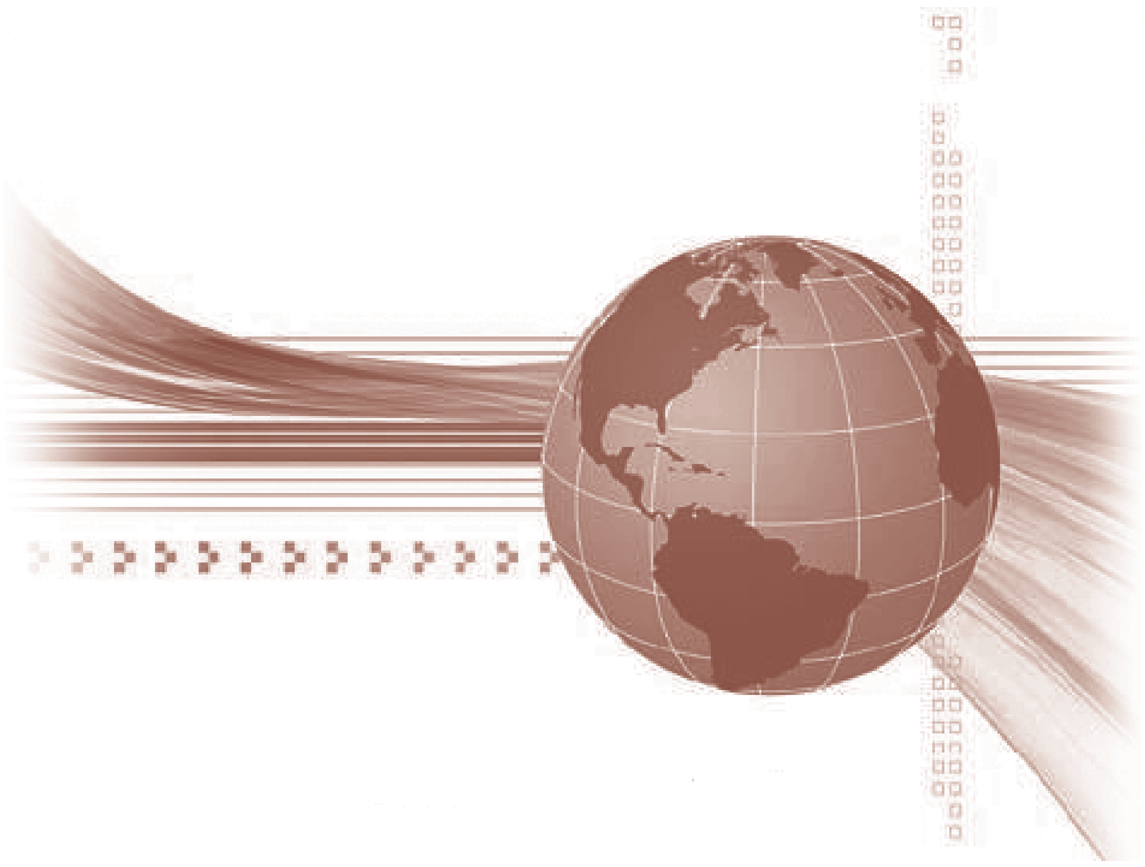
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI

UNIVERSITATEA BABEŞ-BOLYAI
BABEŞ-BOLYAI TUDOMÁNYEGYETEM
BABEŞ-BOLYAI UNIVERSITÄT
BABEŞ-BOLYAI UNIVERSITY
TRADITIO ET EXCELLENTIA



INFORMATICA

2/2024

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 2/2024

July - December

ISSN (online): 2065-9601; ISSN-L: 1224-869X

Published by Babeș-Bolyai University

YEAR	**Volume 69 (LXIX) 2024**
MONTH	**DECEMBER**
ISSUE	**2**

S T U D I A
UNIVERSITATIS BABEȘ-BOLYAI
INFORMATICA

2

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 2, 2024
DOI: 10.24193/subbi.2024.2.01

PYDSBUILDER – A DATASET BUILDER WRITTEN IN PYTHON DJANGO

LIVIU-MARIAN BERCIU

Abstract. Data mining and the analysis of open-source projects have become crucial in recent research, driven by the vast availability of data across multiple programming domains. This paper focuses on two main objectives: first, to present an experience report for designing a software quality data mining tool, and secondly, to provide an open-source solution, PyDs, that facilitates the creation of datasets specifically aimed at analyzing software quality attributes. PyDs, leveraging Python and the Django Framework, provides a comprehensive solution for researchers, encompassing data extraction from repositories, the application of software analysis tools, and the consolidation of results into a coherent format conducive to in-depth experimentation and analysis. This tool addresses the pressing need for effective data mining capabilities in evaluating software quality, allowing the research community to harness the full potential of the vast resources offered by open-source software projects.

1. Introduction

Open-source software development has seen a constant increase in popularity and adoption [17] in both industry and academia in the recent years. Open-source software (OSS) projects are software initiatives made freely accessible by their creators on various online platforms, such as GitHub and Bitbucket. These projects invite a broad audience to utilize the software, adhering solely to the terms of the associated open-source license. The widespread accessibility of these data, which span numerous programming languages, technologies, frameworks, and innovative solutions in various programming subdomains,

Received by the editors: 13 September 2024.

2010 *Mathematics Subject Classification.* 68N99.

1998 *CR Categories and Descriptors.* D2.0 [**Software Engineering**]: General – *Standards*; D2.9 [**Software Engineering**]: Management – *Software Quality Assurance*.

Key words and phrases. Data Mining, Software Quality Analysis Tools, Software Quality, Datasets, Dataset Builder, GitHub Mining.

has significantly propelled research efforts [13]. The integration of versioning systems plays a crucial role in these projects, as they not only facilitate collaborative development but also preserve a comprehensive history of the project's evolution. These historical data are invaluable, providing insight into development practices, trends, and the progression of software solutions over time, thereby enriching the research landscape with a deeper understanding of software development dynamics.

One topic that researchers have focused on when it comes to software projects is the assessment of software quality. Software quality is a multifaceted concept that refers to the degree to which a software product meets specified requirements, customer expectations, and user needs in a reliable, efficient, and maintainable manner. It encompasses various attributes such as maintainability, reliability, and security. The large number of OSS projects consisting of source code, issue tracking systems, and, more often than not, the competition brought about by projects tackling the same software domain has allowed researchers to review software quality in a transparent way [25].

Evaluation of software quality when it comes to open source data requires high volumes of information to be extracted, often due to the constant comparison between quality factors among a multitude of projects. Here, the concept of data mining comes into play [4], which implies tools and solutions that allow researchers to extract experiment data in a format that facilitates the execution of experiments and the drawing of conclusions about the problem studied. In the software engineering domain, examples of experiments include analyzing the technical debt of multiple Java projects [15] and studying maintainability when it comes to the long-term evolution of software projects [20]. When it comes to applying artificial intelligence to software engineering, more often than not, experiments require large amounts of data to be mined [26]. The mining of Github commit messages for natural language processing algorithms [12] and learning from the structure and history of the source code to automate the detection and fixing of bugs [8] are other applications where data mining is valuable.

Data mining often involves custom-created software solutions that facilitate data extraction from the internet. Some projects focus on the extraction and visualization of issue tracker data, such as [11] and [9], while others focus on offering a wider range of data extraction methods, focusing also on source code, commits and diffs, such as [24].

This paper outlines an experience report for the design and implementation of a data-mining tool specifically created for extracting datasets from software versioning systems, such as GitHub. The primary objective of this work is to present a detailed design proposal for a mining software repository tool and to provide an open source artifact called PyDs Builder, drawing from our hands-on experience in developing a solution tailored to our research needs. In addition, we provide an experiment scenario to demonstrate the application and effectiveness of PyDs. We also provide suggestions on the application of the tool in empirical research studies. In developing this tool, we reference established methodologies for data extraction and analysis, ensuring a rigorous approach to our development process.

The insights gained from this project are diverse and offer valuable lessons on the challenges and strategies involved in designing efficient and scalable data mining tools. These lessons could serve as an important resource for researchers looking to develop or enhance their own tools, providing a practical foundation from which to approach similar projects.

The overview of the tool on the side of the article is divided into two main parts. The first one implies extracting issue tracking data and overall data pertaining to a software project. Currently, the tool only supports Github as a versioning system. The second one implies running software quality tools such as SonarQube[23] and SZZ [22] on the extracted data, in order to further refine the dataset and offer insights into the software quality attributes of a specific project.

The paper is structured as follows: in Section 2 we will outline an overview of a subset of the current data mining tools in academia. Section 3 discusses the conceptual design of the solution. In sections 4, 5 and 6, we will go through the architecture, database optimization, and usage of the application, reflecting along the way the design choices matching the concept. Section 7 offers a final overview of the challenges and experiences received while implementing the solution. Sections 8 and 9 will underline future work, possible extensions of the application, and concluding remarks.

2. Related Work

There has been several contributions addressing tools to mine software repository and we intend to present those that we believe are of interest in our approach.

PyDriller [24] is a software engineering tool created to help developers mine Git repositories. Its main features include extracting the repository's source code, differences, commits, and modifications. It is a framework capable of manipulating data and exporting them in the right format. The authors also focused on creating a performant tool, allowing for fast-onboarding and easy usage by developers. In opposition to our tool, PyDriller does not provide a capability of saving the parsed data in a database for persistence, nor does it allow extracting issue tracking data from a specific versioning system such as

GitHub. It does, however, work with any git repository, as it utilizes the git diff feature in order to parse repository data, making it, as stated beforehand, a fast program.

GrumPy [11] is a Python and Django Framework [5] developed web-tool with the purpose of mining issue data from issue trackers. With a focus on GitHub as its main issue tracker of choice, the tool offers database management capabilities without actually having database knowledge, it allows researchers to download repositories issues data in parallel, using multiple task queues and also provides access to data visualization features and statistical analysis of the mined data. There are some similarities and differences between GrumPy and our tool. On the one hand, both tools are implemented using Python and Django. They use custom databases, different from the SQLite default that Django comes with (MongoDB for GrumPy, PostgreSQL for our tool), and the same is done for the queue management technology (Redis, as opposed to RabbitMQ). Both tools also offer issue data mining using the same Github API technology. On the other hand, GrumPy offers a visual overview on the mined issues through a web platform, also targeting people without prior knowledge of programming, while our tool is more technical, offering just the Django admin panel for database visualization. In contrast, we mine more Github information about repositories, such as Commits and Issue timelines, while also allowing for custom tools execution in order to properly build a dataset.

GHTorrent is another project that retrieves data from Github repositories [10]. With this tool, the authors aim to provide persistent data and event streams to the research community, as a service [10]. Data retrieval from Github is done using a specifically implemented crawler, which queries for raw data using the Github API. The extracted data are then sent to a set of RabbitMQ queues, which further refine the data. It is important to note here that this mechanism allows replication on multiple hosts, circumventing the API limits by using different API tokens from multiple research teams. Data persistence is done using MongoDB, due to the database technology's capability of scaling and handling of large amounts of data. While the GhTorrent solution allows for high amounts of data to be processed and put to the community service, it does not process Github issues and also covers a wide view of Github repositories, making it hard to target a specific niche. Instead, our tool provides researchers with a solution in building their own dataset for their specific needs, using a similar host distribution approach in order to allow higher API limit thresholds and also supports tool execution, in order to further enhance and interpret the data extracted.

Lastly, Perceval [7] is a command line tool that supports a multitude of data sources to retrieve data from, such as mailing lists, version control systems, ticketing tools, and Q/A solutions. It comes as either a Python library or as a command line tool, allowing for flexible usage. It is composed of multiple back-end implementations targeting different data sources, with the possibility of extension to support new entries, abiding by the user's needs. While both our tool and Perceval strive to offer data extraction and easy development extension to researchers, there are two main differences to take into consideration. Perceval [7] offers JSON-format data dumps, leaving the user to carry the data persistence responsibility. We, on the other hand, use PostgreSQL in order to save data directly for later use. The second difference is that there is no analysis tools support on the raw data, leaving users to implement/use their own data analysis pipelines. In our case, we support three out-of-the-box tools for data analysis, allowing researchers to easily automate flows and use the solution for an end-to-end dataset generation flow.

More often than not, tools created for data mining depend largely on research purposes, and in many cases, they are created to address the specific needs of a scientific experiment. For this reason, existing solutions are not always enough, as they often have to aggregate information from different tools, making most of the tools rigid and hard to extend for research purposes other than the ones they were built for.

3. Tool design

The requirement of extracting data from different sources in a consistent and organized manner, which can be utilized for various experiments and easily expanded, often leads to the need to create a software solution that encompasses these aspects. When working with large amounts of data, the need to structure and normalize the data is paramount and often involves employing different algorithms that scan, extract and aggregate the needed information in a form that facilitates processing. In order to do this, typically different software tools are used and then their results aggregated in one form or another, necessitating more effort from the researchers. The purpose of this article is outlining the experience and actually creating a solution that can aggregate data from different streams, under different formats, in one, uniform, and general format that can be used for creative experiments and extended as the researchers see fit.

An important aspect in creating a solution like this is **to decide what data formats are supported**. Usually, data are extracted using API requests that are provided by the data sources, or downloaded directly under the form of files. Data thus often appears in JSON, CSV, YAMl or SQL formats, among many others, and are then processed into a single, uniform standard. For our case, we see the need of supporting data conversion from the different formats enumerated beforehand into a standardized format, such as SQL, that enables researchers to conduct complex queries and analyses, and can also scale as the amount of data increases.

Another important aspect is offering a way in running data extraction for large periods of time without the constant supervision of the person using the solution. This is done by implementing automation, under the form of task queues, that allows cloud deployment and a clear set of instructions on how to ensure data is processed continuously. Furthermore, the solution should be implemented in a programming language that is popular in the programming community, has a low learning curve and offers many out-of-the-box features that developers can use, ensuring quick adoption and extension. Thus, another important issue to follow is **automation of data extraction**.

Obtaining the data in the desired format should allow further processing by feeding them to a data pipe of custom tools, each with its own purpose and end results, suiting the specific needs of the researcher. The data obtained from the execution of the tools will then be saved in the same uniform structure decided beforehand, leaving the decision of further processing or concluding the experiment in the hands of the user. In conclusion, if data from several tools are needed, **the decision about flow is important**.

4. PYDS BUILDER SOLUTION

In the following sections, we introduce the specifics and implementation of a data mining tool with a focus on the three important guidelines underlined above: **automation of data, format specification** and **flow decision**.

PyDs Builder is a web, API-based solution that aims to offer a way for researchers to create experimental datasets. It is built using Python and Django Framework, leveraging the capabilities offered by both technologies, such as fast development, scalability, excellent documentation, and an ORM system allowing intuitive database manipulation.

Its main focus is extracting repository data from versioning systems, with the incipient implementation offering support for Github. Data is processed into the desired form and inserted into a custom database, following a **pre-established SQL schema**. Data can be processed further by custom tools in order to complete an experiment's data acquisition goals. Afterwards, the data can be used as the researchers see fit, either by publishing a totally new dataset or feeding the data into an artificial intelligence solution, drawing new conclusions and desired results.

An overall overview of the solution features is enumerated as follows:

- Allow tool interaction through an API interface
- Extract Github repository data such as issues, issue timelines, commits details and source code
- Execute software quality tools such as SonarQube [23], SZZ [22] and PyRef [3] and ensure data persistence
- Provide automation for fast data processing
- Allow contribution and code extension through project modularity and intuitive programming interface.

4.1. **ARCHITECTURE.** From an architectural point of view, a modular monolith approach was used in order to build the application. This approach was taken due to a few considerations. First, Django is a Python framework that is designed as a monolith by default. It consists in a single code base, a shared database and a single deployment. Second, Django has a native application support, meaning it can be designed into modules such that a single module can hold a single responsibility. From those two points came the third, which implied that, by using modularization, we managed to create separate code units for each tool that we support, setting boundaries so that the shared code is held in common modules and tools don't have to interfere with each other. This further enriches the mission of allowing developers to contribute to the PyDs solution by simply using the common modules already defined to create a new module for their own specific needs.

Next, to enable automation, RabbitMQ [21] and Celery were used in order to setup task queues. The task queue functionality provided by Celery allows the application to perform asynchronous work in the background, while RabbitMQ is the message broker that Celery uses in order to exchange messages and run tasks. In this way, the application can be started, for example, on a virtual machine in the cloud and perform work without constant supervision from the user. Furthermore, it bypasses the HTTP request limit, allowing a task to take from a few minutes to a few hours, depending on the needs, without the risk of timeouts. The same rationale can be replicated on multiple instances of virtual machines, allowing parallel execution and data extraction from multiple sources at the same time. With this, we have covered the point about **setting up an automation mechanism for data extraction**.

Data persistence was managed using PostgreSQL as the database engine of choice. The reason this database technology was used, as opposed to using the default SQLite Django database, was the following: it is highly scalable, handling large volumes of data and concurrent users efficiently, it can be hosted in the cloud, it supports JSON data types, and it has a robust security. In contrast, SQLite is a self-contained system that has no server setup, has limited scalability and concurrency, and lacks advanced features found in more complex RDBMS (Relational Database Management System/s), such as ACID transactions, complex query support and concurrency control.

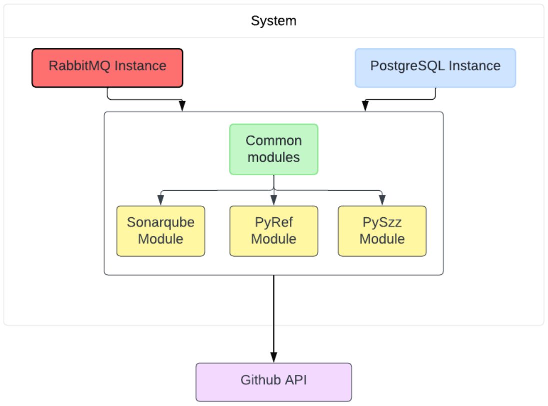


FIGURE 1. PyDs Architecture Diagram

Figure 1 represents a visual overview of the architecture of the system. It can be observed how the main system, comprised of common modules and the specialized tools module, communicates with the internal modules of the database instance and the task queue instance, and the external connection to the Github API service. Common modules implement command-line functionality and a wrapper over the Github API functionality. The modules of specialized tools implement wrappers over SonarQube [23], PyRef [3] and PySzz [22].

4.2. TOOLS AND FLOWS.

The solution code base includes support for three software quality tools. The tools were chosen to serve the research objectives and because of their relevance in the software quality research space which provides ease of use for researchers to build datasets with the results of their execution. The remainder of this section follows a short tool introduction and the steps necessary for researchers to run the tools and extract data using our solution. More details about tool usage can be found in Section 6.

In the context of enhancing software refactorings, PyRef [3] emerges as a dedicated tool optimized for projects developed in Python. This tool conducts a comparative analysis between two versions of a project to accurately identify the refactorings that have occurred. PyRef is specifically engineered to detect a suite of nine method-level refactoring operations, which include: Rename Method, Add Parameter, Remove Parameter, Change/Rename Parameter, Extract Method, Inline Method, Move Method, Pull Up Method, and Push Down Method. PyRef's ability to systematically identify and categorize method-level refactorings enhances its usefulness in gaining a more comprehensive understanding of software evolution and maintenance practices in Python-based projects.

In order to execute PyRef on a project, the PyRef repository must be cloned in the solution root folder. Afterwards, the first thing to do is initialize a repository in the system by calling the *Create repository* API url. The last step is calling the PyRef API url specifying the repository and the commit hash to be analysed. In case a commit hash is not provided, the latest release of the repository, if any, is fetched from the database. It is important to note that the tool will call the command 'git rev-list ¡commit¿' and will compare all pairs of commits that the command returns. The excution results will be saved into the database.

The SZZ algorithm is used in software engineering to automatically identify bug-introducing commits in version control systems. It operates by tracing back from bug-fixing commits to the original commits where the bugs were introduced, using the version history of a software project. An open source implementation of the SZZ algorithm is PySZZ [22], a tool which we selected based on its Python implementation and command line execution capabilities, allowing quick integration with our tool.

For obtaining PySZZ data, the main two steps are preparing the input data for the tool and executing the tool on the input data. The configuration for SZZ is found in chapter 6. Calling the *create_input_file* and *execute* API endpoints on the desired repository will create the input file, will execute the tool on the input file and then, calling the *extract* endpoint, will save the result into the database.

SonarQube is a static code analysis tool designed to enhance software quality standards [23]. It seamlessly integrates into the development workflow, offering multi-language support for static analysis rules and classifying code based on the software quality dimensions of reliability, security, and maintainability. Since its inception in 2008, SonarQube has evolved significantly, as evidenced by its frequent updates and the scholarly attention it has received, including discussions in various scientific articles [15], [14] [19]. A noteworthy development in its evolution is the shift from traditional issue classifications such as bugs, vulnerabilities, and code smells towards the adoption of "Clean Code" principles. These principles are further delineated into categories such as consistency, intentionality, adaptability, and responsibility, each defining specific attributes of code quality. In order to execute the SonarQube flow, users have to configure SonarQube on their work stations. This can be done either by following the installation steps from SonarQube's official documentation or by creating a docker container to hold the service. Subsequently, making sure that the repository was already initialized in the database, the endpoint *sonarQube/analyze* can be called. It is a POST method receiving a body containing the repository owner, repository name, release tag, or commit hash. It will execute SonarQube using the received information and save all the SQ issues and SQ measures found.

We can conclude this section by reiterating the importance of **deciding about a clear flow of data extraction** when it comes to integrating a tool. From setting the incipient data such as the repository to be analyzed, to writing the wrapping code over the tool interface, whether it is command-line or web-based, to finally extracting and processing data in an automated manner, each step has to be properly implemented and executed in the correct order so that data acquisition is successful.

More endpoints are available in the project codebase due to various experiments. They are left there for researchers to explore and use them as they see fit.

5. Database optimization

The codebase includes a database architecture and entities that were used in order to run different software engineering experiments that include data extraction and arrangement. Next, we will take a look at some database best practices and optimizations that can be done so that experiments run optimally and the dataset is arranged as needed.

- **Data Normalization:** Minimize redundancy, improve data integrity but balance to avoid overly complex queries. Try not to create cyclic dependencies between table and keep a tree structure. For example, the Repository can be the main table to which the other parts of the database connect, but the Repository will not reference any of it's dependents.
- **Indexing:** Accelerate record retrieval in frequently searched columns, balancing read performance with write overhead. The SQ Issue table, which can contain millions of results, can have an index on the *commit_hash* field.
- **Partitioning:** Divide large datasets into manageable segments for improved performance and easier management, tailored to query patterns. For example, tables for SonarQube and PyRef do not have any dependencies to eachother, being separated in their own semantic field, ensuring data integrity and proper separation.

- **Denormalization:** Introduce redundancy selectively to speed up read operations where beneficial, with careful consideration of trade-offs. Many-to-many tables can be added to avoid join links such as Repository - Issue - Timeline - Commit, or table fields that can reference the main entity (Commit references repository ID directly).
- **Concurrent Access and Locking Strategies:** Implement suitable locking mechanisms to maintain data integrity during simultaneous access, optimizing for the specific access pattern. Proper selection of the database engine ensures proper concurrent access, hence the choice of PostgreSQL over SQLite.
- **Efficient Query Design:** Craft queries to only fetch necessary data, using joins effectively, and optimize regularly based on usage.

6. Execution and Usage

6.1. **Installation.** There are a few steps that have to be completed in order for the tool to run successfully. The first step implies installing the project dependencies using Python's pip command. Afterwards, docker-compose [6] must be used in order to start the containers necessary for the queue orchestrator, the database and tools such as SonarQube. The tool is then started by running the default Django command for starting a server. An important part after starting the server is to run the database migrations in order to setup the correct database schema to use. The exact steps for installation are enumerated in Listing 1.

```
# Install the requirements
pip install -r requirements.txt
# Start the docker containers
docker-compose up -d
# Start the server
python3 manage.py runserver
# In another terminal, run the
# migrations for the database
python3 manage.py migrate
```

Listing 1. PyDs Setup Steps

6.2. **Configuration.** The tool configuration is done inside the main Django configuration file, namely ‘settings.py‘ found in the main application folder. The settings file contains general information about configuring a Django project, such as the logging level, database connection credentials, installed applications, middlewares, celery queues and custom variable defined by the user. Although the public repository will contain a pre-completed configuration file with examples for values, the code from Listing 2 exemplifies some of the important configuration variables and their meanings.

```
#  SonarQube
SONARQUBE_URL = < SonarQube URL to call>
SONARQUBE_TOKEN = <project analysis tokens>
SONARQUBE_GLOBAL_TOKEN = <>
SONARQUBE_USER_TOKEN = <>
SONARQUBE_SCANNER_URL = <URL of the Sonar scanner if not
     installed locally>
SONARQUBE_PROJECT_KEY = <specific project key to scan>
SONARQUBE_USERNAME = <login username>
SONARQUBE_PASSWORD = <login password>

# Github
GITHUB_TOKENS = [<list of github authentication tokens
    for API calls]
GITHUB_ROOT_DIR = <root directory for github projects
    cloning>

# PySZZ
SZZ_INPUT_FILES_FOLDER = <location of files pre-prepared
     for PySZZ execution
SZZ_OUTPUT_FILES_FOLDER = <location of files after PySZZ
     execution>
SZZ_GITHUB_TOKEN = <specific github token to run only
    with PySZZ>
```

Listing 2. Configuration Variables Example

6.3. **Usage.** The basic usage of the application is done through the REST API exposed through Django [5] views. Django has a MVT (Model View Template) architecture, allowing developers to write API endpoints in specialized *VIEW* classes. A subset of the available API calls is found in Table 1. One important note is that, for mining Github data, we have used a similar endpoint format as in the official Github API documentation, in order to offer familiarity for users who have prior experience with the Github API. For exemplification purposes, we used the Ansible [2] and Pandas [16] repositories, as they are some of the largest Python Github open source projects.

6.4. **Data visualisation.** There are two options available for visualizing the extracted data. The first option is to utilize a specialized SQL visualization tool that enables the examination of the database, execution of queries, and visualization of the overall database structure. Alternatively, the Django admin panel can be used to gain insights directly into the selected tables included in the admin dashboard. Another way to view the data is by implementing fetch requests in the API views of the tool.

7. EXPERIENCE REPORT

Developing PyDs from the ground up inevitably came with its own unique set of obstacles. The following paragraphs outline the challenges encountered during the development process of PyDs.

- We have made the decision to use SQL for data serialization and database schema, instead of opting for the more direct JSON and/or CSV formats that are commonly used for raw data. Although the SQL approach may be more complex, we believe that the long-term benefits, such as optimization and a rich feature set, outweigh the disadvantages.
- The objective was to develop a universal approach for running external tools. Since each tool has its own specific set of instructions for execution, we were able to devise a general method by utilizing the command line capabilities and creating wrapper classes and modules for each individual tool.
- The selection of an appropriate database for automation is crucial. Initially, SQLite was utilized as the preferred database option. However, it was soon realized that SQLite has limitations in terms of capabilities and is not suitable for distributing the workload across multiple machines. As a result, PostgreSQL was chosen as it possesses the necessary capabilities and is compatible with cloud hosting.
- The limitation of data intake was also influenced by the rate limits imposed by open source project platforms. To address this challenge, we developed a wrapper class that can utilize access tokens from multiple researchers. This allows for continuous data retrieval, as when one token reaches its rate limit, the next token in the queue is automatically used.

Purpose/Meaning	**API Call**	**Method**
Create a repository database entry	`/mining/repo/github/pandas-dev/pandas`	POST
Delete a repository from the database	`/mining/repo/ansible/ansible`	DELETE
Fetch all issues for a repository	`/mining/repo/github/ansible/ansible/issues`	POST
Fetch a specific issue for a repository	`/mining/repo/github/ansible/ansible/issues/123`	GET
Extract timelines for all already extracted project issues	`/mining/repo/github/ansible/ansible/issues/timeline`	POST
Extract an issue's specific timeline	`/mining/repo/github/ansible/ansible/issues/4720/timeline`	GET
Run a SonarQube analysis for a specific Github issue	`/sonarqube/repo/github/pandas-dev/pandas/issue/36`	POST
Run a SonarQube analysis for a commit hash or release tag	`/sonarqube/analyze`	POST
Run PySZZ create input file	`/szz/repo/pandas-dev/pandas/create_input_file`	POST
Run PySZZ execute	`/szz/repo/pandas-dev/pandas/execute`	POST
Run PySZZ extract	`/szz/repo/pandas-dev/pandas/extract`	POST
Run PyRef on a repository	`/pyref/repo/ansible/ansible`	POST

TABLE 1. API Requests Overview

8. RESEARCH POSSIBILITIES AND EXTENSION

PyDs is a software solution that can be valuable both for researchers and independent developers. For researchers, it provides a working framework for running complex experiments in an automated way, ensuring data extraction for very large datasets. While initially created for software quality experiments, including running specialized software quality tools in order to uncover maintainability, technical debt and reliability attributes of software projects, it can be extended to allow artificial intelligence integration, it can support multiple database engines such as MongoDB and MySQL and it can be enriched to extract data from other versioning and issue tracking systems such as Bitbucket and Jira. Independent developers can greatly benefit from the PyDs dataset generation tool by gaining enhanced insights into their projects' maintainability and reliability, and by integrating analytics solutions on the extracted data to suit their specific needs. The tool's ability to integrate with various development tools and database engines streamlines the development workflow while offering opportunities for skill enhancement through interaction and extension of the tool. This makes PyDs a versatile and valuable resource for independent developers looking to innovate, improve project health, and efficiently manage their software development processes.

PyDs is also offered as an open source project, allowing community contributions and being subject of the FreeBSD license [1]. The source code can be found by accessing the following link: `https://figshare.com/s/5dd7e88ba4e329acfa4a`.

8.1. Possible scenario for tool usage.

In order to expose the features of the tool, we imagine a possible research scenario in which using PyDs Builder can be beneficial. The objective of this study is to monitor and improve software quality throughout the development lifecycle of a project. The aim is to track the project's quality trajectory, from commit to commit, and visualize the evolution of issues to make informed decisions for continuous quality improvement.

The methodology involves a streamlined process using the data mining tool.

- Selection of a Python Project: Choose a project with a sufficiently large code base.
- Data Extraction: Utilize PyDs Builder API to fetch project data and issues from GitHub.
- Quality Analysis: Analyze the project using SonarQube through the integration with PyDs Builder.
- Issue Prioritization and Resolution: Identify critical issues affecting quality and address them systematically.

- Quality Trajectory Assessment: Evaluate changes in quality metrics over time to gauge improvement or deterioration.

Upon analyzing the extracted data, notable trends in code quality metrics can be observed, particularly in the occurrence and distribution of specific types of issues across various developmental stages. Patterns between SonarQube-reported issues and GitHub issues can be associated, shedding light on the collaborative dynamics of the project contributors as they addressed quality concerns.

Using PyDs allows for tracking the quality trajectory of the project, identifying both problematic and beneficial commits and changes. This analysis not only identifies areas that need improvement, but also facilitates proactive interventions to increase overall project quality and stability. The insights gained provide a detailed view of the project's quality dynamics, highlighting both strengths and areas for improvement in software development practices.

9. CONCLUSIONS

Data mining and open source project analysis have been one of the important subjects of academia in recent years, with data availability comprising multiple programming domains being one of the main factors for its ascendance.

In this paper, we have introduced PyDs, a Python with Django Framework solution that enables researchers to generate datasets for various scientific criteria, primarily focusing on software quality attributes experiments. However, it also allows for potential expansion to apply artificial intelligence to software engineering. We have covered the conceptual design of the application, its underlying principles, and delved into the implementation steps and different perspectives. We have provided detailed instructions for setting up, configuring, and using the application to facilitate quick onboarding for readers. Finally, we have explored potential avenues for extending the application and shared our experience in developing this intricate tool.

The subsequent stages of the application involve its practical implementation in scientific settings and research projects. We are confident that PyDs can serve as a reliable solution in these scenarios, allowing researchers to utilize it for data extraction in popular software quality tools such as SonarQube and SZZ algorithm implementations. Currently, PyDs only supports Github, but there are future plans to expand its capabilities to integrate with the Jira ticketing system and extract CI/CD pipelines data from platforms like Jenkins, as we believe project building steps and performance indicators can provide valuable research data. In the end, PyDs Builder has been successfully utilized to create a dataset focused on open-source Python projects [18], highlighting its flexibility and practical utility.

References

[1] The freebsd license, 2023.

[2] Ansible, I., et al. Ansible: Radically simple IT automation. `https://github.com/ansible/ansible`, 2023.

[3] Atwi, H., Lin, B., Tsantalis, N., Kashiwa, Y., Kamei, Y., Ubayashi, N., Bavota, G., and Lanza, M. Pyref: Refactoring detection in python projects. In *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)* (2021), pp. 136–141.

[4] Chaturvedi, K., Sing, V., and Singh, P. Tools in mining software repositories. In *2013 13th International Conference on Computational Science and Its Applications* (2013), pp. 89–98.

[5] Django Software Foundation. Django.

[6] Docker, Inc. Docker: Empowering app development for developers, 2023. Accessed: 2024-02-17.

[7] Dueñas, S., Cosentino, V., Robles, G., and Gonzalez-Barahona, J. M. Perceval: software project data at your will. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings* (New York, NY, USA, 2018), ICSE '18, Association for Computing Machinery, p. 1–4.

[8] Elmishali, A., Stern, R., and Kalech, M. An artificial intelligence paradigm for troubleshooting software bugs. *Engineering Applications of Artificial Intelligence 69* (2018), 147–156.

[9] Fiechter, A., Minelli, R., Nagy, C., and Lanza, M. Visualizing github issues. In *2021 Working Conference on Software Visualization (VISSOFT)* (2021), pp. 155–159.

[10] Gousios, G., Vasilescu, B., Serebrenik, A., and Zaidman, A. Lean ghtorrent: Github data on demand. pp. 384–387.

[11] Jr., J. M., Santana, R., and Machado, I. Grumpy: an automated approach to simplify issue data analysis for newcomers. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering* (New York, NY, USA, 2021), SBES '21, Association for Computing Machinery, p. 33–38.

[12] Kourtzanidis, S., Chatzigeorgiou, A., and Ampatzoglou, A. Reposkillminer: identifying software expertise from github repositories using natural language processing. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering* (New York, NY, USA, 2021), ASE '20, Association for Computing Machinery, p. 1353–1357.

[13] Krogh, G. v., and Spaeth, S. The open source software phenomenon: Characteristics that promote research. *The Journal of Strategic Information Systems 16*, 3 (2007), 236–253.

[14] Lenarduzzi, V., Lomio, F., Taibi, D., and Huttunen, H. On the fault proneness of sonarqube technical debt violations: A comparison of eight machine learning techniques. *CoRR abs/1907.00376* (2019).

[15] Lenarduzzi, V., Saarimäki, N., and Taibi, D. The technical debt dataset. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering* (Sept. 2019), PROMISE'19, ACM.

[16] McKinney, W., et al. pandas: a powerful Python data analysis toolkit. https://github.com/pandas-dev/pandas, 2023.

[17] Midha, V., and Palvia, P. Factors affecting the success of open source software. *Journal of Systems and Software 85*, 4 (2012), 895–905.

[18] Moldovan, V.-A., Berciu, L.-M., and Patcas, R.-D. The python software quality dataset. In *50th Euromicro Conference Series on Software Engineering and Advanced Applications* (2024).

[19] Molnar, A.-J., and Motogna, S. Long-term evaluation of technical debt in open-source software. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (New York, NY, USA, 2020), ESEM '20, Association for Computing Machinery.

[20] Molnar, A.-J., and Motogna, S. A study of maintainability in evolving open-source software. In *Evaluation of Novel Approaches to Software Engineering* (Cham, 2021), R. Ali, H. Kaindl, and L. A. Maciaszek, Eds., Springer International Publishing, p. 261–282.

[21] RabbitMQ Team. Rabbitmq: Open source message broker. https://www.rabbitmq.com/, 2023. [Online; accessed 10-February-2024].

[22] Rosa, G., Pascarella, L., Scalabrino, S., Tufano, R., Bavota, G., Lanza, M., and Oliveto, R. A comprehensive evaluation of szz variants through a developer-informed oracle. *Journal of Systems and Software 202* (2023), 111729.

[23] SonarSource. Sonarqube: Continuous code quality inspection tool, 2023. [Online; accessed 10-February-2024].

[24] Spadini, D., Aniche, M., and Bacchelli, A. Pydriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (New York, NY, USA, 2018), ESEC/FSE 2018, Association for Computing Machinery, p. 908–911.

[25] Spinellis, D., Gousios, G., Karakoidas, V., Louridas, P., Adams, P. J., Samoladas, I., and Stamelos, I. Evaluating the quality of open source software. *Electronic Notes in Theoretical Computer Science 233* (2009), 5–28.

[26] Wangoo, D. P. Artificial intelligence techniques in software engineering for automated software reuse and design. In *2018 4th International Conference on Computing Communication and Automation (ICCCA)* (2018), pp. 1–4.

Babeș-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu, Cluj-Napoca 400084, Romania

Email address: liviu.berciu@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 2, 2024
DOI: 10.24193/subbi.2024.2.02

LANGDES: A NEW APPROACH FOR IMPROVING THE PERFORMANCE OF PROMPT-BASED IMAGE EDITING IN INTERIOR DESIGN SETTING

VICTOR-EUGEN ZARZU

Abstract. The topic of instruction-based image editing has gotten a lot of attention in recent years with a lot of research conducted due to its immense potential in various applications such as removing unwanted details present in existing images or improving them. However, one of the main problems in addressing this problem is acquiring a dataset for model training. Several methods and variations were proposed, but all of them rely on already-existent data. We propose a method to address this problem by creating a context-specific dataset for interior design with no previously available information by leveraging the knowledge of large language models (LLM). Furthermore, we test and prove the efficiency of the generated dataset on InstructPix2Pix which starts to compute better results for the interior-design setting after the fine-tuning. Moreover, we propose an alternative solution for enhancing the localization of the edit region through cross-attention map regularization based on a text-based segmentation mask.

1. Introduction

Prompt-based image editing is the problem of modifying an input image concerning a natural language edit prompt. Applications of this task consist of reducing the effort in professional image editing (e.g. removing a person from an image will transform into writing a phrase) and increasing the efficiency in graphics.

Received by the editors: 11 October 2024.

2010 *Mathematics Subject Classification.* 68T05, 68T45.

1998 *CR Categories and Descriptors.* I.2.6 [**Learning**]: Subtopic – *Connectionism and neural nets*; I.2.10 [**Vision and Scene Understanding**]: Subtopic – *3D/stereo scene analysis*.

Key words and phrases. Diffusion models, Prompt-based image editing, Deep learning, Attention, Data generation.

The major challenge of this problem is generating a dataset for training and evaluation. While each instance in the dataset needs to consist of an input image, an edit prompt, and the image resulting from the edit of the original one to the prompt, it is challenging to create such a dataset at scale because of the costs involved. Furthermore, reducing the unwanted modifications in the background and objects is also an important part of the problem, being intensely researched and correlated with the noise in the training dataset and the incapability of the model to map the edit instruction to the correct objects in the image.

The main focus of this paper is distributed among three aspects. The first part tries to answer whether a robust, high-quality, and context-specific dataset for the task in a discussion can be generated without previously available data. Secondly, it is proved that this dataset is qualitative and improves the IP2P model in the chosen context (interior design). Lastly, we aim to answer if a referring expression-based image segmentation with the object(s) under edit improves the performance of the InstructPix2Pix model in the general case through cross-attention map regularization.

In summary, we aim to answer the following research questions.

RQ1. *How to generate data for context-specific prompt-based image editing tasks with no previously known data?*

RQ2. *Does the generated data improve the performance of instruction-based image editing in the specific context?*

RQ3. *Does a referring expression-based image segmentation with the object under edits improve the performance of the InstructPix2Pix model in the general case through cross-attention map regularization?*

The rest of the article is structured as follows. Section 2 presents the previous work in the area of prompt-based image editing as well as text-based image segmentation. Afterward, the methodology that aims to respond to the addressed research questions is presented in Section 3. Along with discussions, we showcase the experimental results of the proposed approaches in Section 4, while the article ends with the conclusions and directions for future work in Section 5.

2. Related work

2.1. **Prompt-to-Prompt.** Introduced by Hertz et al. [11], Prompt-to-Prompt is an approach for generating two similar images based on two given prompts based on diffusion models. The method relies on the fact that the geometry and spatial layout of any generated image using text-guided diffusion models depend on the cross-attention maps. The approach generates the two images simultaneously, but while the generation of the first one is normal, in the second one, the cross-attention maps from the first generation are injected. This integration lasts for $\mathcal{T}$ timesteps and controls how similar the resulting images should be, its value variation depending on the area that has to be edited. However, while it tackles the problem of image editing, it is not able to edit an already-existent image, but it opens the possibility of generating a dataset for the prompt-based image editing task for supervised training.

2.2. **InstructPix2Pix.** IntructPix2Pix (IP2P) is an approach introduced by Brooks et al. [3] for editing an already-existent image based on a given prompt. It relies on using a Stable Diffusion [21] checkpoint, incorporating also the input image as conditioning and applying classifier-free guidance [12]. The guidance is done based on the initial image conditioning c_I and edit prompt conditioning c_T. Furthermore, two guidance scales are introduced, one for the image s_I and one for the text s_T, which can be adjusted to trade off the importance of each conditioning in the generated sample. The training is done in a supervised fashion on a dataset generated in two steps by leveraging the knowledge of GPT-3 [4] and Stable Diffusion (SD) [21] combined with the Prompt-to-Prompt approach. In the first step, GPT-3 is fine-tuned, given an initial caption from the LAION-Aesthetics V2 6.5+ [23], to output an edit prompt and the caption edited following this prompt. Secondly, based on the initial and edited caption, Prompt-to-Prompt is applied to generate 100 pairs of images followed by filtering using CLIP [20] to keep the most consistent pairs.

2.3. **Emu Edit.** Introduced by Sheynin et al. [24], Emu Edit addresses the issue of inaccurately interpreting and executing the edit instructions of InstructPix2Pix by being trained on a variety set of problems including classic computer vision and image editing tasks. The model architecture and training are similar, but the diffusion model used is Emu [6], and learned task embeddings are injected into the U-Net architecture to enhance the accuracy of the edit application. The data generation pipeline is also based on an LLM and Prompt-to-Prompt but followed by a more comprehensive filtering approach to reduce the noise and increase the consistency of the data. To generate the textual data, the approach proposes creating via in-context learning task-specific Llama2-70B [27] agents that, given an initial caption, are prompted to return an edit instruction and the final edited caption along with a list of the objects that are edited. The initial Prompt-to-Prompt solution is enhanced to reduce the noise of the original method by binary injecting the masks of the edited objects from the initial image during the editing process to increase image consistency.

2.4. **LIME: Localized Image Editing via Attention Regularization in Diffusion Models.** LIME is a solution for reducing the unwanted modification in the edited image through cross-attention map regularization proposed by Enis et al. [25]. The approach relies on the property that diffusion models can be used for text-based image segmentation tasks by leveraging their intermediate features as introduced by PNVR et al. [19]. With this, the method extracts the feature from various layers of the IP2P's U-Net architecture, followed by their fusing in three steps. Afterward, having a final attention map, the Region of Interest (RoI) is computed based on the top 100 pixels in probability followed by introducing all segments that overlap at least one of these pixels as well. Having the binary mask M computed, the method regularizes the attention scores (QK^T) within the RoI of the unrelated tokens to the edit denoted as S (e.g. <*start of text*>, *stop words*)) as in Formula (1), where α is a large value.

$$R(QK^T, M) = \begin{cases} QK^T_{ijt} - \alpha, & \text{if } M_{ij} = 1 \text{ and } t \in S \\ QK^T_{ijt}, & \text{otherwise} \end{cases} \tag{1}$$

2.5. **GRES.** Introduced by Liu et al. [14], **Generalized Referring Expression Segmentation** (GRES) is a new benchmark that addresses the limitations of the original task by allowing the segmentation of multiple objects within the same image and returning an empty mask if the referred object is not present. They also propose a model called ReLA, achieving state-of-the-art performance in the new GRES benchmark and the original RES one.

3. Methodology

Prompt-based image editing is the problem of computing the image resulting from the editing of an original image based on a given instruction. Existent methods deal with this problem in the general case with no special focus on the interior design setting, and this paper aims to improve the performance in such a setting.

This section introduces the proposed methodology for improving the text-based edit in the interior design setting as well as the edit localization based on a text-based segmentation mask through cross-attention map regularization, approaches that will facilitate the answering to the addressed research question. Firstly, we propose a pipeline for generating context-specific datasets with no previous data, focusing on the text and images in two different sections. Afterward, to show the efficiency of the created dataset, we fine-tune the base InstructPix2Pix model on it. Lastly, we propose a method for improving the edit localization based on ReLA's text-based segmentation mask through cross-attention map regularization.

3.1. **Dataset generation.** The problem of data scarcity is a recurrent problem in prompt-based image editing tasks because of the difficulty in collecting images before and after a specific edit instruction at scale. As stated before, the currently available proposal for acquiring a large dataset for this task was the generation of it. Still, all of them rely on an initial description of the original image. Of course, there is still the option of manually creating such a dataset, but it involves high costs and it is not scalable for large and high-performance models. For a context-specific case like interior design, this data is limited and not diverse enough for a good generalization of the problem.

So, we propose a method for creating such a dataset with no previous data to respond to research question RQ1 by creating a context-specific dataset in this fashion. The proposed approach is composed of the following two steps similar to previous approaches: the textual data generation followed by the creation of the paired images based on the text. Since the instruction-based image editing is treated as a supervised learning problem, each dataset instance will be made of an edit prompt and two images, the original one and the image modified concerning the given prompt.

3.1.1. *Text editing instructions generation.* The initial generated text samples will mimic the structure of the final dataset's instance, but the images are replaced with their textual descriptions. As such, each text instance will be made of three elements: (i) the description of the initial room or object, (ii) the edit instruction, and (iii) the initial description modified with respect to the editing instruction.

For addressing the absence of initial descriptions, GPT-4, the large language model used for experiments, was queried to generate all 3 components, compared to [3] where the last 2 components of the tuple are generated based on a previously known description. By leveraging the knowledge of the language model, there is no need to fine-tune it.

Knowing that different types of rooms usually have different objects that characterize them, room-specific agents can be created via in-context learning. With the proposed approach, by just presenting to the language model the format of the desired output (here JSON) and 3 other examples in that format, it is able to generate a large amount of data in the desired format with a great variety of responses. Moreover, to reduce the noise, GPT-4 [17] was instructed to clearly state that the object is missing or exists in the original description for the add and remove actions respectively. With this approach, 8,831 text samples were generated and published on HuggingFace [29]. Intuitively, this method can be used for any specific context by just providing the language model with examples from the targeted setting.

Additionally, the presented method has the advantage of enabling the creation of data in a hierarchical way of difficulty for the editing model: it first creates paired captions for single objects captions followed by the ones with a description of rooms with more objects. Compared to [3], the presented method can be extended and used for any other special case of prompt-based image editing, without the prior need for data, hence independence on the existent datasets.

Moreover, for the generation of the text captions and instruction GPT-3 was also used as an alternative before GPT-4 became publically available. While both of them produced overwhelming and diverse results, GPT-4 was more diverse in its outputs by different criteria computed using the site introduced by Runker [1] as seen in Table 1.

	MTLD [15] ↑	Dugast's U^2 [8] ↑	Guiraud's Index [7] ↑	Yule's K [9] ↓
GPT-3.5	28.13	12.83	3.87	356.49
GPT-4	**32.72**	**13.73**	**4.52**	**278.80**

Table 1. Comparison of diversity in the textual data generated by GPT models.

3.1.2. *Generating images from the paired editing instructions.* Starting from the paired editing instructions generated with the previous method, the Prompt-to-Prompt based on the Stable Diffusion model approach is used for generating the dataset samples in a supervised way: the image before and after the edit. However, generating one image for each instruction does not guarantee their consistency. For addressing this issue, similar to the approach presented in [3], a large number of image pairs is generated for each pair of captions with different values of p that controls the similarity between the images, followed by CLIP-based metric filtering introduced by Gal et al. [10]. Only the top four pairs of images that are above the image similarity threshold of 0.75 are kept.

Compared to the values used for InstructPix2Pix, where for every pair of captions 100 image pairs are generated before filtering, the proposed approach splits the generation into two parts to reduce the time of generation: for the images with single objects, 30 image pairs are generated, while for rooms with multiple objects, 50 pairs are generated. The choice for fewer pairs for single object images comes from the idea that the fewer the number of objects in the image, the less diversity in the images of the same pair will be present in the

Prompt-to-Prompt generation. With this approach, 4,259 train samples and 1,129 test samples were generated. The training and testing datasets can be accessed on HuggingFace [34, 33]. Furthermore, the generated data, not only that is visually appealing and diverse, but it also exposes the limitations of the InstructPix2Pix model in generalizing for the interior-design case and its poor performance as shown in Figure 1.

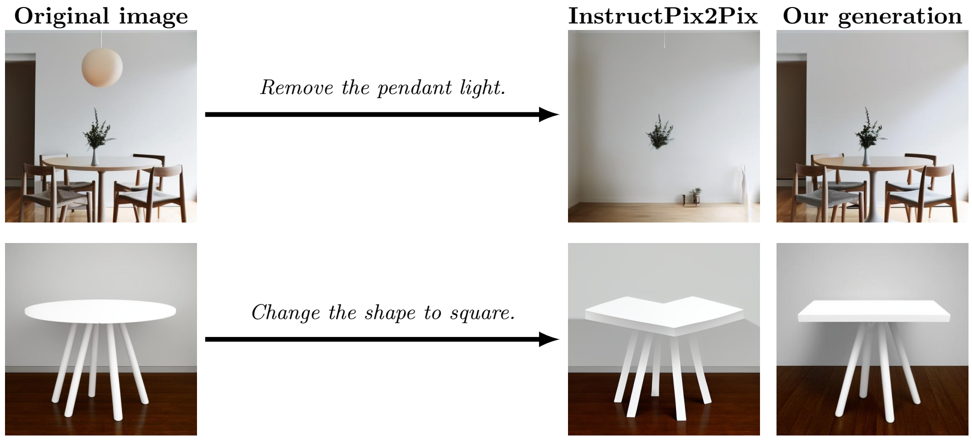


Figure 1. Comparison between the generated data and InstructPix2Pix's performance on it.

Furthermore, for reducing the noise introduced by the models used for generation and for increasing the dataset size, samples with an edit instruction that does not alter the initial image are introduced for augmentation (see Figure 2). Intuitively, this will also enforce the model to correctly identify the Region of Interest for the edit and to learn that in some cases the given prompt can be misleading, a problem that was not addressed in the previous approaches. This additional dataset can be found at [29], and its effects will be studied in the following sections. Moreover, these types of prompts with no effect on the image were also introduced in the initial test set.

3.2. **Fine-tuning InstructPix2Pix on generated dataset.** Having the previously generated data, we investigate its benefits when used to fine-tune InstructPix2Pix in order to answer research question RQ2. However, due to resource limitations, the training setup was modified to satisfy the computing capabilities. For this, the training was run in float16 precision, with a batch of only two images, and the images were resized from an initial dimension of 512x512 to 256x256. Nonetheless, even with these restrictions, the overall training was not affected drastically, and the results are promising. Using the training data generated with the method presented in Section 3.1, Instruct-Pix2Pix was fine-tuned on 300 epochs with a learning rate of 10^{-5}.

3.3. **Enhancing Region of Interest detection using a Referring Expression Segmentation model.** This section presents an alternative approach to the one introduced by Enis et al. [25] by leveraging the overwhelming performance of the recent text-based segmentation model, ReLA. This section aims to respond to research question RQ3 and to explore if such a method is improving the edit application in a general setting.

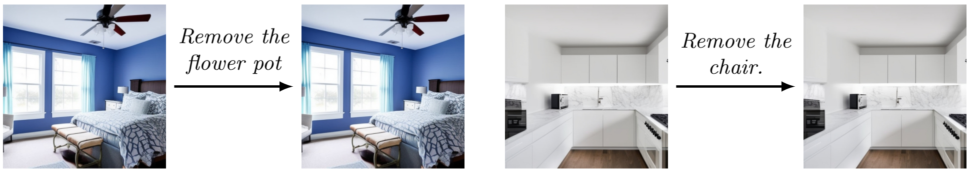


FIGURE 2. Data augmentation with samples containing no change in the output image

The proposed approach differs from the one introduced in LIME, just by how the segmentation map of the region(s) under edit is computed, here using ReLA. The usage of ReLA enhances the edit localization by being state-of-the-art in this task, and, additionally, it allows context-dependent references like *"The right blue chair."*.

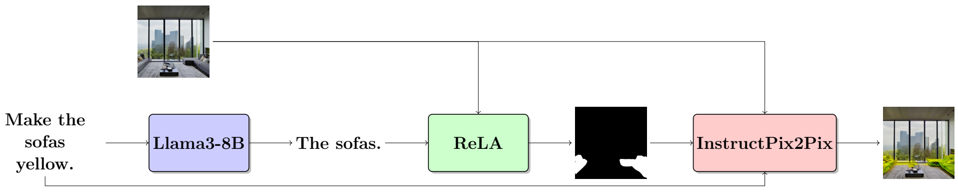


FIGURE 3. The pipeline for computing the edit through cross-attention regularization using ReLA's segmentation mask.

In order to be able to use ReLA in the editing pipeline, we propose the solution showcased in Figure 3. To use the text-based segmentation model, we first need to create a reference to the object(s) to be edited out of the initial edit prompt. This is achieved by creating an LLM agent via in-context learning by injecting the task description and a couple of examples in the model's system prompt as highlighted in Figure 4. Here, we use the 8B version of the most recent Llama3 [16] model. Afterward, we compute the text-based segmentation map determined by ReLA and feed it along with the initial image and the edit prompt in the modified version of InstructPix2Pix for cross-attention map regularization. We use the same approach of negatively regularizing the unrelated tokens to edit (e.g. padding tokens, <*start of text*>, etc.) which also offers the model more freedom in the edit application compared to the positive regularization of the related tokens.

```
system_message = """
You are a bot that needs to take the reference of the text under edit
    from an edit prompt or the object that affects it. Here are some
    examples
    'Replace the top book on the desk.' would transform into the
        following reference 'The top book on the desk.'
    'Add a plate on the wooden table.' would transform into the
        following reference 'The wooden desk.'
Please return just the transformed text as the reference and nothing
    more.
"""

messages = [
    {"role": "system", "content": system_message},
    {"role": "user", "content": edit_prompt},
]
```

FIGURE 4. The prompt used for extracting the object reference from edit prompt via in-context learning with Llama3-8B.

4. RESULTS AND DISCUSSION

This section is focused on presenting the experimental results of the presented approach along with the discussions that emerged from the visuals and analysis on the metrics.

4.1. **Results.**

This section is focused on presenting the experimental results of the proposed methodology for improving the edits in the interior-design context and enhancing the edit localization through cross-attention map regularization.

4.1.1. *Fine-tuning InstructPix2Pix on generated dataset.*

As shown in Table 2, the proposed approach improves the performance of the model considerably across different metrics computed as the cosine similarity between the features extracted using CLIP [20] and DINOv2 [18]. The various CLIP metrics presented in the table compute different types of similarities consisting of the similarity between the input and output images (CLIP_{im}), the similarity between the edited image and its textual description (CLIP$_{\text{out}}$), and the similarity between the changes in the captions and the images (CLIP$_{\text{dir}}$), while DINO only computes the similarity between the initial and edited image. The model resulting from this experiment is publicly available on HuggingFace [30] and can be freely used for image editing.

	CLIP$_{\text{im}}$ ↑	CLIP$_{\text{dir}}$ ↑	CLIP$_{\text{out}}$ ↑	DINO ↑
IP2P	84.25	0.025	26.16	87.67
IP2P-FT	**92.21**	**0.063**	**29.17**	**94.54**

Table 2. Comparisons between the metrics of the base InstructPix2Pix model and the fine-tuned one on the test set.

4.1.2. *Additional fine-tuning on the dataset with unchanged images.* After fine-tuning the model on the train data, an experiment of fine-tuning the model on the dataset with unchanged images was conducted. Doing this for more epochs results in a model that does not apply any modifications to the image, but fine-tuning for just one epoch does not alter the performance completely. Unfortunately, in most cases, the model still learns just not to change the original image at all, ignoring the edit instruction and underperforming in most of the cases. However, in some cases, even though their number is small, the output of this model is better than the previous one, this model also being publicly available [31].

4.1.3. *Enhancing Region of Interest detection using a Referring Expression Segmentation model.* The experiments conducted to incorporate ReLA's segmentation masks into the editing process via cross-attention map regularization did not show positive results up to this point. However, as seen in Figure 5, it enhances the localization of the edit region by forcing the model not to modify the unrelated background or objects. Nonetheless, the application of the edit is not done correctly in most cases by showing colors and shapes that are mostly random and off the edit prompt.

4.2. **Discussion.** The conducted experiments show improvements in instruction-based editing for interior design images which can be extrapolated to any context-specific case. However, there are a lot of observed particularities that occur during the analysis of the experimental results and this section aims to present them.

As shown and stated in Table 2, fine-tuning the base InstructPix2Pix model on the generated data with interior-design samples improves the model's ability to work in such an environment as shown in Figure 6. It can be seen that the dataset not only offers the ground truth for the output image but also a lot of knowledge that is assimilated by the model through supervised learning.

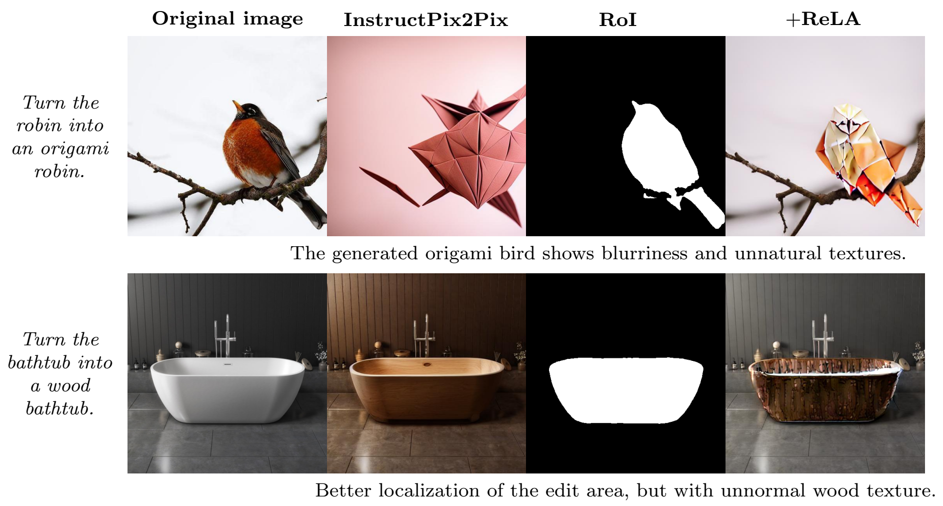

Original image
InstructPix2Pix
RoI
+ReLA
Turn the robin into an origami robin.
The generated origami bird shows blurriness and unnatural textures.
Turn the bathtub into a wood bathtub.
Better localization of the edit area, but with unnormal wood texture.

FIGURE 5. Examples of the images edited after integrating ReLA's segmentation mask for cross-attention map regularization.

Even though the new model's edits are more qualitative, there are still problems with the editing of unwanted parts like background or objects that are not referred to in the edit prompt. This can be seen in Figure 6 where, in the first image, one flower from the table disappears, the table color is changed from light grey to a slightly darker tone, and the lamp disappears. Furthermore, in the second image, the table top is changed correctly, but the color of the floor becomes more cherry.

4.2.1. *Dataset generation.* To eliminate such cases and better improve the performance, a more qualitative dataset needs to be created. First of all, the current dataset is not very diverse in the context and words used which is due to the way the prompts are generated and its reduced volume. Moreover, generating the initial captions can also be done by starting from interior design images available on the Internet and using an image-to-text model that describes the given image. After this step, the same idea introduced by Brooks et al. [3] can be applied by fine-tuning an LLM for the generation of caption pairs. However, this method has a limitation given by the number of publically available images for the context under discussion. This affects the scalability of the method for various contexts and the accuracy as the image-to-text model would also introduce noise to the dataset. For example, for the

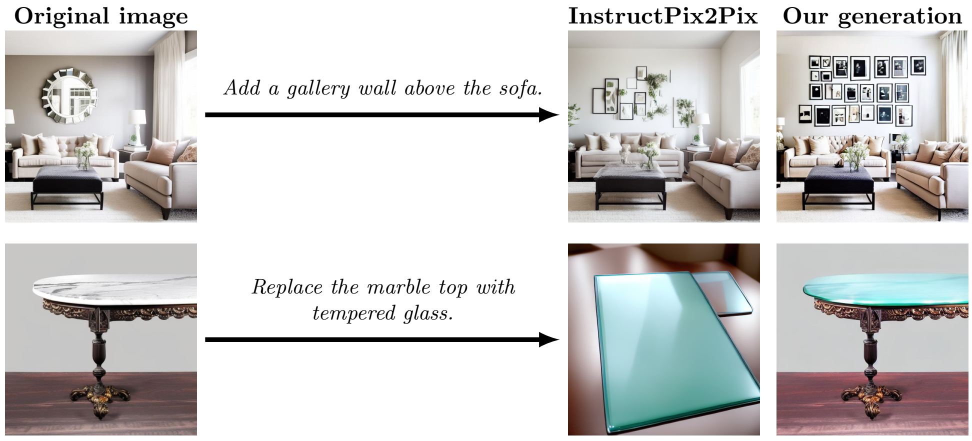

Original image
InstructPix2Pix
Our generation
Add a gallery wall above the sofa.
Replace the marble top with tempered glass.

Figure 6. The new model's edits are a lot more qualitative than the InstructPix2Pix's ones for interior design.

interior design case, one such dataset is the Interior Design IKEA dataset [26] which has only 6,000 images.

On the other hand, the introduced noise also comes from the level of image generation but in two different aspects. Firstly, the diffusion model used, here Stable Diffusion (SD), does not have a good understanding of interior design scenes as shown in Figure 7 and also fails to correctly follow the given prompt. Furthermore, there are also cases when it introduces more objects than presented in the textual description. Secondly, the noise is also introduced during the generation with Prompt-to-Prompt followed by CLIP filtering because, in some cases, the images do not differ only by the resulting actions expressed in the edit prompt as seen in Figure 8.

Comparisons to related work are limited due to the recent publishing date of Emu Edit [24] and LIME [25]. However, as stated before, all the solutions addressed in Section 2 require at least a large volume of initial captions that are used to build the dataset for supervised training. Compared to these, we propose and show the effectiveness of a new approach to generate a context-specific dataset for this task with no previously available information. Hence, in contrast to previous work in this area, the presented approach increases the scalability and the amount of data that can be generated by not relying on any available information.

Furthermore, compared to the approach introduced by Enis et al. [25], using ReLA for computing the Region of Interest applies a regularization effect only if the referred object is present in the input image. However, the LIME approach gives a more general approach by computing the mask for all-purpose

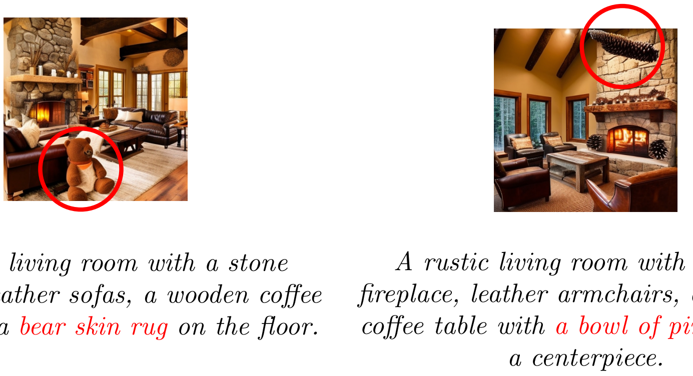

A rustic living room with a stone fireplace, leather sofas, a wooden coffee table, and a bear skin rug on the floor.
A rustic living room with a stone fireplace, leather armchairs, and a pine coffee table with a bowl of pinecones as a centerpiece.

FIGURE 7. Generated images that show the limited knowledge of Stable Diffusion in interior design.

tasks, even for creating masks with initially non-existent objects as minimally, but not explicitly expressed in the paper in just one example. Nonetheless, using ReLA in the Remove and Replace tasks would offer a greater benefit due to its better performance in text-based segmentation tasks, but with the overhead of using an additional language model for extracting the region reference out of the edit prompt. Furthermore, this implies a growth in the time needed to compute the edit because, in LIME, the segmentation map is computed using the internal features already existent in InstructPix2Pix, while we propose a method that uses two additional networks.

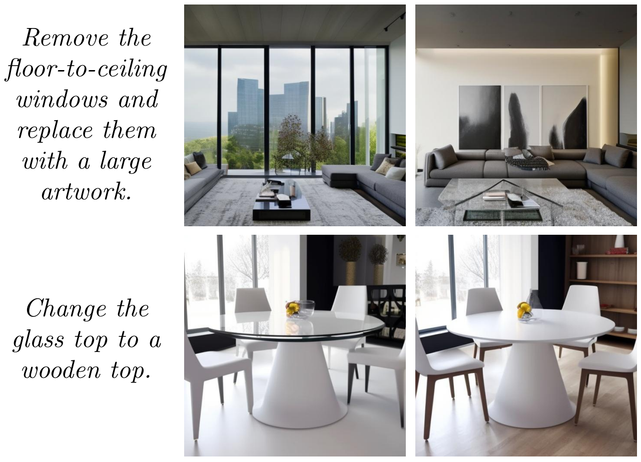

Remove the floor-to-ceiling windows and replace them with a large artwork.
Change the glass top to a wooden top.

FIGURE 8. Examples of generated samples that do not correctly follow the edit instruction.

5. Conclusions and Future Work

This paper introduced `LangDes`, a new approach for improving the performance of the instruction-based image editing task in the interior design setting. Afterward, we proposed a promising approach for improving the future performance on the instruction-based image editing task, followed by experimental results and the associated discussions in Section 4.2.

The conducted experiments in the interior design setting showed overwhelming results and confirmed positive answers to the research questions RQ1-RQ3 formulated in Section 1. So, we proved that the proposed method for generating context-specific with no previous data stays valid, and we showed its efficiency in improving the InstructPix2Pix performance in the interior-design context. Afterward, we experimented with the integration of the text-based segmentation model ReLA in the editing pipeline to improve the edit localization. However, as an answer to research question RQ3, the current experimental results only prove the localization improvement, but the application of the edit is still not under control, with the model returning images with random textures within the RoI.

One first direction for future work would be to increase the quality and diversity of the generated dataset. To increase the latter, the initial textual data needs to be more diverse. A solution for this would be to combine the output of multiple LLMs such as Llama2-70B [27], Gemini [2] or Mistral 8x7B [13]. On the other hand, task-room-specific agents can be created using in-context learning, but with the disadvantage of a large number of possible combinations that will increase the time required for the conducted experiments. Furthermore, to increase the quality and consistency of the image pairs, different text-to-image models like Imagen [22], Muse [5] or an interior design fine-tuned version of Stable Diffusion open-sourced at [32] can be used, as well as applying a more comprehensive filtering pipeline as the one presented in Emu Edit [24].

Another interesting topic is the complexity of the edit prompt. Even though the presented work focuses on prompts with single edits of single objects, a possible direction for experiments can be targeting more complex instructions such as the ones involving multiple actions. This can be both seen as extra evaluation of the model resulted from in presented work, as well as an extension at the level of data generation for creating such samples.

As a last future work direction, we may refer to *enhancing region of interest detection* using various RES models, along with investigating the cause of incorrectly applying the edit despite correctly localizing the targeted area. To improve the editing of parts of the objects, a combination of GRES and Multi-Granularity Referring Expression Segmentation (MRES) [28] can be used. Compared to, GRES, MRES, introduced by Wang et al. [28], supports expressions for segmenting part-level regions of the target objects within a model called UniRES, but with no support for a good performance with multiple objects at the same time. Having these two models, an additional decisional network for detecting which type of segmentation model should be used for computing the mask before the edit.

References

[1] Alex Reuneker. Lexical Diversity Measurements. `https://www.reuneker.nl/files/ld/`, 2017. Accessed: 2024-01-15.

[2] Rohan Anil, Sebastian Borgeaud, and et al. Gemini: A Family of Highly Capable Multimodal Models. *CoRR*, abs/2312.11805, 2023.

[3] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to Follow Image Editing Instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Canada, 2023*, pages 18392–18402. IEEE, 2023.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Subbiah, and et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[5] Huiwen Chang, Han Zhang, and et al. Muse: Text-To-Image Generation via Masked Generative Transformers. In *International Conference on Machine Learning, ICML 2023, Honolulu, Hawaii, USA*, volume 202, pages 4055–4075. PMLR, 2023.

[6] Xiaoliang Dai, Ji Hou, and et al. Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack. *CoRR*, abs/2309.15807, 2023.

[7] Michael Daller. Guiraud's index. 2010.

[8] Daniel Dugast. *La Statistique Lexicale*. SLATKINE, 1980.

[9] G. Udny Yule. The Statistical Study of Literary Vocabulary. *Cambridge University Press*, 1944.

[10] Rinon Gal, Or Patashnik, Haggai Maron, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. StyleGAN-NADA: CLIP-guided domain adaptation of image generators. *ACM Trans. Graph.*, 41(4):141:1–141:13, 2022.

[11] Amir Hertz, Ron Mokady, and et al. Prompt-to-Prompt Image Editing with Cross-Attention Control. In *The Eleventh International Conference on Learning Representations, 2023*. OpenReview.net, 2023.

[12] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *CoRR*, abs/2207.12598:1–14, 2022.

[13] Albert Q. Jiang, Alexandre Sablayrolles, and et al. Mixtral of Experts. *CoRR*, abs/2401.04088, 2024.

[14] Chang Liu, Henghui Ding, and Xudong Jiang. GRES: generalized referring expression segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 23592–23601. IEEE, 2023.

[15] Philip M. McCarthy and Scott Jarvis. MTLD, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42:381–392, 2010.

[16] Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. `https://ai.meta.com/blog/meta-llama-3/`, 2024. Accessed: 2024-05-10.

[17] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[18] Maxime Oquab, Timothée Darcet, and et al. DINOv2: Learning Robust Visual Features without Supervision. *CoRR*, abs/2304.07193, 2023.

[19] Koutilya PNVR, Bharat Singh, Pallabi Ghosh, Behjat Siddiquie, and David Jacobs. Ld-znet: A latent diffusion approach for text-based image segmentation. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 4134–4145. IEEE, 2023.

[20] Alec Radford, Jong Wook Kim, and et al. Learning Transferable Visual Models From Natural Language Supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021,*, volume 139, pages 8748–8763. PMLR, 2021.

[21] Robin Rombach, Andreas Blattmann, and et al. High-Resolution Image Synthesis with Latent Diffusion Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022*, pages 10674–10685. IEEE, 2022.

[22] Chitwan Saharia, William Chan, Saxena, and et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems*, volume 35, pages 36479–36494. Curran Associates, Inc., 2022.

[23] Christoph Schuhmann, Romain Beaumont, and et al. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems*, volume 35, pages 25278–25294. Curran Associates, Inc., 2022.

[24] Shelly Sheynin, Adam Polyak, and et al. Emu Edit: Precise Image Editing via Recognition and Generation Tasks. *CoRR*, abs/2311.10089, 2023.

[25] Enis Simsar, Alessio Tonioni, Yongqin Xian, Thomas Hofmann, and Federico Tombari. LIME: localized image editing via attention regularization in diffusion models. *CoRR*, abs/2312.09256, 2023.

[26] Ivona Tautkute, Aleksandra Mozejko, and et al. What Looks Good with my Sofa: Multimodal Search Engine for Interior Design. *CoRR*, abs/1707.06907, 2017.

[27] Hugo Touvron, Louis Martin, Kevin Stone, and et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*, abs/2307.09288, 2023.

[28] Wenxuan Wang, Tongtian Yue, and et al. Unveiling Parts Beyond Objects: Towards Finer-Granularity Referring Expression Segmentation. *CoRR*, abs/2312.08007, 2023.

[29] Victor-Eugen Zarzu. Dataset for interior design. `https://huggingface.co/datasets/victorzarzu/interior-design-edit-captions`, 2024.

[30] Victor-Eugen Zarzu. Fine-tuned InstructPix2Pix model. `https://huggingface.co/victorzarzu/ip2p-interior-design-ft`, 2024.

[31] Victor-Eugen Zarzu. Fine-tuned InstructPix2Pix model on the dataset with unchanged images. `https://huggingface.co/victorzarzu/ip2p-interior-design-ft-unchanged-one-epoch`, 2024.

[32] Victor-Eugen Zarzu. Interior design fine-tuned version of Stable Diffusion. `https://huggingface.co/stablediffusionapi/interiordesignsuperm`, 2024.

[33] Victor-Eugen Zarzu. Testing data. `https://huggingface.co/datasets/victorzarzu/interior-design-prompt-editing-dataset-test`, 2024.

[34] Victor-Eugen Zarzu. Training data. `https://huggingface.co/datasets/victorzarzu/interior-design-prompt-editing-dataset-train`, 2024.

Babeș-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu, Cluj-Napoca 400084, Romania

Email address: `victor.zarzu@stud.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 2, 2024
DOI: 10.24193/subbi.2024.2.03

DEEP LEARNING APPROACHES FOR DETECTING TEXT GENERATED BY ARTIFICIAL INTELLIGENCE

DAVID BIRIȘ

Abstract. Large language models have been a hot topic for discussion and research for quite a few years, allowing them to infiltrate in many industries, especially education. Their rise in popularity among students was caused by their vast capabilities in giving quick and reliable answers to questions on any topic. The use of these models for the purpose of generating schoolwork can be seen as a challenge to academic integrity. We investigate the development of AI capable of detecting AI-generated texts and explore with training different types of deep learning models, on a mixed dataset, containing essays, both human written and AI-generated, as well as movie reviews and books. We experimented with LSTM (Long short-term memory) and fine-tuning transformer based models. We achieve results close to the state of the art, and, in some particular cases, we surpass a few of these models. For instance, one of our models surpasses a state of the art model on a set of both student written and generated essays, in terms of accuracy by up to 5%, and F1 score by up to 4%, in two different experiments. Furthermore, our another model of ours surpasses a state of the art model on a set of essays, but this time only in terms of precision, by only 1%. These results indicate the potential of properly fine-tuned transformer-based models, as well as the importance of a well-prepared dataset.

1. Introduction

Ever since the revolutionary introduction of the transformer model in 2017 [29], the artificial intelligence industry has experienced a never-before-seen

Received by the editors: 31 July 2024.
2010 *Mathematics Subject Classification.* 68P15, 94A12.
1998 *CR Categories and Descriptors.* I.2.7 [Artificial Intelligence]: Natural Language Processing – Text Analysis; I.2.6 [Artificial Intelligence]: Learning – Deep Learning; H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing – Content Analysis and Feature Selection .
Key words and phrases. machine learning, chatbot, AIGT, detection.

explosion in both performance and popularity. While the transformer was initially designed for translation tasks, it has since been adapted for a varied range of uses in natural language processing and way beyond. Large language models (LLMs) were introduced the very next year, with pioneers such as OpenAI's Generative Pre-Trained Transformer (GPT) [27] and Google's Bidirectional Encoder Representations from Transformers (BERT) [8]. One more year later, LLMs began to be publicly accessible with the release of GPT-2, and, in no time, people realized the immense potential of these models as chat bots.

Today, these chat bots are everywhere and are immensely more capable than their predecessors. They are very popular and widely used by people in diverse everyday tasks. Moreover, tools like ChatGPT have become helpful allies for students, when trying to do their schoolwork. According to [3], 19% of students aware of ChatGPT admitted to having used it for schoolwork. This number is already high, but we can safely assume that, in reality, it might be a lot higher, since only 25% first-year computer science students that participated in a survey for this study [5] declared to have never used ChatGPT for their assignments.

While these tools might not yet be useful for reliably solving complex problems, they certainly can write a satisfactory essay on any topic, especially for students in earlier grades. Research made by OpenAI [22] shows that GPT-4 on its own can even pass many exams, with high grades. Some people might see this as concerning, since students' usage of these tools lead to inaccuracies in the evaluation of their text comprehension, writing abilities and both logical and critical thinking.

To address these concerns, we create a tool designed to help both students and teachers. We aim not to punish students for using large language models to learn and find ideas for their assignments, which can be a good habit, but to discourage the damaging practice of carelessly copying and pasting entire AI-generated essays and sending them as homework solutions.

In the following, we will present the steps we took to achieve to this finished product in great detail. We will start by presenting three state of the art models, in a section dedicated to related work. The next section will show the employed methodology and will briefly discuss the most important architectural aspects of each of the models. The experiments section focuses on first discussing the content of the datasets and then the processing techniques that have been applied to the data. After that, the process of training and hyperparameter tuning is discussed, and the section is ended with results and a detailed comparison between our models and the state of the art presented in the aforementioned section. The article ends with a section that draws conclusions and brings out possible future improvements.

2. Related Work

2.1. Sentence-Level AI-Generated Text Detection with SeqXGPT. Most AIGT (AI-Generated Text) detection strategies are made with the purpose of detecting, with as high of an accuracy as possible, if an entire document is automatically generated by a LLM, rather than using a sentence-by-sentence approach. While this technique may be useful in many cases, people do not generally rely solely on AI to generate entire documents, and, instead, they often use it to modify or enhance content that was originally written by humans. These AI modifications can be simple enhancements or additions to certain sentences, or entire new AI-generated paragraphs scattered between human-written text. Therefore, using a sentence-level AI generated content detection strategy is crucial in some cases.

SeqXGPT [30] is an open-source, advanced method for sentence-Level AI-Generated text detection. Its approach consists of three parts: Perplexity extraction and alignment, Feature Encoder and Linear Classification Layer.

The AIGT detection tests performed on SeqXGPT show a significant difference when compared to other AIGT detection methods, such as DetectGPT [21] or Sniffer [18]. The LLMs used to construct the dataset are GPT2-x1, GPT-Neo, GPT-J and LLaMA.

2.2. Zero-Shot AI-Generated Text Detection with Fast-DetectGPT. According to a report by OpenAI [28], zero-shot detection uses a pre-trained generative model on text generated either by itself or by other similar models. This is done without subjecting the model to any supplementary training. Zero shot means that access to human written or AIGT samples is not assumed to perform detection. Generally, when using zero-shot AIGT detection, the average per-token log probability of the generated text is evaluated.

This model relies on the hypothesis that language models tend to use tokens with higher statistical probability because they have been pre-trained on lots of human written content. In contrast, humans individually do not display such bias since they compose texts based on contexts, meanings, and intents rather than data and statistics [1]. This hypothesis stems from the fact that language models try to mirror collective human writing behaviour instead of individual human writing behaviour, therefore presenting a contextual difference in the choice of words. This means that the conditional probability function $p(\widetilde{x}|x)$ is much higher for an AIGT x, in comparison with human written text.

Fast-DetectGPT works in these three steps:

- Step 1. Sample alternative word choices $\widetilde{x}_i$ for each token x.

- Step 2. Evaluate the conditional probabilities $p(x|x), p(\widetilde{x}_i|x)$ of these generated samples.
- Step 3. Compare them to arrive at a detection decision: $\frac{1}{n}\sum_i \log \frac{p(x|x)}{p(\widetilde{x}_i|x)} \overset{?}{>} \epsilon$

Here, x represents the entire input text, x_i denotes the i-th token in x, $\widetilde{x}_i$ represents an alternative token generated as a substitute for x_i, and n is the total number of tokens in the input text. The decision to classify the text as AI-generated or human-written depends on whether the aggregated score exceeds the threshold ϵ.

Empirical evidence shows an increase of around 75% in detection accuracy for Fast-DetectGPT over its predecessor, DetectGPT [21].

2.3. **Adversarial learning with RADAR.** In the context of AIGT detection tools, we refer as *adversarial learning* to the process in which two models are trained at the same time, with two different, even opposing, goals in mind. One of them is a generative model and the other is a discriminative model that tries to determine, with a certain probability, whether some sample came from the generative model or the training data [12]. The generative model's goal is to make the probability of the discriminative model to make a mistake as high as possible.

RADAR: Robust AI-Text Detection via Adversarial Learning is a framework for AIGT detection, that uses adversarial learning. By this method, the discriminator is a "two player game", composed of the paraphraser and the detector. These two "players" have contrasting objectives, because the paraphraser has to generate human-like content that should be able to avoid AI detection, while the detector's job is to enhance AIGT detectability. In the training phase, the paraphraser learns to rewrite text from a dataset generated by a target LLM, while trying to decrease the probability that the detector will be able to discern the difference between the paraphraser's text and human written text. At the same time, the detector learns to compare AIGT from the training dataset and from the paraphraser's output with human written text, in order to improve the detection performance [15].

3. Proposed models

The task at hand is formulated as a binary classification task with two classes: human written, which is considered to be negative and AI-generated, which is considered to be positive. Each input text belongs to one of these two categories and it will be evaluated at document-level, meaning that there will be no sentence-level analysis, such as in the case of SeqXGPT. We fine tune some relatively lightweight transformer-based models, like BERT and

DeBERTa (Decoding-enhanced BERT with Disentangled Attention) [13], since their small size allows them to be directly integrated in applications, without the need of an API, unlike some newer and heavier models, that cannot directly run on users' personal computers. Additionally, they are free to use and open source. For the same reason, we also train an LSTM, which turns out to be even lighter than the transformer models, at the expense of some performance.

3.1. **LSTM.** We start with the implementation of a LSTM model [14]. The text data is tokenized with the *basic_english* tokenizer provided by PyTorch, in the torchtext package [23]. The vocabulary is built from these tokens. We apply padding to the sequences, since the LSTM expects constant length inputs. We use the LSTM class from PyTorch to build the model and try multiple sets of hyperparameters to find the ideal ones, using Wandb [2].

The training is done using K-fold cross validation, splitting the dataset in K parts. One of the K parts is used as a validation dataset, and the other $K-1$ are used for training. This process is repeated until each of the parts has played the role of a validation dataset. We use the *KFold* function from *scikit-learn* [25] to set up the cross validation and we experiment with $3-5$ splits. The data is shuffled every time before splitting to ensure that the model does not make any connections related to the order of the entries, since the order does not matter. We save the model for every fold and pick the one with the highest performance.

As an optimization algorithm for gradient descent we use Adam.

The loss function used is a Binary Cross Entropy Loss, with a sigmoid over the outputs of the model, called *BCEWithLogitsLoss*, in the PyTorch library:

$$l_i = -w_i\big(y_i \cdot \log \sigma(x_i) + (1-y_i) \cdot \log(1-\sigma(x_i))\big) \tag{1}$$

for the i$^{\text{th}}$ example in the batch, where x_i is the the i^{th} raw output of the neural network, y_i is the actual i^{th} truth label and w_i is a weight associated with the i^{th} example, and can be useful if we want to give different importance to different examples in the batch.

The overall loss for the batch is the mean if the individual losses:

$$l(x,y) = \frac{1}{N}\sum_{i=1}^{N} l_i, \tag{2}$$

where l_i is defined in Equation 1, N is the batch size and $x = \{x_1, x_2, \dots, x_N\}$, $y = \{y_1, y_2, \dots, y_N\}$.

For this LSTM, we create a PyTorch module (a Python class that inherits from the Module class in PyTorch) named *TextLSTM*. Our final model will have type *TextLSTM*. Its constructor initializes the embedding layer, giving as parameters the size of the aforementioned vocabulary and the dimension of the embedding vectors, which we chose to be 100. The constructor also initialized the LSTM class from PyTorch that we have mentioned, as well as a linear layer called *fc* that maps the output of the LSTM (with dimension *hidden_dim*, a hyperparameter we chose to be 256) to the desired output size (which in this case is 1), since our task is binary classification. The *forward* pass function starts by embedding text and then passing it through the LSTM. Then it takes the last layer from the hidden states and passes it through the *fc*, before returning it.

3.2. **BERT.** We also fine-tune multiple transformer-based models, the first one being BERT. We use *bert-base-uncased*, a version of BERT with 110 million parameters. The parameters are basically the total number of weights and biases from the transformer's layers. It has been pre-trained only on English datasets. We choose the uncased version from the presumption that the case of the letters in a text are not significantly relevant in the context of detecting whether a text is AI-generated or not. BERT's tokenizer includes the functionality of encoding the tokens, but has the limitation of admitting a maximum of only 512 tokens per input, which might cause loss of data on entries with texts longer than that. This is why we might want to truncate and split the text into multiple sections of 512 tokens, but then we sometimes run into the problem of losing the necessary context for the transformer to make the needed connections for discriminating between AI and human text. We will handle this problem in the data preprocessing subsection 4.2. If the tokenizer does not find a specific word in the BERT dictionary, it splits it in the largest subwords from the vocabulary. In the rare case when the tokenizer cannot find a subword in the vocabulary, the entire word is tokenized as unknown [9]. If a sequence is already split in 512 tokens (the maximum number), and we need to split a word in subwords, the sequence will be truncated to the maximum length [8].

For training, we split the dataset into three sets: 80% of the data was used for training, 10% for testing, and 10% for validation. The test dataset is used only to evaluate the performance of the final model. The other two sets are used during training, just like in the LSTM: the validation is used to evaluate the model every *evaluation_frequency* steps on data that is new for the model, since it is not included in the training dataset. The number of steps in an epoch is dependent on the batch size, since each epoch is a complete pass through the entire dataset and the batch size indicates how many samples are taken for each forward back propagation of the neural network layer of the transformer. So, the number of steps in an epoch is $S = \frac{\text{dataset size}}{\text{batch size}}$.

We use AdamW as optimization algorithm for gradient descent with learning rate $5 \cdot 10^{-5}$. During the 1500 warm up steps, the learning rate is set to linearly increase to the desired value. The learning rate is set to linearly decrease to 0 by the end of the training process. This learning rate scheduler setup allows for the learning rate to stabilize the training in the early steps, since the model has not yet had the chance to adjust to the dataset and we risk overshooting minima of the loss function. After warm up, the learning rate is set to gradually decrease, and this helps the model tune the weights lightly near the end of the training.

3.3. **DeBERTa.**

We can fine-tune more BERT-based transformers models, such as DeBERTa [13]. We use the base version of DeBERTa, *deberta-v3-base*, with around 86 million parameters. It has 12 layers and a hidden size of 768. We split the dataset in the same way we do for the BERT model. Again, we use the AdamW optimizer, with learning rates like $5 \cdot 10^{-5}$ or $3 \cdot 10^{-5}$. The same 1500 warm up steps are followed and the learning rate linearly decays during training.

The improvement DeBERTa brings over BERT is that it separates the content and position information in different embeddings. This approach allows the model to more effectively focus separately on semantics and positions of the tokens. Additionally, DeBERTa has an upgraded mask decoder which gives it better predictions during training. With these perks, DeBERTa is a generally more efficient and better performing model in natural language processing tasks than BERT or RoBERTa [19].

4. Experiments

A crucial step in training a text classification model is thoughtfully compiling a dataset of content with diverse writing styles. The models train their weights to find patterns that discern real texts from fake ones, so it it very important to not accidentally introduce biases or imbalances that could skew the model's performance and to ensure these sources vary in tone, complexity, and subject matter. We gather content from a wide range of sources including books, reviews and especially essays, since our tool is created with the intent of primarily detecting schoolwork.

4.1. **Datasets.**

4.1.1. *DAIGT-V4.*

[17] is a dataset compiled from a number of different sources. The AI-generated section has texts generated by different models: LLaMA - 15,796 texts, Mistral - 13,439 texts, Falcon - 4,536 texts, GPT - 4,161, DaVinci - 2,099 texts, Claude - 2,000 texts, PaLM - 1,733 texts, Babbage - 698 texts, Curie - 696 texts, Ada - 692 texts, Cohere - 350 texts. The human generated

content is composed of argumentative essays written by 6th-12th grade students [6]. This dataset contains a total of 27,370 human generated texts and 46,200 AI generated texts.

The dataset contains values under multiple labels, such as the text itself, a value that tells us whether the content is AI or human generated, the name of the model it has been generated by, as well as the prompt used for generation. The topics (prompts) of the essays are the same for both human and AI texts.

4.1.2. *DAIGT Gemini-Pro 8.5K Essays.* This dataset [7] brings 8,500 more essays generated using the same prompts as the ones from DAIGT-V4. They are generated by GeminiPro. The CSV file contains multiple labels, such as the text itself, a value that tells us whether the content is generated by AI or written by humans, and also the prompt used to generate the text.

4.1.3. *IMDB 50K Movie Reviews.* This dataset [24] provides a set of 50,000 movie reviews from IMDB, written by humans. The CSV file provides both the review and the opinion reflected by the person in the review (positive or negative sentiment towards the movie), for sentiment analysis. For our purpose, we will not need to use the sentiment, and we will use this dataset as a collection of human written text.

4.1.4. *ArguGPT.* [20] provides 4,038 argumentative essays, on different topics, written by GPT (7 models). The CSV file contains labels for the text, the prompt, as well as an id for each text and prompt and also the individual GPT model used for generation.

4.1.5. *Raw IELTS essays.* Raw IELTS essays [4] is a collection of student-written essays, from the IELTS test. It provides a valuable amount of 4,158 essays, that should definitely help the model find different human writing patterns, during training.

4.1.6. *SeqXGPT's sentence-level detection dataset.* A section of the document-level detection dataset used for evaluating SeqXGPT [30]. It contains, among human generated texts, content from GPT-2, GPT-3, GPT-J, GPT Neo and Llama. We take the GPT-2, GPT-3 and human texts for training, and leave the rest for subsequent testing. GPT-2 and GPT-3 are extensively studied and thoroughly evaluated models, making them suitable choices for establishing a solid training dataset. We reserve GPT-J, GPT-Neo, and LLaMA for testing, in order to ensure that the trained model is evaluated on texts it has not seen during training phase. We use a total of 600 texts for training, from this dataset, 200 human written and 400 human generated.

4.1.7. *Some books.* We also include some books in the dataset. Books offer very high quality examples of human writing, while also being well edited and well reviewed. This should help the model's ability to detect subtle characteristics of human authored text. For diversity of writing styles, we choose both newer and older books. The books we have included in our dataset are *Crime And Punishment*, *The Great Gatsby*, *The Hunger Games*, *Frankenstein*, *Twilight*, *Harry Potter*, *The DaVinci Code* and *Tarzan Of The Apes.* After splitting these books in sections of maximum 450 words (in order to have similar length texts in the dataset) we get about 2,100 small texts.

4.1.8. *Alpaca GPT4.* Alpaca GPT4 [10] [26] is a collection of instruction-following texts generated by GPT-4. It does not include essays, but these texts might help our model understand some more diverse patterns in AI-generated content, so it might be beneficial to not only include essays in the training dataset. For this reason, we choose a random set of 2,100 samples of the total 52,000. Since these texts are not essays, they have an unusual writing style compared to the other AI-generated texts, so we choose 2,100 in order to match the number of texts from the previous human-written dataset, composed of books. This way, we have a balance between AI-generated and human-written texts with different writing styles.

4.1.9. *AI Vs Human Text.* This dataset is a huge cluster of essays, both AI-generated and human written essays[11]. It includes some of the datasets presented above and many more. Since our tool targets detecting AI-generated content in academic scenarios, this dataset is a very good choice due to its large collection of essays, aligned with academic writing styles. This dataset contains around 300,000 human written and 180,000 AI-generated essays. We have used AI-generated some texts from "AI Vs Human Text" only as a filler, for balancing the training dataset, since we ended up with more human written content.

4.1.10. *Testing Dataset.* This dataset will only be used for testing, so we do not count it in with the other training datasets. We use another IELTS essays dataset [16], different from the one used for training. This dataset contains both the question that the students were asked to write the essay about, and the student essay itself. We take a sample of these questions and ask the most popular AI models to write essays as well. Now we have created our test dataset with some student essays and some generated essays. We have, in total, 1332 pieces of writing created by *gpt-3.5-turbo*, *gpt-4-turbo* and *gpt-4o*, the latest model from OpenAI, as of this writing. We have generated these essays ourselves, using the API that OpenAI has made publicly available.

4.1.11. *Data splitting.* The number of texts in the final dataset and their provenience is described in detail in Table 1. We split the dataset in three sections: the training set (80% of the entire dataset), the validation set and the test set (both 10%). The training set is used for the actual process of adjusting the weights of the model while training. The validation dataset helps assessing the model's performance during training and preventing overfitting by providing a separate set of data to evaluate the immediate performance of the model. The test set is a completely unseen section of the data, that provides an unbiased evaluation of the model's performance after the training is completed.

Datasets				
Dataset Name	Human Written Texts	AI-Generated Texts	Total	AI/Human Ratio
DAIGT-V4	27,370	46,200	73,570	1.68
DAIGT Gemini-Pro 8.5K	0	8,500	8,500	-
IMDB 50K Movie Reviews	50,000	0	50,000	-
ArguGPT	0	4,038	4,038	-
Raw IELTS essays	4,158	0	4,158	-
SeqXGPT's sentence-level detection dataset	200	400	600	0.5
Alpaca GPT4	0	2,100	2,100	-
Books	2,100	0	2,100	-
AI Vs Human Text	0	22,590	22,590	-
Total	83,828	83,828	167,656	1

Table 1. Summary of datasets used.

4.2. **Data preprocessing.** Before doing any further processing on our dataset, the texts have been grammatically corrected. Correcting grammar is an important step in processing our datasets. We use a special Python library called *language-tool* to correct all the grammatical errors in all our texts, both AI and human written. We do not want our model to form bias towards labeling a text as AI just because it does not have grammatical errors. After correcting the grammar in the texts, we see that 1,456,283 errors have been corrected in human texts and only 631,083 in AI content. This data supports our previous hypothesis that the model could have been biased towards labeling correctly written text as AI, when trained on an uncorrected dataset.

Since transformer-based models like BERT or DeBERTa have a maximum input size of 512 tokens, we cannot keep texts longer than 512 tokens in our dataset. We could just truncate the longer texts, but this way we could lose meaningful context from those texts. This is why we will just discard the texts with more than 500 words. We choose to count words instead of tokens since the BERT and DeBERTa tokenizers are different, but they both generally split tokens as words. The reason we choose not to split them into chunks, as we have previously done for the books dataset is because there is not enough content that we can work with in these texts. The texts are generally only slightly longer than 500 words, and, by keeping the surplus in a separate chunk, we would have many short pieces of text with no context behind. We will also discard texts with less than 50 words, since they might not provide enough context for the AI model to properly train. The vast majority of the texts lied in the desired range, even before this processing, as can be seen in Figure 1, but, after discarding texts that are too long or too short, Figure 2 shows a nicer, almost Gaussian distribution of text lengths. We have discarded a few too long or too short texts, but we still have 152,386 texts to work with, 76,534 human written and 75,852 AI-generated.

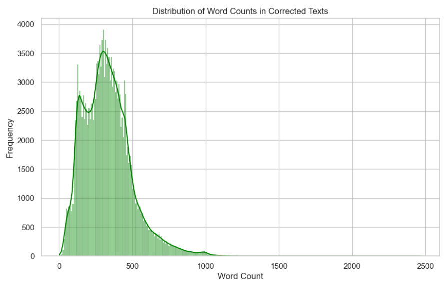


FIGURE 1. Initial distribution of word counts in the dataset.

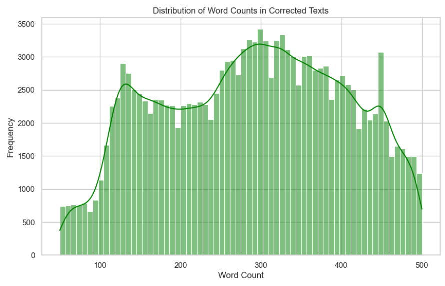


FIGURE 2. Distribution of word counts in the dataset, after discarding too long or too short texts.

4.3. **Metrics.** During training, we compute accuracy, loss, and validation loss every epoch. These values, called metrics, specifically the last two, help us interpret the progress during training. Validation loss is computed on the validation dataset, and loss on the training dataset. If the two loss values become closer and lower, it is a good sign that the training process has steady and good progress. Otherwise, if they are far apart, this might indicate *overfitting.*

We can also track the learning rate and *f1 score.* The f1 score is a function of *precision* and *recall.* The precision is a metric that shows us how accurate the positive predictions are. We will abbreviate true positives by TP, false positives by FP and false negatives by FN. TP are the texts correctly classified as AI-generated, and FN are texts incorrectly classified as human written. Then, the precision has the formula $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$. Recall is the ration between true positives and all the actual positives: $\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$. The f1 score measures the balance between precision and recall and has values between 0 and 1, 1 meaning perfect precision and recall:

$$\text{f1\_score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{3}$$

4.4. **Training and hyperparameter tuning.** It is good practice to implement multiple models, based on different architectures, and tune their hyperparameters, in order to find the best possible solution. The deep learning library of choice is PyTorch [23]. Hyperparameter tuning is done with Wandb. Wandb *Weights & Biases* is a platform that helps with tracking a history of experiments in machine learning. It provides the tools to log each training run and save the training progress, the hyperparameters and the metrics of the model (loss, accuracy, f1_score, etc.), as well as many details about the system's performance during training. In addition, it creates graphs with these metrics. The *sweep* functionality from Wandb allows us to pre-plan multiple sets of hyperparameters and Wandb will train the model with these multiple possible settings so we can choose the preferred one.

When training the LSTM model, we choose to use 5 folds for k-fold cross validation and experiment with different hyperparameters. The best results have been achieved with the dimension of the vector space in which words are represented (embedding dimension) set to 100. We choose the number of neurons in the hidden layers (hidden dimension) to 256, the learning rate to 10^{-3} and the dropout rate to 20% of the neurons. We train this model with a batch size of 128. The training and validation loss progress for each of these 5 folds can be seen in Figure 3.

When fine-tuning BERT, only 3 epochs are needed, since this is a transformer-based model, pre-trained on massive datasets and has already captured a significant number of natural language features. This is the reason why the learning rate we set is much lower, compared to the LSTM one, specifically 10^{-5}. During fine-tuning we only need to finely adjust the model's weights, to fit our new text classification purpose. We also need to set the batch size to

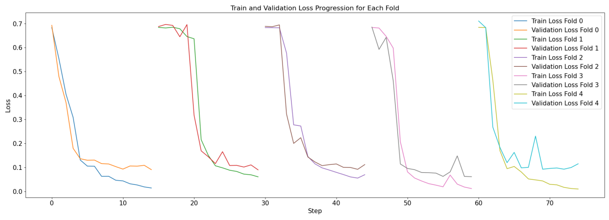


FIGURE 3. BCEWithLogitsLoss (sigmoid wrapped cross entropy loss) progression in the training process of the LSTM model (training vs validation loss), for each of the 5 folds

a lower number, 16, since the multi-head attention mechanisms of the transformer require much more video RAM than the LSTM architecture. The loss progression during the fine-tuning process is displayed in Figure 4.

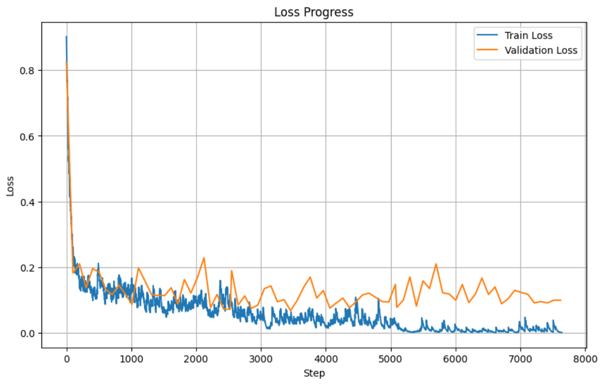


FIGURE 4. Training vs validation loss during the fine-tuning of the BERT model.

We fine-tune DeBERTa for 4 epochs, but this time we use three *sweeps* to automatically find the best hyperparameters, instead of manually changing them and trying again. The best results are achieved with a dropout rate of 0.1, a initial, linearly decaying learning rate of $3 \cdot 10^{-5}$ and a batch size of 16. The first 1000 steps (first 1000 batches) in the first epoch are used as warm up steps, to gradually increase the learning rate to the initial value of $3 \cdot 10^{-5}$. The model is trained for two epochs, and the training progress can be seen in Figure 5.

4.5. **Results.** We will try to find a winner between our models, by subjecting them to a few classification test on our testing dataset. We will go through all

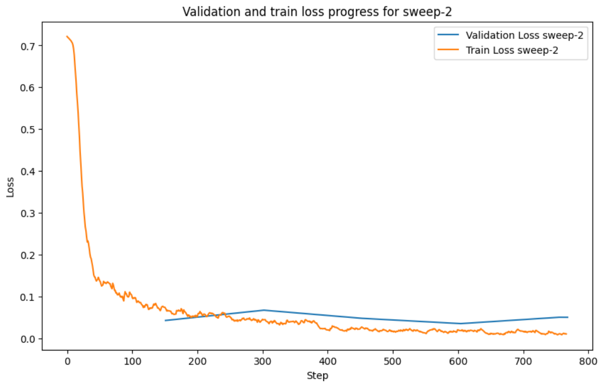


FIGURE 5. Training vs validation loss during the fine-tuning the DeBERTa model.

the sections of the dataset, which, as previously stated, is composed of human written and *gpt-3.5-turbo*, *gpt-4-turbo* and *gpt-4o* generated essays, as well as a section from SeqXGPT's dataset. We will begin by comparing only our three models on the *gpt-3.5-turbo* section, in order to establish a baseline, and then we will continue with a comparison with the state of the art models. We will compare the models by calculating metrics such as accuracy, precision, recall and f1 score for each one of them.

Model	Accuracy	Precision	Recall	F1 Score
Our LSTM model	0.78	0.78	0.78	0.78
Our BERT model	0.86	0.88	0.86	0.86
Our DeBERTa model	**0.90**	**0.91**	**0.90**	**0.90**

TABLE 2. Comparison between results from all our three models on 955 gpt-3.5 turbo vs 955 human texts from our testing dataset.

By doing some tests using our testing dataset we can observe that our DeBERTa model yields the highest performance of all three, as shown in Table 2. This result is expected, since DeBERTa is an improved version of BERT, both in performance and in efficiency, and it also is designed to focus more on semantics and the position of tokens, because of its different approach to embeddings.

Next, we will subject all three models, plus some more, on a series of more tests, and we will see if DeBERTa still retains its performance against our other methods and also against some state of the art methods.

4.6. **Comparison with other methods.** We now make use of the testing dataset that we've compiled to test our models versus some state of the art models presented in the second section. We use different sections of the dataset. Considering which model has generated the text is crucial when evaluating performance, as it allows for an assessment of whether the detection is effective against state-of-the-art generative models or if its capabilities are limited to identifying text generated by older, less advanced models.

Model	Accuracy	Precision	Recall	F1 Score
RADAR	**0.97**	**0.97**	**0.97**	**0.97**
Our LSTM model	0.76	0.78	0.73	0.76
Our BERT model	0.88	0.90	0.88	0.88
Our DeBERTa model	0.93	0.93	0.93	0.93

Table 3. Comparison between results from RADAR vs our models, on a sample of 286 student essays and 286 essays generated by gpt-3.5-turbo.

The AI-generated texts from the dataset for which the models yielded the results presented in Table 3 is different from the one in Table 2, though both are generated by gpt3.5-turbo.

Model	Accuracy	Precision	Recall	F1 Score
RADAR	0.88	0.89	0.88	0.88
Our LSTM model	0.68	0.75	0.53	0.62
Our BERT model	**0.93**	**0.93**	**0.92**	**0.92**
Our DeBERTa model	0.84	0.86	0.84	0.83

Table 4. Comparison between results from RADAR vs our models, on a sample of 94 student and AI essays generated by gpt-4-turbo

This time, in Table 4, the BERT model stands on top, overtaking even RADAR in gpt-4-turbo texts detection.

Again, our BERT model seems to classify texts from OpenAI's latest model, gpt-4-o very well, even slightly surpassing RADAR, as displayed in Table 5.

Model	Accuracy	Precision	Recall	F1 Score
RADAR	0.88	0.89	0.88	0.88
Our LSTM model	0.70	0.73	0.63	0.67
Our BERT model	**0.89**	**0.90**	**0.89**	**0.89**
Our DeBERTa model	0.73	0.79	0.73	0.71

Table 5. Comparison between results from RADAR vs our models, on a sample of 253 student essays and 253 essays generated by gpt-4o

As mentioned in the datasets section, we have used some samples from SeqXGPT's datasets for training, but we have also left some for testing. This time, we will compare our results with the one from the table for document level detection, from SeqXGPT's paper [30], since they have already tested on the same dataset. We will test our models and also the RADAR model on the sections left for testing from this dataset (200 GPT-J texts, 200 GPT-Neo texts and 200 LLaMA texts). Since we have used the human-written texts from this dataset for training the detection models, we cannot fairly compute precision, but we will compare the recall values, since recall is a function of true positives and false negatives and it only deals with truly AI generated texts. True positives are the texts correctly classified as AI-generated, and false negatives are texts incorrectly classified as human written.s

Model	GPT-J	GPT-Neo	LLaMA
Sniffer	0.74	0.83	0.07
Sent-RoBERTa	0.21	0.46	0.10
Seq-RoBERTa	0.26	0.40	0.72
SeqXGPT	**0.96**	**0.99**	**0.90**
RADAR	0.31	0.26	0.23
Our LSTM model	0.27	0.33	0.30
Our BERT model	0.95	0.97	0.89
Our DeBERTa model	0.73	0.81	0.67

Table 6. Comparison between recall values from models compared in SeqXGPT's paper, RADAR and our models, on three of the document-level datasets from SeqXGPT's testing sets.

RADAR and our LSTM seem to be performing particularly poorly on this specific dataset. SeqXGPT though has outstanding performance when compared to all other models in this case. Our bert model comes very close to SeqXGPT's performance, falling behind with only 1% accuracy when it comes to classifying the texts from these datasets as AI generated.

Next up, we compare Fast-DetectGPT [1] with our models, on a section of 80 texts from our IELTS student and gpt-3.5-turbo test dataset. We are constrained to reduce the size of the test dataset for this particular experiment due to the very heavy workload Fast-DetectGPT demands during execution. Fast-DetectGPT yields really good results, with an impressive perfect recall, meaning it correctly guessed all the AI generated texts, as can be seen in Table 7. Our DeBERTa model comes really close, followed by our BERT, and then the LSTM.

Model	Accuracy	Precision	Recall	F1 Score
Fast-DetectGPT	**0.97**	0.95	**1**	**0.97**
Our LSTM model	0.74	0.76	0.70	0.73
Our BERT model	0.89	0.91	0.89	0.88
Our DeBERTa model	0.96	**0.96**	0.96	0.96

Table 7. Comparison between results from Fast-DetectGPT vs our models, on a sample of 80 texts from our IELTS student and gpt-3.5-turbo dataset.

5. Conclusions and Future Work

To conclude, we have focused on developing a tool that aims to diminish academic dishonesty caused by the use of large language models. This dishonesty is not caused by many educationally appropriate use cases of generative pre-trained transformers, such as researching, searching for ideas, finding answers to problems in order to learn solving methods or even receiving feedback for one's own work. However, a problem could arise when students claim entire AI works or very big chunks of generated content as being their own. This is where our tool proves to be useful. We have trained multiple models with multiple architectures, on various datasets, to find, to the best of our ability, the best configuration for creating a tool specialized to detect essays, documents or stories generated by AI. Specifically, we have created an LSTM model, a BERT and a DeBERTa model, which are all lightweight, free to use and open source, so that they can all be run locally on any user's personal computer. Based on the comparison in the previous section, it is hard to pick a winner between BERT and DeBERTa, since they both perform the best between the 3 models developed by us in 3 out of 6 experiments. However, BERT surpasses a state of the art model, RADAR in all metrics in two of our experiments, whereas DeBERTa only manages to achieve a slightly better precision than Fast-DetectGPT, and, therefore, we will declare the BERT model our best.

With some future improvements, these models could become part widely-used tools in schools and universities all around the world. They could benefit from an even larger and more diverse dataset to be trained on, which would require much more computational power, but would also yield much better results. Experimenting with many different other transformer based models and different hyperparameters definitely brings potential for achieving a much higher accuracy. Another potentially big improvement would be creating custom embeddings for specializing models in particular detection applications, meaning detecting generated text for each school subject in particular. We would have, for example, a model specially designed to detect biology essays, another for history, and so on. By using our custom embeddings instead of pre-trained ones, we could much easier train a transformer for subject-specific tasks.

References

[1] Bao, G., Zhao, Y., Teng, Z., Yang, L., and Zhang, Y. Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature, 2024.

[2] Biewald, L. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[3] Center, P. R. About 1 in 5 u.s. teens who've heard of chatgpt have used it for schoolwork, November 2023.

[4] Cheplukov, A. Raw ielts essays, 2024. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/arsenycheplukov/raw-ielts-essays`.

[5] Cipriano, B. P., and Alves, P. "chatgpt is here to help, not to replace anybody" – an evaluation of students' opinions on integrating chatgpt in cs courses, 2024.

[6] Crossley, S. A., Baffour, P., Tian, Y., Picou, A., Benner, M., and Boser, U. The persuasive essays for rating, selecting, and understanding argumentative and discourse elements (persuade) corpus 1.0. *Assessing Writing 54* (2022). https://doi.org/10.1016/j.asw.2022.100667.

[7] Demir, E. Daigt gemini-pro 8.5k essays, 2023. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/datafan07/daigt-gemini-pro-8-5k-essays`.

[8] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[9] Face, H. Natural language processing course, chapter 6. `https://huggingface.co/learn/nlp-course/en/chapter6/6`, 2023.

[10] Gallego, V. Alpaca-gpt4 dataset, 2023. Retrieved May 10, 2024 from `https://huggingface.co/datasets/vicgalle/alpaca-gpt4`.

[11] Gerami, S. Ai vs human text, 2024. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/shanegerami/ai-vs-human-text`.

[12] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

[13] He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

[14] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[15] Hu, X., Chen, P.-Y., and Ho, T.-Y. Radar: Robust ai-text detection via adversarial learning, 2023.

[16] Ibrahim, M. Ielts writing scored essays dataset, 2023. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/mazlumi/ielts-writing-scored-essays-dataset`.

[17] Kłeczek, D. Daigt-v4-train-dataset, 2024. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/thedrcat/daigt-v4-train-dataset/data`.

[18] Li, L., Wang, P., Ren, K., Sun, T., and Qiu, X. Origin tracing and detecting of llms, 2023.

[19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach, 2019.

[20] Liu, Y., Zhang, Z., Zhang, W., Yue, S., Zhao, X., Cheng, X., Zhang, Y., and Hu, H. Argugpt: evaluating, understanding and identifying argumentative essays generated by gpt models, 2023.

[21] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C. Detectgpt: Zero-shot machine-generated text detection using probability curvature, 2023.

[22] OpenAI. Gpt-4. `https://openai.com/index/gpt-4-research/`, 2023.

[23] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

[24] Pathi, L. N. Imdb dataset of 50k movie reviews, 2019. Retrieved May 10, 2024 from `https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews`.

[25] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[26] Peng, B., Li, C., He, P., Galley, M., and Gao, J. Instruction tuning with gpt-4, 2023.

[27] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding with unsupervised learning. *OpenAI Blog* (June 2018).

[28] Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., McCain, M., Newhouse, A., Blazakis, J., McGuffie, K., and Wang, J. Release strategies and the social impacts of language models, 2019.

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.
[30] Wang, P., Li, L., Ren, K., Jiang, B., Zhang, D., and Qiu, X. Seqxgpt: Sentence-level ai-generated text detection, 2023.

Babeș-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu, Cluj-Napoca 400084, Romania
Email address: `david.biris1@stud.ubbcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 2, 2024
DOI: 10.24193/subbi.2024.2.04

HARDML: A BENCHMARK FOR EVALUATING DATA SCIENCE AND MACHINE LEARNING KNOWLEDGE AND REASONING IN AI

TIDOR-VLAD PRICOPE

Abstract. We present HardML, a benchmark designed to evaluate the knowledge and reasoning abilities in the fields of data science and machine learning. HardML comprises a diverse set of 100 challenging multiple-choice questions, handcrafted over a period of 6 months, covering the most popular and modern branches of data science and machine learning. These questions are challenging even for a typical Senior Machine Learning Engineer to answer correctly. To minimize the risk of data contamination, HardML uses mostly original content devised by the author. Current state-of-the-art AI models achieve a 30% error rate on this benchmark, which is about 3 times larger than the one achieved on the equivalent, well-known MMLU-ML. While HardML is limited in scope and not aiming to push the frontier—primarily due to its multiple-choice nature—it serves as a rigorous and modern testbed to quantify and track the progress of top AI. While plenty benchmarks and experimentation in LLM evaluation exist in other STEM fields like mathematics, physics and chemistry, the sub-fields of data science and machine learning remain fairly underexplored.

1. Introduction

Recent advancements in large language models (LLMs) have led to significant progress in natural language processing tasks such as translation, summarization, question answering, and code generation [1, 2]. These models have been extensively evaluated using benchmarks covering a wide range of subjects, providing valuable insights into their capabilities [3, 4]. For instance, the Massive Multitask Language Understanding (MMLU) benchmark

Received by the editors: 22 January 2025.
2020 *Mathematics Subject Classification.* 68T50, 68T07, 68T05, 68T20.
Key words and phrases. Large Language Models, Machine Learning Education, Multiple Choice Benchmark, NLP Benchmarks, Evaluation of AI Systems.

[5] assesses LLMs across diverse disciplines, including STEM fields like mathematics, physics, and chemistry [6, 7, 8]. However, data science (DS) and machine learning (ML) have received relatively little attention in benchmarking efforts. The MMLU test set contains only 112 machine learning questions. Moreover, in the few instances where these domains have been explored, state-of-the-art AI models achieve near-saturation performance, rendering existing benchmarks less effective for distinguishing model capabilities.

It is imperative to devise novel benchmarks that keep up with the rapid advancements in LLMs. This necessity is exemplified in the FrontierMath benchmark [15], which introduces a future-proof evaluation for mathematics by presenting problems that remain unsolved by over 98% of current AI models. Such benchmarks are crucial for continuing to challenge and develop advanced AI systems.

Data science and machine learning are foundational to modern artificial intelligence, driving advancements in everything from healthcare to finance [9, 10]. Mastery in these fields requires not only theoretical understanding but also practical skills in applying algorithms, statistical methods, and computational techniques to solve complex, real-world problems [11, 12]. As AI systems become increasingly involved in DS and ML tasks—ranging from automated model training to data analysis—it is crucial to assess their proficiency and reasoning abilities in these areas. However, as of January 2025, benchmarks in this domain are very limited. The most notable examples include the test ML subsection of MMLU (MMLU-ML) [5], which consists of 112 multiple-choice questions, and MLE-bench [16], introduced by OpenAI, which evaluates practical ML engineering skills using a collection of 75 coding questions modeled after Kaggle competitions.

To address this gap, we propose HardML, a benchmark specifically designed to evaluate the knowledge and reasoning capabilities of AI models in data science and machine learning. HardML employs the same testing framework as MMLU, comprising multiple-choice questions, with the primary difference being that more than one answer can be correct. It differs in scope from MLE-bench, as it does not test coding capabilities but focuses on theoretical understanding and reasoning skills based on theoretical concepts in DS and ML. HardML uses 100 challenging multiple-choice questions, meticulously crafted over six months to cover the most relevant and contemporary topics in DS and ML. The questions are designed to be difficult even for experienced professionals, such as senior machine learning engineers, thereby ensuring that the benchmark assesses advanced understanding and critical problem-solving skills.

A key aspect of HardML is its emphasis on originality and contemporary relevance, featuring questions that reflect the latest advancements in machine learning from the past two years. To minimize the risk of data contamination—where models might have been trained on benchmark content, leading to artificially inflated performance [13]—we developed primarily original questions. By "original," we mean that while the core concepts required to solve these questions may be known (similar to foundational theorems in mathematics), the specific applications and reasoning required are unique. These questions span topics including natural language processing, computer vision, statistics and statistical modeling, classical machine learning algorithms, and more. In this paper, we also introduce EasyML, a benchmark of 85 multiple-choice questions designed to provide a more accessible and slightly easier set of questions than MMLU-ML for evaluating smaller language models, such as GPT4o-mini [21] and LLaMA models with fewer than 70 billion parameters [22].

Our evaluation of state-of-the-art AI models (o1) [20], reveals a substantial performance gap compared to existing benchmarks [Figure 1]. Specifically, these models exhibit an error rate of approximately 30% on HardML, which is significantly higher than the error rate on the machine learning section of MMLU (MMLU-ML) [5]. This disparity highlights the challenges that current AI models face in mastering the complexities of DS and ML, particularly in understanding nuanced concepts and applying them to non-trivial problems.

The initial motivation behind constructing this benchmark was to generate a comprehensive set of interview-preparation questions for individuals seeking positions in machine learning at leading technology companies (FAANG). However, the interesting results obtained during large language model evaluation, purely out of curiosity, led to the development of this paper. Given the relative scarcity of specialized benchmarks in these fields compared to others like mathematics and physics, we believe HardML fills an important gap and provides a foundation for future research and development.

2. Data collection

The data collection involved a multi-step process spanned over 6 months. As mentioned in the last paragraph of the introduction, the initial purpose of this project was to form a set of question-answer for ML interview assessment for entrance of the top tech companies. These are to be used on the platform getaiquestions.com, which is a website similar to leetcode.com for interview preparation. Therefore, the dataset construction wasn't biased towards building problems that the LLMs wouldn't be able to solve, they were fully intended for human use.

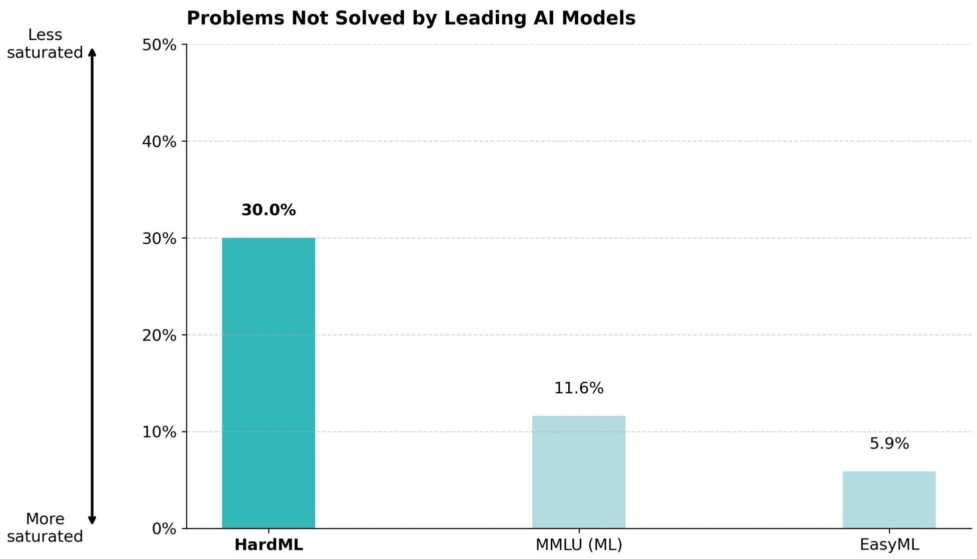


FIGURE 1. Comparison of error rate across 3 DS&ML benchmarks. While existing benchmarks are approaching saturation, HardML keeps an average level above saturation, in line with benchmarks from other fields like MathVista [23] or AIME [24] despite the multiple-choice nature

2.1. **The collection pipeline.** The collection pipeline for the development of HardML and EasyML involved a meticulous 4-step process:

(1) **Raw data collection and scraping.** We have sourced approximately 400 questions from public platforms such as Glassdoor, Blind, Quora, Stack Exchanges, YouTube, as well as from papers and books —including those by Bishop [11]—and from our own writings and public blogs (ptidor.com), among many other reputable sources. As such, we specifically dedicated time in collected ideas from modern sources - very recent interviews on the topic of the latest development in Natural Language Understanding (NLU) or Computer Vision (CV) and collecting ideas from recently published papers (from 2024 and 2023).

Importantly, this sourcing was not limited to simply gathering existing interview questions. Many questions were thoughtfully devised by us, inspired by theoretical concepts presented in books, papers, and online resources. This approach ensured that we had a reasonable amount of questions that were both original and rooted in fundamental principles of data science and machine learning.

(2) **Devising golden solutions and refinement.** In this phase, we crafted definitive "golden" answers for each question, providing clear and accurate solutions. Given that many sourced questions lacked reliable and complete answers, this was a demanding and iterative process that occupied the majority of the six-month development period.

Each question was paired with a golden answer and a list of core ideas—the essential elements required for a respondent to achieve a perfect score. During this stage, we also engaged in refining the questions, which included paraphrasing and restructuring to enhance clarity and coherence. However, to preserve the authenticity of real-world interview scenarios (recall that this was the purpose of this project at that time), not all questions were extensively modified; in some cases, we made only slight adjustments while maintaining the original intent. Upon completion, this raw dataset amounted to an extensive collection exceeding **150** pages (in google docs) of written material.

(3) **Adaptation to Multiple-Choice Format.** This is penultimate step of the process, and it involves transforming the refined dataset of questions, golden answers and core ideas into machine parsable/verifiable input and output. We chose to go with the MMLU (ML) framework of multiple-choice question format, with a small change: at least one answer is correct, instead of exactly one that is correct, hence increasing the difficulty. This required converting the answers and core ideas into a structure where at least one option was correct. This process was non-trivial, as not all questions could be adapted without compromising their essence and level of difficulty. As a result, only about half of the initial questions and answers were successfully transitioned into the multiple-choice format, ensuring that the benchmark remained challenging and faithful to its original purpose.

(4) **Quality control and data contamination prevention.** The is the final step of the process, focusing on rigorous quality assurance and final checks. We meticulously reviewed each multiple-choice question and corresponding answer to ensure accuracy, clarity, and alignment with the benchmark's objectives. This phase involved collaboration with beta testers—colleagues and peers—who interacted with the questions through the user interface (UI) of our platform (that the project was initially intended for), getaiquestions.com.

While no major errors were identified, several ambiguous cases were detected and rectified during this stage, enhancing the robustness and reliability of the final benchmark.

Finally, we conducted a contamination check by evaluating the similarity of our content against existing internet sources to detect potential plagiarism. If any high similarities were identified, the questions and answers were carefully adjusted to ensure originality. Note that this step was applied only to **HardML,** as its purpose is to rigorously assess human ML experts. In contrast, EasyML is intended to test the foundational knowledge and basic reasoning abilities of human entry-level professionals in DS and ML (and potentially smaller language models), and therefore, strict rigor was a secondary consideration.

2.2. **Question difficulty.** The difficulty assignment to each question (between Easy, Medium and Hard) was done by us, as a measure of how difficult a question would appear in our eyes. The author of this paper is a former Lead Machine Learning Engineer with an MSc in AI from The University of Edinburgh with about 5 years of industry experience in machine learning. His research contributions have gathered over 80 citations and his skill set encompasses a broad range of technologies and methodologies, including Python, PyTorch, AWS, GCP, MLOps, distributed computing, and quantitative finance. Most importantly, the author interviewed over 100 candidates throughout his career, driven by a deep passion for interview assessment and a commitment to excellence.

While we acknowledge and don't refute that the difficulty assignments may exhibit slight bias—given that they were determined by a single individual—we have strived to represent the difficulties as accurately as possible. This is substantiated by the results of our benchmark evaluations: HardML yields a significantly higher error rate than MMLU, indicating a higher level of difficulty, whereas EasyML achieves a notably lower error rate. These outcomes corroborate our assessments of the relative difficulties of the benchmarks. HardML comprises only ”hard” questions (according to the categorization system explained above) whereas MMLU-ML –though lacking an official difficulty rating–appears to consist predominantly of ”easy” and ”medium” questions (according to the same categorization system).

3. Dataset composition

The HardML benchmark covers a broad spectrum of contemporary Data Science and Machine Learning spanning from basic data handling methods and classical machine learning to the frontier of Deep Learning and Natural

Language Understanding with state-of-the-art language models and modern training pipelines utilizing tens of thousand of devices.

3.1. **Dataset Statistics.** The distribution over categories is shows in [Table 1]. A comprehensive coverage of topics is essential for effectively evaluating AI systems. Accordingly, the majority of the questions in our benchmark focus on Deep Learning, Natural Language Understanding (NLU), and Computer Vision (CV). This emphasis is intentional and natural, as these fields encompass the most novel approaches and present some of the most challenging questions in contemporary AI research. This distribution is in line with other prominent benchmarks' distributions like FrontierMath [15].

Category	**Percentage**
Deep Learning	33%
Classical Machine Learning	21%
Natural Language Understanding	15%
Data Engineering	11%
Computer Vision	11%
Statistics & Statistical Modeling	9%

Table 1. Percentage distribution of DS and ML sub-fields in the HardML dataset, representing the proportion of each classification relative to the total amount.

3.2. **Comparison to related benchmarks.** HardML differs from the baseline MMLU benchmark in both size—being slightly smaller—and format: each question in HardML may have more than one correct answer. This multi-answer format also sets it apart from MLE-bench, which focuses on coding tasks with a definite answer rather than multiple-choice questions. A detailed comparison of the various datasets used in the research space for LLM evaluation in machine learning is presented in Table 2.

Dataset	**Size**	**Type**
HardML (this paper)	100	multiple-choice
EasyML (this paper)	85	multiple-choice
MMLU [ML subset, test]	112	multiple-choice
MLE-bench (OpenAI)	75	coding

Table 2. Comparison between datasets available in the research space for LLM evaluation in the field of DS and ML.

3.3. **Sample questions from HardML.** In order to accurately provide an intuition of the level of difficulty and form of the questions from HardML, we display below a few examples.

Sample problem 1

Question: You want to train a LLM that can solve challenging math problems properly. To do that, you employ a team of mathematicians to devise problems and solutions for training data. Unfortunately, you require a lot of training data, naturally, and hence you have to employ thousands of people to generate problems and solutions for your LLM. You need some form of quality control to understand if the mathematicians keep an overall good quality and that your LLM won't be trained on corrupted data. You can assume you have 1000 people devising (problem, solution) tasks, one person submits one task. Each task is rated from 5 choices, from 1/5 (lowest) to 5/5 (highest): 1/5,2/5,3/5,4/5,5/5. You want these people to produce, on average, a quality of work of at least 4/5=0.8 and to be 95% sure that is the case. You cannot check all 1000 and compute the average yourself because that would defeat the purpose of employing these people in the first place, so then what's the minimum number N of random tasks you would need to check? For this exercise, you can assume that the task grades follow a normal distribution and the variance of the overall quality is known and it's the maximum it can be, given the range 1/5-5/5. Make sure to normalize the grades in [0.2,1]

A) 4
B) 6
C) 7
D) 8

Answer: B

Sample problem 2

Question: You measure Model FLOPs Utilisation (MFU) by counting all floating point operations in the entire training step—including overhead—and dividing by (time elapsed)×(theoretical hardware FLOPS). You now enable activation (gradient) checkpointing, which re-runs parts of the forward pass to save memory. Assuming you still count all FLOPs and include the extra recomputations in your total, what will happen to your measured MFU?

A) MFU will strictly increase, because you are performing additional FLOPs without proportionally more time.
B) MFU will strictly decrease, because the added time from redoing the forward pass dominates.
C) MFU will remain exactly the same, because both FLOPs and time scale in a fixed ratio.
D) The effect on MFU is ambiguous; you are doing more FLOPs but also increasing the total step time, so the ratio could go up or down.

Answer: D

Sample problem 3

Question: A T5 or FlanT5 model is considered one of the best encoder-decoder models out there (as of 2024). Why aren't these commonly used at scale to train large language models (LLMs) that compete with GPT-4? Select all that apply.

A) The architecture of FlanT5 makes it harder to scale.
B) Decoder-only models allow for simpler partitioning strategies, such as splitting along head dimensions, resulting in more balanced compute, memory, and network costs.
C) T5 is like a sequence of blocks but with more edges representing more complicated data dependencies during compute.
D) The communication between encoder and decoder in encoder-decoder models complicates network architecture and scaling strategies.

Answer: A, B, C, D

Sample problem 4

Question: What is the difference between L2 regularization and weight decay in the context of neural networks, and under which conditions can they be considered equivalent?

A) L2 regularization adds a penalty to the loss function proportional to the sum of squared weights, while weight decay multiplies the weights by a factor slightly less than 1 after each update.
B) L2 regularization and weight decay are always equivalent, regardless of the optimizer used.
C) L2 regularization and weight decay are equivalent only when using stochastic gradient descent (SGD) as the optimizer.
D) Using optimizers like Adam or RMSprop breaks the equivalence between L2 regularization and weight decay.

Answer: A, C, D

Sample problem 5

Question: The backpropagated gradient through a tanh non-linearity is always smaller or equal in magnitude than the upstream gradient.

A) True.
B) Depends on the sign of the inputs.
C) False
D) True only if all the input units are in (-1,1)

Answer: A

Sample problem 6

Question: Where is the temperature applied in the model architecture of Chat GPT-3 or 4?

A) At the input level.
B) After the softmax layer.
C) Right before the softmax layer.
D) At beam-search level when we select the output token based on probability.

Answer: C

We intentionally designed the benchmark to assess fairly up-to-date advancements in AI, as exemplified by questions **2, 3, and 6**. Additionally, we included both reasoning-intensive questions—such as question **1**, which requires code implementation or meticulous hand calculations, and question **5**, which tests comprehension through a comparison between the hyperbolic tangent function (tanh) and its derivative—as well as knowledge-intensive questions like question **4**, which addresses a subtle nuance in the mathematical formula for weight decay and the formula of popular optimizers.

4. Results

4.1. **Accuracy on HardML.** We evaluated 5 leading language models and 1 leading smaller language model (gpt-4o-mini) on our HardML dataset. Due to limited resources and ease of use, we decided to stick only with models from OpenAI and Anthropic, we believe these are enough to convey a good assessment. For instance, o1 is in top 5 in Chatbot Arena LLM Leaderboard from lmarena.ai. The results are present in [Figure 2]. We used the same prompt and batch size for these experiments.

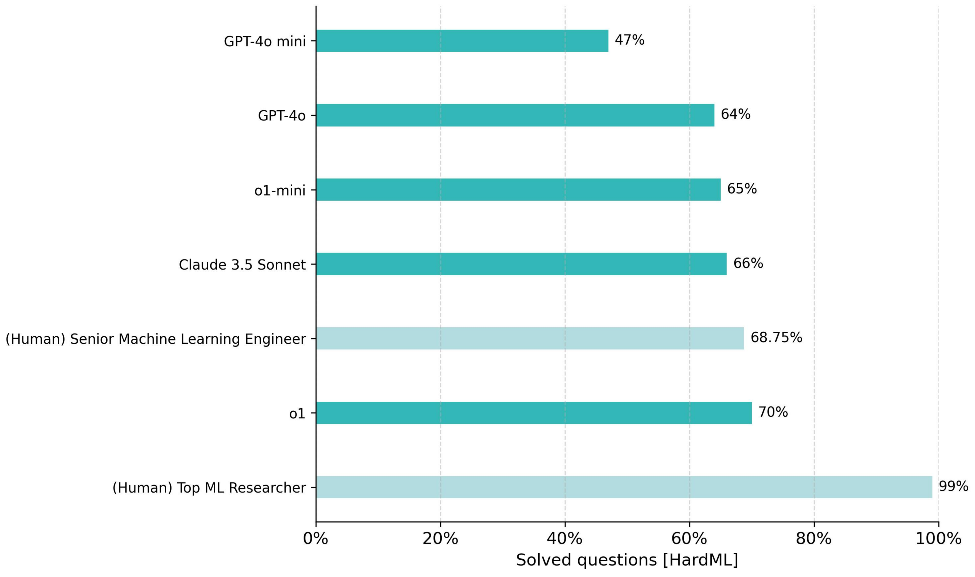


Figure 2. Solved questions in HardML

Based on a single evaluation of the full benchmark, we found that most models solve aout **65%** of the questions with the top performing model (o1) being able to solve **70%**. This is in line with other benchmarks from other fields. For example, in math, current benchmarks that are considered 'hard' like Omni-Math, MathVista and Aime have around **60%**, **70%** and around

70% respectively in accuracy against o1. Interestingly, if we allow a ”soft” figure for solved questions (giving partial credit when the model’s answer is a subset of the correct answer), then the performance goes up by **5** percentage points (to **75.08%** for o1), not a drastic change.

The models are very close together in performance, the precise ordering of model performance should be interpreted with significant caution as multiple runs could switch a few places around. We ran 2 times to make sure the order at the top is consistent, in both, o1 demonstrated the strongest performance.

An interesting observation is that even when the model arrives at a correct result, the underlying reasoning may not be accurate. We made the models output a reasoning field in the output json to observe this behaviour. For example, GPT-4o sometimes selects numerical answers because they are intuitively closer to an expected magnitude (like choosing 7 over 8 because it is smaller), rather than deriving them through rigorous proof. This illustrates a natural limitation of the multiple-choice format—scores can be artificially inflated due to luck or educated guesses that do not reflect true understanding.

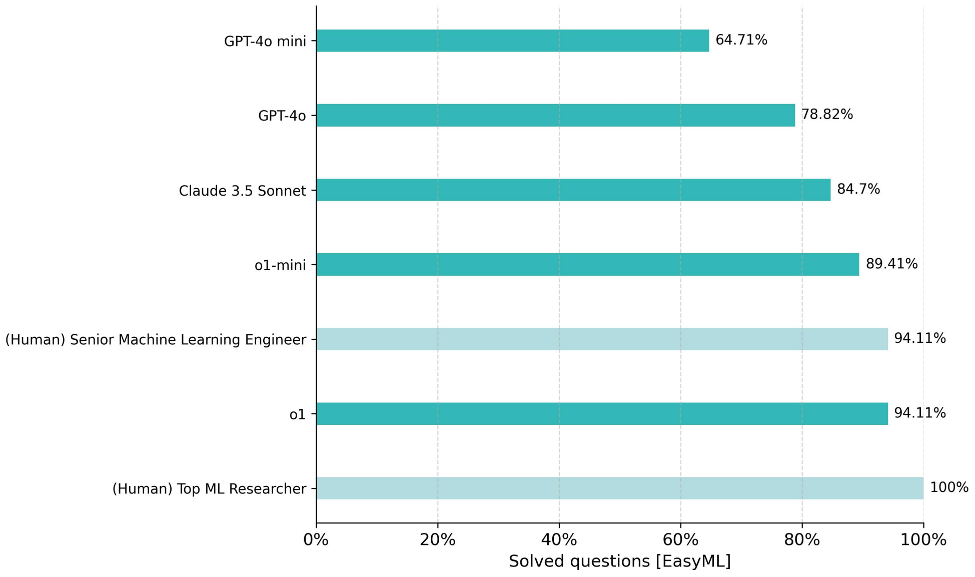


FIGURE 3. Solved questions in EasyML

4.2. **Human performance on HardML.** Examining human performance is particularly insightful, given that the initial intention of this project was to assess candidates during interviews or to filter applicants competing for positions in major technology companies. The results displayed in Figure 2 include human scores for reference. Below, we explain how these human scores were calculated:

A) The first metric (Senior Machine Learning Engineer) was obtained during the beta testing phase of data collection (the final step). We invited actual senior machine learning engineers—friends of the author (see acknowledgements)—to participate in several quizzes consisting of 7 to 8 questions each, sampled, at random, from HardML. Once sampled, the same quizzes were used for each person, we managed to assess 5-6 quizzes per person. After aggregating the results, we found that an individual scored, on average, approximately 5.5 correct answers out of 8 (which translates into 68.75%). Although it is not reflective of the performance on all the questions from HardML (only on a subset), we believe this figure is relevant to be shown. Hence, this performance is reflected in Figure 2 and Figure 3. The participants expressed admiration for the benchmark, noting that the questions required significant thought and were highly challenging.

B) The second number (Top ML Researcher) is purely the author's opinion. Even though we did not have access to a globally recognized top machine learning researcher, we posit that this benchmark would not represent a significant challenge for individuals actively engaged in cutting-edge ML research and who have been at the forefront of the field for the past 20 years.

4.3. **Accuracy on MMLU and EasyML.** Below, we have the equivalent diagram (Figure 4) for the 112 questions present in the testset of MMLU (ML subset) and our proposed EasyML. Observe how o1 is still the top performer, but the scores are significantly higher compared to HardML. Note that, we have not displayed human assessment figures on the MMLU benchmark as this experiment wasn't conducted.

5. Related work

The evaluation of large language models (LLMs) has been extensively explored across various domains, leading to the development of numerous benchmarks that assess different aspects of AI capabilities. In this section, we review the most relevant benchmarks and studies related to our work, focusing on those that evaluate LLMs in the context of machine learning and data science and briefly mentioning a few impressive pieces of work on other fields from STEM.

5.1. **Multitask Language Understanding Benchmarks.** The Massive Multitask Language Understanding (MMLU) benchmark introduced by Hendrycks

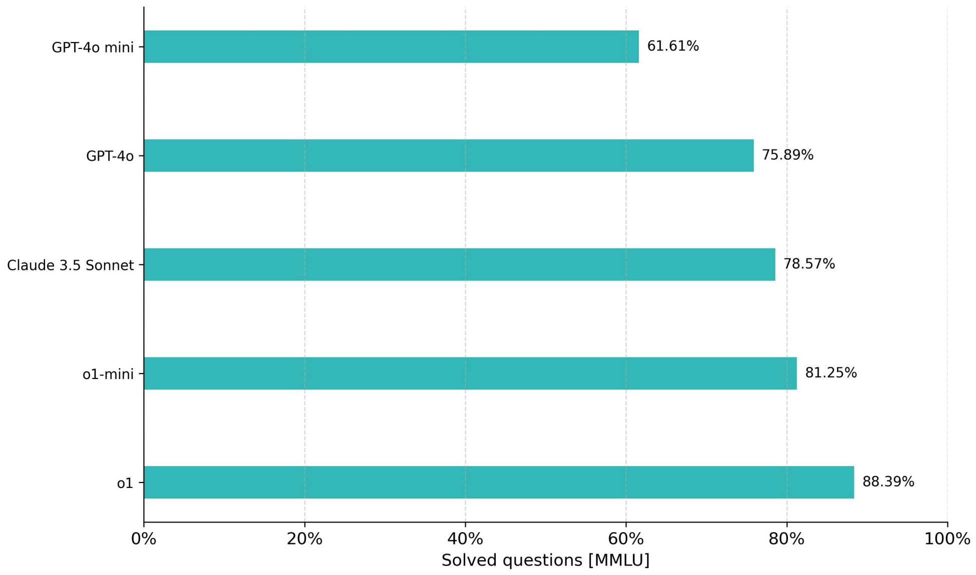


Figure 4. Solved questions in MMLU [ML]

et al. [5] has been a significant step toward assessing the broad academic and professional knowledge of LLMs. MMLU covers 57 subjects across STEM, humanities, social sciences, and more, including a subset dedicated to machine learning with 112 multiple-choice questions. While MMLU has provided valuable insights into the capabilities of models like GPT-3, state-of-the-art models have begun to approach saturation on several subjects, including ML. This near-ceiling performance limits the benchmark's effectiveness in distinguishing the advanced capabilities of newer models. Moreover, the ML subset, due to its relatively small size and scope, may not fully capture the depth and complexity required to evaluate nuanced understanding in ML and DS.

5.2. **Benchmarks for Advanced Reasoning.** To address the limitations of existing benchmarks in measuring advanced reasoning, FrontierMath [15] was introduced as a benchmark comprising exceptionally challenging and original mathematical problems. These problems span major branches of modern mathematics and are designed to require significant effort from expert mathematicians—often multiple hours or days—to solve. FrontierMath effectively minimizes data contamination by using unpublished problems and employs automated verification for reliable evaluation. Remarkably, current AI models solve under 2% of the problems, highlighting a substantial gap between AI capabilities and human expertise in advanced mathematics. This benchmark underscores the importance of creating future-proof evaluations that remain challenging despite rapid advancements in AI. This paper is the inspiration for HardML, we were impressed by it and wanted to replicate some of the work.

5.3. **Practical Machine Learning Engineering Benchmarks.** In parallel, MLE-bench was proposed by Chan et al [16] as a benchmark to evaluate AI agents' performance in machine learning engineering tasks. MLE-bench curates 75 ML engineering-related competitions from Kaggle, encompassing tasks that require practical skills such as data preprocessing, model training, and experimental analysis. By establishing human baselines based on Kaggle's publicly available leaderboards, MLE-bench provides a real-world context for assessing AI agents in practical engineering scenarios. The benchmark evaluates AI setups like OpenAI's o1-preview with AIDE scaffolding, noting that the best-performing agent achieves a bronze medal level in approximately 17% of competitions.

5.4. **Automated Answering and Generation of ML Exams.** In the realm of educational assessments, other researchers explored the automatic answering and generation of machine learning final exam questions in their work titled "From Human Days to Machine Seconds: Automatically Answering and Generating Machine Learning Final Exams." [25] They demonstrated that large language models could pass ML final exams at a human level and generate new exam questions rapidly. Their study focused on the differences between final exams and problem sets, noting that final exams typically have longer, multi-part questions that span a broader set of topics and require more complex reasoning. Notably, in this paper, multiple-choice questions were generated and tested, making it a valuable related benchmark that is, in our opinion, underexplored.

5.5. **Comparison to Our Work.** Our proposed HardML benchmark fills an important gap in existing evaluations by providing a rigorous, modern, and challenging testbed specifically tailored to data science and machine learning. Unlike MMLU's ML subset, HardML offers a more difficult, more diverse and more up-to-date set of questions that delve deeper into advanced topics. In contrast to MLE-bench, which assesses practical engineering skills through coding tasks, HardML focuses on theoretical understanding and the ability to reason about complex concepts.

By emphasizing originality and minimizing data contamination, similar to FrontierMath, we ensure that HardML remains a relevant and challenging benchmark for current and future AI models. Additionally, by including EasyML as a complementary benchmark for evaluating smaller language models, we address the need for scalable evaluations across different model sizes and capabilities. It is challenging to ascertain the long-term applicability of HardML; however, we anticipate that it will remain relevant at the cutting edge of model evaluation for at least one year.

6. Limitations

Even though HardML currently demonstrates reasonable resistance to saturation, we do not believe this resilience will persist for much longer. Models like o3 [26] have already shown improvements over previous frontier models such as o1, and the pace of advancement in AI systems is exceedingly rapid. One of the significant limitations of HardML is its multiple-choice format, which allows for "guesses" or "educated guesses" that can artificially inflate scores—a limitation that has been critically examined in FrontierMath. In benchmarks like MMLU [ML], where only one answer is correct per question, a random guess has a $\frac{1}{4}$ chance of being correct. In comparison, in a multiple-choice format where more than one answer can be correct, a random guess has a probability as high as $\frac{1}{15}$. These probabilities are still substantial, potentially diminishing the benchmark's ability to effectively discriminate between true understanding and chance performance.

Therefore, it is essential to develop benchmarks with automatic evaluations that require machine-verifiable outputs, such as numerical or boolean answers. This approach reduces the likelihood of inflated scores due to guessing. Benchmarks like MLE-bench, which necessitate code implementation or involve advanced mathematical reasoning to arrive at the correct solution—while still being related to data science and machine learning—are exemplary in this regard.

Constructing challenging multiple-choice questions is particularly difficult because adept humans or advanced AI models can employ elimination strategies to identify the correct answers. This means that even if the correct answers are difficult to determine, the benchmark's effectiveness can collapse if the incorrect options are not **equally challenging** to dismiss. Consequently, every answer choice must be nuanced and not obviously incorrect. Achieving this level of subtlety in question design is exceptionally demanding and was a primary reason why the development of this benchmark required such a significant investment of time. Crafting answers that appear plausible yet are subtly incorrect is a skill in itself.

7. Acknowledgements

Special thanks to: Robin Kahlow (Senior ML Engineer at RunwayML), Geo Badita (Senior Software Engineer at Meta), and Chady Dimachkie (Head of ML at Abwab.ai and former Deep Learning Engineer at Nvidia) for taking the challenge and attempting HardML very thoroughly, as well as providing invaluable feedback. Their expertise and rigorous assessments have been instrumental in refining the dataset and validating its efficacy. Special thanks to Paul Chelarescu for his invaluable assistance in curating and organising the raw database in the first step of data collection, which served as the foundation for this work.

8. Conclusion

With this paper, we instigate to further research in the area of LLM benchmarking for cutting edge Data Science and Machine Learning. The dataset of HardML is present in an interactive environment on getaiquestions.com and can also be obtained in clean json format for experiment replication or further research here. Our work contributes to the ongoing efforts to develop benchmarks that can effectively measure and distinguish the advanced capabilities of AI models in rapidly evolving fields.

References

[1] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT. Association for Computational Linguistics.

[2] Brown, T. B., Mann, B., Ryder, N., et al. (2020). Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems, 33, 1877–1901.

[3] Wang, A., Singh, A., Michael, J., et al. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In Proceedings of the EMNLP Workshop. Association for Computational Linguistics.

[4] Raffel, C., Shazeer, N., Roberts, A., et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 21(140), 1–67.

[5] Hendrycks, D., Burns, C., Basart, S., et al. (2021). Measuring Massive Multitask Language Understanding. In Proceedings of the International Conference on Learning Representations (ICLR).

[6] Hendrycks, D., Burns, C., Kadavath, S., et al. (2021). Measuring Mathematical Problem Solving with the MATH Dataset. In Advances in Neural Information Processing Systems.

[7] Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2019). Analysing Mathematical Reasoning Abilities of Neural Models. In International Conference on Learning Representations (ICLR).

[8] Huang, K., Altosaar, J., & Ranganath, R. (2020). ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. arXiv preprint arXiv:1904.05342.

[9] Provost, F., & Fawcett, T. (2013). Data Science and its Relationship to Big Data and Data-Driven Decision Making. Big Data, 1(1), 51–59.

[10] Jordan, M. I., & Mitchell, T. M. (2015). Machine Learning: Trends, Perspectives, and Prospects. Science, 349(6245), 255–260.

[11] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[12] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

[13] Dodge, J., Ilharco, G., Schwartz, R., et al. (2021). Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In Proceedings of the 2021 EMNLP Workshop on Datasets and Benchmarks. Association for Computational Linguistics.

[14] He, T., Singh, M., Achiam, J., et al. (2020). Translatotron: An End-to-End Speech-to-Speech Translation Model. arXiv preprint arXiv:1904.06037.

[15] Glazer, E., Erdil, E., Besiroglu, T., Chicharro, D., Chen, E., Gunning, A., Olsson, C. F., Denain, J.-S., Ho, A., de Oliveira Santos, E., Järviniemi, O., Barnett, M., Sandler, R., Vrzala, M., Sevilla, J., Ren, Q., Pratt, E., Levine, L., Barkley, G., Stewart, N., Grechuk, B., Grechuk, T., Enugandla, S. V. V., & Wildon, M. (2024). FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI. *arXiv preprint arXiv:2411.04872*. Retrieved from https://doi.org/10.48550/arXiv.2411.04872.

[16] Chan, J. S., Chowdhury, N., Jaffe, O., et al. (2024). MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. arXiv preprint arXiv:2410.07095.

[17] OpenAI. (2024). GPT-4o System Card. Retrieved from https://cdn.openai.com/gpt-4o-system-card.pdf

[18] Anthropic. (2024). Introducing Claude. Retrieved from https://www.anthropic.com/news/introducing-claude

[19] OpenAI. (2024). Hello GPT-4o. Retrieved from https://openai.com/index/hello-gpt-4o/

[20] OpenAI. (2024). Introducing OpenAI o1. Retrieved from https://openai.com/index/introducing-openai-o1-preview/?utmsource=chatgpt.com

[21] OpenAI. (2024). GPT-4o Mini: Advancing Cost-Efficient Intelligence. Retrieved from https://openai.com/blog/gpt-4o-mini-advancing-cost-efficient-intelligence

[22] Meta AI. (2024). Introducing Meta Llama 3: The most capable openly available LLM to date. Retrieved from https://ai.meta.com/blog/llama-3/

[23] Lu, Pan et al. (2024). MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts. arXiv:2310.02255 [cs.CV]. URL: https://arxiv.org/abs/2310.02255.

[24] (2024c). Learning to Reason with LLMs. URL: https://openai.com/index/learning-to-reason-withllms/.

[25] Drori, I., Zhang, S. J., Shuttleworth, R., et al. (2022). From Human Days to Machine Seconds: Automatically Answering and Generating Machine Learning Final Exams. arXiv preprint arXiv:2206.05442. Retrieved from https://doi.org/10.48550/arXiv.2206.05442.

[26] Pfister, R., & Jud, H. (2025). Understanding and Benchmarking Artificial Intelligence: OpenAI's o3 Is Not AGI. arXiv preprint arXiv:2501.07458. Retrieved from https://arxiv.org/abs/2501.07458.

Canary Wharf, London, United Kingdom
Email address: `tidor@madsimpleads.com`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXIX**, Number 2, 2024
DOI: 10.24193/subbi.2024.2.05

ELECTRIC VEHICLE ROUTING PROBLEM: A REVIEW OF RECENT APPROACHES AND ALGORITHMS

YINGKAI XU

Abstract. With the rapid advancement of new energy vehicles, electric vehicles (EVs) have become integral to modern transportation systems. Compared with traditional fuel vehicles, EVs are limited by their limited battery capacity and require reasonable charging planning to complete the designated routes efficiently. Therefore, the effective routing of EVs has emerged as a critical research focus in transportation and logistics. This study comprehensively reviews recent advancements in the Electric Vehicle Routing Problem (EVRP) over the past three years. First, the concepts of EVRP are introduced. Then, the problem is classified according to energy consumption models, charging strategies, and constraints. Subsequently, various algorithms employed in these studies are analyzed and summarized. Finally, based on the current state of development in this field, the main challenges faced by EVRP and future research directions are discussed.

1. Introduction

In recent years, greenhouse gas emissions have gained global attention as a critical environmental issue. According to statistics from the European Union, carbon dioxide emissions from road transport contribute approximately one-fifth of the EU's total emissions [1]. In response to climate change, the European Parliament enacted the European Climate Act, which endorses the European Commission's proposal to achieve zero carbon emissions for cars and trucks by 2035 [2]. In this context, logistics distribution, a vital component of urban road transport systems, has increasingly embraced electric vehicles (EVs) as a key strategy to mitigate carbon emissions.

Schneider et al. [38] extended the Vehicle Routing Problem (VRP) by incorporating constraints on time windows and recharging and proposed a Mixed-Integer Programming (MIP) model. This study represents a significant step in optimizing Electric Vehicle Routing Problem (EVRP). Since then, with the rapid advancement of the electric vehicle industry, research on EVRP has significantly increased. To

Received by the editors: 27 January 2025.
2010 *Mathematics Subject Classification.* 90B06, 90C11, 90C59.
1998 *CR Categories and Descriptors.* G.1.6 [**Optimization**]; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods.
Key words and phrases. Electric vehicle routing problem, Classification, Literature review.

systematically explore the evolution and research directions within the EVRP domain, several scholars have conducted comprehensive literature reviews and analyses [34, 51, 22, 40, 48, 19]. Among these, Ye et al. [51] conducted a classified review of 110 studies, categorizing EVRP research. In contrast, Kucukoglu et al. [22] provided a comprehensive review of 136 papers across five key dimensions: objective functions, energy consumption models, constraints, fleet configurations, and solution methodologies. However, existing review studies primarily focus on research published before 2022, and there is a notable lack of systematic reviews covering EVRP developments over the past three years. Therefore, the present study conducts an in-depth review of recent EVRP research from 2022. A total of 42 papers from journals with an impact factor greater than 4 were selected (to ensure high-quality, impactful research and a feasible review scope). This study aims to provide a comprehensive literature review of high-quality research on EVRP conducted over the past three years. First, the fundamental concepts of EVRP are outlined. Then, the objective functions adopted in recent studies are reviewed, and the EVRP are categorized based on three dimensions: energy consumption calculation, charging strategies, and constraints. Subsequently, various solution algorithms proposed in recent studies are analyzed in depth, and their characteristics are summarized. Finally, this field's current state of the art is summarized, and future research directions and potential challenges are presented.

This paper is organized as follows: Section 2 introduces EVRP. Section 3 reviews and categorizes the relevant literature from various perspectives within the scope of this study. Section 4 explores the solution approaches for EVRP. Section 5 discusses a comparison of standard algorithms and provides future research directions. Finally, Section 6 concludes the paper.

2. Electric vehicle routing problem

The EVRP aims to optimize routes for a fleet of EVs, ensuring that all customer nodes are served while minimizing operational costs. Each route starts and ends at a designated depot, and EVs must comply with constraints such as battery capacity limits and time windows[54, 17]. Here, we present the mathematical formulation of the EVRP [38]. Let $V = \{1, 2, \ldots, n\}$ be the set of customer nodes, with nodes 0 and $n + 1$ representing the initial and final depots. Define $V_0 = V \cup \{0\}$ and $V_{N+1} = V \cup \{n+1\}$. Let F be the set of charging stations and F' be the set of dummy nodes required to allow multiple visits to charging stations. The extended sets are defined as $V' = V \cup F', V'_0 = V' \cup \{0\}, V'_{N+1} = V' \cup \{n+1\}, V'_{0,N+1} = V' \cup \{0, n+1\}$. A fleet of homogeneous EVs K is considered. Each EV $k \in K$ travels between nodes $i, j \in V'_{0,N+1}$, with distance d_{ij}, energy consumption rate h, and battery capacity Q. Let x^k_{ij} be a binary variable equal to 1 if the vehicle k travels from the node i to the node j and 0 otherwise; y^k_i be the decision variable used to track the battery level of the vehicle k when it reaches node i. The MIP model for the EVRP is described as follows:

$$\min \sum_{i \in V'_0} \sum_{j \in V'_{n+1}} \sum_{k \in K} d_{ij} x^k_{ij} \tag{1}$$

$$\sum_{j \in V'_{n+1}} \sum_{k \in K} x_{ij}^k = 1, \quad \forall i \in V \tag{2}$$

$$\sum_{j \in V'_{n+1}} \sum_{k \in K} x_{ij} \leq 1, \quad \forall i \in F' \tag{3}$$

$$\sum_{j \in V'} x_{0j}^k \leq 1, \quad \forall k \in K \tag{4}$$

$$\sum_{i \in V'_{n+1}} x_{ji}^k = \sum_{i \in V'_0} x_{ij}^k, \quad \forall j \in V', \forall k \in K \tag{5}$$

$$y_j^k \leq y_i^k - (h \cdot d_{ij}) x_{ij}^k + Q(1 - x_{ij}^k), \quad \forall i \in V, \forall j \in V'_{N+1}, \forall k \in K \tag{6}$$

$$y_j^k \leq Q - (h \cdot d_{ij}) x_{ij}^k, \quad \forall i \in F' \cup \{0\}, \forall j \in V'_{N+1}, \forall k \in K \tag{7}$$

$$y_0^k \leq Q, \quad \forall k \in K \tag{8}$$

The objective function (1) aims to minimize the total distance of electric vehicles. Constraint (2) handle the connectivity of the customer nodes. Constraint (3) ensure that each dummy charging station can be visited at most once. Constraint (4) make sure that each electric vehicle can be used only in one route plan. Constraint (5) ensure that the total number of outgoing arcs is equal to the total number of incoming arcs at customer and charging station nodes, which provides continuity in the routes. Constraints (6)-(8) specify the battery level of an electric vehicle and ensure that it never falls below 0.

3. Classifications of the EVRP

The EVRP is formulated to address real-world logistics distribution needs, thus involving multiple constraints and problem variants in different application scenarios. In order to systematically sort out the research framework of EVRP, this section classifies and summarizes the problem from multiple perspectives, including the objective function, energy consumption model, charging strategies, and constraints.

3.1. **Objective function.**

The objective function is the core component of the EVRP model, directly determining the direction of the optimization problem. This section categorizes EVRP based on commonly used objective functions in the literature. From the collected studies, we classified and summarized 13 common optimization objectives for EVRP:

(1) Total travel distance
(2) Total travel time
(3) Total number of vehicles used
(4) Total energy consumption
(5) Total fixed costs
(6) Total penalty cost
(7) Total recharging cost, recharging time or swapping battery cost
(8) Total waiting time for electric vehicles at charging stations
(9) Total delivery cost

(10) Battery degradation costs
(11) Costs of carbon emissions
(12) Customer service costs
(13) Other costs

In VRP, commonly considered objective functions include environmental costs, travel distance, and travel time [20]. By analyzing Table 1, it can be observed that EVRP shares common objective functions with traditional VRP but also exhibits unique characteristics specific to EVs. Among these, objectives (1), (2), and (3) are more common in both VRP and EVRP studies, which mainly focus on the essential factors of path optimization, such as the minimization of travel distance, travel time, and the number of vehicles used. In contrast, objective (4) highlights the characteristics of EV batteries, which have become one of the core topics in EVRP research. Furthermore, compared to traditional fuel-powered vehicles, the energy replenishment process of EVs is considerably slower. Consequently, optimizing charging time (objective (7)) has emerged as a crucial research focus in EVRP, aiming to meet routing requirements while enhancing delivery efficiency and reducing operational costs.

3.2. **Energy consumption calculation.** Energy consumption models can generally be categorized into two types: simple linear models that directly correlate energy consumption with travel distance or travel time and nonlinear models based on vehicle driving power and terrain load, as discussed in Lera-Romero et al. [24], Fan et al. [15], Xiong et al. [49], Kim and Chung [21], Ren et al. [35], Wang et al. [43], Amiri et al. [5], Ma et al. [28]. Unlike linear models, nonlinear models provide a more comprehensive representation of the complex factors influencing vehicle operations.

Goeke and Schneider [16] introduced key factors such as air resistance, rolling resistance, and gravitational force into energy consumption modelling, converting these resistances into mechanical power and proposing a nonlinear approach to quantify energy consumption. Lera-Romero et al. [24], Fan et al. [15], Xiong et al. [49], Fan [14] conducted EVRP studies based on this model. Among them, Xiong et al. [49] believes that the drivetrain of an EV will lead to a certain amount of energy loss in the process of converting battery energy into wheel torque. So the original model was improved by considering the loss of the driveline.

Ren et al. [35] explicitly incorporated time integration to account for dynamic variations in speed and acceleration while also integrating factors such as departure time, travel speed, travel distance, and load. This comprehensive approach enhances the model's applicability to real-world scenarios. Furthermore, Ma et al. [28] extended energy consumption models by incorporating terrain factors, motor power losses, driving resistance, and energy consumption associated with acceleration and deceleration, thereby improving the model's accuracy and reliability. In the solid waste management context, Peña et al. [32] refined energy models by extending traditional mechanical power calculations. Their approach accounts for energy use during waste loading, compaction, unloading, and regeneration during crushing, addressing gaps in prior research and improving model comprehensiveness for waste management applications.

Table 1. Objective functions of EVRP (Numbers 1-13 correspond to the common objective functions, ✓indicates the presence of a corresponding objective function in the study).

Paper	1	2	3	4	5	6	7	8	9	10	11	12	13
Jia et al. [18]	✓												
Peña et al. [32]	✓												
Zhou et al. [54]	✓			✓	✓								
Kim and Chung [21]				✓									
Fan et al. [15]		✓			✓		✓				✓	✓	
Woo et al. [44]		✓		✓									
Ouyang and Wang [31]				✓	✓	✓							
Ren et al. [35]				✓									✓
Yao et al. [50]		✓	✓				✓						✓
Zhou et al. [55]	✓							✓					
Duman et al. [12]	✓												
Bezzi et al. [6]				✓									
Zhang et al. [52]					✓	✓	✓		✓				
Wang et al. [43]			✓		✓	✓	✓					✓	
Wang et al. [42]	✓												
Rodríguez-Esparza et al. [36]	✓												
Moradi and Boroujeni [30]	✓		✓			✓							
Liu et al. [25]		✓											
İslim and Çatay [17]				✓						✓			
Comert and Yazgan [10]	✓		✓	✓				✓					✓
Cai et al. [7]	✓												
Xiao et al. [46]	✓												
Xia et al. [45]	✓												
Qian et al. [33]	✓					✓	✓						
Dong et al. [11]	✓		✓				✓						
Sadati et al. [37]	✓		✓										
Ma et al. [29]			✓	✓		✓					✓		✓
Longhitano et al. [27]				✓		✓				✓			
Erdem et al. [13]	✓					✓	✓						
Amiri et al. [5]						✓	✓						✓
Agrali and Lee [3]	✓												
Wang and Zhao [41]	✓				✓								
Lera-Romero et al. [24]		✓											
Fan [14]		✓			✓		✓					✓	
Zhou and Zhao [53]	✓		✓			✓	✓						✓
Xiao et al. [47]				✓	✓		✓						
Ma et al. [28]	✓	✓		✓		✓							
Xiong et al. [49]				✓									
Souza et al. [39]	✓												✓
Liu et al. [26]	✓		✓				✓						
Lam et al. [23]	✓						✓						
Çatay and Sadati [8]				✓									✓

3.3. **Charging strategy.** Energy replenishment of EVs can be implemented in three methods: wired charging, wireless charging, and battery swapping. In early research, wired charging was considered the primary method for replenishing the energy of EVs [38]. Although research has expanded into various charging strategies, wired charging remains the most widely adopted method. Excluding battery swapping, charging methods can generally be divided into two categories: full charging and partial charging. Under the full-charge strategy, the EV will fully charge the battery at a charging station [21, 55, 12, 43, 30, 25, 17, 46, 45, 11, 28, 49, 23]. In contrast, the partial charging strategy allows vehicles to terminate charging and leave the charging station once sufficient energy has been acquired to complete the next segment of the journey [15, 31, 6, 42, 10, 37, 27, 13, 5, 3, 41, 47, 14].

Since EVs require some time to charge at charging stations, some researchers have proposed battery swapping as an alternative strategy [35, 52, 7, 33, 29, 53, 39, 26, 8]. In this approach, EVs can swiftly replace their depleted batteries with fully charged ones upon arrival at swapping stations, thereby enhancing operational efficiency in logistics and reducing costs. Meanwhile, some researchers believe that wireless charging technology also effectively reduces the waiting time during the charging process by incorporating it into the EVRP model [35, 31, 4]. Based on the principle of inductive power transfer, wireless charging technology enables EVs to recharge without requiring physical connectors [9]. A key advantage of this technology is its capability to facilitate dynamic charging while the vehicle is in motion.

Furthermore, to improve the accessibility of EV charging and reduce infrastructure costs, researchers have redirected their efforts toward mobile energy replenishment solutions [47, 35, 8, 52]. In this paradigm, dedicated mobile energy vehicles can travel to the location of EVs to provide on-site charging services [47] or battery swapping services [35, 8, 52], thereby alleviating the limitations of the inflexible layout of traditional charging stations.

3.4. **Constraints of the EVRP.** The EVRP involves a range of complex constraints arising from the unique characteristics of EVs and the practical demands of their real-world deployment. In addition to vehicle load and battery capacity limitations, commonly addressed constraints include time windows, pickup and delivery operations, multi-depot configurations, and open and closed routing constraints. This section categorises and summarises the literature concerning these common constraints.

3.4.1. *Time windows.* In the context of EVRP, time constraints can be categorized into hard and soft time windows depending on the degree of flexibility allowed. Hard time windows, which are time constraints currently used in recent studies [54, 35, 55, 12, 52, 30, 17, 7, 46, 33, 37, 13, 3, 41, 47, 26, 23, 8], impose strict time constraints that require the service to be completed within a predetermined window. On the other hand, soft windows provide some flexibility, allowing for slight deviations from the designated schedule; however, exceeding the allowed time window incurs penalty costs. This type of constraint has been gaining increasing attention in recent research [31, 42, 5, 28]. To further enhance customer satisfaction, Zhang et al. [52] proposed the multiple prioritized time windows model, which enables customers to specify one or more prioritized time slots in advance. In addition, Zhou and Zhao [53] introduced the concept of mixed time windows, classifying each delivery point's time constraints into the expected time window and the acceptable time window. Deliveries made within the expected time window incur no penalties, whereas those within the acceptable time window are subject to penalty costs.

3.4.2. *Pickup and Delivery.* In most EVRP models, the primary role of EVs is to deliver goods. For instance, Duman et al. [12] proposed the Flexible Delivery EVRP, an extension of the traditional delivery-based EVRP. In this model, each customer can be associated with multiple delivery locations, each with a corresponding time window. EVs are dispatched from a centralized depot, and deliveries are completed at the customer's pre-specified locations within the predetermined time window. However, in real-world logistics operations, customer demands can generally be categorized into three types: pickup, delivery, or both pickup and delivery. When EVs must simultaneously accommodate pickup and delivery requests, the problem is the EVRP with Pickup and Delivery. Relevant studies in this domain include [31, 55, 46, 3, 26]. Notably, Agrali and Lee [3] explored an innovative pickup and delivery model by introducing transhipment nodes, enabling the efficient transfer and reallocation of goods across different delivery routes.

3.4.3. *Multiple Depots.* The configuration of multiple depots makes path planning more reductive to actual logistics scenarios, where vehicles can depart from multiple depots and return after completing the assigned tasks. This model has significant advantages in solving complex distribution needs and optimizing resource allocation. The EVRP models of Fan [14], Wang et al. [43], Agrali and Lee [3] all adopt the configuration of multiple depots.

3.4.4. *Open/Close.* In EVRP models, 'open' and 'closed' are commonly used to define whether vehicles must return to their depot upon task completion. In the closed model, vehicles must return to their initial depot after completing assigned tasks, making it the most widely applied approach in EVRP studies. A different configuration, the half-open model, permits vehicles to return to the nearest depot rather than return to their original departure depot [14].

4. Recent solution approaches to EVRP

The solution approaches for the EVRP are generally classified into exact and heuristic algorithms. Exact algorithms rely on mathematical programming and commonly utilize approaches such as Branch-and-Price and Dynamic Programming to achieve optimal solutions. In contrast, heuristic and metaheuristic algorithms employ flexible and efficient search strategies to approximate optimal solutions within a computationally feasible time. The distribution of EVRP solutions in this study is shown in Figure.1. Representative methods include Large Neighborhood Search (LNS), Variable Neighborhood Search (VNS), Branch-and-Price (BP), Ant Colony Optimization (ACO), Simulated Annealing (SA), Genetic Algorithm (GA), and Tabu Search (TS).

This section presents an in-depth discussion of the exact and heuristic algorithms applied in EVRP.

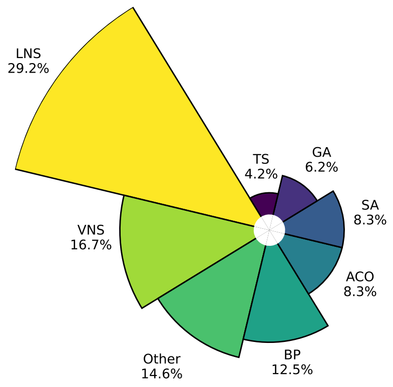


FIGURE 1. Distribution of EVRP solution approaches (% is obtained by reporting the number of uses for each algorithm to the total number of algorithms used in all research methods).

4.1. **Large Neighborhood Search.** LNS, as a practical heuristic approach, has been widely applied to solving the EVRP[44, 31, 35, 52, 42, 29, 13, 5, 3, 41, 47, 28, 49, 55]. This method iteratively removes and reinserts subsets of routes to explore better solutions efficiently. Researchers have improved its computational efficiency and solution optimality for large-scale problems through integration with various optimization techniques. For example, Ren et al. [35] introduced an LNS-QL algorithm based on Q-learning (QL) for joint drone and EV delivery, dynamically selecting destruction and repair operators through reinforcement learning, significantly enhancing solution flexibility and adaptability. In the continued development of LNS, researchers have proposed various improved Adaptive Large Neighborhood Search (ALNS) algorithms to handle the complex constraints and uncertainties of EVRP effectively. For instance, Zhang et al. [52] proposed an extended ALNS incorporating the Variable Neighborhood Descent strategy to achieve the simultaneous optimization of EVs and battery swapping vehicles.

4.2. **Variable Neighborhood Search.** VNS enhances search efficiency by dynamically switching between multiple neighborhood structures, enabling the algorithm to escape local optima. Due to its flexibility and effectiveness in exploring diverse search neighborhoods, VNS and its variants have gained increasing attention in EVRP research [54, 17, 25, 33, 39, 8]. İslim and Çatay [17] introduced a hybrid approach that integrates VNS with a mathematical solver to address battery degradation issues in EVs. This method employs a piecewise linear degradation cost model based on the depth of discharge and state of charge (SoC) to assess the impact of varying charging depths. Liu et al. [25] presented a double adaptive generalized VNS framework, which dynamically adjusts the neighbourhood selection mechanism, substantially improving computational efficiency for unmanned EV routing problems. Moreover, Souza et al. [39] developed an optimization algorithm based on Flexible VNS, incorporating adaptive perturbation and local search strategies.

4.3. **Branch-and-Price.** BP algorithms that combine branch-and-bound and column generation are widely used in EVRP [31, 6, 12, 24, 23]. Ouyang and Wang [31] proposed an improved BP algorithm combined with LNS to overcome formulation challenges faced by conventional methods. Bezzi et al. [6] introduced a path-based BP algorithm incorporating multiple charging technologies and partial charging, using Bi-Directional Dynamic Programming to improve pricing efficiency for large-scale problems. Duman et al. [12] developed a Pulse-enhanced bi-directional BP algorithm with a novel column generation technique that alleviates computational bottlenecks compared to traditional labeling methods. Lera-Romero et al. [24] proposed a BCP algorithm for Time-Dependent EVRP with Time Windows, integrating a customer-based routing heuristic and an efficient labeling algorithm to optimize delivery routes.

4.4. **Ant Colony Optimization.** ACO simulates the pheromone-based foraging behavior of ants and improves path selection through probabilistic decision-making and pheromone updating iterations to efficiently solve EVRP [15, 10, 18]. Fan et al. [15] introduced an improved ACO, which incorporates an adaptive heuristic factor that dynamically adjusts pheromone weights based on the specific characteristics of the problem, achieving a balance between global exploration and local exploitation. Comert and Yazgan [10] investigated three distinct types of multi-objective EVRP and proposed a hierarchical hybrid heuristic approach. The first stage employs a hybrid ACO algorithm, integrating local search operations and the SA criterion to expedite the convergence process of the initial solution. In the second stage, the artificial bee colony algorithm is utilized to refine the solution further, ensuring high-quality results.

4.5. **Simulated Annealing.** SA has been extensively applied to the EVRP due to its capability of accepting suboptimal solutions during the optimization process, thereby facilitating escape from local optima [44, 10, 3, 30, 36]. By effectively balancing exploration and utilization, SA demonstrates strong problem-solving capabilities when combined with other heuristics. SA is frequently combined with LNS. Woo et al. [44] proposed an optimization framework that integrates Adaptive Large Neighborhood Search (ALNS) with SA to provide an effective solution for intelligent fleet management. Agrali and Lee [3] proposed the SA-LNS algorithm, which leverages a greedy heuristic for initial solution generation, SA to escape local optima via the Metropolis criterion, and LNS for iterative refinement through destruction and repair, enhancing routing and charging station optimization. Rodríguez-Esparza et al. [36] proposes a hyper-heuristic algorithm to optimize the paths using adaptive SA and reinforcement learning to minimize the total distance traveled and verifies its superiority on a dataset for large-scale problems.

4.6. **Genetic Algorithm.** GA utilizes its selection, crossover, and mutation mechanisms to navigate the solution space under complex constraints efficiently, providing a practical approach for solving EVRP [27, 43, 32]. In this context, Longhitano et al. [27] proposed a GA-based EVRP approach, which comprehensively considers key state parameters of EVs, including the SoC and the state of health. Furthermore, Wang et al. [43] proposed a bi-objective nonlinear model, utilizing Gaussian Mixture Clustering to classify customers and reduce computational complexity. They further introduced an improved multi-objective GA with TS to balance local and global search, enhancing solution quality.

4.7. **Tabu Search.** TS is a local search-based heuristic that uses a tabu list to avoid revisiting recent solutions, helping to escape local optima. Sadati et al. [37] proposed a hybrid heuristic combining VNS and granular TS. The approach starts with a greedy insertion heuristic for initial solution construction, followed by perturbation techniques such as position exchange and route consolidation. It concludes with a local search to optimize customer sequencing and charging decisions. Wang et al. [42] tackled perishable goods distribution by designing multi-compartment vehicles to meet diverse storage needs. They developed a hybrid ALNS-TS algorithm, where ALNS applies various removal and insertion strategies to optimize routes, and adaptive heuristics adjust temperature and humidity in real-time.

4.8. **Other Methods.** Beyond commonly used optimization algorithms, alternative approaches have been explored for EVRP. For instance, the Double Assistant Evolutionary Multitasking Algorithm [7], Iterated Local Search [21], and the Whale Optimization Algorithm [53]. Moreover, the Memetic Algorithm (MA) has also been utilized [46, 11], among which Dong et al. [11] introduced an Improved MA combining global and local search, reducing operational costs by 10–25% in Dynamic EVRP.

5. Discussion

This section first discusses and compares the strengths and weaknesses of different algorithms used in the last three years of EVRP research. Then, future research directions are identified based on the current advancements in EVRP research.

5.1. **Comparative analysis of recent algorithms for EVRP.** The combination of the BP algorithm with the column generation method provides a guaranteed lower bound, thereby improving solution efficiency. However, since column generation relies on the efficient solution of the shortest path problem, computational complexity grows rapidly with the increase in problem size. In practical applications, BP needs to be combined with heuristic acceleration strategies to balance efficiency and accuracy [12, 31].

Although GA possesses excellent global search capabilities, it typically requires more iterations to converge to an acceptable solution compared to heuristic methods, leading to higher computational costs. In particular, in Longhitano et al. [27], the integration of vehicle dynamics and SoC modeling significantly increases the computational burden of the optimization process.

ACO can explore multiple solutions simultaneously, making it suitable for global optimization. However, in large-scale EVRP problems, the need to simulate numerous ants leads to increased computation time. Thus current research often employs a two-level or two-stage optimization approach, where the first stage decomposes the problem to reduce the number of variables handled per iteration, and the second stage refines routes and optimizes charging strategies to improve solution quality.

The flexibility and global search capability of VNS make it suitable for various complex constraints in EVRP, such as time windows [54], battery swapping [33], and flexible deliveries [37]. Improved VNS methods, such as Flexi-VNS, dynamically adjust charging strategies to enhance solution adaptability. Additionally, VNS, combined with the alternating direction multiplier method, effectively handles energy constraints, achieving better performance in large-scale instances.

ALNS and its variants dominate EVRP solutions. ALNS is more efficient for large-scale problems and is easily integrated with other algorithms. For instance, ALNS combined with SA and QL can further enhance global search capabilities. Specifically, the combination of QL and LNS proves effective in dynamic EVRP, where QL learns operational strategies and improves the search process based on historical experience.

5.2. **Open issues.** Although significant progress has been made in addressing the EVRP, there are still challenges that require further research. Firstly, EVRP involves multiple optimization objectives, such as minimizing operational costs, carbon emissions, and customer service levels. However, existing studies often lack systematic research on multi-objective trade-offs. Developing more efficient multi-objective algorithms to balance conflicting objectives remains a valuable research direction. Secondly, a single algorithm is often insufficient to handle complex EVRP problems. Future research can explore the combination of multiple algorithms, such as integrating heuristic algorithms with reinforcement learning. Reinforcement learning is effective in handling dynamic environments and learning complex decision-making strategies. Lastly, future studies should also incorporate machine learning models to predict factors such as EV energy consumption, charging demands, and traffic flow. These predictions can be integrated into the routing process to achieve more accurate scheduling.

6. Conclusions and future work

This study presents a comprehensive review of recent advancements in EVRP research over the past three years, analyzing 42 papers from various aspects. It presents various classifications of EVRP and examines commonly used algorithms. In terms of objective functions, recent studies mainly focus on single or limited objectives, lacking systematic research on multi-objectives. Regarding algorithms, LNS is widely adopted as one of the most commonly used optimization methods and is often combined with SA, BP, and QL to improve the depth of exploration of the solution and the ability of local optimization. In the future, enhancing these algorithms or developing novel hybrid optimization approaches will continue to be a promising avenue for research. Moreover, integrating machine learning into demand or traffic predictions can further improve EVRP solutions' adaptability.

References

[1] Co2 emissions from cars: facts and figures (infographics), 2019. URL `https://www.europarl.europa.eu/topics/en/article/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics`.

[2] Fit for 55: zero co$_2$ emissions for new cars and vans in 2035, 2023. URL `https://www.europarl.europa.eu/news/en/press-room/20230210IPR74715/fit-for-55-zero-co2-emissions-for-new-cars-and-vans-in-2035`.

[3] Cansu Agrali and Seokcheon Lee. The multi-depot pickup and delivery problem with capacitated electric vehicles, transfers, and time windows. *Computers & Industrial Engineering*, 179:109207, May 2023. ISSN 03608352. doi: 10.1016/j.cie.2023.109207.

[4] Vahid Akbari, Bülent Çatay, and İhsan Sadati. Route optimization of battery electric vehicles using dynamic charging on electrified roads. *Sustainable Cities and Society*, 109:105532, August 2024. ISSN 22106707. doi: 10.1016/j.scs.2024.105532.

[5] Afsane Amiri, Hossein Zolfagharinia, and Saman Hassanzadeh Amin. A robust multi-objective routing problem for heavy-duty electric trucks with uncertain energy consumption. *Computers & Industrial Engineering*, 178:109108, April 2023. ISSN 03608352. doi: 10.1016/j.cie.2023.109108.

[6] Dario Bezzi, Alberto Ceselli, and Giovanni Righini. A route-based algorithm for the electric vehicle routing problem with multiple technologies. *Transportation Research Part C: Emerging Technologies*, 157:104374, December 2023. ISSN 0968090X. doi: 10.1016/j.trc.2023.104374.

[7] Yanguang Cai, Yanlin Wu, and Chuncheng Fang. Double-assistant evolutionary multitasking algorithm for enhanced electric vehicle routing with backup batteries and battery swapping stations. *Expert Systems with Applications*, 237:121600, March 2024. ISSN 09574174. doi: 10.1016/j.eswa.2023.121600.

[8] Bülent Çatay and İhsan Sadati. An improved matheuristic for solving the electric vehicle routing problem with time windows and synchronized mobile charging/battery swapping. *Computers & Operations Research*, 159:106310, November 2023. ISSN 03050548. doi: 10.1016/j.cor.2023.106310.

[9] Tao Chen, Bowen Zhang, Hajir Pourbabak, Abdollah Kavousi-Fard, and Wencong Su. Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems. *IEEE Transactions on Smart Grid*, 9(4): 3563–3572, 2016.

[10] Serap Ercan Comert and Harun Resit Yazgan. A new approach based on hybrid ant colony optimization-artificial bee colony algorithm for multi-objective electric vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 123: 106375, August 2023. ISSN 09521976. doi: 10.1016/j.engappai.2023.106375.

[11] Jinting Dong, Hongfeng Wang, and Shuzhu Zhang. Dynamic electric vehicle routing problem considering mid-route recharging and new demand arrival using an improved memetic algorithm. *Sustainable Energy Technologies and Assessments*, 58:103366, August 2023. ISSN 22131388. doi: 10.1016/j.seta.2023.103366.

[12] Ece Naz Duman, Duygu Taş, and Bülent Çatay. A bidirectional branch-and-price algorithm with pulse procedure for the electric vehicle routing problem with flexible deliveries. *Transportation Research Part C: Emerging Technologies*, 165:104699, August 2024. ISSN 0968090X. doi: 10.1016/j.trc.2024.104699.

[13] Mehmet Erdem, Çağrı Koç, and Eda Yücel. The electric home health care routing and scheduling problem with time windows and fast chargers. *Computers & Industrial Engineering*, 172:108580, October 2022. ISSN 03608352. doi: 10.1016/j.cie.2022.108580.

[14] Lijun Fan. A two-stage hybrid ant colony algorithm for multi-depot half-open time-dependent electric vehicle routing problem. *Complex & Intelligent Systems*, 10(2):2107–2128, April 2024. ISSN 2198-6053. doi: 10.1007/s40747-023-01259-1.

[15] Lijun Fan, Changshi Liu, Bo Dai, Junyu Li, Zhang Wu, and Yuting Guo. Electric vehicle routing problem considering energy differences of charging stations. *Journal of Cleaner Production*, 418:138184, September 2023. ISSN 09596526. doi: 10.1016/j.jclepro.2023.138184.

[16] Dominik Goeke and Michael Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, 2015.

[17] Raci Berk İslim and Bülent Çatay. An effective matheuristic approach for solving the electric traveling salesperson problem with time windows and battery degradation. *Engineering Applications of Artificial Intelligence*, 132:107943, June 2024. ISSN 09521976. doi: 10.1016/j.engappai.2024.107943.

[18] Ya-Hui Jia, Yi Mei, and Mengjie Zhang. Confidence-based ant colony optimization for capacitated electric vehicle routing problem with comparison of different encoding schemes. *IEEE Transactions on Evolutionary Computation*, 26(6): 1394–1408, December 2022. ISSN 1941-0026. doi: 10.1109/TEVC.2022.3144142.

[19] Can Berk Kalaycı and Yusuf Yılmaz. A review on the electric vehicle routing problems. *Pamukkale University Journal of Engineering Sciences-Pamukkale Universitesi Muhendislik Bilimleri Dergisi*, 2023.

[20] Gitae Kim, Yew-Soon Ong, Chen Kim Heng, Puay Siew Tan, and Nengsheng Allan Zhang. City vehicle routing problem (city vrp): A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1654–1666, 2015. doi: 10.1109/TITS.2015.2395536.

[21] Yong Jun Kim and Byung Do Chung. Energy consumption optimization for the electric vehicle routing problem with state-of-charge-dependent discharging rates. *Journal of Cleaner Production*, 385:135703, January 2023. ISSN 09596526. doi: 10.1016/j.jclepro.2022.135703.

[22] Ilker Kucukoglu, Reginald Dewil, and Dirk Cattrysse. The electric vehicle routing problem and its variations: A literature review. *Computers & Industrial Engineering*, 161:107650, 2021.

[23] Edward Lam, Guy Desaulniers, and Peter J. Stuckey. Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. *Computers & Operations Research*, 145:105870, September 2022. ISSN 03050548. doi: 10.1016/j.cor.2022.

105870.

[24] Gonzalo Lera-Romero, Juan José Miranda Bront, and Francisco J. Soulignac. A branch-cut-and-price algorithm for the time-dependent electric vehicle routing problem with time windows. *European Journal of Operational Research*, 312(3): 978–995, February 2024. ISSN 03772217. doi: 10.1016/j.ejor.2023.06.037.

[25] Wenheng Liu, Mahjoub Dridi, Jintong Ren, Amir Hajjam El Hassani, and Shuying Li. A double-adaptive general variable neighborhood search for an unmanned electric vehicle routing and scheduling problem in green manufacturing systems. *Engineering Applications of Artificial Intelligence*, 126:107113, November 2023. ISSN 09521976. doi: 10.1016/j.engappai.2023.107113.

[26] Xiaochang Liu, Dujuan Wang, Yunqiang Yin, and T.C.E. Cheng. Robust optimization for the electric vehicle pickup and delivery problem with time windows and uncertain demands. *Computers & Operations Research*, 151:106119, March 2023. ISSN 03050548. doi: 10.1016/j.cor.2022.106119.

[27] Pedro Dias Longhitano, Christophe Bérenguer, and Benjamin Echard. Joint electric vehicle routing and battery health management integrating an explicit state of charge model. *Computers & Industrial Engineering*, 188:109892, February 2024. ISSN 03608352. doi: 10.1016/j.cie.2024.109892.

[28] Bingshan Ma, Dawei Hu, Yin Wang, Qian Sun, Linwei He, and Xiqiong Chen. Time-dependent vehicle routing problem with departure time and speed optimization for shared autonomous electric vehicle service. *Applied Mathematical Modelling*, 113:333–357, January 2023. ISSN 0307904X. doi: 10.1016/j.apm.2022.09.020.

[29] Hongguang Ma, Rongchao Yang, and Xiang Li. Delivery routing for a mixed fleet of conventional and electric vehicles with road restrictions. *International Journal of Production Research*, pages 1–24, 2024.

[30] Nima Moradi and Niloufar Mirzavand Boroujeni. Prize-collecting electric vehicle routing model for parcel delivery problem. *Expert Systems with Applications*, 259:125183, January 2025. ISSN 09574174. doi: 10.1016/j.eswa.2024.125183.

[31] Kechen Ouyang and David Z.W. Wang. Optimal operation strategies for freight transport with electric vehicles considering wireless charging lanes. *Transportation Research Part E: Logistics and Transportation Review*, 193:103852, January 2025. ISSN 13665545. doi: 10.1016/j.tre.2024.103852.

[32] David Peña, Bernabé Dorronsoro, and Patricia Ruiz. Sustainable waste collection optimization using electric vehicles. *Sustainable Cities and Society*, 105:105343, June 2024. ISSN 22106707. doi: 10.1016/j.scs.2024.105343.

[33] Bin Qian, Fei-Long Feng, Nai-Kang Yu, Rong Hu, and Yu-Wang Chen. An alternating direction multiplier method with variable neighborhood search for electric vehicle routing problem with time windows and battery swapping stations. *Applied Soft Computing*, 166:112141, November 2024. ISSN 15684946. doi: 10.1016/j.asoc.2024.112141.

[34] Hu Qin, Xinxin Su, Teng Ren, and Zhixing Luo. A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management*, 8:370–389, 2021.

[35] Xiao-Xue Ren, Hou-Ming Fan, Ming-Xin Bao, and Hao Fan. The time-dependent electric vehicle routing problem with drone and synchronized mobile battery swapping. *Advanced Engineering Informatics*, 57:102071, August 2023. ISSN 14740346. doi: 10.1016/j.aei.2023.102071.

[36] Erick Rodríguez-Esparza, Antonio D. Masegosa, Diego Oliva, and Enrique Onieva. A new hyper-heuristic based on adaptive simulated annealing and reinforcement learning for the capacitated electric vehicle routing problem. *Expert Systems with Applications*, 252:124197, October 2024. ISSN 09574174. doi: 10.1016/j.eswa.2024.124197.

[37] Mir Ehsan Hesam Sadati, Vahid Akbari, and Bülent Çatay. Electric vehicle routing problem with flexible deliveries. *International Journal of Production Research*, 60(13):4268–4294, July 2022. ISSN 0020-7543. doi: 10.1080/00207543.2022.2032451.

[38] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, 48(4):500–520, 2014.

[39] André L.S. Souza, Marcella Papini, Puca H.V. Penna, and Marcone J.F. Souza. A flexible variable neighbourhood search algorithm for different variants of the electric vehicle routing problem. *Computers & Operations Research*, 168:106713, August 2024. ISSN 03050548. doi: 10.1016/j.cor.2024.106713.

[40] Marios Thymianis, Alexandros Tzanetos, Eneko Osaba, Georgios Dounias, and Javier Del Ser. Electric vehicle routing problem: Literature review, instances and results with a novel ant colony optimization method. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022.

[41] Weiquan Wang and Jingyi Zhao. Partial linear recharging strategy for the electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 308(2):929–948, July 2023. ISSN 03772217. doi: 10.1016/j.ejor.2022.12.011.

[42] Xin Wang, Yijing Liang, Xiangbo Tang, and Xiyan Jiang. A multi-compartment electric vehicle routing problem with time windows and temperature and humidity settings for perishable product delivery. *Expert Systems with Applications*, 233:120974, December 2023. ISSN 09574174. doi: 10.1016/j.eswa.2023.120974.

[43] Yong Wang, Jingxin Zhou, Yaoyao Sun, Jianxin Fan, Zheng Wang, and Haizhong Wang. Collaborative multidepot electric vehicle routing problem with time windows and shared charging stations. *Expert Systems with Applications*, 219:119654, June 2023. ISSN 09574174. doi: 10.1016/j.eswa.2023.119654.

[44] Soomin Woo, Eric Yongkeun Choi, Scott J. Moura, and Francesco Borrelli. Saving energy with eco-friendly routing of an electric vehicle fleet. *Transportation Research Part E: Logistics and Transportation Review*, 189:103644, September 2024. ISSN 13665545. doi: 10.1016/j.tre.2024.103644.

[45] Xiaoyun Xia, Helin Zhuang, Zijia Wang, and Zefeng Chen. Two-stage heuristic algorithm with pseudo node-based model for electric vehicle routing problem. *Applied Soft Computing*, 165:112102, November 2024. ISSN 15684946. doi: 10.1016/j.asoc.2024.112102.

[46] Jianhua Xiao, Jingguo Du, Zhiguang Cao, Xingyi Zhang, and Yunyun Niu. A diversity-enhanced memetic algorithm for solving electric vehicle routing problems with time windows and mixed backhauls. *Applied Soft Computing*, 134: 110025, February 2023. ISSN 15684946. doi: 10.1016/j.asoc.2023.110025.

[47] Jianhua Xiao, Xiaoyang Liu, Tao Liu, Na Li, and Antonio Martinez-Sykora. The electric vehicle routing problem with synchronized mobile partial recharging and non-strict waiting strategy. *Annals of Operations Research*, June 2024. ISSN 1572-9338. doi: 10.1007/s10479-024-06069-3.

[48] Yiyong Xiao, Yue Zhang, Ikou Kaku, Rui Kang, and Xing Pan. Electric vehicle routing problem: A systematic review and a new comprehensive model with nonlinear energy recharging and consumption. *Renewable and Sustainable Energy Reviews*, 151:111567, 2021.

[49] Hao Xiong, Yumiao Xu, Huili Yan, Haoying Guo, and Chen Zhang. Optimizing electric vehicle routing under traffic congestion: A comprehensive energy consumption model considering drivetrain losses. *Computers & Operations Research*, 168:106710, August 2024. ISSN 03050548. doi: 10.1016/j.cor.2024.106710.

[50] Canqi Yao, Shibo Chen, Mauro Salazar, and Zaiyue Yang. Joint routing and charging problem of electric vehicles with incentive-aware customers considering spatio-temporal charging prices. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):12215–12226, November 2023. ISSN 1558-0016. doi: 10.1109/TITS.2023.3286952.

[51] Chong Ye, Wenjie He, and Hanqi Chen. Electric vehicle routing models and solution algorithms in logistics distribution: A systematic review. *Environmental Science and Pollution Research*, 29(38):57067–57090, 2022.

[52] Shuai Zhang, Tong Zhou, Cheng Fang, and Sihan Yang. A novel collaborative electric vehicle routing problem with multiple prioritized time windows and time-dependent hybrid recharging. *Expert Systems with Applications*, 244:122990, June 2024. ISSN 09574174. doi: 10.1016/j.eswa.2023.122990.

[53] Binghai Zhou and Zhe Zhao. Multi-objective optimization of electric vehicle routing problem with battery swap and mixed time windows. *Neural Computing and Applications*, 34(10):7325–7348, May 2022. ISSN 1433-3058. doi: 10.1007/s00521-022-06967-2.

[54] Saiqi Zhou, Dezhi Zhang, Bin Ji, Shaoyu Zhou, Shuangyan Li, and Likun Zhou. A milp model and heuristic method for the time-dependent electric vehicle routing and scheduling problem with time windows. *Journal of Cleaner Production*, 434: 140188, January 2024. ISSN 09596526. doi: 10.1016/j.jclepro.2023.140188.

[55] Saiqi Zhou, Dezhi Zhang, Wen Yuan, Zhenjie Wang, Likun Zhou, and Michael G.H. Bell. Pickup and delivery problem with electric vehicles and time windows considering queues. *Transportation Research Part C: Emerging Technologies*, 167:104829, October 2024. ISSN 0968090X. doi: 10.1016/j.trc.2024.104829.

Department of Computer Science, Babes-Bolyai University, 1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

Email address: `yingkai.x@ubbcluj.ro`