating features like versioning and collaboration tools can improve usability by allowing users to track changes and work together on visualization tasks. However, some of the existing tools focused either on management tasks, such as version control, and the others focused only on visualization without the management capabilities [31].

6.2 RQ4: Future Research Directions

The topics of integration and compatibility are not only challenges but are also relevant for future research. Efforts will be made to increase compatibility by addressing different pipeline schemes [9, 35], such as diverse machine learning algorithms and data types. This could involve exploring methods to generate visualizations directly from existing, unmodified pipeline code and enhancing interoperability with common ML frameworks [11].

Further enhancements in the ease of use could be achieved by addressing visual scalability issues and developing real-time visualization capabilities, particularly for lengthy training processes [16, 20, 21]. Additionally, two-way visualizations, that allow for interactive feedback between users and the visualization tool were discussed [11]. Furthermore, designing views tailored for both experts and non-experts, while simplifying integration and reducing the user learning curve, are areas of focus [17, 21].

Research in the field of machine learning is increasingly shifting towards the development of visual techniques for representing trust in machine learning models, known as trustworthy AI [33]. An illustrative example of this trend is the growing demand for GDPR justifications for decisions made by machine learning models. However, achieving this goal poses significant challenges. Therefore, there is a need to explore explainable AI methods that can provide justifications for model decisions in compliance with such regulations through explanations, which constitutes an important future research direction [31].

These XAI methods can also be used to provide users with suggestions and support in the application and analysis of machine learning models. In addition, anomaly detection techniques are being explored within visualization tools to detect irregular patterns in the data flow and guide the user as intelligent analytics. Finally, an interesting direction in the use of AI is the application of natural language queries and large language models for generating and interacting with visualizations, as for instance Flowsense [39] is doing.

7 Conclusion

Due to the growing complexity of ML pipelines and their wide deployment, ML pipeline visualization tools become a crucial part of ML development. These tools play a critical role in assisting with different tasks ranging from model development to monitoring and explanation. By categorizing these tools into six main purposes such as exploration, explanation, visual development, comparison, monitoring, and domain-specific applications, we have outlined the essential functions they serve while visualizing ML processes.

Furthermore, the paper discusses various integration methods, such as standalone visual systems and library-based tools, each offering different levels of flexibility and accessibility. Additionally, the main visualization techniques used in these tools are described and presented. They include Directed Acyclic Graphs (DAGs), pipeline matrices, and annotated visualizations.

Despite the impressive advancements in ML pipeline visualization, challenges remain, particularly in terms of tool integration flexibility and scalability. As machine learning continues to impact various industries, these tools will play an increasingly important role in AI advancement and make ML development accessible to a larger population.

Acknowledgments

The authors wish to thank Stefan Domanegg and Ali Eren Karaca.

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association is gratefully acknowledged.

References

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291–300. IEEE, 2019.

[2] E. Caveness, P. S. GC, Z. Peng, N. Polyzotis, S. Roy, and M. Zinkevich. Tensorflow data validation: Data analysis and validation in continuous ml pipelines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 2793–2796, 2020.

[3] A. Chatzimparmpas, R. M. Martins, I. Jusufi, and A. Kerren. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization*, 19(3):207–233, 2020. doi: 10.1177/1473871620904671

[4] J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE computer graphics and applications*, 38(4):84–92, 2018.

[5] ClearML. Clearml - your entire mlops stack in one open-source tool, 2024. http://github.com/allegroai/clearml.

[6] R. Du, N. Li, J. Jin, M. Carney, X. Yuan, K. Wright, M. Sherwood, J. Mayes, L. Chen, J. Jiang, et al. Experiencing visual blocks for ml: Visual prototyping of ai pipelines. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–3, 2023.

[7] Evidently AI. Evidently, 2020. https://github.com/evidentlyai/evidently.

[8] Facets. facets - visualizations for machine learning datasets, 2019. https://github.com/PAIR-code/facets.

[9] J. Françoise, B. Caramiaux, and T. Sanchez. Marcelle: composing interactive machine learning workflows and interfaces. In *The 34th Annual ACM symposium on user interface software and technology*, pp. 39–53, 2021.

[10] V. Garousi, M. Felderer, and M. V. Mäntylä. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, 106:101–121, 2019.

[11] S. Grafberger, P. Groth, J. Stoyanovich, and S. Schelter. Data distribution debugging in machine learning pipelines. *The VLDB Journal*, 31(5):1103–1126, 2022.

[12] S. Grafberger, S. Guha, P. Groth, and S. Schelter. mlwhatif: What if you could stop re-implementing your machine learning pipeline analyses over and over? *Proceedings of the VLDB Endowment*, 16(12):4002–4005, 2023.

[13] T. Guo, J. Xu, X. Yan, J. Hou, P. Li, Z. Li, J. Guo, and X. Cheng. Ease the process of machine learning with dataflow. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 2437–2440, 2016.

[14] B. P. Harenslak and J. de Ruiter. *Data pipelines with apache airflow*. Simon and Schuster, 2021.

[15] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.

[16] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. A cti v is: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.

[17] T. Khan, S. S. Shakeel, A. Gul, H. Masud, and A. Ebert. Implementing visual analytics pipelines with simulation data. In *Software Usability*. IntechOpen, 2021.

[18] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. In *EBSE Technical Report*, 2007.

[19] KNIME. Knime, 2024. https://github.com/knime.

[20] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the training processes of deep generative models. *IEEE transactions on visualization and computer graphics*, 24(1):77–87, 2017.

[21] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017. doi: 10.1109/TVCG.2016.2598831

[22] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 13–24, 2017. doi: 10.1109/VAST.2017.8585721

[23] J. P. Ono, S. Castelo, R. Lopez, E. Bertini, J. Freire, and C. Silva. Pipelineprofiler: A visual analytics tool for the exploration of automl pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):390–400, 2020.

[24] PaddlePaddle. Visualdl, 2020. https://github.com/evidentlyai/evidently.

[25] N. Pezzotti, T. H¨ollt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):98–108, 2017.

[26] Z. Qu, Y. Zhou, Q. V. Nguyen, and D. R. Catchpoole. Using visualization to illustrate machine learning models for genomic data. In *Proceedings of the Australasian Computer Science Week Multiconference*, pp. 1–8, 2019.

[27] RapidMiner Studio. Rapidminer, 2020. https://github.com/rapidminer/rapidminer-studio.

[28] D. Sacha, M. Sedlmair, L. Zhang, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. What you see is what you can change: Human-centered machine learning by interactive visualization. *Neurocomputing*, 268:164–175, 2017.

[29] C. Seifert, A. Aamir, A. Balagopalan, D. Jain, A. Sharma, S. Grottel, and S. Gumhold. Visualizations of deep neural networks in computer vision: A survey. *Transparent data mining for big and small data*, pp. 123–144, 2017.

[30] S. Shah, R. Fernandez, and S. M. Drucker. A system for real-time interactive analysis of deep learning training. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2019, Valencia, Spain, June 18-21, 2019*, pp. 16:1–16:6, 2019. doi: 10.1145/3319499.3328231

[31] T. Spinner, U. Schlegel, H. Sch¨afer, and M. El-Assady. explainer: A visual analytics framework for interactive and explainable machine learning. *IEEE transactions on visualization and computer graphics*, 26(1):1064–1074, 2019.

[32] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676, 2017.

[33] S. van den Elzen, G. Andrienko, N. Andrienko, B. D. Fisher, R. M. Martins, J. Peltonen, A. C. Telea, and M. Verleysen. The flow of trust: A visualization framework to externalize, explore, and explain trust in ml applications. *IEEE computer graphics and applications*, 43(2):78–88, 2023.

[34] J. Wang, S. Liu, and W. Zhang. Visual analytics for machine learning: A data perspective survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[35] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2017.

[36] K. Yang, B. Huang, J. Stoyanovich, and S. Schelter. Fairness-aware instrumentation of preprocessing˜ pipelines for machine learning. In *Workshop on Human-In-the-Loop Data Analytics (HILDA'20)*, 2020.

[37] W. Yang, C. Chen, J. Zhu, L. Li, P. Liu, and S. Liu. A survey of visual analytics research for improving training data quality. *Journal of Computer-Aided Design & Computer Graphics*, 35(11):1629–1642, 2023.

[38] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[39] B. Yu and C. T. Silva. Flowsense: A natural language interface for visual data exploration within a dataflow system. *IEEE transactions on visualization and computer graphics*, 26(1):1–11, 2019.

[40] Y. Zhang, B. Asulba, N. Schumacher, M. Sousa, P. Souto, L. Almeida, P. Santos, N. Martins, and J. Sousa. Implementing and deploying an ml pipeline for iot intrusion detection with node-red. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, pp. 247–253. 2023.