The Journal of Systems and Software 222 (2025) 112343

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

On the understandability of machine learning practices in deep learning and reinforcement learning based systems ☆

Evangelos Ntentos [a,*], Stephen John Warnett [a,b], Uwe Zdun [a]

[a] Faculty of Computer Science, Research Group Software Architecture, University of Vienna, Vienna, Austria
[b] UniVie Doctoral School Computer Science DoCS, Faculty of Computer Science, University of Vienna, Vienna, Austria

ARTICLE INFO

Keywords:
Machine learning
Modeling
Best practices
Controlled experiment
Empirical software engineering

ABSTRACT

Machine learning (ML) has emerged as a transformative subject, using various algorithms to help systems analyze data and make predictions. Deep Learning (DL) uses neural networks to address hard problems. Reinforcement Learning (RL) is a way to solve problems by making consecutive decisions.

Understanding ML systems based only on the source code is often a challenging task, especially for inexperienced developers. In a controlled experiment involving one hundred fifty-eight participants, we assessed the understandability of ML-based systems and workflows through source code inspection compared to semi-formal representations in models and metrics.

We hypothesize that ML system diagrams modeling details of ML workflows and practices like transfer learning and checkpoints can enhance the understandability of ML practices in system design comprehension tasks, assessed through task *correctness*. Additionally, providing these sources could lead to an increase in task *duration*, and we expect a significant correlation between *correctness* and *duration*.

Our findings show that providing semi-formal ML system diagrams with the source code improves the effectiveness of the *correctness* for the DL relevant tasks. The control group had an average correctness of 0.7121, while the experimental group had a higher average correctness of 0.7759. On the other hand, participants who received only the system source code showed slightly better performance in the *correctness* task (average *correctness* 0.6808) within the RL relevant tasks compared to those who also received the semi-formal diagrams (average *correctness* of 0.6612). However, no significant difference was found in the *duration* task between the two. The control group, for the DL relevant tasks, took an average of 1571.62 s, whereas the experimental group took an average of 1763.85 s. For the RL relevant tasks, the control group had an average of 1883.80 s, while the experimental group 1925.46 s. However, semi-formal ML system diagrams can benefit specific scenarios.

1. Introduction

In the rapidly evolving field of artificial intelligence, DL, a subset of ML, has become a highly effective tool for addressing complex problems. Use neural networks to solve complex problems. In the same way, RL has become important in solving sequential decision-making tasks and developing automation and intelligence (Winder, 2021). Transfer learning is gaining traction from both DL (LeCun et al., 2015; Pouyanfar et al., 2018) and RL (Zhu et al., 2023; Islam et al., 2023; Tan et al., 2018) which is an effective way to improve the learning process by transferring knowledge from one domain to another. Transfer learning aims to improve the model's adaptability and applicability in various tasks (Valliappa Lakshmanan, 2021).

In software engineering, ML design models serve as a collection of best practices for addressing common challenges and providing solutions. These models refine the collective expertise and insights of experienced professionals. It provides practitioners with widely applicable guidance.

Industry-scale systems often support a wide range of practices and implementations, making it challenging to evaluate whether a given system adheres to best practices and models. Today, in ML, the source code of the ML system is usually the only source to comprehend the ML workflow and the use of best practices and patterns in those workflows (Winder, 2021; Chen et al., 2023; Eisenman et al., 2020; Hosna et al., 2022). This can make it hard to comprehend the ML workflows and assess their conformance to best practices and patterns,

☆ Editor: Alexander Chatzigeorgiou.

* Corresponding author.
E-mail addresses: evangelos.ntentos@univie.ac.at (E. Ntentos), stephen.warnett@univie.ac.at (S.J. Warnett), uwe.zdun@univie.ac.at (U. Zdun).

https://doi.org/10.1016/j.jss.2025.112343
Received 27 March 2024; Received in revised form 18 October 2024; Accepted 9 January 2025
Available online 16 January 2025

as the source code of those workflows can be complex and hard to understand. Many techniques exist to realize the same ML workflows in code, and multiple technologies and library versions further complicate understanding. This is especially true for software developers who are novices in ML, and today, more and more software developers who are not formally trained in ML have to work on such systems. Conversely, data scientists with limited software engineering education often have to perform software engineering tasks in the context of ML workflows. Design and architecture models, commonly used in other software engineering fields to ease the comprehension of complex source code parts, remain virtually unused so far for modeling ML workflows.

This study aims to answer the following research question:

- **RQ** To what extent does the provision of semi-formal architecture models, in addition to system source code, improve the understandability of ML patterns and practices, particularly for software developers with limited ML experience?

This empirical study aims to address this question by evaluating participants' understanding of ML practices in the context of deep learning and reinforcement learning. To do so, we experimented with a total of 158 participants who are trained as software developers but have limited ML experience. We created four groups (see Section 4.3, and we call the tasks in the groups, where the participants get only a system source code repository, the *Control Tasks*, and the tasks in the four groups, where the participants get a system source code repository plus help in the form of models, the *Experiment Tasks*.

We hypothesize that measuring understandability through the dependent variables' *correctness* and *duration* would reveal a significantly better understanding of ML practices when models of a given system are provided. More specifically, *correctness* is used to measure the effectiveness, and *duration* is used to measure the efficiency of understanding ML practices.

Our results show that participants who were given a system source code repository, and semi-formal ML models demonstrated a significantly better understanding than those who only received the system source code repository. The provision of models showed an improvement in the effectiveness of understanding the ML practices without impacting the overall efficiency.

1.1. Structure of this paper

We provide an overview of ML practices and related studies in Section 2. Section 3 discusses related work. Next, we describe the planning of this study in Section 4. In Section 5, we detail the execution of the experiment, and the results are presented in Section 6. We discuss the results in Section 7. Section 8 we describe the threats to validity, and we conclude in Section 9.

2. Background

In this section, we provide an overview of the background for this study. We introduce two ML design patterns (Valliappa Lakshmanan, 2021) and the relevant architectural models. These patterns apply in many forms to both deep learning and reinforcement learning.

2.1. Design patterns

In machine learning, design patterns are useful for addressing the common issues faced when developing and training models. These patterns provide structure and organization to the tasks of data handling, model optimization, and effective workflow management. Checkpoints, for example, present a methodical approach of storing a model state at specific intervals in time, which helps to resume the training phase with minimal loss of data. Transfer learning capitalizes on the use of pre-learned models to apply to new tasks in order to improve the performance of those tasks, thus cutting down on the training time and costs involved. Adoption of such design patterns leads to better resource allocation, ease of model exploitation across various applications, and simplification of maintenance.

Checkpoints. During ML training, interruptions or failures can result in data loss and inefficient use of computing resources. It indicates the need for a mechanism to maintain and restore the ideal state. Winder (2021) which guarantees training recovery to the precise point. Strengthening ML Workflows in the Long-Term Additionally, it is important to be able to specialize the model beyond initial training.

This problem involves the complex, time-consuming process of training ML models and the possibility of training interruption or failure. Restarting training from scratch in such cases can be inefficient and result in the loss of valuable data. Dedicated computation and data loss affect efficiency, especially when there is a need to restart training. Storage and computation need to increase due to the high demands imposed by ML processes (Chen et al., 2023). Adding features, such as saving ML model states, increases the complexity of the code in ML pipelines, and latency becomes a consideration, with additional pipeline steps impacting overall training time.

One solution to address these challenges is to use checkpoints (Eisenman et al., 2020; Winder, 2021). This practice allows storing the complete state of the ML model regularly during training. This facilitates training recovery from a specific point and ensures the reliability of your ML workflow.

The solution details involve key pattern participants and concepts, including the creation of a checkpoint repository to store and organize checkpoints, defining when and how checkpoints are created, managing checkpoints with assigned names or identifiers and metadata, and utilizing checkpoints for purposes such as resuming training, model persistence, and fine-tuning.

However, this pattern also provides certain drawbacks. Storage and computation overheads are incurred, especially with big fashions or extended schooling processes. Managing many checkpoints may additionally turn out to be hard, and there may be an ability advent of computational overhead, despite the fact that that is commonly minimal compared to the benefits received. Implementing and dealing with checkpoints can also introduce complexity into the ML pipeline, requiring extra code and considerations regarding how and when to create checkpoints. Additionally, developing and saving checkpoints at regular durations introduce latency into the training procedure, which may additionally impact actual time or time-touchy packages.

Variations include checkpoints. Agrawal et al. (2023) in deep learning where checkpoints are generated at the end of each training epoch for training the deep neural network and checkpoints in reinforcement learning where algorithm checkpoints determine the state of the algorithm used to ensure the continuity of the reinforcement learning process.

Transfer learning. The need for a mechanism to be able to preserve and recover the model state to enable the recovery of training from specific points is emphasized by the demand for specialized models for particular cases, such as adapting resource-intensive models to efficiently meet specific objectives (Winder, 2021; Hosna et al., 2022).

Standard solutions include the use of transfer learning (Winder, 2021; Hosna et al., 2022; Farahani et al., 2021). This involves loading a saved model from a previously trained model repository. The new model is optimized to address the same problem.

There are many forces influencing this problem and the resulting solutions. Transfer learning emphasizes the efficiency of information. It leverages knowledge from pre-trained models to source functions with sufficient data (Winder, 2021; Hosna et al., 2022; Raffel et al., 2023). This information shows known properties. This reduces reliance on large amounts of job-specific data. Resource efficiency is completed by using reusing the trained model load. This ends in faster model convergence and reduces hardware requirements. Transfer learning also supports domain optimization by adapting to changes in the data distribution. Effective use of relevant features and knowledge.

Transfer learning offers an efficient way to leverage pre-trained models on large labeled datasets. This approach eliminates the need for

extensive custom datasets. It allows small organizations and specialized sectors to develop custom models tailored to specific tasks. They can benefit from collective knowledge using the wealth of research and insights gained from others in the field. Additionally, advances in state-of-the-art transfer learning are powered by images and suites that have significantly increased the capabilities of this technique. Also, this technique enables model similarity and improves performance on specific tasks (Winder, 2021; Hosna et al., 2022; Farahani et al., 2021; Raffel et al., 2023).

Pattern variants consist of transfer learning in deep learning, which includes leveraging a pre-trained version, freezing its weights, and integrating non-trainable layers into a new model. In the context of reinforcement learning, transfer learning empowers agents to improve their overall performance on related obligations via leveraging understanding from previous experiences, overcoming challenges related to data scarcity, and time-consuming training in domains such as robotics, gaming, natural language processing, and autonomous systems.

2.2. ML scripts and pipeline models

The behavior or workflow of the ML system is usually given either as an ML script or a pipeline. Both can be modeled as an extended (i.e., more detailed) activity diagram. An illustrative script/pipeline diagram is presented in Fig. 1.

ML scripts are basically pieces of code written to carry out machine learning tasks step by step. They manage different parts of the process, like bringing in data, cleaning it, training a model, and testing how well the model works. These scripts make sure that all the tasks are done in the right order and help automate things that would otherwise take a lot of time.

In more advanced systems, like reinforcement learning, ML scripts also manage things like setting up the learning environment, training agents, and saving the model during training so it can be used later.

An ML pipeline can be represented using formal model diagrams. We use and extend a formal modeling method based on our prior work (Zdun et al., 2017). All script/pipeline diagrams adhere to a sequential pattern, guiding the flow from an *initial node* to a designated *final node*. These diagrams comprise *pipeline nodes* that symbolize distinct stages within the pipeline. Notable *pipeline node types* encompass steps like *data extraction and analysis*, *data preparation*, and *model training* in deep learning, and *RL environment setup*, *RL model saving*, and *RL model training* in reinforcement learning.

Pipeline nodes can have specific properties. Each pipeline node, at a minimum, features an *automatic* property, which can assume a value of *True* or *False* to signify whether the step executes automatically. Most steps in a script/pipeline are usually automated. Other *properties* includes *runs in* (indicating the component(s) in which the step operates), *output to* (identifying the component(s) receiving the step's output), *invokes* (identifying the component(s) that are called by the step, such as a task), and *environments* (specifying the execution environment(s) in which the step operates). Please note that the execution environments are the cloud or local execution environments in which the components run and not the reinforcement learning environments.

Complex pipelines may involve multiple branching paths, denoted as *fork nodes*, with these distinct paths converging at a *join node*. For instance, if there are multiple triggers for a pipeline run, these triggers branch out from a *fork node*, with their paths later merging at a *join node*. Additionally, there are *decision nodes*, enabling a choice of paths, usually annotations with a question to be decided. Outgoing connections in a process diagram can have labels like *yes* or *no*" For example, if a human needs to decide whether to approve releasing a new model version into production, this decision can be shown with such labels. If a connection is labeled *asynchronous call*, it means that the process can move forward without waiting for the result from another step. The main step continues running while the other one happens at the same time.

3. Related work

This section provides an overview of the existing literature on ML best practices and studies that employ comparable methodologies to our research.

Hosna et al. (2022) contribute to the field of transfer learning by presenting a paper that discusses the domain and scope of transfer learning, considering its situational use based on different periods and applications. The paper delves deeply into techniques like Inductive Transfer Learning, Transductive Transfer Learning, and Unsupervised Transfer Learning, covering aspects such as sample selection and domain adaptation. Agrawal et al. (2023) make a significant observation regarding the sensitivity of model weights to compression during training. They propose a non-uniform quantization scheme, an efficient search mechanism to adjust quantization configurations dynamically, and a quantization-aware delta compression mechanism, all instantiated in DynaQuant - a framework for deep learning workload checkpoint compression. Chen et al. (2023) introduce self-ensemble protection (SEP), a model that leverages checkpoints' gradients. SEP is effective due to its ability to learn from examples ignored during normal training and the orthogonal nature of checkpoints' cross-model gradients, making them diverse without training multiple models. Chen et al. (2020) propose LC-Checkpoint, a lossy compression scheme for checkpoint constructions, optimizing both compression rate and recovery speed. LC-Checkpoint uses quantization and priority promotion to store crucial information for SGD recovery, employing Huffman coding to leverage the non-uniform distribution of gradient scales. Eisenman et al. (2020) introduce Check-N-Run, a scalable checkpointing system designed for training large ML models at Facebook. The system addresses size and bandwidth challenges through differential checkpointing, which tracks and checkpoints the modified part of the model and leverages quantization techniques to reduce checkpoint size without compromising training accuracy.

Warnett and Zdun (2024) carried out a controlled experiment with sixty-three participants to evaluate the understandability of MLOps system architecture descriptions. They compared informal textual and graphical representations with UML-based diagrams. Their results show that task correctness significantly improves when UML-based diagrams are used. Additionally, they noted that incorporating UML-based diagrams does not significantly extend task duration and thus does not hinder understanding. They also found that a significant positive correlation between task correctness and duration, but only when semi-formal diagrams are utilized. Similar to our research, they recruited university students, provided UML-based diagrams as assistance, and statistically tested their hypotheses. However, their study used a between-subject design and focused on MLOps system architecture features rather than ML practices.

Allodi et al. (2020) conducted a study involving seventy-three participants to evaluate the accuracy of security professionals and students with advanced technical education in assessing the severity of software vulnerabilities based on various attributes. In contrast to our focus on comparing different system description methods, they emphasized participants' background knowledge and education. A significant methodological difference is categorizing participants into three groups: students with a BSc in information security enrolled in an MSc in information security degree program, students in an MSc in computer science program, and security practitioners. Moreover, they specifically recruited students with no professional expertise, distinguishing their approach from ours.

Heijstek et al. (2011) conducted a controlled experiment that resembles our investigation. Their primary objective was to assess the efficacy of visual versus textual artifacts in conveying software design decisions to software developers. The study enlisted forty-seven participants from both industry and academia who evaluated UML representations as diagrammatic artifacts and informal textual descriptions. However, distinctions exist between their research and ours. Firstly, all participants

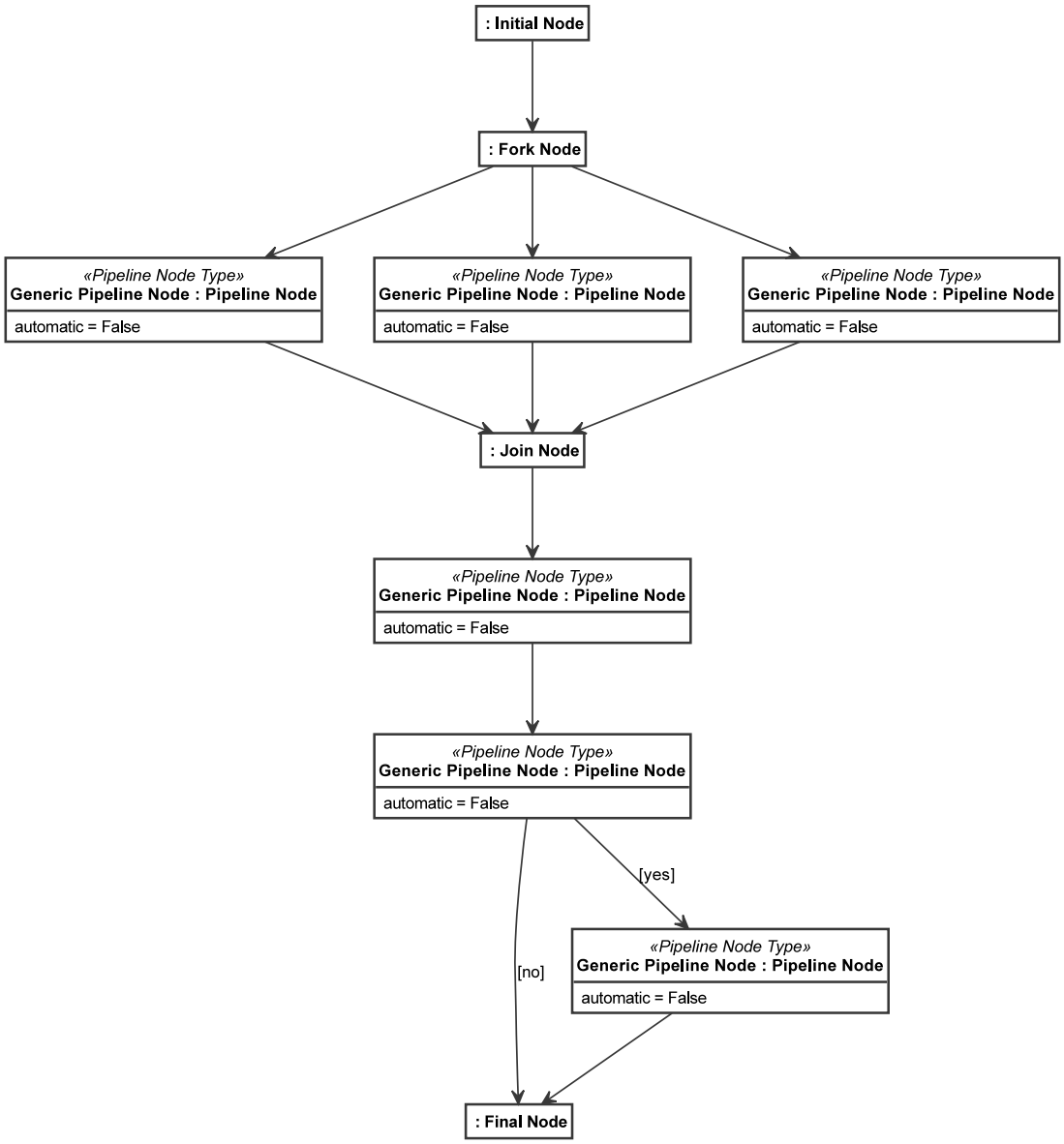

Fig. 1. Generic pipeline diagram illustrating the elements and relations.

in their study assessed both representations, whereas there may be variations in the evaluation process in our research. Secondly, our study specifically delves into ML practices in the context of deep learning and reinforcement learning, setting it apart from the broader scope of Heijstek et al.'s inquiry. These methodological variations underscore the unique aspects of our study, making a valuable contribution to the broader research landscape in this field.

Labunets et al. (2014) conducted a controlled experiment with twenty-nine MSc students to investigate participants' perceptions of visual and textual methods for security risk assessment in terms of effectiveness. Although their study shares similarities with ours in comparing visual and textual representations, notable differences exist. Firstly, the research does not specifically examine the distinctions between formal and informal representations, nor does it revolve around system understanding, which is a key focus of our study. Additionally, their experiment does not investigate ML practices, which is our research's primary area of interest. This underscores the specific scope and objectives of our study.

Allodi et al. (2017) executed a controlled experiment with twenty-nine students to explore the challenges participants face in assessing system vulnerabilities when security requirements change. Unlike our study, they employed a within-subject design, concentrating on variations in system requirements rather than modeling differences. However, similar to our approach, they formulated hypotheses and subjected them to testing using statistical methods.

Our work's main contribution is its study on how using ML system diagrams modeling workflows, including transfer learning and checkpoints alongside source code can help people better understand machine learning systems, especially in deep learning and reinforcement learning. While those related works have looked at best practices in ML, such as transfer learning and checkpoints, this study focuses on how these diagrams can help beginners who have limited experience with ML. By testing how well participants performed tasks and how long it took them, the research shows that these diagrams improve understanding. This work fills a gap in the field by offering practical advice on when and how to use visual aids to make complex ML systems easier to understand, which has not been deeply studied before.

4. Experiment planning

Our research aligns with the empirical research principles outlined in the field of software engineering, as proposed by Jedlitschka et al.

(2008). Furthermore, the study design integrates empirical research guidelines in software engineering from sources including Kitchenham et al. (2002), Wohlin et al. (2012), and Juristo and Moreno (2001). In terms of statistical analysis applied to the gathered data, the study employs the robust statistical methods recommended in empirical software engineering by Kitchenham et al. (2016).

4.1. Goals

In this experiment, the goal is to perform a code and model review in a limited time session of 1.5 h regarding the design of two systems (one DL and one RL) and recommended model training patterns and practices used in them. The goal was to assess understanding through code and model reviews, with a specific emphasis on DL and RL scripts. The focus was on the review of general design aspects and specific recommended model training patterns and practices. In particular, the checkpoints and transfer learning practices described in Section 2 were studied. For both systems, the participants reviewed a system source code. To study the extent to which participants understand a given system, each participant received help in the form of a model of the ML system to be studied in one of the two tasks. In the tasks where this help is present, we advised participants to consult the models first and then study the code if more details are required.

4.2. System description

The selection is based on their domain. The focus is on reinforcement learning including game environments and on medical image classification. Both systems utilize advanced techniques, such as Proximal Policy Optimization (PPO) and transfer learning, demonstrating different machine learning methodologies. Furthermore, the source code availability ensures transparency and reproducibility as well as allows researchers to efficiently examine and compare the results. Moreover, both systems have detailed documentation facilitate understanding and troubleshooting.

The *Reinforcement Learning System using Checkpoints* system[1] (Fig. 2) created in 2023, is a Python-based reinforcement learning framework that trains agents in the Knights-Archers-Zombies environment using Stable-Baselines3 and SuperSuit for multi-agent training and preprocessing. It uses the PPO (Proximal Policy Optimization) algorithm to train agents, with CNN and MLP policies depending on whether the observations are visual or vector-based. The system applies multiple SuperSuit wrappers, including black_death_v3 to handle agent death and ensure a constant number of agents. It trains models for around 81,920 timesteps and evaluates them over 100 games. With around 137 lines of code, it effectively handles agent training and evaluation, using vectorized environments and saving trained models. The system is flexible, allowing for easy adjustments to different agent setups, and supports multi-agent scenarios through the PettingZoo library. We modified the system to use checkpoints.[2]

The *Deep Learning System using Transfer Learning* system[3] (Fig. 3) created in 2020, is a Python-based machine learning pipeline using TensorFlow, Keras, and TensorFlow Hub for transfer learning in both image and text classification tasks. It first trains a model on the colorectal_histology dataset (5000 images, 8 classes) using a pre-trained VGG19 model from TensorFlow Hub with additional layers for classification. The system preprocesses the data, creates batches, and fine-tunes the model for 15 epochs, optimizing using Adam and categorical cross-entropy. Additionally, it trains a sentiment analysis model using the IMDB reviews dataset, leveraging pre-trained gnews-swivel-20dim text embeddings, and adds dense layers for binary classification. The total system includes around 168 lines of code for data loading, model construction, training, and evaluation. It is optimized for efficient transfer learning by freezing pre-trained layers and focusing on the newly added layers.

4.3. Context and design

In this study, we chose to employ a Within-Subjects Design based on the insights provided by Charness et al. (2012) on the advantages of this experimental approach in certain contexts. In a within-subject design, each participant goes through all conditions, which increases statistical accuracy since each person acts as their own control. This method is particularly useful in our study, where we aim to identify small differences in task performance when participants have additional support versus when they do not. Furthermore, within-subject designs tend to align well with theoretical models in which a single individual is exposed to variations in their environment, which matches our interest in observing how different support structures (help versus no help) affect the same participant's performance across two systems.

To minimize potential biases often associated with within-subject designs – such as order effects and demand characteristics – we introduced the reversed-order groups (A2 and B2). This approach addresses the concern that participants may perform differently on the second task due to learning or fatigue from the first task, as highlighted by Charness et al. (2012). By alternating the order of tasks with and without help between groups, we reduce biases and strengthen the experiment's internal validity. This design ensures that any differences in performance are more likely due to the presence or absence of help, rather than the order in which the tasks are done

Each student was assigned two tasks, one with help and one without. The two sub-tasks were called:

- Deep Learning System using Transfer Learning (DLS).
- Reinforcement Learning System using Checkpoints (RLS).

The groups are composed as follows:

- (A1) DLS: system source code repository and additional help with semi-formal models + RLS: only system source code repository.
- (A2) RLS: only system source code repository + DLS: system source code repository and additional help with semi-formal models; identical to A1, only order reversed.
- (B1) RLS: system source code repository and additional help with semi-formal models + DLS: only system source code repository.
- (B2) DLS: only system source code repository + RLS: system source code repository and additional help with semi-formal models; identical to B1, only order reversed.

4.4. Participants

Our study involves advanced Bachelor's students enrolled in the Software Engineering 2 (SE2) and Distributed Software Engineering (DSE) courses. Both courses use an introductory pre-test designed to assess participants' knowledge related to the course. The pre-test allowed us to identify any knowledge gaps and tailor the course content to address specific areas of concern. Participation in the experiment, which consisted of a hands-on task during the lecture, contributed 5% towards the students' overall course points. No further incentives were provided beyond this participation credit. Participants were offered the possibility to engage in the hands-on task while opting out of the experiment. In preceding lectures leading up to the hands-on task, topics such as code reviews, design practices, and general design patterns were discussed. Participants were also introduced to the experiment through

[1] Reinforcement learning source code available at: https://web.archive.org/web/20240119081250/https://pettingzoo.farama.org/tutorials/sb3/kaz/.

[2] Refer to Data and Scripts/RL System Source Code/checkpoints_RL.py at https://doi.org/10.5281/zenodo.14677668.

[3] Transfer learning source code available at: https://github.com/GoogleCloudPlatform/ml-design-patterns/blob/master/04_hacking_training_loop/transfer_learning.ipynb.

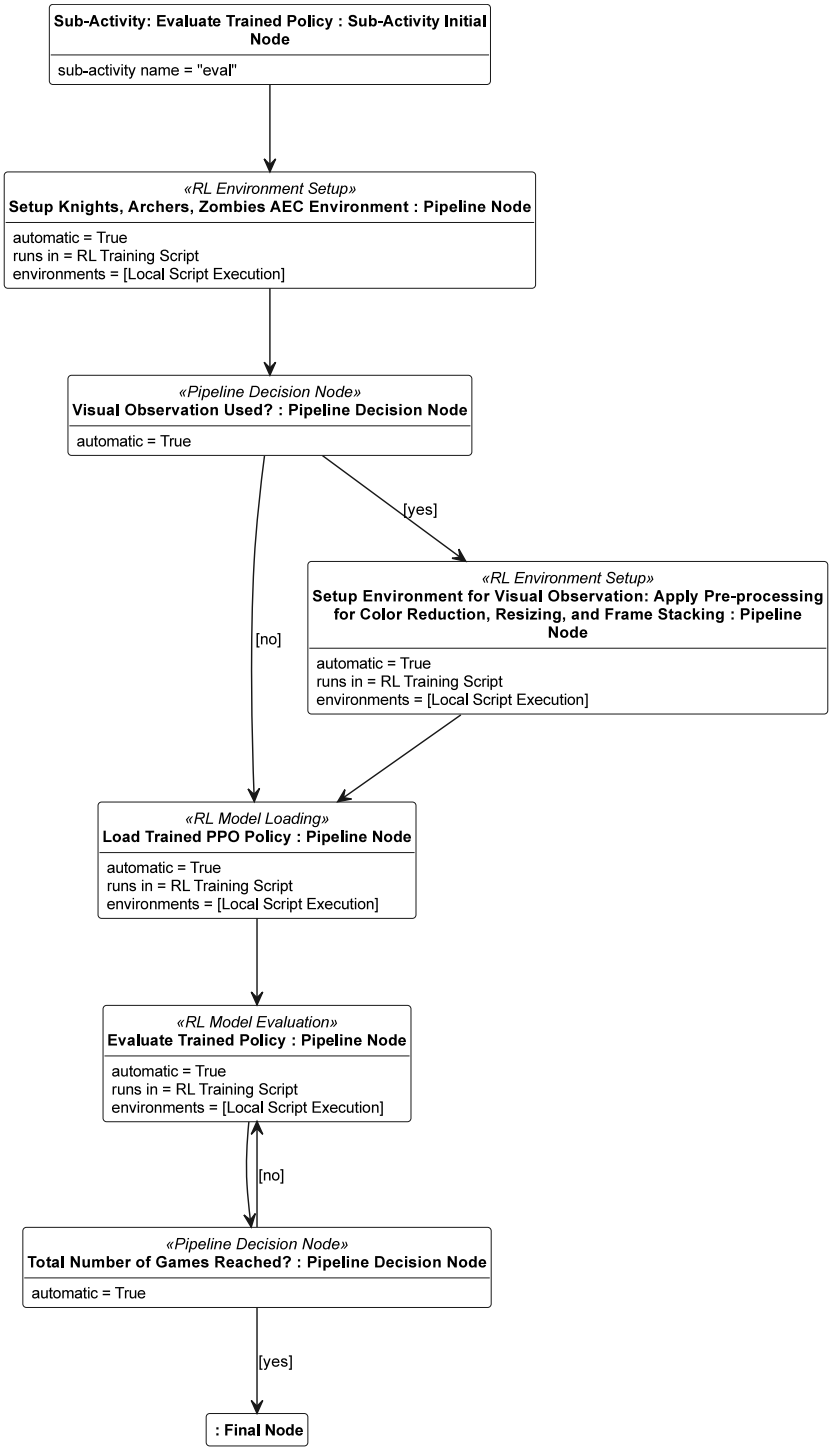

Fig. 2. Semi-formal model of the reinforcement learning script pipeline.

an informational text.[4] This introduction covered background in ML, pertinent deep learning and reinforcement learning practices, and the modeling techniques employed in the experiment.

Moreover, our participants show a varied spectrum of knowledge and expertise in the field of software development. A notable reference point is a 2016 survey conducted on the online platform Stack Overflow, which involved fifty thousand developers.[5] A comparison between the characteristics of our participants and those from the survey reveals noteworthy similarities in key demographics. Notably, all participants in our study possessed diverse levels of programming experience, a critical factor for effectively comprehending and reviewing the provided source code materials (see Section 6.2 for detailed information).

[4] Refer to the *Information Sheet* available in the experiment documents archived online at: https://doi.org/10.5281/zenodo.14677668.

[5] https://survey.stackoverflow.co/2016

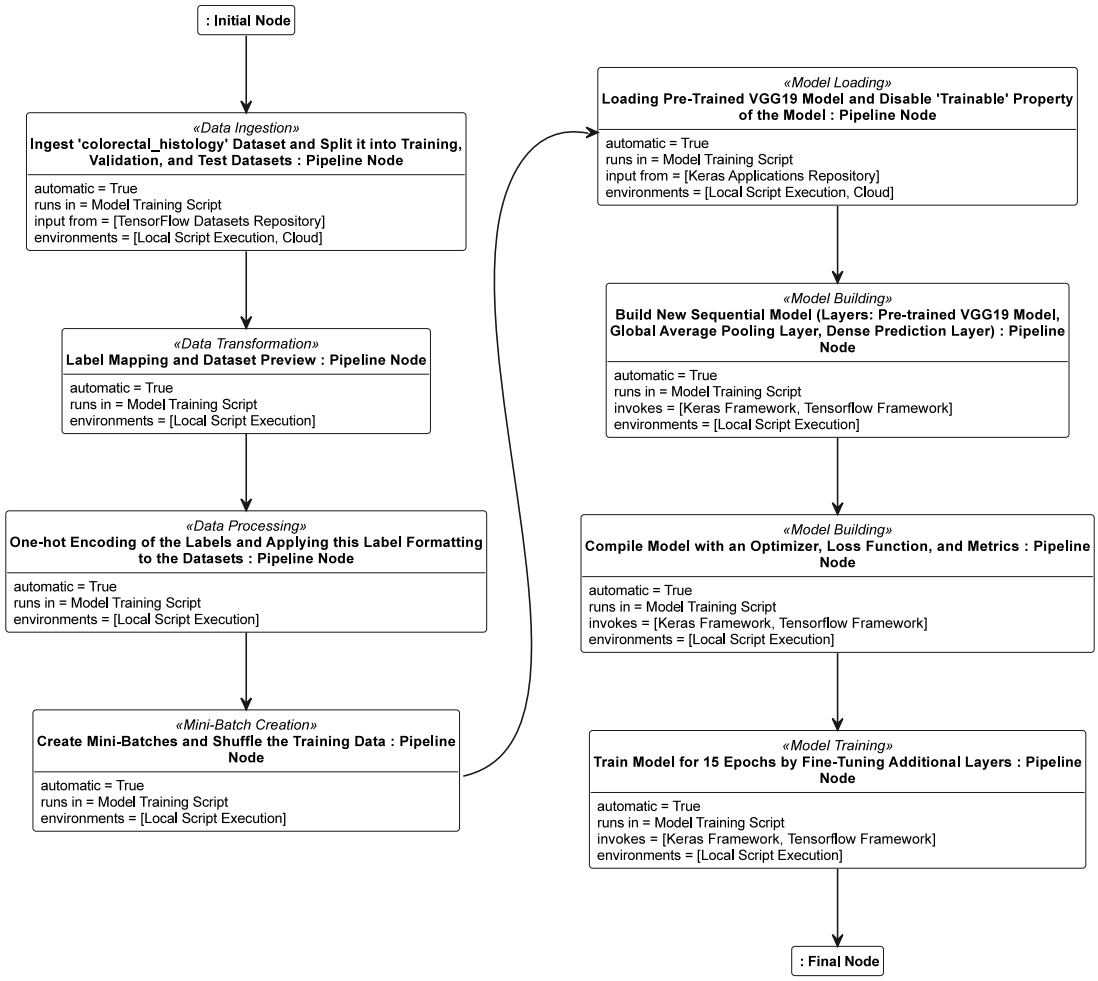

Fig. 3. Semi-formal model of the deep learning script pipeline.

In our study, we use advanced Bachelor's students as proxies for the target population of software developers with limited ML training or data scientists with limited software engineering training. Our study objectives do not revolve around techniques exclusive to a select group of highly trained experts, and as such, we did not specifically target such individuals. In alignment with the findings of Kitchenham et al. (2002), utilizing students in our study is deemed appropriate, as they represent the next generation of software professionals and share similarities with the population of interest. This is reinforced by the fact that some participants in our experiment possessed several years of programming and industry experience. Furthermore, corroborating studies by Höst et al. (2000), Runeson (2003), Svahnberg et al. (2008), Salman et al. (2015), and Falessi et al. (2018) argue that students can serve as valid representatives for professionals in specific empirical software engineering experiments.

4.5. Material and tasks

The experiment is based on a selection of ML practices. The selection includes the practices for *Checkpoints* and *Transfer Learning* summarized in Section 2. The selected software practices are related to the subjects taught in the courses SE2 and DSE, such as related software modeling techniques, software design patterns, and software architecture. This study consists of two major experiment material artifacts[6]:

- **(1) Information Sheet.** A document explaining the ML practices with an example model that was provided to participants two weeks before the experiment was conducted.
- **(2) Survey Form.** Four experiment survey forms per group were handed out to participants during the experiment sessions.

All experiment survey forms are structured the same way, consisting of three parts: (1) a participant information questionnaire; (2) two experiment tasks (for one of the four groups A1, A2, B1, B2 explained in Section 4.3); (3) an overall experiment questionnaire. Each task is divided into sub-tasks to test the participants' understanding of ML practices. The participants were instructed to read the code and descriptions in the given repositories for each system before they started to process the following four sub-tasks:

- Seven tasks with four true/false answers for practices were used to determine the understanding of used/supported/realized techniques in the provided systems. An example from Task RLS.1 follows:
  What is the purpose of using checkpoints during training in this example? (Multiple options might be correct)

  ☐ The checkpoints are part of the model evaluation step and are created to record the validation results.

  ☐ Checkpoints are used during training to save the policy training so far. This is particularly important in case training gets interrupted or for later model evaluation and deployment.

[6] See our replication package published on the long-term open repository Zenodo: https://doi.org/10.5281/zenodo.14677668.

Table 1
Structure of the questionnaire.

Section	Questions/Elements
Demographics	Age (Optional), Gender (Optional), Highest Education Level, Programming Experience, Modeling Experience, Industry Experience, Prior Knowledge in Deep Learning, Reinforcement Learning, and Formal Modeling
Deep Learning Task (DLS)	DLS.1 - Transfer Learning, DLS.2 - Model Layer Configurations, DLS.3 - Epochs Trained, DLS.4 - Component Relationships, DLS.5 - Order of Script Steps, DLS.6 - Automation and Outputs, DLS.7 - Training Practices, DLS.8 - Detailed Relations
Deep Learning Survey	Confidence in answers, Ease of understanding system descriptions and diagrams, Identifying components and relations, Understanding DL/RL practices, Component models for large-scale systems
Reinforcement Learning Task (RLS)	RLS.1 - Checkpoints in Training, RLS.2 - Checkpoint Frequency, RLS.3 - Agent Training, RLS.4 - Component Relationships, RLS.5 - Order of RL Script Steps, RLS.6 - Automation and Outputs, RLS.7 - Transfer Learning Relations, RLS.8 - Detailed Relations
Reinforcement Learning Survey	Confidence in answers, Ease of understanding system descriptions and diagrams, Identifying components and relations, Understanding RL practices
Concluding Survey	Familiarity with textual and graphical descriptions before the study

- ☐ Multiple agents are trained in parallel in the Reinforcement Learning environment. The checkpoints are stored every 1000 training steps using a callback.
- ☐ The checkpoints can be used to compare the results of different stages of the training. While training should usually improve over time, in Reinforcement Learning, earlier iterations may perform better than later ones. If this is the case, a checkpoint can be used instead of the model stored after the training.

- A task involving the appropriate sequencing of pipeline steps was employed to assess comprehension regarding the sequence of steps in the provided systems. For instance, consider Task DLS.5: Put the following major steps of the machine learning script into the correct order in which they are performed. Use numbers like 1, 2, 3 … Some steps are not part of the script. Add no number for those steps. If a step is performed repeatedly or more than once, only put in the number for the first occurrence and add "*".

  _____ Model Building
  _____ Model Training
  _____ Data Processing
  _____ Model Loading
  _____ Model Saving
  _____ Data Transformation
  _____ Data Version Control
  _____ Data Ingestion
  _____ Data Loading
  _____ Model Version Control
  _____ Mini-Batch Creation
  _____ Model Validation
  _____ Model Deployment
  _____ Hyperparameter Tuning

- In addition to the tasks, a task-based questionnaire was used to obtain an objective perspective later (see Section 6.6) of the participants' self-assessment based on their subjective assessment of how confident they were in the correctness of their answers compared with their answers' true correctness.

Table 1 presented above outlines the main structure of the questionnaire used in the hands-on tasks for Deep Learning and Reinforcement Learning

4.6. Variables and hypotheses

To address the research question, we formulated the following hypotheses to explore the relationship between experimental conditions and their impact on understanding ML practices. Our controlled experiment measures the following two dependent variables:

- **Correctness** as achieved in answering the questions, which includes trying to mark the correct answer and filling in the blanks in the tasks. It is used to measure the effectiveness of understanding ML practices.
- **Duration** as the time it took to complete the experiment tasks (excluding breaks). It is used to measure the efficiency of understanding ML practices.

The two dependent variables, *correctness* and *duration*, may be used to gain insight into the overall understandability of system source code and descriptions (Czepa and Zdun, 2019, 2020; Paulweber et al., 2021; Siegmund et al., 2012; Hoisl et al., 2014). Accordingly, we devised the following Null Hypotheses:

- **$H_0$1** There is no significant difference in *correctness* for *Experimental* compared to *Control*.
- **$H_0$2** There is no significant difference in *duration* for *Experimental* compared to *Control*.
- **$H_0$3** There is no significant increase in *correctness* as *duration* increases for *Experimental* compared to *Control*.

and the corresponding Alternative Hypotheses:

- **H_a1** There is significant difference in *correctness* for *Experimental* compared to *Control*.
- **H_a2** There is significant difference in *duration* for *Experimental* compared to *Control*.
- **H_a3** There is significant increase in *correctness* as *duration* increases for *Experimental* compared to *Control*.

5. Experiment execution

This experiment was executed in three stages: a preparation phase, a pilot test, and a procedure phase.

5.1. Preparation

Two weeks before the experiment, we distributed preparatory materials, including the experiment information sheet (refer to Section 2),

through an e-learning platform.[7] This document furnished participants with general information about the upcoming experiment and an introduction to ML practices. Comprising ML patterns and practices, an illustrative model, and a comprehensive description, the document served as a valuable reference. Participants were permitted to use a printed version of this document during the experiment. The distribution of the experiment information document was essential to ensure that all participants were equally well-informed about ML practices in deep learning and reinforcement learning, as outlined in Section 2.

5.2. Pilot test

Following the preparation of our experimental materials, we conducted preliminary tests with student tutors from our research group. Similar to participants in subsequent stages, we provided the tutors with the information sheet two weeks in advance, allowing them to familiarize themselves with the necessary knowledge for the tasks. The tutors participated in the experiment under predefined conditions, which included the option to seek clarifications on the experimental procedure, a ninety-minute timeframe for responding to experiment questions, and no additional support beyond the provided materials.

Beyond the pilot tests, we gathered feedback from the tutors about their experiences. Without making any changes to our experiment design, we took note of the tutors' feedback. The feedback we received suggests that our information sheet was expertly designed, providing clear instructions for tackling the experiment questions. We crafted the information sheet exceptionally well, offering ample guidance for addressing the experiment questions. Given this positive feedback, we maintained our original experiment design.

5.3. Procedure

The experiment took on the format of a closed-book exam, utilizing pen and paper, and the participants were provided with a printed version of the source code for both systems. Participants were restricted to bringing only the preparation material, as outlined in Section 4, to aid in completing the experiment tasks. At the commencement of the experiment, each participant was handed a random experiment survey form (refer to Section Section 4.5). Distribution ensured approximately equal numbers of forms of each type (A1, A2, B1, B2 in Section 4.3). Participants were instructed to systematically fill out and process the survey from the first page to the last page in the specified order. Additionally, a clock displaying seconds granularity was projected onto a wall to furnish timestamp information to the participants. They were directed to record start and stop timestamps while executing the experiment tasks. Subsequently, the participants' task start and stop timestamps were converted into a duration in seconds and aggregated to determine the total duration for all tasks. To safeguard the confidentiality of participant data, an individual not involved in the experiment dissociated personal information (such as name and student number) from the experiment sheets by assigning a unique identification number.

6. Analysis

The statistical analysis for the study was conducted using the R programming language.[8] This analysis contains several steps, including loading the pre-processed dataset outlined in Section 6.1, calculating descriptive statistics for the dependent variables discussed in Section 6.4, performing group-by-group comparisons through relevant statistical hypothesis tests as discussed in Section 6.5, and generating tables and plots. The reproduction of these results requires the installation of specific R library package dependencies.[9]

Fig. 4. Participants' age.

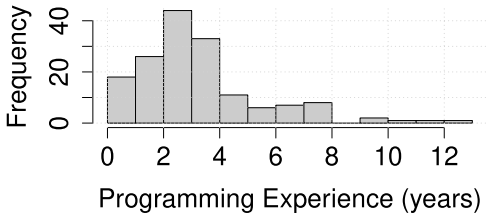

Fig. 5. Participants' programming experience.

6.1. Data-set preparation

The collected raw data[10] from the experiment execution phase (refer to Section 5) was prepared as follows: (1) a Microsoft Excel document was exported to a Comma-Separated Values (CSV) file; (2) the CSV file was imported for further processing; (3) type casting was performed for several data rows; and (4) the overall correctness for all task correctness values was calculated. The data set is published in the long-term open data archive Zenodo[11] together with all documents and R scripts.

6.2. Participant demographics

The experiment collects data on participants' age (as depicted in Fig. 4), gender, course, educational level, programming experience (refer to Fig. 5), industry experience in software (refer to Fig. 6), and knowledge of DL and RL practices (refer to Figs. 7 and 8) to capture their background information and experience.

The tables provide an overview of percentages for participants' experiences. The data is compared between two groups: DLS and RLS, each with both experimental and control conditions.

Table 2 shows participants' experience in the software industry. The largest group of participants (around 58%–65%) reported having no experience. For those with experience, the percentages drop as the years of experience increase. Most participants have between 1 to 2 years of experience, and only a few have more than that. Both the experimental and control groups have very similar patterns, meaning their levels of experience in the software industry are quite comparable.

Table 3 shows participants' experience with deep learning. Most people (about 58%–61%) have no experience, while around 39%–42% have some. This is true for both the experimental and control groups, meaning their knowledge of deep learning is quite similar.

In Table 4, we can see the participants' experience with reinforcement learning. A large majority (87%) reported no experience, while only 13% have some. This is the same for both the experimental and control groups, which suggests that reinforcement learning is not very common among participants.

Table 5 shows the programming experience of participants. There is a wide range of experience here, with many people having between 2 to 4 years. The largest group (around 25%–30%) has 3 years of

[7] https://moodle.univie.ac.at

[8] Refer to https://www.r-project.org for version 4.2.2.

[9] Refer to Data and Scripts/Scripts/install.r at https://doi.org/10.5281/zenodo.14677668.

[10] See Experiment Documents/Questionnaire Results/experiment-results.csv at https://doi.org/10.5281/zenodo.14677668.

[11] https://doi.org/10.5281/zenodo.14677668

Fig. 6. Participants' software industry experience.

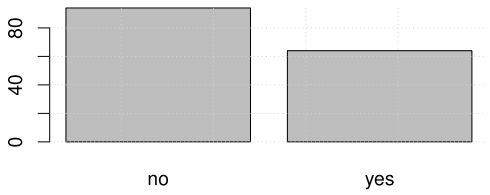

Fig. 7. Participants' deep learning experience.

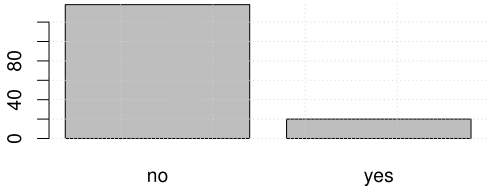

Fig. 8. Participants' reinforcement learning experience.

programming experience, followed by those with 4 years (17%–24%). The experimental and control groups are very similar in this area.

Table 6 shows participants' experience with modeling. Most participants have 2 years of experience (26%–35%), followed by those with 1 year (17%–21%). Both experimental and control groups have nearly the same distribution, meaning there are not big differences between them.

For the RLS the results are reversed for the *Control* and *Experimental* since we applied the Within-Subjects Design (Charness et al., 2012).

We did not specifically remove students with little ML or DL knowledge from the study because their goal was to include a wide range of experience levels, similar to what is often found in real-world projects. Participants had between 0 and 11 years of industry experience, and their programming experience ranged from 0 to 14 years. While some were early in their careers, others had more practical knowledge. The study also used questionnaires to get feedback from students who had some real-world experience, even if they were not experts.

By focusing on practitioners with limited ML expertise, we want to explore how they deal with DL and RL technologies. The fact that many participants were not ML specialists is not seen as a weakness but rather as a realistic reflection of the industry, where many developers must work with ML technologies without deep formal training. The authors argue that this mix of participants better represents the current tech landscape, where engineers often have to use ML in their work without being experts in the field.

While the non experts might not be ideal for understanding the behavior of highly specialized ML engineers or data scientists, we believe it accurately reflects the real-world experience of many software developers. These developers often face the challenge of using ML alongside their broader software engineering work, even if they do not have extensive ML training. We acknowledges that the inclusion of students with limited ML or DL knowledge might affect how well the results apply to more specialized groups, but it still offers valuable insights into how typical developers interact with ML practices.

Table 2
DLS experimental and control group percentage for experience in software industry.

Years	DLS experimental	DLS control
0	65.38	58.23
1	10.26	22.78
2	10.26	11.39
3	5.13	2.53
4	5.13	0.00
5	1.28	2.53
6	1.28	1.27
7	0.00	1.27
11	1.28	0.00

Table 3
DLS experimental and control group percentage for experience in deep learning.

Experience	DLS experimental	DLS control
No	60.76	58.23
Yes	39.24	41.77

Table 4
DLS experimental and control group percentage for experience in reinforcement learning.

Experience	DLS experimental	DLS control
No	87.34	87.34
Yes	12.66	12.66

Table 5
DLS experimental and control group percentage for experience in programming.

Years	DLS experimental	DLS control
0	3.80	6.33
1	5.06	7.59
2	16.46	16.46
3	30.38	25.32
4	17.72	24.05
5	6.33	7.59
6	6.33	1.27
7	3.80	5.06
8	7.59	2.53
10	1.27	1.27
11	0.00	1.27
12	0.00	1.27
13	1.27	0.00

Table 6
DLS experimental and control group percentage for experience in modeling.

Years	DLS experimental	DLS control
0	5.06	13.92
0.33	0.00	1.27
0.5	5.06	2.53
1	17.72	21.52
1.5	3.80	6.33
2	35.44	26.58
2.5	3.80	5.06
3	21.52	12.66
3.5	0.00	1.27
4	5.06	7.59
7	0.00	1.27
8	1.27	0.00
9	1.27	0.00

6.3. Normality assessment

The normal Q-Q plots for DLS (Fig. 9) for *correctness* indicate that the data for both *Control* and *Experimental* appear to follow a normal distribution. Furthermore, examining the normal Q-Q plots for RLS for *correctness* (Fig. 10), it can be inferred that the data for *Control*

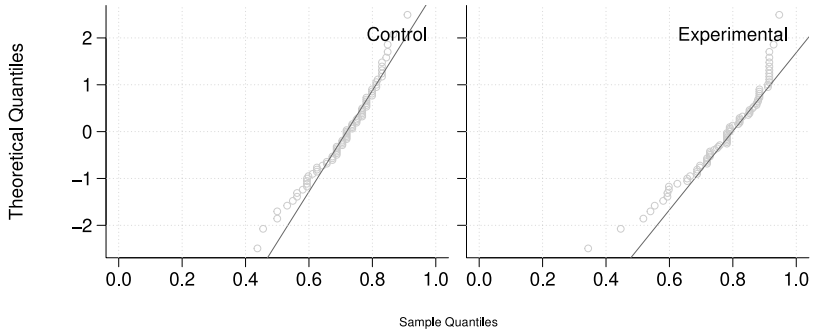

Fig. 9. Normal Q-Q plot of *Correctness* (DLS).

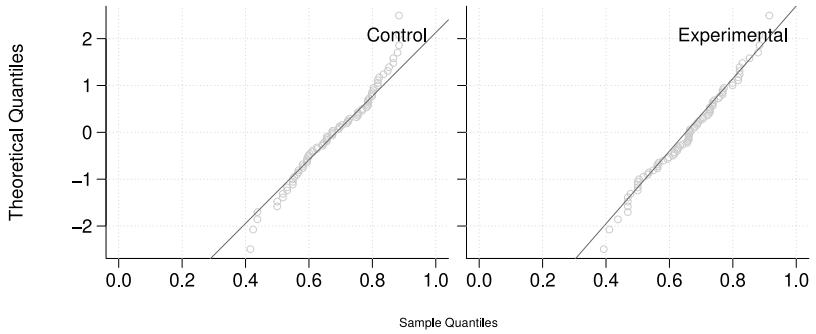

Fig. 10. Normal Q-Q plot of *Correctness* (RLS).

appears normally distributed. However, it is inconclusive regarding the normality of the data for *Experimental*.

To test for normality, we chose the Shapiro–Wilk (Shapiro and Wilk, 1965) normality test since, according to Razali and Yap (Mohd Razali and Yap, 2011), it is more powerful than alternatives (such as Anderson–Darling (Anderson and Darling, 1954), Lilliefors (Lilliefors, 1967), and Kolmogorov–Smirnov (Kolmogorov, 1933)). Assuming $\alpha = 0.05$, the test for DLS shows that the *Control*'s and *Experimental*'s distributions are not significantly different from the normal distribution, with a value $p > \alpha$. For RLS, the test indicated that the *Control*'s distribution is not significantly different from the normal distribution either, whereas *Experimental*'s distribution significantly deviates from the normal distribution, with a value $p \leq \alpha$.

Visual inspection of the normal Q-Q plots for both groups and both systems for *duration*, visible in Figs. 11 and 12, was insufficient to determine whether each group's data were normally distributed. The Shapiro–Wilk normality test for DLS indicated that *Experimental*'s distribution significantly deviates from the normal distribution. In contrast, *Control*'s distribution is not significantly different from the normal distribution, with a value $p \leq \alpha$. However, for RLS, the test indicated that, for *duration*, the groups' distributions significantly deviate from the normal distribution, with a value $p \leq \alpha$ for both groups.

6.4. Descriptive statistics

Correctness

Table 7 and Table 8 show the number of corresponding observations, central tendency measures, and dispersion measures per group for the dependent variable.*correctness*[12] These statistics are illustrated as a kernel density plot in Fig. 13 and Fig. 15 and as box-plots in Fig. 14 and Fig. 16

Visual inspection of the DLS correctness results (Fig. 13) and the values in Table 7 indicates significant differences between *Control* and *Experimental* across various statistical measures. The standard and median absolute deviations demonstrate greater variability in *Experimental*, indicating a wider range of data points. The skewness values indicate that both groups exhibit a degree of asymmetry, with the *Experimental* having a more pronounced leftward skew. Negative kurtosis in *Control* suggests a flatter distribution compared to *Experimental*, which shows a more peaked distribution. The Shapiro–Wilk Test p-values below the conventional significance level (0.05) suggest that both groups deviate from a normal distribution.

Table 7
Descriptive statistics per group of dependent variable *Correctness* (DLS).

	Control	*Experimental*
Number of observations	79	79
Mean	0.7121	0.7759
Standard deviation	0.1006	0.1242
Median	0.7188	0.7902
Median absolute deviation	0.0927	0.1324
Minimum	0.4375	0.3438
Maximum	0.9107	0.9464
Skew	−0.6328	−0.9525
Kurtosis	−0.1057	0.7913
Shapiro–Wilk Test *p*-value	0.0191	0.0002

Table 8
descriptive statistics per group of dependent variable *Correctness* (RLS).

	Control	*Experimental*
Number of observations	79	79
Mean	0.6808	0.6612
Standard deviation	0.1237	0.1233
Median	0.6741	0.6652
Median absolute deviation	0.1588	0.1324
Minimum	0.4152	0.3929
Maximum	0.8839	0.9152
Skew	−0.1647	−0.1286
Kurtosis	−0.9471	−0.7313
Shapiro–Wilk Test *p*-value	0.0383	0.3494

[12] *correctness* is defined as a value in $[0, 1] \cap \mathbb{R}$.

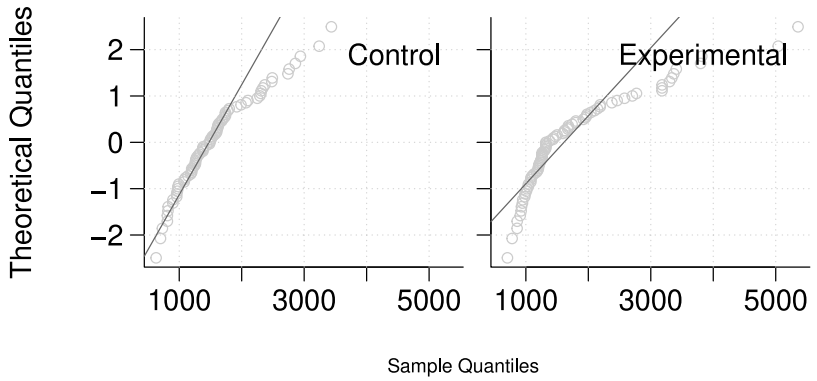

Fig. 11. Normal Q-Q plot of *Duration* (DLS).

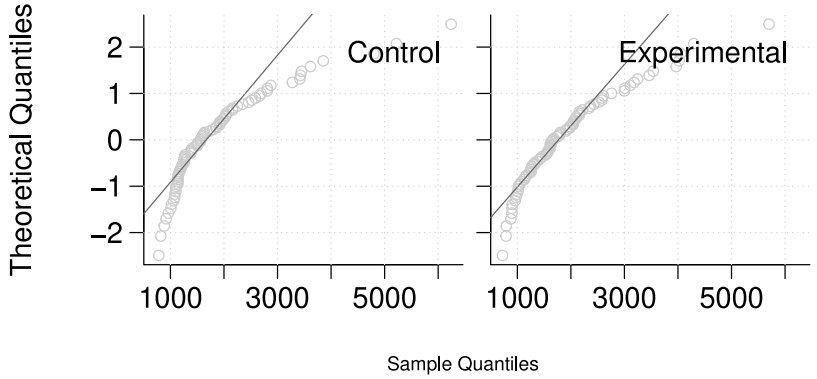

Fig. 12. Normal Q-Q plot of *Duration* (RLS).

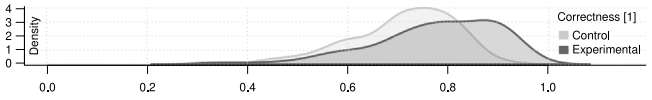

Fig. 13. Kernel density plot of *Correctness* (DLS).

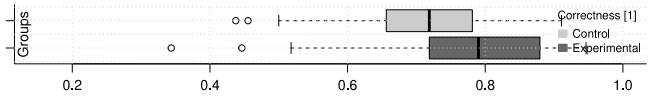

Fig. 14. Box-Plots of *Correctness* (DLS).

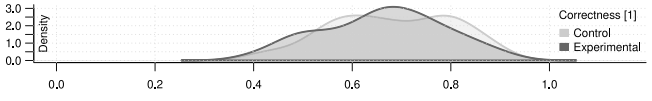

Fig. 15. Kernel density plot of *Correctness* (RLS).

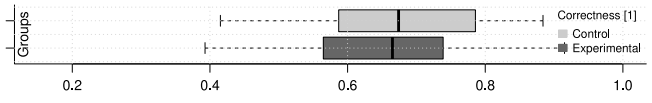

Fig. 16. Box-Plots of *Correctness* (RLS).

Moreover, for the RLS, the statistical measures in Table 8 for the *Control* and *Experimental* groups are presented. The mean values indicate that *Control* has a slightly higher average (0.6808) than *Experimental* (0.6612). Both groups exhibit similar standard deviations (0.1237 for *Control*, 0.1233 for *Experimental*), indicating comparable variability.

Table 9
Descriptive statistics per group of dependent variable *Duration* (DLS).

	Control	*Experimental*
Number of observations	79	79
Mean	1571.62	1763.85
Standard deviation	607.84	954.04
Median	1489	1320
Median absolute deviation	431.44	518.91
Minimum	630	705
Maximum	3436	5357
Skew	0.9501	1.6488
Kurtosis	0.5306	2.6149
Shapiro–Wilk Test *p*-value	0.0003	0.0000

The median values and median absolute deviations are close, suggesting similar central tendencies and dispersion around the median. The minimum and maximum values show that the range of data points is broader in *Control*. Skewness values are negative for both groups, indicating a slight leftward asymmetry. Negative kurtosis values suggest relatively flatter distributions for both groups. The Shapiro–Wilk Test p-values (0.0383 for *Control*, 0.3494 for *Experimental*) indicate that *Control*'s data deviates from a normal distribution, while *Experimental*'s data is more consistent with normality.

Duration

Table 9 shows the number of observations, central tendency measures, and dispersion measures per group for the dependent variable.*duration*[13] Results for DLS shown in Table 9 indicate substantial differences in the distribution of the observed values. The mean values show a notable distinction, with *Control* having a mean of 1571.62 and *Experimental* having a higher mean of 1763.85. The standard deviation

[13] *Duration* is denoted in seconds.

Table 10
Descriptive statistics per group of dependent variable *Duration* (RLS).

	Control	*Experimental*
Number of observations	79	79
Mean	1883.80	1925.46
Standard deviation	1019.86	949.80
Median	1560	1670
Median absolute deviation	622.69	726.47
Minimum	782	723
Maximum	6242	5700
Skew	1.9091	1.3771
Kurtosis	4.2426	2.1888
Shapiro–Wilk Test *p*-value	0.0000	0.0000

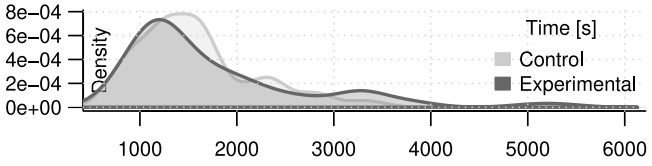

Fig. 17. Kernel density plot of *Duration* (DLS).

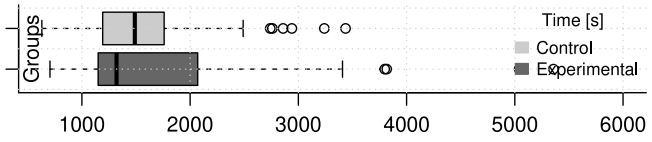

Fig. 18. Box-Plots of *Duration* (DLS).

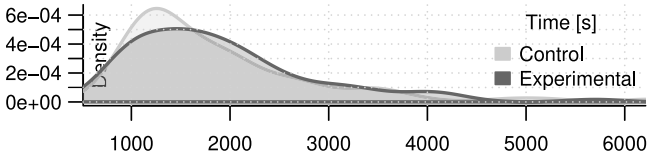

Fig. 19. Kernel density plot of *Duration* (RLS).

and median absolute deviation are considerably larger for *Experimental*, signifying greater variability in the data compared to *Control*. The median values indicate that the center of the data is lower in *Experimental* (1320) than in *Control* (1489). The range of values, as reflected by the minimum and maximum, is wider in *Experimental*. Skewness values reveal that both groups exhibit positive skewness, indicating a tail towards higher values, with *Experimental* having a more pronounced skew. The kurtosis values suggest that the distributions for both groups are more heavy-tailed than a normal distribution. The Shapiro–Wilk Test p-values are very low for both groups, indicating a significant difference from normality.

Moreover, for RLS, the data in Table 10 indicates differences in the distribution of the observed values. The mean values show that *Experimental* has a slightly higher mean (1925.46) than *Control* (1883.80). The standard deviation is lower in *Experimental* (949.80) compared to *Control* (1019.86), suggesting less variability in *Experimental*. The median values indicate that the center of the data is higher in *Experimental* (1670) than in *Control* (1560). The range of values, as reflected by the minimum and maximum, is wider in *Control*. Skewness values reveal that both groups exhibit positive skewness, indicating a tail towards higher values, with the Control group having a more pronounced skew. The kurtosis values suggest that the distribution for *Control* is highly heavy-tailed, while *Experimental* is also heavy-tailed but to a lesser extent. The Shapiro–Wilk Test p-values shows similar results to DLS.

These statistics are illustrated as a kernel density plot in Fig. 17 and Fig. 19 and as box-plots in Fig. 18 and Fig. 20

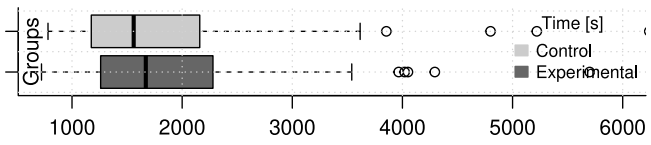

Fig. 20. Box-Plots of *Duration* (RLS).

Table 11
Hypothesis tests per group for correctness.

System: DLS		System: RLS	
	cont vs. *expr*		*cont* vs. *expr*
Cliff's δ Test		Cliff's δ Test	
Cliff's δ	0.3616	Cliff's δ	−0.0803
s_δ	0.0854	s_δ	0.0922
v_δ	0.0073	v_δ	0.0085
z_δ	4.2336	z_δ	−0.8706
CI (low)	0.1837	CI (low)	−0.2569
CI (high)	0.5166	CI (high)	0.1015
$P(X > Y)$	0.3089	$P(X > Y)$	0.5329
$P(X = Y)$	0.0205	$P(X = Y)$	0.0144
$P(X < Y)$	0.6706	$P(X < Y)$	0.4527
p	0.0000	p	0.3853

6.5. Hypothesis testing

Correctness and *Duration*

Suppose multiple groups are being compared on several dependent variables, as is the case in this study. Then, it is customary to employ the Multivariate Analysis of Variance (MANOVA) statistical test under the condition that specific assumptions are satisfied (Bray and Maxwell, 1982). This test helps determine whether independent variables impact the dependent variables, individually or in combination. As Section 6.4 mentions, the distribution of *Control* for *correctness* and *duration* does not significantly differ from the normal distribution. However, the distribution of *Experimental* significantly differs from the normal distribution for *correctness*, but not for *duration*. It is important to note that MANOVA requires all distributions to be normally distributed. Therefore, it was essential to compare the variances of both groups for both dependent variables to assess their equality, as this information would guide the selection of appropriate statistical tests. In this study, we opted to use Cliff's δ (Cliff, 1993) as a robust and nonparametric test recommended by Kitchenham for scenarios where differing distributions between populations, or unequal variances are present. Although Cliff's δ was originally designed for measuring ordinal data, it is equally applicable to the quantitative and continuous data used in this study (Delaney and Vargha, 2002; Hsu, 2004). This test estimates the probability that a randomly selected observation from one group is larger than a randomly selected observation from another group, taking into account the reverse probability (Cliff, 2010).

When conducting multiple hypothesis tests using a single method (in this case, Cliff's δ was applied twice), it is necessary to adjust the significance level (α) to mitigate the risk of Type I errors.[14] Several methods can be employed for α adjustment, such as the false discovery rate (Benjamini and Hochberg, 1995) or the Bonferroni–Dunn (Bonferroni, 1936; Dunn, 1961) correction. The Bonferroni–Dunn correction is the most stringent form of correction and can be calculated using Equation: $\alpha' = \frac{\alpha}{n}$ or $\alpha' = \frac{\alpha}{n}$ where n is the number of times a test was applied. In our study, this results in $\alpha' = \frac{0.05}{2} = 0.025$ where $\alpha = 0.05$ and $n = 2$. The results of the one-tailed Cliff's δ test are shown in Table 11 for *correctness* and Table 12 for *duration*.

[14] It is important to note that there is no need to adjust the significance level (α) when conducting tests for normality or comparing variances.

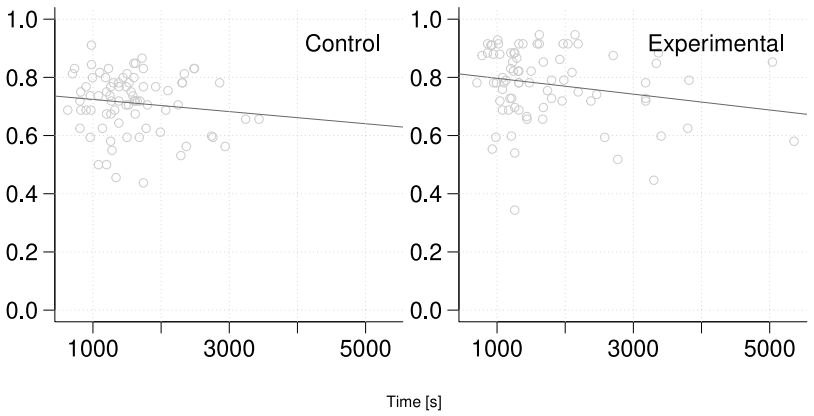

Fig. 21. Scatter plot per group of the dependent variables *Correctness* to *Duration* (DLS).

Table 12
Hypothesis tests per group for duration.

System: DLS	
	cont vs. *expr*
Cliff's δ Test	
Cliff's δ	0.0333
s_δ	0.0929
v_δ	0.0086
z_δ	0.3587
CI (low)	−0.1483
CI (high)	0.2128
$P(X > Y)$	0.4799
$P(X = Y)$	0.0069
$P(X < Y)$	0.5132
p	0.7203

System: RLS	
	cont vs. *expr*
Cliff's δ Test	
Cliff's δ	0.0570
s_δ	0.0926
v_δ	0.0086
z_δ	0.6159
CI (low)	−0.1248
CI (high)	0.2351
$P(X > Y)$	0.4679
$P(X = Y)$	0.0072
$P(X < Y)$	0.5249
p	0.5388

Table 13
Correlation per group of the dependent variables *Correctness* with *Duration* per Group (DLS)

	Control	*Experimental*
Spearman's ρ	−0.0651	−0.1640
p	0.5688	0.1486
S	87 506.3029	95 635.5933

Table 14
Correlation per group of the dependent variables *Correctness* with *Duration* per Group (RLS)

	Control	*Experimental*
Spearman's ρ	−0.1185	0.0811
p	0.2982	0.4773
S	91 897.0184	75 495.2409

For *correctness*, Cliff's δ indicates by $p \leq \alpha'$ that *Experimental* scored significantly higher than *Control*. For *duration*, Cliff's δ yielded $p > \alpha'$, so we cannot conclude that *Experimental* took significantly longer than *Control* to complete the experiment.

DLS. Based on Cliff's δ we fail to reject the null hypothesis $\mathbf{H_0 1}$, indicating that there is not enough statistical evidence to support the alternative hypothesis $\mathbf{H_a 1}$.

RLS. Concerning *correctness*, we reject the null hypothesis $\mathbf{H_0 1}$, which suggests sufficient evidence to support the alternative hypothesis $\mathbf{H_a 1}$.

Regarding *duration*, for both DLS and RLS, we again fail to reject the null hypothesis $\mathbf{H_0 2}$, indicating insufficient evidence for the alternative hypothesis $\mathbf{H_a 2}$.

Correlation Between *Correctness* and *Duration* for DLS

Upon visually examining the scatter plot in Fig. 21, which explores potential correlations between the two dependent variables *correctness* and *duration*, no evident linear correlation was observed for either group. For both *Control* and *Experimental*, there was a decrease in *correctness* with respect to time.

After the visual inspection, we deemed it necessary to perform a correlation test. Spearman's ρ test was selected for this purpose.

Control group. Spearman's ρ coefficients (see Table 13) indicated a negative association between *correctness* and *duration*. However, given that $p > \alpha'$ (where α' is derived from the earlier adjustment of α), we fail to reject the null hypothesis $\mathbf{H_0 3}$, indicating insufficient evidence to support the alternative hypothesis $\mathbf{H_a 3}$ for both tests.

Experimental group. Spearman's ρ coefficients (see Table 13) also revealed a negative association between *correctness* and *duration*. Despite this, the p-value $p > \alpha'$) suggests that the observed association is not statistically significant, leading us to fail to reject the null hypothesis $\mathbf{H_0 3}$. Additionally, the value of S obtained from Spearman's ρ test indicates that the ranks of the two variables are not identical, but this evidence is not strong enough to support the alternative hypothesis.

Correlation Between *Correctness* and *Duration* for RLS

A visual inspection of potential correlations in Fig. 22 per group did not reveal any significant linear correlation. In the case of *Control*, there was a minimal decrease in *correctness* relative to time. On the other hand, for *Experimental*, although there seemed to be a rise in *correctness* with *duration*, the data points were widely dispersed from the indicated reference line, making it inappropriate to assume a linear correlation.

Control group. Spearman's ρ coefficients (see Table 14) yielded similar results to DLS, indicating a weak positive association between *correctness* and *duration*. However, given that $p > \alpha'$, we also fail to reject the null hypothesis $\mathbf{H_0 3}$, providing insufficient evidence to support the alternative hypothesis $\mathbf{H_a 3}$.

Experimental group. Spearman's ρ coefficients (see Table 14) showed a very weak positive association between *correctness* and *duration*. As with DLS, the p-value suggests that we cannot reject the null hypothesis $\mathbf{H_0 3}$, indicating insufficient evidence for the alternative hypothesis $\mathbf{H_a 3}$.

6.6. Observation

Following the details provided in Section 4.5, participants were instructed to fill out a survey after each task to evaluate their confidence level in the accuracy of their responses. This self-assessment score was calculated using a formula that considers both the *correctness* of participants' answers and their level of confidence in the correctness. The derived formula for calculating the self-assessment score is presented in Eq. (1):

$$self_{assessment} = self_{correctness} - \frac{5 - self_{confidence}}{5} \quad (1)$$

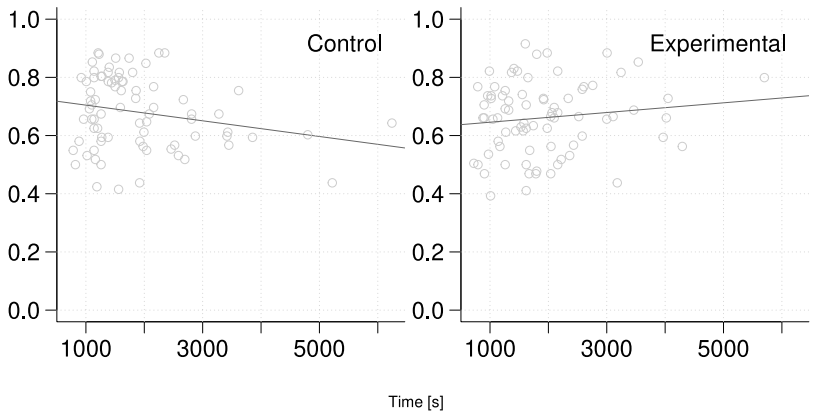

Fig. 22. Scatter plot per group of the dependent variables *Correctness* to *Duration* (RLS).

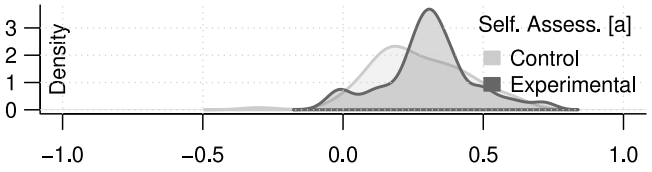

Fig. 23. Kernel density plot per group of participants' self assessment (DLS).

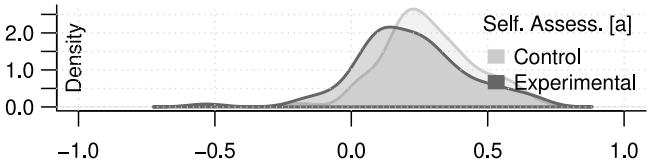

Fig. 24. Kernel density plot per group of participants' self assessment (RLS).

where $self_{correctness}$ represents the participants' average *correctness* as defined in 6.4, with 0 indicating entirely incorrect answers and 1 indicating completely correct answers.

The variable $self_{confidence} \in [1,5] \cap \mathbb{R}$ represents the average confidence of participants based on a survey that utilized a five-point Likert scale. Each point on the scale is assigned equidistant values within the range of 1 to 5. A value of 1 indicates high confidence in the correctness of the answers, while a value of 5 indicates low confidence.

The variable $self_{assessment} \in [-1,1] \cap \mathbb{R}$ represents the average self-assessment score of participants. A value less than 0 ($self_{assessment} < 0$) indicates that participants tend to overestimate the correctness of their answers. A value of 0 ($self_{assessment} = 0$) indicates that participants accurately estimate the correctness of their answers. A value greater than 0 ($self_{assessment} > 0$) indicates that participants tend to underestimate the correctness of their answers.

Fig. 23 illustrates the kernel density plot of participants' overall self-assessment score per group. The plot shows that, on average, participants in both *Control* and *Experimental* groups underestimated their *correctness* across all tasks. Similarly, Fig. 24 indicates that both *Control* and *Experimental* groups underestimated their *correctness*.

7. Discussion

7.1. Interpretation of the results

Correctness and *Duration*

In the DLS study, our investigation of $\mathbf{H_0 1}$ revealed a significant finding: there was a difference in task *correctness*, indicating that providing semi-formal ML system diagrams alongside source code improves performance. However, when we tested $\mathbf{H_0 2}$, we found no significant difference in task *duration* between *Control* and *Experimental*. Regarding $\mathbf{H_0 3}$, we observed that for both *Control* and *Experimental*, there was no significant correlation between *correctness* and *duration*.

In contrast, during the RLS study, our examination of $\mathbf{H_0 1}$ showed that *Control* performed similarly to, and slightly better than, *Experimental* in task *correctness*. This result can be caused by a number of factors. This includes the relatively small size of the system's source code and the added complexity from multiple diagrams. Participants in the *Experimental* group may struggle to fully understand some of the details in these diagrams, leading to minor misunderstandings. Meanwhile, those in the *Control* group could more easily grasp the underlying practices since they only had to deal with a modest amount of source code.

Additionally, we noticed that modeling recurring elements as a sub-activity within another model might have introduced unnecessary complexity. To address this, we included a third sub-model, but it became clear that these sub-models might have added structural complexities similar to those in the source code itself. This suggests that models need to reach a certain level of abstraction to be truly useful, though determining the exact threshold for this abstraction remains an area for future research.

Similarly, when testing $\mathbf{H_0 2}$, we found no significant difference in task *duration* between *Control* and *Experimental*. Moreover, in evaluating task *duration*, it was apparent that participants in the *Experimental* group took slightly more time to answer the questions. This result can be expected given the additional cognitive load associated with analyzing both the source code and the accompanying diagrams. When testing $\mathbf{H_0 3}$, we discovered that within *Control*, no significant positive correlation between *correctness* and *duration* was discerned; conversely, within *Experimental*, a positive correlation between *correctness* and *duration* was detected.

Furthermore, we must recognize that the complexity inherent in the source code of DL systems presents unique challenges. It requires a careful effort to identify and understand the various practices embedded within them. When we contrast the modeling approaches of RLS and DLS systems, an interesting observation emerges. The RLS model delved into finer details, offering a comprehensive depiction of the workflow, while the DLS model embraced abstraction, providing a broader overview. This difference suggests that the value of models may lie in their ability to offer a level of abstraction distinct from the source code. It leads us to consider that if the source code is available, models might be most beneficial when they operate at a higher level of abstraction. We employed different modeling strategies in developing these two systems, each serving a unique purpose.

Correlation Between *Correctness* and *Duration*

Our initial impressions and intuition about DLS led us to expect a positive correlation between *correctness* and *duration*. However, contrary to

our expectations, in the case of *Control*, there was no positive correlation between *correctness* and *duration*. This indicates that participants in *Control* were satisfied with the correct answer within a reasonable amount of time. and continue the work accordingly.

On the other hand, in the case of RLS, as shown in Figs. 21 it is clear that *Experimental* shows improvement with more time in *correctness* in contrast to *Control*. Consistent with our initial hypothesis in our results for *Experimental* in both *correctness* and *duration* show a positive correlation. It is possible that participants in *Experimental* adopted a different strategy for answering the questions. Since the only difference between *Control* and *Experimental* was the inclusion of semi-formal ML system diagrams, it is plausible that participants heavily relied on these resources in *Experimental*. Alternatively, participants may have initially consulted the source code for answers and then used the semi-formal ML system diagram to review and edit their answers. This leads to a positive relationship between *correctness* and *duration*. This resulted in a positive correlation between *correctness* and *duration*.

Therefore, participants working on source code should be aware that spending more time on such tasks does not necessarily lead to better performance in terms of *correctness*. Participants in *Control* spent the same amount of time on average as compared to participants in *Experimental* before being satisfied with the answer. This suggests that other factors may affect the performance. In this study, semi-formal models of ML systems have proven to be crucial, especially when the time constraints are the same. This result further supports the necessity of providing semi-formal ML system diagrams for practitioners.

Addressing the RQ

In the DLS, we found that giving participants models along with source code did help improve their accuracy in answering questions, meaning they understood the ML practices better. However, the extra help did not make them complete the tasks any faster. This might be because the models added more complexity, especially since the system was fairly simple. In cases where the task was simple, just having the source code was enough for participants to understand well. The extra diagrams may have actually caused some confusion.

Overall, the study suggests that semi-formal models can improve understanding, but how useful they are depends on the complexity of the task and how clearly the models are designed. For easier tasks, source code alone might work better, but for more complex systems, the models can make a positive difference.

We also found that in the RLS, participants who had the models took more time but were more accurate in their answers. This suggests that the models helped them think more deeply. So, while models can be useful, they should not be complex to be helpful for developers who are new to ML.

Self Assessment

In Section 6.6, our examination of participants' survey responses in the context of DLS yielded an interesting finding: Those who did not receive semi-formal ML system diagrams were more likely to estimate their performance slightly more accurately than those with the additional materials. Conversely, in the RLS, participants who did not receive semi-formal ML system diagrams tended to underestimate their performance slightly more than those who did receive them. The provision of additional semi-formal ML system diagrams may have contributed to a reduction in confidence levels. It is possible that these participants may be overwhelmed with information or have doubts about their complete understanding of semi-formal ML system diagrams.

This result deviated from our initial expectations as we expected higher levels of confidence in participants who performed better. However, this finding does not strongly support or oppose the use of semi-formal ML system diagrams. Practitioners' tendencies to display overconfidence, underconfidence, or a position somewhere in between are subjective and context-dependent; they cannot be solely deduced from the relationship between *correctness* and *duration*.

Therefore, this observation remains neutral, with no definitive tremendous or negative effect on our hypothesis. However, we acknowledge the significance of careful consideration throughout our analysis.

8. Threats to validity

Threats to Internal Validity

The experiment proceeded without any problems that could affect the process. Participants received clear instructions and an opportunity to ask questions. There are no major concerns that may affect the sessions and any individual questions will be handled personally.

The limited time for each session helped reduce the chance of any changes over time, and no such effects were observed. Each participant took part in only one session, preventing any learning between sessions. Any learning that happened within a session did not give an advantage to either *Control* or *Experimental*. All participants had an equal chance to earn points for their performance, no matter which group they were in, avoiding any bias in scoring. The random assignment of participants to groups also helped prevent selection bias.

Although it was not possible to completely stop participants from discussing the experiment with others, steps were taken to reduce the chance of this happening between sessions. Participants were not allowed to take any materials with them or use electronic devices during the experiment. The complexity of systems and tasks, including the distance between activities, reduces the opportunity for participants to gain an advantage over the session. The random allocation of groups attempts to ensure an equitable distribution of benefits. The ban on electronic devices also prevents participants from accessing outside information. As described in Section 5.3, the only authorized materials are the printed information sheets described in Section 4.5 and, together with restricting access to the source code, prevent participants from using other external sources.

Threats to External Validity

One issue that may affect the external validity of our study is the size of the data, which may not be sufficient to produce statistically significant results. To answer this concern we use robust statistical methods that are tailored to the size of our sample. This ensures that the analysis is carried out appropriately and accurately given the available data.

Another consideration of external validity arises from the use of students rather than non-student professionals in our study. This raises questions about the generalizability of our findings to practical settings. To alleviate this problem we took measures to familiarize students with the ML-related concepts used in the experiment. All participants have different levels of theoretical background in software engineering, distributed system programming, and industry experience. According to the Stack Overflow industry survey (refer to Section 4.4), 69% of respondents were self-taught, 43% held a bachelor's degree in computer science or a related field, 19% had a master's degree, and 2% had a Ph.D. These statistics indicate that a significant portion of the fifty thousand developers surveyed, even on a widely respected software development platform, are self-taught and lack formal degrees. This implies that having a degree may not necessarily be a prerequisite for qualifying as a professional developer.

Given this context, we assert that students can reasonably serve as substitutes for developers in our study. Consequently, we propose that our findings may apply to professional software developers, at least to some extent. However, it would be prudent to replicate similar experiments with practitioners to confirm the absence of significant differences compared to the population of professional developers.

While developers would normally use IDEs in real life, our experiment was designed to test how well people understood the code when diagrams were included, not how fast they could navigate it. Using

printed code helped us see how the diagrams affected understanding without other factors getting in the way. We agree that using IDEs might change how long tasks take, but the controlled setup we used still provides useful insights.

Threats to Construct Validity

Challenges to construct validity occur when there is uncertainty about whether the methods used to measure and define variables truly represent the concepts being studied. These challenges can include doubts about the accuracy of measurement methods, unclear definitions, instruments that are not sensitive enough, or bias introduced by the researcher. In Section In Section 4.6, we focused on *correctness* and *duration* as dependent variables to assess understandability, but we acknowledged that other metrics might be more appropriate. Additionally, there could be better ways to measure participants' confidence in answering our research question.

It is important to ensure that the times recorded by participants accurately reflect the time they spent completing tasks. Threats to construct validity here could include unclear task definitions, leading to inconsistent timekeeping. We tried to reduce this risk by giving participants clear instructions and guidance.

To ensure reliable self-timing, we carefully checked the reliability of all recorded timestamps when preparing the dataset, removing any inconsistent or unrealistic entries. We also addressed any missing or implausible time values to reduce the chance of unreliable time data affecting our results. Participants were asked to use a centrally controlled clock with high readability and accuracy to avoid errors in timing, such as manual manipulation or misreading, and to ensure consistency across all sessions.

We carefully created the diagrams to show the workflows of both systems (DLS and RLS) in a clear and structured way. However, because RLS is more complex, we needed multiple diagrams to cover all the details. This might have made the diagrams harder to understand for RLS compared to DLS, leading to a bigger gap between the diagrams and the code. This could be one reason why the results for RLS were different.

To make sure the diagrams were useful for both systems, we designed the questions at the same level of detail, so the diagrams would be equally helpful. We also checked the diagrams to ensure they were accurate and captured the main parts of each system without adding too much complexity. Still, we recognize that the bigger gap between the RLS diagrams and the code may have affected how participants understood the system, and this is something future studies could look into further.

Threats to Content Validity

We consider no threat to the integrity of the content. This is because the subject and topic of the test are related to the university courses of all participants. Regardless of their group assignment, the information materials provided to participants also provided sufficient background knowledge to enable them to effectively participate in the study. Any inconsistencies or ambiguities in the experiment material would have impacted both groups uniformly.

Threats to Conclusion Validity

Considering that the experiment's topic and subject matter were pertinent to all participants' university courses, regardless of their group assignment, we anticipate no threats to content validity. Additionally, the information sheet provided to participants contained the basic knowledge needed to participate in the study. Discrepancies or uncertainties in the experimental materials affected both groups equally to ensure a fair and consistent experience.

9. Conclusion

Our investigation of DLS and RLS provides interesting insights into the effectiveness of including semi-formal ML system diagrams alongside source code. Our analysis in DLS shows significant differences in task *correctness*, indicating that semi-formal ML system diagrams, together with the source code, contribute to better results.

Concerning RLS, our investigation revealed that *Control* shows comparable or slightly superior performance to *Experimental* on the *correctness* task, contrary to expectations. We found no significant difference in the *duration* function between *Control* and *Experimental*. In this case, *Experimental* participants faced challenges in understanding the complex details in the diagrams and the complexity of multiple diagrams. In contrast, participants in *Control* are required to handle only reasonably sized source code and, therefore, show better understanding. Additionally, we found a positive correlation between *correctness* and *duration* within *Experimental*, indicating a different approach to answering questions between participants.

In addition, examining participants' survey responses yielded some interesting findings. In DLS, participants who did not receive semi-formal ML system diagrams tended to estimate their performance slightly more accurately, whereas, in RLS, participants in RLS who did not receive these diagrams underestimated their performance. This outcome highlights the impact of additional materials on participants' confidence levels.

In conclusion, while providing semi-formal ML system diagrams appears beneficial in certain contexts, their effectiveness varies across different learning environments. The choice to incorporate these diagrams should be made with careful consideration of the specific objectives of practitioners.

CRediT authorship contribution statement

Evangelos Ntentos: Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Formal analysis, Data curation, Conceptualization. **Stephen John Warnett:** Writing – review & editing, Investigation, Formal analysis. **Uwe Zdun:** Writing – review & editing, Supervision, Methodology, Formal analysis.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Evangelos Ntentos reports was provided by University of Vienna. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the FFG (Austrian Research Promotion Agency) project MODIS (no. FO999895431).

Data availability

No data was used for the research described in the article.

References

Agrawal, A., Reddy, S., Bhattamishra, S., Nookala, V.P.S., Vashishth, V., Rong, K., Tumanov, A., 2023. DynaQuant: Compressing deep learning training checkpoints via dynamic quantization. arXiv:2306.11800.

Allodi, L., Biagioni, S., Crispo, B., Labunets, K., Massacci, F., Santos, W., 2017. Estimating the assessment difficulty of CVSS environmental metrics: An experiment. In: Dang, T.K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., Neuhold, E.J. (Eds.), Future Data and Security Engineering. Springer International Publishing, Cham, pp. 23–39.

Allodi, L., Cremonini, M., Massacci, F., Shim, W., 2020. Measuring the accuracy of software vulnerability assessments: experiments with students and professionals. Empir. Softw. Eng. 25, http://dx.doi.org/10.1007/s10664-019-09797-4.

Anderson, T.W., Darling, D.A., 1954. A test of goodness of fit. J. Amer. Statist. Assoc. 49 (268), 765–769. http://dx.doi.org/10.1080/01621459.1954.10501232, URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1954.10501232, arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1954.10501232.

Benjamini, Y., Hochberg, Y., 1995. Controlling the false discovery rate - a practical and powerful approach to multiple testing. J. R. Stat. Soc. Ser. B 57, 289–300. http://dx.doi.org/10.2307/2346101.

Bonferroni, C., 1936. Teoria statistica delle classi e calcolo delle probabilita. Pubbl. Del R Ist. Super. di Sci. Econ. E Commer. di Firenze 8, 3–62.

Bray, J.H., Maxwell, S.E., 1982. Analyzing and interpreting significant MANOVAs. Rev. Educ. Res. 52 (3), 340–367, URL: http://www.jstor.org/stable/1170422.

Charness, G., Gneezy, U., Kuhn, M.A., 2012. Experimental methods: Between-subject and within-subject design. J. Econ. Behav. Organ. 81 (1), 1–8.

Chen, Y., Liu, Z., Ren, B., Jin, X., 2020. On efficient constructions of checkpoints. CoRR arXiv:2009.13003.

Chen, S., Yuan, G., Cheng, X., Gong, Y., Qin, M., Wang, Y., Huang, X., 2023. Self-ensemble protection: Training checkpoints are good data protectors. arXiv:2211.12005.

Cliff, N., 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. Psychol. Bull. 114, 494–509.

Cliff, N., 2010. Answering ordinal questions with ordinal data using ordinal statistics. Multivar. Behav. Res. 31, 331–350. http://dx.doi.org/10.1207/s15327906mbr3103_4.

Czepa, C., Zdun, U., 2019. How understandable are pattern-based behavioral constraints for novice software designers? ACM Trans. Softw. Eng. Methodol. (TOSEM) 28, 1–38.

Czepa, C., Zdun, U., 2020. On the understandability of temporal properties formalized in linear temporal logic, property specification patterns and event processing language. IEEE Trans. Softw. Eng. 46, 100–112.

Delaney, H.D., Vargha, A., 2002. Comparing several robust tests of stochastic equality with ordinally scaled variables and small to moderate sized samples. Psychol. Methods 7 4, 485–503.

Dunn, O.J., 1961. Multiple comparisons among means. J. Amer. Statist. Assoc. 56, 52–64.

Eisenman, A., Matam, K.K., Ingram, S., Mudigere, D., Krishnamoorthi, R., Annavaram, M., Nair, K., Smelyanskiy, M., 2020. Check-n-run: A checkpointing system for training recommendation models. CoRR arXiv:2010.08679.

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M., 2018. Empirical software engineering experts on the use of students and professionals in experiments. Empir. Softw. Eng. 23, http://dx.doi.org/10.1007/s10664-017-9523-3.

Farahani, A., Pourshojae, B., Rasheed, K., Arabnia, H.R., 2021. A concise review of transfer learning. arXiv:2104.02144.

Heijstek, W., Kühne, T., Chaudron, M.R.V., 2011. Experimental analysis of textual and graphical representations for software architecture design. In: 2011 International Symposium on Empirical Software Engineeringand Measurement. pp. 167–176.

Hoisl, B., Sobernig, S., Strembeck, M., 2014. Comparing three notations for defining scenario-based model tests: A controlled experiment. In: Proceedings- 2014 9th International Conference on the Quality of Information and Communications Technology. QUATIC 2014, pp. 95–104. http://dx.doi.org/10.1109/QUATIC.2014.19.

Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., Azim, M.A., 2022. Transfer learning: a friendly introduction. J. Big Data 9 (1), 102.

Höst, M., Regnell, B., Wohlin, C., 2000. Using students as subjects - a comparative study of students and professionals in lead-time impact assessment. Empir. Softw. Eng. 5, 201–214. http://dx.doi.org/10.1023/A:1026586415054.

Hsu, L.M., 2004. Biases of success rate differences shown in binomial effect size displays. Psychol. Methods 9 2, 183–197.

Islam, T., Abid, D.M.H., Rahman, T., Zaman, Z., Mia, K., Hossain, R., 2023. Transfer learning in deep reinforcement learning. In: Yang, X.-S., Sherratt, S., Dey, N., Joshi, A. (Eds.), Proceedings of Seventh International Congress on Information and Communication Technology. Springer Nature Singapore, Singapore, pp. 145–153.

Jedlitschka, A., Ciolkowski, M., Pfahl, D., 2008. Reporting experiments in software engineering. Empir. Softw. Eng. - ESE 201–228. http://dx.doi.org/10.1007/978-1-84800-044-5_8.

Juristo, N., Moreno, A., 2001. Basics of Software Engineering Experimentation. http://dx.doi.org/10.1007/978-1-4757-3304-4.

Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., Pohthong, A., 2016. Robust statistical methods for empirical software engineering. Empir. Softw. Eng. 22 (2), 579–630, URL: http://dro.dur.ac.uk/18658/.

Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., Emam, K., Rosenberg, J., 2002. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Softw. Eng. 28, 721–734. http://dx.doi.org/10.1109/TSE.2002.1027796.

Kolmogorov, A.N., 1933. Sulla determinazione empirica di una legge didistribuzione. Giorn Dell' Inst Ital Degli Att 4, 89–91.

Labunets, K., Paci, F., Massacci, F., Ruprai, R., 2014. An experiment on comparing textual vs. Visual industrial methods for security risk assessment. In: 2014 IEEE 4th International Workshop on Empirical Requirements Engineering EmpiRE 2014 - Proceedings. http://dx.doi.org/10.1109/EmpiRE.2014.6890113.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Lilliefors, H.W., 1967. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. J. Amer. Statist. Assoc. 62, 399–402.

Mohd Razali, N., Yap, B., 2011. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. J. Stat. Model. Anal. 2.

Paulweber, P., Simhandl, G., Zdun, U., 2021. On the understandability of language constructs to structure the state and behavior in abstract state machine specifications: A controlled experiment. J. Syst. Softw. 178, 110987.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.-L., Chen, S.-C., Iyengar, S.S., 2018. A survey on deep learning: Algorithms, techniques, and applications. ACM Comput. Surv. 51 (5), 1–36.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv:1910.10683.

Runeson, P., 2003. Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data. In: Proceedings 7th International Conference on Empirical Assessment & Evaluation in Software Engineering. p. 95–102.

Salman, I., Misirli, A.T., Juzgado, N.J., 2015. Are students representatives of professionals in software engineering experiments? In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. vol. 1, pp. 666–676.

Shapiro, S.S., Wilk, M.B., 1965. An analysis of variance test for normality (complete samples). Biometrika 52, 591–611.

Siegmund, J., Kästner, C., Apel, S., Liebig, J., Schulze, M., Dachselt, R., Papendieck, M., Leich, T., Saake, G., 2012. Do background colors improve program comprehension in the #ifdef hell? Empir. Softw. Eng. 18, 1–47. http://dx.doi.org/10.1007/s10664-012-9208-x.

Svahnberg, M., Aurum, A., Wohlin, C., 2008. Using students as subjects - an empirical evaluation. In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM '08, Association for Computing Machinery, New York, NY, USA, pp. 288–290, URL: https://doi.org/10.1145/1414004.1414055.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning. In: International Conference on Artificial Neural Networks. Springer, pp. 270–279.

Valliappa Lakshmanan, M.M., 2021. Machine Learning Design Patterns. O'Reilly.

Warnett, S.J., Zdun, U., 2024. On the understandability of MLOps system architectures. IEEE Trans. Softw. Eng. http://dx.doi.org/10.1109/TSE.2024.3367488.

Winder, P., 2021. Reinforcement Learning, Industrial Applications of Intelligent Agents. O'Reilly.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer, Germany, http://dx.doi.org/10.1007/978-3-642-29044-2.

Zdun, U., Navarro, E., Leymann, F., 2017. Ensuring and assessing architecture conformance to microservice decomposition patterns. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (Eds.), Service-Oriented Computing. Springer International Publishing, Cham, pp. 411–429.

Zhu, Z., Lin, K., Jain, A.K., Zhou, J., 2023. Transfer learning in deep reinforcement learning: A survey. IEEE Trans. Pattern Anal. Mach. Intell. 45 (11), 13344–13362. http://dx.doi.org/10.1109/TPAMI.2023.3292075.