Applying Machine Learning to Gaze Data in Software Development: a Mapping Study

Peng Kuang
peng.kuang@cs.lth.se
Lund University
Sweden

Emma Söderberg
emma.soderberg@cs.lth.se
Lund University
Sweden

Diederick C. Niehorster
diederick_c.niehorster@humlab.lu.se
Lund University
Sweden

Martin Höst
martin.host@cs.lth.se
Lund University
Sweden

ABSTRACT

Eye tracking has been used as part of software engineering and computer science research for a long time, and during this time new techniques for machine learning (ML) have emerged. Some of those techniques are applicable to the analysis of eye-tracking data, and to some extent have been applied. However, there is no structured summary available on which ML techniques are used for analysis in different types of eye-tracking research studies.

In this paper, our objective is to summarize the research literature with respect to the application of ML techniques to gaze data in the field of software engineering. To this end, we have conducted a systematic mapping study, where research articles are identified through a search in academic databases and analyzed qualitatively. After identifying 10 relevant articles, we found that the most common software development activity studied so far with eye-tracking and ML is program comprehension, and Support Vector Machines and Decision Trees are the most commonly used ML techniques. We further report on limitations and challenges reported in the literature and opportunities for future work.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools**; • **Computing methodologies → Machine learning**; • **Human-centered computing → Empirical studies in HCI**; **HCI design and evaluation methods**.

KEYWORDS

eye-tracking, machine learning, software development

ACM Reference Format:
Peng Kuang, Emma Söderberg, Diederick C. Niehorster, and Martin Höst. 2023. Applying Machine Learning to Gaze Data in Software Development: a Mapping Study. In *2023 Symposium on Eye Tracking Research and Applications (ETRA '23), May 30–June 02, 2023, Tubingen, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3588015.3589190

ETRA '23, May 30–June 02, 2023, Tubingen, Germany

ACM ISBN 979-8-4007-0150-4/23/05.
https://doi.org/10.1145/3588015.3589190

1 INTRODUCTION

Both eye tracking (ET) and machine learning (ML) technology have matured a lot in the last decade; eye tracking has reached a point where it is being integrated into consumer laptops[1], and ML has seen significant progress (e.g., AlphaGo [Silver et al. 2017] and ChatGPT [van Dis et al. 2023]). Programming can benefit from these two developments since eye trackers generate a lot of multifaceted data that may reveal a wealth of information about the user and their intentions, while machine learning is needed to distill these high dimensional data into actionable information [Bishop and Nasrabadi 2006]. Specifically, there is an opportunity to utilize ML techniques to both understand programmers' gaze behavior and to explore gaze-assisted developer tools [Kuang et al. 2023].

However, despite the initial development of gaze data sets intended to accelerate research in this area [Al Madi et al. 2021; Bednarik et al. 2020], there are limited guidelines for how to apply machine learning techniques to this data. Existing guidelines for employing eye tracking for studies in the field of software engineering focus on data gathering and metrics [Sharafi et al. 2020], but have little to say about making the collected data useful for the training and evaluation of machine learning models. Existing surveys of eye-tracking research studies in software engineering [Kuang et al. 2023; Obaidellah et al. 2018; Sharafi et al. 2015] give overviews with a focus on data gathering, experimental setups, and metrics [Sharafi et al. 2015], but likewise do not discuss the use of machine learning techniques in these studies, nor whether data gathered in these studies is suitable for the development of machine learning models. Likewise, more general guidelines for eye tracking studies [e.g. Holmqvist et al. 2022] do not pay particular attention to machine learning approaching in gaze data analysis.

In this paper, we aim to contribute to filling this gap by providing an overview of how machine learning techniques have been applied to gaze data so far in the software engineering literature. To this end, we present a mapping study focused on how machine learning has been applied to gaze data. We start by presenting our method in Section 2, then we present our results in Section 3, before we discuss the results in Section 4 and conclude in Section 5.

```
SUBJAREA ( comp ) AND TITLE-ABS-KEY (
( {debugging} OR {programming} OR {source code navigation} OR {code browsing} OR {code search}
      OR {code review} OR {code reading} OR {code comprehension} OR {program comprehension} )
AND ( {eye tracking} OR {gaze} OR {eye movement} OR {eye-tracker} )
AND ( {machine learning} OR {ML} OR {AI} OR {deep learning} OR {artificial intelligence}
      OR {classifier} OR {classification} OR {support vector} OR {SVM} OR {decision tree}
      OR {random forest} OR {RF} OR {naive bayes} OR {naïve bayes} OR {NB} OR {linear regression} OR {LR}
      OR {clustering} OR {imitation learning} OR {reinforcement learning} ) )
AND ( EXCLUDE ( DOCTYPE , "cr" ) )
```

Figure 1: Search string used for data gathering. SUBJAREA limits the area to Computer Science. TITLE-ABS-KEY searches titles, abstracts, and keywords of publications. DOCTYPE with the option "cr" means "conference review". The highlighted parts show the keywords added in the second iteration of the search string.

2 METHOD

In order to investigate the overarching question of 'how have machine learning techniques been applied to gaze data gathered from software development activities?', we carried out a systematic mapping study [Petersen et al. 2015], by first breaking down our overarching question into the following three lower-level research questions:

- **RQ_1** What problems have been tackled by using machine learning on gaze data in software development?
- **RQ_2** How is machine learning used with gaze data in software development?
- **RQ_3** What are the open challenges in using machine learning on gaze data in software development?

We then composed the search string focused on software development activities, gathering of gaze data, and machine learning, shown in Fig. 1 (without the highlighted part which was added later). When we ran this search string on Scopus[2] on Jan 25, 2023, we got 50 papers in the search result. The first three authors then screened the title, abstract, and keywords of these papers using the following inclusion and exclusion criteria:

- **Include** paper if it employs eye-tracking, machine learning, and software development activity (SDA).
- **Exclude** paper if it is not in English, not available in full text, or is not a primary study. If a study described in a paper does not contain an experiment, then we did not consider it to be a primary study. However, if the study uses a pre-existing data set with gaze data gathered from an experiment, we considered this to be a primary study.

We employed a rule of minority inclusion when there was a disagreement on a paper. That is, even if only one author deemed that a paper should be included, we included it for further examination. Our rationale was that applying this rule would minimize the risk of missing a relevant study. After screening by all authors and the resolution of disagreements, 13 papers remained (i.e., 37 papers were excluded). Next, all three authors read the full texts of the remaining 13 papers, using the same inclusion and exclusion criteria from the screening. After the full-text review, we discussed any disagreement on inclusion and managed to reach a consensus for all discussed papers. In the end, 8 papers remained (i.e., 5 were excluded).

The first author then proceeded to collect the following data from the 8 papers: topic domain (using software development activities as codes), ML algorithms used, the purpose of ML use (a.k.a, prediction task), use of a pre-existing data set, input and output of the ML models, stimulus type, study setting, participant type, the number of participants, additional sensors used, limitations/threats to validity and future work.

Meanwhile, we informally reviewed past publications from the workshop venue - Eye Movement in Programming (EMIP). We found two papers that had the potential to be added to the final paper set but were not captured by the search string mentioned above. The reason was the use of the term "code reading" as a representative software development activity, which we had not included in our initial search string formulation. In addition, when we were coding the data gathered from the 8 papers mentioned above, we identified some frequently-used machine learning techniques, e.g. Random Forest. Hence, we refined the search string to include these techniques to potentially capture more papers of interest, the additions are highlighted in Fig. 1. When we used this search string on Scopus on Feb 1, 2023, we got 83 papers in the search result, with an overlap of 50 papers with the result of the previous search string, i.e., we found 33 new papers. We then went through the same process as described earlier with these 33 papers and ended up including 2 further papers, resulting in a final set of 10 papers.

Since there was strong inter-rater reliability (Kappa=0.84) between the first three authors during the first round of screening and we always reached a consensus after discussion throughout the screenings, we deemed it sufficient for the first author to proceed alone with the coding of the gathered data [3].

When coding experiment-related attributes such as the number of participants and participant type, we collected the information included in publications connected to the study described in the reviewed paper if it is not reported or omitted in the paper we reviewed. Otherwise, we used or prioritized the information reported in the reviewed paper. In some papers we reviewed, we found that their data sample was a subset of the data from an earlier study, e.g. to meet some custom research needs [Harada and Nakayama 2021].

[1] "Eye Tracking fully integrated and baked right into the very latest high performance gaming devices from Alienware, Acer and MSI". https://gaming.tobii.com/products/laptops/ (Visited Feb 15, 2023)
[2] https://www.scopus.com
[3] https://portal.research.lu.se/en/publications/applying-machine-learning-to-gaze-data-in-software-development-a-

3 RESULTS

We start by presenting the demographics of the papers we selected and then we present the results with regard to each research question.

3.1 Demographics

The demographics of the selected papers are listed in Table 1. Almost all papers (9 out of 10) were published in 2018 or later, with a peak in 2021 (3 papers). In terms of venue distribution, 8 papers were published in a conference or a symposium, and 2 in journals. The International Conference on Program Comprehension (ICPC) is the top venue with 2 papers. The majority of the papers (7) presented a study that included data collection, while 3 papers reused an existing data set. The majority of papers (8) report on using educational material as stimuli, while the remaining papers (2) report on using code snippets from either open-source software (OSS) or closed-source software (CSS). When it comes to the choice of stimuli, the same type of material, e.g., educational material, is commonly used for both novices and experts, but there might be an increase in task difficulty or code complexity (or experts may get additional tasks). For studies with only practitioners, researchers tended to employ CSS material for the purpose of capturing realistic gaze behaviors in the workplace.

The majority of studies (6) were carried out in a laboratory setting, with a few (2) also including a workplace setting. One study was carried out in a classroom, and one study was carried out in a workplace. The studies in a mixed setting are usually those contrasting novices with experts in programming expertise, where novices are usually university students and experts are practitioners in the industry. Depending on the goals of the study, sometimes researchers asked both groups of participants to perform the tasks in the lab, while in some other cases, researchers conducted the experiments with experts in their offices.

The majority of papers (8) report using students as participants, which in a number of cases (6) were mixed with researchers and practitioners. The remaining 2 papers report using only practitioners. The number of participants ranges from 15 to 157, with a mean of 48, a median of 30, and a standard deviation of 45. Finally, half of the studies (5) used additional sensors; for Electrodermal Activity (EDA), Galvanic Skin Response (GSA, coded as a type of EDA), Electroencephalogram (EEG), Heart Rate Variability (HRV), mouse, and camera. The most common additional sensors were EDA (3) and EEG (2).

3.2 RQ1: What problems have been tackled by using machine learning on gaze data in software development?

To address this research question, we collected data on the software development activities studied. The top row of Table 2 gives an overview of the activities we found and the bottom row in the same table shows the total number of occurrences of an ML technique being applied to a data set gathered from this activity. Code comprehension is the most studied activity (12), followed by code review (2), pair programming (2), and visualization of debugging gaze data (1).

Beyond activity, we also collected data on the purpose of using an ML technique. Across almost all papers (9 of 10), we found the purpose to be some kind of prediction task, summarized by the top row of Table 3 and counted in the bottom row of the same table. One paper, not shown in the table, used graph embeddings for mapping graphs into a 2D space. Note that multiple papers describe studies where several ML techniques were applied to the same data set [Abbad-Andaloussi et al. 2022; Ahsan and Obaidellah 2020; Fritz et al. 2014; Villamor and Rodrigo 2018]. One paper may also describe more than one prediction task [Lee et al. 2018]. As such the totals in these tables (17) exceed the number of papers (10). The most common ML task was to predict programmer expertise (6), followed by task difficulty (4). After that we saw prediction of mentally demanding code (2), success of pair programming (2), quality of code review (1), affect in code review (1), and code reading ability (1).

3.3 RQ2: How is machine learning used with gaze data in software development?

To address this question, we consider what ML techniques are applied on gaze data and how. From the heatmap presentations in Table 2 and Table 3, we see that Support Vector Machines (SVM) and Decision Trees are the two most commonly used techniques, followed by Random Forest (2), K-nearest neighbor (KNN) (2), and Naive Bayes (2). The preference for the top two techniques and Random Forests is due to their capability to handle large data sets [Lee et al. 2018; Vrzakova et al. 2020], overfitting [Al Madi et al. 2021; Lee et al. 2018], and effective ranking/selection of features [Al Madi et al. 2021; Vrzakova et al. 2020], while the preferences for KNN and Naive Bayes is due to simplicity [Fritz et al. 2014] and/or explainability [Hijazi et al. 2021].

To dig deeper into how the ML techniques are applied, we consider the ML techniques from a perspective of a model taking input and providing output, summarized in Table 4. The listed models correspond to the ML techniques listed earlier in Table 2 and Table 3. In the input column, we convey our interpretation of the details from the descriptions provided in the papers along with examples for clarification. We find a fairly high degree of heterogeneity, which may not be surprising given the available granularity/properties/dimensions of gaze data. However, in general, we see a tendency to use raw gaze data in combination with embedded feature selection algorithms (e.g., [Abbad-Andaloussi et al. 2022; Lee et al. 2018]) to help pinpoint the prominent features for a specific task. In cases where several sensors are used (e.g., EDA, EEG) or data is gathered via other methods (e.g., answers to a questionnaire for participant performance assessment [Ahsan and Obaidellah 2020]), the data from the other sensors are incorporated in the training data set. Further, some researchers also used analysis results or derived metrics (e.g., CRQA attributes [Villamor and Rodrigo 2018]) of gaze data as the input for training their models.

3.4 RQ3: What are the open challenges in using machine learning on gaze data in software development?

We address this research question from two aspects: limitations and challenges. We constructed the following themes from the

Table 1: Overview of selected papers. ICPC = International Conference on Program Comprehension, ISSRE = International Symposium on Software Reliability Engineering, ETRA = Eye Tracking Research and Applications Symposium, APSIPA ASC = Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, JSS = Journal of Systems and Software, VISSOFT = Working Conference on Software Visualization, UMAP = Conference on User Modeling, Adaptation and Personalization, JCC = Journal of Cluster Computing, ICSE = International Conference on Software Engineering, EDA = Electrodermal Activity, HRV = Heart Rate Variability, EEG = Electroencephalogram, CSS = Closed Source Software, OSS = Open Source Software.

Paper	Year	Venue	Use of data set	Stimulus type	Study setting	Participant type	Nbr. participants	Extra sensors
ASW22 [Abbad-Andaloussi et al. 2022]	2022	ICPC	No	Edu.	Lab	Mixed	16	EDA
APSM21 [Al Madi et al. 2021]	2021	ICPC	Yes	Edu.	Mixed	Mixed	20	-
HCC+21 [Hijazi et al. 2021]	2021	ISSRE	No	Edu.	Lab	Mixed	21	HRV
HN21 [Harada and Nakayama 2021]	2021	ETRA	Yes	Edu.	Lab	Mixed	157	-
AO20 [Ahsan and Obaidellah 2020]	2020	APSIPA	No	Edu.	Lab	Students	66	-
VBMB20 [Vrzakova et al. 2020]	2020	JSS	No	CSS	Workplace	Practitioners	37	EDA + Mouse
ZSPSY19 [Zhang et al. 2019]	2019	VISSOFT	Yes	OSS	Mixed	Mixed	22	-
VR18 [Villamor and Rodrigo 2018]	2018	UMAP	No	Edu.	Classroom	Students	84	-
LHJNL18 [Lee et al. 2018]	2018	JCC	No	Edu.	Lab	Mixed	38	EEG
FBMYZ14 [Fritz et al. 2014]	2014	ICSE	No	Edu.	Lab	Practitioners	15	EEG + EDA + Camera

Table 2: Heatmap of software development activity x ML technique.

ML / Activity	Code comprehension	Code review	Pair programming	Debugging	Total
Support Vector Machine	4	0	0	0	4
Decision Tree	3	0	0	0	3
K-nearest Neighbours	1	1	0	0	2
Naive Bayes	1	0	1	0	2
Random Forest	1	1	0	0	2
Ensemble Learning	1	0	0	0	1
Graph Embedding	0	0	0	1	1
K-means Clustering	1	0	0	0	1
Linear Regression	0	0	1	0	1
Total	12	2	2	1	17

limitations reported in the set of selected papers (not all papers reported limitations), sorted on occurrence in papers:

- **Limited sample size** (4 papers): the size of the sample is not large enough (perhaps despite being representative [Al Madi et al. 2021]) and thus may impact the applicability or generalizability of the model [Abbad-Andaloussi et al. 2022; Fritz et al. 2014; Vrzakova et al. 2020].
- **Unbalanced data set** (3 papers): skewness in sampling, insufficient or absence of representation of certain cohorts in the data set [Fritz et al. 2014; Vrzakova et al. 2020], e.g., lack of samples of professional programmers [Al Madi et al. 2021].
- **Binary categorization** (2 papers): specifically refers to the potential risk of over-simplification of programmer expertise and the task difficulty with a binary categorization method, e.g., novices vs. experts [Hijazi et al. 2021], and easy vs. hard [Fritz et al. 2014].
- **Information trade-off** (1 paper): loss or compromise of certain information due to the use of an ML technique, e.g.,

Table 3: Heatmap of prediction problem x ML technique.

ML / Prediction	Programmer expertise	Task difficulty	Mentally demanding code	Success of pair prog.	Quality of code review	Affect in code review	Code reading ability	Total
Support Vector Machine	2	2	0	0	0	0	1	5
Decision Tree	1	1	1	0	0	0	0	3
K-nearest Neighbours	1	0	0	0	1	0	0	2
Naive Bayes	0	1	0	1	0	0	0	2
Random Forest	1	0	0	0	0	1	0	2
Ensemble Learning	0	0	1	0	0	0	0	1
K-means Clustering	1	0	0	0	0	0	0	1
Linear Regression	0	0	0	1	0	0	0	1
Total	6	4	2	2	1	1	1	17

Table 4: Overview of input, output and ML model used.

Paper	Input	Model	Output
ASW22 [Abbad-Andaloussi et al. 2022]	line and fragment level gaze data, e.g., "features derived from fixation, saccade, pupil, scan-path"	Decision Tree, Ensemble Leaning	mentally demanding code fragments
APSM21 [Al Madi et al. 2021]	token level gaze data, e.g., "Single Fixation Duration, First Fixation Duration"	Radom Forest	programmer expertise (novice or expert)
HCC+21 [Hijazi et al. 2021]	biometric features and non-biometric features, e.g., HRV, scan time, code complexity, experience level	K-nearest Neighbour	code review quality of code regions
HN21 [Harada and Nakayama 2021]	eye movement features, e.g., overall eye movement horizontally and vertically	Support Vector Regression	code reading ability
AO20 [Ahsan and Obaidellah 2020]	total fixation duration (TFD) and participants' answers to questions	K-means clustering, Decision Tree, K-nearest Neighbour, Support Vector Machine	expertise of novice programmer
VBMB20 [Vrzakova et al. 2020]	gaze data + gaze shift data + EDA data + mouse logs, e.g., Euclidean distance and velocity derived from consecutive gaze samples	Random Forest	affect of the code review
ZSPSY19 [Zhang et al. 2019]	eye movement trajectories and graphs of the software program, e.g., scanpaths	Graph Embedding (node2vec)	embedded space: positions in a 2D space and their distances to each other
VR18 [Villamor and Rodrigo 2018]	Cross-Recurrence Quantification Analysis (CRQA) metrics, e.g., "average diagonal length (L) means the in-sync fixation paths"	Naive Bayes, Linear Regression	success of pair programming
LHJNL18 [Lee et al. 2018]	psycho-physiological sensors data (eye movement + EEG), e.g., total fixation duration	Support Vector Machine	programmer expertise and task difficulty
FBMYZ14 [Fritz et al. 2014]	eye movement data + EEG + EDA, e.g., total fixation duration	Naive Bayes, Support Vector Machine, Decision Tree	task difficulty

the nuanced differences between two gaze trajectories became difficult to identify after having been mapped into a 2D space in the work by Zhang et al. [Zhang et al. 2019].

Similarly, we created themes based on the challenges reported in the paper, as follows:

- **Experiment management**: the inherently difficult-to-control conditions of an eye-tracking experiment, potentially interwoven with other advanced sensors (e.g., EDA [Abbad-Andaloussi et al. 2022; Fritz et al. 2014], EEG [Fritz et al. 2014], or HRV [Vrzakova et al. 2020]), and sensitivity to environmental variables such as illumination, the Hawthorne effect [McCambridge et al. 2014] (participants may attempt to act in a manner pleasing to the researcher), and learning effects while performing the tasks [Fritz et al. 2014]. In addition, a mismatch between a laboratory setting and a natural work environment is most likely unavoidable.
- **Generalizability and validity of ML models**: ML models by nature are vulnerable to bias embedded in the training data set [Ahsan and Obaidellah 2020; Fritz et al. 2014; Vrzakova et al. 2020] and overfitting when trained on a small data set [Abbad-Andaloussi et al. 2022; Al Madi et al. 2021; Fritz et al. 2014; Vrzakova et al. 2020]. This sometimes calls the validity and generalizability/applicability of the trained models in question. However, some researchers deliberately

chose certain ML algorithms, e.g., Random Forest [Al Madi et al. 2021; Vrzakova et al. 2020], over others to mitigate this problem. Regardless, researchers still stated that due to the lack of diversity in their data and a small-sized sample, further validation [Ahsan and Obaidellah 2020; Lee et al. 2018] or in-situ evaluation [Hijazi et al. 2021; Vrzakova et al. 2020] with a larger or unseen data set is needed or recommended.

Additionally, the challenge of getting access to professional programmers was also mentioned [Abbad-Andaloussi et al. 2022], as well as the challenge of finding the optimal parameters to derive metrics for training [Villamor and Rodrigo 2018].

4 DISCUSSION

Similar to the findings reported by Kuang et al. [Kuang et al. 2023], we found that the majority of studies applying machine learning to gaze data are carried out with students, in a laboratory setting, and using educational material. As such, it could be questioned whether the reported results generalize to practitioners in the workplace, a concern also raised in the selected papers. A further concern is the small data sets, in the context of machine learning, used by the studies reported here. Small data sets limit the complexity of the deep learning models that can be developed [Raudys et al. 1991; Vabalas et al. 2019]. We find the community effort to publish more open data [McChesney and Bond 2021] and to curate larger data sets collaboratively [Bednarik et al. 2020] very promising for the future application of ML techniques. Still, despite these efforts, it may be challenging to compose really large data sets. We recommend that more attention is given to the use of existing techniques for mitigating this problem, e.g., data augmentation [Wong et al. 2016; Zemblys et al. 2018] or the generation of suitable training sets [Zemblys et al. 2019].

4.1 Opportunities for Future Research

Below we synthesize the future work stated in the studied papers as opportunities, supplemented with our own insights (sorted after occurrences in papers):

- **Extension of application** (4 papers): the possibility of applying the trained models to a similar sub-area [Lee et al. 2018; Vrzakova et al. 2020; Zhang et al. 2019] or a neighboring field rather than software development, e.g., computer science education [Hijazi et al. 2021].
- **Prediction task expansion** (4 papers): the aspiration to expand the prediction of the trained model to cover other similar or finer-grained tasks [Ahsan and Obaidellah 2020; Al Madi et al. 2021; Fritz et al. 2014; Vrzakova et al. 2020], e.g., "predict where and when the eyes move across a line of code" [Al Madi et al. 2021] and detect "confusion and misinterpretation" [Vrzakova et al. 2020].
- **Model validation and enhancement** (3 papers): this action concerns employing another data set to test the validity of the trained model [Ahsan and Obaidellah 2020], and two ways of enhancing the models with enriched data: training data diversification (e.g., [Villamor and Rodrigo 2018]) and balancing [Abbad-Andaloussi et al. 2022]. One study may mention both enhancement methods. Both training data diversification and balancing increase the scale of the training data.
  - **Training data diversification** (3 papers): include new data from a completely new cohort or scenario [Abbad-Andaloussi et al. 2022; Ahsan and Obaidellah 2020], or with new features of existing data points [Villamor and Rodrigo 2018].
  - **Training data balancing** (1 paper): increase sampling of the current under-represented or hard-to-recruit participant cohorts [Abbad-Andaloussi et al. 2022].
- **System improvement** (3 papers): the envisioning of integrating the trained model into existing systems or tools to better support programmers [Fritz et al. 2014; Lee et al. 2018; Vrzakova et al. 2020].
- **In-situ evaluation** (3 papers): the intention to evaluate the trained model on data from a work environment [Hijazi et al. 2021; Vrzakova et al. 2020] or with a more realistic user study [Zhang et al. 2019].

4.2 Threats to Validity

The typical limitations of a mapping study [Budgen et al. 2018] relevant to the work presented here are the study selection process and the data extraction process. For the study selection process, the initial set of articles was based on a search string in an academic database. We chose a well-known database, which we believe is rather complete. The search string was iterated upon in order to find all relevant articles, but there may still be articles that we did not find with our search string. To mitigate this concern, we have included extra search terms to find relevant articles from the EMIP community, as described in Section 2. The selection of studies was further performed by consensus of multiple authors. Given these actions and our own knowledge of the field, we believe that we have identified a reasonably good set of articles. For the data extraction process, the extracted data was iteratively discussed within the author group, e.g., based on previous research and knowledge of the research field. Since we decided to collect rather well-known, accepted and explicit types of data, such as the type of machine learning model, we believe that no large errors or misunderstandings have occurred in the data extraction and analysis.

5 CONCLUSIONS

This systematic mapping study aims at shedding light on the state of the art of applying ML techniques to gaze data in software engineering. We found a small set of papers matching our criteria but at the same time a growing trend in the number of papers being produced in this area. We identified the software development activities studied so far with eye-tracking and machine learning: code comprehension, code review, pair programming, and debugging. We further found that machine learning has been used for prediction of programmer expertise, task difficulty, mentally demanding code fragments, success of pair programming, quality of and affect in code review, and code reading ability. The most frequently used machine learning techniques are Support Vector Machines and Decision Trees. Overall for all the considered studies, the main limitations reported are limited sample size, unbalanced data set, binary

categorization, and information trade-off, and we further found two overarching challenges in experiment management and generalizability/validity of machine learning models. These limitations and challenges set the stage for many opportunities for future work, e.g., including applications in neighboring fields like computer science education, or to expand into new prediction problems.

ACKNOWLEDGMENTS

This work was partly funded by the Swedish Foundation for Strategic Research (grant nbr. FFL18-0231), the Swedish Research Council (grant nbr. 2019-05658), and Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

Amine Abbad-Andaloussi, Thierry Sorg, and Barbara Weber. 2022. Estimating developers' cognitive load at a fine-grained level using eye-tracking measures. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 111–121.

Zubair Ahsan and Unaizah Obaidellah. 2020. Predicting expertise among novice programmers with prior knowledge on programming tasks. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 1008–1016.

Naser Al Madi, Drew Guarnera, Bonita Sharif, and Jonathan Maletic. 2021. EMIP Toolkit: A Python Library for Customized Post-processing of the Eye Movements in Programming Dataset. In *ACM Symposium on Eye Tracking Research and Applications*. 1–6.

Roman Bednarik, Teresa Busjahn, Agostino Gibaldi, Alireza Ahadi, Maria Bielikova, Martha Crosby, Kai Essig, Fabian Fagerholm, Ahmad Jbara, Raymond Lister, Pavel Orlov, James Paterson, Bonita Sharif, Teemu Sirkiä, Jan Stelovsky, Jozef Tvarozek, Hana Vrzakova, and Ian van der Linde. 2020. EMIP: The eye movements in programming dataset. *Science of Computer Programming* 198 (08 2020), 102520. https://doi.org/10.1016/j.scico.2020.102520

Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.

David Budgen, Pearl Brereton, Sarah Drummond, and Nikki Williams. 2018. Reporting systematic reviews: Some lessons from a tertiary study. *Information and Software Technology* 95 (2018), 62–74.

Thomas Fritz, Andrew Begel, Sebastian C Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th international conference on software engineering*. 402–413.

Hiroto Harada and Minoru Nakayama. 2021. Estimation of reading ability of program codes using features of eye movements. In *ACM Symposium on Eye Tracking Research and Applications*. 1–5.

Haytham Hijazi, José Cruz, João Castelhano, Ricardo Couceiro, Miguel Castelo-Branco, Paulo de Carvalho, and Henrique Madeira. 2021. iReview: an Intelligent Code Review Evaluation Tool using Biofeedback. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 476–485.

K. Holmqvist, S. L. Örbom, I. T. C. Hooge, D. C. Niehorster, R. G. Alexander, R. Andersson, J. S. Benjamins, P. Blignaut, Anne-Marie Brouwer, L. L. Chuang, K. A. Dalrymple, D. Drieghe, M. J. Dunn, U. Ettinger, S. Fiedler, T. Foulsham, J. N. van der Geest, D. W. Hansen, S. Hutton, E. Kasneci, A. Kingstone, P. C. Knox, E. M. Kok, H. Lee, J. Y. Lee, J. M. Leppänen, S. Macknik, P. Majaranta, S. Martinez-Conde, A. Nuthmann, M. Nyström, J. L. Orquin, J. Otero-Millan, S. Y. Park, S. Popelka, F. Proudlock, F. Renkewitz, A. J. Roorda, M. Schulte-Mecklenbeck, B. Sharif, F. Shic, M. Shovman, M. G. Thomas, W. Venrooij, R. Zemblys, and R. S. Hessels. 2022. Eye tracking: empirical foundations for a minimal reporting guideline. *Behavior Research Methods* (2022). https://doi.org/10.3758/s13428-021-01762-8

Peng Kuang, Emma Söderberg, Diederick C. Niehorster, and Martin Höst. 2023. Towards Gaxe-Assisted Developer Tools. In *International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE.

Seolhwa Lee, Danial Hooshyar, Hyesung Ji, Kichun Nam, and Heuiseok Lim. 2018. Mining biometric data to predict programmer expertise and task difficulty. *Cluster Computing* 21 (2018), 1097–1107.

Jim McCambridge, John Witton, and Diana R Elbourne. 2014. Systematic review of the Hawthorne effect: new concepts are needed to study research participation effects. *Journal of clinical epidemiology* 67, 3 (2014), 267–277.

Ian McChesney and Raymond Bond. 2021. Eye Tracking Analysis of Code Layout, Crowding and Dyslexia - An Open Data Set. In *ACM Symposium on Eye Tracking Research and Applications* (Virtual Event, Germany) *(ETRA '21 Short Papers)*. Association for Computing Machinery, New York, NY, USA, Article 33, 6 pages. https://doi.org/10.1145/3448018.3457420

Unaizah Obaidellah, Mohammed Al Haek, and Peter C.-H. Cheng. 2018. A Survey on the Usage of Eye-Tracking in Computer Programming. *ACM Comput. Surv.* 51, 1 (2018).

Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.

Sarunas J Raudys, Anil K Jain, et al. 1991. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on pattern analysis and machine intelligence* 13, 3 (1991), 252–264.

Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. 2020. A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering* 25 (06 2020). https://doi.org/10.1007/s10664-020-09829-4

Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. 2015. A systematic literature review on the usage of eye-tracking in software engineering. *Information and Software Technology* 67 (2015), 79–107.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

Andrius Vabalas, Emma Gowen, Ellen Poliakoff, and Alexander J Casson. 2019. Machine learning algorithm validation with a limited sample size. *PloS one* 14, 11 (2019), e0224365.

Eva AM van Dis, Johan Bollen, Willem Zuidema, Robert van Rooij, and Claudi L Bockting. 2023. ChatGPT: five priorities for research. *Nature* 614, 7947 (2023), 224–226.

Maureen Villamor and Ma Mercedes Rodrigo. 2018. Predicting successful collaboration in a pair programming eye tracking experiment. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. 263–268.

Hana Vrzakova, Andrew Begel, Lauri Mehtätalo, and Roman Bednarik. 2020. Affect recognition in code review: An in-situ biometric study of reviewer's affect. *Journal of Systems and Software* 159 (2020), 110434.

Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. 2016. Understanding Data Augmentation for Classification: When to Warp?. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 1–6. https://doi.org/10.1109/DICTA.2016.7797091

Raimondas Zemblys, Diederick C Niehorster, and Kenneth Holmqvist. 2019. gazeNet: End-to-end eye-movement event detection with deep neural networks. *Behavior Research Methods* 51, 2 (2019), 840–864. https://doi.org/10.3758/s13428-018-1133-5

Raimondas Zemblys, Diederick C. Niehorster, Oleg Komogortsev, and Kenneth Holmqvist. 2018. Using machine learning to detect events in eye-tracking data. *Behavior Research Methods* 50, 1 (01 Feb 2018), 160–181. https://doi.org/10.3758/s13428-017-0860-3

Li Zhang, Jianxin Sun, Cole Peterson, Bonita Sharif, and Hongfeng Yu. 2019. Exploring eye tracking data on source code via dual space analysis. In *2019 Working Conference on Software Visualization (VISSOFT)*. IEEE, 67–77.