11-2023

On the sustainability of deep learning projects: Maintainers' perspective

Junxiao HAN

Jiakun LIU

David LO
Singapore Management University, davidlo@smu.edu.sg

Chen ZHI

Yishan CHEN

See next page for additional authors

Citation

HAN, Junxiao; LIU, Jiakun; LO, David; ZHI, Chen; CHEN, Yishan; and DENG, Shuiguang. On the sustainability of deep learning projects: Maintainers' perspective. (2023). *Journal of Software: Evolution and Process*. 1-20.
Available at: https://ink.library.smu.edu.sg/sis_research/8481

On the Sustainability of Deep Learning Projects: Maintainers' Perspective

Junxiao Han[a,*], Jiakun Liu[b,*], David Lo[b], Chen Zhi[c], Shuiguang Deng[d], Minghui Wu[a]

[a]*School of Computer "& Computing Science Hangzhou City University Hangzhou China*
[b]*School of Information Systems Singapore Management University Singapore*
[c]*School of Software Technology Zhejiang University Ningbo China*
[d]*College of Computer Science and Technology Zhejiang University Hangzhou China*

Abstract

Deep learning (DL) techniques have grown in leaps and bounds in both academia and industry over the past few years. Despite the growth of DL projects, there has been little study on how DL projects evolve, whether maintainers in this domain encounter a dramatic increase in workload, and whether or not existing maintainers can guarantee the sustained development of projects. To address this gap, we perform an empirical study to investigate the sustainability of DL projects, understand maintainers' workloads and workloads evolution in DL projects, and compare them with traditional OSS projects. In this regard, we first investigate how DL projects evolve, then, understand maintainers' workload in DL projects, and explore the workload evolution of maintainers as DL projects evolve. After that, we mine the relationships between maintainers' activities and the sustainability of DL projects. Eventually, we compare it with traditional OSS projects. Our study unveils that although DL projects show increasing trends in most activities, maintainers' workloads present a decreasing trend. Meanwhile, the proportion of workload maintainers afford in DL projects is significantly lower than in traditional OSS projects. Moreover, there are positive and moderate correlations between the sustainability of DL projects and the number of maintainers' releases, pushes, and merged pull requests. Our findings shed lights that help understand maintainers' workload and growth trends in DL and traditional OSS projects, and also highlight actionable directions for organizations, maintainers, and researchers.

1. Introduction

Deep learning (DL) has grown in leaps and bounds in both academia and industry over the past few years, producing a wide variety of deep learning algorithms for various applications such as image processing Ciregan et al. (2012); He et al. (2016); Acuna et al. (2019), speech recognition Hinton et al. (2012); Nassif et al. (2019), autonomous driving Chen et al. (2015); Huval et al. (2015); Tian et al. (2018), disease diagnosis and drug discovery Chen et al. (2018); Oktay et al. (2018); Petersen et al. (2019), and financial fraud detection Roy et al. (2018). However, compared with open-source software (OSS) projects, DL projects have an additional set of DL-specific issues Wan et al. (2019). For example, DL projects encode the network structure of a satisfying DL model and use a large amount of data to train the task-solving model, while traditional OSS projects directly encode the model to solve a target problem Zhang et al. (2018). To facilitate the development of DL projects, software engineering researchers have devoted significant effort Sonnenburg et al. (2007); Guo et al. (2019); Han et al. (2020a); Zhang et al. (2020); Cambronero et al. (2019); Mathew and Stolee (2021); Wei et al. (2019); Pradel and Sen (2018).

The popularity of DL projects has given rise to a widespread trend: while the development and deployment of DL systems are relatively fast and cost-effective, their sustainability proves to be challenging and costly Wan et al. (2019); Penzenstadler et al. (2012). The concept of sustainability is certainly multifaceted and no general definition exists. In this study, we adopted the terminology of sustainability and its indicator introduced in Valiev et al. Valiev et al. (2018) and Zhang et al.'s Zhang et al. (2022) studies, where they consider a project is dormant when it did not receive any commit for more than six months after its last commit. To better analyze the relationships between maintainers' activities and the sustainability of DL projects

*Corresponding authors
Email addresses: junxiaohan@zju.edu.cn (Junxiao Han), jkliu@smu.edu.sg (Jiakun Liu), davidlo@smu.edu.sg (David Lo), zjuzhichen@zju.edu.cn (Chen Zhi), dengsg@zju.edu.cn (Shuiguang Deng), mhwu@zucc.edu.cn (Minghui Wu)

over time, we changed this term from a static state to a dynamic state (see Section 2.2), and used the number of commits to characterize the sustainability of projects.

Intuitively, the maintenance of a project is related to its sustainability. According to the definition of ISO/IEC 25051 (ISO/IEC, 2014), maintainers modify a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment to maintain the project. However, maintainers may have a heavy workload so that they cannot deal with tasks in time. As illustrated in Figure 1, a contributor from a DL project complained that the maintainers were so busy that the pull request he raised had not been noticed and replied to for a long time.[1] As DL projects continue to expand, the projects draw increasing numbers of participants and accumulate a growing body of pull requests. We would like to understand: is the workload of maintainers steadily increasing? Can maintainers in DL projects effectively manage and sustain the rapid development of DL systems?

Despite numerous studies characterizing the work of maintainers in DL projects Zhou et al. (2017); Tan et al. (2020a), their focus has been limited to technical aspects, such as the number of files maintained, commits committed, and authors managed. However, a comprehensive understanding of maintainers' work is crucial for ensuring the long-term sustainability of DL projects and may also offer valuable guidance to other OSS projects.

To fill the gap, in this paper, we characterize maintainers' activity in DL projects. To compare the differences between DL and traditional OSS projects in terms of maintainers' activity, we collect 19 popular DL projects and 19 popular traditional OSS projects. Then, we holistically understand how to characterize the workload of maintainers (i.e., including not only technical works but also non-technical works) in DL projects, along with understanding the evolution of projects and maintainers' workload. Our preliminary study shows that besides technical tasks (e.g., commits committed), maintainers also write comments under issues and write WiKis to sustain the coordination and collaboration of participants. To understand how the DL and traditional OSS projects evolve and whether the evolution is sustainable, we assemble all activities (i.e., events identified via GitHub API) of these projects to answer the following research questions:

RQ1. How to characterize the workload of maintainers holistically in DL projects?

[1] https://github.com/fastai/fastai/issues/575

Our findings show that we can employ the number of issues, closed issues (i.e., issues that are solved by maintainers), issue comments, commits, commit comments, pushes, pull requests, merged pull requests (i.e., pull requests that are merged by maintainers), pull request reviews, pull request review comments, and releases to characterize the workload of maintainers holistically.

RQ2. How do DL projects evolve and how do the overall workloads of maintainers evolve in this process?

We observe that there is a significant difference in the evolution of most activities between DL and traditional OSS projects. Meanwhile, although the overall workload of most activities is increasing in DL projects, the maintainers' workload of most activities is decreasing. In contrast, in traditional OSS projects, the overall workload and the maintainers' workload are both stable. Statistical test results indicate that the proportion of workload that maintainers afford on many activities in DL projects is significantly lower than those in traditional OSS projects.

RQ3. How does the average workload of maintainers evolve as DL projects grow?

On average, maintainers in both DL and traditional OSS projects are experiencing an increasing workload in some activities, such as Pull Requests, merged Pull Requests, Pull Request Review Comments, and Releases. However, the Commit Comment activity per maintainer performed is the only activity that shows a significant difference in growth rates between DL and traditional OSS projects. Additionally, when it comes to each project, most DL projects demonstrate significant differences in growth rates across various activities of maintainers, while traditional OSS projects with a larger size tend to show no significant differences in growth rates across various activities for maintainers.

RQ4. What are the relationships between maintainers' activities and the sustainability of DL projects?

We observe that the sustainability of DL projects is positively and moderately correlated with the number of releases, pushes, and merged pull requests of maintainers. However, the number of releases of maintainers shows no significant correlation with the sustainability of traditional OSS projects.

Based on our findings, we suggest that GitHub or project owners propose new gamification systems to encourage maintainers to maintain DL projects. For example, maintainers could be rewarded with a DL-specific badge or a certain number of points for each release, push, and merged pull request. Moreover, we suggest that organizations, developers, maintainers, and researchers should be aware of the significant differences

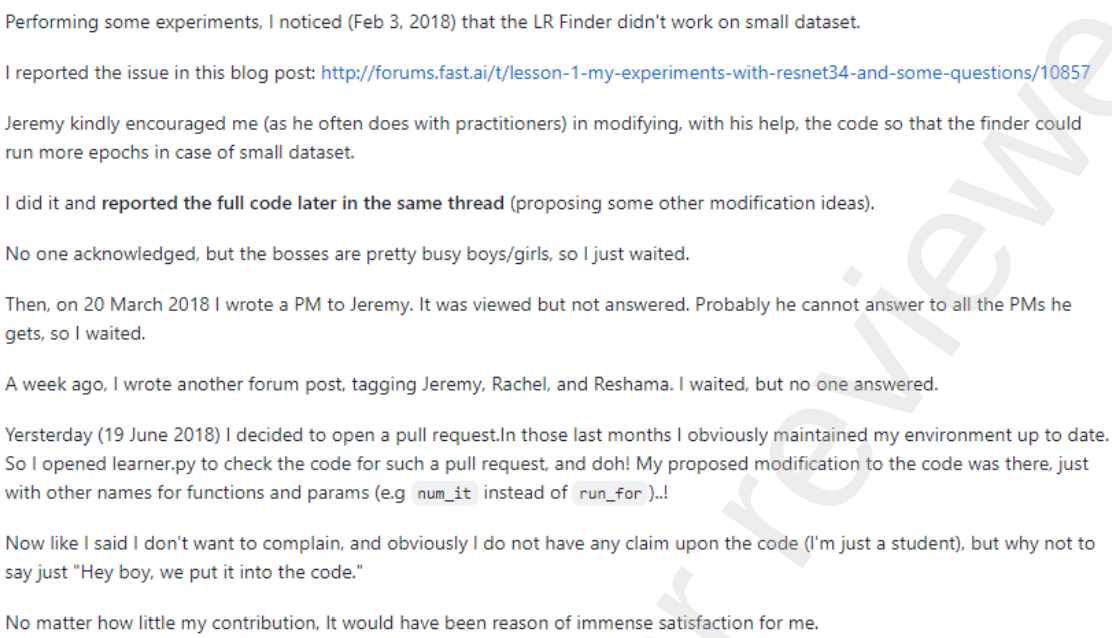


Figure 1: An issue in fastai project. This example shows that the DL project maintainers did not respond to a pull request for a long time.

in the evolution of activities between DL and traditional OSS projects, as well as the significant differences in workload ratios that maintainers afford in DL and traditional OSS projects. This awareness will help them differentiate projects from different domains in their work processes.

In summary, we make the following contributions to this paper:

- To the best of our knowledge, we are the first to explore the sustainability of DL projects from the maintainers' perspective. By comparing with traditional OSS projects, we understand the specificity of the evolution of DL projects, the specificity of DL maintainers' workload and its evolution. Moreover, we also expound on the specific relationships between maintainers' activities and the sustainability of DL projects.
- We highlight some practical implications for organizations, maintainers, and researchers in both the DL and other OSS domains. We help organizations, maintainers, and researchers recognize the significant differences in the evolution of DL and traditional OSS projects. We also help them develop a more comprehensive understanding of maintainers' workloads in DL and traditional OSS projects. Additionally, by pointing out the relationships between maintainers' activities and the sustainability of DL projects, we provide valuable suggestions to maintainers for ensuring the sustained development of DL projects and shed light on researchers for further research.

The rest of this paper is organized as follows. Section 2 describes the research methodology. Section 3 presents the findings and insights of our research questions. Section 4 introduces the related works and discusses the difference between our work and other papers that are close to ours. Section 5 discusses the threats to the validity of our findings. Finally, Section 6 concludes this paper, gives directions to future work, and provides replication packages.

2. Methodology

In this section, we introduce how we construct our dataset, including how we determine targeted projects, the process to obtain activity data, the approach to identify maintainers, and the approach to identify maintainers' workload.

2.1. Collecting Data

To characterize the differences between DL and traditional OSS projects regarding maintainers' activities, we need to collect both DL and traditional OSS projects. To safeguard the quality of the dataset, we use the following rules to select targeted projects: 1) projects must be large enough and have maintained relatively long

traceable records on GitHub (at least 500 commits, 10 contributors, and survived for 2 years), 2) DL projects that are tutorials, examples, courses, handbooks, or learning notes were excluded (by manually observing), 3) the selected OSS projects are with a wide range of tech stacks and application domains, such as a web development framework (react), a team collaboration tool (zulip), and a database (tidb). As a result, 19 DL projects and 19 traditional OSS are collected.

Table 1 and 2 list the project names, and detailed descriptions are publicly available at `https://github.com/HJXPaperData/SustainabilityofDL`. The average duration of the collected DL projects is 6 years, accompanied by an average of 28,905 stars, 163,624KB of sizes, 13,325 commits, and 576 contributors. In comparison, the collected traditional OSS projects have been operational for 8 years on average, boasting 28,588 stars, 186,448KB of sizes, 11,889 commits, and 536 contributors. These statistics suggest a relative similarity between the two constructed datasets.

After collecting the targeted projects, we employed Google BigQuery to extract all the histories of activities for each project. GH Archive provides most of the activity events such as issues, issue comments, pull requests, and pull request reviews for each project Arc (2021), and is available as a public dataset on Google BigQuery Big (2021). We ran SQL-like queries over the entire dataset to collect activity events that are performed in each targeted project from January 2015 to April 2022 Arc (2021); Wang et al. (2020). Consequently, we obtained 15 activity types with about 12GB for the 19 DL projects and about 16GB for the 19 traditional OSS projects. The 15 activity types are Issues, Issue Comments, Commit Comments, Pull Requests, Pull Request Review, Pull Request Review Comments, Push, Release, Gollum (editing wiki), Watch, Member, Public, Fork, Create, and Delete.[2].

2.2. Characterizing the Project Sustainability

The definition of sustainability is multi-faceted Coelho and Valente (2017); Valiev et al. (2018); Mendez et al. (2018); Qiu et al. (2019); Manotas et al. (2016); Noman et al. (2022); Trinkenreich et al. (2021). For instance, the sustainability of OSS projects can be regarded as the success of OSS projects Crowston et al. (2006); Coelho and Valente (2017); Valiev et al. (2018) from the technical perspective and the cooperation perspectiveNoman et al. (2022); Trinkenreich et al. (2021). More specifically, Valiev et al. Valiev et al. (2018) and Zhang et al. Zhang et al. (2022) considered a project is dormant (i.e., unsustained) if a project did not receive a new commit for more than six months after its last commit. However, the sustainability in prior studies Valiev et al. (2018); Zhang et al. (2022) is static, which cannot be simply applied to our study. Since we tend to analyze the relationships between maintainers' activities and the dynamic sustainability of DL projects over time, we thus changed this term from a static state to a dynamic state, and also used the number of commits to characterize the sustainability of projects. This motivates us to investigate the correlations between maintainers' monthly works and project sustainability (detailed in Section 3.3).

2.3. Identifying Maintainers

Maintainers are different from ordinary developers in OSS projects: they devote a significant proportion of their work to maintaining the OSS projects. Maintainers in smaller projects are usually core developers, while in bigger projects, they may need to excel in many other activities beyond coding Dias et al. (2021), e.g., communications with other developers and users. To obtain the list of maintainers, Zhou et al. Zhou et al. (2017) get the maintainers for the Linux kernel ecosystem from the file named `MAINTAINERS`. `MAINTAINERS` contains information about maintainers, including the names of Linux kernel's subsystems, people who maintain it, and the files associated with different subsystems. However, in our dataset, the organization membership is unavailable from GitHub. Therefore, we cannot get the list of maintainers from GitHub directly. Nevertheless, Zhou et al. observed that maintainers are some of the contributors that can commit to projects directly. This motivates us to identify maintainers from all contributors that can commit to projects directly.

To do so, we collected all the committers of our targeted projects and surveyed them to understand how they define a maintainer. Consequently, we obtained 2,615 committers for DL projects and 2,166 committers for traditional OSS projects. After that, we surveyed them and asked *1) "We defined maintainers as the contributors who have the commit privilege to projects. Do you agree with that?"*, and *2) "In your opinion, how do you define maintainers?"* As a result, we received 30 responses from DL committers and 21 responses from traditional OSS committers. Out of those, 10 out of 30 DL committers and 7 out of 21 traditional OSS committers disagreed with the definition. We then applied open card sorting Tan et al. (2020b); Han et al. (2021)

[2] `https://docs.github.com/en/developers/webhooks-and-events/events/github-event-types`

to analyze the responses of those committers to the second question using the following steps: 1) the first author and a Ph.D. candidate separately read the answers to the second question, 2) they generated the initial coding schema separately, 3) searching if the cards have already existed, 4) classifying the generated cards into potential themes for the theme similarity (as described in LaToza et al.'s study LaToza et al. (2006)), 5) discussing and determining the final themes. We then computed the Cohen's Kappa value Cohen (1960) to examine the agreement between the two labelers. Consequently, the Kappa value is 0.89, which indicates an excellent agreement. Results to the second question reveal that, among DL and traditional OSS committers, about 40% of them believed that maintainers are those contributors who also have the merge privilege.

According to the definition of maintainers in ISO/IEC 25000 (ISO/IEC, 2014) and the definition of maintenance in ISO/IEC 25051:2014 (see Section 1), we thus defined maintainers as contributors possessing both merge and commit privileges. This resulted in 155 maintainers for DL projects and 266 maintainers for traditional OSS projects.

2.4. Identifying of Maintainers' activities

To understand why maintainers need the merge privilege and the commit privilege, we characterize maintainers' activities. In Section 2.1, we have gathered 15 types of activities from both DL and traditional OSS projects from GitHub. However, it is unclear whether maintainers only perform these activities in practice and what are their most common activities.

To understand maintainers' activities, we surveyed the maintainers, listed all types of activities, and asked *"Among these activities, which activities can be considered as maintainers' workload in their maintenance progress?"* and *"Apart from these activities mentioned above, do you afford other workloads in your maintenance process?"*.

As a result, for the first question, more than half of both DL and traditional OSS respondents believe that maintainers' workload includes: Issue Comment, Pull Request Review, Pull Request Review Comment, Release, Commit, Pull Request, Issue, Push, and Commit Comment. Regarding the second question, it was found that among the 30 DL respondents, 10 did not provide any answers, 7 gave ambiguous answers, and 6 indicated that the activities mentioned in the first question had already covered their workload. Among the remaining DL respondents, several mentioned that they "use the forum and slack to collaborate with other maintainers", "put proposals for new work", "promote their projects", "update libraries", "solve issues", and "merge pull requests". Simultaneously, among the 21 traditional OSS respondents, 6 did not provide any answers to this question, and 3 believed that the activities identified in the first question had already covered their workload. Among the remaining traditional OSS respondents, they stated that they "solve issues", "merge pull requests", "plan roadmaps for projects", and "manage forums, documentation, and contributors". This indicates that maintainers need the commit privilege to "solve issues" and need the merge privilege to "merge pull requests" to maintain the projects. The solved issues and the merged pull requests can be used to correct faults and improve performance (corresponding to the definition of maintainers in ISO/IEC 25000 (ISO/IEC, 2014) and the definition of maintenance in ISO/IEC (25051:2014)).

Based on the responses from maintainers, we use the number of issues, closed issues (i.e., issues that are solved by maintainers), issue comments, commits, commit comments, push, pull requests, merged pull requests (i.e., pull requests that are merged by maintainers), pull request reviews, pull request review comments, and releases, to characterize the workload of maintainers holistically. This answers RQ1.

Notably, we do not consider communication on forums and other channels, as well as project promotion and planning in our work. This is because we aim to mine the development data hosted on GitHub, and the information hosted outside the github.com website (e.g., forums and other channels) is hard to monitor.

3. Results

This section presents the results for RQ2-4.

3.1. RQ2: How do DL projects evolve and how do the overall workloads of maintainers evolve in this process?

Motivation: Here, we would like to have a basic understanding of how DL projects evolve (i.e., the sustainability of these projects). Meanwhile, we would like to have a first impression of how much the workload is afforded by maintainers among various activities in DL projects evolve. More specifically, considering there are various participants during the life cycle of a software project, we also compare the growth trends of the number of maintainers, the number of authors the maintainers were obligated to deal with, and the number of newcomers attracted between DL and traditional OSS projects.

Approach: To this end, we explore the growth trends of various activities in DL projects and compare them with those of traditional OSS projects. To do so, we count the number of each type of activity in each month for both DL and traditional OSS projects. Afterward, we performed the Kolmogorov-Smirnov test Massey Jr (1951) to examine whether there exist significant differences in the distribution of various activities between DL and traditional OSS projects. The null hypothesis is that DL and traditional OSS projects have the same distribution in various activities.

Besides, we also explore the growth trends of the workload that is afforded by maintainers in DL projects and compare them with those of traditional OSS projects. To do so, for each type of activity, we count the number of activities that are conducted by maintainers who have participated in the activity each month respectively. Then, for both DL and traditional OSS projects, for each type of activity, we calculate the proportion of activities that are conducted by maintainers among all activities respectively. Meanwhile, we ran the Kolmogorov-Smirnov test to check whether there exist significant differences in the distribution of workload ratios of maintainers between DL and traditional OSS projects, and the null hypothesis is that maintainers in DL and traditional OSS projects have the same distribution of workload ratios.

Furthermore, we also explore the growth trends of maintainers, authors, and newcomers in DL projects, and compare them with those of traditional OSS projects. To do so, we depict the number of maintainers, authors, and newcomers each month for both DL and traditional OSS projects. We also ran the Kolmogorov-Smirnov test to check whether there exist significant differences in the distribution towards the number of maintainers, authors, and newcomers in DL and traditional OSS projects, and the null hypothesis is that the distributions of maintainers, authors, and newcomers are the same in DL and traditional OSS projects.

Result: We observe that (1) there are significant differences between DL and traditional OSS projects in the evolution of 8 out of 11 types of activities, i.e., Push, Commit Comment, Issue Comment, Release, Pull Request, merged Pull Request, Pull Request Review, and Pull Request Review Comment. Besides, (2) there are significant differences between DL and traditional OSS projects in terms of the proportion of workload that is afforded by maintainers for 10 out of 11 types of activities (except for Release). To better understand the differences between DL and traditional OSS projects, we take activities Commit, PullRequest, Issue, IssueComment, and Release as examples. Fig. 2 to Fig. 6 depict the trend lines of sampled activities in DL and traditional OSS projects, along with the proportion of workload handled by maintainers.

For example, Fig. 2 shows the number of commits in DL and traditional OSS projects per month, as well as the proportion of workload that is afforded by maintainers. Our statistical test results highlight a significant difference in the distribution of commit activity between DL and traditional OSS projects. The number of commits in DL projects shows almost linear growth before the beginning of 2019 and becomes relatively stable after early 2019. However, the number of commits in traditional OSS projects appears to have a decreasing trend at the end of 2017 and keeps relatively stable during 2018 and 2022. Meanwhile, we also find that the proportion of commits that maintainers afford all has a decreasing trend over time, while maintainers in traditional OSS projects have a higher ratio of commits than maintainers in DL projects. However, no significant difference is found in workload ratios between maintainers in DL and traditional OSS projects.

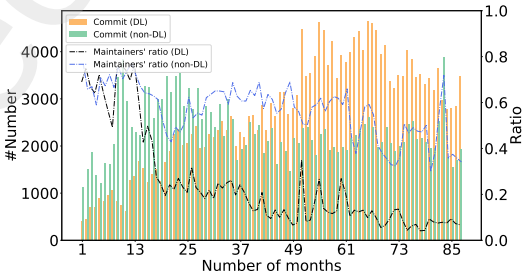


Figure 2: Number of commits in DL and traditional OSS projects, along with the proportion of workload that is afforded by maintainers.

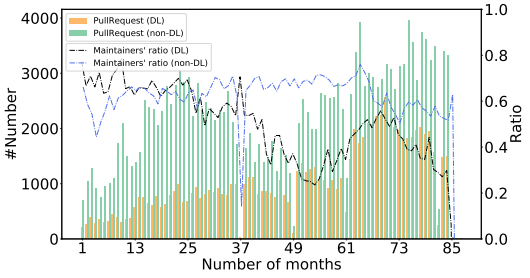


Figure 3: Number of pull requests in DL and traditional OSS projects, along with the proportion of workload that is afforded by maintainers.

Fig. 3 illustrates the number of pull requests in DL and traditional OSS projects per month, as well as the proportion of workload that is afforded by maintainers. Our statistical test results show a significant difference in the distribution of pull request activity between DL

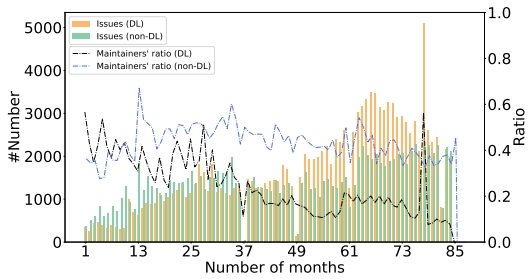


Figure 4: Number of issues in DL and traditional OSS projects, along with the proportion of workload that is afforded by maintainers.

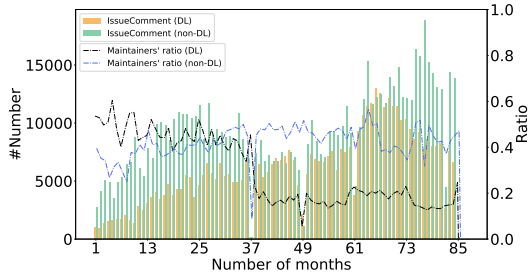


Figure 5: Number of issue comments in DL and traditional OSS projects, along with the proportion of workload that is afforded by maintainers.

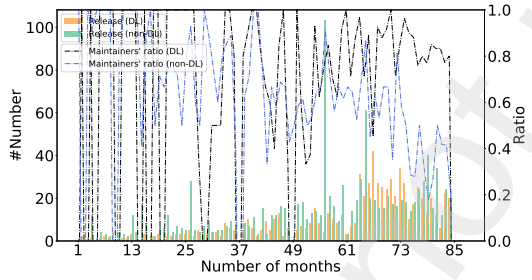


Figure 6: Number of releases in DL and traditional OSS projects, along with the proportion of workload that is afforded by maintainers.

and traditional OSS projects, and there also exists a significant difference in workload ratios of maintainers between DL and traditional OSS projects. As illustrated in Fig. 3, the number of pull requests in DL projects increases almost consistently, while the proportion of pull requests maintainers afford in DL projects decreases in this process. This phenomenon indicates that more and more authors and users are joining DL projects, and have made more and more contributions. Meanwhile, the number of pull requests in traditional OSS projects experiences fluctuations, with a pronounced increase before the end of 2016, followed by a decline until the beginning of 2019, and a subsequent upsurge. During this period, the proportion of pull requests maintainers afford in traditional OSS projects remains relatively stable and high, with an average ratio of approximately 0.6. This phenomenon implies that maintainers in traditional OSS projects are still burdened with a considerable workload in managing the project. Remarkably, maintainers in DL projects exhibit a lower workload ratio concerning pull requests than those in traditional OSS projects.

Fig. 4 depicts the number of issues in DL and traditional OSS projects per month, as well as the proportion of workload that is afforded by maintainers. Our statistical test results indicate that there is no significant difference in the evolution of issue activity between DL and traditional OSS projects. However, there is a significant difference in the workload ratios of issue activity for maintainers in DL and traditional OSS projects. Fig. 4 reveals that both DL and traditional OSS projects show an increasing number of issues, where the growth trend is slower for traditional OSS projects compared to DL projects. Simultaneously, the proportion of issues handled by maintainers is relatively stable in traditional OSS projects, while it exhibits a decreasing trend in DL projects, indicating that more and more issues are being handled by common authors and other users in DL projects. Remarkably, maintainers in DL projects show a lower ratio of issues than maintainers in traditional OSS projects.

As for issue comments, our statistical test results show a significant difference in the distribution of issue comment activity between DL and traditional OSS projects, and there also exists a significant difference in the workload ratios of maintainers in DL and traditional OSS projects. Results in Fig. 5 show that maintainers in DL projects have a lower ratio of issue comments than those in traditional OSS projects. We can also observe that DL projects have almost linear growth before the end of 2020, then a decrease after the beginning of 2021. Conversely, traditional OSS projects show an increasing trend in the number of issue comments. Especially, the proportion of issue comments maintainers afford in traditional OSS projects fluctuates over time but remains relatively stable, while it shows a decreasing trend for maintainers in DL projects.

Fig. 6 displays the number of releases in DL and traditional OSS projects per month, as well as the proportion of workload that is afforded by maintainers. It shows that the number of releases is small for both DL and traditional OSS projects before the end of 2019. However, in the last two years of our studied period, the release frequency increased substantially, with DL projects exhibiting a higher number than traditional

OSS projects. During this time, maintainers in DL projects accounted for a higher proportion of releases than maintainers in traditional OSS projects. Our statistical test results further confirm a significant difference in the distribution of release activity between DL and traditional OSS projects. However, there is no significant difference in the workload ratios of maintainers between DL and traditional OSS projects.

Since we have presented a comprehensive analysis of the evolution of DL projects, and compared them with traditional OSS projects. Specifically, we also depicted the monthly changes in the number of maintainers, authors, and newcomers for both DL and traditional OSS projects, as shown in Fig. 7 to Fig. 9. Our findings in Fig. 7 indicate that traditional OSS projects have a higher number of maintainers than DL projects during the studied period. Moreover, the number of maintainers in traditional OSS projects exhibits a more dynamic pattern than in DL projects. In terms of the number of authors in Fig. 8, we observe a sharp increase in DL projects before the middle of 2020, whereas the number of authors in traditional OSS projects grows sharply only until 2017, after which it remains relatively constant. These findings confirm that DL projects are growing fast and have attracted more and more contributions from various authors Han et al. (2020b).

Regarding the number of newcomers, as illustrated in Fig. 9, we can observe that although the inflow of newcomers fluctuates, it exhibits a long-term increasing trend in DL projects. Notably, the growth trend of authors is steeper than the growth trend of newcomers in DL projects, suggesting that an increasing number of newcomers are joining DL projects and staying to make consistent contributions over time. In contrast, the inflow of newcomers in traditional OSS projects also fluctuates but exhibits a relatively stable trend in the long run. Based on the results of our statistical tests, we found a significant difference in the number of maintainers between DL and traditional OSS projects, while there was no significant difference observed in the number of authors and newcomers between the two types of projects.

In summary, the results obtained in this study reveal significant differences in the evolution of most activities between DL and traditional OSS projects. Additionally, there also exist significant differences in the workload ratios of maintainers across almost all activities between DL and traditional OSS projects. Specifically, maintainers' workload ratios in DL projects are significantly lower than those in traditional OSS projects, and they show a decreasing trend for most activities. Furthermore, the number of maintainers in DL projects is also significantly lower than that in traditional OSS projects, and it exhibits a fluctuating trend.

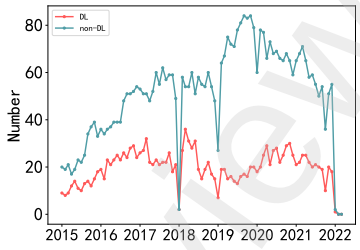


Figure 7: Number of maintainers in DL and traditional OSS projects.

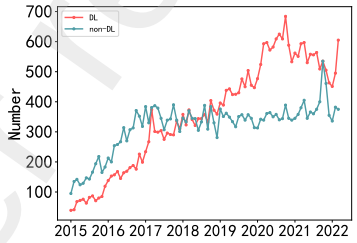


Figure 8: Number of authors in DL and traditional OSS projects.

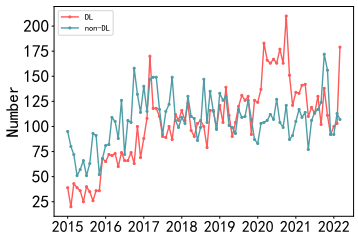


Figure 9: Number of newcomers in DL and traditional OSS projects.

3.1.1. Implications.

Organizations and developers: Organizations and developers who are planning to undertake DL projects should be aware of the significant differences in the evolution of activities between DL and traditional OSS projects, as well as the significant differences in workload ratios that maintainers afford in DL and traditional OSS projects. Especially, as maintainers in DL projects tend to have a lower workload compared to those in traditional OSS projects, therefore, organizations should consider this point when planning and allocating resources for DL projects, and developers should consider this point when choosing OSS projects in different domains to contribute to.

Maintainers: Maintainers in both DL and traditional OSS projects should be aware of this phenomenon, closely monitor their workload ratios in various activities and try their best to maintain a healthy workload distribution. Meanwhile, given the rapid influx of authors and newcomers observed in DL projects, to alleviate the workload of maintainers in user management, we recommend maintainers in DL projects provide DL-specific workflows and detailed readme files to authors and newcomers, making the development process of their managed projects clear and precise. Additionally, we suggest that maintainers examine and identify outstanding authors and integrate them into their group to ensure the long-term sustainability of DL projects from maintainers' perspectives.

Researchers: Meanwhile, researchers in the software engineering domain should recognize the significant discrepancies discussed above between DL and traditional OSS projects, and be cautious when selecting open-source projects in different areas for their studies. Furthermore, the decreasing trend in the workload ratios of maintainers in DL projects for most activities should be further investigated by researchers to identify the reasons behind this trend. They can also investigate the factors contributing to the differences in the evolution of various activities between DL and traditional OSS projects.

Moreover, the number of maintainers in DL projects is significantly lower than the number of maintainers in traditional OSS projects and shows a fluctuating trend. Hence, it highlights the need for further investigation into the factors that influence the number of maintainers in DL projects. Researchers could check the impact of factors such as project goals, company domination patterns, and project complexity on the number of maintainers in DL projects. Researchers can also explore practical strategies to help attract more maintainers. In this way, researchers can help ensure the long-term sustainability of DL projects from maintainers' perspectives.

There is a significant difference in the evolution of most activities between DL and traditional OSS projects, and there is also a significant difference in workload ratios of maintainers in DL and traditional OSS projects. Specifically, maintainers' workload ratios on many activities in DL projects are significantly lower than those in traditional OSS projects, and they show a decreasing trend for most activities.

3.2. RQ3: How does the average workload of maintainers evolve as DL projects grow?

Motivation: Findings in RQ2 reveal a decreasing trend in the workload ratios of maintainers across many activities in DL projects. Meanwhile, the number of maintainers is also showing a fluctuating trend. Therefore, further investigation is required to explore the evolution of the average workload per maintainer afford. Moreover, given the continuous influx of authors and newcomers in DL projects, it remains unclear whether existing maintainers in DL projects can handle the ever-increasing authors and newcomers. Hence, in this RQ, we analyze historical data to investigate the dynamic evolution of maintainers' workload during their maintenance process. Understanding the evolution of maintainers' workloads can help us better understand the sustainability of their work.

Approach: To determine if the workload per maintainer afford has increased with the development of DL and traditional OSS projects, we pictured the monthly changes in the average workload per maintainer afford in our sampled projects. Subsequently, to examine whether there exist significant differences in growth rates of various activities for maintainers in DL and traditional OSS projects, we applied a one-way ANOVA Heiberger and Neuwirth (2009) test to verify it. The null hypothesis is that the growth rates of various activities for maintainers in DL and traditional OSS projects are identical.

Then, to gain insights into the growth trends of maintainers' various activities within each project, we derived the monthly changes in the average workload of maintainers in each sampled project. Due to space constraints, we only present the growth trends for two sampled projects. Take the complex DL project - Tensorflow, and the complex traditional OSS project - React, as examples. We depicted the monthly changes in the average workload of maintainers in these two projects, as shown in Fig. 11 and Fig. 12, respectively.

Subsequently, we further calculated the average monthly growth rates of various activities per maintainer for each project. Accordingly, we have 11 growth rates for each sampled project (comprising 19 DL projects and 19 traditional OSS projects). We then performed a one-way ANOVA test again to verify if there is any difference in the 11 growth rates of maintainers for each sampled project. In this way, we can gain a better understanding of whether there exist significant differences in the average workload of maintainers across various activities.

Result: Our findings indicate that there almost exist no significant differences in the growth rates of average

workload on various activities per maintainer between our sampled DL and traditional OSS projects. Figure 10 displays the monthly changes in average workload per maintainer for our sampled projects. On average, DL project maintainers show increasing workloads in Pull Requests, merged Pull Requests, Pull Request Review Comments, Releases, and Pushes, with a sharp increase during 2020 and 2022. Meanwhile, traditional OSS project maintainers show increasing workloads in Issue Comments, Pull Requests, merged Pull Requests, Pull Request Review Comments, and Releases. Notably, we find that maintainers in DL projects release new versions more frequently than those in traditional OSS projects during 2020 and 2022, which is consistent with previous findings that ML libraries release new versions more frequently Dilhara et al. (2021). Specifically, it is worth noting that different scales are used to emphasize trend similarities among various activities. For instance, the commit number is multiplied by 10, and the number of commit comments is multiplied by 100. Our statistical test results show that there only exists a significant difference ($F_{(1,170)} = 3.717, p < 0.05$) in the growth rate of Commit Comment for maintainers between DL and traditional OSS projects.

The monthly changes in the average workload of maintainers in Tensorflow and React are shown in Fig. 11 and Fig. 12. Results in Fig. 11 demonstrate that maintainers in Tensorflow experienced a high average workload across various activities from 2016 to 2018. However, their workload appears to drop at the beginning of 2019. This finding is in line with the evolutionary history of Tensorflow Han et al. (2020b), where the project was first released in November 2015. Therefore, in its initial stages, most works tend to be accomplished by the project's maintainers. As the project grew, more developers and contributors joined, resulting in a decrease in the average workload per maintainer.

In contrast, Fig. 12 shows that there exist fluctuating trends of the average workload of maintainers in React. In general, per maintainers' average workload in React are stable for most activities during the studied period, while their average workload on Issue Comment, Pull Request, merged Pull Request, and Pull Request Review Comment fluctuates dramatically during this period. Nevertheless, it still presents a decreasing pattern in the long run.

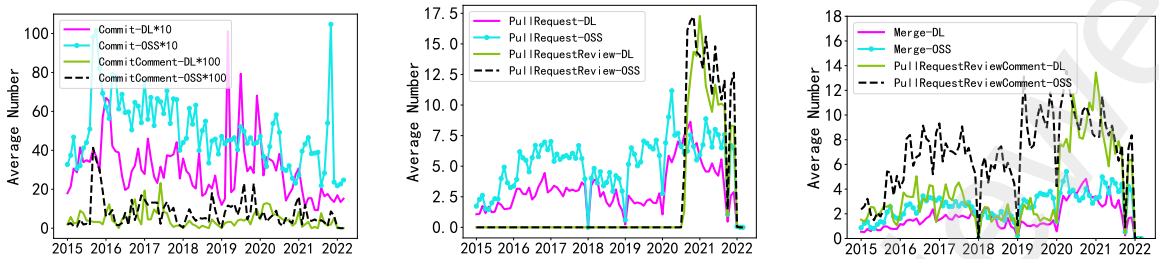
Table 1 uncovers that there indeed have significant differences in the growth rates across various activities of maintainers in most DL projects, with 14 out of 19 (74%) DL projects demonstrating significant differences. The remaining DL projects, such as Transformers, Keras, MMdnn, PyTorch-Lightning, and TensorFlow, are mostly DL frameworks or tools. This indicates that maintainers in these projects present relatively even growth rates on various activities. This finding is consistent with the findings in Fig. 11. Meanwhile, Table 2 reveals that significant differences exist in the growth rates across various activities of maintainers in most traditional OSS projects, with 11 out of 19 (58%) showing significant differences. The remaining projects are mostly platforms or frameworks with a larger size or a larger number of maintainers.

Table 1: Statistical Tests for Growth Rates across Various Activities of Maintainers in Each Sampled DL Project. The Second and Third Columns show F-values and P-values for Average Growth Rates across Various Activities.

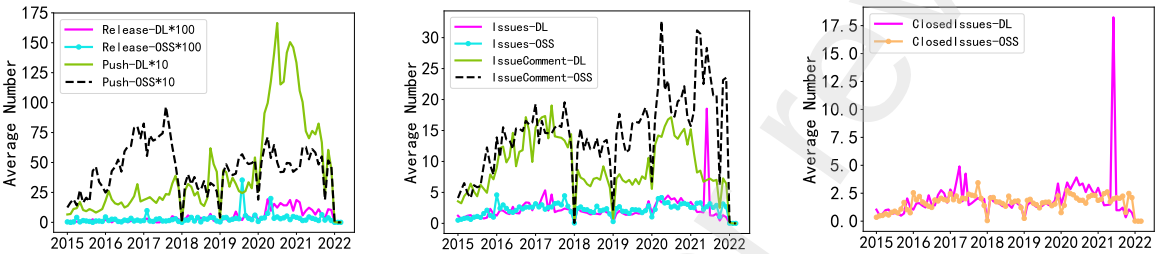
Project	F-value	P-value
caffe	2.088	<0.05 (*)
fastai	3.042	<0.001 (***)
transformers	1.609	>0.05
gocv	5.756	<0.001 (***)
autokeras	4.472	<0.001 (***)
keras	1.004	>0.05
Lasagne	5.428	<0.001 (***)
MMdnn	1.808	>0.05
DeepSpeech	5.957	<0.001 (***)
DIGITS	4.035	<0.001 (***)
mmdetection	4.136	<0.001 (***)
photoprism	3.932	<0.001 (***)
fairseq	2.001	<0.001 (***)
pytorch	2.139	<0.05 (*)
pytorch-lightning	0.977	>0.05
pytorch-image-models	5.787	<0.001 (***)
ncnn	4.527	<0.001 (***)
tensorflow	1.322	>0.05
Theano	10.23	<0.001 (***)

3.2.1. Implications.

Maintainers: Maintainers working on both DL and traditional OSS projects should recognize that, except for the Commit Comment activity, there are no significant differences in the monthly changes of average workload across most activities. Therefore, maintainers who tend to contribute to both types of projects can utilize similar approaches to manage their workload. In addition, as shown in Fig. 10, maintainers in DL and traditional OSS projects all experience an increasing workload in activities related to pull requests, such as submitting, merging, reviewing, and commenting on them. Therefore, maintainers must be mindful of the growth rates of pull request-related activities and identify outstanding contributors to integrate into their team, so that they can ensure the long-term sustainability of their management works. Maintainers can also adopt automatic tools to help them deal with some management tasks, e.g.,

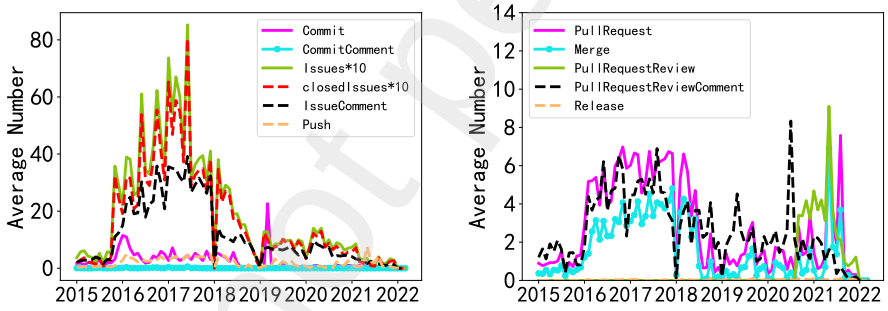


(a) Commit & Commit Comment (b) Pull Request & Pull Request Review (c) merged-Pull Request & Pull Request Review Comment



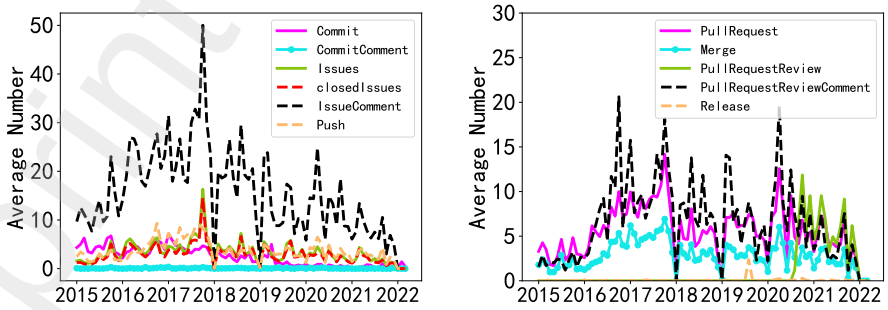
(d) Release & Push (e) Issues & Issue Comment (f) closed-Issues

Figure 10: Comparison of the trends of an average number of activities conducted by per maintainer in DL and OSS projects. Notably, different scales are used to emphasize trend similarities among various activities. For instance, the commit number is multiplied by 10, and the number of commit comments is multiplied by 100.



(a) Activity Set1 (b) Activity Set2

Figure 11: Trends of maintainers' average workload on various activities in Tensorflow.



(a) Activity Set1 (b) Activity Set2

Figure 12: Trends of maintainers' average workload on various activities in React.

adopting GitHub's new detailed code review tools[3] to tackle pull request review and review comments.

Although the monthly changes in the average work-

[3]https://github.com/features/code-review/

Table 2: Statistical Tests for Growth Rates across Various Activities of Maintainers in Each Sampled Traditional OSS Project. The Second and Third Columns show F-values and P-values for Average Growth Rates across Various Activities.

Project	F-value	P-value
aframe	4.575	<0.001 (***)
goaccess	6.278	<0.001 (***)
airflow	1.058	>0.05
openwhisk	2.486	<0.001 (***)
superset	1.601	>0.05
zookeeper	8.576	<0.01 (**)
fresco	6.445	<0.001 (***)
jest	3.23	<0.001 (***)
react	1.103	>0.05
gardener	4.006	<0.001 (***)
ExoPlayer	0.843	>0.05
ZeroNet	17.17	<0.01 (**)
manageiq	0.783	>0.05
Moya	4.261	>0.05
tidb	0.935	>0.05
rclone	4.51	<0.001 (***)
tesseract	2.489	<0.01 (**)
zulip	1.279	>0.05

load of maintainers for different activities almost have no significant differences between DL and OSS projects, when it comes to the individual perspective, we found significant differences in the growth rates of maintainers' average workload for different activities within individual projects. In this regard, a proportion of 74% DL projects confirmed the above findings, compared to 58% in traditional OSS projects (see Table 1 and 2). Therefore, maintainers in DL projects should pay more attention to their workload distribution on various activities over time, and tailor maintenance strategies accordingly.

Besides, as illustrated in Fig. 10, there is also a rising workload of issue comment activities for maintainers in traditional OSS projects. Therefore, we emphasize the importance that maintainers should provide clear and actionable documentation for practitioners. In this way, more and more practitioners will become familiar with the details of OSS projects, which in turn, reduces the number of issues and also reduces maintainers' workload on issue comments.

Researchers: Researchers in the software engineering domain should recognize the significant differences in the growth rates of maintainers' average workload for different activities within individual projects, especially in individual DL projects. They should be cautious when selecting open-source projects in different domains to study maintainers' works. Furthermore, they could consider a larger scale of DL and traditional OSS projects to verify if this finding is also applicable to most DL and traditional OSS projects.

Moreover, to alleviate maintainers' workload in pull request-related activities, researchers can explore the development of automatic code generation tools based on existing studies Hu et al. (2019); Bernaschina et al. (2019); Liu et al. (2020), derive automatic code review tools on top of existing studies Chen and Zhou (2018), and help to process code comments automatically. Furthermore, they can also contribute to generating usable models that help maintainers find available contributors to engage in dealing with management tasks, such as the multiple-committer model adopted in the Linux kernel community Tan et al. (2020a).

> Except for the activity of Commit Comment, there are no significant differences in the growth rates of average workload on various activities per maintainer between sampled DL projects and traditional OSS projects. Regarding each project, our findings show that most DL projects demonstrate significant differences in growth rates across various activities of maintainers, and traditional OSS projects with a larger size tend to show no significant differences in growth rates across various activities for maintainers.

3.3. RQ4: What are the relationships between maintainers' activities and the sustainability of DL projects?

Motivation: Maintainers perform various activities when they maintain DL projects, and their activities may influence the sustainability of DL projects. Therefore, we investigate the relationships between maintainers' monthly activities and the sustainability of DL projects, and how the evolution of maintainers' activities influences the sustainability of DL projects.

Approach: In this section, we use the metric defined in Section 2.2 to assess project sustainability. We then conduct a Spearman's rank correlation test Sedgwick (2014); Borges et al. (2016) to examine the correlation between maintainers' monthly activities and project sustainability in DL projects. We also compare these results with those of traditional OSS projects.

Results: Our findings uncover that the relationships between project sustainability and maintainer activities differ significantly between DL and traditional OSS projects. Specifically, only one type of maintainer activity, i.e., the number of pull request review comments made by maintainers, exhibits a similar correlation with sustainability in both DL and OSS projects. Other maintainer activities all have different correlations with project sustainability in DL and OSS projects. For instance, the number of pull requests merged by maintain-

ers, the number of releases and pushes made by maintainers are positively and moderately correlated with the sustainability of DL projects. However, the number of pull requests merged by maintainers only has a low correlation with the sustainability of traditional OSS projects, and the number of releases has no correlation with the sustainability. Fig. 13 illustrates the monthly changes in the number of commits for DL and traditional OSS projects, and Fig. 14(a) and Fig. 14(b) show the relationships between the number of commits and maintainers' activities in DL and traditional OSS projects, respectively.

Fig. 14(a) reveals that there is a high positive correlation between the number of commits in DL projects and the number of pushes made by maintainers, and a moderate correlation between the number of commits and the number of pull requests merged by maintainers and the number of releases made by maintainers. This implies that maintainers' push, merge, and release activities have a positive and relatively high effect on the sustainability of DL projects. However, there only exists a low correlation between the number of commits and the number of issues, closed issues, pull requests, pull request reviews, and pull request review comments. Additionally, there is no significant relationship between the sustainability of DL projects and the number of commit comments and issue comments made by maintainers.

By comparing with traditional OSS projects, we also obtained the relationships between the indicators of sustainability and maintainers' activities in traditional OSS projects, which is shown in Fig. 14(b). We can observe that the number of commits is positively and moderately correlated with the number of commit comments, issues, closed issues, pull requests, and pushes made by maintainers. However, there only exists a low correlation between the number of commits and the number of issue comments, pull requests merged by maintainers, and pull request review comments made by maintainers. This result is somewhat different from DL projects, where pull requests merged by maintainers have a moderate correlation with DL projects' sustainability. In addition, there is no significant relationship between the sustainability of DL projects and the number of pull request reviews and releases made by maintainers.

3.3.1. Implications.

Maintainers: Based on the findings, it is important for maintainers to consider the differences between DL and traditional OSS projects in terms of project sustainability. For example, through maintaining a high number of pull requests merged, releases, and pushes, maintainers in DL projects can positively and highly impact the sustainability of DL projects. However, these activities made by maintainers may not be effective in traditional OSS projects. Hence, maintainers in DL and traditional OSS projects may need to focus on specific activities that are effective in improving the sustainability of projects. In this way, they can ensure the long-term sustainability of projects from maintainers' perspectives.

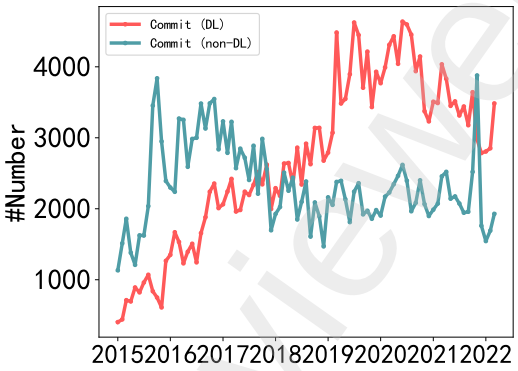


Figure 13: Sustainability of DL and traditional OSS projects, using the metric of the number of commits.

Researchers: Results in this RQ derive empirical evidence of the relationships between maintainers' activities and the sustainability of DL and traditional OSS projects. These preliminary results call for further investigations to establish definitive causal relationships between the indicators of sustainability and maintainers' activities. Researchers can delve deeper into various other factors that may impact the sustainability and compare DL projects with traditional OSS projects. By doing so, we can gain a better understanding of how maintainers can effectively contribute to sustainable projects.

Furthermore, as previous studies Dilhara et al. (2021) stated that the proportion of new Python projects that depend on ML libraries has increased from 2% in 2013 to 50% in 2018; hence, we recommend software engineering researchers use our findings as a starting point to investigate the fine-grained tasks that maintainers faced, and explore the sustained maintenance practices specific to DL projects. By doing so, we can better understand the unique challenges and opportunities in managing DL projects, promote sustainable development of DL projects, and also provide guidance to many other OSS projects.

(a) DL (b) OSS

Figure 14: Relationships between the sustainability of projects (i.e., the number of commits) and maintainers' activities in DL and OSS projects. The blue color indicates a positive correlation, and the red color indicates a negative correlation. The darker the color is, the stronger the correlation is. "X" implies that the correlation is insignificant Dong et al. (2020). An absolute value of correlation that is less than 0.4 means a low correlation, an absolute value of correlation that ranges from 0.4 to 0.7 means a moderate correlation, an absolute value of correlation that ranges from 0.7 to 0.9 means a high correlation, and an absolute value of correlation that great than 0.9 means a very high correlation Guilford (1950).

The study reveals that the correlation between project sustainability and maintainer activities differs substantially between DL and traditional OSS projects. For instance, the number of pull requests merged, the number of releases and pushes made by maintainers are positively and moderately correlated with the sustainability of DL projects, while the number of pull requests merged by maintainers only has a low correlation with the sustainability of traditional OSS projects, and the number of releases has no correlation with the sustainability of traditional OSS projects.

4. Related Work

In this section, we describe the related works regarding the behaviors and works of maintainers, the evolution and sustainability of open source projects, and the particular analyses of deep learning projects.

4.1. The behaviors and works of maintainers

A tremendous amount of research effort has focused on studying developers' works and the evolution of developer communities Sonnentag (1995); Mockus et al. (2000); Zhou and Mockus (2010, 2012); Bao et al. (2019); Wang et al. (2020). Closely related to our work, Wang et al. Wang et al. (2020) conducted an empirical study to investigate elite developers' fine-grained activities in open source projects, and studied the impacts of these activities on projects' quality and productivity. Although they revealed a set of tasks performed by elite developers, it is unclear whether these elite developers are maintainers or not.

Hence, there is still little research on the observation of maintainers and maintainers' works. Among them, Dias et al. Dias et al. (2021) investigated to unveil the unique attributes that great OSS maintainers might have. Eghbal et al. Eghbal (2020) analyzed the OSS maintenance process and found that some hidden costs of maintaining OSS projects exist. Zhou et al. Zhou et al. (2017) reported an empirical study to understand the scalability of the Linux Kernel and paid more attention to characterizing the workload of maintainers. Although Zhou et al.'s work is closely related to ours, their paper only focused on the technical works (i.e., the number of commits, files, authors, and new joiners) of maintainers, but neglected the fact that maintainers are also responsible for many other tasks. Hence, in this study, we study maintainers' activities from various dimensions to deeper understand maintainers' tasks in DL projects and traditional OSS projects.

4.2. The evolution and sustainability of open source projects

As open source software evolves continuously, it becomes increasingly large and complex Lehman et al. (1997). Due to its longevous evolution, there have yielded a considerable body of studies to investigate the evolution and sustainability of software evolution. Among them, Lehman et al. Lehman et al. (1997) initially elaborated on the laws of software evolution, whereas Scacchi et al. Scacchi (2003) conducted a study to examine whether and how the evolution of open source software conforms to the laws of traditional software evolution. Mockus et al. Mockus et al. (2000) examined the development process of the Apache web server. By extracting data from email archives and issue reports, they generated several critical views of the OSS project, including developer participation, core team size, code ownership, defect density, productivity, and problem resolution interval.

Prior works are concentrated on the evolution and sustainability of open source projects, whereas our study focuses on maintainers' activities and the sustainability of DL prjects, and compares with traditional OSS projects, which has not been investigated in previous studies.

4.3. The analyses of deep learning systems

Deep learning techniques grow at a rapid pace, which has led to a tremendous amount of empirical research effort. Among them, a considerable body of literature puts their effort into studying bugs, failures, and faults of deep learning projects Thung et al. (2012); Zhang et al. (2018); Islam et al. (2019, 2020); Zhang et al. (2020); Humbatova et al. (2020). For instance, Thung et al. Thung et al. (2012) conducted an empirical study on the bugs in machine learning systems, to find a sample set of bugs and corresponding fixes. Zhang et al. Zhang et al. (2018) collected program bugs existed in deep learning projects that depend on TensorFlow and endeavored to determine the root causes and symptoms of these bugs.

Simultaneously, there also exist some empirical studies exploring the migration process of deep learning libraries. Han et al. Han et al. (2020a) put their effort into dependency networks of deep learning libraries. They studied the dependency degrees that projects depend on deep learning libraries, the update behaviors, and reasons when updating deep learning libraries, and the version distributions of deep learning projects. Dilhara et al. Dilhara et al. (2021) examined to mine how developers in Software-2.0 use deep learning libraries, and whether or not the deep learning library evolution affects their code. Moreover, they also expounded on the challenges of DL library evolution by performing a survey on developers involving deep learning libraries.

Most prior studies focused on the program bugs/failures/faults of deep learning projects or the migration process of DL libraries. However, these prior studies have not explored maintainers' activities, the evolution of maintainers' activities, not to mention the evolution of DL projects and the relationships between maintainers' activities and the sustainability of DL projects, which is studied in our paper.

5. Threats to Validity

Internal Threats. One potential threat can be attributed to the selection of DL and traditional OSS projects. To mitigate this threat, in this study, we select 5 popular DL frameworks and 14 other DL software projects to compose our DL dataset. We also select 19 traditional OSS projects with diverse application domains to make a comparison. Another threat may be the number of selected projects. Although having more projects is desirable, practically, to ensure the feasibility of manual observations and generate reasonable statistical results, we comply with rules defined in Kalliamvakou et al.'s study Kalliamvakou et al. (2014) and GitHub's annual report git (2020). In this regard, we select projects that are representative in DL and traditional OSS domains and have sufficient records to trace in GitHub. We also manually check the uniqueness and verify the correctness of the extracted data, where our results declare the correctness of our dataset.

The other internal threat is related to the determination of maintainers' workload. To alleviate this threat, we surveyed maintainers and asked for their help choosing activities they performed in their maintenance process. After that, we analyzed their responses and determined the maintainers' main workload according to their choices.

External Threats. The uniqueness of DL projects limits its external validity. As we gather DL projects to study their evolution and sustainability, therefore, our findings focused on DL projects may not be generalizable to other OSS projects. However, we compare the findings in DL projects with traditional OSS projects, therefore, the findings in traditional OSS projects can also be generalizable to other OSS projects. Hence, we believe that our findings can be helpful to both DL and OSS projects.

Construct Threats. Since there are multiple definitions of the sustainability of OSS projects, researchers who

investigated the sustainability of OSS projects from different angles would adopt different definitions. Some studies Crowston et al. (2006); Valiev et al. (2018); Yin et al. (2021) employed the success of OSS projects as the indicator of the sustainability of OSS projects. However, there also exist many other indicators such as transcribing the sustainability of OSS projects to metrics of productivity and popularity Yin et al. (2021), and transcribing the sustainability of the open source community to the attraction and retention of newcomers in the community Valiev et al. (2018). To mitigate the threat of determining the definition of sustainability, we adopt the indicator – the number of commits submitted, to characterize the sustainability of DL projects. In this way, we can characterize the sustainability of DL projects dynamically.

6. Conclusion and Future Work

In this paper, we conduct an empirical study to explore maintainers' workload in DL projects, investigate the evolution and sustainability of DL projects, understand the workload evolution of maintainers, and compare them with traditional OSS projects. To achieve that, we collect 19 DL projects and 19 traditional OSS projects, and extract all the histories of activities of these projects to characterize the workload of maintainers holistically (RQ1), the evolution of DL projects (RQ2), and the workload evolution of maintainers (RQ3), as well as the relationships between maintainers' activities and the sustainability of DL projects (RQ4). Our analysis uncovers the following findings: 1) there exists a significant difference in the evolution of most activities between DL and traditional OSS projects, and maintainers in DL projects afford significantly lower workloads than maintainers in traditional OSS projects; 2) although DL projects show increasing trends on most types of activities, maintainers' workload on most activities show a decreasing trend, which is quite different with traditional OSS projects; 3) there only exists a significant difference in the growth rates of average workload on Commit Comment of maintainers between DL and traditional OSS projects; 4) there exist positive and moderate correlations between the sustainability of DL projects and maintainers' releases and pushes as well as the number of pull requests merged by maintainers. However, the number of releases of maintainers does not correlate with the sustainability of traditional OSS projects.

In the future, we plan to consider more DL projects with various sizes to expand the generalization of our results. Moreover, we also encourage further studies to extend our work, e.g., to find the definitive causalities between the sustainability of DL projects and other factors, to generate actionable tools that can recommend proper contributors to maintainers in DL projects, etc. To facilitate replications or other types of future work, we make the data and scripts used in this study publicly available at `https://github.com/HJXPaperData/SustainabilityofDL`.

References

, 2020. Github annual report. URL: `https://octoverse.github.com/`.

, 2021. Gh archive. URL: `https://www.gharchive.org/`.

, 2021. Google bigquery. URL: `https://cloud.google.com/bigquery/docs`.

Acuna, D., Kar, A., Fidler, S., 2019. Devil is in the edges: Learning semantic boundaries from noisy annotations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11075–11083.

Bao, L., Xia, X., Lo, D., Murphy, G.C., 2019. A large scale study of long-time contributor prediction for github projects. IEEE Transactions on Software Engineering .

Bernaschina, C., Falzone, E., Fraternali, P., Gonzalez, S.L.H., 2019. The virtual developer: Integrating code generation and manual development with conflict resolution. ACM Transactions on Software Engineering and Methodology (TOSEM) 28, 1–38.

Borges, H., Hora, A., Valente, M.T., 2016. Understanding the factors that impact the popularity of github repositories, in: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE. pp. 334–344.

Cambronero, J., Li, H., Kim, S., Sen, K., Chandra, S., 2019. When deep learning met code search, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 964–974.

Chen, C., Seff, A., Kornhauser, A., Xiao, J., 2015. Deepdriving: Learning affordance for direct perception in autonomous driving, in: Proceedings of the IEEE international conference on computer vision, pp. 2722–2730.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., Blaschke, T., 2018. The rise of deep learning in drug discovery. Drug discovery today 23, 1241–1250.

Chen, Q., Zhou, M., 2018. A neural framework for retrieval and summarization of source code, in: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE. pp. 826–831.

Ciregan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification, in: 2012 IEEE conference on computer vision and pattern recognition, IEEE. pp. 3642–3649.

Coelho, J., Valente, M.T., 2017. Why modern open source projects fail, in: Proceedings of the 2017 11th Joint meeting on foundations of software engineering, pp. 186–196.

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educational and psychological measurement 20, 37–46.

Crowston, K., Howison, J., Annabi, H., 2006. Information systems success in free and open source software development: Theory and measures. Software Process: Improvement and Practice 11, 123–148.

Dias, E., Meirelles, P., Castor, F., Steinmacher, I., Wiese, I., Pinto, G., 2021. What makes a great maintainer of open source projects?, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE. pp. 982–994.

Dilhara, M., Ketkar, A., Dig, D., 2021. Understanding software-2.0: a study of machine learning library usage and evolution. ACM Transactions on Software Engineering and Methodology (TOSEM) 30, 1–42.

Dong, Y., Zhang, P., Wang, J., Liu, S., Sun, J., Hao, J., Wang, X., Wang, L., Dong, J., Dai, T., 2020. An empirical study on correlation between coverage and robustness for deep neural networks, in: 2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS), IEEE. pp. 73–82.

Eghbal, N., 2020. Working in public: the making and maintenance of open source software. Stripe Press.

Guilford, J.P., 1950. Fundamental statistics in psychology and education .

Guo, Q., Chen, S., Xie, X., Ma, L., Hu, Q., Liu, H., Liu, Y., Zhao, J., Li, X., 2019. An empirical study towards characterizing deep learning development and deployment across different frameworks and platforms, in: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE. pp. 810–822.

Han, J., Deng, S., Lo, D., Zhi, C., Yin, J., Xia, X., 2020a. An empirical study of the dependency networks of deep learning libraries, in: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE. pp. 868–878.

Han, J., Deng, S., Lo, D., Zhi, C., Yin, J., Xia, X., 2021. An empirical study of the landscape of open source projects in baidu, alibaba, and tencent, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE. pp. 298–307.

Han, J., Shihab, E., Wan, Z., Deng, S., Xia, X., 2020b. What do programmers discuss about deep learning frameworks. Empirical Software Engineering 25, 2694–2747.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Heiberger, R.M., Neuwirth, E., 2009. One-way anova, in: R through excel. Springer, pp. 165–191.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine 29, 82–97.

Hu, X., Men, R., Li, G., Jin, Z., 2019. Deep-autocoder: Learning to complete code precisely with induced code tokens, in: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), IEEE. pp. 159–168.

Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A., Tonella, P., 2020. Taxonomy of real faults in deep learning systems, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, pp. 1110–1121.

Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., et al., 2015. An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:1504.01716 .

Islam, M.J., Nguyen, G., Pan, R., Rajan, H., 2019. A comprehensive study on deep learning bug characteristics, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 510–520.

Islam, M.J., Pan, R., Nguyen, G., Rajan, H., 2020. Repairing deep neural networks: Fix patterns and challenges, in: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), IEEE. pp. 1135–1146.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D., 2014. The promises and perils of mining github, in: Proceedings of the 11th working conference on mining software repositories, pp. 92–101.

LaToza, T.D., Venolia, G., DeLine, R., 2006. Maintaining mental models: a study of developer work habits, in: Proceedings of the 28th international conference on Software engineering, pp. 492–501.

Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., Turski, W.M., 1997. Metrics and laws of software evolution-the nineties view, in: Proceedings Fourth International Software Metrics Symposium, IEEE. pp. 20–32.

Liu, F., Li, G., Zhao, Y., Jin, Z., 2020. Multi-task learning based pre-trained language model for code completion, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pp. 473–485.

Manotas, I., Bird, C., Zhang, R., Shepherd, D., Jaspan, C., Sadowski, C., Pollock, L., Clause, J., 2016. An empirical study of practitioners' perspectives on green software engineering, in: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), IEEE. pp. 237–248.

Massey Jr, F.J., 1951. The kolmogorov-smirnov test for goodness of fit. Journal of the American statistical Association 46, 68–78.

Mathew, G., Stolee, K.T., 2021. Cross-language code search using static and dynamic analyses. arXiv preprint arXiv:2106.09173 .

Mendez, C., Padala, H.S., Steine-Hanson, Z., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L., Patil, N., Sarma, A., Burnett, M., 2018. Open source barriers to entry, revisited: A sociotechnical perspective, in: Proceedings of the 40th International Conference on Software Engineering, pp. 1004–1015.

Mockus, A., Fielding, R.T., Herbsleb, J., 2000. A case study of open source software development: the apache server, in: Proceedings of the 22nd international conference on Software engineering, pp. 263–272.

Nassif, A.B., Shahin, I., Attili, I., Azzeh, M., Shaalan, K., 2019. Speech recognition using deep neural networks: A systematic review. IEEE access 7, 19143–19165.

Noman, H., Mahoto, N.A., Bhatti, S., Abosaq, H.A., Al Reshan, M.S., Shaikh, A., 2022. An exploratory study of software sustainability at early stages of software development. Sustainability 14, 8596.

Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., et al., 2018. Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 .

Penzenstadler, B., Bauer, V., Calero, C., Franch, X., 2012. Sustainability in software engineering: a systematic literature review, in: 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), IET, Ciudad Real, Spain. pp. 32–41. URL: https://digital-library.theiet.org/content/conferences/10.1049/ic.2012.0004, doi:10.1049/ic.2012.0004.

Petersen, J., Jäger, P.F., Isensee, F., Kohl, S.A., Neuberger, U., Wick, W., Debus, J., Heiland, S., Bendszus, M., Kickingereder, P., et al., 2019. Deep probabilistic modeling of glioma growth, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer. pp. 806–814.

Pradel, M., Sen, K., 2018. Deepbugs: A learning approach to name-based bug detection. Proceedings of the ACM on Programming Languages 2, 1–25.

Qiu, H.S., Nolte, A., Brown, A., Serebrenik, A., Vasilescu, B., 2019. Going farther together: The impact of social capital on sustained participation in open source, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE. pp. 688–699.

Roy, A., Sun, J., Mahoney, R., Alonzi, L., Adams, S., Beling, P., 2018. Deep learning detecting fraud in credit card transactions, in: 2018 Systems and Information Engineering Design Symposium (SIEDS), IEEE. pp. 129–134.

Scacchi, W., 2003. Understanding open source software evolution.

applying, breaking and rethinking the laws of software evolution.
Sedgwick, P., 2014. Spearman's rank correlation coefficient. Bmj 349.
Sonnenburg, S., Braun, M.L., Ong, C.S., Bengio, S., Bottou, L., Holmes, G., LeCunn, Y., Muller, K.R., Pereira, F., Rasmussen, C.E., et al., 2007. The need for open source software in machine learning .
Sonnentag, S., 1995. Excellent software professionals: Experience, work activities, and perception by peers. Behaviour & Information Technology 14, 289–299.
Tan, X., Zhou, M., Fitzgerald, B., 2020a. Scaling open source communities: an empirical study of the linux kernel, in: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), IEEE. pp. 1222–1234.
Tan, X., Zhou, M., Sun, Z., 2020b. A first look at good first issues on github, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 398–409.
Thung, F., Wang, S., Lo, D., Jiang, L., 2012. An empirical study of bugs in machine learning systems, in: 2012 IEEE 23rd International Symposium on Software Reliability Engineering, IEEE. pp. 271–280.
Tian, Y., Pei, K., Jana, S., Ray, B., 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars, in: Proceedings of the 40th international conference on software engineering, pp. 303–314.
Trinkenreich, B., Guizani, M., Wiese, I., Conte, T., Gerosa, M., Sarma, A., Steinmacher, I., 2021. Pots of gold at the end of the rainbow: What is success for open source contributors? IEEE Transactions on Software Engineering 48, 3940–3953.
Valiev, M., Vasilescu, B., Herbsleb, J., 2018. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the pypi ecosystem, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 644–655.
Wan, Z., Xia, X., Lo, D., Murphy, G.C., 2019. How does machine learning change software development practices? IEEE Transactions on Software Engineering 47, 1857–1871.
Wang, Z., Feng, Y., Wang, Y., Jones, J.A., Redmiles, D., 2020. Unveiling elite developers' activities in open source projects. ACM Transactions on Software Engineering and Methodology (TOSEM) 29, 1–35.
Wei, B., Li, G., Xia, X., Fu, Z., Jin, Z., 2019. Code generation as a dual task of code summarization. arXiv preprint arXiv:1910.05923 .
Yin, L., Chen, Z., Xuan, Q., Filkov, V., 2021. Sustainability forecasting for apache incubator projects. arXiv preprint arXiv:2105.14252 .
Zhang, R., Xiao, W., Zhang, H., Liu, Y., Lin, H., Yang, M., 2020. An empirical study on program failures of deep learning jobs, in: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE), IEEE. pp. 1159–1170.
Zhang, Y., Chen, Y., Cheung, S.C., Xiong, Y., Zhang, L., 2018. An empirical study on tensorflow program bugs, in: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 129–140.
Zhang, Y., Stol, K.J., Liu, H., Zhou, M., 2022. Corporate dominance in open source ecosystems: a case study of openstack, in: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1048–1060.
Zhou, M., Chen, Q., Mockus, A., Wu, F., 2017. On the scalability of linux kernel maintainers' work, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 27–37.
Zhou, M., Mockus, A., 2010. Developer fluency: Achieving true mastery in software projects, in: Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, pp. 137–146.
Zhou, M., Mockus, A., 2012. What make long term contributors: Willingness and opportunity in oss community, in: 2012 34th International Conference on Software Engineering (ICSE), IEEE. pp. 518–528.