Towards Mining Eye-Tracking Datasets for Stress Detection: An Empirical Study of Parameter-tuning

Mahnaz Behroozi
CSC Department
North Carolina State University
Raleigh, NC
Email: mbehroo@ncsu.edu

***Abstract*—Problem-solving on a whiteboard is a popular technical interview technique used in industry. However, several critics have raised concerns that whiteboard interviews can cause excessive stress and cognitive load on candidates, ultimately reinforcing bias in hiring practices. Due to the increased availability of eye tracking devices, it is becoming increasingly feasible for researchers in the software engineering field to collect eye gaze data. Depending on the type of eye tracker, a large amount of data with many different attributes are generated in a short amount of time. We posit that such a dataset (after cleansing) is a rich source for additional information about candidate's stress and cognitive load. This paper will focus on repeatability and improvement of applying the state of the art algorithms by proposing empirical software science approaches such as parameter tuning.**

***Keywords*—Data mining, software engineering, eye-tracking, technical interviews**

I. Introduction

A technical interview is a stage of a job interview, which for software developers, often includes a programming component performed on a whiteboard. A common goal of a technical interview is to obtain visibility into and verifiability of the cognitive processes used by a candidate [13]. Although technical interviews are the most common assessment technique, they have several limitations:

- **Medium and affordances.** Whiteboards are often selected for high visibility of the problem-solving work by interviewers; however, whiteboards lack affordances, such as syntax highlighting, which can cause higher cognitive load [34].
- **Stress.** Public performance, time pressure, and self-efficiency can influence cognitive state [17].
- **Interruption.** Technical interviews require that a candidate perform talk-aloud while problem-solving, which imposes high cognitive load [16, 20] and eliminates opportunity for reflection [9].

When software processes and tools are not well-aligned with the cognitive processing styles of certain populations [20], discrimination can occur. For example, research into gender differences has established that many software tools are designed to be more supportive of problem-solving processes used by males than by females [4]. Critics of whiteboard interviews argue that the technique systematically biases hiring certain candidates [31, 1], due to lack of time to practice for the interview setting, and familiarity with problems. Finally, research in criteria used to evaluate candidates during technical interviews finds that factors [13], such as confidence matter as much as problem-solving ability, meaning that when selecting between two candidates producing the same quality answer, the least visibly stressed candidate is more likely to get the job. Thus, traditional technical interviews can overlook more skillful candidates who do not perform well in these settings.

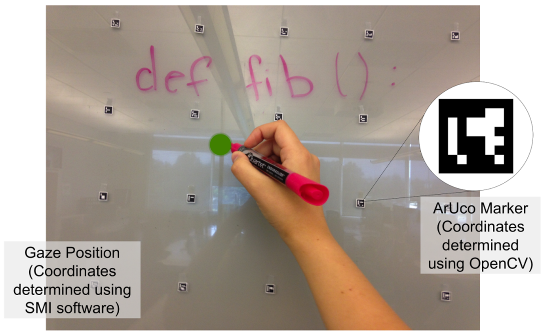
def fib():
ArUco Marker (Coordinates determined using OpenCV)
Gaze Position (Coordinates determined using SMI software)

Figure 1. Feasibility study of using ArUco markers to calculate regressions.

Our work investigates the use of head-mounted eye-trackers that can obtain eye movement measures without restriction to movement.

RQ1: Our primary research question is: *Is cognitive load in technical interviews detectable through eye-tracking?*

We hypothesize that the public setting of the whiteboard will increase cognitive load while the private setting of the paper will reduce it. We validate our hypothesis by asking the participants in which setting they experienced more stress. Also, we analyzed the data with two labelling methods:

1) based on the stress
2) based on the paper or the whiteboard setting.

We applied classification algorithms on the labelled data. The results show that eye-movement can be representative of cognitive load and stress.

RQ2: *: Which eye-tracking measurements are predictive of cognitive load?*
This question arises from RQ1. We can explore those eye-

Figure 2. A head-mounted eye-tracker from SMI used in this study for the sake of free mobility of the candidates.

tracking measurements which have correlation with the class labels. We explored the predictive eye-tracking measurements in the labelled data based on the setting and our finding confirms the literature of eye-tracking.

RQ3: *How much do the properties of the interview setting affect the eye-movement measurements?*
To answer this question, we applied the classification algorithms to our data with two different labelling and partitioning:

1) Labelling based on user survey (stressed/not stressed). In this labelling, we split the data into 4 parts: saccade in whiteboard setting, visual intake in whiteboard setting, saccade in paper setting, visual intake in paper setting.

2) Labelling based on the paper and whiteboard setting. Here we partitioned the data into two parts: saccade and visual intake (this time regardless of the setting they have been taken from).

Then, we compared the performance of the classifiers on these two partitioning and labelling of the data.

RQ4: *Does parameter tuning enhance classification results?*
We applied parameter tuning on our design from RQ3 to see how it changes the performance of the classifiers.

II. Background

Eye-tracking is the process of locating the eye-position and measuring the eye-movement of a subject. Eye-trackers are designed to monitor and collect eye-movement data while the subject is looking at a stimulus during an experiment [11, 23]. A stimulus is an object, such as a piece of code, which is of interest during an eye-tracking study. Generally, there are two types of eye-trackers: head-mounted eye-trackers and remote eye-trackers. Head-mounted eye-trackers have embedded infrared cameras and the subjects wear them. Remote eye-trackers have a screen with sensors and the subjects sit in front of them, such as the ones used to study eye movements of developers while using IDEs [28, 29].

Some common eye-tracking terms and measures are as follows:

- **Areas of Interest (AOI)**: A fixed region corresponding to an object of study.
- **Fixation or Visual Intake (VI)**: Stabilization of the eye on a particular point of the stimulus, typically between 200 to 300 ms. Fixations can be acquired from the eye-tracker in the form of a time stamp and x-y pixel coordinates. Privitera et al. suggest that most of the cognitive processes and information acquisitions happens during fixations [22].
- **Saccade**: A sudden rapid eye movement which occurs between two fixations, typically lasting between 40–50 ms. Prior studies claim that cognitive processing and information acquisition is not notable during saccades [22, 11].
- **Regression**: A backward movement of the eye from an AOI to a previously visited AOI.
- **Cognitive Load**: Several studies report eye-tracking measures which reveal cognitive load. Chen et al. conclude that fixation duration and fixation rate are indicators of an increment in the attention [7]. Increase or decrease of fixation duration depends on the characteristics of the task. Fixation duration will decrease in stressful tasks that need rapid responses (such as driving in a car [33] or airplane piloting [15]). In contrast, in the tasks that needs more cognitive processes such as reading texts of increasing complexity, fixation duration will increase [24]. Chen et al. [7] also included the measurements of saccade velocity and saccade length in order to investigate human mental effort. Their results show that saccade velocity and length are highly discriminatory parameters. Similarly, Manuel et al. [19] find that a decrease in saccade velocity indicates tiredness and an increase of saccade velocity indicates a higher task complexity. High blink latency and a low blink rate indicate high mental effort [7].

The science of eye-tracking analysis goes back to over a century ago in reading research [21]. Today, eye-tracking as a research is growing in popularity in different disciplines such as usability analysis, sports, psychology, marketing , human computer interaction (HCI), software engineering and many more. Software engineering community started showing interest in eye-tracking studies in the early nineties in investigating the reading strategies in code comprehension [27] The first eye-tracking study in software engineering has been conducted by Crosby et al. [8] on analyzing reading strategies and their impact on procedural code comprehension.

However, before 2006, eye-trackers have not been vastly used in software engineering society due to high price of eye-trackers and the difficulties in using them [3]. As a result, many of the previous studies in software engineering domain used Restricted Focus Viewer approach (RFV) [22]. In this approach, only a small portion of the screen has been displayed to the user and the rest of the screen was blurred. A user controls which part of the screen is of regard with the help of the computer mouse. Since RFV impacts the study results by restricting the subjects and unrealistic conditions, software engineering researchers started using other strategies to conduct their research since 2006.

One method of annotating videos is to instrument applications with a plug-in that records data as a task is completed [26]. The instrumentation approach builds on previous methods in that it provides support for non-static applications, but it fails to capture gazes in non-instrumented areas. Therefore, all possible regions must be determined a priori. For example, prior studies require the participants use Eclipse exclusively. In order to improve this approach, the researcher would need to similarly instrument other applications, such as browsers and text editors. A second method is to defer video annotation until after data collection, but this can be limiting and ineffective.

Unfortunately for researchers, the support available to tag dynamic AOIs in programming environments like Eclipse is limited [25]. The main focus areas of software engineering eye-tracking studies can be grouped in 5 main categories [27] from which we are more interested in the first two categories:

- Model comprehension
- Code comprehension
- Debugging
- Collaborative interactions
- Traceability

Under the category of model comprehension, De Smet et al. in [10] investigated the impact of different design patterns on code comprehension using UML class diagrams. They ran their study on 18 students and 8 faculty members and the variables they took into consideration from EyeLink II eye-tracker were spatial density, transitional matrix, average fixation duration, time and scan-path distance. Another study of the same category has been conducted by Soh et al. [30] where they studied the relation between expertise and professional status for UML diagram comprehension.

In code comprehension studies, the subjects will be asked to read pieces of source code to answer comprehension questions. One of the earliest studies of this category is the research of Crosby and Stelovsky in 1990 [8]. In their study, using Pascal source codes of binary search algorithm, they tried to assess the impact of expertise on the developers' comprehension strategies. Another study conducted by Busjahn et al. [6] investigates the differences between source code reading and natural text reading. They also studies attention distribution on code elements to classify experts' and novices' code reading strategies in [5]. Fritz et al. [14] studied the task difficulty using psycho-physiological measures such as electroencephalography (EEG), eye-gaze, electrodermal activity (EDA), and NASA TLX scores [16].

Our motivation to conduct this research is to find out that what adds to the stress level of the candidates while taking a technical interview and our ultimate goal in our future work is to suggest better settings to help the candidates show their skills better and to help the employers to recruit better candidates.

III. Experimental Design and Performance Criteria

A. *Placing boundaries for the coding area*

The SMI head-mounted eye-tracker we used in our experiment generates the pixel coordinates of the gaze point of the participants. ArUco markers [] can be placed in the environment in order to help map gaze coordinates to physical locations (See Figure 1). These markers are robust and perform well when they are viewed from an angle or in the case of occlusion. We implemented ArUco marker detection in Python using OpenCV. Each ArUco marker has a unique pattern which corresponds to an ID from 0 to 1024. We used an online tool to generate the markers.

B. *Determining AOIs for Coding Area*

We placed markers in a grid pattern on the whiteboard. The size of each marker on the board is 1x1 cm and the distance between each is 9 cm. There are 15 columns and 11 rows of markers. We placed markers on the board in row-wise ascending order. We selected the grid design for several reasons. First, to account for subjects changing their distance from the whiteboard (getting too close), we ensured the distance between markers was close enough to always be visible in the video recordings. Second, the grid pattern enabled us to easily calculate gaze regressions in our analysis.

For the paper setting, we used 0.5x0.5 cm ArUco markers printed in a Legal page layout. We placed the markers with 4.4 cm distance for top and bottom of the frame and 2 cm distance between the markers for the sides of the frame.

C. *Calculating Gaze Regressions*

For the whiteboard setting, in each video frame we determined the nearest markers to the gaze point. We considered regression as any backward movement of the gaze more than one row. For the paper setting, we mapped the pixel distances into actual centimeters since the distance of the user to the paper makes the pixel distances vary in each frame. After calculating the actual distances, we considered 2 cm gaze point backtrack as the regression threshold.

D. *Pilot Methodology*

We recruited 8 graduate and 3 undergraduate participants to participate in our study. First, we helped participants don and calibrate the SMI head-mounted eye-tracking glasses. The glasses were connected to a mobile phone placed in the participant's pocket. Next, we assigned participants two tasks of comparable difficulty from "Elements of Programming Interviews in Java" [2]: (1) reversing all words in an input string, and (2) testing a string for being a palindrome. We allowed participants to write code to complete each task in any programming language of their choice. Each participant completed tasks in two settings: on a whiteboard and on paper. The order of settings varied randomly across participants to account for learning and ordering effects. Each participant completed both tasks in less than 45 minutes. We then conducted debriefing interviews with each participant to see which setting makes them experience more stress.

E. *Looking for predictive eye-tracking measurements*

For the first phase of our experiment we decided to look into 12 eye-tracking measurements to see if the results confirm the previous studies which noted that visual intake and saccade can show the cognitive load. In addition to the outputs of the SMI eye-tracker, we also implemented the regression frequency estimator since we believe that regression also can be representative of the cognitive load.

F. *Focusing on the predictive ability of saccade and visual intake measurements*

To study more on the visual intake and saccade in absence of other eye-tracking measurements, we decided to just consider those measurements related to them and analyze the data considering VI-based and saccade-based measurements. There are 7 VI-related and 11 saccade-related measurements. we labeled the data into two classes of stressed/not stressed based on our participant survey. For this part of our study, we applied two data splitting strategies: first, we split the data into 4 files including saccade from whiteboard setting, visual intake from whiteboard setting, saccade form paper setting and visual intake from paper setting. Second, we split the data into 2 files including all the saccadic data and all the visual intake data regardless of the setting they came from.
We applied the following classification algorithms on our data: Naive Bayes (NB), Random Forest (RF), Multi-Layer Perceptron (MLP), SVM, KNN, Logistic Regression (LR) and Decision Tree (DT).

G. *Studying the effect of parameter-tuning on the performance of the state-of-the-art algorithms*

In order to see whether parameter tuning enhances the classification results or not, we repeated applying the classifiers via parameter tuning.

H. *Performance Measures and Statistical Methods*

Statistical hypothesis test: For answering RQ1 and RQ2 and finding relevant eye-tracking measurements we used Wilcoxon signed-rank test which is a non-parametric statistical hypothesis test.

Model Validation: Throughout the study, we used 10-fold-cross-validation technique for validating the generalization of our results obtained from applying classification algorithms on our data.

Statistical analysis of binary classification: To evaluate the accuracy of the results obtained from classifiers, we reported weighted F-measure along with accuracy, weighted precision and weighted recall of both classes. Their definitions are as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F = 2.\frac{Precision.Recall}{Precision+Recall}$$

To illustrate the diagnostic ability of each of the binary classifiers on our data, we also reported receiver operating characteristic curve (ROC Area).

IV. Results

A. *Results Answering RQ1 and RQ2*

To identify statistically significant differences between the whiteboard and paper setting, we performed a Wilcoxon signed-rank test. Our results indicated that from 12 eye-tracking measurements used in our first phase of the research, 6 of them reveal the differences between the two settings (see Table I). Statistically significant *p*-values has been shown in green, with median values highlighted in yellow. We found no statistical differences in any measure or time performance between the two tasks.

a) Under pressure.: We observed significantly shorter duration for fixations when participants were solving problems on the whiteboard. We believe that the whiteboard setting may have placed participants under pressure to keep shorter periods of attention [33, 15]. As a result, this may limit the ability for a participant to reflect and reason during problem solving. In contrast, participants finished their tasks on the paper in 50 seconds on average. After the experiment, most participants rated the whiteboard setting as more stressful. We also observed participants display several visible *nervous tics*, such as humming and face and hand twitches, that were only noticeable when solving problems on the whiteboard. Finally, it is interesting to consider that for some problem-solving tasks, limited amounts of stress can enhance performance to a limited degree [35]. In some cases, whiteboard interviews can enhance performance for some candidates; however, by increasing difficulty or time pressure, these effects can be quickly reversed.

b) Higher cognitive load.: We observed several measures related to higher cognitive load. When solving problems on the whiteboard, participants had 3x as many regressions versus paper. During cognitively demanding or stressful tasks, a programmer may have more difficulty sustaining attention in working memory. Regression frequency shows that how many times the participant needed to look back to their previous lines of code, which can indicate memory failure as well as higher uncertainty. Higher saccade duration average and saccade velocity average in the whiteboard setting indicate higher stress and lower concentration while doing the task. Finally, higher blink duration average happens during complex cognitive process. During whiteboard problem-solving, we observe a 2x increase in blink duration, which indicates that the participants experienced higher cognitive load in the whiteboard setting. Some of these measures are consistent with stress caused by the public performance of the task; however, other measures may have resulted from the larger coding area of the whiteboard. For example, higher saccade velocity and duration may be consistent with increased difficulty in locating code fragments over a large space, which is less likely to occur in a paper setting.

Table I. STATISTICAL SIGNIFICANT DIFFERENCES (p-VALUE$<$0.05) OF VARIOUS SUMMARY EYE MEASURES BETWEEN DIFFERENT INTERVIEW SETTINGS.

	WHITEBOARD[1]		PAPER		
Eye measure[2]	mean	median	mean	median	p-value
`Trial duration [s]`	405.3	407.5	453.7	416.7	0.438
`Visual Intake Frequency [count/s]`	1.94	2.2	2.25	2.2	0.74
`Visual Intake Duration Average [ms]`	276.8	295.4	353.8	363.2	0.04
`Visual Intake Dispersion Average [px]`	132.2	117.7	41.86	30.7	0.0001
`Saccade Frequency [count/s]`	1.73	1.9	2.01	2.1	0.89
`Saccade Duration Average [ms]`	375.8	96.9	69.8	62.7	0.002
`Saccade Amplitude Average [Â°]`	5.84	5.3	4.14	3.8	0.059
`Saccade Velocity Average [Â° /s]`	2326.5	213.3	131.2	79	0.010
`Saccade Latency Average [ms]`	1432.3	356.1	432.3	397.8	0.22
`Blink Frequency [count/s]`	0.12	0.1	0.2	0.1	0.24
`Blink Duration Average [ms]`	1330.4	571.8	269.1	248.5	0.003
`Regression Frequency [count/s]`	0.15	0.14	0.08	0.04	0.039

[1] WHITEBOARD predominately has statistical differences in measures associated with higher cognitive load.
[2] High cognitive load associated with higher regressions, higher fixation dispersion, longer blinks, larger saccade velocity, longer saccade duration. Stress associated with shorter fixation duration, larger saccade velocity, and longer saccade duration.

Table II. RESULTS FROM APPLYING CLASSIFIERS ON THE LABELLING BASED ON THE PARTICIPANT SURVEY (STRESSED/NOT STRESSED) USING 10-FOLD CROSS VALIDATION AND **WITHOUT** PARAMETER TUNING.

Classifier	Measure	Board Saccade	Board VI	Paper Saccade	Paper VI
NB	Accuracy%	80.83	80.08	56.51	72.77
	Precision	0.80	0.781	0.65	0.73
	Recall	0.80	0.80	0.57	0.73
	F-measure	0.77	0.77	0.45	0.73
	ROC Area	0.69	0.82	0.75	0.76
RF	Accuracy%	90.35	95.80	87.92	96.80
	Precision	0.90	0.96	0.88	0.97
	Recall	0.90	0.96	0.88	0.97
	F-measure	0.90	0.96	0.88	0.97
	ROC Area	0.94	0.99	0.95	0.99
MLP	Accuracy%	81.40	92.08	82.81	93.07
	Precision	0.80	0.92	0.83	0.93
	Recall	0.81	0.92	0.83	0.93
	F-measure	0.78	0.92	0.83	0.93
	ROC Area	0.79	0.96	0.88	0.97
SVM	Accuracy%	78.03	81.01	79.89	77.21
	Precision	0.80	0.81	0.80	0.78
	Recall	0.78	0.81	0.80	0.78
	F-measure	0.69	0.77	0.80	0.78
	ROC Area	0.51	0.61	0.80	0.78
KNN	Accuracy%	81.10	92.08	80.52	96.71
	Precision	0.81	0.92	0.81	0.97
	Recall	0.81	0.92	0.81	0.97
	F-measure	0.81	0.92	0.81	0.97
	ROC Area	0.72	0.90	0.80	0.97
LR	Accuracy%	83.25	90.06	79.44	77.57
	Precision	0.82	0.90	0.80	0.78
	Recall	0.83	0.90	0.80	0.78
	F-measure	0.81	0.90	0.80	0.78
	ROC Area	0.83	0.91	0.84	0.83
DT	Accuracy%	86.93	93.83	84.89	95.28
	Precision	0.87	0.94	0.85	0.95
	Recall	0.87	0.94	0.85	0.95
	F-measure	0.87	0.94	0.85	0.95
	ROC Area	0.82	0.94	0.87	0.96

Table III. RESULTS FROM APPLYING CLASSIFIERS ON THE LABELLING BASED ON THE PARTICIPANT SURVEY (STRESSED/NOT STRESSED) USING 10-FOLD CROSS VALIDATION AND **WITH** PARAMETER TUNING.

Classifier	Measure	Board Saccade	Board VI	Paper Saccade	Paper VI
NB	Accuracy%	82.06	81.29	56.08	73.56
	Precision	0.83	0.80	0.72	0.79
	Recall	0.82	0.81	0.56	074
	F-measure	0.78	0.78	0.43	0.73
	ROC Area	0.76	0.88	0.85	0.87
RF	Accuracy%	100	99.99	100	100
	Precision	1	1	1	1
	Recall	1	1	1	1
	F-measure	1	1	1	1
	ROC Area	1	1	1	1
MLP	Accuracy%	81.89	92.71	83.53	86.34
	Precision	0.82	0.93	0.84	0.87
	Recall	0.82	0.93	0.84	0.86
	F-measure	0.78	0.92	0.84	0.86
	ROC Area	0.82	0.97	0.88	0.93
SVM	Accuracy%	78.23	84.36	80.08	79.47
	Precision	0.81	0.85	0.80	0.80
	Recall	0.78	0.84	0.80	0.80
	F-measure	0.69	0.82	0.80	0.80
	ROC Area	0.51	0.69	0.80	0.80
KNN	Accuracy%	86.51	93.69	85.81	100
	Precision	0.86	0.94	0.86	1
	Recall	0.86	0.94	0.86	1
	F-measure	0.86	0.94	0.86	1
	ROC Area	0.92	0.98	0.94	1
LR	Accuracy%	83.15	90.01	79.43	78.26
	Precision	0.82	0.90	0.79	0.78
	Recall	0.83	0.90	0.79	0.78
	F-measure	0.81	0.90	0.79	0.78
	ROC Area	0.83	0.91	0.84	0.82
DT	Accuracy%	94.49	98.23	92.59	98.01
	Precision	0.94	0.98	0.93	0.98
	Recall	0.95	0.98	0.93	0.98
	F-measure	0.94	0.98	0.93	0.98
	ROC Area	0.96	0.99	0.97	0.99

B. Results Answering RQ3

Comparing the results from Table II and Table IV, we can see that classifiers were more successful and accurate on the stressed/not stressed labelling. Hence we can infer that although the properties of the setting might influence the eye-tracking measurements but still it cannot change the hypothesis that whiteboard setting bears more cognitive load on the candidates.

C. Results Answering RQ4

To identify if parameter tuning affects the results of classification, we applied CVParameterSelection in WEKA [18]. Table VI shows the parameters we tried to tune in each classifier. Table III and Table V show the results after parameter tuning. Improvements are highlighted in green. Results show that the improvements were more notable when we used the labelling based on stressed/not stressed and the most successful classifiers were Random Forest, Decision Tree and KNN. Hence, we can confidently say that parameter tuning is capable

Table IV. RESULTS FROM APPLYING CLASSIFIERS ON THE LABELLING BASED ON THE SETTINGS (PAPER/WHITEBOARD) USING 10-FOLD CROSS VALIDATION AND **WITHOUT** PARAMETER TUNING.

Classifier	Measure	Saccade	VI
NB	Accuracy%	63.30	76.28
	Precision	0.64	0.77
	Recall	0.63	0.76
	F-measure	0.55	0.75
	ROC Area	0.63	0.79
RF	Accuracy%	77.52	96.45
	Precision	0.77	0.97
	Recall	0.78	0.97
	F-measure	0.77	0.97
	ROC Area	0.85	0.99
MLP	Accuracy	65.44	89.02
	Precision	0.65	0.89
	Recall	0.65	0.89
	F-measure	0.61	0.89
	ROC Area	0.65	0.95
SVM	Accuracy%	62.61	75.27
	Precision	0.72	0.76
	Recall	0.63	0.75
	F-measure	0.50	0.74
	ROC Area	0.52	0.71
KNN	Accuracy%	66.66	93.1
	Precision	0.67	0.93
	Recall	0.67	0.93
	F-measure	0.67	0.93
	ROC Area	0.65	0.93
LR	Accuracy%	66.17	75.56
	Precision	0.67	0.75
	Recall	0.66	0.76
	F-measure	0.61	0.75
	ROC Area	0.69	0.79
DT	Accuracy%	72.20	94.63
	Precision	0.72	0.95
	Recall	0.72	0.95
	F-measure	0.72	0.95
	ROC Area	0.76	0.95

Table V. RESULTS FROM APPLYING CLASSIFIERS ON THE LABELLING BASED ON THE SETTINGS (PAPER/WHITEBOARD) USING 10-FOLD CROSS VALIDATION AND **WITH** PARAMETER TUNING.

Classifier	Measure	Saccade	VI
NB	Accuracy%	63.33	76.86
	Precision	0.64	0.77
	Recall	0.63	0.77
	F-measure	0.55	0.76
	ROC Area	0.66	0.79
RF	Accuracy%	80.80	100
	Precision	0.80	1
	Recall	0.81	1
	F-measure	0.80	1
	ROC Area	0.90	1
MLP	Accuracy	65.90	88.33
	Precision	0.65	0.89
	Recall	0.66	0.88
	F-measure	0.65	0.88
	ROC Area	0.68	0.96
SVM	Accuracy%	62.96	76.42
	Precision	0.68	0.77
	Recall	0.63	0.76
	F-measure	0.52	0.75
	ROC Area	0.53	0.72
KNN	Accuracy%	69.80	96.40
	Precision	0.69	0.96
	Recall	0.70	0.96
	F-measure	0.69	0.96
	ROC Area	0.75	0.996
LR	Accuracy%	66.21	75.55
	Precision	0.67	0.75
	Recall	0.66	0.76
	F-measure	0.61	0.75
	ROC Area	0.69	0.79
DT	Accuracy%	72.79	97.88
	Precision	0.72	0.98
	Recall	0.73	0.98
	F-measure	0.73	0.98
	ROC Area	0.76	0.99

Table VI. LIST OF PARAMETERS TUNED FOR EACH CLASSIFIER

Classifier	Parameters
RF	max features, ,max depth
MLP	Hidden layers, learning rate, momentum
SVM	C
KNN	number of neighbors
LR	number of boosting iterations
DT	Confidence factor

Table VII. RESULTS FROM AUTOWEKACLASSIFIER PARAMETER-TUNING FOR THE STRESS/NOT STRESS LABELLING

Classifier	Measures	Board saccade	Board VI	Paper saccade	Paper VI
RF	Accuracy%	99.27	99.54	98.71	99.01
	Precision	0.99	0.995	0.99	0.99
	Recall	0.99	0.995	0.99	0.99
	F-measure	0.99	0.995	0.99	0.99
	ROC Area	0.999	1	0.999	1

of enhancing classification results.

We also tried another parameter tuning named as autoWeka-Classifier [32]. This method automatically finds the best model with its best parameter settings for a given dataset (see Table VII and Table VIII). The results again shows that random forest is successful in this domain.

V. THREATS TO VALIDITY

There are multiple threats to the validity of any empirical study. This study is susceptible to the following threats:

Bias toward hypothesis - We hypothesized that the white-board setting bears a higher cognitive load on the candidates. This hypothesis is not firm among all the candidates and it might vary from person to person. Although the majority of the candidates stated the same but this hypothesis has to be studied more on a larger number of candidates.

Generalizing on a small dataset - As stated before, this small dataset is not sufficient to draw a rule from. Although each experiment has thousands of records but still we have to investigate more experiments to be confident about generalization of the results.

Bias toward participants' survey - We surveyed the cognitive load experience of the participants but still we cannot claim that the participant is experiencing the same amount of cognitive load or stress throughout the whole session. As a result, we have to find a way to reach a better estimation or score of cognitive load during the session. This makes the study more realistic.

Weak parameter tuning - We just studied a limited range of parameter tuning on few classifiers. A better study of parameter tuning techniques with a larger number of classifiers is needed to see the power of parameter tuning in enhancing

Table VIII. RESULTS FROM AUTOWEKACLASSIFIER PARAMETER-TUNING FOR THE SETTING LABELLING

Classifier	Measures	Saccade	Visual Intake
RF	Accuracy%	98.90	98.96
	Precision	0.99	0.99
	Recall	0.99	0.99
	F-measure	0.99	0.99
	ROC Area	0.999	1

classification results. Although AutoWekaClassifier gained notable results with Random Forest classifier, it seems that it is at the risk of overfitting.

VI. Discussion and future work

Our preliminary results demonstrate that our approach is a promising way to understand the impact of technical interviews. However, more studies are needed to better understand what characteristics contribute to high cognitive load. We highlight some future steps:

1) Simple vs. Complex tasks on a whiteboard
2) Solving problems on whiteboard, but in a private room
3) Effect of interviewer interruption
4) Effect of previous problem practice

The first study would allow us to isolate factors related to problem solving problem on a large space. The second study would allow us to study the effect of public performance. The third study would allow us to study how interruptions may further add stress and high cognitive load. The fourth study examines the effect of previous practice on reducing performance anxiety. In addition to future studies, collecting alternative measures of information, such as heart rate variability, can offer deeper insight into the cognitive state of a programmer. Finally, we can analyze in more detail how problem strategies and solutions change with interview settings.

Once we better understand the impact of certain interview practices, we can then design alternative procedures or interventions that reduce unnecessary cognitive load in candidates. We will explore a set of interventions and evaluate their impact on cognitive load. One example intervention includes the concept of an interview "blackout" [12]. For example, an interviewer might say, "now that I have explained the problem, I will step out for about 4 minutes to allow you to digest the problem." This simple measure can allow candidates to reflect on a problem in isolation, potentially reducing anxiety and allowing the candidate the opportunity to reflect on potential approaches uninhibited.

Finally, we can derive a recommended set of interview practices that have been validated in terms of enabling assessment of the candidate while minimizing unnecessary cognitive load.

VII. Conclusion

Technical interviews provide visibility into a candidate's cognitive state and thought processes. Programming is a cognitive intensive task that defies expectations of constant feedback that today's interview processes follow. This has left a gap in understanding what goes on during the programming interview process and how to properly assess programming skills of candidates to succeed at these interviews. With the proposed approach, we will begin to comprehend the cognitive state and sustained attention of candidates during technical interviews to refine the interview process.

A lesson that we learned from this study is that it is necessary to tune the classifiers. Moreover, a classifier with a specific parameter setting which performs well in one domain will not necessarily work for another domain. Finally, the most important lesson is that data matters in a study. In another words, if data is not of high quality and not cleaned up, then it does not matter how powerful are the classifier and how much they are tuned.

References

[1] Chris Alvino. *Technical Interviews an Instrument of Exclusion and Discrimination*. 2014. URL: https://careerconservatory.com/technical-interviews-an-instrument-of-exclusion-and-discrimination/.

[2] Adnan Aziz, Tsung-Hsien Lee, and Amit Prakash. *Elements of Programming Interviews in Java: The Insiders' Guide*. USA: CreateSpace Independent Publishing Platform, 2015. ISBN: 1517435803, 9781517435806.

[3] Roman Bednarik and Markku Tukiainen. "Effects of Display Blurring on the Behavior of Novices and Experts During Program Debugging". In: *CHI '05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '05. Portland, OR, USA: ACM, 2005, pp. 1204–1207. ISBN: 1-59593-002-7. DOI: 10.1145/1056808.1056877. URL: http://doi.acm.org/10.1145/1056808.1056877.

[4] Margaret Burnett et al. "Finding Gender-Inclusiveness Software Issues with GenderMag: A Field Investigation". In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI '16. Santa Clara, California, USA: ACM, 2016, pp. 2586–2598. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858274. URL: http://doi.acm.org/10.1145/2858036.2858274.

[5] Teresa Busjahn, Roman Bednarik, and Carsten Schulte. "What Influences Dwell Time During Source Code Reading?: Analysis of Element Type and Frequency As Factors". In: *Proceedings of the Symposium on Eye Tracking Research and Applications*. ETRA '14. Safety Harbor, Florida: ACM, 2014, pp. 335–338. ISBN: 978-1-4503-2751-0. DOI: 10.1145/2578153.2578211. URL: http://doi.acm.org/10.1145/2578153.2578211.

[6] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. "Analysis of Code Reading to Gain More Insight in Program Comprehension". In: *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*. Koli Calling '11. Koli, Finland: ACM, 2011, pp. 1–9. ISBN: 978-1-4503-1052-9. DOI: 10.1145/2094131.2094133. URL: http://doi.acm.org/10.1145/2094131.2094133.

[7] Siyuan Chen et al. "Eye Activity As a Measure of Human Mental Effort in HCI". In: *Proceedings of the 16th International Conference on Intelligent User Interfaces*. IUI '11. Palo Alto, CA, USA: ACM, 2011, pp. 315–318. ISBN: 978-1-4503-0419-1. DOI: 10.1145/1943403.1943454. URL: http://doi.acm.org/10.1145/1943403.1943454.

[8] Martha E. Crosby and Jan Stelovsky. "How Do We Read Algorithms? A Case Study". In: *Computer* 23.1 (Jan. 1990), pp. 24–35. ISSN: 0018-9162. URL: http://dl.acm.org/citation.cfm?id=77577.77580.

[9] Jane Dawson. "Reflectivity, creativity, and the space for silence". In: *Reflective Practice* 4.1 (2003), pp. 33–39.

[10] Benoît De Smet et al. "Taupe: Visualizing and analyzing eye-tracking data". In: *Science of Computer Programming* 79 (Jan. 2014), pp. 260–278. ISSN: 0167-6423. DOI: 10.1016/j.scico.2012.01.004.

[11] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN: 1846286085.

[12] Denae Ford, Titus Barik, and Chris Parnin. "Studying Sustained Attention and Cognitive States with Eye Tracking in Remote Technical Interviews". In: *Eye Movements in Programming: Models to Data* (2015), p. 5.

[13] Denae Ford et al. "The Tech-talk Balance: What Technical Interviewers Expect from Technical Candidates". In: *Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE '17. Buenos Aires, Argentina: IEEE Press, 2017, pp. 43–48. ISBN: 978-1-5386-4039-5. DOI: 10.1109/CHASE.2017.8. URL: https://doi.org/10.1109/CHASE.2017.8.

[14] Thomas Fritz et al. "Using psycho-physiological measures to assess task difficulty in software development". In: *Proceedings of the 36th International Conference on Software Engineering*. ACM. 2014, pp. 402–413.

[15] SJ Gerathewohl. "Inflight measurement of pilot workload: A panel discussion." In: *Aviation, space, and environmental medicine* (1978).

[16] Sandra G Hart and Lowell E Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research". In: *Advances in psychology* 52 (1988), pp. 139–183.

[17] S. Klitzman et al. "Work stress, nonwork stress, and health." In: *Journal of behavioral medicine* 13.3 (June 1990), pp. 221–243. ISSN: 0160-7715. URL: http://view.ncbi.nlm.nih.gov/pubmed/2213867.

[18] Ron Kohavi. *Wrappers for performance enhancement and oblivious decision graphs*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1995.

[19] Victor Manuel et al. "AdELE: A Framework for Adaptive E-Learning through Eye Tracking". In: *Proceedings of IKNOW 2004*. 2004, pp. 609–616.

[20] Meredith Ringel Morris, Andrew Begel, and Ben Wiedermann. "Understanding the Challenges Faced by Neurodiverse Software Engineering Employees: Towards a More Inclusive and Productive Technical Workforce". In: *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. ASSETS '15. Lisbon, Portugal: ACM, 2015, pp. 173–184. ISBN: 978-1-4503-3400-6. DOI: 10.1145/2700648.2809841. URL: http://doi.acm.org/10.1145/2700648.2809841.

[21] Alex Poole and Linden J Ball. "Eye tracking in HCI and usability research". In: *Encyclopedia of human computer interaction* 1 (2006), pp. 211–219.

[22] Claudio M. Privitera and Lawrence W. Stark. "Algorithms for defining visual regions-of-interest: Comparison with eye fixations". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.9 (2000), pp. 970–982.

[23] Keith Rayner. "Eye movements in reading and information processing: 20 years of research." In: *Psychological bulletin* 124.3 (1998), p. 372.

[24] Keith Rayner. "Eye movements in reading: Eye guidance and integration". In: *The processing of visible language I* ().

[25] Paige Rodeghero et al. "Improving Automated Source Code Summarization via an Eye-tracking Study of Programmers". In: *ICSE*. Hyderabad, India, 2014, pp. 390–401.

[26] Timothy R. Shaffer et al. "iTrace: Enabling Eye Tracking on Software Artifacts Within the IDE to Support Software Engineering Tasks". In: *FSE*. Bergamo, Italy, 2015, pp. 954–957.

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

[28] B. Sharif et al. "Tracking Developers' Eyes in the IDE". In: *IEEE Software* 33.3 (2016), pp. 105–108. ISSN: 0740-7459.

[29] Bonita Sharif et al. "Eye Movements in Software Traceability Link Recovery". In: *Empirical Softw. Engg.* 22.3 (June 2017), pp. 1063–1102. ISSN: 1382-3256. DOI: 10.1007/s10664-016-9486-9. URL: https://doi.org/10.1007/s10664-016-9486-9.

[30] Zéphyrin Soh et al. "Professional status and expertise for UML class diagram comprehension: An empirical study". In: *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*. IEEE. 2012, pp. 163–172.

[31] Rachel Thomas. *How to Make Tech Interviews a Little Less Awful*. 2017. URL: https://medium.com/@racheltho/how-to-make-tech-interviews-a-little-less-awful-c29f35431987.

[32] Chris Thornton et al. "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 847–855.

[33] P. Unema and M. Rotting. "Differences in eye movements and mental workload between experienced and inexperienced motor-vehicle drivers." In: *Visual Search* (1990), pp. 193–202.

[34] Erik Wstlund et al. "Effects of VDT and paper presentation on consumption and production of information: Psychological and physiological factors". In: *Computers in Human Behavior* 21.2 (2005), pp. 377 –394. ISSN: 0747-5632. DOI: http://dx.doi.org/10.1016/j.chb.2004.02.007. URL: http://www.sciencedirect.com/science/article/pii/S0747563204000202.

[35] Robert M. Yerkes and John D. Dodson. "The relation of strength of stimulus to rapidity of habit-formation". In: *Journal of Comparative Neurology and Psychology* 18.5 (1908), pp. 459–482. ISSN: 1550-7149. DOI: 10.1002/cne.920180503. URL: http://dx.doi.org/10.1002/cne.920180503.