

On the sustainability of deep learning projects: Maintainers' perspective

Junxiao Han¹  | Jiakun Liu² | David Lo² | Chen Zhi³ | Yishan Chen⁴ | Shuguang Deng⁵

¹Big Graph Center, School of Computer & Computing Science, Hangzhou City University, Hangzhou, China

²School of Computing and Information Systems, Singapore Management University, Singapore, Singapore

³School of Software Technology, Zhejiang University, Ningbo, China

⁴College of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, China

⁵College of Computer Science and Technology, Zhejiang University, Hangzhou, China

Correspondence

Jiakun Liu, School of Computing and Information Systems, Singapore Management University, Singapore, Singapore.

Email: jkliu@smu.edu.sg

Funding information

The Scientific research project of Zhejiang Provincial Education Department, Grant/Award Number: Y202351453; National Science Foundation of China, Grant/Award Number: U20A20173

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 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 growth in DL projects, and compare them with traditional open-source software (OSS) projects. In this regard, we first investigate how DL projects grow, then, understand maintainers' workload in DL projects, and explore the workload growth 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 conducted 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.

KEY WORDS

deep learning, maintainers, sustainability, workload

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 DL algorithms for various applications such as image processing,^{1–3} speech recognition,^{4,5} autonomous driving,^{6–8} disease diagnosis and drug discovery,^{9–11} and financial fraud detection.¹² However, compared with open-source software (OSS) projects, DL projects have an additional set of DL-specific issues.¹³ For instance, 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.¹⁴

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^{13,15}. The concept of sustainability is certainly multifaceted, and no

general definition exists. In this regard, Valiev et al.¹⁶ denote a project as dormant if it has an average of less than one commit per month in the 12 months prior to its most recent commit. Zhang et al.¹⁷ denote a project as dormant if it has not received any commit for more than 6 months after its most recent commit. These two studies all adopt the frequency of commits to determine the sustainability of OSS projects. Therefore, in this study, we also adopt the frequency of commits to determine the sustainability. Instead of generating labels for sustainability, we use the number of commits over time to characterize the sustainability of projects. In this way, we can better understand the relationships between maintainers' monthly activities and the sustainability of DL projects over time.

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. 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,^{18,19} 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), along with understanding the growth 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 DL and traditional OSS projects grow and whether the growth 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?

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 and traditional OSS projects grow? How do the overall workloads of maintainers grow? And if there exist differences regarding these between DL and traditional OSS projects?

We observe that there is a significant difference in the growth 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,

Performing some experiments, I noticed (Feb 3, 2018) that the LR Finder didn't work on small dataset.

I reported the issue in this blog post: <http://forums.fast.ai/t/lesson-1-my-experiments-with-resnet34-and-some-questions/10857>

Jeremy kindly encouraged me (as he often does with practitioners) in modifying, with his help, the code so that the finder could run more epochs in case of small dataset.

I did it and reported the full code later in the same thread (proposing some other modification ideas).

No one acknowledged, but the bosses are pretty busy boys/girls, so I just waited.

Then, on 20 March 2018 I wrote a PM to Jeremy. It was viewed but not answered. Probably he cannot answer to all the PMs he gets, so I waited.

A week ago, I wrote another forum post, tagging Jeremy, Rachel, and Resham. I waited, but no one answered.

Yesterday (19 June 2018) I decided to open a pull request. In those last months I obviously maintained my environment up to date. So I opened learner.py to check the code for such a pull request, and doh! My proposed modification to the code was there, just with other names for functions and params (e.g. num_it instead of run_for)..!

Now like I said I don't want to complain, and obviously I do not have any claim upon the code (I'm just a student), but why not to say just "Hey boy, we put it into the code."

No matter how little my contribution, It would have been reason of immense satisfaction for me.

FIGURE 1 An issue (#575*) in the fastai project. This example shows that the deep learning (DL) project maintainers did not respond to a pull request for a long time.

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 performed on many activities in DL projects is significantly lower than those in traditional OSS projects.

RQ3. How does the average workload of maintainers grow as DL and traditional OSS projects evolve? And if there exist differences between them?

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 sustainability of DL and traditional OSS 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 in the growth of activities between DL and traditional OSS projects, as well as the significant differences in the proportion of workload that maintainers performed 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 maintainers' perspective. By comparing with traditional OSS projects, we understand the specificity of the growth of DL projects, the specificity of DL maintainers' workload, and its growth. 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 growth 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. For DL projects, we selected the top 15 most starred projects that depend on five popular and widely investigated DL frameworks,^{20–22} respectively, where the five DL frameworks are TensorFlow[†], PyTorch[‡], Caffe[§], Keras[¶], and Theano[#]. In this regard, 75 DL projects are collected. To safeguard the quality of the dataset, we then use the following rules to select targeted DL 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) and (2) DL projects that are tutorials, examples, courses, handbooks, or learning notes were excluded. After that, we manually observe the remaining DL projects, excluding those DL projects with lower contributors or commits, and finally, 19 projects remained.

For OSS projects, we adopted the 20 OSS projects published in Wang et al.'s²³ study and further obtained the top 10 most starred projects that come from Apache and OpenStack ecosystems, respectively. The rationale why we further select these two ecosystems is that they are

representative in traditional OSS domains and are widely investigated by many researches.^{24,25} As a result, we obtained 40 traditional OSS projects. To safeguard the quality of the dataset, we conducted the filtering rules applied to DL projects again and further selected traditional OSS projects 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`). Finally, we also obtained 19 traditional OSS projects.

Table 1 lists the names of DL and traditional OSS projects, and detailed descriptions are publicly available online (<https://github.com/HJXPaperData/SustainabilityofDL>). The average age 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 toward the characteristics between the two constructed datasets. They are typically large in size, have a relatively long traceable history, and involve a significant number of commits and contributors.

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 project²⁶ and is available as a public dataset on Google BigQuery.²⁷ 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.^{23,26} Consequently, we obtained 15 activity types with about 12 GB for the 19 DL projects and about 16 GB 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.²⁸

2.2 | Characterizing the project sustainability

The definition of sustainability is multifaceted.^{16,28–33} Most prior studies focused on investigating the success of OSS projects.^{28,34,35} In this regard, Joblin et al³⁵ defined successful projects as those who have long-term popularity, exhibiting highly active development histories with tens of thousands of commits, and within a large development community with hundreds to thousands of contributors. Coelho et al²⁸ characterized failed projects as those who achieved a high level of popularity but eventually abandoned.

Nevertheless, there are still several studies that investigated the sustainability of OSS projects. Valiev et al¹⁶ denoted a project as dormant if it has an average of less than one commit per month in the 12 months prior to its most recent commit. Otherwise, it can be regarded as sustainable. Zhang et al¹⁷ defined a project as dormant if it has not received any commit for more than 6 months after its most recent commit. In this

TABLE 1 List of collected DL and traditional OSS projects.

DL project	OSS project
caffe	aframe
fastai	goaccess
transformers	airflow
gocv	openwhisk
autokeras	superset
keras	zookeeper
Lasagne	fresco
MMDnn	jest
DeepSpeech	react
DIGITS	gardener
mmdetection	ExoPlayer
photoprism	ZeroNet
fairseq	manageiq
pytorch	Moya
pytorch-lightning	tidb
pytorch-image-models	rclone
ncnn	tesseract
tensorflow	finagle
Theano	zulip

Abbreviations: DL, deep learning; OSS, open-source software.

regard, Valiev et al¹⁶ and Zhang et al¹⁷ all defined project sustainability from the perspective of commit frequency. Moreover, Noman et al³² revealed that the technical (code-based) aspect of sustainability is considered most important by a majority of professionals, since most relevant studies define sustainability from technical perspective and commit frequency. Therefore, to analyze the relationships between maintainers' monthly activities and project's dynamic sustainability (detailed in Section 3.3), we thus used the monthly number of commits to approximately characterize the dynamic sustainability of projects.

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,³⁶ for example, communications with other developers and users. To obtain the list of maintainers, Zhou et al¹⁸ 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 2615 committers for DL projects and 2166 committers for traditional OSS projects. After that, we surveyed these 4781 committers 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^{37,38} 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 study³⁹), and (5) discussing and determining the final themes. We then computed Cohen's Kappa value⁴⁰ 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 | Identification 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 conduct 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 the following: 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.

The number of issues, closed issues, issue comments, commits, commit comments, push, pull requests, merged pull requests, pull request reviews, pull request review comments, and releases are adopted to characterize the workload of maintainers holistically.

3 | RESULTS

This section presents the results for RQ2-4.

3.1 | RQ2: How do DL and traditional OSS projects grow? How do the overall workloads of maintainers grow? And if there exist differences regarding these between DL and traditional OSS projects?

Motivation: Here, we would like to have a basic understanding of how DL projects grow. Meanwhile, we would like to form a first impression on how much workload is performed by maintainers among various activities in DL projects. 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 perform the Kolmogorov-Smirnov test⁴¹ 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 conducted 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 the proportions of workload for 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 the proportions of workload.

Furthermore, we also depict the growth trends of maintainers, authors, and newcomers in DL projects and compare them with those of traditional OSS projects. To do so, we count 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 toward 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, that is, 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 conducted 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. Figures 2 to 6 depict the trend lines of sampled activities in DL and traditional OSS projects, along with the proportion of workload handled by maintainers.

Specifically, Figure 2 shows the number of commits in DL and traditional OSS projects per month, as well as the proportion of workload that is conducted 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 made by maintainers over all commits of projects has a decreasing trend over time, while maintainers in traditional OSS projects have a higher proportion of commits than maintainers in DL projects. However, no significant difference is found in the proportion of workload between maintainers in DL and traditional OSS projects.

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

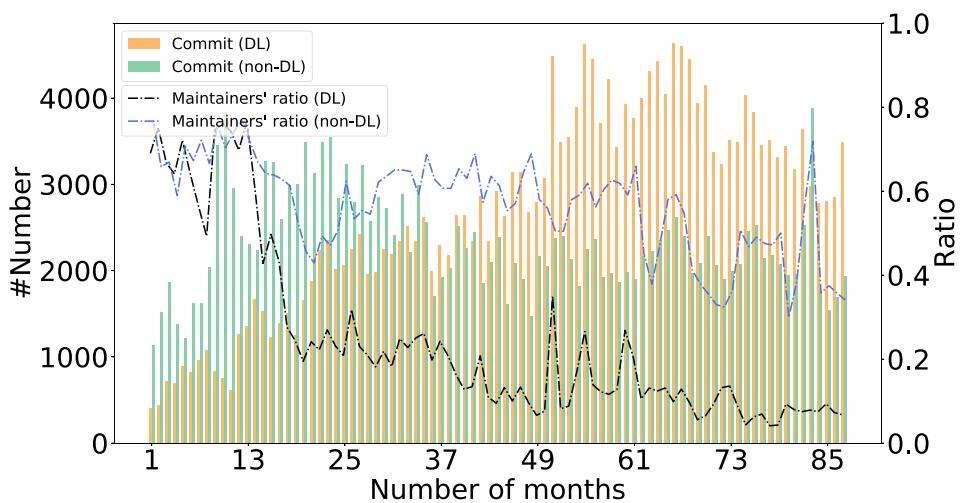


FIGURE 2 Number of commits in deep learning (DL) and traditional open-source software (OSS) projects, along with the proportion of workload (written as “ratio” on the right side) that is conducted by maintainers.

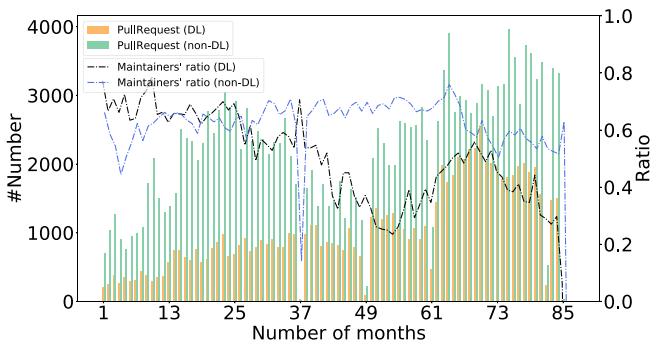


FIGURE 3 Number of pull requests in deep learning (DL) and traditional open-source software (OSS) projects, along with the proportion of workload (written as “ratio” on the right side) that is conducted by maintainers.

projects. As illustrated in Figure 3, the number of pull requests in DL projects increases almost consistently, while the proportion of pull requests maintained by maintainers 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 maintained by maintainers remains relatively stable and high, with an average proportion 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 proportion of workload concerning pull requests than those in traditional OSS projects.

Figure 4 depicts the number of issues in DL and traditional OSS projects per month, as well as the proportion of workload that is conducted 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 proportion of workload of issue activity for maintainers in DL and traditional OSS projects. Figure 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 with 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 proportion 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 proportion of workload for maintainers in DL and traditional OSS projects. Results in Figure 5 show that maintainers in DL projects have a lower proportion 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 maintained by maintainers conducted in traditional OSS projects fluctuates over time but remains relatively stable, while it shows a decreasing trend for maintainers in DL projects.

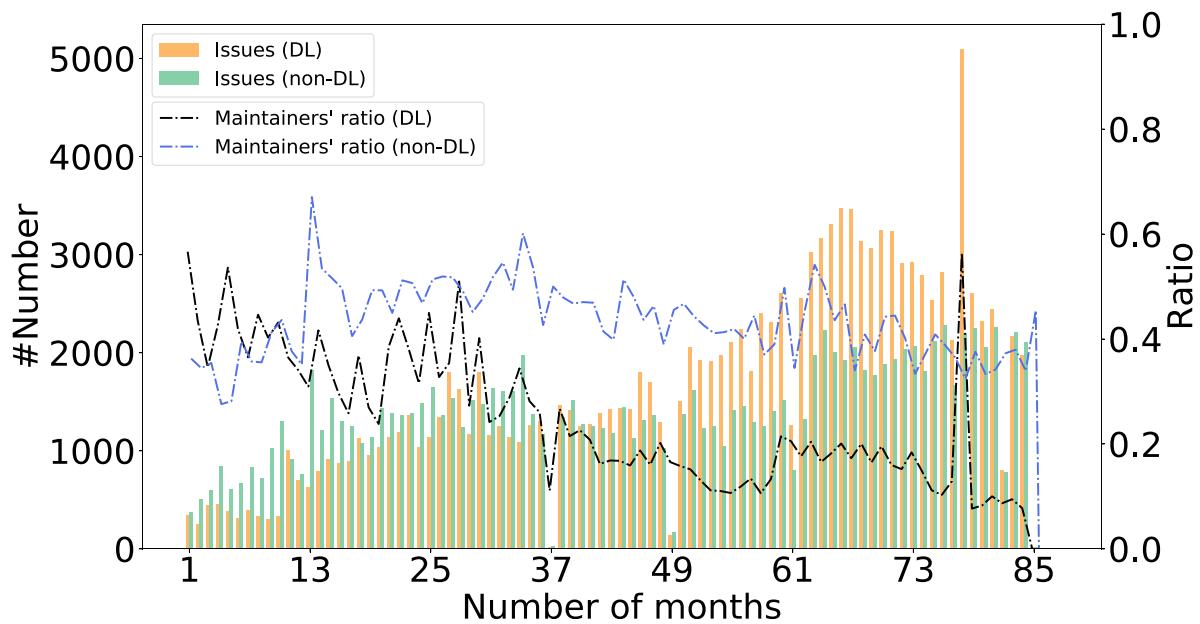


FIGURE 4 Number of issues in deep learning (DL) and traditional open-source software (OSS) projects, along with the proportion of workload (written as “ratio” on the right side) that is conducted by maintainers.

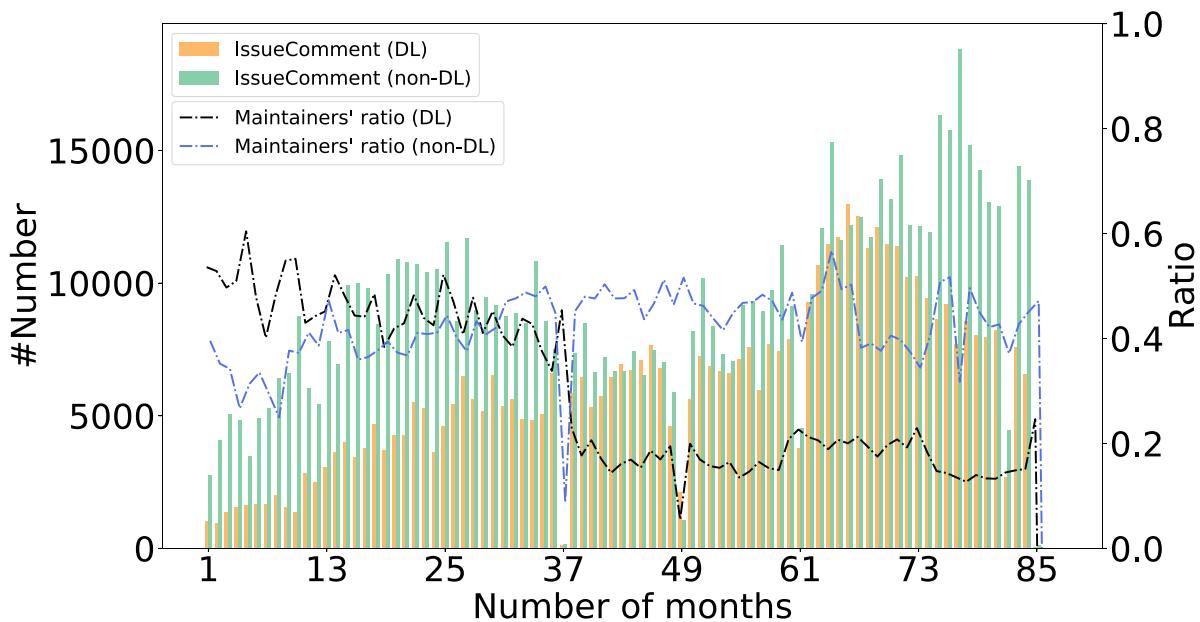


FIGURE 5 Number of issue comments in deep learning (DL) and traditional open-source software (OSS) projects, along with the proportion of workload (written as “ratio” on the right side) that is conducted by maintainers.

Figure 6 displays the number of releases in DL and traditional OSS projects per month, as well as the proportion of workload that is conducted 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 2 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 proportion of workload for maintainers between DL and traditional OSS projects.

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 Figures 7 to 9. Our findings in Figure 7 indicate that traditional OSS projects have a higher number of maintainers than DL projects during the

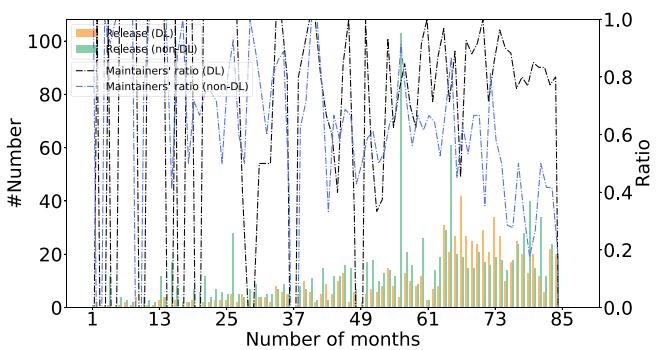


FIGURE 6 Number of releases in deep learning (DL) and traditional open-source software (OSS) projects, along with the proportion of workload (written as “ratio” on the right side) that is conducted by maintainers.

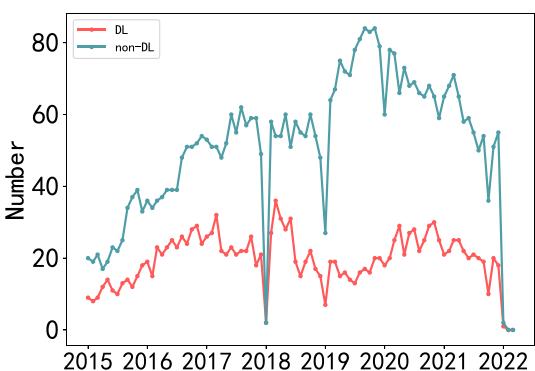


FIGURE 7 Number of maintainers in deep learning (DL) and traditional open-source software (OSS) projects.

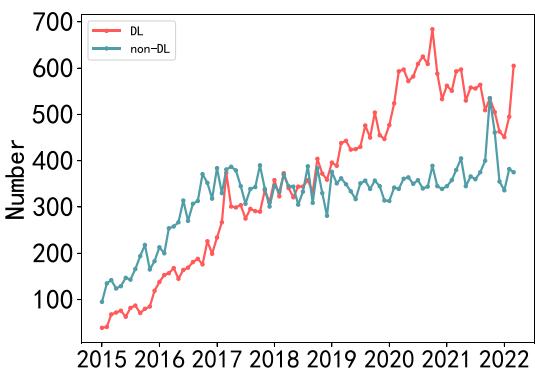


FIGURE 8 Number of authors in deep learning (DL) and traditional open-source software (OSS) projects.

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 Figure 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.⁴²

Regarding the number of newcomers, as illustrated in Figure 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 proportion of workload for maintainers across almost all activities

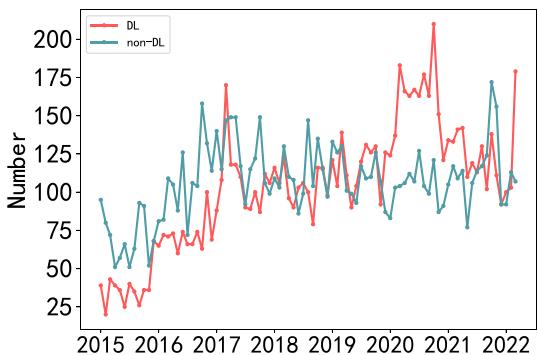


FIGURE 9 Number of newcomers in deep learning (DL) and traditional open-source software (OSS) projects.

between DL and traditional OSS projects. Specifically, the proportion of workload for maintainers in DL projects is 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.

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

3.1.1 | 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 the proportion of workload that maintainers conducted in DL and traditional OSS projects. Especially, as maintainers in DL projects tend to have a lower workload compared with those in traditional OSS projects, therefore, organizations should consider this point when planning and allocating resources for DL projects. Besides, developers should consider this point when choosing OSS projects in different domains to contribute to and consider how to collaborate with maintainers effectively.

Maintainers: Maintainers in both DL and traditional OSS projects should be aware of this phenomenon, closely monitor their proportion of workload 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 to 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 proportion of workload for 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.

3.2 | RQ3: How does the average workload of maintainers grow as DL and traditional OSS projects evolve? And if there exist differences between them?

Motivation: Findings in RQ2 reveal a decreasing trend in the proportion of workload for 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 perform. 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 conducted has increased with the development of DL and traditional OSS projects, we pictured the monthly changes in the average workload per maintainer conducted 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 ANOVA⁴³ 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 Figures 11 and 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.⁴⁴ 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 Figures 11 and 12. Results in Figure 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,⁴² where the project was

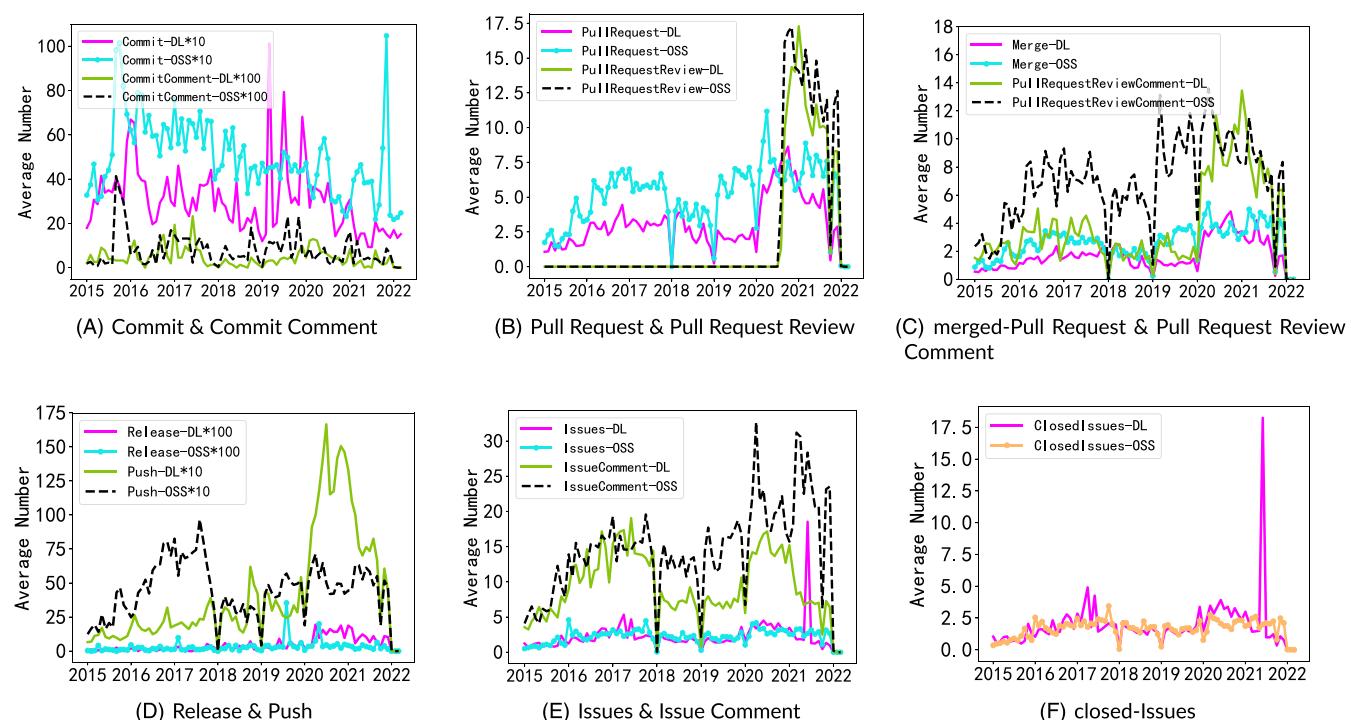


FIGURE 10 Comparison of the trends of an average number of activities conducted by per maintainer in deep learning (DL) and open-source software (OSS) projects. Notably, different scales are used to emphasize trend similarities among various activities. For instance, the commit number is multiplied by 10, and the number of commit comments is multiplied by 100

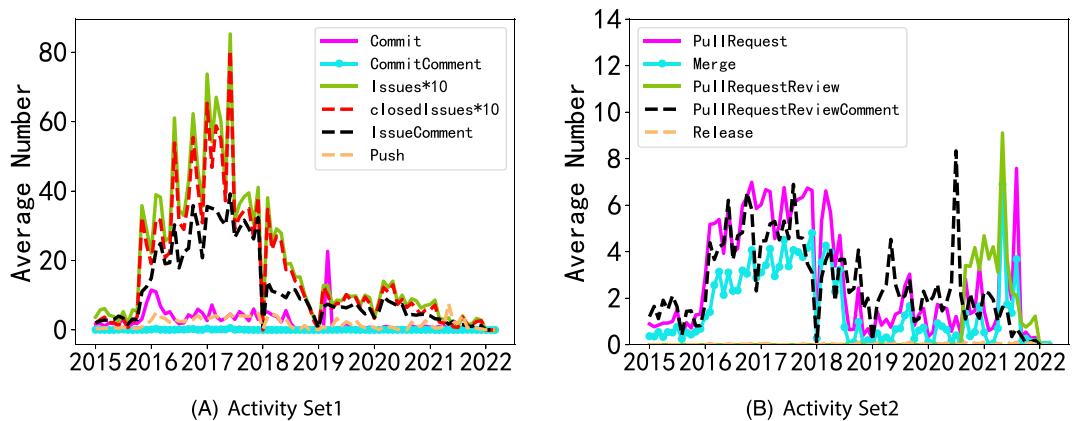


FIGURE 11 Trends of maintainers' average workload on various activities in Tensorflow.

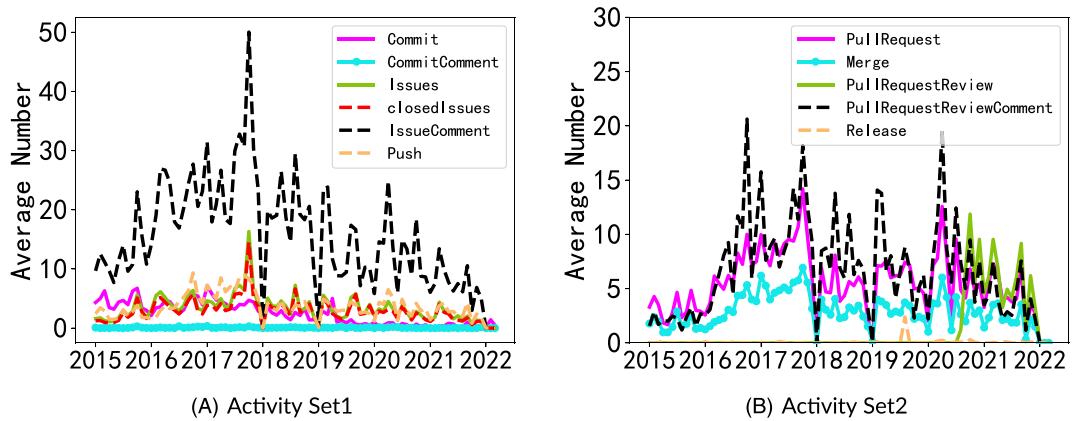


FIGURE 12 Trends of maintainers' average workload on various activities in react.

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, Figure 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 fluctuate dramatically during this period. Nevertheless, it still presents a decreasing pattern in the long run.

Table 2 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 Figure 11. Meanwhile, Table 3 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.

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.

3.2.1 | 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 Figure 10, maintainers in DL and

TABLE 2 Statistical tests for growth rates across various activities of maintainers in each sampled DL project.

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 (***)

Note: The second and third columns show *F*-values and *p*-values for average growth rates across various activities.

Abbreviation: DL, deep learning.

TABLE 3 Statistical tests for growth rates across various activities of maintainers in each sampled traditional OSS project.

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 (**)
finagle	4.67	<0.01 (**)
zulip	1.279	>0.05

Note: The second and third columns show *F*-values and *p*-values for average growth rates across various activities.

Abbreviation: OSS, open-source software.

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, for example, adopting GitHub's new detailed code review tools** to tackle pull request review and review comments.

Although the monthly changes in the average workload 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 with 58% in traditional OSS projects (see Table 2 and 3). 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 Figure 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,^{45–47} derive automatic code review tools on top of existing studies,⁴⁸ 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.¹⁹

3.3 | RQ4: What are the relationships between maintainers' activities and sustainability of DL and traditional OSS 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 analyze 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^{49,50} 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, that is, 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 maintainers, 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. Figure 13 illustrates the monthly changes in the number of commits for DL and traditional OSS projects, and Figure 14A,B shows the relationships between the number of commits and maintainers' activities in DL and traditional OSS projects, respectively.

Figure 14A 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 Figure 14B. 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.

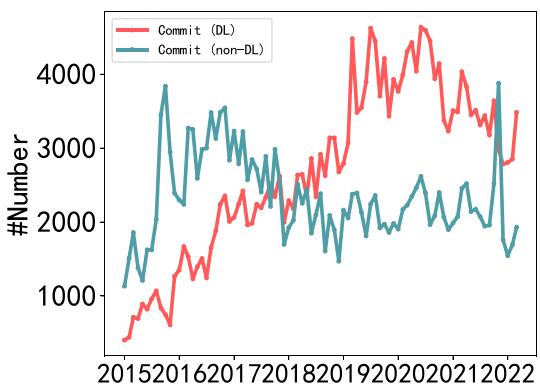


FIGURE 13 Sustainability of deep learning (DL) and traditional open-source software (OSS) projects, using the metric of the number of commits.

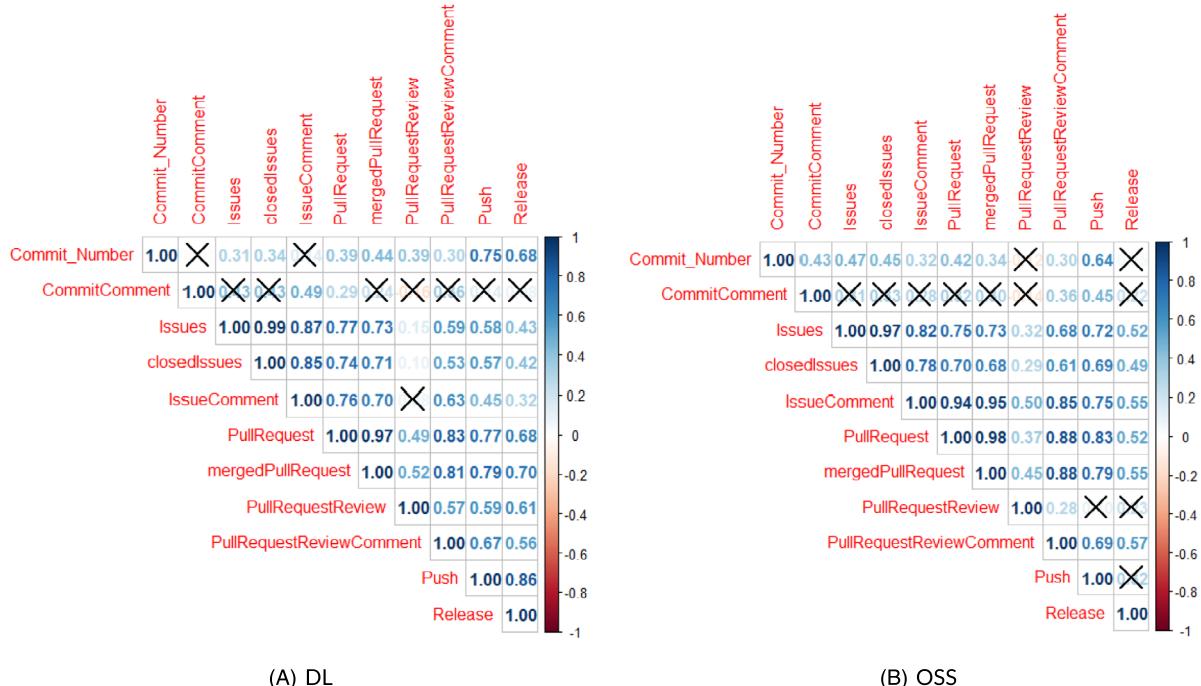


FIGURE 14 Relationships between the sustainability of projects (i.e., the number of commits) and maintainers' activities in deep learning (DL) and open-source software (OSS) projects. The blue color indicates a positive correlation, and the red color indicates a negative correlation. The darker the color is, the stronger the correlation is. “X” implies that the correlation is insignificant.⁵¹ 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.⁵²

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.

3.3.1 | Implications

Organizations and developers: Organizations and developers should understand the positive correlations between specific activities of maintainers and project sustainability. They can provide their support to these activities to help improve the sustainability of projects to a certain degree.

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.

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, previous studies⁴⁴ 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 that 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.

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 DL 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.^{23,53-57} Closely related to our work, Wang et al²³ 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, Kalliamvakou et al⁵⁸ conducted an empirical study of software engineering management at Microsoft. Specifically, they aimed to understand, for developers and engineering managers, what manager attributes are important and why. Dias et al³⁶ investigated to unveil the unique attributes that great OSS maintainers might have. Eghbal⁵⁹ analyzed the OSS maintenance process and found that some hidden costs of maintaining OSS projects exist. Zhou et al¹⁸ 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 OSS evolves continuously, it becomes increasingly large and complex.⁶⁰ 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 al⁶⁰ initially elaborated on the laws of software evolution, whereas Scacchi et al⁶¹ conducted a study to examine whether and how the evolution of OSS conforms to the laws of traditional software evolution. Mockus et al⁵⁴ 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. Furthermore, Valiev et al¹⁶ conducted a study on the sustained activity of projects within the PyPI ecosystem and revealed that the relative position of a project in the ecosystem significantly influences its sustainability. Yin et al²⁴ incorporated social-technical networks to predict the real-time sustainability of Apache incubator projects and employed the LIME model to identify the key factors that influence the sustainability.

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 projects and compares with traditional OSS projects, which has not been investigated in previous studies.

4.3 | The analyses of DL systems

DL 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 DL projects.^{14,20,21,62–64} For instance, Thung et al.⁶² conducted an empirical study on the bugs in machine learning systems, to find a sample set of bugs and corresponding fixes. Zhang et al.¹⁴ collected program bugs existed in DL projects that depend on TensorFlow and endeavored to determine the root causes and symptoms of these bugs. Islam et al.^{20,21} collected data from Stack Overflow and GitHub with the aim to understand the bugs types, root causes, bug impacts on outcomes, bug-prone stages as well as bug fixing patterns of DL projects. Simultaneously, Zhang et al.⁶³ presented an empirical study on program failures of DL jobs in Microsoft. Liu et al.⁶⁴ analyzed the prevalence, types, and distribution of self-admitted technical debt in seven popular DL frameworks.

Simultaneously, there also exist some empirical studies exploring the migration process of DL libraries. Han et al.⁶⁵ put their effort into dependency networks of DL libraries. They studied the dependency degrees that projects depend on DL libraries, the update behaviors, and reasons when updating DL libraries, and the version distributions of DL projects. Dilhara et al.⁴⁴ examined to mine how developers in Software-2.0 use DL libraries and whether or not the DL library evolution affects their code. Moreover, they also expounded on the challenges of DL library evolution by performing a survey on developers involving DL libraries.

Most prior studies focused on the program bugs/failures/faulds of DL projects or the migration process of DL libraries. However, these prior studies have not explored maintainers' activities, the growth of maintainers' activities, not to mention the growth 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 study⁶⁶ and GitHub's annual report.⁶⁷ 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. Another 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 first external validity relates to the small dataset of 19 DL projects and 19 OSS projects, which limits the generalizability of our findings. To alleviate this threat, we referred to extant literature and found that Wang et al.'s²³ study only collected 20 OSS projects to unveil elite developers' activities, and Joblin et al.³⁵ obtained 32 OSS projects to understand how socio-technical factors affect the success of projects. Moreover, Yue et al.'s²⁵ study selected 54 OSS projects to investigate the correlations between newcomers' dynamic contribution patterns with their technical success in their early careers. Our constructed dataset includes 38 OSS projects, which is quite similar in scale with prior studies. This helps to alleviate this threat to a certain degree. The second external validity involves the limitation of DL projects. As we gather DL projects to study their growth 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. This helps to mitigate this threat.

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 growth 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 growth 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 growth of most activities between DL and traditional OSS projects, and maintainers in DL projects conduct 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 small and medium sizes to expand the generalization of our results. Moreover, we also encourage further studies to extend our work, for example, 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, and so on. To facilitate replications or other types of future work, we make the data and scripts used in this study publicly available online (<https://github.com/HJXPaperData/SustainabilityofDL>).

ACKNOWLEDGMENTS

This work was supported by the Scientific research project of Zhejiang Provincial Education Department (Y202351453), and was supported in part by the National Science Foundation of China under grant U20A20173. Besides, we also acknowledged the support of the advanced computing resources provided by the Super computing Center of Hangzhou City University.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in SustainabilityofDL (<https://github.com/HJXPaperData/SustainabilityofDL>).

ORCID

Junxiao Han  <https://orcid.org/0000-0001-7630-8667>

ENDNOTES

- † <https://github.com/tensorflow/tensorflow>
- ‡ <https://github.com/pytorch/pytorch>
- § <https://github.com/Theano/Theano>
- ¶ <https://github.com/BVLC/caffe>
- # <https://github.com/keras-team/keras>
- || <https://docs.github.com/en/developers/webhooks-and-events/events/github-event-types>
- * <https://github.com/fastai/fastai/issues/575>
- ** <https://github.com/features/code-review/>

REFERENCES

1. Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition IEEE; 2012:3642-3649.
2. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016:770-778.
3. Acuna D, Kar A, Fidler S. Devil is in the edges: learning semantic boundaries from noisy annotations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019:11075-11083.
4. Hinton G, Deng L, Yu D, et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Sig Process Mag*. 2012;29(6):82-97.
5. Nassif AB, Shahin I, Attili I, Azzeh M, Shaalan K. Speech recognition using deep neural networks: a systematic review. *IEEE Access*. 2019;7:19143-19165.
6. Chen C, Seff A, Kornhauser A, Xiao J. Deepdriving: learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision; 2015:2722-2730.
7. Huval B, Wang T, Tandon S, et al. An empirical evaluation of deep learning on highway driving. arXiv preprint arXiv:150401716; 2015.
8. Tian Y, Pei K, Jana S, Ray B. Deeptest: automated testing of deep-neural-network-driven autonomous cars. In: Proceedings of the 40th International Conference on Software Engineering; 2018:303-314.
9. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T. The rise of deep learning in drug discovery. *Drug Discov Today*. 2018;23(6):1241-1250.
10. Oktay O, Schlemper J, Folgoc LL, et al. Attention u-net: learning where to look for the pancreas. arXiv preprint arXiv:180403999; 2018.
11. Petersen J, Jäger PF, Isensee F, et al. Deep probabilistic modeling of glioma growth. In: International Conference on Medical Image Computing and Computer-Assisted Intervention Springer; 2019:806-814.
12. Roy A, Sun J, Mahoney R, Alonzi L, Adams S, Beling P. Deep learning detecting fraud in credit card transactions. In: 2018 Systems and Information Engineering Design Symposium (SIEDS) IEEE; 2018:129-134.
13. Wan Z, Xia X, Lo D, Murphy GC. How does machine learning change software development practices? *IEEE Trans Softw Eng*. 2019;47(9):1857-1871.
14. Zhang Y, Chen Y, Cheung S-C, Xiong Y, Zhang L. An empirical study on tensorflow program bugs. In: Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis; 2018:129-140.
15. Penzenstadler B, Bauer V, Calero C, Franch X. Sustainability in software engineering: a systematic literature review. In: 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012). IET; 2012; Ciudad Real, Spain:32-41. <https://digital-library.theiet.org/content/conferences/10.1049/ic.2012.0004>

16. Valiev M, Vasilescu B, Herbsleb J. 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; 2018:644-655.
17. Zhang Y, Stol K-J, Liu H, Zhou M. 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; 2022:1048-1060.
18. Zhou M, Chen Q, Mockus A, Wu F. On the scalability of linux kernel maintainers' work. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering; 2017:27-37.
19. Tan X, Zhou M, Fitzgerald B. Scaling open source communities: an empirical study of the linux kernel. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE) IEEE; 2020:1222-1234.
20. Islam MJ, Nguyen G, Pan R, Rajan H. 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; 2019:510-520.
21. Islam MJ, Pan R, Nguyen G, Rajan H. Repairing deep neural networks: fix patterns and challenges. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE) IEEE; 2020:1135-1146.
22. Chen J, Liang Y, Shen Q, Jiang J, Li S. Toward understanding deep learning framework bugs. ACM Transactions on Software Engineering and Methodology; 2022.
23. Wang Z, Feng Y, Wang Y, Jones JA, Redmiles D. Unveiling elite developers' activities in open source projects. *ACM Trans Softw Eng Methodol (TOSEM)*. 2020;29(3):1-35.
24. Yin L, Chen Z, Xuan Q, Filkov V. Sustainability forecasting for apache incubator projects. arXiv preprint arXiv:210514252; 2021.
25. Yue Y, Wang Y, Redmiles D. Off to a good start: dynamic contribution patterns and technical success in an OSS newcomer's early career. *IEEE Trans Softw Eng*. 2022;49(2):529-548.
26. GH archive. <https://www.gcharchive.org/>; 2021.
27. Google bigquery. <https://cloud.google.com/bigquery/docs>; 2021.
28. Coelho J, Valente MT. Why modern open source projects fail. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering; 2017:186-196.
29. Mendez C, Padala HS, Steine-Hanson Z, Hilderbrand C, Horvath A, Hill C, Simpson L, Patil N, Sarma A, Burnett M. Open source barriers to entry, revisited: a sociotechnical perspective. In: Proceedings of the 40th International Conference on Software Engineering; 2018:1004-1015.
30. Qiu HS, Nolte A, Brown A, Serebrenik A, Vasilescu B. 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; 2019:688-699.
31. Manotas I, Bird C, Zhang R, Shepherd D, Jaspan C, Sadowski C, Pollock L, Clause J. An empirical study of practitioners' perspectives on green software engineering. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE) IEEE; 2016:237-248.
32. Noman H, Mahoto NA, Bhatti S, Abosaq HA, Al Reshan MS, Shaikh A. An exploratory study of software sustainability at early stages of software development. *Sustainability*. 2022;14(14):8596.
33. Trinkenreich B, Guizani M, Wiese I, Conte T, Gerosa M, Sarma A, Steinmacher I. Pots of gold at the end of the rainbow: what is success for open source contributors? *IEEE Trans Softw Eng*. 2021;48(10):3940-3953.
34. Crowston K, Howison J, Annabi H. Information systems success in free and open source software development: theory and measures. *Softw Process: Improv Pract*. 2006;11(2):123-148.
35. Joblin M, Apel S. How do successful and failed projects differ? A socio-technical analysis. *ACM Trans Softw Eng Methodol (TOSEM)*. 2022;31(4):1-24.
36. Dias E, Meirelles P, Castor F, Steinmacher I, Wiese I, Pinto G. What makes a great maintainer of open source projects? In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) IEEE; 2021:982-994.
37. Tan X, Zhou M, Sun Z. 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; 2020:398-409.
38. Han J, Deng S, Lo D, Zhi C, Yin J, Xia X. 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; 2021:298-307.
39. LaToza TD, Venolia G, DeLine R. Maintaining mental models: a study of developer work habits. In: Proceedings of the 28th International Conference on Software Engineering; 2006:492-501.
40. Cohen J. A coefficient of agreement for nominal scales. *Educ Psychol Meas*. 1960;20(1):37-46.
41. Massey Jr FJ. The Kolmogorov-Smirnov test for goodness of fit. *J Am Stat Assoc*. 1951;46(253):68-78.
42. Han J, Shihab E, Wan Z, Deng S, Xia X. What do programmers discuss about deep learning frameworks. *Empir Softw Eng*. 2020;25(4):2694-2747.
43. Heiberger RM, Neuirth E. One-way ANOVA. *R Through Excel*: Springer; 2009:165-191.
44. Dilhara M, Ketkar A, Dig D. Understanding software-2.0: a study of machine learning library usage and evolution. *ACM Trans Softw Eng Methodol (TOSEM)*. 2021;30(4):1-42.
45. Hu X, Men R, Li G, Jin Z. Deep-autocoder: learning to complete code precisely with induced code tokens. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Vol. 1 IEEE; 2019:159-168.
46. Bernaschina C, Falzone E, Fraternali P, Gonzalez SLH. The virtual developer: integrating code generation and manual development with conflict resolution. *ACM Trans Softw Eng Methodol (TOSEM)*. 2019;28(4):1-38.
47. Liu F, Li G, Zhao Y, Jin Z. Multi-task learning based pre-trained language model for code completion. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering; 2020:473-485.
48. Chen Q, Zhou M. A neural framework for retrieval and summarization of source code. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE) IEEE; 2018:826-831.
49. Sedgwick P. Spearman's rank correlation coefficient. *Bmj*. 2014;349.
50. Borges H, Hora A, Valente MT. Understanding the factors that impact the popularity of GitHub repositories. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME) IEEE; 2016:334-344.
51. Dong Y, Zhang P, Wang J, Liu S, Sun J, Hao J, Wang X, Wang L, Dong J, Dai T. 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; 2020:73-82.
52. Guilford JP. Fundamental statistics in psychology and education; 1950.

53. Sonnentag S. Excellent software professionals: experience, work activities, and perception by peers. *Behav Inform hnl*. 1995;14(5):289-299.
54. Mockus A, Fielding RT, Herbsleb J. A case study of open source software development: the apache server. In: Proceedings of the 22nd International Conference on Software Engineering; 2000:263-272.
55. Zhou M, Mockus A. Developer fluency: achieving true mastery in software projects. In: Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering; 2010:137-146.
56. Zhou M, Mockus A. What make long term contributors: willingness and opportunity in OSS community. In: 2012 34th International Conference on Software Engineering (ICSE) IEEE; 2012:518-528.
57. Bao L, Xia X, Lo D, Murphy GC. A large scale study of long-time contributor prediction for GitHub projects. *IEEE Transactions on Software Engineering*; 2019.
58. Kalliamvakou E, Bird C, Zimmermann T, Begel A, DeLine R, German DM. What makes a great manager of software engineers? *IEEE Trans Softw Eng*. 2017;45(1):87-106.
59. Eghbal N. *Working in Public: The Making and Maintenance of Open Source Software*: Stripe Press; 2020.
60. Lehman MM, Ramil JF, Wernick PD, Perry DE, Turski WM. Metrics and laws of software evolution-the nineties view. In: Proceedings Fourth International Software Metrics Symposium IEEE; 1997:20-32.
61. Scacchi W. Understanding open source software evolution. Applying, breaking and rethinking the laws of software evolution. Citeseer; 2003.
62. Thung F, Wang S, Lo D, Jiang L. An empirical study of bugs in machine learning systems. In: 2012 IEEE 23rd International Symposium on Software Reliability Engineering IEEE; 2012:271-280.
63. Zhang R, Xiao W, Zhang H, Liu Y, Lin H, Yang M. An empirical study on program failures of deep learning jobs. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE) IEEE; 2020:1159-1170.
64. Liu J, Huang Q, Xia X, Shihab E, Lo D, Li S. Is using deep learning frameworks free? Characterizing technical debt in deep learning frameworks. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society; 2020:1-10.
65. Han J, Deng S, Lo D, Zhi C, Yin J, Xia X. An empirical study of the dependency networks of deep learning libraries. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME) IEEE; 2020:868-878.
66. Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D. The promises and perils of mining GitHub. In: Proceedings of the 11th Working Conference on Mining software Repositories; 2014:92-101.
67. GitHub annual report. <https://octoverse.github.com/>; 2020.

How to cite this article: Han J, Liu J, Lo D, Zhi C, Chen Y, Deng S. On the sustainability of deep learning projects: Maintainers' perspective. *J Softw Evol Proc*. 2023;e2645. <https://doi.org/10.1002/smr.2645>