2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)

Sustainability Forecasting for Deep Learning Packages

Junxiao Han*, Yunkun Wang†, Zhongxin Liu‡, Lingfeng Bao‡, Jiakun Liu§, David Lo§, Shuiguang Deng†
*School of Computer & Computing Science, Hangzhou City University, Hangzhou, China
†College of Computer Science and Technology, Zhejiang University, Hangzhou, China
‡The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China
§School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
hanjx@hzcu.edu.cn, {wangykun, liu_zx, lingfengbao}@zju.edu.cn, {jkliu, davidlo}@smu.edu.sg, dengsg@zju.edu.cn

Abstract*—Deep Learning (DL) technologies have been widely adopted to tackle various tasks. In this process, through software dependencies, a multi-layer DL supply chain (SC) is formed, with DL frameworks acting as the root, DL packages acting as the bridge nodes, and downstream DL projects acting as the periphery. However, most Open Source Software (OSS) projects may fail. Considering the crucial position of DL packages in the DL SC, to foster the sustainable development of DL SCs and DL packages, we aim to forecast the long-term sustainability of DL packages. Here, sustained activity is adopted as the main proxy of sustainability, and the sustainability status is classified as "sustainable" or "dormant". Relatedly, a DL package is considered as "sustainable" if it has sustained activity in its last 12 months. Otherwise, it is deemed as "dormant". To this end, we propose an approach that begins with obtaining longitudinal features for each DL package in each month. Then, we develop a model to forecast the sustainability of DL packages by incorporating the longitudinal features, which can aptly predict sustainability with an accuracy of up to 0.81. Subsequently, an interpretable module is developed to interpret the determinants (i.e., important features) that impact the sustainability of DL packages. Finally, we generate sustainability trajectories for each DL package to better understand the monthly changes of their sustainability status. Our findings uncover that for most DL packages, fewer but more centralized developers and a balanced collaboration are more likely to help sustain the DL packages. Furthermore, although some DL packages are sustainable, their sustainability trajectories present statistically decreasing trends over time. Based on the findings, we shed light on the dynamic sustainability of DL packages, highlight future research directions, and provide practical suggestions to DL package maintainers, developers, users, and software engineering researchers.

***Index Terms*—Deep Learning Packages, Sustainability, Prediction Model**

I. Introduction

Deep learning (DL) has achieved tremendous success in many cutting-edge domains in the past decade. In this regard, a wide variety of DL frameworks, packages, and projects emerged to deal with various tasks [27], such as image recognition [16], [67], natural language processing [24], autonomous driving [33], [56], crime prediction [68], [31], and source code mining [28], [14], [13].

The explosion of DL technologies is inseparable from DL frameworks. Simultaneously, due to the diverse needs when using DL frameworks, numerous DL enthusiasts have produced

Junxiao Han and Jiakun Liu are both corresponding authors.

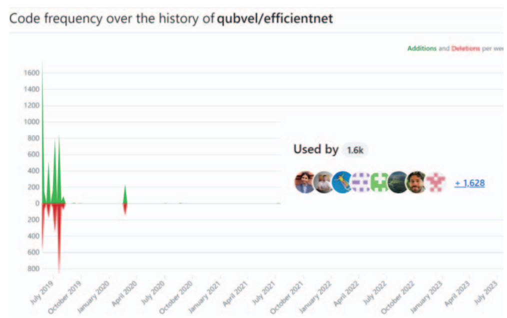


Fig. 1. An example of `efficientnet` [3] in GitHub, a DL package with many dependent packages and projects that has no longer been developed or maintained for a prolonged time.

comprehensive packages that offer specific functionalities based on these DL frameworks, either directly or transitively. These DL packages can further serve as crucial nodes, acting as bridges to attract downstream projects. Gradually, a DL supply chain (SC) is formed, starting from a DL framework, branching out into DL packages that serve as bridge nodes, and ultimately expanding to a plethora of downstream projects as the periphery [54].

Rich anecdotal evidence indicates a widespread adoption and usage of DL packages [19]. However, over 80% of Open Source Software (OSS) projects tend to be abandoned over time [50]. Due to the wide adoption and critical position of DL packages in DL SCs, once they are abandoned without development and maintenance activities, it will pose a huge threat to the sustainable development of downstream DL packages and projects that depend on these DL packages. This issue is exemplified by the case of `efficientnet` [3] in Figure 1, a DL package with relatively high popularity (i.e., more than 2,000 stars). 21 packages and 1,639 repositories depend on it. It has no longer been developed or maintained for a prolonged time, resulting in many open issues waiting for resolution, which could hinder the healthy and sustainable development of downstream DL packages and projects.

Furthermore, an open issue from project *keras-applications* on 06 July 2021 asks, 'Will keras applications keep updating?' They state, 'It is truly useful, but it was last updated a year ago.' This issue further highlights the importance of understanding and predicting the sustainability of DL pack-

2640-7574/24/$31.00 ©2024 IEEE
DOI 10.1109/SANER60148.2024.00106

ages. If there is a tool that can predict the sustainability of DL packages, participants can check the sustainability status at any point, and maintainers can intervene and adjust the sustainability when needed. Therefore, this motivates us to understand and predict the sustainability of DL packages early on. In this regard, since the definition of sustainability is multifaceted [59], [62], we thus adopted sustained activity as the proxy of project sustainability, following previous work [12], [59], [62] (detailed in Section II).

Prior research studied the dependency and evolution of DL libraries and projects [29], [19], [57], [54], [20]. These studies have primarily focused on investigating the update behaviors, library usage, and evolution of DL libraries [29], [19], [57], as well as the structure and application domains of packages and projects in DL SCs [54], [20]. However, no prior work has investigated to forecast the sustainability of DL packages.

To address this gap and foster the sustainable development of DL SCs and DL packages, in this paper, we start by constructing a DL package SC, which consists of 262 DL packages. Subsequently, we construct technical (code-related) networks for each DL package in each month. This effort is to investigate the potential for effectively predicting the sustainability of DL packages by leveraging temporal traces. This forms the first research question:

RQ1: *How effectively can we predict the sustainability of DL packages based on temporal traces?*

To achieve this, we construct temporal technical networks for each DL package and train LSTM models to perform the prediction task. Our findings indicate that the best predictive performance was observed at 33 months, with accuracy, precision, recall, and F1-score values of 0.81, 0.79, 0.81, and 0.80, respectively. However, DL models, such as LSTMs, are black-box and lack of interpretability. This makes it challenging for users to identify which features have the most significant impact on sustainability. To better explain it, we further develop an interpretable model. Hence, we have the next research question:

RQ2: *What are the determinants of sustainability?*

Subsequently, we implement a global interpretable model to interpret the predicted outcome of the trained LSTM model, and identify the determinants (i.e., features of vital importance) that impact sustainability. Consequently, we find that for most DL packages, fewer but more centralized developers and a balanced collaboration of contributions are more likely to foster sustainable development. Instead, simply a large number of developers and commits are likely to be detrimental to sustainability. Furthermore, with our interpretable results, can we better understand why sustainability changes over time for DL packages? To gain a more profound understanding of this question, we come to the final research question:

RQ3: *What are the trajectories of sustainability for DL packages?*

In the final research question, we derive trajectories of sustainability changes for each DL package. We further perform in-depth case studies to explain critical turning points with the aid of our interpretable results. Results indicate that although some DL packages are sustainable, their sustainability trajectories exhibit statistically decreasing trends over time. This warns that maintainers and developers of these DL packages should pay more attention and perform timely interventions to reverse this status.

This paper makes the following contributions:

- We present a novel longitudinal dataset comprising development traces of hundreds of DL packages, accompanied by the labeled sustainability status. This dataset is publicly available at https://github.com/greenlight2000/DL_Package_Sustainability.
- By incorporating technical network and project features, we propose a model that can aptly predict the future sustained activity of DL packages.
- We interpret the prediction output of the model, and mine the determinants impacting sustainability for DL packages.
- We depict the trajectories of sustainability changes for each DL package, and conduct in-depth case studies to investigate the events that happened during critical turning points.

The remainder of this paper is organized as follows. Section II introduces the background and related work of our study. Section III introduces the approach conducted in this paper, and Section IV presents the dataset. Section V reports the findings for each of the four research questions, and Section VI discusses the implications. Section VII discusses threats to the validity of our study, and Section VIII concludes this paper and presents future work.

II. Background and Related Work

A. Definition of DL Package SC

As defined in Tan et al.'s study [54], the DL SC is defined as a directed graph $G = \langle V_{from}, V_{to}, E \rangle$, with the root project being a DL framework such as TensorFlow or PyTorch. Here, V_{from} represents a set of downstream projects, V_{to} is a set of upstream projects imported by focal projects (i.e., the current project), and $E \subseteq V_{from} \times V_{to}$ is a set of directed edges reflecting import relationships. In their study, they constructed two DL SCs for TensorFlow and PyTorch, separately, and studied the structure of the two DL SCs. Moreover, they also studied the domain distribution of packages and projects in the two DL SCs. Gao et al. [20] constructed two hierarchical DL package SCs for TensorFlow and PyTorch, separately, and explored the domains, clusters, and disengagement reasons of DL packages in their constructed DL package SCs. However, they only uncovered the disengagement reasons for some DL packages. They did not explore the dynamic sustainability of DL packages, not to mention the determinants of sustainability, as well as the trajectories of sustainability changes for DL packages, which are studied in our paper.

B. Project Success and Sustainability

Although there is no unified definition of success [22], a considerable body of literature has focused on the investigation of project success. Ghapanchi et al. [23] conducted a literature survey to explore various areas of OSS success and provided a measurement taxonomy consisting of six success areas for

OSS projects. Midha et al. [45] proposed two measures of project success, including market success (defined as the level of interest shown by project consumers) and technical success (defined as the effort expended by project developers), and investigated which factors can affect the success of projects over time. Joblin et al. [38] selected 32 OSS projects to investigate how socio-technical factors are associated with project success, and identified key features that have a significant impact on project success. Piggott et al. [49] applied machine learning techniques to develop a model capable of accurately determining the developmental stage of a software project. Coelho et al. [18] investigated the reasons why OSS projects fail, and uncovered the discriminative maintenance practices between failed and successful projects. Avelino et al. [12] applied a mixed-method approach to understand the abandonment of a project by its core developers, and the survival of popular OSS projects.

Although project success is similar to sustainability, they still have discrepancies. Sustainability somewhat reflects the dynamic status of OSS projects, while success represents a static status [63]. Nevertheless, there are only a few studies that have investigated project sustainability. Valiev et al. [59] studied the sustained activity of projects in the PyPI ecosystem, and found that the relative position in the ecosystem plays a vital role in the sustainability of projects. Xiao et al. [62] investigated the relationship between early participation factors and long-term project sustainability, and found that the steady attention and commitment of core developers positively affect future sustained activity. Yin et al. [63] incorporated social-technical networks to forecast the real-time sustainability of Apache Software Foundation Incubator (ASFI) projects, and applied the LIME model to identify the determinants of sustainability. Their study is most relevant to ours. However, their study is limited to ASFI projects. Additionally, they adopted social-technical networks to form predictive features, and emails were adopted as their social traces. However, emails cannot be obtained for DL packages in GitHub. Since technical (code-based) activities are major activities in OSS projects [64], and can be easily obtained; we thus constructed technical networks with project features to predict the long-term sustainability of DL packages. Besides, they only generated and analyzed trajectories of sustainability changes for three projects, while we generated trajectories for all DL packages in our dataset.

Furthermore, several studies explored the health of OSS projects and ecosystems. For instance, Xia et al. [61] applied recent data to predict multiple health indicators of open-source projects. Liao et al. [41] proposed health indicators and analyzed the healthy development trend of the GitHub ecosystem. In summary, development activity has always been adopted as a proxy for OSS success, sustainability, and health. However, as Nyman et al. [47] stated, sustainability refers to the capacity of OSS projects to continue serving the needs of their developers and users, whereas success and health are measured at a specific moment. Therefore, in this paper, we adopted sustained activity as the main proxy of project sustainability, following previous work [12], [59], [62].

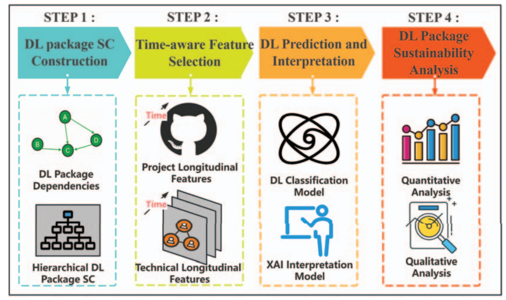


Fig. 2. The overview of our approach.

C. Research on SE for DL

A surge of recent work has focused on SE for DL [21] based on the data collected from GitHub, Stack Overflow, interviews, and surveys. Most of the research in this area revolves around DL bugs. Thung et al. [55] studied a sample set of bugs in machine learning systems, along with their corresponding fixes. Zhang et al. [66] investigated the root causes and symptoms of program bugs that existed in DL projects that depend on TensorFlow. Islam et al. [34], [35], Jia et al. [37] and Chen et al. [15] explored the characteristics, root causes, effects, anti-patterns, and fixing patterns of bugs in representative DL frameworks. Additionally, Humbatova et al. [32] generated a comprehensive taxonomy of real faults in DL systems, and Zhang et al. [65] mined bugs associated with DL jobs. Furthermore, there are also several studies focused on the dependency evolution of DL frameworks. Han et al. [29] explored the application domains, update behaviors, and distribution of dependency versions for DL projects that depend on DL frameworks. Dilhara et al. [19] examined the usage and update of DL frameworks in DL projects, while Tidjon et al. [57] studied the usage of DL library combinations, and the distribution of DL library dependencies across different ML workflows.

III. Approach

In this section, we first introduce the construction of our DL package SC. Then, we introduce the longitudinal features we obtained. After that, we describe how LSTM is trained and employed to predict long-term sustainability based on the longitudinal features. Ultimately, we describe how the SHAP [42] model is performed to explain the correlation between longitudinal features and the prediction outcome (i.e., sustainability status). The overview of our approach is illustrated in Figure 2.

A. DL Package SC Construction

In this paper, we leverage the DL packages published in Tan et al.'s study [54] to construct our DL package SC. Tan et al. [54] have constructed two DL SCs for TensorFlow and PyTorch, separately. However, by manual observation, too many packages belong to both SCs in their study. Therefore, we adopted the SC construction algorithm introduced in their

TABLE I
THE AGGREGATE OF ALL FEATURES BELONGS TO THE TWO GROUPS.

Group	Feature Name	Description
Project Features	`active_devs`	The count of developers who actively engage in making commits or participating in issue discussions on a monthly basis
	`code_activities`	The count of source code commits in the focal package on a monthly basis
	`num_files`	The number of source code files created in each month
	`c_percentage`	The percentage of commit activities performed by the active developers in each month
	`inactive_c`	The sum of the time intervals of the top 3 longest interruptions between successive commits in each month
Technical Features	`c_nodes`	The number of nodes in each technical network in each month
	`c_edges`	The number of edges in each technical network in each month
	`c_mean_degree`	The average degree of nodes in each technical network in each month
	`c_long_tail`	The degree of the *75th* percentile of nodes in each technical network in each month
	`c_c_coef`	The ratio of connected triplets among the total number of triplets in each technical network in each month

study to construct a unified DL package SC to reduce redundancy. Notably, the two constructed DL SCs in Tan et al.'s study [54] only start with one root project of TensorFlow or PyTorch, while our unified DL package SC starts with two root projects of TensorFlow and PyTorch. After that, we find packages that import TensorFlow or PyTorch as the next layer of our DL package SC and point to TensorFlow or PyTorch, respectively. For packages that import both TensorFlow and PyTorch, it points to both TensorFlow and PyTorch. This process is repeated until no new packages exist.

B. Longitudinal Features/Metrics of Interest

In this section, we present the features we have chosen and the sustainability label.

Technical Features: Technical (code-related) activities are major activities in OSS projects [64], Stănciulescu et al. [52] identified that code-related variables may be associated with project sustainability. Therefore, in this paper, we include commits to source code files as technical activities to extract features. Moreover, network view is always applied to study the dynamics of OSS projects [38]; we thus construct a technical network for each package in each month. Specifically, if developer *D1* and *D2* both committed to the same source code file(s) *F* in a month, we derived an undirected edge between them. In this way, we can generate a dynamic technical network for each package in each month.

Subsequently, we extract the technical network-related features, which are always extracted in network analysis [63], [58]. These include the number of nodes `c_nodes` and edges `c_edges` in each technical network. Furthermore, we determine the mean degree `c_mean_degree` by dividing the sum of degrees of all nodes by the total number of nodes. Additionally, we compute the `c_long_tail`, which represents the degree of the *75th* percentile of nodes in the network. The clustering coefficient `c_c_coef` is also calculated, which is determined by dividing the number of connected triplets by the total number of triplets in the monthly technical network.

Project Features: Besides, project features are also widely adopted in prior studies for modeling OSS project sustainability [59], [63], [62]. We thus select the following project features that are suitable for our problem, based on prior studies. The feature `active_devs` denotes the count of developers who actively engage in making commits or participating in issue discussions on a monthly basis, the feature `code_activities` is the count of source code commits in the focal package, the feature `num_files` indicates the number of source code files created in each month. Simultaneously, we also obtained the feature `c_percentage`, which is the percentage of commit activities performed by the active developers, and `inactive_c`, which is determined by dividing the sum of the top 3 longest intervals between successive commits by the interval between the first and last commit in each month.

Consequently, to make the presentation of all features clear, we integrate them into Table I. Notably, the two groups of features are all longitudinal.

Sustainability Label: Following previous studies [59], [62], we consider a DL package as "sustainable" if it has sustained activity in its last 12 months prior to its most recent commit. That is, its average commit per month in the 12 months prior to its most recent commit is greater than one. Otherwise, it can be regarded as "dormant". In this regard, 192 DL packages are labeled as "sustainable", and 70 DL packages are labeled as "dormant" (as detailed in Section IV). We further define the sustainability status as a binary variable, with $0 = dormant$ and $1 = sustainable$.

C. Model Setup

Given the gathered longitudinal features, our next task is to train a model that can effectively predict the sustainability of DL packages. This problem is essentially a binary classification task. In this study, we employ an LSTM-based recurrent neural network [26], where the inputs are the longitudinal features and the outputs are the sustainability status, as illustrated in Figure 3. The rationale for selecting LSTM is that: 1) LSTM is particularly suitable for sequential data, and is less sensitive to gradient disappearance and explosion issues during training on long sequences [63], 2) LSTM outperforms all other baseline machine learning models (e.g., XGBoost, Random Forest, and Logistic Regression) in our dataset in terms of the evaluation metrics (as detailed in Section V-A).

Implementation Details: To obtain sequential data as input for each DL package, we aggregated longitudinal records into monthly data, spanning from the creation date to the point of becoming sustainable or dormant. Notably, to avoid data leakage, the longitudinal features in the 12 months used for labeling are not included in our dataset. Since longitudinal features have different magnitudes and many of them do

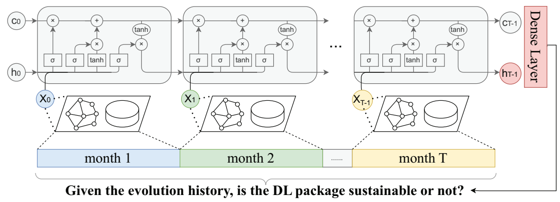


Fig. 3. Demonstration of the prediction process based on LSTM.

not conform to normal distributions, we thus performed the *MinMaxScaler* function to standardize all predictive features. We then implemented a 1-layer LSTM model with 64 neurons, and applied drop-out [51] with the drop rate set to 0.3. Additionally, we utilized a dense layer that employed the softmax function, which yielded the prediction outcome for the classification task (sustainable/dormant). During the training process, we adopted a binary cross-entropy as the loss function, and used *Adam* as the optimizer. For each hyperparameter of timestep, denoted as n, monthly data was truncated to match the current timestep when their duration time (i.e., the lifespan of DL packages) exceeded it. Then, this truncated data was input into the model. For instance, as depicted in Figure 3, when the timestep is set to T, our monthly data is truncated, retaining data only up to the first T-th month, and subsequently fed into the model. By training multiple LSTM models with varying timestep values (in months), our approach can predict the sustainability of DL packages with diverse lifespans.

Followed by prior studies [63], [38], we randomly divided the studied packages into training and testing sets, with an 80%-20% ratio. Consequently, we obtained predicted sustainability outcomes for each DL package in each month, and thus obtained the predicted sustainability trajectories for each DL package. Repeating the above process five times, we derived the final result with their error bounds.

Notably, we pick a simple LSTM model in this paper. One might try other models, which we leave to future work. Our goal is to show that a simple time-series classification model is sufficient to get good prediction results.

Evaluation Metrics: To assess the effectiveness of the LSTM models in forecasting sustainability, we employed commonly adopted metrics in SE tasks, i.e., the Accuracy, Precision, Recall, and F1-score.

D. SHAP-based Interpretable Model

DL models, such as LSTMs, have gained vital attention due to their impressive predictive accuracy. However, they are black-box and lack of interpretability, making them less ideal for decision-making tasks. To better explain model decisions, we adopt an interpretable model [36] to explain the output of DL models. Model-agnostic explanation methods, such as SHapley Additive exPlanations (SHAP) [42], is a global interpretable model that treats the DL model as a black-box, and attempts to approximate the relationship between the input sample and the output prediction. It leverages a game-theory-based approach to compute shap values for each feature. Shap values represent the contribution of each input feature toward explaining the predicted outcome. Larger shap values indicate a more significant contribution of input features, while smaller shap values indicate a more minor contribution.

Implementation Details: Specifically, we first constructed an explainer using the *DeepExplainer* function from the Python SHAP package. The SHAP package was designed to explain the output of any machine learning model [42]. By using the *shap_values* function, and passing a test set *x_test*, we can obtain the interpretable scores *shap_values* of the explainer on the test set. Notably, the returned *shap_values* is an interpretation matrix M; an entry M[n, i, j, k] indicates the probability offset towards the direction of label *n* to the predicted result, which is contributed by feature *k* in project *i* at month *j*.

Project-Specific vs. Project-Overall: We employed SHAP results in two levels: project-specific level and project-overall level. In the former, we leveraged SHAP to compute monthly shap values for each feature, and aggregated them over all months for each DL package to form a project-specific distribution. Conversely, in the latter, we aggregated project-specific shap values over all packages. Thus, we obtained the shap values for each feature over all packages.

IV. DATASET

Applying the method for constructing DL package SC discussed in Section III-A, we collected 1,033 packages in our DL package SC. Notably, the goal of Tan et al.'s study [54] is to construct two DL SCs as large as possible. However, our goal is to predict long-term sustainability and understand the determinants of sustainability for each package in the DL SC. Therefore, to safeguard the quality of our dataset, we performed a filtering process. Specifically, we removed packages if they had less than or equal to 2 contributors, less than 15 issues (i.e., the median number of issues in the DL package SC), or an age shorter than 2 years. The rationale is to eliminate personal projects [39], inactive projects [32] and ensure that the filtered packages have traceable records from a set of contributors [60]. Besides, since TensorFlow and PyTorch are widely adopted and popular enough [30], [34], we also removed them from further analysis. Finally, 262 DL packages remained for later analysis. Among them, 90 packages only depend on TensorFlow, 82 only depend on PyTorch, and 90 depend on both TensorFlow and PyTorch. Besides, 192 packages are labeled as "sustainable", and 70 are labeled as "dormant".

Subsequently, we proceed to obtain SC-related attributes for each DL package based on the constructed DL package SC as well as some other attributes of DL packages. The rationale for obtaining SC-related attributes and other attributes is to aid in comprehending why certain longitudinal features exhibit inconsistent effects across different DL packages, which is discussed in Section V-B. Here, *layer_tf* and *layer_torch* are the number of layers of each DL package in the DL SC. *upstream_project* and *downstream_project* are the count of upstream packages imported by the focal package, and the number of packages dependent on the focal package. Additionally, we calculate the *up_mutual_contributors* and

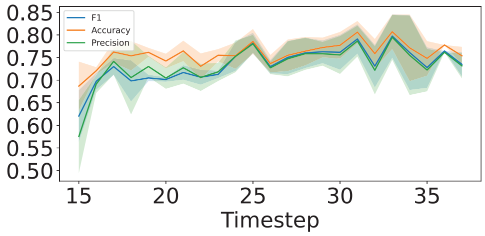


Fig. 4. Performance metrics of the LSTM models across various timesteps (in months), with the standard errors depicted in the shadow area. The orange line indicates the best accuracy of 0.81 achieved at 33 months.

down_mutual_contributors, which are the number of contributors who contribute to both the focal package and its upstream packages, and the number of contributors who contribute to both the focal package and its downstream packages. Furthermore, we compute the number of dormant upstream dependencies *upstream_dormant* for each DL package, and compute the sustainability rate of both upstream and downstream packages for the focal package *sc_sus_rate*. The sustainability rate is determined by dividing the number of sustainable upstream and downstream packages by the total number of packages in the upstream and downstream. Ultimately, we derived SC-related attributes for each DL package. Then, we further obtained more attributes of DL packages via GitHub API, including the number of stars, contributors, commits, issues, sizes, etc.

V. Results

Here, we present the results of our RQs.

A. RQ1: How effectively can we predict the sustainability of DL packages based on temporal traces?

We evaluate the predictive performance of our model across various timestep values. We also conduct a comparative analysis between our model and several baseline machine learning models to demonstrate the competitiveness of LSTM in addressing this problem.

1) Experimental Performance: Figure 4 presents the performance curves of the LSTM models over various timestep values. It unveils that the LSTM models demonstrate promising predictive performance during months 25 to 34, and reach its peak at 33 months, with an accuracy equal to 0.81, i.e., 81% of the DL packages can be correctly classified; with a precision equal to 0.79, i.e., for every five DL packages classified as sustainable, about four are correctly classified, and one is wrongly classified as dormant. Besides, it has a recall of 0.81 and an F1-score of 0.8. These findings suggest that the LSTM models can aptly predict the future sustained activity of DL packages.

To investigate why predictive performances are relatively promising during months 25 to 34, we further investigated the distribution of duration time for DL packages, as illustrated in Figure 5. Our findings indicate that most DL packages have duration time ranging from 27 to 36 months, with some surviving for more than 60 months and some surviving for

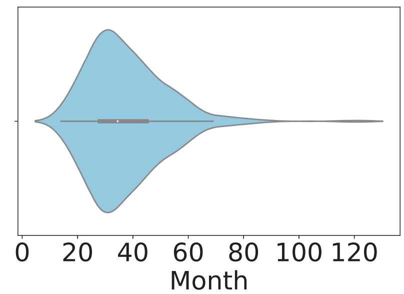


Fig. 5. The distribution of duration time (in months) of DL packages.

less than 15 months. Therefore, the insufficient data below 27 and above 36 months results in a decrease in the predictive performance of the LSTM models, which is in line with the duration time of DL packages.

TABLE II
Effectiveness of our approach and baseline methods.

Approach	Accuracy	Precision	Recall	F1-score
NB	0.42	**0.74**	0.42	0.41
KNN	0.70	0.61	0.70	0.64
XGBoost	0.68	0.64	0.68	0.65
RF	0.70	0.60	0.70	0.64
LR	0.73	0.64	0.73	0.65
MLP	0.74	0.57	0.74	0.63
LSTM (avg)	**0.75**	0.65	**0.75**	**0.69**
LSTM (best)	**0.81**	**0.79**	**0.81**	**0.80**

2) Comparison with Baselines: To demonstrate the competitiveness of LSTM on this problem, we compare our model with a range of baseline methods. These baseline methods include Naive Bayes (NB), K-Nearest Neighbors (KNN), XGBoost, Random Forest (RF), Logistic Regression (LR), and MultiLayer Perceptron (MLP), which are popular models with widespread use in software engineering [53], [48], [62]. As shown in Table II, our model outperforms all other baseline models across various evaluation metrics. Notably, as our LSTM models involve various timestep values, we additionally presented **LSTM (avg)**, which signifies the average performance of our LSTM models with diverse timestep values. This also serves to demonstrate the robustness of our approach. Moreover, **LSTM (best)** represents the best performance achieved as described in Section V-A1. In this regard, we observe that, in most cases, both **LSTM (avg)** and **LSTM (best)** exhibit better performance than the baseline methods, confirming the efficacy and robustness of our approach.

> Our LSTM models can effectively predict the future sustained activity of DL packages, and achieve an accuracy of up to 0.81 at month 33.

B. RQ2: What are the determinants of sustainability?

Here, we leverage the SHAP model to interpret the outcome of the LSTM models, with the purpose of demonstrating the contribution of each feature towards the predicted sustainability by generating monthly shap values for each feature. Take the sustainable DL package *Jax* [9], as an example,

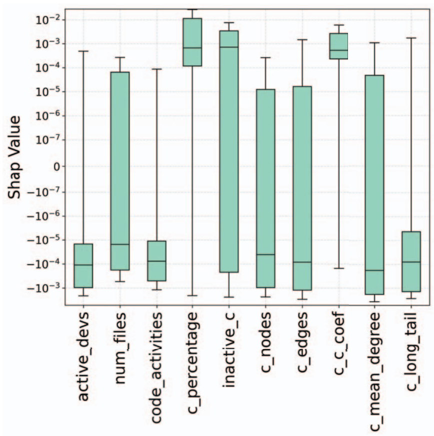


Fig. 6. The shap values of all features from a sustainable project named "Jax". The shap values are aggregated over all months, showing the overall contribution of each feature at the project-specific level.

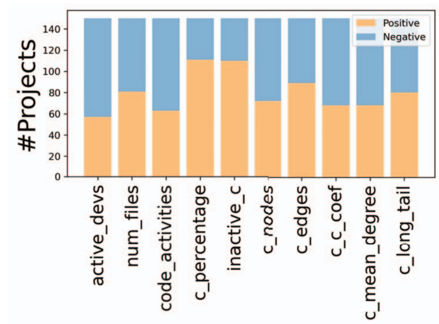


Fig. 7. The overall impact of all features across all DL packages, where blue indicates a negative impact and orange indicates a positive impact.

we illustrate how we interpret the output at the project-specific level, as depicted in Figure 6. It is worth noting that the shap values are aggregated over all months, and the median values provide a comprehensive understanding of each feature's relative importance in sustainability prediction for each DL package.

Specifically, we observe that the number of active developers `active_devs` and the number of commits `code_activities` are negatively correlated with the sustainability of package *Jax*. This finding appears to be contrary to conventional knowledge. A potential explanation for this finding may be that an excessive number of active developers or commits each month may lead to code fragmentation, and increase the complexity of code and interpersonal relationships, which negatively affects the sustainability of the DL package. Another possible explanation may be that although the number of active developers and commits are high each month, the package may still be under-resourced regarding code development, bug maintenance, and other aspects. For instance, although there are many active developers and commits each month, we still observe a significant number of accumulated bugs [6] in this package, which further corroborates our conjecture.

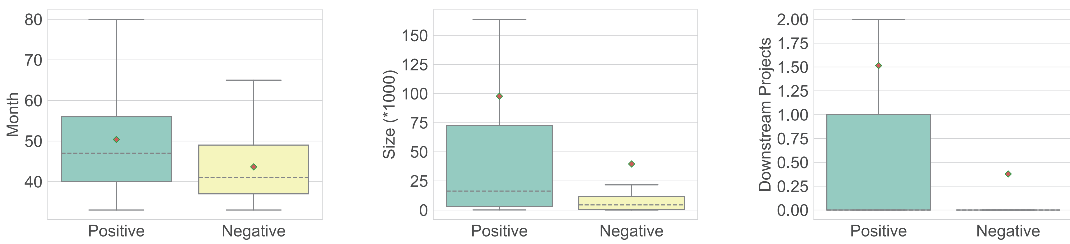
However, the percentage of commits performed by active developers `c_percentage` and the clustering coefficients `c_c_coef` in technical networks are positively correlated with the sustainability of the package. This suggests that a focused effort by a core group of developers, along with the collaborations of contributors, can positively sustain the DL package. This finding is in line with the above findings, which indicate that DL packages with fewer but more centralized developers and a balanced collaboration of their contributions are more likely to become self-sustainable. Instead, just a large number of developers and commits may be detrimental to the sustainability of DL packages.

Subsequently, we obtained the signs of shap values over all months across all DL packages to determine the consistency of a feature's contribution. That is, whether a feature is positive to the sustainability across all DL packages, or it is only positive to the sustainability of several packages, while negative to most other packages. As a result, we derived the aggregated signs of shap values of all features across all DL packages, as depicted in Figure 7. Here, orange indicates a positive effect, and blue indicates a negative effect.

Results uncover that these features present inconsistent effects across various DL packages. For instance, the percentages of commit activities performed by active developers `c_percentage`, and the ratio of the top 3 longest intervals between successive commits `inactive_c`, show positive correlations with the sustainability of 74% and 73% of DL packages, respectively. This implies that, in most DL packages, the focused efforts of a core group of developers can promote the sustainable growth of DL packages. Meanwhile, by our manual observation, the value of `inactive_c` is relatively small for most sustainable DL packages, and this feature presents a positive effect on the sustainability of most DL packages. However, the value of `inactive_c` is relatively large in some dormant DL packages, which in turn, negatively affects the sustainability of minor DL packages.

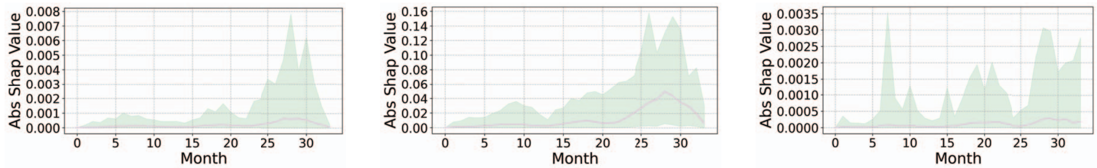
To better comprehend why certain features exhibit inconsistent effects across different DL packages, for each feature, we compare the distribution of attributes between DL packages where the feature has a positive effect and packages where the feature has a negative effect. The attributes include duration time, package sizes, and SC-related attributes (as described in Section IV). Then, we select several features as examples to depict the corresponding boxplots, which is shown in Figure 8. We further perform the Wilcoxon rank-sum test [44] to investigate if there exist statistically significant differences between the two groups, and use Cliff's delta [17] to examine the effect sizes.

Figure 8(a) reveals that the duration time of DL packages where the feature `active_devs` has a positive effect is longer than that of DL packages where the feature `active_devs` has a negative effect. Moreover, the statistical test shows that the difference is statistically significant, with a small effect size. As for the feature `c_percentage` in Figure 8(b), we find that DL packages where the feature has a positive effect are larger in size than those where the feature



(a) active_dev (b) c_percentage (c) c_c_coef

Fig. 8. Comparison of Duration time (a), Sizes (b), and number of downstream projects (c) for DL packages where the features display positive/negative effects.



(a) c_mean_degree (b) c_percentage (c) num_files

Fig. 9. The overall absolute shap values of three sample features, which are the mean degree in technical networks (a), the percentage of commits conducted by active developers (b), and the number of code files (c).

has a negative effect. And the statistical test result indicates that the difference is statistically significant, with a medium effect size. Furthermore, Figure 8(c) illustrates that the feature `c_c_coef` has a positive effect on DL packages with more downstream projects, while displaying a negative effect on DL packages with fewer downstream projects. Simultaneously, the statistical test indicates that the difference is statistically significant, and the effect size is negligible. These results imply that although a specific feature has inconsistent effects across different DL packages, for DL packages where the feature has a positive effect and DL packages where the feature has a negative effect, there are statistically significant differences between them on some attributes.

Notably, we only establish correlations between predictive features and package attributes here, rather than providing causalities. Therefore, we recommend future researchers to further investigate the causal relationships between package attributes and the inconsistent effects of predictive features.

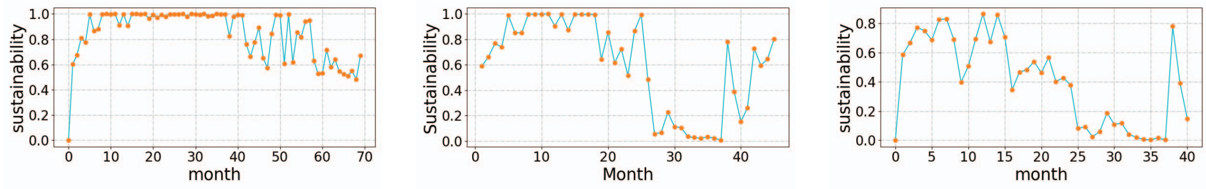
As SHAP can interpret LSTM models for each month, we thus can check the dynamics of shap values over time for each feature. Taking the features of `c_mean_degree`, `c_percentage`, and `num_files` as examples, we illustrate their effect trajectories in Figure 9. The results uncover that the effects of `c_mean_degree` and `c_percentage` become increasingly important before 28 months, after which their importance decreases. Nevertheless, these two features present a more vital effect during months 25 to 30. However, the effect of `num_files` differs from the other two features. Its importance remains relatively vital throughout the entire duration time, and becomes more important in the second half of the duration time. These phenomena may be explained by the changes in sizes, contributors, and other factors during the evolution of DL packages. Before the 28 months, DL packages may grow fast, leading to a gradual increase in the effects of many features. After that, the sizes or the number of contributors in DL packages may reach saturation; DL packages may become relatively stable, making the effects of features begins to decrease.

> Interpretable results reveal that `c_percentage` and `c_c_coef` are positively correlated with the sustainability of most DL packages, while `active_devs` and `code_activities` are negatively correlated with the sustainability of most DL packages. In this regard, for these DL packages, fewer but more centralized developers and a balanced collaboration of the contributions are more likely to foster the sustainable development. Instead, simply a large number of developers and commits are prone to be detrimental to sustainability.

C. RQ3: What are the trajectories of sustainability for DL packages?

To have a deeper understanding of the changes of sustainability for DL packages over time, we generated sustainability trajectories for each DL package, and made it publicly available at https://github.com/greenlight2000/DL_Package_Sustainability. Then, we applied the Mann-Kendall test [43], [40], [25] to effectively assess if there is a monotonic upward or downward trend of the sustainability trajectories for each DL package over time. Notably, a monotonic upward or downward trend means that the variable consistently increases or decreases over time, but the trajectories may or may not be linear.

Consequently, we find that 6% of sustainable DL packages (e.g., *edx-enterprise* [2] and *innvestigate* [8]) present statistically decreasing trends over time, 63% of sustainable DL packages (e.g., *dace* [1] and *neural-tangents* [10]) present statistically increasing trends over time, and the other sustainable DL packages (e.g., *GPflow* [7] and *f2format* [5])



(a) *GPflow* (b) *edx-enterprise* (c) *nucleus*

Fig. 10. Sustainability changes of DL packages with different types of trajectories. *GPflow* is sustainable, with statistically no trend, although it presents a decreasing trajectory in the second half of duration time; *edx-enterprise* is sustainable, while presents a statistically significant decreasing trajectory; and *nucleus* is dormant, and presents a statistically significant decreasing trajectory.

show no trend. As for dormant DL packages, 34% of them (e.g., *encodermap* [4] and *nucleus* [11]) show statistically decreasing trends over time, while 10% of them show statistically increasing trends, and the others exist no trend. By generating sustainability trajectories for each DL package and assessing their upward/downward trend, we can provide effective guidance for developers and users, to ensure that they select the suitable DL packages.

After that, we further take 3 representative DL packages (2 sustainable and 1 dormant) as examples to display their sustainability trajectories in Figure 10, and discuss the critical turning points with the aid of our interpretable model. As a result, we find that the sustainability trajectory of *GPflow* in Figure 10(a) exhibits an overall increasing trend during the first 10 months, keeps high sustainability for a prolonged period, but experiences fluctuations and declines after the 40th month. Combined with interpretable results, the increasing trend during the first 10 months may be due to the increasing number of active developers in this package and the frequent commits contributed by developers. After that, the development process becomes stable with a relatively high and stable number of active developers. However, its sustainability comes to fluctuations and declines after the 40th month. This may be due to the distinct decreasing number of active developers and their contributions, while the fluctuations may be due to the big releases during this period.

Regarding the sustainability trajectory of *edx-enterprise* in Figure 10(b), it displays a significant downturn at months 26 to 38, while experiencing a surprising upturn after month 41. Combined with interpretable results, the significant downturn may be due to the rapid reduction of active developers and their contributions, where the number of active developers decreases from a dozen or so to only 0 to 6 during this period. There are also very few development activities such as code commits. After the 41th month, the number of active developers comes to increase, along with an increase of development activities, making a surprising upturn afterward.

As for the sustainability trajectory of *nucleus* in Figure 10(c), it experiences fluctuations in the first half of duration time, while coming to dormant in the second half of duration time. By incorporating the interpretable results, we observe that the number of active developers and their code contributions is relatively low during the first half of the duration time. Meanwhile, the time intervals between successive code commits are also very long. When it comes to the second half of the duration time, some core developers left, and only 0 or 1 active developers remain. Meanwhile, there are hardly any development activities happening anymore, making the package dormant.

> Although some DL packages are sustainable, their sustainability trajectories present statistically decreasing trends over time, with approximately 6% of sustainable DL packages demonstrating such a trend. Meanwhile, it is worth noting that some DL packages, even if they have been dormant for a certain period, can still have the potential to return to be sustainable.

VI. Implications

Automated sustainability prediction tool: Results in RQ1 reveal that our constructed LSTM models can effectively predict the sustainability of DL packages. Therefore, we recommend practitioners to make use of it. We also recommend researchers to incorporate as many DL packages as possible, and develop automatic tools to forecast and monitor the sustainability of as many DL packages as possible. Hence, they can help DL package maintainers, developers, and users comprehend the dynamic status of various DL packages, which, in turn, foster the sustainable development of DL packages. Moreover, researchers can extend the automatic tools to other domains, e.g., the packages in the PyPI ecosystem, to foster the sustainable development of Python programming communities.

Balance of quantity and collaboration: Our summary of findings in RQ2 reveals that these package maintainers and developers should be aware of that, in contrast to conventional knowledge, simply improving the number of developers and commits may not always benefit the sustainability of DL packages. When there is an excessive number of developers and their commit contributions, it is likely to be detrimental to sustainability. Therefore, these package maintainers and developers should encourage collaboration among a core group of developers and strive to maintain a balance between the quantity and the collaboration of developers in DL packages.

Discussing features objectively and specifically: Additionally, findings illustrated in Figure 7 of RQ2 also reveal that a particular feature exhibits inconsistent effects across different DL packages. However, DL packages where a certain feature has positive effects and those where a certain feature has negative effects may show significant differences in some specific attributes. For instance, the feature `c_c_coef` displays a positive effect on DL packages with more downstream projects, while exhibiting a negative effect on DL packages with fewer downstream projects. Therefore, package maintainers and developers should recognize that there is no universally applicable approach to ensure sustainability based on the given features. Especially, they should apply our interpretable module to generate specific shap values for these features in their DL packages, and then, in accordance with the specific shap values and the specific attributes of their DL packages to comprehend the effect of these features on their DL packages. In this way, they can foster the long-term sustainability of their DL packages, and provide more solid dependencies for downstream packages and projects.

DL packages monitoring and selecting: Results in RQ3 furnished us with the sustainability trajectories of all DL packages in our dataset. Findings indicate that approximately 6% of the sustainable DL packages present statistically significant decreasing trends over time. This serves as a warning to DL package maintainers that, although DL packages are sustainable in a certain period, their sustainability trajectory may decline, and eventually become dormant in the near future. Therefore, we recommend that DL package maintainers leverage the dynamic sustainability forecast in RQ3, and closely monitor the points of downturn in their packages. In this way, they can react proactively and foster the long-term sustainability of their DL packages.

Simultaneously, as we have derived the sustainability trajectories for DL packages, developers and users can employ the trends of these sustainability trajectories in conjunction with other metrics (e.g., the stars of a given DL package) to inform a better assessment of which DL packages to choose. Besides, given that some DL packages have already been dormant, or shown statistically significant decreasing trends even though they are sustainable, developers and users must be careful when depending on such DL packages. They should conduct dependency replacement for this kind of DL package if necessary [46].

VII. Threats To Validity

Construct validity. The construct threat relates to the label of sustainability for DL packages. In this paper, our label of sustainability for DL packages is based on extant literature [59], [62]. A prior study [59] indicates that a repository can be considered as "dormant" if its average commit per month in the 12 months prior to its most recent commit is less than one. Otherwise, it can be considered as "sustainable". Similarly, another study [62] adopted sustained activity as the main proxy of project sustainability. They also applied commit activity to represent the long-term sustained activity. Therefore, to alleviate this threat, we adopted the proxy of sustainability in Xiao et al.'s [62] study and defined "sustainable"/"dormant" status according to the definition in Valiev et al.'s [59] study. In this way, we can ensure a relatively long-term dynamics of the definition.

Internal validity. The first threat pertains to the construction of our DL package SC. Previous studies [54], [20] have independently constructed two hierarchical DL SCs for TensorFlow and PyTorch. However, they aimed to construct DL SCs as large as possible by incorporating as many dependent packages and projects as possible. In this regard, there are many redundant DL packages and projects that belong to both SCs. To mitigate the redundancy threat and ensure the quality of our dataset, we incorporate DL packages that depend on Tensorflow and PyTorch to construct a unified DL package SC according to their dependencies. Another threat concerns the selection of longitudinal features. Due to limitations in data accessibility through the GitHub API, we do not include social-related features in our study, since email communications cannot be obtained via GitHub API. However, since technical (code-based) activities are major activities in OSS projects [64], which lowers the risk to a certain degree.

External validity. A crucial external threat involves the generalizability of our findings. The constructed DL package SC in this paper is based on the dataset published in [54]. To ensure the quality of our dataset, we have further filtered out some DL packages that may be personal projects or survive for a short time. Hence, our findings may not be applicable to all DL packages. Besides, since we focus on the sustainability of DL packages, thus, the findings may not generalize to packages in other domains. Regarding this, incorporating additional packages from diverse domains may ameliorate this risk.

VIII. Conclusion and Future Work

This paper reports a study focusing on forecasting the long-term sustainability of DL packages. We conduct our study on 262 DL packages and train LSTM models to forecast the long-term sustainability of DL packages (RQ1), interpret and find the determinants of sustainability (RQ2), and generate trajectories of sustainability for each DL package (RQ3). Our results uncover that our LSTM models can effectively predict the sustainability of DL packages. Moreover, fewer but more centralized developers and a balanced collaboration are more likely to help sustain the DL packages. Furthermore, although some DL packages are sustainable, their sustainability trajectories show statistically significant decreasing trends over time. In the future, we seek to investigate the effectiveness of our model, extend our analysis to more DL packages, and enhance the generalizability of the findings to a broader spectrum of OSS packages in various domains.

Acknowledgment

This research is supported by Zhejiang Provincial Natural Science Foundation of China (No. LQ24F020019), Scientific research project of Zhejiang Provincial Education Department (No. Y202351453 and No. FX2023067), National Science Foundation of China (No.U20A20173 and No.62372398), and the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (Award ID: MOET32020-0004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

REFERENCES

[1] "dace." [Online]. Available: https://github.com/spcl/dace
[2] "edx-enterprise." [Online]. Available: https://github.com/openedx/edx-enterprise
[3] "efficientnet." [Online]. Available: https://github.com/qubvel/efficientnet
[4] "encodermap." [Online]. Available: https://github.com/AG-Peter/encodermap
[5] "f2format." [Online]. Available: https://github.com/pybpc/f2format
[6] "google/jax/issues." [Online]. Available: https://github.com/google/jax/issues
[7] "Gpflow." [Online]. Available: https://github.com/GPflow/GPflow
[8] "innvestigate." [Online]. Available: https://github.com/albermax/innvestigate
[9] "Jax." [Online]. Available: https://github.com/google/jax
[10] "neural-tangents." [Online]. Available: https://github.com/google/neural-tangents
[11] "nucleus." [Online]. Available: https://github.com/google/nucleus
[12] G. Avelino, E. Constantinou, M. T. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–12.
[13] N. D. Bui, Y. Yu, and L. Jiang, "Self-supervised contrastive learning for code retrieval and summarization via semantic-preserving transformations," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 511–521.
[14] J. Cambronero, H. Li, S. Kim, K. Sen, and S. Chandra, "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*, 2019, pp. 964–974.
[15] J. Chen, Y. Liang, Q. Shen, J. Jiang, and S. Li, "Toward understanding deep learning framework bugs," *ACM Transactions on Software Engineering and Methodology*, 2022.
[16] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1290–1299.
[17] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.
[18] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, 2017, pp. 186–196.
[19] M. Dilhara, A. Ketkar, and D. Dig, "Understanding software-2.0: A study of machine learning library usage and evolution," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 4, pp. 1–42, 2021.
[20] K. Gao, R. He, B. Xie, and M. Zhou, "Characterizing deep learning package supply chains in pypi: Domains, clusters, and disengagement," *arXiv preprint arXiv:2306.16307*, 2023.
[21] K. Gao, Z. Wang, A. Mockus, and M. Zhou, "On the variability of software engineering needs for deep learning: Stages, trends, and application types," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 760–776, 2022.
[22] B. Gezici, N. Özdemir, N. Yılmaz, E. Coşkun, A. Tarhan, and O. Chouseinoglou, "Quality and success in open source software: a systematic mapping," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2019, pp. 363–370.
[23] A. H. Ghapanchi, A. Aurum, and G. Low, "A taxonomy for measuring the success of open source software projects," *First Monday*, 2011.
[24] M. Gheini, X. Ren, and J. May, "Cross-attention is all you need: Adapting pretrained transformers for machine translation," *arXiv preprint arXiv:2104.08771*, 2021.
[25] R. O. Gilbert, *Statistical methods for environmental pollution monitoring*. John Wiley & Sons, 1987.
[26] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.
[27] Q. Guo, S. Chen, X. Xie, L. Ma, Q. Hu, H. Liu, Y. Liu, J. Zhao, and X. Li, "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, 2019, pp. 810–822.
[28] R. Gupta, S. Pal, A. Kanade, and S. Shevade, "Deepfix: Fixing common c language errors by deep learning," in *Proceedings of the aaai conference on artificial intelligence*, vol. 31, no. 1, 2017.
[29] J. Han, S. Deng, D. Lo, C. Zhi, J. Yin, and X. Xia, "An empirical study of the dependency networks of deep learning libraries," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 868–878.
[30] J. Han, E. Shihab, Z. Wan, S. Deng, and X. Xia, "What do programmers discuss about deep learning frameworks," *Empirical Software Engineering*, vol. 25, pp. 2694–2747, 2020.
[31] C. Huang, J. Zhang, Y. Zheng, and N. V. Chawla, "Deepcrime: Attentive hierarchical recurrent networks for crime prediction," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 1423–1432.
[32] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1110–1121.
[33] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.
[34] M. J. Islam, G. Nguyen, R. Pan, and H. Rajan, "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, pp. 510–520.
[35] M. J. Islam, R. Pan, G. Nguyen, and H. Rajan, "Repairing deep neural networks: Fix patterns and challenges," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1135–1146.
[36] J. V. Jeyakumar, J. Noor, Y.-H. Cheng, L. Garcia, and M. Srivastava, "How can i explain this to you? an empirical study of deep neural network explanation methods," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4211–4222, 2020.
[37] L. Jia, H. Zhong, X. Wang, L. Huang, and X. Lu, "The symptoms, causes, and repairs of bugs inside a deep learning library," *Journal of Systems and Software*, vol. 177, p. 110935, 2021.
[38] M. Joblin and S. Apel, "How do successful and failed projects differ? a socio-technical analysis," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–24, 2022.
[39] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 92–101.
[40] M. G. Kendall, "Rank correlation methods." 1948.
[41] Z. Liao, M. Yi, Y. Wang, S. Liu, H. Liu, Y. Zhang, and Y. Zhou, "Healthy or not: A way to predict ecosystem health in github," *Symmetry*, vol. 11, no. 2, p. 144, 2019.
[42] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.
[43] H. B. Mann, "Nonparametric tests against trend," *Econometrica: Journal of the econometric society*, pp. 245–259, 1945.
[44] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.
[45] V. Midha and P. Palvia, "Factors affecting the success of open source software," *Journal of Systems and Software*, vol. 85, no. 4, pp. 895–905, 2012.
[46] C. Miller, C. Kästner, and B. Vasilescu, ""we feel like we're winging it:" a study on navigating open-source dependency abandonment."
[47] L. Nyman and J. Lindman, "Code forking, governance, and sustainability in open source software," *Technology Innovation Management Review*, vol. 3, no. 1, 2013.
[48] S. Pan, J. Zhou, F. R. Cogo, X. Xia, L. Bao, X. Hu, S. Li, and A. E. Hassan, "Automated unearthing of dangerous issue reports," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 834–846.
[49] J. Piggott, "Open source software attributes as success indicators," *Univ. of Twente*, 2013.
[50] C. M. Schweik and R. C. English, *Internet success: a study of open-source software commons*. MIT Press, 2012.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[52] Stănciulescu, L. Yin, and V. Filkov, "Code, quality, and process metrics in graduated and retired asfi projects," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 495–506.

[53] C. Stanik, L. Montgomery, D. Martens, D. Fucci, and W. Maalej, "A simple nlp-based approach to support onboarding and retention in open source communities," in *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2018, pp. 172–182.

[54] X. Tan, K. Gao, M. Zhou, and L. Zhang, "An exploratory study of deep learning supply chain," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 86–98.

[55] F. Thung, S. Wang, D. Lo, and L. Jiang, "An empirical study of bugs in machine learning systems," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, 2012, pp. 271–280.

[56] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[57] L. N. Tidjon, B. Rombaut, F. Khomh, A. E. Hassan *et al.*, "An empirical study of library usage and dependency in deep learning frameworks," *arXiv preprint arXiv:2211.15733*, 2022.

[58] A. Utture, S. Liu, C. G. Kalhauge, and J. Palsberg, "Striking a balance: pruning false-positives from static call graphs," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 2043–2055.

[59] M. Valiev, B. Vasilescu, and J. Herbsleb, "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, pp. 644–655.

[60] Z. Wang, Y. Feng, Y. Wang, J. A. Jones, and D. Redmiles, "Unveiling elite developers' activities in open source projects," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 3, pp. 1–35, 2020.

[61] T. Xia, W. Fu, R. Shu, R. Agrawal, and T. Menzies, "Predicting health indicators for open source projects (using hyperparameter optimization)," *Empirical Software Engineering*, vol. 27, no. 6, p. 122, 2022.

[62] W. Xiao, H. He, W. Xu, Y. Zhang, and M. Zhou, "How early participation determines long-term sustained activity in github projects?" *arXiv preprint arXiv:2308.06005*, 2023.

[63] L. Yin, Z. Chen, Q. Xuan, and V. Filkov, "Sustainability forecasting for apache incubator projects," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1056–1067.

[64] Y. Yue, Y. Wang, and D. Redmiles, "Off to a good start: Dynamic contribution patterns and technical success in an oss newcomer's early career," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 529–548, 2022.

[65] R. Zhang, W. Xiao, H. Zhang, Y. Liu, H. Lin, and M. Yang, "An empirical study on program failures of deep learning jobs," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 1159–1170.

[66] Y. Zhang, Y. Chen, S.-C. Cheung, Y. Xiong, and L. Zhang, "An empirical study on tensorflow program bugs," in *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*, 2018, pp. 129–140.

[67] Z. Zhang, H. Zheng, R. Hong, M. Xu, S. Yan, and M. Wang, "Deep color consistent network for low-light image enhancement," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1899–1908.

[68] F. Zhou, B. Zhou, S. Zhao, and G. Pan, "Deepoffense: a recurrent network based approach for crime prediction," *CCF Transactions on Pervasive Computing and Interaction*, vol. 4, no. 3, pp. 240–251, 2022.