The State of the ML-universe: 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub

Danielle Gonzalez
Rochester Institute of Technology
Rochester, NY, USA
dng2551@rit.edu

Thomas Zimmermann
Microsoft Research
Redmond, WA, USA
tzimmer@microsoft.com

Nachiappan Nagappan
Microsoft Research
Redmond, WA, USA
nachin@microsoft.com

ABSTRACT

In the last few years, artificial intelligence (AI) and machine learning (ML) have become ubiquitous terms. These powerful techniques have escaped obscurity in academic communities with the recent onslaught of AI & ML tools, frameworks, and libraries that make these techniques accessible to a wider audience of developers. As a result, applying AI & ML to solve existing and emergent problems is an increasingly popular practice. However, little is known about this domain from the software engineering perspective. Many AI & ML tools and applications are open source, hosted on platforms such as GitHub that provide rich tools for large-scale distributed software development. Despite widespread use and popularity, these repositories have never been examined as a *community* to identify unique properties, development patterns, and trends.

In this paper, we conducted a large-scale empirical study of AI & ML Tool (700) and Application (4,524) repositories hosted on GitHub to develop such a characterization. While not the only platform hosting AI & ML development, GitHub facilitates collecting a rich data set for each repository with high traceability between issues, commits, pull requests and users. To compare the AI & ML community to the wider population of repositories, we also analyzed a set of 4,101 unrelated repositories. We enhance this characterization with an elaborate study of developer workflow that measures collaboration and autonomy within a repository. We've captured key insights of this community's 10 year history such as it's primary language (Python) and most popular repositories (Tensorflow, Tesseract). Our findings show the AI & ML community has unique characteristics that should be accounted for in future research.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**; • **Software and its engineering** → *Collaboration in software development*; *Software libraries and repositories.*

KEYWORDS

machine learning, artificial intelligence, mining software repositories, software engineering, Open Source, GitHub

MSR '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387473

ACM Reference Format:
Danielle Gonzalez, Thomas Zimmermann, and Nachiappan Nagappan. 2020. The State of the ML-universe: 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3379597.3387473

1 INTRODUCTION

In the last few years, artificial intelligence (AI) and machine learning (ML) have become ubiquitous terms. AI & ML tools are increasingly used in day-to-day applications. At the same time, the need for AI & ML applications has led to a tremendous growth in the GPU market. The 2019 Global Developer Population and Demographic Study by Evans Data Corporation estimates that about 7 million developers use artificial intelligence or machine learning in their development work, and another 9.5 million are expected to use it within the next twelve months [23]. With new emerging technologies, it is important to understand how existing development practices are affected. Initial work has focused on interviews and surveys to understand how AI & ML projects are different [1, 54], and the challenges that developers face [3, 21, 37, 58].

In this paper, we contribute additional insights into AI & ML development and triangulate results from existing studies. We characterize the landscape of AI & ML repositories on GitHub in order to understand the AI & ML boom in recent years and the differences between AI & ML and traditional software development. Specifically, we conduct a large-scale empirical study of GitHub to characterize and compare software development across three types of repositories (Section 2):

(1) **AI & ML Tools**: 700 AI & ML frameworks & libraries
(2) **Applied AI & ML**: 4,524 repositories using AI & ML
(3) Comparison: 4,101 repositories unrelated to AI & ML

GitHub is not the only platform hosting AI & ML software development. However, we chose to focus on GitHub due to its integration of collaborative development artifacts (issues, pull requests) into the repositories, allowing us to leverage mining tools to collect a rich dataset for each repository from a single source.

The research goal is to understand, among others things, the timeline of the AI & ML boom, ownership of AI & ML software, their popularity, and programming language use. In addition, we investigate collaboration and autonomy because they have been found to be important factors related to productivity [42, 49]. Some of our findings include (Sections 4 and 5.1):

- The oldest active AI & ML repository (cilib [9]) on GitHub was created in 2009. The annual proportion of new repositories related to AI & ML gradually rose since 2012, until the

"boom" in 2017. More applications of AI & ML are created annually than tools, libraries, and frameworks.
- The primary language for AI & ML is Python.
- Users own the majority (79.1%) of applied AI & ML repositories, but organizations own more (51.43%) of the AI & ML tools.
- IBM owns the most (61) AI & ML repositories.
- AI & ML Tools are more popular than Applied AI & ML repositories. Tensorflow [19] is the most popular tool, and has over 100,000 more stars than Tesseract [18], the most popular Applied AI & ML repository.

Our findings show the AI & ML community has unique characteristics that should be accounted for in future research (Section 6): (1) more research and support is needed for Python as the main AI & ML programming language; (2) the significant differences between internal and external contributors in AI & ML projects suggest that empirical studies need to account for contribution types; (3) since a company owns the most AI & ML repositories, many public AI & ML projects on GitHub will have commercial interests and involve paid software developers; and (4) as the most popular AI & ML projects, TensorFlow and Tesseract should be included in any AI & ML-related research; (5) the collaboration study found users collaborate through interactions like discussions across all artifacts, which are not considered in current collaboration studies; (6) several measurements show Applied AI & ML and AI & ML Tool repositories should be treated as related but unique groups, and (7) the measurements for collaboration and autonomy can be applied for groups of repositories or at the individual level, with each scope leading to interesting insights. A supplementary data package containing .csv files of the mined and generated repository data is also provided: https://doi.org/10.5281/zenodo.3722449

This paper is organized as follows. Section 2 describes the data collection and selection criteria for the repositories. Section 3 describes the analysis methods. In Section 4, we present the results based on quantitative measures such as ownership, programming language, timeline, and popularity. In Section 5.1, we discuss AI & ML repositories with respect to collaboration and autonomy. In Section 6, we present the implications of this paper for AI & ML and SE research. We discuss in Section 7 the threats to validity, in Section 8 the related work, and we conclude in Section 9.

2 DATA COLLECTION

To identify projects that apply or develop artificial intelligence or machine-learning software, we deviated from traditional approaches such as topic-modelling that require parsing repository artifacts [30, 34, 43, 44, 46, 48]. These are inefficient when the repository's topic *is* the selection criteria over 'all of GitHub'. Instead, we treated GitHub as a search engine by using the API to curate a list of relevant repository topic labels [25] and then searching for projects with these labels. Additionally, we sampled the rest of GitHub to create a set of Non-AI or ML *Comparison* projects.

Collecting AI & ML Repositories First, the API was queried for repository topic labels related to *artificial intelligence*, *deep learning*, and *machine learning*. Including the search terms, the result was 439 topic labels. The new terms were sub-topics (e.g. *adversarial-machine-learning*), technologies (e.g. *tensorflow*), and techniques (e.g. *natural-language-processing*) related to AI & ML. Next, we searched the API for all repositories that had at least 1 of these labels. 53,427 public repositories had at least 1 of the AI & ML labels in our search set. We collected the metadata returned by the API for each search result.

Distinguishing AI & ML Tools & Applications We also categorized each AI & ML repository as *Applied* or *Tool*. This helped to determine if observations made during analysis were unique to these sub-classes. For example, the Tensorflow project is a well-known AI & ML framework (*Tool*), and the Faceswap [11] project *applies* an AI & ML framework towards solving a problem. To identify Tool repositories we used two approaches. First, a well-known and actively maintained list of AI & ML tools [40] was cross-referenced with our list of repositories. Second, the description of each remaining repository was parsed for terms such as *Tool*, *framework*, *toolkit*, *library*, *'code/models for…'*, etc. Each remaining repository was manually classified based on its GitHub page.

Collecting a Comparison Set To sample the rest of the GitHub repository population, the API was queried for 10,000 repositories updated within the year 2019, sorted by stars. These extra parameters were included because this search space was much larger. Repositories in the query results containing 1 or more of the AI & ML topic tags were removed (but remain in the AI & ML set).

Filtering Our goal was to curate representative samples of *active software projects* (1) applying or developing artificial intelligence and machine learning software and (2) the rest of the repository population. To achieve this, we manually reviewed all the collected metadata to filter the repositories by the following criteria:

(1) **Size**: Must have size greater than 0 (KB)
(2) **Popularity:** Must have ≥5 stars *OR* ≥5 forks
(3) **Activity:** The last commit must have been within 2019
(4) **Data Availability:** Repository data must be accessible via the GitHub API *and* GHTorrent [27]
(5) **Content:** Must be a *software project* and not a tutorial, homework assignment, coding challenge, 'resource' storage, or collection of model files/code samples

This criteria was adapted from best practices [28, 35, 41] to remove inactive, unused, and non-software repositories. The criteria for popularity and size are purposefully lax to ensure the study represents the whole community and not just the 'top' repositories. To verify the *Content* criteria, each repository's name and description were manually reviewed. If this was not sufficient, the repository's GitHub page was inspected.

Data Summary After collecting and filtering both repository samples, the study proceeded with 5,224 repositories applying (4,524) or developing (700) artificial intelligence and machine learning software, and a comparative set of 4,101 repositories. We feel that this procedure resulted in representative samples that allowed us to characterize and differentiate AI & ML software development on GitHub. In Table 1, the number of repositories in the data set per class (Applied, Tool, Comparison) are shown. These counts are also subdivided by owner type as some analyses compare user and organization-owned repositories in each class.

Data for each repository was collected from the GitHub API and the (June 2019) GHTorrent database. From GHTorrent we collected

Table 1: Summary of Repository Data Sets

Owner Type/ Repository Type	**Total**	Organization	User
Applied Use of AI & ML	4,524	1,273	3,253
AI & ML Tool	700	344	360
Comparison	4,101	1,346	2,755
Total	**9,325**	2,963	6,368

detailed information about repository *artifacts*: contributors, issues, commits, and pull requests.

3 METHODS OF ANALYSIS

Repositories using and applying machine learning & artificial intelligence have not previously been studied as a unique community within GitHub's ecosystem. Our analysis strategy was designed to provide novel insights into the scope, scale, and character of these repositories and how they are developed. To contextualize findings and highlight unique properties of this community, we include data from our comparison set of repositories unrelated to artificial intelligence or machine learning.

3.1 Characterization

Analysis started by using the repository data to *define* GitHub's AI & ML community, inspired by the "State of the Octoverse" [26] reports that characterize development on the platform. We establish the history of AI & ML development on GitHub, quantify characteristics (e.g. languages), and identify trends in contribution, popularity and growth. For example, we reviewed repository creation dates and found the oldest AI & ML repository was created in 2009. To contextualize the growth of this community over time, we measured the proportion of new repositories of each type created annually. Starting in 2017, more AI & ML repositories were created annually than projects in our Comparison set. When it is significant, we also highlight trends based on ownership. The "State of the ML-verse" report is detailed in Section 4.

3.2 Workflow: Collaboration & Autonomy

To study development workflow, we have designed a quantitative approach to measure *collaboration* and *autonomy* within a repository. The decision to measure these factors has two motivations. The first is that they reflect the *shared repository* and *fork-and-pull* workflows common in distributed open source development. If most repository contributors have direct commit access (high autonomy) it is likely a shared repository; if they submit pull requests to be merged by others , low autonomy) it is likely fork-and-pull. Second, recent works have advocated for changes to how *productivity* in software development is measured because traditional metrics (e.g. lines of code) are scoped to *individual* developers, which can be inaccurate or harmful [47]. However, team collaboration and autonomy have been identified in recent studies as factors that influence developer's *perceptions* of productivity, and can be measured at the team level [34, 49, 53]. These factors are usually measured with qualitative methods (e.g. interviews) [34, 49] and have not, to our knowledge, previously been measured using repository data.

Our measurement approach calculates repository (team)-level metrics for each factor using only metadata from commits, issues, and pull requests. To make inferences for the AI & ML community as a whole, we aggregated the results from each repository.

Measure Collaboration Through User-to-User Interactions To quantitatively measure *how collaborative* a development team is, we must first acknowledge that *commits are not the only way two users collaborate within a repository*. Consider all the actions and roles related to a single artifact: pull requests, issues, and commits can have authors, maintainers, commentators, etc. It was crucial to define all possible interaction types between users within an artifact. The **5 user-to-user collaborative interactions** are:

(1) **Contribution:** The (distinct) author & committer of a single commit.
(2) **Maintenance:** Two users that initiate an event (e.g. close) for the same issue or pull request (except comments), and neither user is the reporter or opener of the artifact.
(3) **Process:** The reporter or opener of an issue/pull request and another user who initiates a maintenance event.
(4) **Review:** A commentator on a commit, issue, or pull request and it's author/reporter/opener.
(5) **Discussion:** Two commentators for a commit, issue, or pull request for which neither is the author/reporter/opener.

We developed an automated script to parse the action and history data from GHTorrent for every pull request, commit, and issue in our data set and create a record for each instance of the 5 collaborative interactions. An **interaction record** includes the interaction & artifact types and the unique identifiers for the project, artifact, and user IDs.

In the context of these interactions, we developed measurements for two **collaboration perspectives**:

(1) **Users per Artifact**: Total unique users who had collaborative interactions for each artifact.
(2) **Interactions per Artifact**: Total interactions per type for each artifact.

For individual repositories and repository groups, these measurements can be used to identify patterns such as the most common interactions for each artifact and which artifacts have the highest concentration of unique users.

Measure Autonomy Through User Actions on Artifacts Beecham et al. defined autonomy as "[The] freedom to carry out tasks, allowing roles to evolve..." [24]. In distributed development environments like GitHub, a user's freedom and tasks are dependent on their role & permissions within a repository and the repository's development model. Repositories using the *fork and pull model* [29] require external contributors to submit pull requests that are reviewed & merged by a user with write access to the main repository. In this case, the external contributor is *dependent* on the "core team" user. In the *shared repository* [29] model, contributors have write access to the repository and commit their own code. When a contributor can author *and* merge/commit their own changes, they are working *autonomously*. To scale this idea to the team level, in an **autonomous team** a *majority* of contributors have push access and/or the freedom to merge their own pull requests. Measuring team autonomy could potentially suggest which development model is being used.

An automated, rule-based approach was applied to record every *user-to-artifact* interaction from all pull requests, commits, and issues in each repository. This data was collected from GHTorrent. All possible actions (e.g. merge, commit, subscribe) for each artifact were accounted for. A **user action record** includes the artifact type, artifact & user IDs, the action (e.g. 'opened'), and the user's role (e.g. 'reporter') in the action. Each user's records were then parsed to count how many times they had each role. For example, a user's commit-based actions were used to count their commits authored, commits self-pushed, and commits pushed by others. The count data for each user was used to label them with **user types**:

(1) **Maintainer**: A user who has merged or closed pull requests and/or issues which they did not open.
(2) **Autonomous Contributor**: A majority of the users' commits were also committed by that user, and/or a majority of their pull requests were self-merged.
(3) **Dependent Contributor**: A majority of the users' commits were committed by another user, and/or a majority of their pull requests were merged/closed by another user.

Continuing the previous example, a user whose count of self-committed commits is higher than the count of their commits pushed by someone else, is an *autonomous* contributor. A user can be a maintainer *and* a contributor, but they cannot be an autonomous and dependent contributor. User action records were also used to identify *internal* and *external* users; see Section 4.

To determine team autonomy, **user type proportions** (% of users who are maintainers, autonomous, and dependent) were computed for each repository. These values can be used to easily recognize *autonomous* and *dependent* development teams. The proportion of maintainers also provides insights into users who manage the repository but may not commit code. To examine trends within each repository type, we looked at the distributions of these metrics.

4 THE STATE OF THE ML-VERSE

A Decade of AI & ML Development: Origins & Growth Trends To establish a timeline of AI & ML development, we looked at how many repositories of each type were created annually. All repositories studied were created between January 2008 and May 2019. Figure 1 shows the annual type (Applied, Tool, or Comparison) distribution for new repositories. The oldest (still-active) AI & ML repositories were created in 2009: 2 Tools and 5 Applied use projects. The honor of oldest project goes to *cilib* [9], a Scala 'Computational Intelligence Library', and the most well-known repository created this year was the Python *Natural Language Toolkit (NLTK)* [5]. Most of the 2009 repositories (4) are owned by Organizations.

For the next 4 years (2010-2013), less than 10% of new repositories were related to artificial intelligence or machine learning. This changed in 2014, where 17.66% of new repositories were either Tools (42) or Applications of (85) AI & ML. A dramatic "boom" occurred in 2017 with over 1,000 new AI & ML repositories: 1,066 Applied & 179 Tools. From 2017 onward, more AI & ML repositories are created annually than our comparison repositories, and more Applied projects are created annually than Tools. When the data is filtered by owner type, it is revealed that the 'boom' (more AI & ML projects created than Comparison) happened earlier for organizations: in 2016 only 49.07% of organization-owned repositories were in the Comparison group. Also, users create more repositories per year than Organizations.

Takeaways for Origins & Growth: The oldest active AI & ML repository (Cilib) was created in 2009. Since 2012, the annual proportion of new repositories related to AI & ML gradually rose, until a 'boom' in 2017 started a trend of new AI & ML repositories outnumbering our comparison repositories. More Applications of AI & ML are created annually than Tools. For Organization-owned repositories, the 'boom' occurred a year earlier, but users create more repositories each year.

Baskets of Eggs: Repository Ownership Most of the repositories used in this analysis (68.25%) are owned by users. This was also true for individual repository types as shown in Table 1. 403 accounts in our data set (4.32%) own at least 2 repositories and 42 own at least 5. Users make up the majority of these accounts (57%), and as shown in Table 2, 60% of accounts with 10 or more repositories are owned by users.

Table 2: Top 5 Accounts with Multiple AI & ML Repositories

Owner	Owner Type	Repositories
IBM	Organization	61
benedekrozemberczki	user	26
Microsoft	Organization	23
Stick-To	user	17
proycon	user	10

There are 2 organization accounts representing industry software companies: IBM and Microsoft. Accounts with multiple repositories tend to have a lot of Applied projects. All of IBM's repositories are *applied* uses of AI & ML, but only 43% of Microsoft's repositories are Applied. The 3 *users* with the most AI & ML repositories are graduate-level computer science students: each has more than 50% Applied projects.

Takaways for Repository Ownership: Users own the majority (79.1%) of Applied AI & ML repositories, but Organizations own more (51.43%) of the AI & ML Tools. More users own multiple repositories, but an Organization (IBM) owns the *most* (61) AI & ML repositories. The top 3 users with multiple repositories were graduate students, and Applied repositories were the majority owned by the overall top 5 accounts.

Roll Call: Internal & External Users per Repository To measure user participation in repositories, we classified them into 2 groups based on their participation within a repository. Figure 2 shows the distribution (outliers omitted) of the unique **internal users** per repository, who participate by authoring & pushing commits, maintaining the repository and artifacts (e.g. closing/merging pull requests), and leaving comments. We examine different types of contributions in our collaboration and autonomy analysis in Sections 5.1& 5.2. Applied AI & ML and Comparison repositories had a median of 2 internal users, but AI & ML Tools had a median of 4. Tensorflow [19] (Tool) had the most contributing users (1,690) of all repositories. The Applied repository with the most contributors was the Magic engine mage [13] (203), and CoreFX [38], a

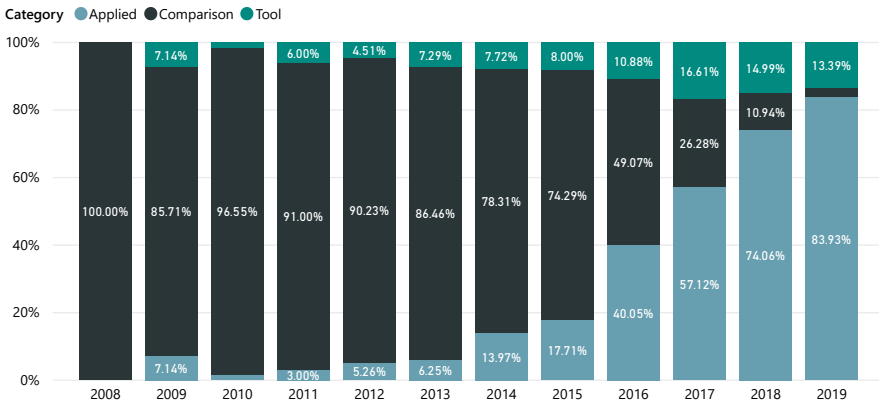

Figure 1: Annual Ratios of New AI & ML Application, Tool, & Comparison Repositories From January 2008 to May 2019

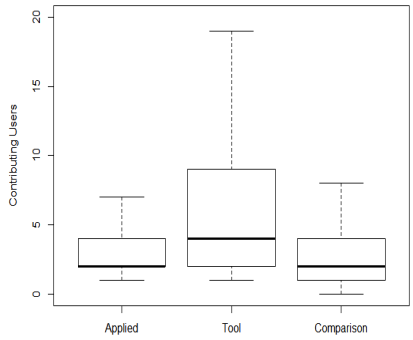

Figure 2: Internal (Contributing) Users (outliers omitted)

set of core classes for Microsoft .NET, had the most (814) of the Comparison repositories.

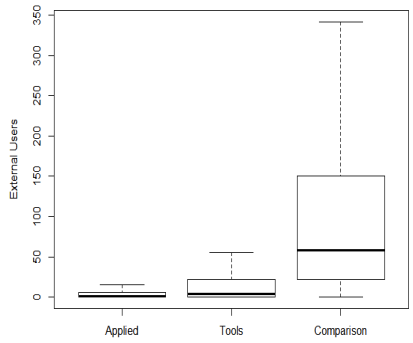

Figure 3: External Users (outliers omitted)

Figure 3 shows the distribution (outliers omitted) of the unique **external users** per repository, who *only* participate by leaving comments and opening issues. An example of an external user is someone who uses the software being designed, and opens/comments on an issue about a bug they found. The median values for number of external users were: Comparison (58), AI & ML Tools (4), and Applied AI & ML (2). A Kruskal-Wallis test was applied to the internal and external user distributions, which found ($p < 0.001$) they were not equal. VSCode [39] was the Comparison project with the most external users (46,559), followed by the AI & ML Tool Tensorflow (20,868). The Applied project Openkore [15], an automation client for Ragnarok Online, had 2,339 external users.

> **Takeaways for Contribution:** AI & ML Tools had more internal (contributing) users than Applied AI & ML *and* Comparison repositories. However, Comparison repositories have the most external users, who leave comments and open issues but do not contribute to the project.

Unique Commit Authors For a closer look at contribution, we measured unique commit *authors* per repository. The distributions for each repository type are shown in Figure 4. The median values

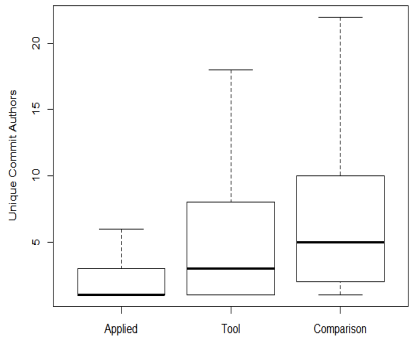

Figure 4: Unique Commit Authors (outliers omitted)

for unique commit authors are: Comparison (5), AI & ML Tools (3), and Applied AI & ML (1). A Kruskal-Wallis test confirmed ($p < 0.001$) that the distributions are not equal across groups.

The top 4 repositories with the most unique commit authors were all AI & ML Tools: Tensorflow [19] with 1,610 followed by scikit-learn [17] (1,539), PyTorch [16] (1,098), and Julia [12](802). Fifth on the list was CoreFX [38] (771), a Comparison repository. Mage [13], an engine for playing the Magic card game online, was the repository with the most unique commit authors (194) for the Applied AI & ML group. Tensorflow, CoreFX, and Mage were also the projects with the most internal (contributing) users for each repository type.

Takeaway for Unique Commit Authors: AI & ML repositories have less unique commit authors than the Comparison repositories. However, the top 4 repositories with the *most* unique commit authors were all AI & ML Tools.

Authors vs Committers: Who Has the Power? There are 2 common development models on GitHub: *fork-and-pull*, where contributions are committed by project maintainers via pull requests, and *shared repository* where everyone has commit permissions. A key indicator of which model a repository uses is how many new contributions were pushed to a repository by the author vs another user. For each repository, we looked at the ratio of commits pushed by the author (self-committed) and commits pushed by another user (other-committed).

AI & ML Tools have the highest median contributions that were self-committed (15) *and* other-committed (120). Applied AI & ML had the lowest medians for each (self:6, other: 21). The Comparison repositories had a median of 11 self-committed contributions and a median of 97 other-committed contributions. For all repository types, most contributions are not committed by the author, indicating a *fork-and-pull* development model. Kruskal-Wallis tests were applied and found the distributions of self-committed and other-committed contributions are not the same ($p < 0.001$) for each repository type. In Section 5.2, we use this data to explore repositories with development models that facilitate high contributor *autonomy*.

Takeaway for Development Models: All repository types favor a development model (e.g. fork-and-pull) where commit authors do not push their own contributions to the repository.

Star Power: GitHub's Popularity Contest On GitHub, the popularity of a repository is often measured by how many *stars* it has. In OSS-based research, the number of stars is often used as a filtering metric [6, 28]. Figure 5 shows the distribution (outliers omitted) of stars for all AI & ML repositories, as well as Applied & Tool repositories separately. We exclude our Comparison set from this analysis because we used number of stars as a sort criteria when searching for these repositories but not for the AI & ML data set. As mentioned in Section 2, this was done to optimize sampling quality non-AI & ML repositories from GitHub's massive search space.

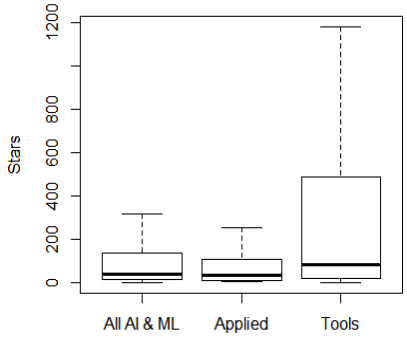

Figure 5: Stars in AI & ML Repositories (outliers omitted)

For all AI & ML repositories, the median number of stars is 37. For Tool repositories the median is 83, more than twice that of Applied repositories (37). However, all of the Applied AI & ML repositories have at least 4 stars, while some of the Tools have 0. As described in Section 2, we included repositories with at least 5 stars *or* 5 forks to include more of the community in our sample. A Mann-Whitney test found the star distributions of the AI & ML Tools and Applied AI & ML projects were different at $p < 0.001$.

Both groups had several extreme outliers, omitted from Figure 5 for readability. In the Tool group, Tensorflow [19] (131,135) was most popular, with over 100,000 *more* stars than the most popular Applied repository tesseract [18] (28,207), a ML-based OCR program. For reference, the most popular Comparison repository in our data set is Microsoft's VSCode [39] with 81,416 stars.

Takeaways for Popularity: AI & ML Tools are more popular than Applied AI & ML projects. Tensorflow was the most popular Tool, and had over 100,000 *more* stars than Tesseract, the most popular Applied AI & ML repository, and almost 50,000 more than the top Comparison repository, VSCode.

Python: The Champion Language for AI & ML Development

Figure 6 shows the 10 most popular languages for AI & ML repositories. The majority of these projects are written in Python (56.8%), followed by Jupyter Notebooks. While not a language, these projects have a unique format blending code with text. We do not make assumptions about the language used *in* the notebooks, because only the declared language for the repository was observed. This trend also occurs for just Applied repositories: 58.5% Python, 18.5% Jupyter. For Tool projects, the 2nd most common language in AI & ML Tool projects was C++. In the Comparison set, the top language was JavaScript (33.2%), followed by Java (14.5%). Python was third, labeled as the primary language for 12.9% of Comparison repositories. We also observed language trends over time using

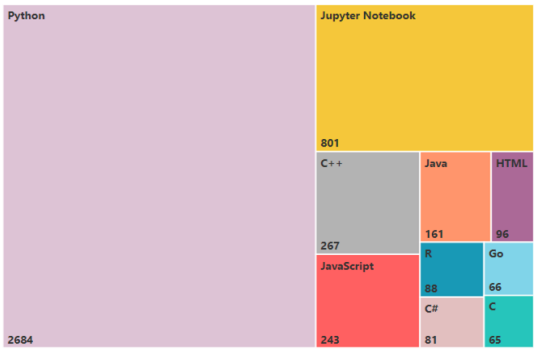

Figure 6: Top 10 Languages for AI & ML Repositories

repository creation date, which is typically when language labels are applied to a repository. Python has been the most common language for AI & ML repositories for the last 8 years. In 2012, the first repository was labeled as Jupyter Notebook. This 'language' gradually gained popularity over the years, tying for second place with JavaScript in 2015 and surpassing it the next year. The #2 language for Applied AI & ML repositories shifted from JavaScript (Java in 2013) to Jupyter Notebooks in 2015. For AI & ML Tool repositories, C++ has been the 2nd most popular language since 2011. In the Comparison set, Ruby was surpassed by JavaScript for most popular in 2010. Python replaced Ruby for second most-frequent language in 2012.

Takeaways for Repository Language: Since 2011, the majority of AI & ML software was written in **Python**. C++ was the second-most popular language for Tool repositories, and Jupyter was second for Applied projects. The top 3 languages for our Comparison repositories are JavaScript, Java, & Python.

Hot Topics: Machine Learning, Deep Learning & Python During data collection, we curated a list of GitHub's topic labels related to AI & ML, which we used to identify relevant repositories. Looking at how many repositories had each of the 439 topics in our list reveals the most popular *sub-topics*, languages, and tools. For both Applied and Tool repositories, the top 3 labels were *machine-learning*, *deep-learning*, and *python*. The top 3 labels for our Comparison set correspond to our findings of most popular languages: *Javascript*, *Android*, and *Python*. The *Andriod* label hints that mobile development is a hot topic for non-ML repositories (182) but was only added to 60 of the AI & ML repositories.

Takeaways for Repository Topics: The most popular topic labels for AI & ML repositories represent sub-topics (*deep learning*), languages (*python*), and tools (*tensorflow*). In the Comparison set, the most common topics were language (*JavaScript, Python* and platform (*Android*)-based.

5 COLLABORATION & AUTONOMY

As described in Section 3.2, we developed several measurements for examining *autonomy* and *collaboration* within a repository. These measurements provide contextual insights into the development workflow of AI & ML software on GitHub, and these values have also been identified as factors influencing developers perceptions of productivity [34, 47, 49, 53].

Table 3: Artifacts per Repository

Artifact	Category	Median
Commits	Applied AI & ML	8
	AI & ML Tools	22
	Comparison	20
Issues	Applied AI & ML	4
	AI & ML Tools	27
	Comparison	12
Pull Requests	Applied AI & ML	3
	AI & ML Tools	3
	Comparison	5

5.1 Collaboration

Metadata for every pull request, issue & commit in all projects was analyzed to record instances of 5 unique interactions between users: contribution, maintenance, process, review, and discussion. The formal definition for each interaction is provided in Section 3.2. For these measurements, we include interactions from internal *and* external users, as defined in Section 4.

This analysis relies heavily on repositories having multiple contributors, issues, commits, and pull requests. Our goal was to examine the entire community, so our filtering requirements for these artifacts were loose. The consequence of this is that 555 repositories could not be included in this study. Thus, the following analysis is based on the 8,770 projects in our data set with sufficient artifacts. Table 4 shows the categorical breakdown of this subset. For perspective on the distribution of each artifact per repository group, Table 3 shows the median number of each artifact per repository.

Table 4: Dataset for Collaboration Study

Owner Type/ Repository Type	Total	Organization	User
Applied Use of AI & ML	4,143	1,213	2,930
AI & ML Tool	660	332	328
Comparison	3,967	1,324	2,643
Total	**8,770**	2,869	5,901

Users per Artifact Our first measurement calculates the total number of *unique* users who had at least 1 collaborative interaction with any of each repository's commits, issues, or pull requests. For example, at the time of data collection (June 2019), 22,157 unique users had interacted with the 17,989 issues in the Tensorflow [19] repository. Figure 7 shows the distributions of this metric for each repository type.

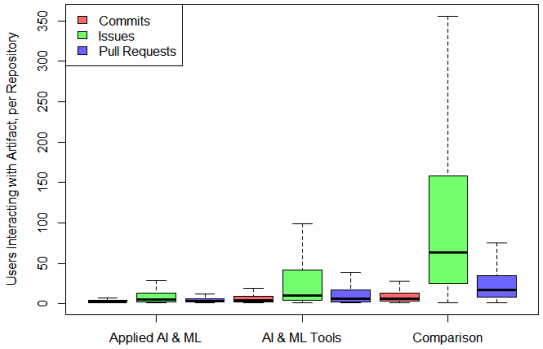

Figure 7: Users per Artifact Type (outliers omitted)

The ranking of users per artifact follows the same trend for each repository group: issues, pull requests, commits. For Applied AI & ML projects the median users for issues is 5, 3 for pull requests, and 2 users per commit. In AI & ML Tool repositories, issues had a median of 10 users, pull requests had 6, and a median of 4 users collaborated on commits. The median values in the Comparison set are quite high: issues have a median of 63 users, pull requests have median 17, and commits have median 6. We applied the Kruskal-Wallis test for the 12 combinations of repository & artifact type (e.g. 'Tool Commits') and found the distributions of users to be different ($p < 0.001$). A Dunn's test with bonferroni adjustment showed that the only pair of user-artifact distributions that were *not* significantly different ($p < 0.001$) are pull requests in AI & ML Tools and commits in Comparison repositories.

Summary: Issues have the most unique collaborating users for all repository types. All artifacts in AI & ML Tool repositories have more collaborating users than Applied AI & ML projects, and overall Comparison repositories had the most users per artifact.

Interactions per Artifact This measurement is designed to identify which collaborative interaction is most common for each artifact. The artifact-interaction distributions for each repository type

are shown in Figures 8- 9. In these figures, the gaps within artifacts represent inapplicable interactions. Specifically, issues and pull requests will not have *contribution* interactions, and commits do not have *process* or *maintenance* interactions. All artifacts can have *discussion* and *review* interactions.

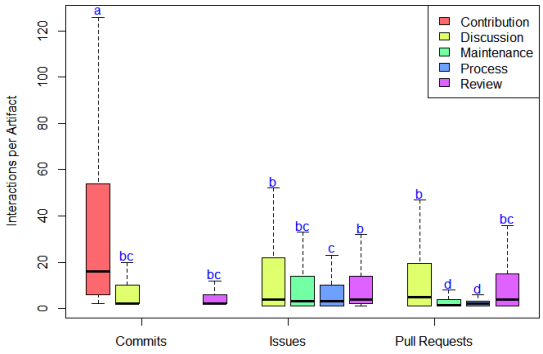

Figure 8: Collaborative Interactions per Artifact in Applied AI & ML Repositories (outliers omitted)

Artifact-interaction distributions for the Applied AI & ML repositories are shown in Figure 8. For commits, the median contributions is 16, and 2 for discussion & review interactions. Issues have a median of 4 discussion and review interactions and 3 maintenance and process interactions. Pull requests have a median of 5 discussions, 1.5 maintenance interactions, 4 reviews, and 2 process interactions. Thus, the most common artifact-interaction group for Applied AI & ML repositories are *commit contributions*, followed by *pull request discussions*. Kruskal-Wallis tests applied to all the artifact-interaction distributions indicated ($p < 0.001$) different distributions. A Dunn test (bonferroni adjustment) found commit comments are the only artifact-interaction group with a significantly unique distribution. Figure 8 shows the significance letters for each group above its boxplot; distributions that do not share a letter are different.

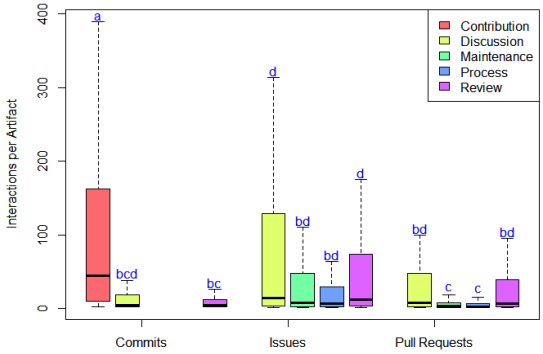

Figure 9: Collaborative Interactions per Artifact in AI & ML Tool Repositories (outliers omitted)

For AI & ML Tools, artifact-interaction distributions are shown in Figure 9. Commits have medians of 44 contribution interactions and 4 discussion and review interactions. Issues in AI & ML Tool repositories have medians of 14 discussions, 12 reviews, 4 process interactions and median 2 maintenance interactions. For pull requests the median discussions is 8, review interactions have a median of 7, maintenance interactions have median 3, and finally a median of 2 process interactions. Thus, the most common artifact-interaction group for AI & ML Tool repositories are *commit contributions*, followed by *issue discussions*. A Kruskal-Wallis test found evidence ($p < 0.001$) of different distributions, so Dunn's test (bonferroni adjustment) was applied. Commit contributions were also the only significantly different distribution for AI & ML Tools. Each distribution pair in Figure 9 with different letters are significantly different.

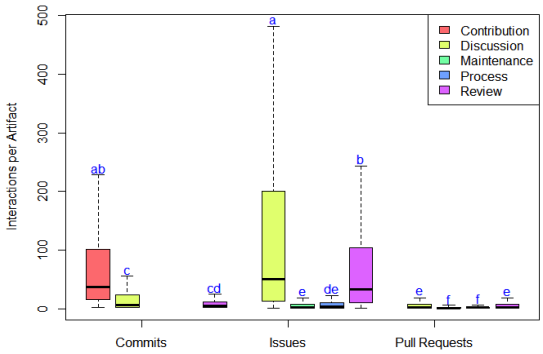

Figure 10: Collaborative Interactions per Artifact in Comparison Repositories (outliers omitted)

Figure 10 shows the 5 interaction distributions for the Comparison repositories. For commits, the median number of contributions interactions is 38; the median is 6 for discussion and 5 for review interactions. Issues have a median of 51 discussions, 34 review interactions, 4 process and 2 maintenance interactions. Finally, Comparison pull requests have a median of 3 discussion and review interactions, 2 process interactions, and 1 maintenance interaction. Thus, the most common artifact-interaction group for Comparison repositories are *issue discussions*, followed by *commit contributions*. Kruskal-Wallis tests found the distributions of each artifact-interaction pair for this group to be different ($p < 0.001$). We then applied Dunn's test and found that *no* artifact-interaction group has a unique distribution. However, there are *subsets* of significantly different distributions as shown by the significance letters in Figure 10.

Takeaways for Collaborative Interactions per Artifact: In all groups, the most common interaction for issues and pull requests is *discussions*. The most common interaction for AI & ML repositories are *commit contributions*. AI & ML Tool repositories have more (of most) interactions per artifact type than Applied AI & ML projects. In the Comparison repositories, the most common artifact-interaction was issue discussions. AI & ML Tools have the most commit contributions and pull request discussion & pull request review interactions overall.

5.2 Autonomy

To measure team autonomy, we parsed each repository's history and event data and recorded all user-to-artifact interactions (see Section 3.2). These records were used to identify three types of users: maintainers, autonomous contributors, and dependent contributors. In this section, we report the distributions of user-type proportions and discuss the number of autonomous teams for each group.

User Type Proportions Once each contributing user's type is determined, team autonomy can be measured by calculating user

type *proportions*: # users of type *n* to total contributing users. These values can be used to compare repositories or observe trends within groups because they normalize for variations in total users. Figure 11 shows the *distributions of user type proportions* per group. Both AI & ML groups had a median of 33% autonomous contributors. Tools had more dependent contributors (median 25%) than the Applied group (median 14%), but Applied AI & ML projects had more maintainers, with median 50%. The Comparison group had the lowest median proportion of autonomous contributors, (25%) but the highest median proportion of dependent contributors (33%). This group also had a median 38% maintainers, lower than both AI & ML groups. Kruskall-Wallis tests showed the distributions of each user type were different for each group at $p < 0.001$.

460 repositories had 100% autonomous contributors: 84.78% were Applied AI & ML, 7.83% were Tools, and 7.39% were non-AI & ML. However, these all-autonomous projects had median of 1 contributor, mean 1.29, and max 12. The 'most autonomous' repositories with at least 10 contributors per group are: (Comparison) NEKit [14], 98%, 163 contributors; (AI & ML Tool) cilib [9], 92%, 60 contributors; (Applied AI & ML) LEGALST-190 [10], 89%, 18 contributors. We also used the proportion measurements to count the number of autonomous teams in each group. A total of 2,637 repositories were found to have autonomous teams. The Applied AI & ML group had the most (1,777) autonomous teams; the Comparison group had 616 and AI & ML Tools had 244.

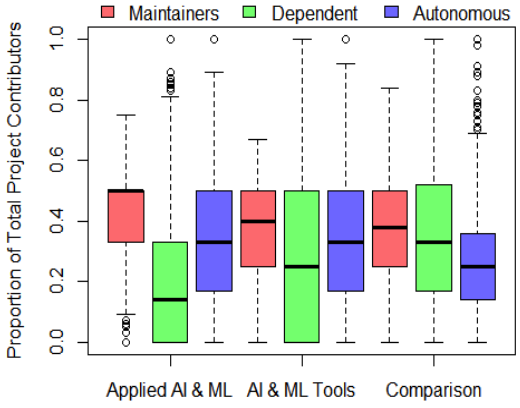

Figure 11: Distribution of User Type Proportions

Takeaways for User Types: Based on the proportion of autonomous-to-total contributors per repository, most development teams in all groups are *not* autonomous, implying *fork-and-pull* development. AI & ML repositories have more autonomous contributors than non-AI & ML, but Applied projects have a higher proportion of maintainers than the Tools and the lowest proportion of dependent contributors. The Tools group had the least autonomous teams the and Applied AI & ML group had the most.

6 IMPLICATIONS FOR SE RESEARCH

Our study has several implications for SE research. The findings help quantify and highlight challenges and opportunities for SE researchers. We present some of the important implications below.

Our study showed that the majority of AI & ML software were written in Python. The SE research community should aim to focus new research on Python projects which have not gotten as much attention as needed to keep up with the "AI & ML boom".

We observed that individuals *and* organizations (e.g. IBM) own AI & ML projects. Users own the majority (79.1%) of Applied AI & ML repositories, but organizations own more than (51.43%) of the AI & ML Tools. SE researchers should account for this in their studies as organizations have different motivations and expectations for the projects they support. The ownership type could bias any analysis when combined with open-source projects driven by volunteers.

Internal & external contributors have significantly different characteristics in AI & ML projects. SE researchers must focus on the right populations in order to target the right set of individuals when they perform qualitative analysis and survey/interviews. As the populations are different mixing the two or picking the wrong population could to lead to biases in the results.

Tensorflow [19] is the most popular tool and has over 100,000 more stars than Tesseract [18]. For SE researchers looking to investigate AI & ML Tools, Tensorflow should serve as a baseline given its popularity and significance in terms of users and contributions. Any study of AI & ML tools not involving Tensorflow will miss a large and important part of the AI & ML population.

Collaboration in a repository is multi-dimensional. Although *commit contributions* were the top collaborative interaction in AI & ML repositories, *discussion* and *review* interactions were also very common. Since the autonomy measurements indicate that a large proportion of contributors are *dependent* on the actions of other users in pull requests and commits, studies should not consider commit data sufficient to measure collaboration.

The aggregated autonomy metric was used to examine the proportion of each repository type that have autonomous teams. We found that most AI & ML repositories had more dependent contributors, indicating use of the *fork-and-pull* model. However, the Applied AI & ML group had the most autonomous teams, indicating more repositories in this group use a *shared repository* model. This and several other observations show that AI & ML Tools and Applied AI & ML have distinct characteristics that future studies of AI & ML repositories should account for.

Finally, at the individual repository level, the collaboration metrics can be applied to *specific* artifacts or artifact types. At this level, this information can be combined with *fine-grained* details to identify things like the *relationship between issue labels and interaction types* which might reveal if the team collaborates more or less on specific tasks, bugs, etc. For autonomy, large repository teams could use these measurements to prevent imbalances in proportions of dependent & autonomous contributors and maintainers, which might lead to bottlenecks like unreviewed pull requests. We encourage further studies on the applications of such metrics, especially how they relate to developers perceptions of productivity.

7 THREATS TO VALIDITY

We identify the following threats to validity.

Construct validity refers to the extent the operational measures in this paper really represent what we intended to measure. To identify the AI & ML projects we relied on the topic labels

that developers manually assign to repositories. This may lead to false positives and negatives, some projects may have not been assigned the correct labels and as a result incorrectly included or excluded from the AI & ML project sample. Since we followed best practices [28, 35, 41] and only considered active, used software repositories, we expect that there is only limited amount of mislabeling. In addition, to distinguish between AI & ML Applications and Tools we relied on keyword search and manually classified repositories. Our measures of collaboration and autonomy are based on activity on issues, pull requests, and commits GitHub. However, not all developer activity is recorded in software repositories [2] and as a result there may be aspects of collaboration and autonomy that are not captures by the measures in this paper.

External validity is concerned with to what extent the findings of this paper can be generalized. We focused the analysis on public repositories on GitHub. The results might differ for private GitHub repositories, other code hosting sites, or projects in companies. While we expect to see some differences, we also expect many similarities, since some of the most popular AI & ML tools (TensorFlow) are hosted on GitHub.

8 RELATED WORK

GitHub is one of the largest hosts of software with 40+ million users and 100 million repositories as of September 2019 [26]. GitHub has been frequently studied in the context of software development. Prominent examples are studies of programming languages and code quality [4, 45, 46] and bug resolution characteristics [56]. GitHub has also been studied to understand social coding [22] and development techniques such as continuous integration [52]. For a more comprehensive overview of studies related to GitHub, we refer to the systematic mapping study by Cosentino et al. [20]. Research has been facilitated by GHTorrent project, which provides researchers a continuous stream of activity data from GitHub [27].

In this paper, we studied GitHub with a focus on projects that are related to artificial intelligence and machine learning (AI & ML). After the the boom of AI & ML applications in recent years, several papers have focused on how software development practices has changed for machine learning systems. Wan et al. [54] conducted a mixed-methods study with 14 interviewees and 342 survey respondents to identify differences between AI & ML and other software development. They found that exploratory requirements elicitation and iterative processes are more common in ML development, ML systems are more complex, and a stronger demand for unique solutions & ideas in ML development. Amershi et al. [1] identified three differences through a mixed-methods study with 14 interviewees and 551 survey respondents at Microsoft: (1) discovering, managing, and versioning the data needed for machine learning applications is much more complex and difficult, (2) model customization and model reuse require different skills than are typically found in software teams, and (3) AI components are more difficult to handle as distinct modules than traditional software components. Other work on understanding ML-based development focused on identifying design patterns for machine learning systems [55] or what developers ask about ML libraries on Stack Overflow [33].

Several papers have focused on understanding the *challenges* that software developers face with AI & ML applications. Kim et al. [36] identified several challenges for data scientists in software teams. The challenges included data quality, data availability, data preparation, scale, machine learning, and working with non-experts. Zhang et al. [58] identified five main challenges when working with deep learning systems based on an analysis of Stack Overflow questions: API misuse, incorrect hyper parameter selection, GPU computation, static graph computation, and limited debugging and profiling support. Nascimento et al. [21] identified the following challenges using interviews and focus groups: identifying the clients' business metrics, lack of a defined development process, and designing the database structure Belani et al. [3] discussed Challenges in Building AI-Based Complex Systems from a requirements engineering perspective. Lwakatare et al. [37] proposed a taxonomy of software engineering challenges for machine learning systems along five evolution stages of ML component use (from prototyping to autonomous components requiring minimum human intervention).

A prominent topic of research have been *bugs* in AI & ML systems. Research has focused on creating taxonomies of faults, root causes, and symptoms by analyzing data from Stack Overflow, issue tracking systems, bug fix commits, and developer interviews for a wide range of AI & ML tools and frameworks such Caffe, Keras, Tensorflow, Theano, and Torch [31, 32, 50, 51, 59]. Zhang et al. [57] presented the results of a comprehensive survey of 144 papers of machine learning testing research.

In this paper, we characterize a large sample of AI & ML projects with a focus on collaboration & autonomy because they have emerged as two important aspects of software engineering productivity. *Collaboration* has been extensively studied in software development. Most prominently, Cataldo, Herbsleb and colleagues investigated coordination requirements and socio-technical congruence [7, 8]. Other examples of studies on collaboration in GitHub include the work by Dabbish et al. [22] on transparency and collaboration in GitHub projects. Beecham et al. [24] defined *autonomy* as the "freedom to carry out tasks, allowing roles to evolve." and identified autonomy as an important factor for motivation. Independent studies of productivity at Google [42] and Microsoft [49] have identified autonomy as one of the three most influential factors for productivity. In this paper, we quantify the degree of collaboration and autonomy and compare across three populations of GitHub projects (Applied AI & ML; AI & ML Tools; non-AI & ML projects).

9 CONCLUSIONS

As a result of the AI boom over the past few years, artificial intelligence and machine learning are now used by millions of software developers. Hence, it is important for the SE community to understand the differences to traditional software development. In this paper, we analyzed the state of "ML-universe" and compared 5,224 AI & ML repositories to 4,101 other repositories on GitHub. Our analysis focused on aspects such as origins, growth, ownership, contributors, popularity, collaboration, and autonomy. Among other findings (Section 6), we found that Python is used by most AI & ML projects, yet little academic research is focused on the Python language. Any AI & ML research in software engineering should also consider the TensorFlow and Tesseract projects because they were the most popular. This is just the beginning. AI & ML will require significant research efforts to address in the future. A supplementary data package containing .csv files of the mined and generated repository data is also provided: https://doi.org/10.5281/zenodo.3722449

REFERENCES

[1] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: a case study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE SEIP)*. IEEE Press, Piscataway, NJ, USA, 291–300.

[2] Jorge Aranda and Gina Venolia. 2009. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proceedings of the 31st international conference on software engineering*. IEEE Computer Society, IEEE, New York,NY, 298–308.

[3] H. Belani, M. Vukovic, and Ž. Car. 2019. Requirements Engineering Challenges in Building AI-Based Complex Systems. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, New York,NY, 252–255. https://doi.org/10.1109/REW.2019.00051

[4] Emery D. Berger, Celeste Hollenbeck, Petr Maj, Olga Vitek, and Jan Vitek. 2019. On the Impact of Programming Languages on Code Quality: A Reproduction Study. *ACM Trans. Program. Lang. Syst.* 41, 4 (2019), 21:1–21:24. https://doi.org/10.1145/3340571

[5] Steven Bird, Edward Loper, and Ewan Klein. 2009. NLTK: Natural Language Toolkit. https://github.com/nltk/nltk.

[6] H. Borges, A. Hora, and M. T. Valente. 2016. Understanding the Factors That Impact the Popularity of GitHub Repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, New York,NY,USA, 334–344. https://doi.org/10.1109/ICSME.2016.31

[7] Marcelo Cataldo, James D. Herbsleb, and Kathleen M. Carley. 2008. Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/1414004.1414008

[8] Marcelo Cataldo, Patrick A. Wagstrom, James D. Herbsleb, and Kathleen M. Carley. 2006. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06)*. Association for Computing Machinery, New York, NY, USA, 353–362. https://doi.org/10.1145/1180875.1180929

[9] Cilib Development Community. 2009. Cilib: Typesafe, purely functional Computational Intelligence. https://github.com/cirg-up/cilib.

[10] DS-Modules Development Community. 2017. Data, Prediction, and Law. https://github.com/ds-modules/LEGALST-190.

[11] Faceswap Development Community. 2017. Faceswap: Deepfakes Software For All. https://github.com/deepfakes/faceswap.

[12] JuliaLang Development Community. 2011. The Julia Language: A fresh approach to technical computing. https://github.com/JuliaLang/julia.

[13] Mage Development Community. 2012. XMage: Magic, Another Game Engine. https://github.com/magefree/mage.

[14] NEKit Development Community. 2016. A toolkit for Network Extension Framework. https://github.com/zhuhaow/NEKit.

[15] OpenKore Development Community. 2016. A free/open source client and automation tool for Ragnarok Online. https://github.com/OpenKore/openkore.

[16] PyTorch Development Community. 2016. PyTorch:Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://github.com/pytorch/pytorch.

[17] SciKit-Learn Development Community. 2010. SciKit-Learn:Tensors and Dynamic neural networks in Python with strong GPU acceleration. https://github.com/scikit-learn/scikit-learn.

[18] Tesseract Development Community. 2014. Tesseract Open Source OCR Engine. https://github.com/tesseract-ocr/tesseract.

[19] Tensorflow Development Community. 2015. Tensorflow: An Open Source Machine Learning Framework for Everyone. https://github.com/tensorflow/tensorflow.

[20] Valerio Cosentino, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2017. A Systematic Mapping Study of Software Development With GitHub. *IEEE Access* 5 (2017), 7173–7192. https://doi.org/10.1109/ACCESS.2017.2682323

[21] E. d. S. Nascimento, I. Ahmed, E. Oliveira, M. P. Palheta, I. Steinmacher, and T. Conte. 2019. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, New York,NY, 1–6. https://doi.org/10.1109/ESEM.2019.8870157

[22] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, New York City, NY, 1277–1286.

[23] Evans Data Corporation. 2019. Global Developer Population and Demographic Study. https://evansdata.com/reports/viewRelease.php?reportID=9.

[24] A. C. C. França, T. B. Gouveia, P. C. F. Santos, C. A. Santana, and F. Q. B. da Silva. 2011. Motivation in software engineering: A systematic review update. In *15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011)*. IEEE, New York, NY, USA, 154–163. https://doi.org/10.1049/ic.2011.0019

[25] Shay Frendt. 2017. Introducing Topics. https://github.blog/2017-01-31-introducing-topics/

[26] GitHub. 2019. The State of the Octoverse. https://octoverse.github.com/.

[27] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE Press, Piscataway, NJ, USA, 233–236. http://dl.acm.org/citation.cfm?id=2487085.2487132

[28] G. Gousios and D. Spinellis. 2017. Mining Software Engineering Data from GitHub. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, Buenos Aires, Argentina, 501–502. https://doi.org/10.1109/ICSE-C.2017.164

[29] GitHub Help. 2019. About Collaborative Development Models. https://help.github.com/en/articles/about-collaborative-development-models

[30] Abram Hindle, Neil A. Ernst, Michael W. Godfrey, and John Mylopoulos. 2011. Automated Topic Naming to Support Cross-project Analysis of Software Maintenance Activities. In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR '11)*. ACM, New York, NY, USA, 163–172. https://doi.org/10.1145/1985441.1985466

[31] Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. 2019. Taxonomy of Real Faults in Deep Learning Systems. arXiv:cs.SE/1910.11015

[32] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A Comprehensive Study on Deep Learning Bug Characteristics. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2019)*. Association for Computing Machinery, New York, NY, USA, 510–520. https://doi.org/10.1145/3338906.3338955

[33] Md Johirul Islam, Hoan Anh Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. What Do Developers Ask About ML Libraries? A Large-scale Study Using Stack Overflow. arXiv:cs.SE/1906.11940

[34] Eirini Kalliamvakou, Daniela Damian, Leif Singer, and Daniel M German. 2014. *The code-centric collaboration perspective: Evidence from github.* Technical Report DCS-352-IR. University of Victoria.

[35] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21, 5 (01 Oct 2016), 2035–2071. https://doi.org/10.1007/s10664-015-9393-5

[36] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Trans. Software Eng.* 44, 11 (2018), 1024–1038. https://doi.org/10.1109/TSE.2017.2754374

[37] Lucy Ellen Lwakatare, Aiswarya Raj, Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2019. A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In *Agile Processes in Software Engineering and Extreme Programming*, Philippe Kruchten, Steven Fraser, and François Coallier (Eds.). Springer International Publishing, Cham, 227–243.

[38] Microsoft. 2014. The foundational class libraries for .NET Core. https://github.com/dotnet/corefx.

[39] Microsoft. 2015. Visual Studio Code. https://github.com/microsoft/vscode.

[40] Joseph Misiti. 2015. Awesome Machine Learning: A curated list of awesome Machine Learning frameworks, libraries and software. https://github.com/josephmisiti/awesome-machine-learning

[41] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. 2017. Curating GitHub for engineered software projects. *Empirical Software Engineering* 22, 6 (01 Dec 2017), 3219–3253. https://doi.org/10.1007/s10664-017-9512-6

[42] E. Murphy-Hill, C. Jaspan, C. Sadowski, D. Shepherd, M. Phillips, C. Winter, A. Knight, E. Smith, and M. Jorde. 2019. What Predicts Software Developers' Productivity? *IEEE Transactions on Software Engineering* null, null (2019), 1–1. https://doi.org/10.1109/TSE.2019.2900308

[43] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshynanyk, and A. De Lucia. 2013. How to effectively use topic models for software engineering tasks? An approach based on Genetic Algorithms. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, New York,NY, 522–531. https://doi.org/10.1109/ICSE.2013.6606598

[44] A. Rastogi and N. Nagappan. 2016. Forking and the Sustainability of the Developer Community Participation – An Empirical Investigation on Outcomes and Reasons. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. IEEE, Osaka,Japan, 102–111. https://doi.org/10.1109/SANER.2016.27

[45] Baishakhi Ray, Daryl Posnett, Premkumar Devanbu, and Vladimir Filkov. 2017. A Large-Scale Study of Programming Languages and Code Quality in GitHub. *Commun. ACM* 60, 10 (Sept. 2017), 91–100. https://doi.org/10.1145/3126905

[46] Baishakhi Ray, Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. 2014. A Large Scale Study of Programming Languages and Code Quality in Github. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014)*. ACM, New York, NY, USA, 155–165. https:

//doi.org/10.1145/2635868.2635922

[47] Caitlin Sadowski and Thomas Zimmermann. 2019. *Rethinking Productivity in Software Engineering*. Apress Open, New York, NY.

[48] Abhishek Sharma, Ferdian Thung, Pavneet Singh Kochhar, Agus Sulistya, and David Lo. 2017. Cataloging GitHub Repositories. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. ACM, New York, NY, USA, 314–319. https://doi.org/10.1145/3084226.3084287

[49] M. Storey, T. Zimmermann, C. Bird, J. Czerwonka, B. Murphy, and E. Kalliamvakou. 2019. Towards a Theory of Software Developer Job Satisfaction and Perceived Productivity. *IEEE Transactions on Software Engineering* null, null (2019), 1–1. https://doi.org/10.1109/TSE.2019.2944354

[50] X. Sun, T. Zhou, G. Li, J. Hu, H. Yang, and B. Li. 2017. An Empirical Study on Real Bugs for Machine Learning Programs. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE Press, Piscataway, NJ, USA, 348–357. https://doi.org/10.1109/APSEC.2017.41

[51] F. Thung, S. Wang, D. Lo, and L. Jiang. 2012. An Empirical Study of Bugs in Machine Learning Systems. In *2012 IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, New York City, NY, 271–280. https://doi.org/10.1109/ISSRE.2012.22

[52] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, New York City, NY, 805–816.

[53] Stefan Wagner and Melanie Ruhe. 2018. A Systematic Review of Productivity Factors in Software Development. arXiv:cs.SE/1801.06475

[54] Z. Wan, X. Xia, D. Lo, and G. C. Murphy. 2019. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering* 0, 0 (2019), 1–1. https://doi.org/10.1109/TSE.2019.2937083 To appear.

[55] H. Washizaki, H. Uchida, F. Khomh, and Y. Guéhéneuc. 2019. Studying Software Engineering Patterns for Designing Machine Learning Systems. In *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. IEEE, New York City, NY, 49–55. https://doi.org/10.1109/IWESEP49350.2019.00017

[56] Jie Zhang, Feng Li, Dan Hao, Meng Wang, Hao Tang, Lu Zhang, and Mark Harman. 2019. A Study of Programming Languages and Their Bug Resolution Characteristics. *IEEE Transactions on Software Engineering* null, null (2019), 1–1.

[57] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine Learning Testing: Survey, Landscapes and Horizons. arXiv:cs.LG/1906.10742 To appear in IEEE Transactions on Software Engineering.

[58] Tianyi Zhang, Cuiyun Gao, Lei Ma, Michael R. Lyu, and Miryung Kim. 2019. An empirical study of common challenges in developing deep learning applications. In *Proceedings of the 30th International Symposium on Software Reliability Engineering, ISSRE 2019*. IEEE, New York,NY, 12.

[59] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. 2018. An Empirical Study on TensorFlow Program Bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018)*. Association for Computing Machinery, New York, NY, USA, 129–140. https://doi.org/10.1145/3213846.3213866