Engineering Generalizable Features for Eye-Tracking Data Through a Cloud-Based Machine Learning Platform

Aslak Hollund and August Sollesnes Solvang

Master's Thesis in Informatics, Spring 2021

Artificial Intelligence Group
Department of Computer and Information Science
Faculty Of Information Technology, Matematics and Electrical Engineering

Table of Contents

1. Introduction

1.1. Motivation

1.1.1. Cognition

The evaluation of cognition or cognitive performance is essential for a variety of fields. Cognition describes the different workings of our mental capacities. Weintraub et al. [83] identified executive function, episodic memory, language, processing speed, working memory as the most critical cognition subdomains for health, success in school and work, and independence in daily functioning. The NIH Toolbox Cognition Batteries is a well-established set of tests of cognition [83]. While the tests in the NIH Toolbox are thorough and well tested, they are also quite involved and take much time to perform. We need a proxy if we are to evaluate cognitive performance in a micro context.

Cognitive workload describes the level of mental resources that one person requires to perform a task and naturally falls within the purview of cognitive performance. The perceived workload can be affected by many different factors such as the complexity of the task, the task's size, and external factors like how well one slept that night. Cognitive workload has also been shown to influence one's eye movements. [71, 3, 30].

Our project will include data gathered from studies handling many different cognitive tasks. We will argue that each task has some measure, such as a score, that correlates to the cognitive performance for that task.

1.1.2. Eye-tracking

Eye-tracking is the process of tracking and recording the position of a subject's eyes while interacting with digital devices. The gaze point for each of the eyes of a subject is recorded over time. Eye-tracking can be used as input control for interacting with systems [20], or to record user behavior when interacting with digital and physical systems [34, 17]. As the equipment involved has become cheaper and more readily available, eye-tracking has emerged as a promising non-invasive way to evaluate many facets of interactions with digital systems. These include collaborative work in MOOC learners [67], colaboration when interacting with Intelligent Tutoring Systems [70], detecting task-demand [71, 30].

1.1.3. Generalizability

Universality and generalism of research are important concepts, and their role in information systems research is an ongoing debate [25, 23]. Davison et al. point out that one must be explicitly aware of the context of one's research and avoid unjustified generalization [25]. Cheng et al. agree that a close look at the context of a study is critical. However, they underline that generalizability is the ultimate goal of research, and well-reasoned generalizability should be pursued [23]. Our work is acutely aware of the context of each of the datasets we use, and we will aim to engineer features that are generalizable to other contexts. We will evaluate the degree of generalizability of each of the pipelines used to produce our results through statistical analysis.

While generalizability is an oft pursued goal in machine learning, there is little focus on feature generalizability in the literature. Inspired by the methods of Sharma et al. [69], we aim to engineer features that are produced with data from one or more specific contexts and can be used to predict the same variables on datasets gathered in other contexts. The usefulness of understanding the rationale behind features and how they might differ when predicting the same target on different types of data has been studied by Rogers et al. in the context of text

mining [61]. Feature generalizability has also been considered a central goal in feature selection within music information retrieval [63]. It has also been shown to be important in predicting students' affect during learning [39].

Sharma et al. engineered generalizable features from physiological data and facial expressions. They evaluated their features with a proposed "feature generalizability index" metric. [69]. We will follow their methodology to develop generalizable features from gaze data. In addition, we will present a machine learning platform we have developed to effectively run experiments to investigate generalizability and produce features from eye-tracking data.

1.2. Research Questions

This study aims to create a system that can perform feature generalizability experiments on gaze data. To determine which features and which methods for reducing the feature space are generalizable. From this, we can extract the following research questions.

- **RQ 1** What are the most generalizable features for predicting cognitive performance using gaze data?
- **RQ 1 a** What are the most context-specific features for predicting cognitive performance using gaze data?

1.3. Terminology

In this section, we will introduce the reader to a few key terms from the eye-tracking world.

The term fixation refers to when the eye holds steady on an object or a position. In a fixation, recorded gaze points vary little in both time and space. Fixations usually reflect attention to a stimulus [71]. When discussing fixations, we will often refer to the fixation duration, which is the amount between the start of the fixation and the end of the fixation

Saccades are small rapid eye movements that occur between two fixations. It is moving the eye from one stimulus to another. Cognitive information is not processed during saccadic eye movements. However, studies have shown that they still can provide information about viewing behavior and indicate changes in cognition [71]. We will primarily refer to the measures saccade duration and saccade length. Saccade length is the euclidian distance between the two fixations. Saccade duration is the time that the eye movements take, the time between the two fixations.

Pupillary changes are changes in pupil size. We will most commonly refer to pupil diameter, which is the size in pixels or millimeters that the pupil has at a point in time. Changes to the pupil diameter can serve as a reliable proxy of mental effort [71].

1.4. Outline

Section 1: Introduction

Our introduction introduces the motivation for the work, our research questions, some key terminology, and this outline.

Section 2: Related work

The related work chapter situates our work in relation to the established literature and provides the theoretical background that outlines our assumptions.

Section 3: Study

This chapter outlines how we approach engineering generalizable features from gaze data and presents the datasets used in our experiments.

Section 4: Implementation

Our chapter on implementation presents the requirements for our platform, describes the architecture we propose to meet these requirements, and presents details on how we implement that architecture, including how and which features we engineer and how we evaluate our pipelines.

Section 5: Results

This chapter presents a range of tables and plots representing the evaluations of our pipelines and provides assistance in interpreting the presented materials.

Section 6: Discussion

The discussion delves further into the results, presents our thoughts on the patterns displayed by the results, and discusses the results in relation to the literature.

Section 7: Conclusion

At last, the conclusion presents a summary of the work, the contributions of the work, and suggestions for further work.

2. Related Work

2.1. Eye Tracking and Cognitive Performance

Since gaze data of a human can be associated with cognition [59], multiple studies have been conducted to find relationships between eye-tracking and cognitive performance. As cognitive load can be a proxy between a task and a subject's performance, [36], literature concerning gaze data and cognitive load is highly relevant for this thesis. Zhang et al. [85] classified a driver's cognitive load in two classes, using the direction of the subject's gaze and the mean and standard deviation of pupil diameter changes. They achieved significant results using a decision tree classifier. Haapalainen et al. [36] used multiple physiological sensors and combinations of their signals to determine their usefulness. Their results show that the pupillometry was one of the least valuable features for their problem. However, Steichen et al. [74] investigated how to infer cognitive abilities from gaze data and showed aggregated features from gaze data could predict a subject's perceptual speed and visual working memory reliably. Toker et al. [78] has also shown that including pupil dilation to the same feature set as Steichen has a significant effect on predictions of confusion and skill acquisition. Chen and Epps [22] did binary classification on cognitive load using a Gaussian mixture model with pupil dilation and blink frequency. Their work showed promising results for the automatic prediction of cognitive load with gaze data.

There are also novel features engineered from gaze data. Duchowski et al. [30] engineered a measure of the frequency of pupil diameter oscillation which captures an indicator of cognitive load, called Index of Pupillary activity (IPA). IPA was proposed as an alternative to the existing Index of Cognitive Activity (ICA) since ICA is not available to the public. Later they proposed an improvement to IPA called The Low High Index of pupillary activity [29], which can discriminate cognitive load in several experiments where IPA fails do so. Sharma et al. [68] used heatmaps generated from the fixtures of subjects to predict students' performance.

Features from the heatmaps were extracted with a pre-trained VGG19 model. They showed that cognitive performance could be predicted with an error rate of less than 5%.

Even though gaze-data has shown promising results in predicting cognitive workload and performance, there is little work in the literature that addresses inferring cognitive performance between contexts from gaze data.

2.2. Generalizability

While universalism and context-sensitivity in information systems research remain essential research topics, [25, 23] generalizability continues to be a chief concern in machine learning research. It is named as one of the primary goals in a slew of fields ranging from healthcare analytics [27, 9], business research [64], finance [46] to psychiatry [21]

Turney has worked to provide a formal definition of context sensitive features within concept learning [80, 81, 79]. His work is strengthened by John et al. [41] and others. While our view of context is tangential to his, the definition of context-sensitive features as features only relevant given a set context is still applicable to our work. We look to engineer generalizable features that provide predictive power on data gathered outside of the feature's context.

Bouchard et al. work with technologies for ambient intelligence to increase the autonomy of the elderly through enhanced tracking to assist medical personnel. Their work with Bluetooth beacons revealed a feature based on the received signal strength indication time-series that generalize between contexts made up of different hardware, different hardware configurations, and different floor plans [19].

Ferentzi et al. investigated whether information gained from a single interoceptive modality (mode of understanding one's own body) can be generalized to other modalities. [31] They investigated a group of students' ability to count their heartbeats, sense their gastric fullness, sense bitterness, pain threshold, proprioceptive sensitivity (ability to position their limbs), and sense of balance. The correlations

of each of the experiments were estimated used Spearman correlation [84]. They argue that their findings strongly support that interoceptive accuracy assessed with a single modality cannot be generalized to other modalities.

Feature generalizability has seen some popularity in Brain-Computer Interfaces (BCI) research. Wang et al. demonstrate generalizable features generated using a simple deep neural network that decodes learning strategies from electroencephalography (EEG) signals gathered when subjects execute a learning strategy task that included context switching. Nurse et al. also showed an approach including a high-dimension neural network used to do real-time EEG classification for use in BCI. In this approach, a neural network acted as both feature extractor and classifier, and the technique was shown to generalize [50].

Recently feature generalizability has also been shown to have relevance when predicting driver's intentions at intersections in an automotive context [57], when using ML for personality assessment [15], in speech enhancement systems [13], in face-based attention mind wandering detection [75], and in music information retrieval [63].

2.3. Platform

Machine learning platforms and pipelines are essential in many research projects. In bioinformatics, Guzzetta et al. present an L1L2 based machine learning pipeline that is effective for fitting quantitative phenotypes from genome data [35]. Fountain-Jones et al. demonstrate how a flexible ensemble pipeline can be used for predicting pathogen exposure in animal ecology. To demonstrate their pipeline, they model pathogen exposure risk in two different viruses that infect African lions [32] Sutton et al. have developed PhysOnline, a pipeline built on the Apache Spark platform. PhysOnline uses streaming physiological data and can be used for real-time classification in the biomedical and health informatics field [76] Shaikh et al. tackle the complicated task of creating a machine learning pipeline that can ensure fairness in decision-making. The system can understand policies written in natural language and will assist users in ensuring that set fairness policies are followed [66].

There has been some research that focuses on the pipelines themselves. Olson and Moore present TPOT, a tree-based tool for optimizing pipelines, which has recently shown much promise [52]. This system exists in the new and promising field of AutoML, which seeks to automate the design of pipelines for machine learning, democratizing machine learning further. Mohr et al. have contributed to predicting pipeline runtimes to improve one of the significant limitations of current AutoMl technology [47]. The systems we have available today take a considerable amount of time to arrive at good pipeline designs if the case is even slightly complex. Effectively predicting pipelines would enable us to significantly decrease the cost of AutoML systems, as one could ignore pipelines that would not complete within the systems timeout limits.

There have also been several attempts at creating more generally applicable platforms for machine learning. López García et al. presents the DEEP-Hybrid-DataCloud framework as "a distributed architecture top provide machine learning practitioners with a set of tools and cloud services to cover the whole machine learning

development cycle" [45]. The DEEP framework represents an important step in democratizing machine learning and deep learning for all areas of research. Ribeiro et al. present an architecture to provide scalable machine learning capabilities as a service, along with an open-source implementation [60]. Kraska et al. have developed MLBase, which includes a simple declarative language for specifying machine learning tasks to be used by both researchers and end-users [42]. The available resources for machine learning in proposed architectures, open-source and commercial platforms and pipelines, tooling, and other supporting technologies have exploded in the last years. Machine learning tools to process large-scale data for decision assistance, automation, and interactive data analytics are growing and will continue to be essential for research and business [49].

The ability to reproduce a scientific works central finding, reproducibility, is key to the scientific process, and as such is essential in information systems research [58, 55, 65, 62]. This challenge is both more complicated and more straightforward in the age of digital. Analyses that can be made are made on larger, more complex datasets and with ever more complex digital tools supporting the analysis. While this added complexity adds to the difficulty in adequately describing the process involved in reaching our conclusions, digital tools can also give us the ability to share the exact code that confirms our results' [58, 40]. There has been a growing push to include code with materials published along with scientific findings [40, 11]. Ince et al. argue that even with the complete source code, one is not guaranteed reproducibility. Local software and hardware environments introduce noise that could impact results. They argue for including descriptions of the environments in which the code was executed [40]. With the advent of containerization and commercial cloud platform, it is possible to reach close to the same descriptive rigor of the hardware and software environments as sharing code represents for the system. [28, 48, 82]

3. Study

The study design, described below, is inspired by the study performed by Sharma et al. [69].

We have selected three datasets used in published articles that all include eye-tracking data from subjects completing one or more cognitive tasks. The studies that produced the datasets were distinct, and their differences allow us to argue that the datasets have different contexts. With these datasets, we generate several different pipelines consisting of different feature groups, different dimensionality reduction or feature selection techniques, and different combinations of the datasets. These pipelines are then evaluated using both **out-of-sample testing** to determine the predictive power on unseen data and **out-of-study testing** to assess the generalizability of the pipeline.

For each pipeline, we designate either one or two datasets as in-study and one dataset as out-of-study. In the cases where two datasets are selected as in-study, they are combined into one larger dataset. If we use only one dataset as in-study, we do not include the third dataset in that pipeline.

We split the in-study dataset into three parts: training, validation, and testing data by leave-one-participant-out. The training data is used to train the model; the validation data is used to set weights for the classifiers in the voting ensemble, and the testing data is used to evaluate the predictive power of each pipeline on unseen data. The testing of the pipeline on unseen data from the same dataset(s) is our **out-of-sample testing**.

Our next step is the **out-of-study-testing**. In this step, we use the model outputted from a pipeline to make predictions on data gathered in separate contexts. This allows us to analyze the generalizability of the pipeline.

3.1. Datasets

We have been working with three different datasets gathered and published by other researchers. They are all gathered during cognitive tasks and, as such, are suitable for investigating cognitive workload. We have used the shorthand names EMIP, CSCW, and Fractions to refer to the datasets. They are further described in the following section.

3.1.1. EMIP

The Eye-Movements In Programming (EMIP) dataset is a large eye-tracking dataset collected as a community effort involving 11 research teams across four continents. The goal was to provide a substantially large dataset open and free to stimulate research relating to programming and eye-tracking [14].

Participants

Each site recruited participants by opportunity sampling. Data from 216 participants at differing levels of expertise is included in the dataset. There were 41 female and 175 male participants; their mean age was 26.56 with a standard deviation of 9.28. The participants were primarily university students enrolled in computing courses but included academic and administrative staff and professional programmers [14].

Tasks

The participants were recorded while performing two code comprehension tasks. They filled out a questioneer where they self-reported their level of programming expertise from the options none, low, medium, and high, recorded years of programming experience. In addition, they answered several other questions that make up the metadata that accompanies the eye-tracking data. The participants also selected the language to be used in the task, from Java, Scala, or Python.

After the questioner, the eye-tracker was calibrated, and the code comprehension task was started. The task consisted of reading two different pieces of code called Vehicle and Rectangle, each comprising 11-22 code lines. After reading and trying to understand a piece of code, the participants would indicate that they were ready to proceed by pressing the space bar. Next, the system presented the participants with a multiple-choice question that evaluated code comprehension. After completing the question for Vehicle, they were presented with the code for Rectangle and subsequently its accompanying question.

The rectangle code consists of a class representing a rectangle, instantiated through a constructer that takes four coordinates. The class also has methods to width, height, and area. In the *main* function of the class, two rectangles are instantiated, and their areas are printed. The code was based on a code comprehension task developed by Hansen [37], and the EMIP researchers translated code from python two Scala, and Java. The vehicle code is a class that represents a vehicle with several variables and a function to accelerate the car. A constructor takes a producer, a type, and a top speed. In the *main* function, a vehicle is created, and its speed is accelerated [14].

Technology and experimental setup

The recording was performed using a screen-mounted SMI RED25 mobile video-based eye tracker. The tracker has a sample rate of 250 Hz with an accuracy $< 0.4\circ$ and a precision of $\approx 0.03\circ$ of visual angle. Stimuli were presented on a laptop computer screen with a resolution of 1920 x 1080 pixels and were viewed without a headrest in a fashion that closely simulated a familiar programming environment. The data collection procedure was implemented in the SMI Experimental Suite software. The laptop, eye-tracker, and software were shipped between the locations to minimize differences in setup [14].

Description of the data

Below is a list of the contest of the dataset provided by Bednarik et al. [14].

Contents of the dataset

- rawdata: a folder with 216 TSV files containing raw eye movement data.
- stimuli: screenshots of the experiment slides in JPG-format and CSV files with AOI (area of interest) coordinates for the stimulus program
- metadata: a CSV file with information with background information on the participants, results from the questioner, and the order in which the stimulus programs were shown.
- data: TXT file specifying when the dataset was uploaded.

3.1.2. CSCW

The CSCW dataset was gathered during the Computer-Supported Cooperative Work and Social Computing course at École Polytechnique fédérale de Lausanne. The study intended to show the different gaze patterns across learners in Massive Open Online Courses (MOOCS) and study the priming effect on learning [67].

Participants

The dataset contains gaze data from 98 students in the CSCW course at EPFL. There were 49 participants in each of the two priming groups [67].

Tasks

The experiment centered around a collaborative task with a contextual primer. Participants were presented with pre-tests, which also served as primers. They were given either a schematic or a text-based primer. The textual primer had questions described in text-form, while the schematic primer had the same questions but presented as a schema. Participants that received the textual primer were called T, and participants that received the schematic primer were called S.

After the primer, participants watched a video from Khan Academy [5] on the topic of "resting membrane potential." Arranged in 16 TT pairs, 16 SS pairs, and 17 ST pairs, each team collaborated to create a concept map using IHMC CMap tools [4]. Lastly, the participants also completed a post-test [67].

Technology and experimental setup

Gaze data was recorded using SMI RED250 eye trackers [2]. The participants were while they completed watched the video and during the collaborative concept mapping task [67].

Description of the data

The dataset contains two files with gaze data for each participant. One file describes the video watching phase, and one part describes the concept mapping phase. We will not be considering any links between the two and will treat them as separate. While the concept mapping task was cooperative, all measurements are individual. We will be working with the data on the individual level [67].

3.1.3. Fractions

The dataset that we refer to as Fractions was gathered by Olson et al. [51]. It is an eye-tracking dataset from an experiment intending to investigate the differences between individual and collaborative performance when working on conceptually or procedurally oriented problems in a intelligent tutoring system (ITS) designed to teach fractions.

Participants

The study was conducted with 84 4th and 5th graders from two US elementary schools in the same school district. The students left their regular instruction during the school day to participate in the study. Teachers from the student's classes paired the students based on their mathematical abilities and who would work well together. Before participating in the experiment, the students worked with the Fractions Tutor during two of their regular classes to acclimatize them to the software. The pairs of students were randomly assigned to four groups completing different tasks. They where: collaborative conceptual, collaborative procedural, individual conceptual and individual procedural. Twice as many pairs were assigned the collaborative tasks as the individual [51].

Tasks

Olsen et al. hypothesized that students working collaboratively would show learning gains on both procedural and conceptual tasks, and that of those working on conceptual tasks, students working collaboratively would have stronger learning gains than those working individually. They also hypothesized that students working individually would have greater learning gains than those working cooperatively for procedural tasks.

To investigate these hypotheses the pairs of students worked with their assigned tasks in an ITS. The tasks used different techniques to assist the students in learning equivalent fractions. Participants also completed a pre-test on the morning of the experiment and a post-test the next day [51].

Technology and experimental setup

Students participating in the study completed their tasks in an interactive tutoring system developed by the researchers. They communicated verbally through a skype connection. No video signal was transmitted. Gaze data recorded using SMI RED250 eye trackers [2, 51].

Description of the data

The data includes individual files with gaze data for each student as well as a file describing all the results from the pre and post-tests. Our dataset consists of only the data used by Sharma et al. [70]. This only includes the data from the pairs that worked on the collaborative tasks, not the students that worked individually.

3.2. Contexts

Our work seeks to investigate generalizability between specific contexts; thus, we must be aware of our contextual biases. We have gathered datasets that we consider to cover a significant spectrum of cognitive processes.

EMIP is an individual task that is about reading and understanding programming code. We hypothesize that the task in EMIP relies primarily on three of the cognitive subdomains deemed most critical by Weintraub et al. [83]. Reading and understanding code is a trained skill reliant heavily on ones understanding of language. Understanding the entirety of a class requires keeping all functions of the class in one's working memory. The post-test is organized so that one needs to remember these functions for a short time after reading the code. As with almost all cognitive tasks, attention is a critical part of performing well when reading and understanding code.

CSCW is a task where participants collaborate in creating a concept map from a video they have watched individually. Naturally, language will be an important part of any collaborative work as one needs to express one's understanding of the content to one collaborator. Executive function, specifically planning, is essential for creating concept maps. Concept maps include tying different pieces of information together in a complete whole. Before starting the concept map, they viewed a video explaining the concept they were to map. In order to remember information presented in the videos, the cognitive subdomain episodic memory is at work.

Again, attention is essential when viewing a video for learning and successful collaboration with another party.

The Fractions dataset also stems from a collaborative task. Students work together to learn about equivalent fractions in an ITS. For the collaborative aspect, attention and language are again important.

All cognitive tasks likely include some aspect of all cognitive subdomains. What we intend with this section is to illustrate how our three datasets cover tasks that rely more heavily on five of the six cognitive subdomains presented by Weintraub et al. [83] as most important. Of the six cognitive subdomains, our datasets do not include tasks that rely heavily on processing speed. Processing speed is an important factor in good collaboration, but we did not consider this subdomain to be as central in any of the tasks and thus will not claim to cover it with these datasets.

4. Implementation

Our goal with this system is to create a platform on which we can perform our feature generalizability experiments efficiently and consistently.

In order to achieve this goal, multiple components have to be present.

1. We need methods to standardize datasets so the units are the same and the data is in the same form.

2. We need to clean the data to achieve high data quality, which can produce good features.

3. We need a platform that can generate computationally expensive features for multiple large datasets.

4. We need a platform that can run multiple concurrent pipelines for combinations of datasets, features, and methods for dimensionality reduction.

5. We need an evaluation step that collects the results from all the pipelines and can prove pipelines generalizable.

6. We need reproducibility.

Figure 1 shows the outlines of the architecture we propose. The first step is data preprocessing, explained in Section 4.1. Preprocessed data is then fed to the feature extraction step explained in Section 4.2. Then we organize 219 separate pipelines that consist of unique combinations of datasets, feature groups, and methods for reducing the feature space explained in Section 4.3. All pipelines use the same ensemble classifier. Results are then evaluated in the evaluation step, Section 4.4, and logged Section 4.5.

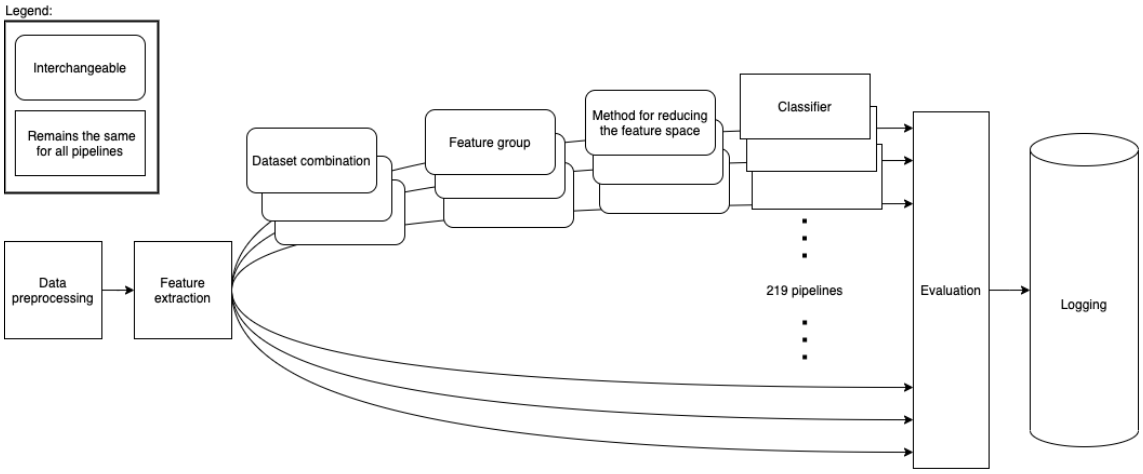

Figure 1. Diagram of our architecture

4.1. Preprocessing

This subsection explains how we achieved goals 1 & 2 of creating a platform for generating generalizable features.

> We need methods to standardize datasets so the units are the same and the data is in the same form.
>
> We need to clean the data to achieve high data quality, which can produce good features.

4.1.1. Standardization of Datasets

We use three datasets from different experiments with different contexts. Each dataset has its own set of column names, and they also use different units. Some of the datasets measure time in milliseconds, while others measure it in microseconds. In standardizing the dataset, we converted all differing units and renamed columns that represented the same values so that all datasets are consistent with each other. Some subjects were missing labels; this was solved by removing the subject from the dataset. We also fixed inconsistencies such as wrong capitalizations of filenames.

For the EMIP dataset, we were provided with the raw data without fixations calculated. In order to use this dataset, we calculated fixations ourselves with the library PyGaze Analyzer [6]. The algorithm used is a dispersion-based algorithm that requires us to set the maximum distance and minimum duration for a fixation. We set the minimum duration by finding the minimum duration of fixations from our two other datasets. Pieter Blignaut [16] suggests that the maximum distance of a fixation when using a dispersion algorithm for finding fixations should be 1º of the foveal angle. The distance between the subject and the stimulus was 700 mm for EMIP; thus, 1º of the foveal angle works out to about 45 pixels. We only used fixations where the subject was reading code and disregarded data gathered during setup and calibration.

4.1.2. Normalization and Outlier removal

As our subjects come from multiple contexts, the need for normalization and outlier removal is extra apparent. It is essential to normalize pupil diameter. Pupil diameter can be affected by several factors such as lighting conditions and how well-rested the subject is [56]. We chose to min-max normalize the pupil diameter in the range of 0 to 1 per subject.

Our time recordings are made at the start point of each fixation. This can be problematic, as there are situations where more time passes between fixations than would be reasonable for regular saccades. This might be due to blinking, the subject looking outside the range of the eye-tracker, or technical malfunction. To mitigate this, we remove the outliers by setting a threshold of 1000 ms for saccade duration. All gaps in time over 1000ms were reduced to 1000ms.

4.2. Feature extraction

This subsection explains how we achieved goal 3.

> We need a platform that can generate computationally expensive features for multiple large datasets

4.2.1. Flow

The flow of the feature extraction job is as follows:

1. Download the datasets from google cloud storage
2. Load data as a list of pandas data frames
3. Standardize data. This step is different for each dataset
4. Normalize data
5. Generate aggregated attributes, such as saccade duration, saccade_length, and angle of saccades.
6. Take the rolling mean of the signals
7. Generate features
8. Upload features to google cloud storage

4.2.2. Features

The features we generate can be separated into three different groups.

- Timeseries Features
- Eyetracking Features
- Heatmap features

4.2.3. Timeseries Features

Timeseries features are features that are agnostic to the source of data. They are a description of the signal rather than the meaning behind the signal. The signals we describe with timeseries features are pupil diameter, fixation duration, saccade duration, and saccade length.

From each of these signals, we calculate five features.

- Power Spectral Histogram.
- ARMA.
- GARCH.
- Hidden Markov models.
- LHIPA.

Power Spectral Histogram.

The power spectrum decomposes the timeseries into the frequencies in the signal and the amplitude of each frequency. Once computed, they can be represented as a histogram called the power spectral histogram. We computed the centroid, variance, skew, and kurtosis of the power spectral histogram.

The power spectral histogram describes the repetition of patterns in the data. We hypothesize that similar pattern repetitions will be present in subjects that display a high cognitive performance across contexts.

Autoregressive Moving Average model (ARMA)

We know that a cognitive process takes place over a span of time, and our gaze data is organized as a timeseries. If our goal is to model cognitive workload, we need to capture the temporal aspects of cognition. ARMA combines an auto-regressive part and a moving average part to model movements in a timeseries. It uses information for previous events to predict future events with the same source.

An ARMA process describes a time series with two polynomials. The first of these polynomials describes the autoregressive (AR) part of the timeseries. The second part describes the moving average (MA). The following formula formally describes ARMA:

$$X_t = \sum_{j=1}^{p} \phi_j X_{t-j} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

The features we extract from ARMA are extracted with the following algorithm

```
best_fit = null
for p up to 4
    for q up to 4
        fit = ARMA(timeseries, p, q)
        if(best_fit.aic > fit.aic)
            best_fit = fit
return best_fit["ar"], best_fit["ma"], best_fit["exog"]
```

GARCH

GARCH is similar to ARMA, but it is applied to the variance of the data instead of the mean.

ARMA assumes a uniform distribution of the events it is modeling with shifting trends. However, we know that this is not entirely descriptive of cognitive processes. When working with a cognitive task, one has periods of intense focus to understand one aspect of the problem and other periods where one's mind rests. The heteroskedasticity aspect of GARCH accounts for this grouping of events. In the analysis of financial markets, GARCH can account for the spikes and sudden drops in prices of specific investment vehicles [18], we posit that GARCH will allow us to model the perceived grouping of cognitive workload spikes in a like manner.

We extract features from GARCH similar to how we extract features from ARMA.

```
best_fit = null
for p up to 4
    for q up to 4
        fit = GARCH(timeseries, p, q)
        if(best_fit.aic > fit.aic)
            best_fit = fit
return [best_fit["alpha"],
       best_fit["beta"],
       best_fit["gamma"],
       best_fit["mu"],
       best_fit["omega"]]
```

Hidden Markov Models

Hidden Markov Models contains the Markov assumption, which assumes that the present value is conditionally independent on its non-ancestors given its n last parents. Similarly to ARMA, this means that the current value is a function of the past values. While ARMA models continuous values, HMM is discrete. Hence we discretize the values into 100 buckets before we model the transitions between these buckets with HMM. We then use the resulting transition matrix as the feature.

The reasoning behind using HMM is the same as to why we chose ARMA. We hypothesize that HMM might model the changing state nature of cognitive work.

```
normalized_timeseries = normalize the timeseries between 0 and 1
discretized_timeseries = discretize timeseries in 100 bins
best_fit = null
for i up to 8
   fit = GaussianHMM(covariance_type="full").fit(discretized_timeseries,
   if(best_fit.aic > fit.aic)
           best_fit = fit
flat_transistion_matrix = best_fit.transition_matrix.flatten()
padded_transition_matrix = pad flat_transistion matrix with n zeroes so
return padded_transition_matrix
```

The Low/High Index of Pupillary Activity (LHIPA)

LHIPA [29] enhances the Index of Pupillary Activity [30], which is a metric to discriminate cognitive load from pupil diameter oscillation. LHIPA partially models the functioning of the human autonomic nervous system by looking at the low and high frequencies of pupil oscillation.

Cognitive load has been shown to correlate with cognitive performance [38]. The Yerkes-Dodson law describes the relationship between the two, indicating an optimal plateau where a certain degree of cognitive workload is tied to maximized cognitive performance. If the cognitive workload is increased beyond this point, cognitive performance is diminished [38].

Our implementation of LHIPA is based on the code found in [30, 29]

4.2.4. Eyetracking features

These are features that are connected to the domain of eye-tracking and not signal processing features.

Information Processing Ratio

Global information processing (GIP) is often synonymous with skimming text. When skimming, one's gaze jumps between large sections of the material and does not stay in place for extended periods. This manifests as shorter fixations and longer saccades. Local information processing (LIP) is the opposite; one's gaze focuses on smaller areas for longer durations and does not move around as much. For this metric, fixations are measured in time, while saccades are measured as distance. Hence we capture both the spatial and temporal dimensions.

The information processing ratio describes how much a subject skimmed the material versus how much they focus intently. To compute the ratio, we divide GIP by LIP.

The following algorithm extracts the feature:

```
upper_threshold_saccade_length = get 75 percentile of saccade_lengths
lower_threshold_saccade_length = get 25 percentile of saccade_lengths
upper_threshold_fixation_duration = get 75 percentile of fixation_durati
lower_threshold_fixation_duration = get 25 percentile of fixation_durati

LIP = 0
GIP = 0
for saccade_length, fixation_duration in saccade_lengths, fixation_durat
    fixation_is_short = fixation_duration <= lower_threshold_fixation_du
    fixation_is_long = upper_threshold_fixation_duration <= fixation_dur
    saccade_is_short = saccade_length <= lower_threshold_saccade_length
    saccade_is_long = upper_threshold_saccade_length <= saccade_length

    if fixation_is_long and saccade_is_short:
        LIP += 1
    elif fixation_is_short and saccade_is_long:
        GIP += 1

return GIP / (LIP + 1)
```

Skewness of saccade speed

Saccade velocity skewness has been shown to correlate with familiarity [53]. If the skewness is highly positive, that means that the overall saccade speeds were high. This indicates that the subject is familiar with the stimulus and can quickly maneuver to the relevant sections when seeking information. Saccade speed does not necessarily indicate expertise in the relevant subject matter. A subject could be familiar with the material and hence know where to look for information, but an expert would also quickly assert what information they are seeking.

To calculate this feature, we calculated the speed by dividing the saccade length by the saccade duration. We then got the skew of the distribution outputted.

```
get_skewness_of_saccades(saccade_duration, saccade_length):
    saccade_speed = saccade_length/saccade_duration
    return saccade_speed.skew()
```

Verticality of Saccades

By verticality of saccades, we mean the ratio of which saccades move vertically over horizontally. Our intuition for generating this feature is based on the difference between how we read code versus how we read text. An experienced coder is read vertically, focusing on definitions and conditionals. Traditional text, on the other hand, is read line by line in a horizontal fashion. Based on this anecdotal observation, we were interested in how well the verticality of saccades would generalize.

To calculate the feature, we get the angle between every consecutive fixation with respect to the x-axis. We do that with arctan2, which outputs the angle in radians between pi and -pi. Since we are only interested in the verticality of the saccade, we take the absolute value of the angle.

```
radians = atan2(y2 - y1, x2 - x1)
return abs(radians)
```

To describe the horizontality of each point in a range between 0 and 1, we take the sine of every angle.

```
for angle in angles
   angle = sin(angle)
```

Then we average all the sinus values.

```
verticality = angles.average()
```

Entropy of Gaze

The entropy of gaze explains the size of the field of focus for a subject. Entropy is higher the subject's attention is more evenly spread over the stimulus and lower if the subject focuses on a minor part of the stimulus.

To calculate entropy, of gaze we create a grid of 50 by 50 px bins. We then normalize the x and y positions of each fixation in a range from 0 to 1000. Further, we place each fixation in one of these bins based on which bin its x and y position corresponds to. When we have this grid, we flatten it and take the entropy of the resulting distribution.

The following algorithm extracts the feature:

```
x_normalized = normalize x between 0 and 1000
y_normalized = normalize y between 0 and 1000

x_axis = [50, 100, 150 ... ,1000]
y_axis = [50, 100, 150 ... ,1000]
2d_histogram = 2d_histogram(xaxis, yaxis, x_normalized, y_normalized)
return entropy(2d_histogram.flatten())
```

4.2.5. Heatmaps

A heatmap is a graphical representation of data where values are depicted by color. Areas of higher activity will be highlighted more. Our heatmaps represent the gaze position of a subject over time. To capture both spatial and temporal data, we create multiple heatmaps for each subject.

These are the steps we take to create our heatmaps

1. Split the data from each subject into 30 partitions

2. Create a 1920 * 1080 image

3. Plot the gaze position with heatmappy [12]

4. Resize the image to 175*90

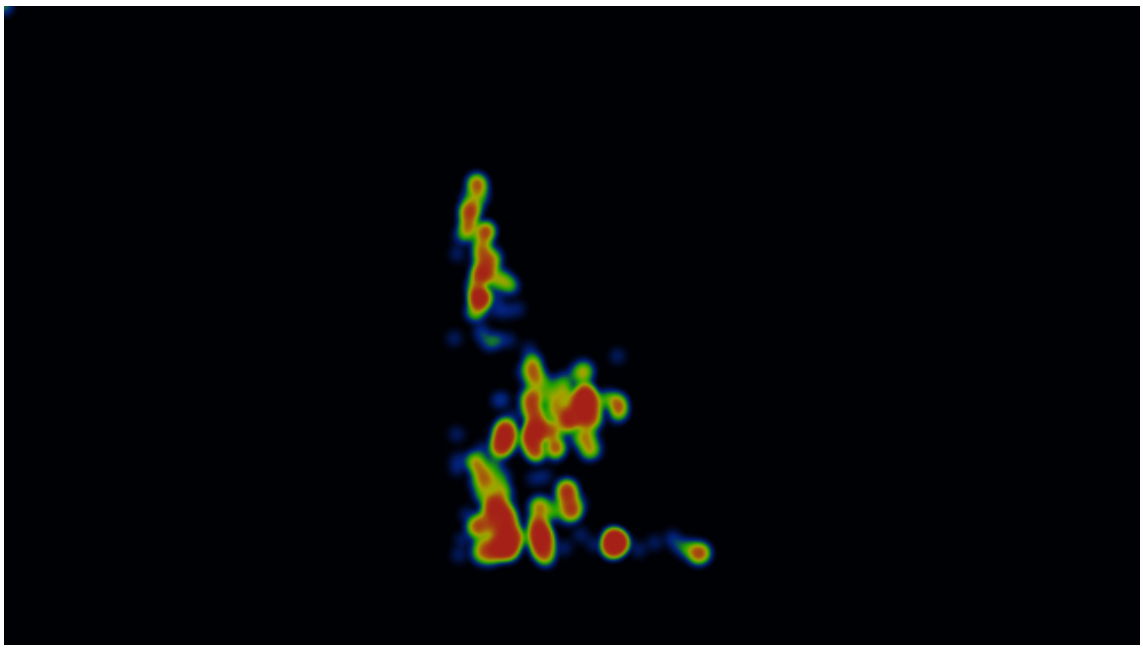

Figure 2. Heatmap without stimulus from EMIP

From the heatmaps used a pretrained vgg19 model [72] with the imagenet weights [43] to generate a feature vector per image.

1. Scale the images down using the preprocess_input function found in `keras.applications.image_netutils`

2. Use the pre-trained VGG-19 model to extract features per image

3. Flatten the matrices outputted by the VGG19 model to a single list of values and concatenate them.

As shown by Sharma et al. [68], deep features of heatmaps from gaze data can predict cognitive performance in learning activities.

Pseudocode

```
frames = Split each subject into 30 partitions
features = []
for frame in frames
    image = image of with dimensions 1920, 1080
    heatmap = heatmappy.heatmap_on_img(frame.x_and_y_postions, image)
    scaled_down_heatmap = keras.applications.image_netutils(heatmap)
    heatmap_features = vgg19model.predict(scaled_down_heatmap)
    features.append(heatmap_feature.flatten())
return features.flatten()
```

4.2.6. Final Feature set

After feature extraction, these are the features that are generated for each subject.

Name	**Description**
Information Processing Ratio	LIP divided by GIP
Saccade Speed Skewness	Skewness of the saccade speed distribution
Entropy of Gaze	Entropy of the gaze
Verticality Of Saccades	Metric between 0, 1 describing the average angle of the saccades
Heatmaps	Features extracted from heatmaps on VGG19

Spectral Histogram - Pupil Diameter	Skew, Kurtosis, Mean, and Variance of the power spectral histogram
LHIPA - Pupil Diameter	The Low/High Index of Pupillary Activity
HMM - Pupil Diameter	Transistion matrix of fitted markov model
ARMA - Pupil Dialation	Ar, ma, and Exog attributes of a fitted ARMA model
GARCH - Pupil Diameter	mu, omega, alpha, gamma, and beta attributes of a fitted Garch model
Spectral Histogram - Fixation Duration	Skew, Kurtosis, Mean, and Variance of the power spectral histogram
HMM - Fixation Duration	Transistion matrix of fitted markov model
ARMA - Fixation Duration	Ar, ma, and Exog attributes of a fitted ARMA model
GARCH - Fixation Duration	mu, omega, alpha, gamma, and beta attributes of a fitted Garch model
Spectral Histogram - Saccade Length	Skew, Kurtosis, Mean, and Variance of the power spectral histogram
HMM - Saccade Length	Transistion matrix of fitted markov model

ARMA - Saccade Length	Ar, ma, and Exog attributes of a fitted ARMA model
GARCH - Saccade Length	mu, omega, alpha, gamma, and beta attributes of a fitted Garch model
Spectral Histogram - Saccade Duration	Skew, Kurtosis, Mean, and Variance of the power spectral histogram
HMM - Saccade Duration	Transistion matrix of fitted markov model
ARMA - Saccade Duration	Ar, ma, and Exog attributes of a fitted ARMA model
GARCH - Saccade Duration	mu, omega, alpha, gamma, and beta attributes of a fitted Garch model

4.3. Pipelines

This section explains how we solved goal 4 of creating our platform.

> We need a platform that can run multiple concurrent pipelines for combinations of datasets, features, and methods for reducing the feature space.

By a pipeline, we mean a specific combination of datasets, feature groups, and methods for reducing the feature space (dimensionality reduction or feature selection). For simplicity, we will refer to these as pipeline components.

4.3.1. Datasets

We have three different datasets: EMIP, Fractions, and CSCW. As discussed in Section 3, we designate either one or two datasets as in-study for each of our pipelines. All pipelines include a single dataset as the out-of-study dataset. No dataset is used twice in one pipeline.

We refer to the pipelines where one dataset is designated in-study as 1-to-1 pipelines, as these pipelines train on one dataset and test on another. Pipelines, where two datasets are designated in-study, are referred to as 2-to-1 pipelines since we train on two datasets combined and test on one dataset. We have three datasets, which make up nine dataset combinations, six of which create 1-to-1 pipelines and three go into 2-to-1 pipelines.

4.3.2. Feature groups

Initially, we considered running pipelines to test all combinations of features. These would prove to be unfeasible. With 22 features, 9 dataset combinations, and two methods of reducing the feature space, there would be 75 597 472 pipelines. With a theoretical runtime of 1 second per pipeline, the total computing time necessary to tackle this challenge would be approximately 2 years, and we would not make our deadline for this thesis.

As an alternative, we decided to group our features manually. We created one group for the eye-tracking features, four for the type of signal, and five for the different time series features. In addition, we have a separate group for heatmaps and, lastly, one group with includes all the features. The groups are presented in the following table:

Name	**Features**

Eye tracking Features	Information Processing Ratio, Saccade Speed Skewness, Entropy Of Gaze, Verticality Of Saccades
Heatmaps	Heatmaps
Power Spectral Histogram	Spectral Histogram - Pupil Diameter, Spectral Histogram - Fixation Duration, Spectral Histogram - Saccade Length, Spectral Histogram - Saccade Duration
LHIPA	LHIPA - Pupil Diameter
Markov	HMM - Fixation Duration, HMM - Pupil Diameter, HMM - Saccade Duration, HMM - Saccade Length
ARMA	ARMA - Pupil Diameter, ARMA - Fixation Duration, ARMA - Saccade Length, ARMA - Saccade Duration
Garch	GARCH - Saccade Duration, GARCH - Fixation Duration, GARCH - Pupil Diameter, GARCH - Saccade Length

Pupil diameter	Spectral Histogram - Pupil Diameter, LHIPA - Pupil Diameter, HMM - Pupil Diameter, ARMA - Pupil Diameter, GARCH - Pupil Diameter
Fixation Duration	Spectral Histogram - Fixation Duration, HMM - Fixation Duration, ARMA - Fixation Duration, GARCH - Fixation Duration
Saccade Length	Spectral Histogram - Saccade Length, HMM - Saccade Length, ARMA - Saccade Length, GARCH - Saccade Length
Saccade duration	Spectral Histogram - Saccade Duration, HMM - Saccade Duration, ARMA - Saccade Duration, GARCH - Saccade Duration
ALL	ALL

4.3.3. Reducing the feature space

Our pipelines perform either feature selection or dimensionality reduction to reduce the number of features and improve predictive power. The focus of our thesis was on testing different features, and as such, we decided not to test a wide range of alternatives. The effect of different methods for reducing the feature space on generalizability is a potential area for further study. The method we use for dimensionality reduction is PCA, and the one for feature selection is LASSO. For all pipelines, we also use a zero-variance filter to remove the features that have no variance in their data

Lasso was selected as our feature selection algorithm because it has been shown to perform very well when the number of samples is less than the number of features [77, 33], which is the case for most of our feature groups.

4.3.4. Prediction: Ensemble Learning

Our pipelines were tested with the same classifier, a weighted voting ensemble with a KNN-regressor, a Random forest regressor, and a Support Vector regressor. To find the weights for the voting, we perform cross-validation on each of the regressors and set their respective weights as 1 - Root Mean square error. We use an ensemble because it has been shown to improve stability and robustness, capture linear as well as non-linear relationships, and the variance in our datasets suggests that a single model would perform inadequately [69].

4.4. Evaluation

This section will outline how we achieve the fifth goal of our platform:

> We need an evaluation step that collects the results from all the pipelines and can prove pipelines generalizable.

Models produce by our pipelines are evaluated in two ways, with out-of-sample testing and out-of-study testing. Out-of-sample testing uses a subset of the in-study dataset to evaluate the predictive power of the model. Out-of-study testing uses the dataset designated as out-of-study to evaluate generalizability. This subsection explains how we evaluate the results of each test.

4.4.1. Evaluation Criteira

We use normalized root mean squared error (NRMSE) as our primary evaluation metric. Since the datasets' labels are in different ranges, we need a metric that can be compared across datasets. NRMSE is normalized between 0 and 1, where 1 is the maximum error possible for the dataset. NRMSE is commonly used in learning technologies to evaluate learning predictions [1] and is the preferred metric for student models. [54] Since we normalize the labels in a range from 0 to 1 before training, RMSE is equivalent to NRMSE in this thesis. Hence we refer to it as NRMSE in this thesis.

The following formulas calculate NRMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{\text{Number of samples}} \left(\text{predicted}_i - \text{original}_i\right)^2}{\text{Number of Samples}}}$$

$$NRMSE = \frac{\text{NRMSE}}{\text{original}_{\text{max}} - \text{original}_{\text{min}}}$$

4.4.2. Feature Generalizability Index (FGI)

We measure the generalizability of features by comparing the distributions of NRMSE values from the out-of-sample testing and the out-of-study testing, as done in [69]. Since the ground truth in both the in-study datasets and the out-of-study datasets indicate cognitive performance, we can safely assume that similar behavior will indicate similar results. To compare the distributions of NRMSE, we use analysis of similarity (ANOSIM), which is a non-parametric function, to find the similarity of two distributions [24].

$$\frac{\text{mean ranks between groups} - \text{mean ranks within groups}}{N} \frac{N-1}{4}$$

The denominator normalizes the values between -1 and 1, where 0 represents a random grouping. Pipelines that have an FGI value closer to zero are more generalizable.

4.5. Reproducability

This section explains how we reached the sixth goal of our platform.

> We need reproducibility.

Our reproducibility strategy primarily consists of three components. The version-control tool, git; the machine learning management tool, comet.ml; the python package management tool, poetry; and google cloud platform.

4.5.1. comet.ml

comet.ml is a machine learning management tool [10]. It can handle user management, visualization, tracking of experiments, and much more. We use comet.ml to track the results of our experiments, the relevant hyperparameters, the git commit hash, which indicates our software state, and the virtual machine on the google cloud platform which executed the code.

4.5.2. Poetry

Poetry is a dependency manager for python [8]. As with any large software project, our platform relies on several third-party libraries. To ensure both reproducibility and ease of use and development, we use poetry to track the versions of the libraries we use. Poetry stores these versions in a lock-file which git tracks.

4.5.3. Git

Git keeps track of all versions of our source code [7]. We have set up safeguards that ensure that all changes to the local code are committed before a run can begin. In addition, all parameters of experiments are represented in the code. As a result, the complete state of the software, including configuration and hyper-parameters, is recorded in git. The commit hash, which uniquely identifies the point of commitment in our history, is logged, and we can reproduce the software side of our experiment.

When we run an experiment in the cloud, we log the start parameters of the system and the hash associated with the commit.

4.5.4. Google Cloud Platform

Our experiments are run on virtual machines in the google cloud platform (GCP) [44]. GCP is one of several providers of commercial cloud and containerization services. Their products allow us to run our experiments on identical virtual machines, which ensures reproducibility also in the hardware aspects of our work.

4.5.5. Seeding

All of our experiments ran with seeded randomness. Our implementation for seeding

```
seed = 69420
np.random.seed(seed)
random.seed(seed)
os.environ["PYTHONHASHSEED"] = str(seed)
tf.random.set_seed(seed)
```

4.5.6. Datasets

At this point, we can reproduce any of the experiments presented in this work. However, as we cannot share the data we received from other researchers, complete external repeatability is impossible.

5. Results

We ran a total of 216 pipelines with different combinations of datasets, dimensionality reduction, and feature groups. 144 of these pipelines were 1-to-1 pipelines, where we trained on only one dataset and tested on another. 72 of the pipelines were 2-to-1, where two datasets are designated in-study and one dataset designated out-of-study. In this section, we will present the results from these pipelines. In Section 5.1, we explain how baselines for the different datasets are calculated. Section 5.2 presents how pipelines performed in out-of-sample testing and which components performed the best overall by aggregating the results. The pipelines that show generalizability are presented in Section 5.3. The section also presents which components make up those pipelines. Pipelines that exhibit more context sensitivity are presented in Section 5.4.

5.1. Baselines

We calculate a specific for each dataset combination. We use the NRMSE value equivalent to predicting the mean label value from the in-study dataset as the baseline for each combination. Our pipelines are evaluated against the baseline for the given dataset combination.

Dataset	**Baseline (rmse)**
cscw	0.205
Emip	0.310
Fractions	0.229

Emip & Fractions	0.294
Emip & cscw	0.289
cscw & Fractions	0.234

Pseudocode for the in-study baseline:

```
def get_baseline(labels):
    error = labels - labels.mean()
    error_squared = (error**2).mean()
    baseline = math.sqrt(error_squared)
    return baseline
```

5.2. Out-of-sample testing

Our work focuses on engineering generalizable features. This section outlines the results from the out-of-sample testing, which does not indicate generalizability. However, the section is presented to the reader to benchmark the range of predictive power that our features exhibit in more traditional tasks.

5.2.1. Aggregation of the results

To evaluate how each dimensionality reduction method and each feature combination performs across all the pipelines, we need to aggregate the pipelines' results. We do this by ranking, giving each pipeline a rank for NRMSE, then grouping on either dimensionality reduction, feature combination, then taking the average. This mean of ranks gives us the results of what pipeline components perform best when testing in the same context it was trained.

5.2.2. Dimensionality reduction and feature selection

Figure 3 shows which method reduces the feature space for the pipelines with the five smallest NRMSEs per dataset. Which method performs better seems to rely heavily on the in-study dataset, making it hard to conclude which of the two performs better.

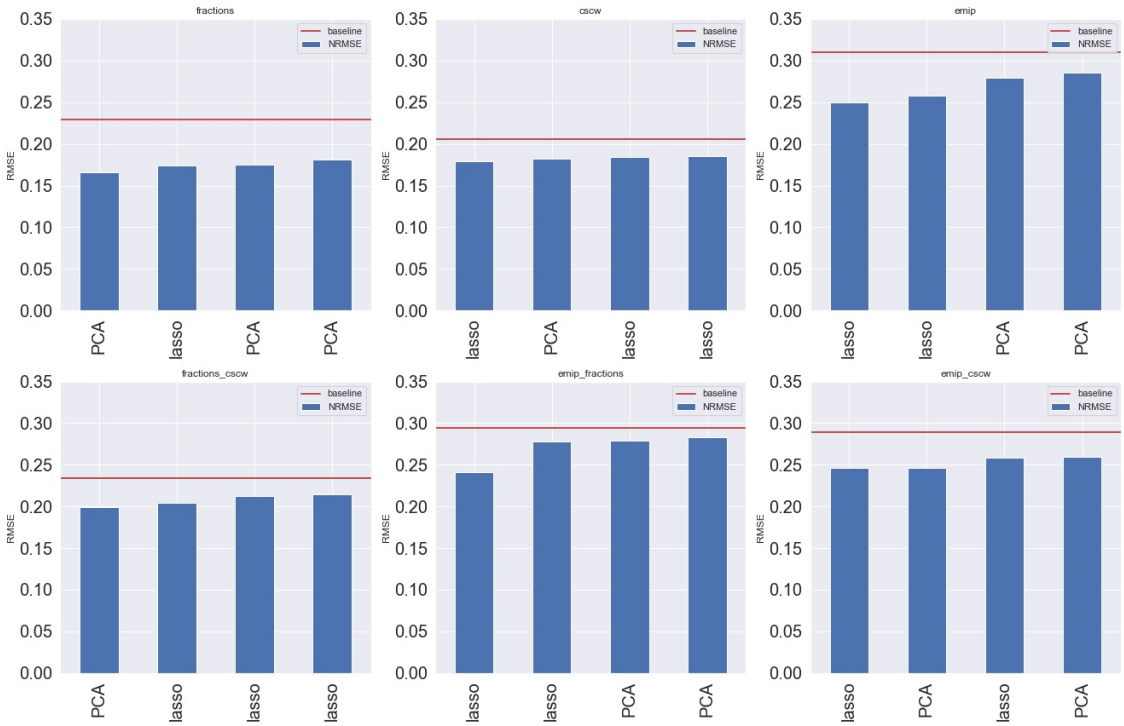

Figure 3. The four best pipelines by NRMSE per dataset and if whether used dimensionality reduction or feature selection

However, when we aggregate the results as seen in Table 1, and Table 2, we can see that LASSO performs slightly better than PCA across all 1-to-1 pipelines, and clearly better across the 2-to-1 pipelines. So our results indicate that feature selection performs better than dimensionality reduction in predicting cognitive performance on gaze data for out-of-sample testing.

Table 1. NRMSE Rank and mean NRMSE for PCA and LASSO for all 1-to-1 pipelines

Method	NRMSE	NRMSE_RANK
lasso	0.265	71.5
PCA	0.269	73.5

Table 2. NRMSE Rank and mean NRMSE for PCA and LASSO for all 2-to-1 pipelines

Method	NRMSE	NRMSE_RANK
lasso	0.272	32.7
PCA	0.283	40.3

5.2.3. Features

Figure 4 shows the best four pipelines by NRMSE and their feature groups. It seems that several different feature groups perform quite well for the different dataset combinations. We note that fixation duration performs well in most combinations and that spectral histograms perform well in several pipelines and ALL perform well, particularly for fractions_cscw.

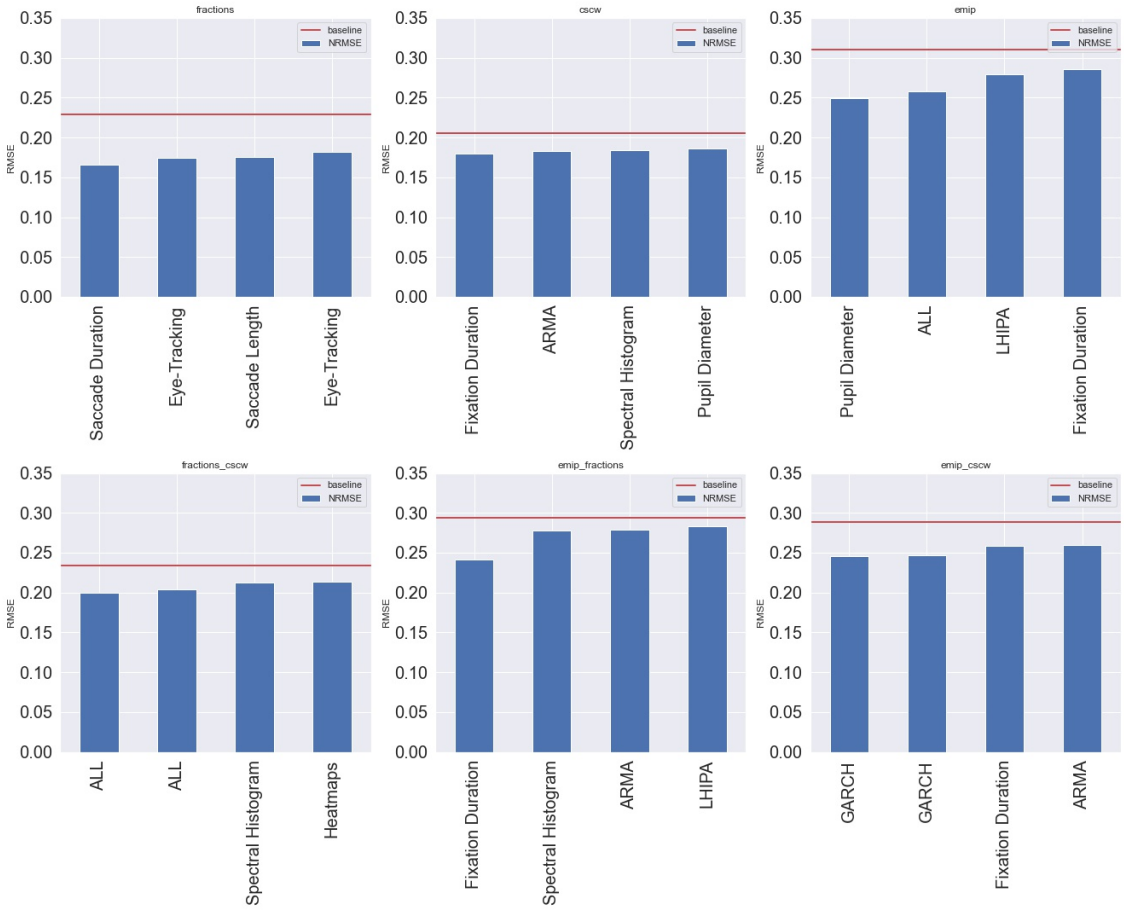

Figure 4. The four best pipelines by NRMSES per dataset and what feature group it contained.

When we aggregate the results, as seen in Table 3 and Table 4. We see that eye-tracking is the best feature group for out-of-sample testing amongst the 1-to-1 pipelines, performing slightly better than heatmaps and fixation duration. 2-to-1 pipelines that include GARCH seem to outperform other 2-to-1 pipelines, but fixation duration also performs quite well.

Table 3. The four best pipelines by NRMSE Rank and mean NRMSE for the feature groups across all 1-to-1 pipelines.

Feature Group	**rmse**	**RMSE_RANK**

Eye Tracking Features	0.246	54.8
Heatmaps	0.253	61.1
Fixation Duration	0.263	69.7
Saccade Duration	0.266	71.2
Power Spectral Histogram	0.268	71.8

Table 4. The four best pipelines by NRMSE Rank and mean NRMSE for the feature groups across all 2-to-1 pipelines.

Feature Group	**rmse**	**RMSE_RANK**
GARCH	0.264	27.8
Fixation Duration	0.269	29.8
ALL	0.267	32.7
Power Spectral Histogram	0.271	34.8
HMM	0.278	37.2

5.3. Generalizability

We evaluate the generalizability of the pipelines with the out-of-study dataset. We compare the errors from making predictions on the out-of-study dataset to the out-of-sample results using FGI, explained in Section 4.4.2. In order to identify the most generalizable pipelines, we need to filter out pipelines that perform poorly in out-of-sample testing. We do this by filtering out the pipelines that do not perform better than the baseline. From 216 pipelines, 72 beat the baseline.

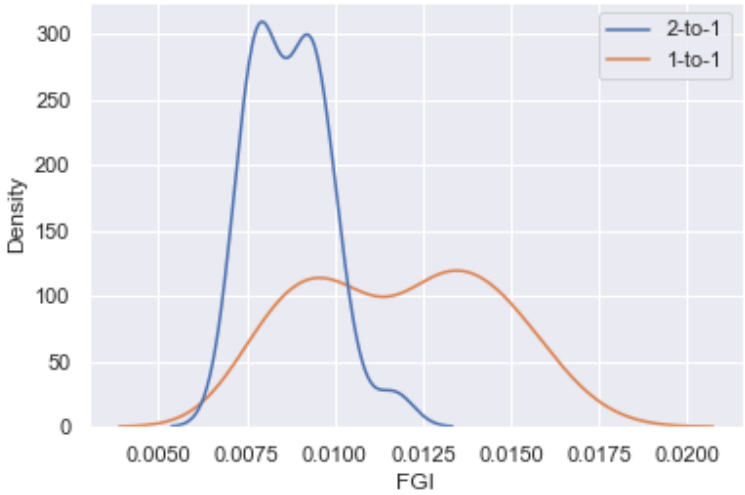

Figure 5. Kernel Density Estimation plot of the FGI by pipeline type

As represented in Figure 5, 2-to-1 pipelines are in general more generalizable than 1-to-1 pipelines. This is in line with the results of [69]. For simplicity and since our focus is generalizability, we will only refer to the 2-to-1 pipelines in the following sections.

Table 5. Generalizable pipelines

FGI	**In Study**	**Feature Group**	**PCA or Lasso**
0.007	fractions_cscw	GARCH	lasso

0.0074	fractions_cscw	Power Spectral Histogram	lasso
0.0074	fractions_cscw	ALL	PCA
0.0077	fractions_cscw	Saccade Duration	lasso
0.0077	fractions_cscw	Heatmaps	PCA
0.0077	fractions_cscw	Saccade Length	lasso
0.0077	fractions_cscw	Heatmaps	lasso
0.0078	fractions_cscw	ALL	lasso
0.0078	fractions_cscw	HMM	lasso
0.0081	emip_cscw	GARCH	PCA

Table 5 shows the 10 most generalizable pipelines. The first and most prevalent factor for generalizable pipelines is which datasets were designated in-study datasets for that pipeline. When the in-study dataset combines Fractions and CSCW (fractions_cscw), and EMIP is the out-of-study dataset, the pipelines are more generalizable. Nine of the ten more generalizable pipelines contain this combination of datasets.

LASSO is the most represented method of feature space reduction among the more generalizable pipelines. We note that there are more pipelines with LASSO that beat the baseline and might explain the trend we are seeing. LASSO is also included in the two pipelines with the best FGI. PCA performs very well for the

feature groups ALL and heatmap; this might be explained by the fact that these are the largest feature groups with hundreds of thousands of values.

Seven of our twelve feature groups are represented among the ten most generalizable pipelines. However, three feature groups show up more than once, ALL, heatmaps, and GARCH. This indicates that they might be more generalizable feature groups. It should also be noted that GARCH is also the only pipeline with an in-study dataset that is not fractions_cscw of the ten most generalizable pipelines.

5.4. Context Sensitivity

The bottom third of the filtered baselines contain the pipelines that are more context-specific.

Table 6. Context sensitive pipelines

FGI	**In Study**	**Feature Group**	**PCA or Lasso**
0.0093	emip_cscw	Heatmaps	lasso
0.0093	emip_cscw	ARMA	PCA
0.0095	emip_fractions	Power Spectral Histogram	lasso
0.0096	emip_cscw	HMM	lasso
0.0096	emip_fractions	LHIPA	PCA
0.0098	emip_cscw	Saccade Length	lasso

0.0102	emip_cscw	Eye Tracking Features	lasso
0.0102	emip_cscw	Pupil Diameter	lasso
0.0117	emip_cscw	ALL	lasso

Again we can see that the dataset combination is an important factor in which pipelines are exhibit context sensitivity. LASSO is more represented among the more context-specific pipelines. As with the generalizable pipelines, there are more pipelines with LASSO that beat the baseline. For the feature groups, we observe some differences from the generalizable pipelines. ARMA, LHIPA, Eye Tracking, and Pupil Diameter are present in the context-sensitive pipelines but not in the more generalizable pipelines. This indicates that these feature groups are more likely to produce context-sensitive pipelines. Heatmaps, Spectral Histogram, Hidden Markov Models, ALL, and Saccade Length appear in generalizable and context-sensitive pipelines. This further supports the observation that dataset combination is an essential variable for generalizability.

6. Discussion

6.1. Findings

- 2-to-1 pipelines are more generalizable than 1-to-1 pipelines.
- 2-to-1 pipelines where Fractions and CSCW are the in-study datasets produce more generalizable pipelines.
- GARCH, Power Spectral Histogram, heatmaps, saccade duration, saccade length, and Markov can produce generalizable pipelines.
- Pupil Diameter, Eye tracking, LHIPA, and ARMA can produce context-sensitive pipelines, and we do not observe any tendencies to generalize.

6.2. Filtering on baseline

Our goal is to identify generalizable pipelines, and for that, we have the FGI measure. FGI describes how similar the distribution of errors from out-of-sample testing and out-of-study testing and is a measure of generalizability. However, it is not enough that the two distributions are similar. Generalizability should be a measure of usable predictive power in outside contexts. Random guesses would create identical distributions of error in out-of-sample testing and out-of-study testing, and as such, would have a good FGI value. To counteract this, we chose to disregard all pipelines that do not outperform their respective baselines for the purpose of measuring generalizability. We do this by filtering the set of completed baselines.

6.3. 2-to-1 versus 1-to-1

As shown in Figure 5 2-to-1 pipelines tend to generalize better then 1-to-1 pipelines. Combining two datasets from two separate contexts for training introduces more variance. Unsurprisingly, these pipelines are more generalizable than those that use only one dataset for training. These results align with the findings of Sharma et al. [69]. In further discussions, we will focus on results from the 2-to-1 pipelines.

6.4. Generalizability of our datasets

Our results indicate that pipelines that combine Fractions and CSCW for training and use EMIP for testing are more generalizable than pipelines where EMIP is included in the training set. The EMIP dataset consists of gaze data recorded of people completing individual tasks. All datasets include information from one or more context-specific tasks, but the Fractions and CSCW also have a collaborative aspect. We theorize that pipelines trained on EMIP do worse at describing this collaborative aspect and that the variance introduced by the interactions that come with collaborative work assists those pipelines in generalizing. Pinpointing the effectiveness of training on collaborative data when generalizing to contexts without a collaborative aspect is a promising area for more experimentation.

6.5. Why do the generalizable features generalize?

In this section, we will argue a few reasons we believe as to why the features generalize.

6.5.1. Heatmaps

We observe that heatmaps are among the generalizable features. Our heatmap feature is the gaze-position for each subject over time, split into 30 partitions, converted to a heatmap, and fed to a pre-trained VGG19 model. The resulting feature vector is the feature we call heatmaps. While the stimulus is not included in the heatmap image, the stimulus is captured through the subject's interaction with it, as the gaze pattern will follow the stimulus. The feature vector we call heatmaps represents how the gaze interacts with the given stimuli. Taking EMIP as an example, a heatmap will tell us the shape of the code being presented to the subject for all subjects. However, it will also tell us how much time a subject spends reading the syntactic structures versus the variable names or conditional logic. The latter is a focus point for experienced coders. Heatmaps generalize well because instead of just capturing how a subject interacts with their stimulus, it also captures their interaction patterns and how they relate to their presented stimuli. In addition, heatmaps encode information about time and space, rather than just one of the dimensions, introducing more variance.

6.5.2. GARCH, Hidden Markov Models and Power Spectral Histogram

GARCH is a statistical modeling technique used to model timeseries data. Our data suggest that GARCH is the most generalizable feature group of the ones we have tested. GARCH is models the variance of data and has the heteroscedasticity property. Heteroscedasticity means that the random disturbance is different across the elements. In our case, this means the variance of indicators of cognitive performance varies from point to point in the timeseries. In different contexts, the timing of the indicators will vary based on the task and other factors. This might explain why GARCH is generalizable across contexts.

Our results indicate that spectral histograms generalize pretty well. Spectral histograms are the distribution of frequencies and their corresponding amplitudes.

Spectral histograms capture repetitiveness and hence might capture patterns of behavior exhibited by the subjects. Repetitiveness indicates a flow that suggests that the subject is comfortable in their task; on the other hand, chaotic non-repetitive behavior might occur when one is not comfortable. We suggest that this relationship should be represented across contexts and could be why spectral histograms seem to generalize quite well.

We do not know why HMM generalizes this is a mystery and a subject for future discussions with Sharma et al. [69, 68, 67, 70]

6.5.3. Saccade duration and Saccade length

Both saccade duration and saccade length seem to generalize. Saccades are the movements of the eyes from one fixation to another. As one example, both lower saccade durations and longer saccade lengths might indicate familiarity with the stimulus. As discussed in Section 4.2.4.2 higher saccade speeds correlate with familiarity with the stimulus [53]. Saccade speed is a function of saccade duration and saccade length; both factors map familiarity to some degree. Since familiarity with the interface would be relevant in all tasks that our datasets contain, this might be why saccade duration and saccade length generalize. De Luca et al. [26] show that saccadic length increases when reading longer pseudowords. According to this perspective, saccadic length correlates with perceived task difficulty. Perceived task difficulty or increased cognitive workload will correlate to cognitive performance across contexts.

6.6. Context-specificity

In this section, we will argue a few reasons we believe as to why the features are context-specific.

6.6.1. ARMA

Our results indicate that ARMA produces quite context-sensitive pipelines. ARMA models the conditional mean of a timeseries, as opposed to the variance which GARCH models. Intuitively the mean value would be more dependent on the measurement's context than the variance. For example, an ARMA model would be impacted by a stimulus with longer distances between points of interest when modeling saccade length, while GARCH would not.

6.6.2. LHIPA and Pupil Diameter

LHIPA was developed to be an indicator of cognitive load [29]. The cognitive load of a task will have many factors, but the specific complexities of the given task must undoubtedly be a part of those. LHIPA likely models these complexities and other factors directly related to the task. Our observations that LHIPA can produce context-sensitive pipelines are in line with this thinking.

Pupil diameter has been shown to be affected by several task-specific factors. Both task difficulty and time limits have an impact on pupil dilation [71]. These are task-related factors, and as such, it might explain why pupil diameter seems to produce context-specific pipelines.

Shojaeizadeh et al. also point out that pupil size might convey information about variation in cognitive effort. This factor seems more likely to generalize and is what we model with GARCH. GARCH of pupil diameter is included in this feature group. However, the group primarily consists of features that model the mean of pupil diameter.

6.6.3. Eye-Tracking Features

The eye-tracking feature tries to encapsulate different strategies for interacting with the stimulus. The information processing ratio represents the tendency to skim text versus more focused reading. A strategy of either skimming or focused reading might be more appropriate for a specific task, which might be why this indicates context-specificity. However, this is not an entirely specific trait. Some skill might be involved in picking the correct strategy when presented with a stimulus, and greater familiarity might lead to a faster transition to focused reading.

Entropy models the spread of the gaze over the stimulus, which might model the generalizable aspect of focus; however, it is also affected by the task design. The verticality of saccades is also certainly context-specific as it relies heavily on the nature of and how the stimulus is organized.

6.7. Limitations and further work

In Section 3.2, we outline how we believe our datasets are representative of a significant portion of human cognition. However, it would be presumptuous to say that three datasets from three different contexts could represent all of the cognitive processes. Our goal has been to generalize between our three contexts, and we hope that our methods provide meaningful insights into how our one could create generalizable features for other contexts. We do not mean to say that our features will generalize to any context.

Our results show some indications that datasets from individual tasks generalize poorly to contexts that include collaborative work. Had individual work been better represented in our data, we might be able to say more about how individual tasks generalize in general. Ideally, we should have had at least one more dataset for individual tasks.

Our work assumes that cognitive performance can be characterized by labels in our datasets and represented in gaze data. For our approach, we need an object,

quantifiable, metric to assess cognitive performance, but as with many other things in cognition, the reality is likely more complex.

For complete external repeatability, we would ideally publish the data we used to perform our experiments. However, the scope of our thesis project was such that it would be impossible to gather our own data to perform the analyses we have performed. As a result, we had to turn to generous researchers who allowed us to work with their data, which means that the data is not ours to share.

Due to the considerable effort put into creating our experimental platform, it would be possible to expand the different pipeline components we test greatly. In our work, we tested 22 features in 12 feature groups, three datasets in 9 combinations, two methods for reducing the feature space, and a single ensemble classifier. While our tested features are quite exhaustive, we limited how many feature-space reduction methods we worked with and tested only a single ensemble classifier. It would be possible to investigate the effects of other variants of these pipeline components on generalizability in further work.

While we can identify feature groups that can produce generalizable pipelines, we do not know how the individual features in each group affect the generalizability. It is also likely that combinations of features from different groups would create very generalizable pipelines.

7. Conclusion and contributions

In this work, we apply the methodology presented by Sharma et al. [69] to engineer generalizable features from gaze data. The work draws on three independent studies with data gathered on a total of XXX subjects. We also propose an architecture to efficiently and easily test the generalizability of pipelines constructed of different combinations of pipeline components. The code for our platform and the analysis is available on GitHub [73].

Our contributes include: - The list of generalizable features presented in Table 5. - The list of context-sensitive features presented in Table 6. - The architecture and implementation for experimenting on modular pipelines for generalizability experiments presented in Section 4.

8. Bibliography

[1] “Prediction in MOOCs: A Review and Future Research Directions.” https://ieeexplore.ieee.org/document/8412110.

[2] “SMI RED250 - iMotions,” *Imotions Publish.* https://imotions.com/hardware/smi-red250/.

[3] “The Low/High Index of Pupillary Activity | Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.” https://dl.acm.org/doi/abs/10.1145/3313831.3376394.

[4] “Cmap | CmapTools.” https://cmap.ihmc.us/.

[5] “Khan Academy | Free Online Courses, Lessons & Practice,” *Khan Academy.* https://www.khanacademy.org/.

[6] “PyGaze | Open Source Eye-Tracking Software and More.” .

[7] “Git.” https://git-scm.com/.

[8] “Poetry - Python Dependency Management and Packaging Made Easy.” https://python-poetry.org/.

[9] “Machine Learning and Artificial Intelligence Research for Patient Benefit: 20 Critical Questions on Transparency, Replicability, Ethics, and Effectiveness | The BMJ.” https://www.bmj.com/content/368/bmj.l6927.abstract.

[10] “Citation - Comet.Ml.” https://www.comet.ml/docs/citation/.

[11] “Devil in the Details,” *Nature*, vol. 470, no. 7334, pp. 305–306, Feb. 2011, doi: 10.1038/470305b.

[12] “LumenResearch/Heatmappy.” Lumen Research, May-2021.

[13] D. Baby and S. Verhulst, “Biophysically-Inspired Features Improve the Generalizability of Neural Network-Based Speech Enhancement Systems,” in *Interspeech 2018*, 2018, pp. 3264–3268, doi: 10.21437/Interspeech.2018-1237.

[14] R. Bednarik *et al.*, “EMIP: The Eye Movements in Programming Dataset,” *Science of Computer Programming*, vol. 198, p. 102520, Oct. 2020, doi: 10.1016/j.scico.2020.102520.

[15] W. Bleidorn and C. J. Hopwood, “Using Machine Learning to Advance Personality Assessment and Theory,” *Personality and Social Psychology Review*, vol. 23, no. 2, pp. 190–203, May 2019, doi: 10.1177/1088868318772990.

[16] P. Blignaut, “Fixation Identification: The Optimum Threshold for a Dispersion Algorithm,” *Attention, Perception, & Psychophysics*, vol. 71, no. 4, pp. 881–895, May 2009, doi: 10.3758/APP.71.4.881.

[17] A. Bojko, “Eye Tracking in User Experience Testing: How to Make the Most of It,” *Proceedings of the 14th Annual Conference of the Usability Professionals' Association (UPA). Montréal, Canada*, Jan. 2005.

[18] T. Bollerslev, “Generalized Autoregressive Conditional Heteroskedasticity,” *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, Apr. 1986, doi: 10.1016/0304-4076(86)90063-1.

[19] K. Bouchard, M. R. Eusufzai, R. Ramezani, and A. Naeim, “Generalizable Spatial Feature for Human Positioning Based on Bluetooth Beacons,” in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2016, pp. 1–5, doi: 10.1109/UEMCON.2016.7777884.

[20] V. Cantoni and M. Porta, “Eye Tracking as a Computer Input and Interaction Method,” in *Proceedings of the 15th International Conference on Computer Systems and Technologies*, New York, NY, USA, 2014, pp. 1–12, doi: 10.1145/2659532.2659592.

[21] A. M. Chekroud *et al.*, "Cross-Trial Prediction of Treatment Outcome in Depression: A Machine Learning Approach," *The Lancet Psychiatry*, vol. 3, no. 3, pp. 243–250, Mar. 2016, doi: 10.1016/S2215-0366(15)00471-X.

[22] S. Chen and J. Epps, "Using Task-Induced Pupil Diameter and Blink Rate to Infer Cognitive Load," *Human-Computer Interaction*, vol. 29, Apr. 2014, doi: 10.1080/07370024.2014.892428.

[23] A. Cheng, A. Dimoka, and P. Pavlou, "Context May Be King, but Generalizability Is the Emperor!," *Journal of Information Technology*, vol. 31, Sep. 2016, doi: 10.1057/s41265-016-0005-7.

[24] K. Clarke, "Nonparametric Multivariate Analyses of Changes in Community Structure," *Austral Ecology*, vol. 18, pp. 117–143, Mar. 1993, doi: 10.1111/j.1442-9993.1993.tb00438.x.

[25] R. M. Davison and M. G. Martinsons, "Context Is King! Considering Particularism in Research Design and Reporting," *Journal of Information Technology*, vol. 31, no. 3, pp. 241–249, Sep. 2016, doi: 10.1057/jit.2015.19.

[26] M. De Luca, M. Borrelli, A. Judica, D. Spinelli, and P. Zoccolotti, "Reading Words and Pseudowords: An Eye Movement Study of Developmental Dyslexia," *Brain and Language*, vol. 80, no. 3, pp. 617–626, Mar. 2002, doi: 10.1006/brln.2001.2637.

[27] G. P. Dexter, S. J. Grannis, B. E. Dixon, and S. N. Kasthurirathne, "Generalization of Machine Learning Approaches to Identify Notifiable Conditions from a Statewide Health Information Exchange," *AMIA Summits on Translational Science Proceedings*, vol. 2020, pp. 152–161, May 2020.

[28] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow Enables Reproducible Computational Workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, Apr. 2017, doi: 10.1038/nbt.3820.

[29] A. T. Duchowski, K. Krejtz, N. A. Gehrer, T. Bafna, and P. Bækgaard, "The Low/High Index of Pupillary Activity," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2020, pp. 1–12, doi: 10.1145/3313831.3376394.

[30] A. T. Duchowski *et al.*, "The Index of Pupillary Activity: Measuring Cognitive Load Vis-à-Vis Task Difficulty with Pupil Oscillation," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2018, pp. 1–13, doi: 10.1145/3173574.3173856.

[31] E. Ferentzi, T. Bogdány, Z. Szabolcs, B. Csala, Á. Horváth, and F. Köteles, "Multichannel Investigation of Interoception: Sensitivity Is Not a Generalizable Feature," *Frontiers in Human Neuroscience*, vol. 12, 2018, doi: 10.3389/fnhum.2018.00223.

[32] N. M. Fountain-Jones, G. Machado, S. Carver, C. Packer, M. Recamonde-Mendoza, and M. E. Craft, "How to Make More from Exposure Data? An Integrated Machine Learning Pipeline to Predict Pathogen Exposure," *Journal of Animal Ecology*, vol. 88, no. 10, pp. 1447–1461, 2019, doi: 10.1111/1365-2656.13076.

[33] M. N. Giannakos, K. Sharma, I. O. Pappas, V. Kostakos, and E. Velloso, "Multimodal Data as a Means to Understand the Learning Experience," *International Journal of Information Management*, vol. 48, pp. 108–119, Oct. 2019, doi: 10.1016/j.ijinfomgt.2019.02.003.

[34] L. Granka, T. Joachims, and G. Gay, "Eye-Tracking Analysis of User Behavior in WWW-Search," *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Apr. 2004, doi: 10.1145/1008992.1009079.

[35] G. Guzzetta, G. Jurman, and C. Furlanello, "A Machine Learning Pipeline for Quantitative Phenotype Prediction from Genotype Data," *BMC Bioinformatics*, vol. 11, no. 8, p. S3, Oct. 2010, doi: 10.1186/1471-2105-11-S8-S3.

[36] E. Haapalainen, S. J. Kim, J. F. Forlizzi, and A. K. Dey, "Psycho-Physiological Measures for Assessing Cognitive Load," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Copenhagen Denmark, 2010, pp. 301–310, doi: 10.1145/1864349.1864395.

[37] M. E. Hansen, "Quantifying Program Complexity and Comprehension," p. 6.

[38] D. O. Hebb, "Drives and the C. N. S. (Conceptual Nervous System)," *Psychological Review*, vol. 62, no. 4, pp. 243–254, 1955, doi: 10.1037/h0041823.

[39] S. Hutt, J. Grafsgaard, and S. D'Mello, "Time to Scale: Generalizable Affect Detection for Tens of Thousands of Students across An Entire School Year," in *CHI '19: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–14, doi: 10.1145/3290605.3300726.

[40] D. C. Ince, L. Hatton, and J. Graham-Cumming, "The Case for Open Computer Programs," *Nature*, vol. 482, no. 7386, pp. 485–488, Feb. 2012, doi: 10.1038/nature10836.

[41] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 121–129.

[42] T. Kraska, A. Talwalkar, and J. Duchi, "MLbase: A Distributed Machine-Learning System," p. 7.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[44] M. Kumar, "Google Cloud Platform: A Powerful Big Data Analytics Cloud Platform," *International Journal for Research in Applied Science & Engineering Technology*, vol. 4, pp. 387–392, Nov. 2016.

[45] Á. López García *et al.*, "A Cloud-Based Framework for Machine Learning Workloads and Applications," *IEEE Access*, vol. 8, pp. 18681–18692, 2020, doi: 10.1109/ACCESS.2020.2964386.

[46] S. McNally, J. Roche, and S. Caton, "Predicting the Price of Bitcoin Using Machine Learning," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2018, pp. 339–343, doi: 10.1109/PDP2018.2018.00060.

[47] F. Mohr, M. Wever, A. Tornede, and E. Hullermeier, "Predicting Machine Learning Pipeline Runtimes in the Context of Automated Machine Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3056950.

[48] R. Nagler, D. Bruhwiler, P. Moeller, and S. Webb, "Sustainability and Reproducibility via Containerized Computing," *arXiv:1509.08789 [cs]*, Sep. 2015.

[49] G. Nguyen *et al.*, "Machine Learning and Deep Learning Frameworks and Libraries for Large-Scale Data Mining: A Survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 77–124, Jun. 2019, doi: 10.1007/s10462-018-09679-z.

[50] E. S. Nurse, P. J. Karoly, D. B. Grayden, and D. R. Freestone, "A Generalizable Brain-Computer Interface (BCI) Using Machine Learning for Feature Discovery," *PLOS ONE*, vol. 10, no. 6, p. e0131328, Jun. 2015, doi: 10.1371/journal.pone.0131328.

[51] J. K. Olsen, D. M. Belenky, V. Aleven, and N. Rummel, "Using an Intelligent Tutoring System to Support Collaborative as Well as Individual Learning," in *Intelligent Tutoring Systems*, vol. 8474, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, Eds. Cham: Springer International Publishing, 2014, pp. 134–143.

[52] R. Olson and J. Moore, “TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning,” 2019, pp. 151–160.

[53] I. Pappas, K. Sharma, P. Mikalef, and M. Giannakos, “Visual Aesthetics of E-Commerce Websites: An Eye-Tracking Approach,” 2018, doi: 10.24251/HICSS.2018.035.

[54] R. Pelanek, “Metrics for Evaluation of Student Models,” *Journal of Educational Data Mining*, vol. 7, no. 2, pp. 1–19, 2015.

[55] R. D. Peng, “Reproducible Research in Computational Science,” *Science*, vol. 334, no. 6060, pp. 1226–1227, Dec. 2011, doi: 10.1126/science.1213847.

[56] B. Pfleging, D. K. Fekety, A. Schmidt, and A. L. Kun, “A Model Relating Pupil Diameter to Mental Workload and Lighting Conditions,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose California USA, 2016, pp. 5776–5788, doi: 10.1145/2858036.2858117.

[57] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, “Generalizable Intention Prediction of Human Drivers at Intersections,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1665–1670, doi: 10.1109/IVS.2017.7995948.

[58] K. Ram, “Git Can Facilitate Greater Reproducibility and Increased Transparency in Science,” *Source Code for Biology and Medicine*, vol. 8, no. 1, p. 7, Feb. 2013, doi: 10.1186/1751-0473-8-7.

[59] G. E. Raptis, C. A. Fidas, and N. M. Avouris, “Using Eye Tracking to Identify Cognitive Differences: A Brief Literature Review,” in *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, Patras Greece, 2016, pp. 1–6, doi: 10.1145/3003733.3003762.

[60] M. Ribeiro, K. Grolinger, and M. A. M. Capretz, “MLaaS: Machine Learning as a Service,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 896–902, doi: 10.1109/ICMLA.2015.152.

[61] B. Rogers, C. Justice, T. Mathur, and J. E. Burge, "Generalizability of Document Features for Identifying Rationale," in *Design Computing and Cognition '16*, Cham, 2017, pp. 633–651, doi: 10.1007/978-3-319-44989-0_34.

[62] L. Rupprecht, J. C. Davis, C. Arnold, Y. Gur, and D. Bhagwat, "Improving Reproducibility of Data Science Pipelines through Transparent Provenance Capture," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3354–3368, Aug. 2020, doi: 10.14778/3415478.3415556.

[63] P. Saari, T. Eerola, and O. Lartillot, "Generalizability and Simplicity as Criteria in Feature Selection: Application to Mood Classification in Music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1802–1812, Aug. 2011, doi: 10.1109/TASL.2010.2101596.

[64] J. Salminen, V. Yoganathan, J. Corporan, B. J. Jansen, and S.-G. Jung, "Machine Learning Approach to Auto-Tagging Online Content for Content Marketing Efficiency: A Comparative Analysis between Methods and Content Type," *Journal of Business Research*, vol. 101, pp. 203–217, Aug. 2019, doi: 10.1016/j.jbusres.2019.04.018.

[65] M. Schwab, N. Karrenbach, and J. Claerbout, "Making Scientific Computations Reproducible," *Computing in Science Engineering*, vol. 2, no. 6, pp. 61–67, Nov. 2000, doi: 10.1109/5992.881708.

[66] S. Shaikh, H. Vishwakarma, S. Mehta, K. R. Varshney, K. N. Ramamurthy, and D. Wei, "An End-To-End Machine Learning Pipeline That Ensures Fairness Policies," *arXiv:1710.06876 [cs]*, Oct. 2017.

[67] K. Sharma, D. Caballero, H. Verma, P. Jermann, and P. Dillenbourg, "Looking AT versus Looking THROUGH: A Dual Eye-Tracking Study in MOOC Context," in *CSCL*, 2015.

[68] K. Sharma, M. Giannakos, and P. Dillenbourg, "Eye-Tracking and Artificial Intelligence to Enhance Motivation and Learning," *Smart Learning Environments*,

vol. 7, no. 1, p. 13, Apr. 2020, doi: 10.1186/s40561-020-00122-x.

[69] K. Sharma, E. Niforatos, M. Giannakos, and V. Kostakos, "Assessing Cognitive Performance Using Physiological and Facial Features: Generalizing across Contexts," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 95:1–95:41, Sep. 2020, doi: 10.1145/3411811.

[70] K. Sharma, J. K. Olsen, V. Aleven, and N. Rummel, "Measuring Causality between Collaborative and Individual Gaze Metrics for Collaborative Problem-Solving with Intelligent Tutoring Systems," *Journal of Computer Assisted Learning*, vol. 37, no. 1, pp. 51–68, 2021, doi: 10.1111/jcal.12467.

[71] M. Shojaeizadeh, S. Djamasbi, R. C. Paffenroth, and A. C. Trapp, "Detecting Task Demand via an Eye Tracking Machine Learning System," *Decision Support Systems*, vol. 116, pp. 91–101, Jan. 2019, doi: 10.1016/j.dss.2018.10.012.

[72] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015.

[73] S. Sollesnes August and A. Hollund, "S0lvang/Ideal-Pancake." May-2021.

[74] B. Steichen, G. Carenini, and C. Conati, "User-Adaptive Information Visualization: Using Eye Gaze Data to Infer Visualization Tasks and User Cognitive Abilities," in *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, New York, NY, USA, 2013, pp. 317–328, doi: 10.1145/2449396.2449439.

[75] A. Stewart, N. Bosch, and S. K. D'Mello, "Generalizability of Face-Based Mind Wandering Detection Across Task Contexts," p. 8.

[76] J. R. Sutton, R. Mahajan, O. Akbilgic, and R. Kamaleswaran, "PhysOnline: An Open Source Machine Learning Pipeline for Real-Time Analysis of Streaming Physiological Waveform," *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 1, pp. 59–65, Jan. 2019, doi: 10.1109/JBHI.2018.2832610.

[77] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–

288, 1996.

[78] D. Toker, S. Lallé, and C. Conati, “Pupillometry and Head Distance to the Screen to Predict Skill Acquisition During Information Visualization Tasks,” in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, New York, NY, USA, 2017, pp. 221–231, doi: 10.1145/3025171.3025187.

[79] P. D. Turney, “Robust Classification with Context-Sensitive Features,” in *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh, Scotland, 1993, pp. 268–276.

[80] P. Turney, “The Identification of Context-Sensitive Features: A Formal Definition of Context for Concept Learning,” *CoRR*, vol. cs.LG/0212038, Dec. 2002.

[81] P. D. Turney, “Exploiting Context When Learning to Classify,” in *Proceedings of the European Conference on Machine Learning*, Berlin, Heidelberg, 1993, pp. 402–407.

[82] P. Vaillancourt *et al.*, “Reproducible and Portable Workflows for Scientific Computing and HPC in the Cloud,” in *Practice and Experience in Advanced Research Computing*, Portland OR USA, 2020, pp. 311–320, doi: 10.1145/3311790.3396659.

[83] S. Weintraub *et al.*, “Cognition Assessment Using the NIH Toolbox,” *Neurology*, vol. 80, no. Issue 11, Supplement 3, pp. S54–S64, Mar. 2013, doi: 10.1212/WNL.0b013e3182872ded.

[84] C. Wissler, “The Spearman Correlation Formula,” *Science*, vol. 22, no. 558, pp. 309–311, 1905.

[85] Y. Zhang, Y. Owechko, and J. Zhang, *TuC4.1 Driver Cognitive Workload Estimation: A Data-Driven Perspective*. .
Last updated 2021-05-16 17:51:23 +0200