GazeGraph: Graph-based Few-Shot Cognitive Context Sensing from Human Visual Behavior

Guohao Lan, Bailey Heit, Tim Scargill, Maria Gorlatova
Duke University, Durham, North Carolina
{guohao.lan, bailey.heit, ts352, maria.gorlatova}@duke.edu

ABSTRACT

In this work, we present GazeGraph, a system that leverages human gazes as the sensing modality for cognitive context sensing. GazeGraph is a generalized framework that is compatible with different eye trackers and supports various gaze-based sensing applications. It ensures high sensing performance in the presence of heterogeneity of human visual behavior, and enables quick system adaptation to unseen sensing scenarios with few-shot instances. To achieve these capabilities, we introduce the *spatial-temporal gaze graphs* and the *deep learning-based representation learning method* to extract powerful and generalized features from the eye movements for context sensing. Furthermore, we develop a *few-shot gaze graph learning module* that adapts the 'learning to learn' concept from meta-learning to enable quick system adaptation in a data-efficient manner. Our evaluation demonstrates that GazeGraph outperforms the existing solutions in recognition accuracy by 45% on average over three datasets. Moreover, in few-shot learning scenarios, GazeGraph outperforms the transfer learning-based approach by 19% to 30%, while reducing the system adaptation time by 80%.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing theory, concepts and paradigms**.

KEYWORDS

Eye tracking, cognitive context sensing, few-shot learning.

ACM Reference Format:
Guohao Lan, Bailey Heit, Tim Scargill, Maria Gorlatova. 2020. GazeGraph: Graph-based Few-Shot Cognitive Context Sensing from Human Visual Behavior. In *The 18th ACM Conference on Embedded Networked Sensor Systems (SenSys '20), November 16–19, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3384419.3430774

1 INTRODUCTION

Recently, human visual behavior has emerged as a new sensing modality to capture *cognitive contexts*, such as attention, emotion, and knowledge acquisition of the user [1]. Specifically, as eye movement is strongly related to the cognitive processes of visual perception [2], such as visual memory [3], emotion [4], and mental workload [5], gaze-based sensing has extended the current mobile sensing spectrum with a new cognitive dimension and has opened the gates to the world of cognition-aware applications [4–10]. Indeed, the industry has seized this opportunity and released a smorgasbord of commercial products that are integrated with eye tracking for cognitive-aware computing. For instance, gaming laptops from Alienware are enhanced with eye trackers to capture the user's visual attention and awareness in game playing and use them as metrics for professional game training [11], while driver monitoring systems in BMW and Volvo cars use eye tracking to capture the driver's attention [12] and identify subconscious driving behavior [13]. Although much exciting research has been done in gaze-based context recognition [4–10, 14, 15], two practical aspects have been largely overlooked in existing solutions.

First, existing works heavily rely on hand-crafted features to analyze the eye movement patterns [3–9]. However, human visual behavior is highly heterogeneous across subjects [8, 16, 17], visual stimuli [14, 15], and eye tracking devices. For instance, eye movements involved in reading articles are diverse among subjects and reading materials (e.g., the layout and salience content) [15]. Due to this intrinsic diversity, hand-crafted features devised for a specific gaze-sensing condition, i.e., for specific subjects, visual stimuli, and eye tracking hardware, are difficult to generalize to new conditions. This is known as the domain shift problem [18], which significantly degrades the performance of existing gaze-based recognition solutions for 'in the wild' scenarios.

Second, to improve the recognition accuracy, one might think of leveraging deep neural networks (DNNs), e.g., the convolutional neural networks (CNN) [19] or the long short-term memory networks (LSTM) [20], owing to their superior feature learning and classification performance [21]. Such capabilities, however, rely on the availability of abundant gaze instances that cover diverse sensing conditions. Unfortunately, the countless possible gaze-sensing conditions as well as the rising privacy concerns about eye movement data [22–24] make the collection of a large-scale gaze dataset infeasible, and thus significantly limit the efficiency of applying DNNs directly to gaze-based sensing problems. Indeed, when investigating the accuracy of an LSTM-based classifier on two cognitive context recognition tasks [14, 15], we observe a 60% accuracy drop when limited training samples are available (Section 3.2.2).

To move beyond these limitations, we present GazeGraph, a generalized gaze-based sensing framework that is compatible with different eye trackers and supports various cognitive context sensing applications. It promises high recognition performance in the presence of heterogeneity of human visual behavior, and enables quick system adaptation to unseen sensing scenarios with few-shot instances. These capabilities are made possible by a suite of novel techniques devised in this work.

SenSys '20, November 16–19, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-7590-0/20/11…$15.00
https://doi.org/10.1145/3384419.3430774

First, GazeGraph incorporates the *spatial-temporal gaze graph* (Section 6) and the *gaze graph classifier* (Section 7.1). The former is a novel data modeling technique that converts the original eye movement sequence into a *spatial-temporal graph*. The constructed graph pays selective attention to the gazes in the sensing window and preserves the eye movement patterns that are embedded in the original signal in both temporal domain (through modeling the local pairwise relation between adjacent gazes) and spatial domain (through the construction of a k-hop neighbor graph). The gaze graph classifier leverages a deep neural network to extract a low-dimensional data representation of the constructed graph for classification. *The obtained representation is sensitive to minute eye movement details embedded in the graph, but insensitive to large irrelevant variations that result from the diversity of visual behavior.* As demonstrated (Section 7.1), the two techniques we devised lead to significant accuracy improvements over conventional hand-crafted feature-based methods [14, 15] and over an LSTM-based classifier. GazeGraph also exhibits a strong generalization capability across eye trackers and applications (e.g., compatible with three types of eye trackers and supports two gaze-based sensing applications).

Second, to enable fast system adaptation with few gaze instances required, we frame the task as a few-shot learning problem [25, 26], that is: how to train the gaze graph classifier such that it can quickly adapt to new sensing conditions (e.g., new subjects) after a few learning iterations (e.g., ten gradient steps) with a small number of instances from the new conditions (e.g., five instances per class). Specifically, we borrow the principles of 'learning to learn' from meta-learning [26–31], and design *the few-shot gaze graph learning module* (Section 7.2) to identify a good model initialization for the gaze graph classifier in training. Evaluation on three datasets (Section 8.3) indicates that, in scenarios with few-shot instances, GazeGraph achieves significant improvement in both recognition accuracy and system adaptation time over the state-of-the-art transfer learning-based approach [32, 33].

Our major contributions are summarized as follows:

- We introduce a novel approach that models human visual behavior as graphs for gaze-based cognitive context sensing. We develop the spatial-temporal gaze graph and the CNN-based gaze graph classifier to improve the recognition performance and to ensure a better generalization capability across different eye trackers and gaze-based sensing applications.
- We formulate the gaze-based recognition task as a few-shot learning problem and design a few-shot gaze graph learning module to ensure good recognition performance and fast system adaptation in new gaze-sensing scenarios with few training instances.
- We conduct a comprehensive evaluation of GazeGraph on three datasets, two public and one collected by ourselves, which covers three types of eye trackers and two gaze-based sensing applications. The results demonstrate that GazeGraph outperforms the conventional feature-based method and an LSTM-based classifier by 45% and 20% on average over three datasets. Moreover, in 5-shot and 10-shot scenarios, GazeGraph outperforms the transfer learning-based approach by 30% and 19% on average, respectively, while reducing the system adaptation time by 80%.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 provides the background and the motivation. Section 4 presents the system overview. The design details of the signal processing pipeline, the spatial-temporal gaze graph, and the few-shot graph learning model are given in Sections 5, 6, and 7, respectively. We present the evaluation in Section 8, discuss the limitations and future work in Section 9, and conclude the paper in Section 10. The dataset collected in this work is available at https://github.com/GazeGraphResource/GazeGraph.

2 RELATED WORK

Cognitive context recognition. GazeGraph is related to a rich body of work on eye tracking-based cognitive context sensing, including sedentary activity recognition [6, 14, 34, 35], reading behavior analysis [7, 8, 15, 36], and emotion recognition [4]. All these works heavily rely on hand-crafted features for classification. However, due to the intrinsic heterogeneity in human visual behavior, hand-crafted features usually lead to poor recognition performance in highly diverse sensing conditions (Section 3.2). By contrast, GazeGraph ensures better feature learning capability by modeling the eye movements as graphs and leveraging the gaze graph classifier to capture the spatial-temporal information embedded in the eye movements. There are also works that use eye-area images for emotion recognition [37, 38]. For instance, EMO [38] leverages the CNN to capture sophisticated features from eye-area images for emotion recognition. However, due to the visual behavior diversity, EMO requires subject-dependent optimization and requires 285 images per class during the training. Similarly, SPIDERS [37] requires 1000 images per class for the training. By contrast, GazeGraph is robust against the visual behavior heterogeneity and requires only five instances per class when adapting to new subjects.

Graph-based context recognition. There is a large body of work on applying graph models together with machine learning techniques for context recognition [39, 40]. Applications such as inferring the functional annotations for genes protein [41], feature learning for molecular structures [42], text classification [43], as well as human action recognition [44, 45], are leveraging graph models and DNNs to ensure good performance. GazeGraph shares similar motivation with these recent advances: leveraging graph-modeling and DNNs to learn a better data representation [39]. However, GazeGraph is the first work that focuses on gaze-based sensing problems. We also devise techniques that model the human gaze signal as spatial-temporal graphs, and introduce new contributions to address the unique domain shift challenges that result from the heterogeneity of human visual behavior.

Few-shot learning. The problem of enabling machine learning models to quickly adapt to unseen scenarios with limited training samples, known as few-shot learning, has attracted much effort in recent years [25–31, 46, 47]. Meta-learning, in particular, has shown promise in few-shot image classification tasks [26, 28–31]. GazeGraph shares the underlying concepts of 'learning to learn' [27] with the recent meta-learning advances that achieve few-shot learning by first training the classification model on a set of source tasks to identify a good model initialization and then fine-tuning the optimized model to unseen tasks with only a few gradient descent updates. The study most related to ours is MetaSense [47] which applies the model-agnostic meta-learning framework [28] to tackle the individual condition problem in activity and speech recognition.

To the best of our knowledge, GazeGraph is the first work that leverages the 'learning to learn' concept to address the domain shift problem in gaze-based context sensing.

3 BACKGROUND AND MOTIVATION

3.1 Primer on Eye Tracking

Eye tracking is the process of estimating the point of gaze at which the subject is looking. Existing eye tracking solutions can be categorized into the following three classes:

Video-oculography (VOG) is the most popular tracking approach, and has been widely adopted in commercial systems [12, 48–51]. A VOG-based eye tracker consists of an infrared light emitting diode that creates reflections on the cornea, and a camera that is sensitive to the infrared light to capture images of the eyes [52–54]. As the positions of the cornea reflections are almost constant during the eye movement, they are leveraged as reference points. Then, by calculating the relative positions between the center of the pupil and the corneal refection, the movement of the pupil can be estimated. Based on the application scenario, VOG-based eye trackers can be further classified into mobile-based trackers (MVOG), which are typically head mounted devices worn by the user (e.g., Pupil Core [53, 55]), and stationary trackers (SVOG) that are installed beneath a computer screen [48] or a car dashboard [12].

Electro-oculography (EOG) works on the principle that the electrical potential on the skin around the eye changes with the eye movements. Thus, electrodes are placed on the skin to measure the potentials, which can be leveraged to estimate the angular position of the eye [6]. Although EOG-based trackers are more computationally efficient than VOG-based counterparts, they suffer from poor tracking accuracy, owing to the interference from the facial muscles and head movements of the user [7].

Scleral search coils (SSC) is widely regarded as the most accurate tracking approach [7, 54]. In SSC, a contact lens embedded with a magnetic field sensor is inserted into the subject's eye. During tracking, an electromagnetic field is applied to the tracker by placing a magnetic frame around the user. The magnetic field sensor then provides measurement to estimate the eye movement [56]. However, as the subject needs a topical anesthetic before the use of SSC, it makes the approach less feasible in daily sensing scenarios.

3.2 Challenges in Gaze-based Activity Sensing

While recent studies have demonstrated the great potential of gaze-based cognitive context sensing [22, 57, 58], two important challenges, the *heterogeneity in human visual behavior* and the *infeasibility of collecting a large-scale gaze dataset* during the system training phase, hinder the pervasive adoption of existing solutions.

3.2.1 ***Heterogeneity in human visual behavior.*** Human visual behaviors are heterogeneous across *subjects*, *visual stimuli*, and *eye tracking devices*. Researchers have shown that subject differences, including the subject's language proficiency [8], personal interests [16], and engagement [17] with the visual material, have significant impact on the visual behavior. In addition, eye movement patterns are also influenced by the visual stimuli [14, 15]: gaze patterns are diverse among different visual stimuli, even though the same subject is performing the same activity. Lastly, the heterogeneity of eye tracking devices, owing to the difference in their underlying hardware (e.g., EOG versus VOG) and deployment scenarios (e.g., mobile versus stationary), also introduces additional variations in the captured eye movement signal.

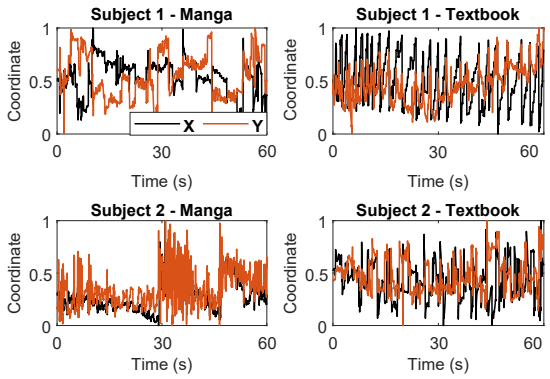


Figure 1: Examples of heterogeneity in visual behavior.

To illustrate the heterogeneity in human visual behavior, we take the public reading behavior dataset [15] as an example. Figure 1 shows the normalized 2D coordinates (with X and Y coordinates range from 0 to 1) of the captured gazes when two subjects are reading. Specifically, two variants of reading material, i.e., a manga (Japanese comics) and a textbook, are used as different visual stimuli. As shown in Figure 1, for the same reading activity, the eye movement patterns (the coordinates of the gazes) are highly diverse among subjects (i.e., subfigures in the same column indicate that different subjects read the same document differently) and visual stimuli (i.e., subfigures in the same row indicate that the eye movement patterns of the same subject differ among the stimuli).

Although the heterogeneity in eye movements is widely known to the eye tracking community [14, 15], its impact on the sensing performance is still largely overlooked. Existing gaze-based recognition systems rely heavily on hand-crafted features [6, 14, 15], which are extracted from different visual behaviors such as fixations, saccades, and blinks, to train supervised learning algorithms for classification. However, hand-crafted features are vulnerable to the intrinsic heterogeneity. Thus, in scenarios with high subject and stimuli diversities, the recognition accuracy of existing systems is poor with small sensing window (e.g., less than 60% with a sensing window of 15s [6, 14]). To improve the accuracy, a potential solution is to enlarge the sensing window. However, the improvements are modest: the overall accuracy is still less than 73% when a sensing window of 150s is used [6, 14, 34].

3.2.2 ***Accuracy deficiency due to limited gaze samples.*** An alternative solution to improve sensing accuracy is to collect a sufficiently large gaze dataset that covers all possible conditions, and takes advantage of the superior feature learning and classification capabilities of DNNs [59]. However, when deployed 'in the wild', a gaze-based recognition system will face the diversity in subjects, visual stimuli, and eye tracking hardware. Considering the countless possible combinations of all the dependencies, collecting a dataset that covers all different conditions is almost impossible. In addition, rising concerns on compromising user privacy from the eye movement data [22–24] also make such data collection infeasible. A recent survey of 124 people shows that over 70% of the participants

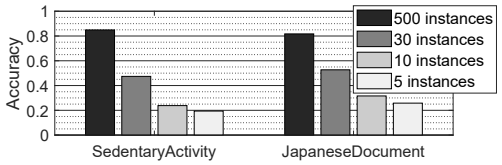


Figure 2: The recognition accuracy of an LSTM-based classifier on two public datasets, when different number of training instances are available for a new subject.

refuse to share their eye tracking data due to privacy concerns [24].

To demonstrate how the availability of gaze data affects the sensing performance, we examine the accuracy of an LSTM-based classifier (which has been proven to be powerful in time-series data classification [60–62]) on two public gaze datasets, i.e., SedentaryActivity [14] and JapaneseDocument [15]. The classifier has a single LSTM layer with 160 memory units. It takes the 2D coordinates of 900 gaze samples as the input. We investigate the recognition accuracy of the classifier when a *different number of training instances per class* are available for a new subject. The results are shown in Figure 2. Comparing with the *500-instances-per-class* base case, the classifier experiences over 30% and 60% accuracy drop when there are only 30 and 5 training instances available per class, respectively. Clearly, the lack of sufficient gaze data will lead to significant performance deficiency for gaze-based sensing systems.

4 SYSTEM OVERVIEW

System design: the overall design of GazeGraph is shown in Figure 3. An eye tracker captures the gaze of the subject and feeds a time series of gazes to GazeGraph as input. As a generalized system, GazeGraph is compatible with different eye trackers. In this work, we consider three types of eye trackers, specifically, two MVOG-based (i.e., SMI wearable eye tracking glasses [63] and Pupil Core [55]) and one SVOG-based (i.e., Tobii Pro X2 [64]). We focus mainly on the VOG-based eye tracking due to its wide adoption in commercial systems (e.g., advanced laptops [48, 49] and modern AR and VR headsets [50, 51]). The raw gaze signal is processed by the following system components:

- **Signal preprocessing (Section 5):** consists of a gaze filter, signal interpolation, and signal normalization units.
- **Gaze graph modeling (Section 6):** converts the preprocessed gaze signal into graphs which preserve both the temporal and spatial information of the original gaze signal. The gaze graphs are represented by two matrices, the *gaze distance matrix* and the *gaze orientation matrix*, that are used as the inputs of the few-shot gaze graph classifier for feature learning and classification.
- **Few-shot gaze graph learning (Section 7):** consists of the CNN-based gaze graph classifier and the few-shot gaze graph learning module. The former learns a low-dimensional representation of the gaze graph and uses it for recognition, while the latter enables the gaze graph classifier to quickly adapt to new gaze-based sensing scenarios (e.g, new sensing subjects) with a limited number of gaze instances required from the new scenarios for training.

Applications: we consider two typical gaze-based cognitive sensing applications:

- **Sedentary activity recognition:** tracking sedentary activities has several benefits. First, sedentary activities, such as reading and watching videos, occupy a dominant amount of time in people's daily lives. A recent report indicates that 25% of Americans are sedentary for more than eight hours a day [65]. Unfortunately, a sedentary lifestyle is associated with poor health, including an increased risk of heart disease and type 2 diabetes [65]. Thus, proactively monitoring the daily sedentary activities can provide more fine-grained information about people's lifestyles. Second, it enhances the context-awareness of existing systems. For instance, by detecting whether a user is playing video games or reading articles, the environment (e.g., smart home) or mobile AR/VR systems can be augmented to provide more immersive user-environment interaction. In this work, we consider two sedentary activity datasets, one public [14] and one collected by ourselves (Section 8.1), to examine the performance of GazeGraph in this application.

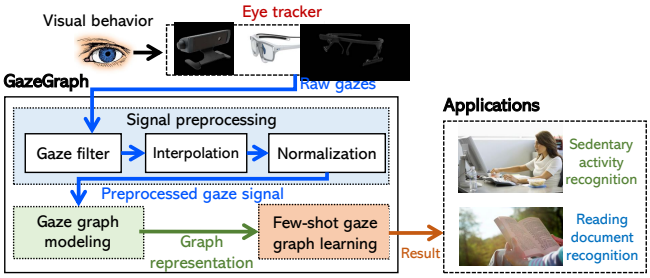


Figure 3: System architecture of GazeGraph.

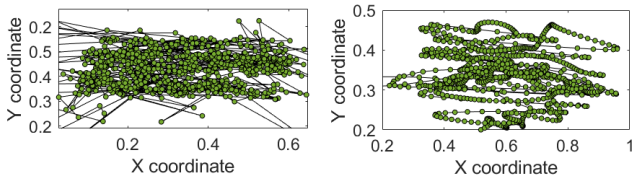
- **Reading behavior recognition:** reading is a cognitive activity that people perform every day. A sensing application that can track *what the user is reading* and *how long the user has read* is particularly valuable. For instance, teachers can leverage this application to monitor the reading habits of the students, and quantify their daily knowledge acquisition. Moreover, reading behavior recognition also provides insights about the user's reading interests which can be used by recommendation systems to deliver reading articles of interest to the user. In this work, leveraging the public reading activity dataset [15], we investigate the performance of GazeGraph in recognizing the type of document the subject is reading.

Other "killer apps": we believe that GazeGraph can enable many other game-changing applications. For instance, in the healthcare domain, GazeGraph can be used to monitor both physical and psychological health conditions, such as Alzheimer disease [66], bipolar disorder [67], and autism [68], among others [69]. In the field of industrial maintenance and training, GazeGraph can be coupled with AR/VR systems [50, 70] to monitor the user's "brain power" [70], assess trainee's cognitive workload in learning [9, 71], and to monitor worker's safety awareness when performing high-risk tasks [72].

5 SIGNAL PREPROCESSING

The signal preprocessing component prepares the raw gaze signal captured by the eye tracker for graph modeling and activity recognition. As shown in Figure 3, it consists of the gaze filter, interpolation, and normalization units.

First, the gaze filter removes raw gaze samples that are corrupted (e.g., eye tracker fails to estimate the gaze when the user is blinking or user's eyes are closing). Different eye trackers have different indicators to assess their confidence on the correctness of the gaze measurements. For the Tobii eye tracker [64], corrupted samples



(a) Raw gaze signal (b) Preprocessed gaze signal

Figure 4: Example of the raw and processed gaze signals when a subject is reading an article.

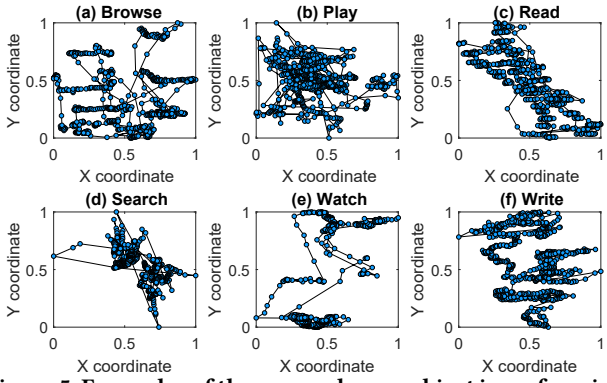
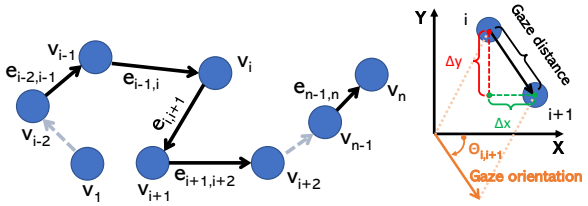


Figure 5: Examples of the gazes when a subject is performing six sedentary activities: (a) Browsing the Internet; (b) Playing video games; (c) Reading articles; (d) Searching the Internet; (e) Watching videos; and (f) Writing essays.

are removed by filtering any gaze measurements with a validity code larger than 0, whereas, for Pupil and SMI eye trackers [55], incorrect measurements are removed by filtering any samples with confidence level lower than 0.6. After filtering out the corrupted samples, we apply a median filter with a sliding window of 10s to detect and filter the outliers in the gazes (i.e., gaze samples that have a large Euclidean distance to the remaining samples in the sliding window). After filtering, spline interpolation is used to harmonize and resample the filtered gaze signal to its original length. Lastly, the normalization component scales the gaze measurements to a normalized 2D plane (with X and Y coordinates range from 0 to 1). The normalization eliminates the impact of different hardware calibrations on the gaze signal. Since the actual measurement of the gaze depends on the subject's field of view and the calibration process in correlating the gaze measurement range with the field of view, thus, comparing with the raw measurements provided by the eye tracker, the normalized signals are more robust against the variations in the hardware calibration. Figure 4 compares the raw gaze signal and the gaze signal processed by the three processing units when a subject is reading an article. The raw signal is noisy and contains many outliers. By contrast, the processed signal is clear and exhibits smooth 'left-to-right' eye movement patterns.

6 SPATIAL-TEMPORAL GRAPH MODELING FOR HUMAN VISUAL BEHAVIOR

Eye movements captured by eye trackers are represented by a time series of gazes. For instance, Figure 5 shows the gazes when a subject is performing six different sedentary activities. Each of the subfigures shows a sequence of 900 gazes in a normalized 2D plane, where gazes are denoted by nodes, and any two sequentially



(a) Example of a gaze graph. (b) Edge weights.

Figure 6: (a) Example of a temporal gaze graph $\mathcal{G}_{TGG}$ constructed from a time series of n gaze samples. (b) The gaze distance and gaze orientation between nodes v_i and v_{i+1}.

recorded gazes are connected by a link. In mathematics, this type of "node-link" structure, which models the geometric and pairwise relations among nodes, is known as the graph. Indeed, as shown in Figure 5, we can consider the whole "node-link" structure formed by the 900 gazes as a graph. Moreover, for different activities, the constructed graphs are distinct in their geometric patterns, which introduces opportunities for graph-based context recognition. Although graph-like depictions have been proposed for eye movement visualization [73], e.g., attention map for fixation [74] and scanpath visualization [75], there is no literature that has modeled human gazes as graphs for the purpose of cognitive context sensing. Below, we present the spatial-temporal gaze graph and introduce how to construct it from a time series of gazes.

6.1 Modeling Human Gazes as Graphs

6.1.1 Basic definitions. We first introduce some definitions that are used in graph modeling [76, 77]:

- **Definition 1: Graph**. *A graph is denoted as $\mathcal{G} = (V, E)$ with a set of n nodes $V = \{v_1, ..., v_n\}$ and a set of edges $E = \{e_{i,j}\}_{i,j=1}^{n}$, where $e_{i,j} \in E$ is an edge from node v_i to node v_j. Moreover, $\mathcal{G}$ is a homogeneous graph when all the nodes have the same node type and all the edges have the same edge type.*
- **Definition 2: Weighted Graph**. *A weighted graph is a graph in which each edge $e_{i,j} \in E$ is associated with a weight scalar $s_{i,j}$. For nodes v_i and v_j that are not linked by an edge, their weight $s_{i,j} = 0$, otherwise, $s_{i,j} > 0$. The weights of all the edges are stored in a $|V| * |V|$ weight matrix denoted as $\mathcal{W} = [s_{i,j}]_{i,j=1}^{n}$.*
- **Definition 3: Directed Graph**. *A directed graph is a graph in which each edge $e_{i,j} \in E$ with $s_{i,j} > 0$ is associated with a direction.*

6.1.2 Gaze graph modeling. Given the above definitions, we introduce the proposed gaze graph. Specifically, we first introduce a simple graph model that only captures the temporal structure of the gazes, followed by a complete model that preserves both spatial and temporal information of the gazes.

- **Definition 4: Temporal Gaze Graph (TGG)**. *For a sequence of n gazes, the associated temporal gaze graph, denoted as $\mathcal{G}_{TGG} = (V_{TGG}, E_{TGG})$, is a homogeneous weighted directed graph with weight matrix $\mathcal{W}_{TGG} = [s_{i,j}]_{i,j=1}^{n}$. Each node $v_i \in V_{TGG}$ corresponds to a gaze sample in the time series, and is assigned with a coordinate vector $\tilde{c}_i = (x_i, y_i)$ in the normalized 2D plane. Every pair of adjacent nodes, i.e., v_i and v_{i+1} for $1 \leq i < n$, are linked by a directed edge $e_{i,i+1}$. Thus, we have $|V_{TGG}| = n$ and $|E_{TGG}| = n - 1$.*

As an example, Figure 6(a) shows the temporal gaze graph con-

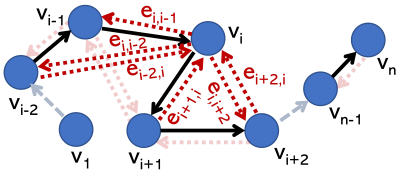


Figure 7: Example of building the spatial-temporal gaze graph with $k = 2$.

structed from a time series of n gaze samples. To characterize the local pairwise relation between the linked nodes, we propose the *gaze distance* and *gaze orientation* as two weight metrics. Figure 6(b) shows the two metrics for edge $e_{i,i+1}$. The gaze distance, denoted as $\mathsf{dist}_{i,i+1}$, reflects the *Euclidean distance* between nodes v_i and v_{i+1}, and can be obtained as:

$$\mathsf{dist}_{i,i+1} = ||\tilde{c}_{i+1} - \tilde{c}_i||, \tag{1}$$

where $\tilde{c}_i$ and $\tilde{c}_{i+1}$ are the coordinate vectors of v_i and v_{i+1}, respectively, and $||\cdot||$ is the Euclidean norm operation. Moreover, to capture *the geometry of the eye movement scanpath*, i.e., the eye movement direction between succeeding gazes, we introduce the gaze orientation, denoted as $\mathsf{orient}_{i,i+1}$, which can be obtained by:

$$\mathsf{orient}_{i,i+1} = \arctan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right), \tag{2}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) are the normalized coordinates of the two succeeding nodes. Then, the weight of directed edge $e_{i,i+1}$ can be defined as $\mathrm{s}_{i,i+1} = \left[\mathsf{dist}_{i,i+1}, \mathsf{orient}_{i,i+1}\right]$, and the weight matrix of the graph can be obtained by:

$$\mathcal{W}_{TGG} = \left[\mathrm{s}_{i,j}\right]_{i,j=1}^{n}, \text{where} \tag{3}$$

$$\mathrm{s}_{i,j} = \begin{cases} \left[\mathsf{dist}_{i,j}, \mathsf{orient}_{i,j}\right] & \text{if } v_i \text{ and } v_j \text{ are linked,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Finally, we can model the original n gaze samples by the constructed weight matrix $\mathcal{W}_{TGG}$ which consists of an $|n| * |n|$ gaze distance matrix $\mathcal{M}_{\mathsf{dist}} = \left[\mathsf{dist}_{i,j}\right]_{i,j=1}^{n}$ and an $|n|*|n|$ gaze orientation matrix $\mathcal{M}_{\mathsf{orient}} = \left[\mathsf{orient}_{i,j}\right]_{i,j=1}^{n}$.

6.1.3 Constructing the spatial-temporal gaze graph. The TGG defined in **Definition 4** captures only the *local pairwise relation between the temporally adjacent gazes*, i.e., only the succeeding nodes in the time series are connected in the graph. However, capturing the sequential and local pairwise relations between gazes is not enough for eye movement modeling. In practice, *many visual behaviors are represented by a small group of spatially correlated gazes*. These gazes are geographically close to each other in the 2D plane, but are not linked by any edges. For instance, a cluster of closely located gazes forms a visual fixation [14, 78] (the stationary state of the eyes) which is a widely used pattern for gaze-based sensing [6, 14, 15, 34]. To address this limitation, we propose the *k-hop spatial-temporal gaze graph* to effectively preserve both the sequential and the spatial information of the eye movement. We introduce the following definition:

• **Definition 5: k-hop Spatial-temporal Gaze Graph (k-STGG).** *For a sequence of n gazes, the associated k-STGG, denoted as $\mathcal{G}_{STGG} = (V_{STGG}, E_{STGG})$, is an extended TGG, where $V_{STGG} = V_{TGG}$ and $E_{STGG} \supseteq E_{TGG}$. Specifically, for nodes v_i and v_j in V_{STGG}, if they are within k hops, we link them by two directed edges, $e_{i,j}$ and $e_{j,i}$.*

The approach for building the k-STGG from the gaze samples is by examining each node $v_i \in V_{STGG}$ (where $V_{STGG} = V_{TGG}$, as

Algorithm 1 k-STGG Construction

Input: (1) A time series of n gaze samples with normalized coordinate vectors $\tilde{C} = \{\tilde{c}_i\}_{i=1}^{n}$ where $\tilde{c}_i = (x_i, y_i)$; (2) a parameter k that indicates k-hop neighbors to be connected in the graph.

$\mathcal{M}_{\mathsf{dist}} = \mathcal{M}_{\mathsf{orient}} = [0]_{n\times n}$; ▷ initiate the weight matrices
for $i = 1, ..., n$ **do** ▷ Examining all n nodes
  # ***For node i, examining all its k-hop neighbors***
  for $j \in \{i-k, ..., i+k\}$ and $j > 0$ **do**
    # ***Calculating the gaze distance and gaze orientation***
    $\mathsf{dist}_{i,j} = ||\tilde{c}_j - \tilde{c}_i||$; $\mathsf{orient}_{i,j} = \arctan\left(\frac{y_j - y_i}{x_j - x_i}\right)$;
    # ***Updating the weight matrices***
    $\mathcal{M}_{\mathsf{dist}}(i, j) = \mathsf{dist}_{i,j}$; $\mathcal{M}_{\mathsf{orient}}(i, j) = \mathsf{orient}_{i,j}$;

Output: $\mathcal{M}_{\mathsf{dist}}$, $\mathcal{M}_{\mathsf{orient}}$;

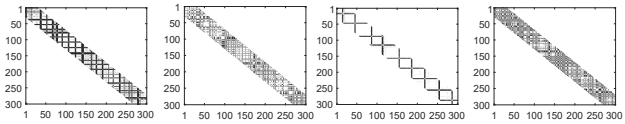


Figure 8: Examples of $\mathcal{M}_{\mathsf{dist}}$ and $\mathcal{M}_{\mathsf{orient}}$ of the k-STGG (with $k = 30$) that is constructed from a sequence of $n = 300$ gazes: (a) and (b) show the $\mathcal{M}_{\mathsf{dist}}$ and $\mathcal{M}_{\mathsf{orient}}$ when the subject is 'browsing the Internet'; (c) and (d) are the matrices when the subject is 'reading articles'.

STGG and TGG have the same node set), and adding edges between v_i and each of its k-hop neighbors. As an example, Figure 7 shows how to construct k-STGG with $k = 2$ from the original TGG. As shown, in addition to the edges of the original TGG (indicated by black solid lines), six new edges (indicated by red dashed lines) are added to the examining node v_i by connecting it with all its 2-hop neighbors (v_{i-2}, v_{i-1}, v_{i+1} and v_{i+2}). Then, the k-STGG is constructed by examining and adding new edges for all the n nodes. The details of constructing the k-STGG for a time series of n gaze samples are shown in Algorithm 1. The final outputs are the gaze distance matrix $\mathcal{M}_{\mathsf{dist}}$ and the gaze orientation matrix $\mathcal{M}_{\mathsf{orient}}$. As an example, Figure 8 visualizes the $\mathcal{M}_{\mathsf{dist}}$ and $\mathcal{M}_{\mathsf{orient}}$ of the k-STGG (with $k = 30$) that is constructed from a time series of $n = 300$ gazes. Figures 8(a) and (b) show the $\mathcal{M}_{\mathsf{dist}}$ and $\mathcal{M}_{\mathsf{orient}}$ of the gazes when the subject is 'browsing the Internet'. In comparison, Figures 8(c) and (d) visualize the two matrices when the subject is 'reading an article'. We can clearly see the distinct patterns in the constructed matrices of the two activities.

Note that by configuring k, k-STGG pays *selective attention* to the gaze samples in the sensing window. It only preserves the pairwise relations between the gazes that are temporally close (within k-hop) and discards the others. A careful selection of k not only ensures a lower computational burden in graph construction, but also maintains the most *targeted* and essential gaze pairs (gazes with close spatial-temporal relation) and discards the maleficent ones (gazes with loose spatial-temporal relation) for the gaze graph classifier. In summary, the k-STGG converts a time series of gazes into a graph which largely preserves both the temporal (through the local pairwise relation between the single-hop nodes) and the spatial (through the k-hop neighbor nodes) structure of the original eye movement signal. Below, we show how to leverage the constructed k-STGG for feature learning and cognitive context recognition.

Table 1: The network design of the gaze graph classifier.

Layer	Size In	Size Out	Filter
conv1	$90 \times 90 \times 2$	$90 \times 90 \times 32$	5×5, 1
conv2	$90 \times 90 \times 32$	$86 \times 86 \times 32$	5×5, 1
pool1	$86 \times 86 \times 32$	$43 \times 43 \times 32$	2×2, 2
conv3	$43 \times 43 \times 32$	$39 \times 39 \times 64$	5×5, 1
pool2	$39 \times 39 \times 64$	$19 \times 19 \times 64$	2×2, 2
conv4	$19 \times 19 \times 64$	$15 \times 15 \times 64$	5×5, 1
pool3	$15 \times 15 \times 64$	$7 \times 7 \times 64$	2×2, 2
flatten	$7 \times 7 \times 64$	3136	
fc	3136	64	
fc	64	$\|Class\|$	

7 FEW-SHOT GRAPH REPRESENTATION LEARNING AND CLASSIFICATION

7.1 CNN-based Gaze Graph Classifier

Learning a robust and generalized low-dimensional data representation of the gaze graphs is a challenging task. The difficulty comes from the complex graph structure and the intrinsic heterogeneity in human visual behavior which introduces additional variations in the graphs. Below, we propose the gaze graph classifier for the graph-based data representation learning and classification.

The classifier takes the graph matrices generated by the k-STGG as the inputs, which can be considered as data modalities in the form of 2D arrays containing graph weights in two different information channels: the Euclidean distance and the orientation. Then, the spatial-temporal features embedded in the graph are obtained by conducting multiple levels of non-linear operations. Each of the operations transforms the data representation learnt at the previous level (starting with the original weight matrices) into a representation at a higher and slightly more abstract level [21]. In particular, the multi-layer non-linear operations (in the form of nonlinear activation function or pooling layers) make the obtained data representation sensitive to minute details embedded in the matrices (e.g., temporal and spatial features of the gazes), and insensitive to large irrelevant variations that result from the eye movement diversity. Lastly, the learnt gaze graph representation is fed into the fully connected layers for context recognition.

The details of the classifier are given in Table 1. The classifier consists of four convolutional layers, three max pooling layers, one flatten layer, and two fully connected layers. The ReLU is used as the activation function after each of the convolutional layers. Moreover, a dropout layer has been added after each of the three max pooling layers to prevent overfitting when a small-scale training dataset is used [79]. Before feeding to the network, we reshape both of the two weight matrices to the size of 90×90. Thus, the size of the input to the first convolutional layer is 90×90×2, where the two channels correspond to the reshaped $\mathcal{M}_{\mathsf{dist}}$ and $\mathcal{M}_{\mathsf{orient}}$, respectively. The output size of the last fully connected layer can be adjusted based on the number of classes in the sensing task ($|Class|$).

7.2 Few-shot Gaze Graph Learning Module

Although the proposed gaze graph classifier adopts a shallow network design, it still requires a large-scale gaze dataset to ensure good performance. Unfortunately, the countless 'in the wild' variations and the rising privacy concerns about eye movement data make the collection of a large-scale gaze dataset infeasible (Section 3.2). To address the performance limitation in this 'small gaze data' regime, we formulate the gaze-based cognitive context sensing as a few-shot learning problem. In general, the goal of few-shot learning is to train a machine learning model that can adapt to a new learning task with few samples [25, 26]. A broad family of techniques to address this problem is known as meta-learning [27–29, 46] which aims to learn a new task by learning how to learn [27]. Borrowing concepts from the model-agnostic meta-learning [28], we design a few-shot gaze graph learning module which incorporates the *meta-training* and the *deployment* stages. In the meta-training stage, the module learns a good parameter initialization for the gaze graph classifier using an unseen source dataset, such that, in the deployment stage, the classifier can quickly adapt to new sensing conditions (e.g., new subjects, unseen visual stimuli, or new eye trackers) after a few learning iterations (e.g., ten gradient steps) with a small number of gaze instances from the new conditions (e.g., five instances per class).

Problem formulation and overall design. Formally, we denote the gaze graph classifier as f with network parameters θ. In the meta-training stage, f is trained on a set of tasks, denoted as $\mathcal{T}$, generated from a source dataset $\mathcal{D}_S$ which contains gaze instances collected from diverse sensing conditions (e.g., different subjects). Each task $\mathcal{T}_i \in \mathcal{T}$ is a K-shot M-way classification problem, where the classifier aims to recognize M classes of activity by using K labelled instances for each of the activities (K is a small number, e.g., 5 or 10). Moreover, each task $\mathcal{T}_i$ is associated with a support set $S_{\mathcal{T}_i}$ and a query set $S_{\mathcal{Q}_i}$. The two sets are disjoint with each other ($\mathcal{S}_{\mathcal{T}_i} \cap \mathcal{Q}_{\mathcal{T}_i} = \emptyset$) and each set contains only $K \cdot M$ instances. In essence, in the concept of meta-learning, the entire tasks are treated as training examples [26, 28], and the classifier learns how to solve a future K-shot M-way classification task by learning from this collection of tasks $\mathcal{T}$. The output of the meta-training stage is a good initialization for θ, denoted as $\acute{\theta}$, which is learned from $\mathcal{T}$. Lastly, in the deployment stage, $f_{\acute{\theta}}$ can be adapted to any new sensing task with a few gradient steps and training instances needed.

Meta-training. The classifier f is randomly initialized with parameters θ_0, and then adapted to all individual tasks in $\mathcal{T}$. We can consider each task $\mathcal{T}_i$ mimicking the situation where f is adapted to learn and solve an unseen K-shot M-way gaze-based classification problem with only $K \cdot M$ samples. Specifically, for each task $\mathcal{T}_i \in \mathcal{T}$, f is trained using the associated support set $S_{\mathcal{T}_i}$, and it learns a new task-specific parameters $\theta'_{\mathcal{T}_i}$ (tuned from the initial parameters θ_0) via gradient descent update:

$$\theta'_{\mathcal{T}_i} = \theta_0 - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_0}, S_{\mathcal{T}_i}), \tag{5}$$

where α is a fixed hyperparameter that controls the learning rate of individual tasks; $\mathcal{L}_{\mathcal{T}_i}(f_\theta, S_{\mathcal{T}_i})$ is the task-specific cross-entropy loss of f_θ on the support set $S_{\mathcal{T}_i}$ and can be further defined as:

$$\mathcal{L}_{\mathcal{T}_i}(f_\theta, S_{\mathcal{T}_i}) = \sum_{(\mathrm{x}^j, \mathrm{y}^j) \in S_{\mathcal{T}_i}} \mathrm{y}^j \log f_\theta(\mathrm{x}^j) + (1 - \mathrm{y}^j) \log f_\theta(1 - \mathrm{x}^j), \tag{6}$$

where $(\mathrm{x}^j, \mathrm{y}^j)$ denotes the jth sample in S_{T_i}. After obtaining the task-specific parameters for all $\mathcal{T}_i$, an across-tasks parameters $\acute{\theta}$ can be computed by optimizing the following objective function:

$$\arg\min_\theta \sum_{\mathcal{T}_i \in \mathcal{T}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}}, Q_{\mathcal{T}_i}), \tag{7}$$

which minimizes the sum of the task-specific loss for all tasks in $\mathcal{T}$. Note that the testing loss for each task $\mathcal{T}_i$ is obtained by applying

Algorithm 2 Meta-training for the Gaze Graph Classifier

Input: (1) Source dataset $\mathcal{D}_S$; (2) gaze graph classifier f; (3) α and β.

1: $f_\theta \leftarrow \theta_0$; ▷ initialize f with random parameters θ_0
2: **while** not done **do**
3: Generate a batch of tasks $\mathcal{T}$ from source dataset $\mathcal{D}_S$;
4: **for all generated** $\mathcal{T}_i \in \mathcal{T}$ **do**
5: $\mathcal{S}_{\mathcal{T}_i} \leftarrow$ Sample $K \cdot M$ instances for $\mathcal{T}_i$;
6: $\mathcal{Q}_{\mathcal{T}_i} \leftarrow$ Sample $K \cdot M$ instances for $\mathcal{T}_i$ where $\mathcal{S}_{\mathcal{T}_i} \cap \mathcal{Q}_{\mathcal{T}_i} = \emptyset$;
7: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ on $\mathcal{S}_{\mathcal{T}_i}$ based on loss $\mathcal{L}_{\mathcal{T}_i}(f_\theta, S_{\mathcal{T}_i})$;
8: Compute the task-specific model parameters $\theta'_{\mathcal{T}_i}$ using gradient descent: $\theta'_{\mathcal{T}_i} = \theta_0 - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_0}, S_{\mathcal{T}_i})$;
9: Obtain the across-task initialization $\acute{\theta}$ via minimizing the sum of all task-specific losses: $\acute{\theta} \leftarrow \theta_0 - \beta \nabla_\theta \sum_{\mathcal{T}_i \in \mathcal{T}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}}, Q_{\mathcal{T}_i})$;

Output: Output gaze classifier $f_{\acute{\theta}}$ with parameters $\acute{\theta}$;

the updated task-specific classifier $f_{\theta'_{\mathcal{T}_i}}$ on the corresponding query set $Q_{\mathcal{T}_i}$. The across-tasks optimization is performed via stochastic gradient descent (SGD) [28], such that $\acute{\theta}$ is obtained by:

$$\acute{\theta} \leftarrow \theta_0 - \beta \nabla_\theta \sum_{\mathcal{T}_i \in \mathcal{T}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_{\mathcal{T}_i}}, Q_{\mathcal{T}_i}), \tag{8}$$

where β is a fixed hyperparameter that determines the learning rate of SGD optimization. The final outputs of the meta-training stage are the optimized parameters $\acute{\theta}$ for the classifier. The algorithm for training the few-shot gaze graph classifier is given in Algorithm 2.

Deployment. After initializing with the optimized parameters $\acute{\theta}$, the classifier $f_{\acute{\theta}}$ can quickly adapt to any new sensing task (e.g., deployment scenario with new sensing subjects, new visual stimuli, or new eye tracker) with only $K \cdot M$ samples. The new sensing task can be considered as a K-shot M-way classification problem $\mathcal{T}_{new}$ with dataset $\mathcal{D}_T$ (e.g., a new gaze dataset that is collected from a new subject). Given the almost countless 'in the wild' sensing conditions for gaze-based activity recognition, $\mathcal{D}_T$ and $\mathcal{D}_S$ are disjoint ($\mathcal{D}_S \cap \mathcal{D}_T = \emptyset$). We can adapt the optimized gaze graph classifier $f_{\acute{\theta}}$ to the new task $\mathcal{T}_{new}$ using a few gradient steps. The new parameters θ_T that are fine-tuned on $\mathcal{D}_T$ can be defined as:

$$\theta_T = \acute{\theta} - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_T}(f_{\acute{\theta}}), \tag{9}$$

where the value of α is as same as that in Equation 5. The final output of the few-shot learning module is the gaze graph classifier f_{θ_T} with parameters θ_T that are fine-tuned to the new sensing task $\mathcal{T}_{new}$. We evaluate the few-shot learning module in Section 8.3.

8 EVALUATION

8.1 Datasets

We consider two gaze-based sensing applications and three datasets in the evaluation to demonstrate the generality of GazeGraph.

8.1.1 Dataset we collected. First, we have collected a gaze dataset, denoted as **DesktopActivity**, from eight subjects (four female and four male, aged between 24 and 35; all subjects are fluent in English, with Spanish (1), English (2), and Chinese (5) as their first language) using the Pupil Core eye tracker [53, 55]. The study is approved by our institution's Institutional Review Board. During data collection, the subjects wear the eye tracker and sit in front of the computer screen (a 34-inch display) at a distance of approximately 50cm. We conduct the manufacturer's default on-screen five-points calibration for each of the subjects. Note that only one calibration is needed per subject, and the subjects can move their heads and upper bodies freely during the experiment. This is achieved as GazeGraph focuses on the relative movements of gazes rather than the exact coordinates of the gazes.

Activities: we consider six different desktop activities that are commonly performed in daily life.

- **Browse**: subjects browse public news websites or blogs. The websites visited by the subjects are different: three subjects gravitated toward visiting websites written in English while the other five subjects visited websites written mainly in Chinese.
- **Play**: subjects are asked to play simple online games. We consider two different games: one requiring the subjects to look ahead horizontally (Classic Super Mario [80]), while the other requiring the subjects to look in all directions to navigate the game character (Agario [81]). The instructions of the games are given to the subjects before playing.
- **Read**: subjects read digital content displayed on a computer screen. Three reading materials in English are prepared: a Wikipedia article, a research paper in a two-column format, a textbook in a single-column format. These materials differ in both text layout and the number of figures embedded.
- **Search**: we ask subjects to search for answers to a list of predefined questions using a web-based search engine. For each of the subjects, the questions are randomly ordered to ensure variations (and thus, different visual stimuli). The search history is cleared before every session so that all subjects start from the same baseline.
- **Watch**: subjects watch a short video played on the screen. We consider two videos with a different number of main characters (one with two main characters and the other with more than three characters); also one video has subtitles shown on the bottom.
- **Write**: subjects are asked to write an essay in English using the Microsoft Word installed on the computer.

The subjects are asked to perform each of the six activities for five minutes. They can choose one of the stimuli prepared for the Read, Watch, and Play. The gazes is recorded at a 30Hz sampling rate. Examples of the recorded gazes are shown in Figure 5.

8.1.2 Other datasets. We consider two public datasets:

SedentaryActivity [14]: collected from 24 subjects (16 male and 8 female, aged between 24 and 48). The gazes are recorded at 30Hz using a Tobii Pro X2 [64] SVOG-based eye tracker. Subjects perform eight activities including: five common desktop activities, i.e., *read*, *watch*, *browse*, *search*, and *play*; and three software development activities, i.e., *interpret* (interpreting the output of a short program code), *debug* (fixing bugs in a computer program), and *write* (implementing three program functions). Each of the activities has three variants to mimic different visual stimuli. Subjects perform each of the activities for five minutes. The sensing task is to *recognize which activity the subject is performing*.

JapaneseDocument [15]: collected from eight subjects (four male and four female, aged between 21 and 32). The subjects read five different types of documents written in Japanese: *novel, manga, fashion magazine, newspaper, and textbook*. During data collection, all subjects read each of the five documents for 10 minutes. Their gazes were recorded at 30Hz using a mobile head-mounted eye tracker, the SMI wearable eye tracking glasses [63]. The sensing

Table 2: List of the 18 hand-crafted eye movement features used by the conventional feature-based methods.

		Features
Saccade	length	sacc-mean, sacc-variance, sacc-std
	direction	sacc-up, sacc-up-right, sacc-down-right, sacc-right, sacc-up-down, sacc-down-left, sacc-left, sacc-left-up
Fixation	duration	fix-mean, fix-variance, fix-std
	rate	fix-rate
	slope	fix-slope
	dispersion	fix-disp-area
	count	fix-count

task is to *recognize the type of the document* the subject is reading.

8.2 Overall Performance

We implement GazeGraph using the Keras 2.3 library on top of the TensorFlow 2.0 framework. We employ the Adam optimizer [82] during the training. We use the F1 score as the performance metric.

8.2.1 Performance improvement over state-of-the-art methods. We compare GazeGraph with a conventional *feature-based method* [3, 15], and an *LSTM-based classifier* [20]. Specifically, for the feature-based method, we first implement the fixation filter [83] to obtain the fixations and saccades from the raw gaze signal. Then, we extract 18 commonly used features [3, 14, 15] from the detected fixations and saccades (shown in Table 2), and feed them to the Support Vector Machines (SVM) for training and classification. In brief, the saccade-based features include: three statistical features that capture the mean, variance, and standard deviation of the saccade length, and eight direction features that count the number of saccades that appear in each of the eight saccade directions [15]. Similarly, the fixation-based set includes: three statistical features that capture the mean, variance and standard deviation of the fixation duration; the number of fixations that appear per second (fix-rate); the slope over the fixations in the sensing time window (fix-slope); the dispersion area of the fixations (fix-disp-area); and the total number of fixations in the sensing time window (fix-count). The LSTM-based classifier has a single LSTM layer with 160 memory units. It takes the preprocessed 2D coordinates of the gaze signal as the input. The LSTM classifier is trained using the Adam optimizer with a learning rate of 0.001. We consider LSTM as a baseline classifier given its good performance in time-series data classification [60–62]. Lastly, for GazeGraph, we construct the gaze graph using the k-STGG model (Algorithm 1) with $k = 10$.

We evaluate the three classification methods on the three datasets. For each of the datasets, we mix the data from all subjects together and perform the 10-fold cross-validation. Three small sensing window sizes are considered, i.e., 10s, 20s, and 30s, to ensure short recognition delay. We have carefully monitored the training, validation, and testing curves of the classifier to ensure there is no overfitting. The results are shown in Figure 9. First, for all three methods, the F1 score increases with the window size, as a larger sensing window contains more information about eye movements. Second, in all scenarios, GazeGraph achieves the highest performance. Specifically, given different window sizes, GazeGraph outperforms the conventional feature-based method significantly by 37–39%, 49–54%, and 50–53% on the SedentaryActivity, JapaneseDocument, and DesktopActivity datasets, respectively. Similarly, GazeGraph outperforms the LSTM-based method by 10–16%, 18–22%, and 19–23% on the three datasets, respectively.

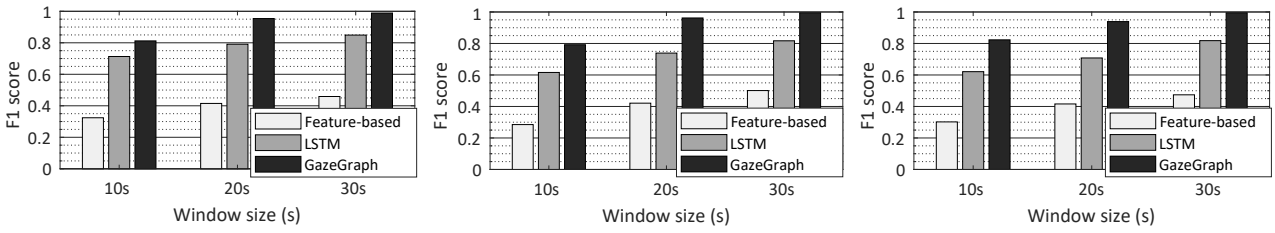
The improvements of GazeGraph over the conventional hand-crafted feature based method indicate its superiority in feature learning. Moreover, since the LSTM-based classifier is powerful in learning the *temporal and sequential* information of the gaze signal [60, 61] (and thus, it outperforms the feature-based method by a large margin), the improvements of GazeGraph over the LSTM-based method further demonstrate the capability of the proposed graph modeling and the CNN-based gaze graph classifier in preserving and learning the *spatial-temporal* features of the gaze signal.

8.2.2 Micro-benchmarks. Below, we examine how different weight metrics and graph construction methods affect GazeGraph.

Impact of weight metrics: first, we evaluate the performance of GazeGraph given different edge weight metrics used in the graph modeling. Specifically, we compare the proposed gaze distance and gaze orientation with conventional metrics, i.e., Euclidean, Cosine, and Manhattan distances, that are widely used in the graph modeling literature [76]. The Euclidean distance captures the straight-line distance between the two gaze points in the 2D space, and it is calculated the same as the gaze distance (Equation 1). The Cosine distance for two nodes v_i and v_j with 2D coordinate vectors $\tilde{c}_i$ and $\tilde{c}_j$ can be calculated by $\mathsf{Cosine}_{i,j} = 1 - \frac{\tilde{c}_i \cdot \tilde{c}_j}{||\tilde{c}_i|| \times ||\tilde{c}_j||}$, where the latter part is known as the cosine similarity and has been widely used to measure the cohesion and similarity between two vectors irrespective of their sizes [76]. Lastly, the Manhattan distance measures the absolute differences between coordinates of a pair of gaze points v_i and v_j, and it is obtained by $\mathsf{Manhattan}_{i,j} = |x_j - x_i| + |y_j - y_i|$.

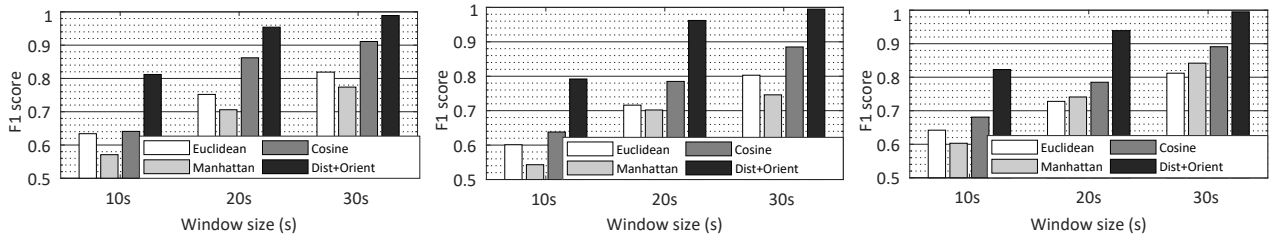
For all the four weight metrics, we construct the gaze graphs using the k-STGG model with $k = 10$. The results are shown in Figure 10. The proposed weight metrics, i.e., gaze distance and gaze orientation (Dist + Orient), outperform all the examined conventional weight metrics. Specifically, Dist + Orient outperforms the best conventional metric, Cosine distance, by 8–17%, 11–18%, and 10–15% on the three datasets, respectively. Moreover, when comparing with the second-best conventional metric, Euclidean distance, Dist + Orient achieves 15–18%, 19–25%, and 18–21% improvement on the three datasets, respectively. These results indicate the effectiveness of the proposed weight metrics in capturing both the *pairwise distance* (by the gaze distance metric) and the *geometry of the eye movement scanpath* (by the gaze orientation metric).

Impact of graph construction methods: next, we investigate how different constructions of the gaze graph will affect the recognition performance. Specifically, we use different k values when generating the k-STGG from the gaze signal (Algorithm 1). We consider six different configurations, namely, $k \in \{1, 2, 5, 10, 30, n\}$, where n is the total number of gaze samples in the sensing window. In particular, $k = 1$ represents the simplest temporal gaze graph (Definition 5), whereas $k = n$ results in a directed complete graph where every pair of nodes in the graph is connected. Figure 11 shows how k affects the recognition performance of GazeGraph on the three datasets. The results indicate that the optimal selection of k differs among the datasets (sensing task on hand): GazeGraph achieves the highest performance on the JapaneseDocument and DesktopActivity datasets with $k = 5$, while $k = 10$ leads to the best performance for the SedentaryActivity dataset. By contrast, at the two ends of the k spectrum, k-STGG with either $k = 1$ (the



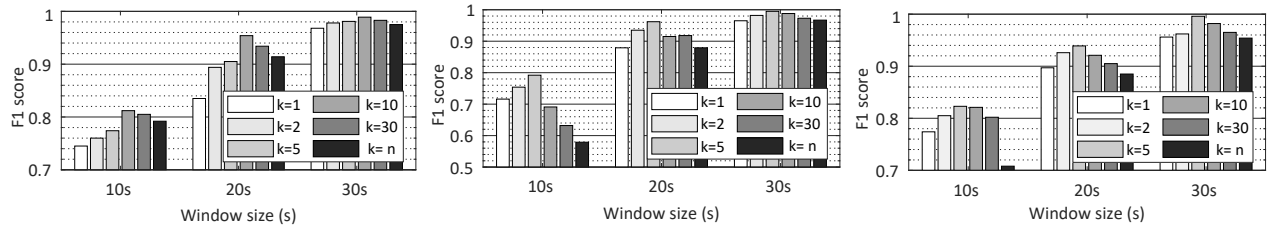
(a) SedentaryActivity (b) JapaneseDocument (c) DesktopActivity

Figure 9: Performance of different classification methods on the three datasets.



(a) SedentaryActivity (b) JapaneseDocument (c) DesktopActivity

Figure 10: Performance of GazeGraph on the three datasets given different weight metrics.



(a) SedentaryActivity (b) JapaneseDocument (c) DesktopActivity

Figure 11: Performance of GazeGraph given different k used in the construction of k-STGG.

single-hop graph that captures only the pairwise relations between temporally adjacent gazes) or $k = n$ (the complete graph that blindly includes the pairwise relations between all gazes) results in poor performance, especially when the window size is small (10s).

Moreover, as discussed in Section 6.1.3, the k-STGG is designed to capture the spatial relation between neighboring gazes, such as fixation, for which gazes are spatially clustered together. Physiological studies [84] have reported that the mean fixation duration in reading, scene perception, and visual research is 180 to 330ms. Thus, with a 30Hz sampling rate, there are 5 to 10 gaze samples within this fixation duration. Interestingly, as shown in Figure 11, k-STGG achieves the highest recognition performance when k equals to 5 or 10. This finding ties the parameter selection of GazeGraph to the intrinsic physiological behavior. Overall, with a 30s window size, GazeGraph achieves the F1 score of 0.96 on all the three datasets given a properly selected k.

8.3 Performance in Few-shot Scenarios

Below, we examine GazeGraph in the few-shot learning scenarios. Specifically, we formulate the recognition tasks on the SedentaryActivity, JapaneseDocument, and DesktopActivity datasets as K-shot M-way classification problems, where M is the number of classes in the dataset (i.e., M equals to 8, 5, and 6 for the three datasets, respectively), and we consider the 5 and 10-shot cases (i.e., K equals to 5 or 10). Specifically, for each of the three datasets, we conduct leave-one-subject-out experiments, in which the data collected from one subject is used as the *target dataset* and the data collected from all the other subjects acts as the *source dataset*. The testing subject simulates the scenario where the system is deployed to a new subject with limited training samples available ($K \cdot M$ samples). Note that, as the three datasets contain different numbers of classes, we are not able to perform cross-dataset evaluation, and only consider the evaluation in the dataset-dependent manner. We leave the across-applications and across-eye trackers evaluations as the future work when such datasets become available.

Baselines: we compare the proposed few-shot learning module with two baseline training strategies: (1) we use the few-shot samples from target dataset to train the gaze graph classifier and test it using the remaining data in the target dataset. This represents the subject-dependent training strategy; (2) we employ transfer learning [33] to transfer the knowledge from the source domain (e.g., known subjects) to the target domain (e.g., new subjects). Transfer learning is widely used in mobile sensing applications [32, 85] as well as eye tracking-based emotion recognition [38] to address the domain adaptation problem. In brief, we first train the gaze graph classifier on the source dataset. Then, we fix the pre-trained parameters of all the convolutional layers (architecture shown in Table 1), and fine-tune the parameters of the fully connected layers using

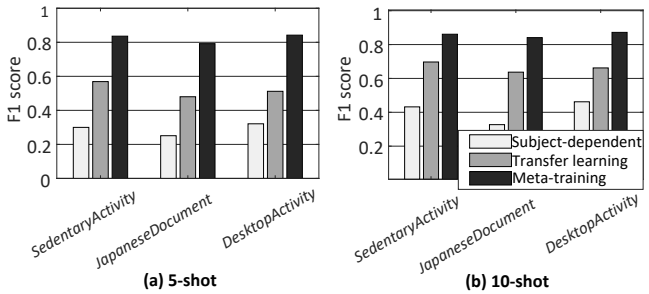


Figure 12: Performance of gaze graph classifier in the few-shot learning scenarios with different training strategies.

the few-shot samples from the target dataset. The assumption here is that the pre-trained parameters of the convolutional layers are reusable for similar learning problem [33]; (3) lastly, we train the gaze graph classifier using the source dataset by following the meta-training phase (Algorithm 2), and fine-tune it on the target dataset. Specifically, the meta-training phase randomly generates tasks from the source dataset. Each of the generated tasks is associated with the disjoint support set and query set. In our implementation, both support set and query set contain $M \cdot K$ samples from the source dataset. In the meta-training stage, the task-specific parameters ($\theta'_{\mathcal{T}_i}$) and the across-task optimized parameters ($\acute{\theta}$) are obtained by five gradient descent updates. The hyperparameters α and β are set as 0.01 and 0.001, respectively. In the deployment stage, we use ten gradient steps to fine-tune the parameters $\acute{\theta}$ for the target dataset. Note that all three methods use the gaze graph classifier as the base classifier and leverage the k-STGG with $k = 5$ to construct gaze graphs from the gaze signal. The sensing window size is 30s.

Recognition accuracy: Figure 12 shows the F1 score of the three methods. The results are averaged over all the subjects used in the leave-one-subject-out experiment. There are 24, 8, and 8 subjects in the SedentaryActivity, JapaneseDocument, and DesktopActivity datasets, respectively. As shown, the subject-dependent training strategy achieves the worst performance, as the limited training samples ($K \cdot M$) lead to overfitting. The transfer learning approach performs better, but its F1 score is lower than 60% and 70% in the 5-shot and 10-shot scenarios, respectively. In all scenarios, the proposed meta-training strategy achieves the best performance: in the 5-shot case, it outperforms transfer learning by 27%, 31%, and 33% on the three datasets, respectively; in the 10-shot case, it achieves a 16%, 20%, and 21% improvement on the three datasets, respectively. The results demonstrate the superiority of the proposed few-shot gaze graph learning module in dealing with the 'in the wild' unseen sensing tasks/scenarios with limited training samples available. This allows us to significantly diminish the expensive and privacy-compromising large-scale data collection by simply collecting 5 or 10 samples per class from the new subject.

Adaptation overhead: another advantage of the few-shot learning strategy over conventional approaches is its low adaptation overhead, i.e., the training time that is needed to adapt the classifier to the new sensing task/scenario. To demonstrate this, we investigate how the recognition accuracy of meta-training and the transfer learning counterpart change over the training epoch, and how many training epochs are required for them to converge. Here, one epoch equals to the time that is needed to train the classifier

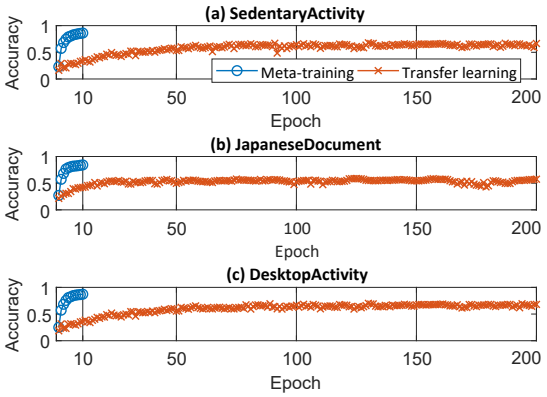


Figure 13: Recognition accuracy changes over training epochs on three different datasets.

on the entire samples ($K \cdot M$ samples) in training dataset. Figure 13 shows how the recognition accuracy changes over the number of epochs in the 10-shot case. We can see that meta-training requires significantly less adaptation overhead (only ten epochs) and achieves a higher recognition accuracy compared to the transfer learning approach (more than 50 epochs) on all the three datasets. The results indicate that the proposed few-shot gaze graph learning module can adapt quickly to new subjects with few training samples while maintaining good recognition performance.

The sub-par performance of transfer learning results from its underlying mechanism: during training, it fine-tunes only fully connected layers of the classifier, but reuses the convolutional layers that are pre-trained on data from the source domain (under the assumption that the feature distribution of the target domain is similar to that of the source domain, so that the pre-trained parameters are transferable [33]). However, due to the diversity between subjects, when fine-tuning the classifier with only a few-shot instances from the target domain (a new subject), the large gap between the feature distributions of the source and target domains invalidates the underlying assumption of transfer learning. *Consequently, regardless of the number of training epochs, transfer learning always results in a sub-par solution.* By contrast, instead of assuming the similarity between the two domains, the meta-training approach identifies a more generalized network initialization by learning from a large number of tasks $\mathcal{T}$ that have different feature distributions, such that the classifier can quickly adapt to a new target domain with few-shot instances while assuring good performance.

8.4 System Profiling

Measurement setup. We consider two application scenarios: (1) the user is using the *stationary eye trackers*, e.g., Tobii Pro X2 [64], that are integrated with desktops and laptops [11]. The captured gazes are processed directly on the powerful desktops or laptops; and (2) the user is wearing the *mobile eye trackers*, e.g., the Pupil Core [55]. As most mobile eye trackers do not support on-device inference with DNNs, we consider the case where the captured gaze samples are offloaded over a WiFi network to the edge server for processing and classification. We employ a laptop (equipped with an Intel i7-7700HQ CPU and a Nvidia GTX 1050 GPU) and a desktop (equipped with an Intel i7-8700k CPU and a Nvidia GTX 1080 GPU) as the edge platforms.

Table 3: The averaged computation latency (in ms) of GazeGraph when deployed on different hardware platforms. The variances of latency are shown in the parentheses.

		Laptop	Desktop
Signal preprocessing		0.46 (0.08)	0.32 (0.04)
Gaze graph construction	k=1	9.51 (0.13)	7.44 (0.37)
	k=2	10.39 (0.47)	7.52 (0.35)
	k=5	10.52 (0.66)	7.63 (0.19)
	k=10	11.45 (0.34)	7.73 (0.24)
	k=30	11.52 (0.32)	7.92 (0.32)
	k=900	12.62 (0.51)	8.09 (0.21)
Inference	CPU	3.07 (0.09)	1.48 (0.11)
	GPU	1.31 (0.13)	1.03 (0.07)
Overall (with $k = 5$)		12.29	8.98

Computation latency: we tear down the system pipeline into three components: the *signal preprocessing*, the *gaze graph construction*, and the *inference* using the CNN-based gaze graph classifier. The signal preprocessing and the gaze graph construction modules are realized in Matlab, and the gaze graph classifier is implemented using Keras 2.3 on top of the TensorFlow 2.0 framework. We run 400 trials of the end-to-end pipeline for each of the three datasets and report the average time consumed by each of the three system components. The results are given in Table 3. The lion's share of computation latency is the construction of the gaze graph. Specifically, the latency increases with k. As discussed in Section 6.1.3, a larger k leads to more overhead in calculating the pairwise distances between each of the gaze points with its k-hop neighbors. By contrast, the inference latency only accounts for a small portion of the end-to-end latency. Overall, with a 30s sensing window and $k = 5$ in the graph construction, the end-to-end latency is 12.29ms and 8.89ms on the laptop and the desktop, respectively. For more precise measurement in the future, we will use the NVIDIA Jetson Nano [86] to profile GazeGraph on advanced IoT platforms.

Communication latency: as the Pupil Core does not support wireless transmission, we use the Magic Leap One AR headset [50] as the proxy to study the communication latency in data offloading. The Magic Leap One is embedded with the VOG-based eye tracker that captures the user's gaze at a 30Hz sampling rate. We consider a 30s sensing window, and transmit a sequence of 900 raw gaze points encapsulated in a single JSON object from the Magic Leap One (the client) to the desktop (the edge server) via an HTTP POST request. After receiving the data, the server sends an HTTP RESPONSE to the client. The two devices are connected by a single-hop WiFi network in the 5GHz frequency band. We run 100 trials of the data transmission and measure the round-trip time (RTT) for each of the trials. The average RTT is 85.01ms with a standard deviation of 6.2ms. Note that, as the Magic Leap One is known to be inefficient in wireless data transmission [87], we can expect a lower communication latency when the system is deployed on other mobile eye trackers.

End-to-end system latency: with a 30s sensing window and $k = 5$, the end-to-end system latency of GazeGraph is 8.98ms and 93.99ms for the stationary eye tracker (i.e., computation is performed on the desktop) and mobile eye tracker (i.e., computation is offloaded to the edge server), respectively. Overall, the results indicate that GazeGraph ensures high sensing accuracy while maintaining low system latency.

9 DISCUSSION

In this section, we discuss the limitations of this study and outline possible research directions.

Advanced DNN models for time series-based cognitive context sensing. In the current presentation, we consider the classical LSTM as the RNN model for time series-based cognitive context sensing. Despite the wide use of LSTM in time series-based recognition tasks [60–62], recent efforts in attention-based RNN models [88] have shown better recognition performance. In our future work, we will consider more advanced RNN models, such as the dual-stage attention-based RNN [89] and the sequence-to-sequence model [90], for time series-based cognitive context sensing. In addition, deep metric learning-based approaches [91, 92] are also good candidates to handle time series-based classification.

Different designs for the few-shot learning module. In the current design, our few-shot gaze graph learning module adopts the MAML [28] as the backbone algorithm to achieve few-shot learning. However, MAML requires higher-order derivatives that may lead to high computational cost when the dataset is large [93]. For the future work, one can use the implicit MAML algorithm [93] or the meta-transfer learning [94] to improve the learning efficiency. Moreover, the initial network model trained by MAML can be biased towards a subset of tasks that are generated during the meta-training phase, and may lack the ability to generalize to new domains. To alleviate this, one can use the task-agnostic meta-learning [30] algorithm to improve the model generalizability.

Graph Convolutional Networks-based design. Another interesting future direction is to leverage the Graph Convolutional Networks (GCN) [95] as the backbone network for the gaze graph classifier. GCN-based models are promising in handling data with graph structures and have shown state-of-the-art performance in many graph-based applications, such as text classification [43], skeleton-based action recognition [44], and molecular recognition [42].

10 CONCLUSION

In this paper we present GazeGraph, a generalized framework for gaze-based cognitive context sensing. GazeGraph advances the literature by a suite of new capabilities. We introduce a novel method that models human visual behavior as spatial-temporal graphs for better feature learning and high performance recognition. We also devise the few-shot graph learning module to enable fast system adaptation in new gaze-sensing scenarios with limited gaze instances needed. Our comprehensive evaluation shows that GazeGraph outperforms the existing solutions by 45% on average over three datasets when a large training dataset is available. Moreover, in 5-shot and 10-shot scenarios, GazeGraph outperforms the transfer learning-based approach by 30% and 19% on average, respectively, while reducing the system adaptation time by 80%.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and the shepherd for their insightful comments and guidance. We would also like to thank Namrata Srivastava and Kai Kunze for sharing their datasets. This work was supported in part by the Lord Foundation of North Carolina and by NSF awards CSR-1903136 and CNS-1908051.

REFERENCES

[1] A. Bulling and T. O. Zander, "Cognition-aware computing," *IEEE Pervasive Computing*, vol. 13, no. 3, pp. 80–83, 2014.

[2] A. Bulling, D. Roggen, and G. Troester, "What's in the eyes for context-awareness?" *IEEE Pervasive Computing*, vol. 10, no. 2, pp. 48–57, 2010.

[3] A. Bulling and D. Roggen, "Recognition of visual memory recall processes using eye movement analysis," in *Proceedings of ACM UbiComp*, 2011.

[4] K. Kassem, J. Salah, Y. Abdrabou, M. Morsy, R. El-Gendy, Y. Abdelrahman, and S. Abdennadher, "DiVA: Exploring the usage of pupil diameter to elicit valence and arousal," in *Proceedings of ACM MUM*, 2017.

[5] 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 ACM CHI*, 2016.

[6] A. Bulling, J. A. Ward, H. Gellersen, and G. Troster, "Eye movement analysis for activity recognition using electrooculography," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 741–753, 2010.

[7] O. Augereau, C. L. Sanches, K. Kise, and K. Kunze, "Wordometer systems for everyday life," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, p. 123, 2018.

[8] J. Karolus, P. W. Wozniak, L. L. Chuang, and A. Schmidt, "Robust gaze features for enabling language proficiency awareness," in *Proceedings of ACM CHI*, 2017.

[9] T. Kosch, M. Hassib, P. W. Wozniak, D. Buschek, and F. Alt, "Your eyes tell: Leveraging smooth pursuit for assessing cognitive workload," in *Proceedings of ACM CHI*, 2018.

[10] A. T. Duchowski, K. Krejtz, I. Krejtz, C. Biele, A. Niedzielska, P. Kiefer, M. Raubal, and I. Giannopoulos, "The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation," in *Proceedings of ACM CHI*, 2018.

[11] "Laptops that are integrated with eye tracking," https://gaming.tobii.com/products/laptops/.

[12] P. Norloff. Eye tracking technology is making new cars safer. [Online]. Available: https://eyegaze.com/eye-tracking-technology-is-making-new-cars-safer/

[13] Tobii. Eye tracking for driver safety. [Online]. Available: https://www.tobiipro.com/fields-of-use/psychology-and-neuroscience/customer-cases/audi-attitudes/

[14] N. Srivastava, J. Newn, and E. Velloso, "Combining low and mid-level gaze features for desktop activity recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, p. 189, 2018.

[15] K. Kunze, Y. Utsumi, Y. Shiga, K. Kise, and A. Bulling, "I know what you are reading: Recognition of document types using mobile eye tracking," in *Proceedings of ACM ISWC*, 2013.

[16] Y. Li, P. Xu, D. Lagun, and V. Navalpakkam, "Towards measuring and inferring user interest from gaze," in *Proceedings of ACM WWW*, 2017.

[17] D. Lagun, C.-H. Hsieh, D. Webster, and V. Navalpakkam, "Towards better measurement of attention and satisfaction in mobile search," in *Proceedings of ACM SIGIR*, 2014.

[18] M. Sugiyama and A. J. Storkey, "Mixture regression for covariate shift," in *Proceedings of NeurIPS*, 2007.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of IEEE CVPR*, 2015.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] C. Katsini, H. Opsis, Y. Abdrabou, G. E. Raptis, M. Khamis, and F. Alt, "The role of eye gaze in security and privacy applications: Survey and future HCI research directions," in *Proceedings of ACM CHI*, 2020.

[23] A. Liu, L. Xia, A. Duchowski, R. Bailey, K. Holmqvist, and E. Jain, "Differential privacy for eye-tracking data," in *Proceedings of ACM ETRA*, 2019.

[24] J. Steil, I. Hagestedt, M. X. Huang, and A. Bulling, "Privacy-aware eye tracking using differential privacy," in *Proceedings of ACM ETRA*, 2019.

[25] F.-F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[26] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *Proceedings of ICLR*, 2019.

[27] S. Thrun, "Lifelong learning algorithms," in *Learning to learn*. Springer, 1998, pp. 181–209.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of ICML*, 2017.

[29] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proceedings of ICLR*, 2016.

[30] M. A. Jamal and G.-J. Qi, "Task agnostic meta-learning for few-shot learning," in *Proceedings of IEEE CVPR*, 2019.

[31] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proceedings of AAAI*, 2018.

[32] S. A. Rokni, M. Nourollahi, and H. Ghasemzadeh, "Personalized human activity recognition using convolutional neural networks," in *Proceedings of AAAI*, 2018.

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of NeurIPS*, 2014.

[34] J. Steil and A. Bulling, "Discovery of everyday human activities from long-term visual behaviour using topic models," in *Proceedings of the ACM UbiComp*, 2015.

[35] P. Kiefer, I. Giannopoulos, and M. Raubal, "Using eye movements to recognize activities on cartographic maps," in *Proceedings of ACM SIGSPATIAL*, 2013.

[36] K. Kunze, K. Masai, M. Inami, Ö. Sacakli, M. Liwicki, A. Dengel, S. Ishimaru, and K. Kise, "Quantifying reading habits: counting how many words you read," in *Proceedings of ACM UbiComp*, 2015.

[37] J. Nie, Y. Hu, Y. Wang, S. Xia, and X. Jiang, "SPIDERS: Low-cost wireless glasses for continuous in-situ bio-signal acquisition and emotion recognition," in *Proceedings of IEEE/ACM IoTDI*, 2020.

[38] H. Wu, J. Feng, X. Tian, E. Sun, Y. Liu, B. Dong, F. Xu, and S. Zhong, "EMO: Real-time emotion recognition from single-eye images for resource-constrained eyewear devices," in *Proceedings of ACM MobiSys*, 2020.

[39] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Engineering Bulletin*, 2017.

[40] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2019.

[41] V. Gligorijević, M. Barot, and R. Bonneau, "deepNF: Deep network fusion for protein function prediction," *Bioinformatics*, vol. 34, no. 22, pp. 3873–3881, 2018.

[42] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of NeurIPS*, 2015.

[43] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of AAAI*, 2019.

[44] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of AAAI*, 2018.

[45] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proceedings of IEEE CVPR*, 2016.

[46] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *Proceedings of ICLR*, 2019.

[47] T. Gong, Y. Kim, J. Shin, and S.-J. Lee, "MetaSense: Few-shot adaptation to untrained conditions in deep mobile sensing," in *Proceedings of ACM SenSys*, 2019.

[48] "How to use eye tracking on your Alienware 17 R4," https://gaming.tobii.com/onboarding/alienware17-eye-tracking-how-to/.

[49] "Acer Predator 21x," https://gaming.tobii.com/product/acer-predator-21x/.

[50] "Magic Leap," https://www.magicleap.com/.

[51] "Tobii Pro VR integration," https://www.tobiipro.com/product-listing/vr-integration/.

[52] J. Sigut and S.-A. Sidha, "Iris center corneal reflection method for gaze tracking using visible light," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 2, pp. 411–419, 2010.

[53] M. Kassner, W. Patera, and A. Bulling, "Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction," in *Proceedings of ACM UbiComp*, 2014.

[54] M. Tonsen, J. Steil, Y. Sugano, and A. Bulling, "InvisibleEye: Mobile eye tracking using multiple low-resolution cameras and learning-based gaze estimation," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, p. 106, 2017.

[55] "Pupil Labs eye tracker," https://pupil-labs.com/.

[56] D. A. Robinson, "A method of measuring eye movement using a scleral search coil in a magnetic field," *IEEE Transactions on Biomedical Electronics*, vol. 10, no. 4, pp. 137–145, 1963.

[57] A. T. Duchowski, "A breadth-first survey of eye-tracking applications," *Behavior Research Methods, Instruments, & Computers*, vol. 34, no. 4, pp. 455–470, 2002.

[58] M. Khamis, F. Alt, and A. Bulling, "The past, present, and future of gaze-enabled handheld mobile devices: Survey and lessons learned," in *Proceedings of ACM MobiHCI*, 2018.

[59] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[60] T. Plötz and Y. Guan, "Deep learning for human activity recognition in mobile computing," *Computer*, vol. 51, no. 5, pp. 50–59, 2018.

[61] Y. Guan and T. Plötz, "Ensembles of deep LSTM learners for activity recognition using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–28, 2017.

[62] Z. Jia, X. Lyu, W. Zhang, R. P. Martin, R. E. Howard, and Y. Zhang, "Continuous low-power ammonia monitoring using long short-term memory neural networks," in *Proceedings of ACM SenSys*, 2018.

[63] "SMI eye tracking glasses," https://imotions.com/hardware/smi-eye-tracking-glasses/.

[64] "Tobii Pro X2 eye tracker," https://www.tobiipro.com/product-listing/tobii-pro-x2-30/.

[65] E. N. Ussery, J. E. Fulton, D. A. Galuska, P. T. Katzmarzyk, and S. A. Carlson, "Joint prevalence of sitting time and leisure-time physical activity among US adults, 2015-2016," *Jama*, vol. 320, no. 19, pp. 2036–2038, 2018.

[66] D. Lagun, C. Manzanares, S. M. Zola, E. A. Buffalo, and E. Agichtein, "Detecting cognitive impairment by eye movement analysis using automatic classification

algorithms," *Journal of Neuroscience Methods*, vol. 201, no. 1, pp. 196–203, 2011.
[67] A. García-Blanco, L. Salmerón, M. Perea, and L. Livianos, "Attentional biases toward emotional images in the different episodes of bipolar disorder: An eye-tracking study," *Psychiatry Research*, vol. 215, no. 3, pp. 628–633, 2014.
[68] S. Wang, M. Jiang, X. M. Duchesne, E. A. Laugeson, D. P. Kennedy, R. Adolphs, and Q. Zhao, "Atypical visual saliency in autism spectrum disorder quantified through model-based eye tracking," *Neuron*, vol. 88, no. 3, pp. 604–616, 2015.
[69] K. Harezlak and P. Kasprowski, "Application of eye tracking in medicine: A survey, research issues and challenges," *Computerized Medical Imaging and Graphics*, vol. 65, pp. 176–190, 2018.
[70] "HP Omnicept," https://www8.hp.com/us/en/vr/reverb-g2-vr-headset-omnicept-edition.html.
[71] R. S. Khan, G. Tien, M. S. Atkins, B. Zheng, O. N. Panton, and A. T. Meneghetti, "Analysis of eye gaze: Do novice surgeons look at the same location as expert surgeons during a laparoscopic operation?" *Surgical Endoscopy*, vol. 26, no. 12, pp. 3536–3540, 2012.
[72] A. Burova, J. Mäkelä, J. Hakulinen, T. Keskinen, H. Heinonen, S. Siltanen, and M. Turunen, "Utilizing VR and gaze tracking to develop AR solutions for industrial maintenance," in *Proceedings of ACM CHI*, 2020.
[73] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl, "State-of-the-art of visualization for eye tracking data," in *Proceedings of EuroVis*, 2014.
[74] P. Blignaut, "Visual span and other parameters for the generation of heatmaps," in *Proceedings of ACM ETRA*, 2010.
[75] D. Noton and L. Stark, "Scanpaths in eye movements during pattern perception," *Science*, vol. 171, no. 3968, pp. 308–311, 1971.
[76] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.
[77] D. B. West *et al.*, *Introduction to graph theory*. Prentice Hall Upper Saddle River, 2001, vol. 2.
[78] S. Eraslan, Y. Yesilada, and S. Harper, "Scanpath trend analysis on web pages: Clustering eye tracking scanpaths," *ACM Transactions on the Web*, vol. 10, no. 4, p. 20, 2016.
[79] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[80] "Super Mario Bros game," https://www.classicgames.me/super-mario-bros.html.
[81] "Agario game," https://agar.io/.
[82] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of ICLR*, 2015.
[83] P. Olsson, "Real-time and offline filters for eye tracking," 2007, Master Thesis, KTH Royal Institute of Technology.
[84] K. Rayner and M. Castelhano, "Eye movements," *Scholarpedia*, vol. 2, no. 10, p. 3649, 2007.
[85] R. Fallahzadeh and H. Ghasemzadeh, "Personalization without user interruption: Boosting activity recognition in new subjects using unlabeled data," in *Proceedings of ACM ICCPS*, 2017.
[86] "NVIDIA Jetson Nano," https://developer.nvidia.com/embedded/jetson-nano.
[87] M. Glushakov, Y. Zhang, Y. Han, T. J. Scargill, G. Lan, and M. Gorlatova, "Edge-based provisioning of holographic content for contextual and personalized augmented reality," in *Proceedings of IEEE SmartEdge*, 2020.
[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of NeurIPS*, 2017.
[89] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of IJCAI*, 2017.
[90] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of NeurIPS*, 2014.
[91] Z. Che, X. He, K. Xu, and Y. Liu, "DECADE: A deep metric learning model for multivariate time series," in *Proceedings of KDD Workshop on Mining and Learning from Time Series*, 2017.
[92] S. Li, D. Hong, and H. Wang, "Relation inference among sensor time series in smart buildings with metric learning," in *Proceedings of AAAI*, 2020.
[93] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *Proceedings of NeurIPS*, 2019.
[94] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of IEEE CVPR*, 2019.
[95] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2017.