Towards Automation for MLOps: An Exploratory Study of Bot Usage in Deep Learning Libraries

Akond Rahman*, Farzana Ahamed Bhuiyan†, Mohammad Mehedi Hassan‡, Hossain Shahriar§ and Fan Wu¶
*Tennessee Tech University, † Meta, ‡Independent University, ‡Kennesaw State University, §Tuskegee University
*akond.rahman.buet@gmail.com, †fbhuiyan42@fb.com, ‡mehedi.bueteee.23@gmail.com
§hshahria@kennesaw.edu, ¶fwu@tuskegee.edu

***Abstract*—Machine learning (ML) operations or MLOps advocates for integration of DevOps-related practices into the ML development and deployment process. Adoption of MLOps can be hampered due to a lack of knowledge related to how development tasks can be automated. A characterization of bot usage in ML projects can help practitioners on the types of tasks that can be automated with bots, and apply that knowledge into their ML development and deployment process. To that end, we conduct a preliminary empirical study with 135 issues reported mined from 3 libraries related to deep learning: Keras, PyTorch, and Tensorflow. From our empirical study we observe 9 categories of tasks that are automated with bots. We conclude our work-in-progress paper by providing a list of lessons that we learned from our empirical study.**

***Index Terms*—bots, deep learning, deployment, devops, empirical study, machine learning, mlops**

I. Introduction

Upon construction of machine learning (ML) models, practitioners face challenges in deploying those models into production efficiently and reliably [2]. To that end, the concept of ML operations or MLOps has been set in motion in recent years [2]. MLOps advocates for integration of DevOps-related practices into the ML development and deployment process so that ML models can be deployed in production reliably and efficiently [2]. The MLOps market is projected to reach $126 billion by 2025 [1], which shows the interest amongst practitioners in adopting MLOps. Information technology (IT) organizations, such as NVIDIA [25] and TransLink [28] has started the adoption of MLOps. Adoption of MLOps has yielded benefits for IT organizations, e.g., with MLOps TransLink was able to deploy 16,000 ML models in production that were used for predicting departure and arrival times for a Canada-based transportation system [28].

Despite growing interest, practitioners face challenges in accomplishing automation-related objectives for MLOps [2]. These challenges are further aggravated by a lack of knowledge on how to apply software engineering practices to automate tasks that are related to ML development and deployment [9]. As automation is pivotal for mature adoption of MLOps [22], [30], mitigation of automation-related challenges for practitioners is necessary. One approach to mitigate ML-related automation challenges is characterizing the use of automated agents, such as bots in established ML projects. Bots are used to automate development tasks in order to make the entire software development process efficient [31], [27]. As practitioners prefer to learn from other practitioners in the same domain [18], [23], a characterization study of bot usage in established ML projects, such as the Keras [12] can help practitioners gain knowledge on how established ML projects are using bots, and apply that knowledge in their own ML development process.

Accordingly, we answer the following research question: ***RQ*: What development tasks are automated with bots for deep learning libraries?**

We conduct an empirical study with 135 issue reports mined from three deep learning libraries: Keras [12], PyTorch [20], and Tensorflow [1], which are well-known software projects that provide ML APIs for practitioners to use [13]. By applying open coding [26] we identify a list of development tasks that are automated with bots for these three libraries.

Our contribution is a list of development tasks that are automated with bots in deep learning libraries.

We organize rest of the paper as follows: we provide methodology of our paper in Section II. We provide our findings in Section III. In Section IV we summarize the lessons learned from our empirical study. We discuss related work in Section V. Finally, we conclude the paper in Section VI.

II. Methodology

We conduct our empirical study by mining issue reports from three deep learning libraries: Keras, PyTorch, and TensorFlow. We select these libraries as these are established and well-known software projects that provide ML APIs for practitioners to use [13]. An overview of our methodology is presented in Figure 1. We describe our methodology as follows:

A. Mining Issue Reports

Altogether, we mine 68,400 issue reports from the repositories of Keras, PyTorch, and TensorFlow on Nov 15, 2021. After collecting these repositories we apply a filtering criteria to identify issues in which bots have been involved with. For

[1]https://neu.ro/2021-mlops-platforms-vendor-analysis-report/
[2]https://datatonic.com/insights/ai-automation-mlops/

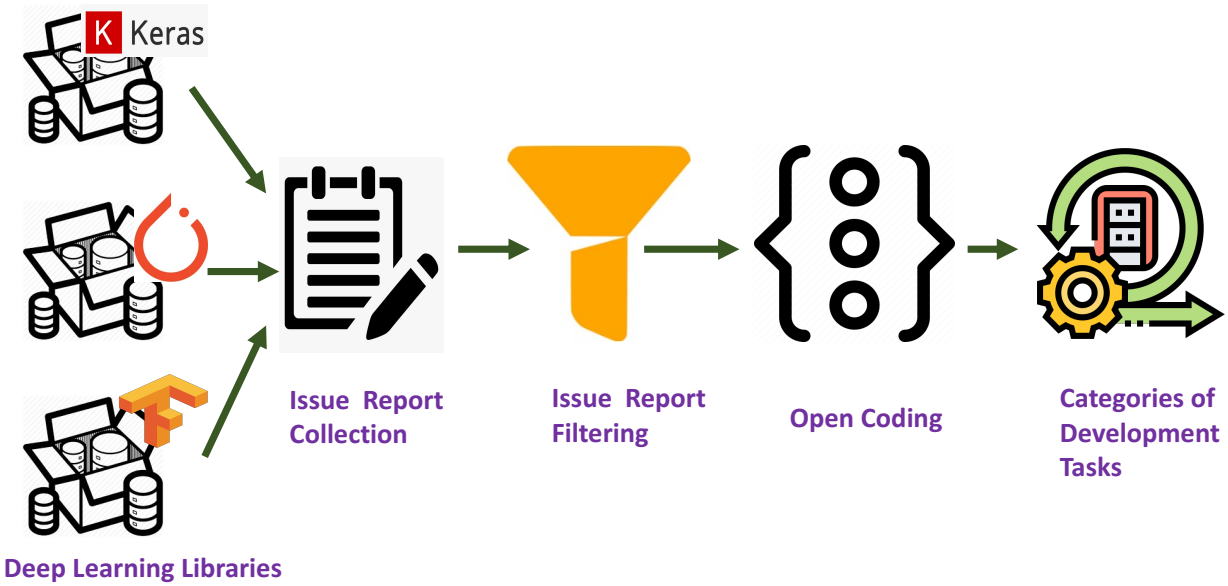

Fig. 1: An overview of our research methodology.

filtering issues where bots are mentioned, we *first* apply a keyword search where we inspect for the word 'bot' in the titles, descriptions, and comments for all issues. *Second*, we apply manual inspection to eliminate false positives so that we can ensure that the search results from the first step are actually bot-related.

By applying keyword search, we identify 254 issue reports, out of which 135 are true positives. We use these 135 issue reports to perform open coding as described in the next section.

B. Answer to RQ: What development tasks are automated with bots for deep learning libraries?

We apply a qualitative analysis technique called open coding [26], which is used to derive categories from unstructured text based on commonalities within text. The first author is the rater who applies open coding for each issue report. The rater inspects the content of each issue report to *first*, identify if bots are mentioned in the issue, and *second*, identify the development task that is being performed by the bot. Upon completion, the rater groups all identified activities into categories, and provides a mapping between each category and the bot that performs the identified task.

Rater Verification: Our open coding approach is conducted by the first author, which makes the open coding process susceptible to bias. We mitigate this bias by allocating another rater, who is the second author of the paper. The second author is a PhD student in the department, with two years of professional experience in software engineering. The second author is provided the set of 135 issue reports, and asked to map each issue to one or multiple categories identified by the first author. The second author independently conducts the process. Upon completion, the agreement rate between the first and second author is computed using Cohen's Kappa [6]. For the 135 issue reports the Cohen's Kappa between the two raters is 0.87, which is 'substantial' according to Landis and Koch [15].

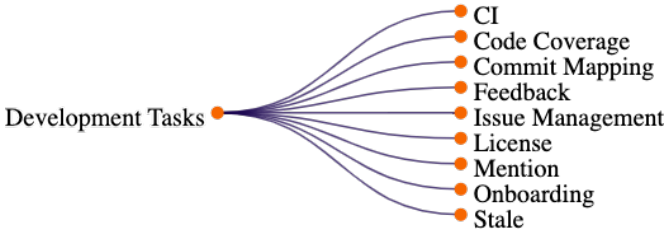

Fig. 2: Nine categories of development tasks automated with bots in deep learning libraries.

III. Results

In this section, we provide answer to **RQ: What development tasks are automated with bots for deep learning libraries?** With our open coding analysis we identify 9 categories of development tasks that are automated with bots, as summarized in Figure 2. A mapping of each development task and the identified bots is presented in Table I. We also report the proportion of issues in which a specific category of bot is mentioned in the 'Issue Proportion (%)' column. For example, in 2.2% of 136 issues we observe CI bots to be mentioned. We describe these categories alphabetically as follows:

I. Code Coverage: This category refers to the calculation of code coverage through static analysis. For example, in PyTorch the `codecov` bot is used to calculate code coverage. Use of bots, such as `codecov` eliminates involvement of developers, but can be susceptible to bugs as mentioned in an issue [24].

TABLE I: Mapping of Development Tasks and Bots

Category	Bot Names	Deep Learning Library	Issue Prop. (%)
Code Coverage	code-coverage-bot	PyTorch	0.7
Commit Mapping	facebook-github-bot	PyTorch	0.7
Continuous Integration	build-bot, ci-bot	Keras, Tensorflow	2.2
Feedback	tensorflow-bot, google-ml-butler, facebook-github-bot	PyTorch	13.3
Issue Mgmt.	pytorch-probot, zenhub-bot, pytorchbot, tensorflow-butler, tensorflow-capybara, tensorflow-bot, google-ml-butler, facebook-github-bot	PyTorch , Tensorflow	39.4
License	cla-bot	PyTorch	0.7
Mention	mention-bot	PyTorch	0.7
Onboarding	tensorflow-butler, tensorflow-capybara, tensorflow-bot, google-ml-butler	Tensorflow	14.1
Stale	tensorflow-butler, stale-bot, google-ml-butler	Keras, Tensorflow	28.1

According to the issue, the bot is known to provide incorrect code coverage, which motivated the contributor to abandon the use of codecov for another project: "*I dealt with this [incorrect code coverage] for NumPy a while back and the only robust solution was to ban the bot at the org level*" [24].

II. Commit Mapping: This category refers to the task of finding a commit that addresses a feature request or a bug fix. Bots that belong to this category automatically maps commits to issues in a project. The purpose is to reduce the manual effort in mapping submitted code changes in forms of commits and pull requests to open issues. Despite their perceived benefits, they can be pose challenges for contributors as observed in one issue [5]: " *I'm not able to make any sense of that PR and the many bot statuses and info listed there*".

III. Continuous Integration: This category refers to tasks used to build code changes using continuous integration (CI). CI is the practice of integrating code changes by automatically compiling, building, and executing test cases upon submission of code changes [8]. We observe bots used for CI to trigger builds in the CI server, and notify contributors about the status of each build. In this manner, contributors and maintainers are notified about the integration status of code changes automatically. However, as discussed in a Keras-related issue [16], we observe practitioners to face difficulties in obtaining accurate build results from CI servers because of the bot being buggy. In the issue a contributor commented "*As we can see, PR #12336 has been tackled by the abnormal CI bot for long days. It means that the CI system of the TensorFlow repository does not efficiently help reviewers that want to concentrate for productive code review*".

IV. Feedback: This category refers to the task of obtaining feedback from contributors. For example, in the case of TensorFlow the google-ml-butler bot asks a two-item question to obtain feedback from contributors related to issue resolution [14].

V. Issue Management: This category refers to management-related tasks for issues. We identify four categories of tasks that are performed by issue management bots: issue assignment, edits for issue descriptions, issue triaging, and issue closing. Typically, the above-mentioned tasks are performed by maintainers manually that results in significant development and maintenance effort [3]. Issue management bots are used in software projects to ease the effort in maintenance, but there are examples where we find these bots to incorrectly close issues or incorrectly assign issues. For example, in the case of PyTorch, the facebook-github-bot incorrectly closed an issue before adequate resolution of a documentation-related concern [29].

VI. License: This category refers to the tasks that are related with license usage. We identify one bot, the cla-bot, used by PyTorch, which checks if contributors have signed licensing agreements and add labels to pull requests [3].

VII. Mention: This category refers to the task of finding potential reviewers for a pull request. The perceived benefit of using mention bots is that it will reduce the turnaround time for pull request acceptance. The TensorFlow project uses the mention-bot to identify potential reviewers for pull requests.

VIII. Onboarding: This category refers to tasks that help newcomers on how to contribute to the code base and/or report a bug or a feature request. Whether it is submitting code changes or submitting issues for feature requests, a set of tasks must be completed by the contributor. Bots that map to this category inspects if these set of tasks are completed, and if not then the contributor is notified. For example, for the Tensorflow project, upon submission of a feature request, we observe the TensorflowButler bot to remind a contributor that relevant information, such as 'TensorFlow version'. 'Bazel version', 'CUDA/cuDNN version', 'GPU model and memory' are missing [19]. In this manner, a newcomer is automatically reminded of the necessary information that needs to accompany a feature request or reporting of a bug.

[3]https://github.com/apps/cla-bot

Despite reported benefits, bots used for onboarding can create negative impressions for contributors as they succumb to parsing errors, and fail to address the root cause. For example, in the case of reporting a documentation-related bug, a contributor was asked to provide system-level details by the `tensorflowbutler` bot. In response to the bot, the contributor provides a negative response stating "*Seriously? Read it – it specified ALL SYSTEMS/SYSTEM INDEPENDENT. So seriously: the issue I raised is about an obvious barrier to helping you improve your documentation. Your response inadvertently highlights another barrier ... Sorry to sound so hostile, and it's nothing personal, but I have found your 'Quick Start' documentation among the slowest I have ever worked through*" [19].

IX. Stale: This category refers to the task of finding development inactivity. Bots used for this category automatically identify tasks or feature requests for which no development activity has been recorded for a certain amount of days. For example, Wessel et al. [32] reported projects to use a median of 60 days to detect inactivity with `stale` bots. In the case of Tensorflow, the `tensorflowbutler` bot notifies issue participants when there is no activity for 14 days. In the case of Keras, the `google-ml-butler` bot notifies issue participants if there is no activity for 7 days, and automatically closes the issue if there is no activity for 14 days. A threshold-based approach used by `stale` bots can abruptly close issues, which in turn can annoy contributors. A contributor for the Tensorflow library remarks: "*It's so frustrating that such important issue just got ignored and secretly closed* " [7].

Our categorization of development tasks have been reported in existing work. Wessel et al. [31] conducted an empirical study with 93 projects, and observed the collected set of projects to use bots to automate tasks related to code coverage, CI, issue management, and licensing. In another work, Wessel et al. [32] investigated the use of `stale` bots in software development. Peng et al. [21] investigated how `mention` bots are used. Along with these task categories, we also observe bots in deep learning libraries to automate tasks related to onboarding, commit mapping, and detecting development inactivity.

IV. Lessons Learned

We summarize our findings as follows:

1) Bots, such as the `code-coverage-bot`, which are used in general purpose software projects are also used for deep learning libraries.
2) For deep learning libraries, both self-developed bots, such as `tensorlfow-bot`, and third-party bots (e.g., `google-ml-butler`) are used.
3) Bugs in code coverage and CI-related bots hamper productivity.
4) Documentation-related inadequacies can provide challenges for contributors who use bots for onboarding.
5) Use of bots for license, code coverage, commit mapping, and developer feedback is not common across the three deep learning libraries.
6) Negative human-bot interaction, such as frustrations about onboarding-related bots is prevalent, which necessitates integration of recommended design practices [4], [17] for development of ML-related bots.
7) For determining inactivity with `stale` bots, project maintainers should use fine-grained thresholds so that `stale` bots do not abruptly close issues. Liu et al. [17]'s guidelines can be helpful in this regard.

V. Related Work

Our paper is closely related to prior research that have focused on bot usage in GitHub-based software development. Wessel et al. [31] investigated bot usage on GitHub projects, classified the bots, collected several metrics to compare the state of the project before and after bot adoption, and interviewed project maintainers. Brown and Parnin [4] explained that due to poor bot design, human-bot interaction on GitHub can be inconvenient, and result in negative feedback from maintainers and contributors. Wessel et al. [32] investigated the `stale` bot, which closes abandoned issues and pull requests on GitHub, and examined the bots' configuration settings and changes over time. Peng et al. [21] analyzed `mention-bot` that has been integrated into the pull request workflow to perform the task of mentioning reviewers. Golzadeh et al. [10] detected the automatic answer of the bot on GitHub from the comments of pull requests and issue comments to prevent bias in socio-technical studies related to software development. In another work, Golzadeh et al. [11] proposed a classification model to identify bots in GitHub pull request activity and validated the model on a ground-truth dataset composed of several thousands of GitHub accounts and their associated repositories.

VI. Conclusion

As production-level machine learning development becomes increasingly relevant, gaining an understanding of how bots can be used to automate development tasks could be of relevance to practitioners. A characterization of how bots are used in deep learning libraries can help practitioners gain an understanding on how bots could be used to automate development tasks. To that end, we conduct a preliminary empirical study where we systematically categorize the development tasks that are automated with bots in three deep learning libraries: Kears, PyTorch, and TensorFlow. From our empirical study we observe bots are used to automated nine categories of development tasks. We also find that bots to not be a silver bullet for automating development tasks as they can be buggy, create false alerts, and not adequately address practitioner needs. Our work-in-progress paper lays the groundwork for future bot-related research in the domain of machine learning development.

ACKNOWLEDGMENT

We thank the PASER group at Tennessee Tech University for their valuable feedback. This research was partially funded by the U.S. National Science Foundation (NSF) award # 2026869.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[2] S. Alla and S. K. Adari, "What is mlops?" in *Beginning MLOps with MLFlow*. Springer, 2021, pp. 79–124.

[3] J. Anvik and G. C. Murphy, "Reducing the effort of bug report triage: Recommenders for development-oriented decisions," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 3, aug 2011. [Online]. Available: https://doi.org/10.1145/2000791.2000794

[4] C. Brown and C. Parnin, "Sorry to bother you: designing bots for effective recommendations," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 2019, pp. 54–58.

[5] CnybTseng, "build pytorch from source on ubuntu, building error from fbgemm::SparseAdaGradSignature," https://github.com/pytorch/pytorch/issues/47993, 2020, [Online; accessed 02-Jan-2022].

[6] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960. [Online]. Available: http://dx.doi.org/10.1177/001316446002000104

[7] davidnunes, "tf.nn.embedding_lookup_sparse Converting sparse IndexedSlices Warning," https://github.com/tensorflow/tensorflow/issues/23566, 2018, [Online; accessed 02-Jan-2022].

[8] P. Duvall, S. M. Matyas, and A. Glover, *Continuous Integration: Improving Software Quality and Reducing Risk (The Addison-Wesley Signature Series)*. Addison-Wesley Professional, 2007.

[9] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, p. 111031, 2021.

[10] M. Golzadeh, A. Decan, E. Constantinou, and T. Mens, "Identifying bot activity in github pull request and issue comments," *arXiv preprint arXiv:2103.06042*, 2021.

[11] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in github issue and pr comments," *Journal of Systems and Software*, vol. 175, p. 110911, 2021.

[12] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[13] 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*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 510–520. [Online]. Available: https://doi.org/10.1145/3338906.3338955

[14] jbuisine, "Cuda 11.0 with cuDNN v8.0.1 problem when using GPU : libcudnn.so.7: cannot open shared object fil," https://github.com/tensorflow/tensorflow/issues/41041, 2020, [Online; accessed 01-Jan-2022].

[15] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. [Online]. Available: http://www.jstor.org/stable/2529310

[16] leemgs, "[CI] The Travis build is pending for 3 days to check PR 12357," https://github.com/keras-team/keras/issues/12417, 2019, [Online; accessed 01-Jan-2022].

[17] D. Liu, M. J. Smith, and K. Veeramachaneni, "Understanding user-bot interactions for small-scale automation in open-source development," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1–8. [Online]. Available: https://doi.org/10.1145/3334480.3382998

[18] E. Murphy-Hill and G. C. Murphy, "Peer interaction effectively, yet infrequently, enables programmers to discover new tools," in *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, ser. CSCW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 405–414. [Online]. Available: https://doi.org/10.1145/1958824.1958888

[19] omatai, "There is an issue with your "new issue" page," https://github.com/tensorflow/tensorflow/issues/16350, 2018, [Online; accessed 11-Jan-2022].

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[21] Z. Peng and X. Ma, "Exploring how software developers work with mention bot in github," *CCF Transactions on Pervasive Computing and Interaction*, vol. 1, no. 3, pp. 190–203, 2019.

[22] A. Quinn, "Implementing Automation and an MLOps Framework for Enterprise-scale ML," https://www.iguazio.com/blog/implementing-automation-and-an-mlops-framework-for-enterprise-scale-ml/, 2021, [Online; accessed 09-Jan-2022].

[23] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, "Synthesizing continuous deployment practices used in software development," in *Proceedings of the 2015 Agile Conference*, ser. AGILE '15. USA: IEEE Computer Society, 2015, p. 1–10. [Online]. Available: https://doi.org/10.1109/Agile.2015.12

[24] rgommers, "Please stop the codecov bot from posting comments," https://github.com/pytorch/pytorch/issues/44187, 2020, [Online; accessed 12-Jan-2022].

[25] Rick Merrit, "What Is MLOps?" https://blogs.nvidia.com/blog/what-is-mlops/, 2020, [Online; accessed 11-Jan-2022].

[26] J. Saldana, *The Coding Manual for Qualitative Researchers*. SAGE, 2015.

[27] M.-A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 928–931.

[28] TechCrunch, "Why firms are welcoming MLOps into the fold of software development," https://techcrunch.com/sponsor/microsoftazure/why-firms-are-welcoming-mlops-into-the-fold-of-software-development/, 2022, [Online; accessed 12-Jan-2022].

[29] vadimkantorov, "TPlease stop the codecov bot from posting comments," https://github.com/pytorch/pytorch/issues/29697, 2019, [Online; accessed 11-Jan-2022].

[30] L. Visengeriyeva, A. Kammer, I. Bar, A. Kniesz, and M. Plod, "MLOps Principles," https://ml-ops.org/content/mlops-principles, 2022, [Online; accessed 09-Jan-2022].

[31] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in oss projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–19, 2018.

[32] M. Wessel, I. Steinmacher, I. Wiese, and M. A. Gerosa, "Should i stale or should i close? an analysis of a bot that closes abandoned issues and pull requests," in *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*. IEEE, 2019, pp. 38–42.