Mouse2Vec: Learning Reusable Semantic Representations of Mouse Behaviour

Guanhua Zhang
guanhua.zhang@vis.uni-stuttgart.de
University of Stuttgart
Stuttgart, Germany

Zhiming Hu*
zhiming.hu@vis.uni-stuttgart.de
University of Stuttgart
Stuttgart, Germany

Mihai Bâce†
mihai.bace@kuleuven.be
KU Leuven
Leuven, Belgium

Andreas Bulling
andreas.bulling@vis.uni-stuttgart.de
University of Stuttgart
Stuttgart, Germany

ABSTRACT

The mouse is a pervasive input device used for a wide range of interactive applications. However, computational modelling of mouse behaviour typically requires time-consuming design and extraction of handcrafted features, or approaches that are application-specific. We instead propose Mouse2Vec – a novel self-supervised method designed to learn semantic representations of mouse behaviour that are reusable across users and applications. Mouse2Vec uses a Transformer-based encoder-decoder architecture, which is specifically geared for mouse data: During pretraining, the encoder learns an embedding of input mouse trajectories while the decoder reconstructs the input and simultaneously detects mouse click events. We show that the representations learned by our method can identify interpretable mouse behaviour clusters and retrieve similar mouse trajectories. We also demonstrate on three sample downstream tasks that the representations can be practically used to augment mouse data for training supervised methods and serve as an effective feature extractor.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Artificial intelligence**; • **Human-centered computing → Human computer interaction (HCI)**.

KEYWORDS

Mouse input, Representation learning, Self-supervised learning, Data augmentation, Behaviour retrieval, Transformer

*Corresponding author
†A significant part of this work was conducted while at the University of Stuttgart

ACM Reference Format:
Guanhua Zhang, Zhiming Hu, Mihai Bâce, and Andreas Bulling. 2024. Mouse2Vec: Learning Reusable Semantic Representations of Mouse Behaviour. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3613904.3642141

1 INTRODUCTION

In human-computer interaction (HCI), modelling users' interactive behaviour is crucial given that behaviour contains information about the users themselves [22, 48], the interfaces they interact with [69], and the tasks they perform [16]. Among the different input devices, the mouse is one of the most widely studied modality for behaviour modelling given that it is readily available in a large number of systems and pervasively used in daily life [27, 60]. Consequently, mouse behaviour modelling has been explored for various intelligent applications, such as user identification [12], interactive task recognition [15, 16, 36], or next activity prediction [19, 38, 74]. A key challenge in all of these applications is that they require computational representations that can capture the temporal, spatial, and semantic information contained in mouse trajectories. While some researchers modelled mouse behaviour from the perspective of optimal control [25, 34, 63] and information theory [26], we focus on data-driven methods. Handcrafted features and supervised training based on ground-truth annotated data have been commonly used to learn mouse representations [73]. However, designing and extracting meaningful features require expert domain knowledge and are laborious [13]. Additionally, data collection and annotation are cumbersome, costly, and time-consuming [11, 59].

More recently, self-supervised methods have gained increasing adoption in HCI, e.g. to learn latent embeddings of user interfaces [43, 65], visualisations [42], artwork [71] or speech [53, 59]. These methods learn from the data themselves, without requiring any extra annotations. Another advantage is that these representations, if learned properly, can capture the semantics of the input data and their internal relationships in the latent embedding space [43, 71]. As such, these representations have been shown to generalise well across datasets and tasks [14, 29]. Despite these advantages, no self-supervised approach has yet been proposed for learning representations of mouse behaviour.

To fill this gap, we introduce Mouse2Vec – the first method to learn reusable semantic representations of mouse behaviour in a self-supervised fashion. At the core of our method is a Transformer-based encoder-decoder, not only leveraging the time and frequency domain of the continuous cursor locations, but also encoding discrete mouse events (click vs. movements), which are key characteristics of mouse behaviour. Mouse2Vec is trained on two large-scale, publicly available mouse datasets: Buffalo [61] and EMAKI [73], covering both laboratory and out-of-lab settings. We further propose a multi-task training scheme: After randomly dropping slices of the time and frequency domain signals, our model is tasked to reconstruct the entire signals as well as to detect individual mouse events.

We first show that the representations learned by Mouse2Vec capture latent relationships between mouse behaviours, reflect human-interpretable semantics of these behaviours (the underlying interaction goals), and allow for retrieving mouse behaviours with similar characteristics. Second, we demonstrate the practical use of Mouse2Vec for data augmentation and as a generic feature extractor for three sample downstream tasks: user identification, interactive task recognition, and next activity prediction. We chose these tasks because they are essential for intelligent interactive systems and adaptive user interfaces to understand users, the interaction context, and intended future activities [12, 19, 31, 32]. Our evaluations demonstrate consistent improvements on all of these tasks when using Mouse2Vec representations. Mouse2Vec caters to both novice and experienced users. While novice users can directly apply our pretrained model on their data or task, experienced users with deep learning skills can fine-tune our model to achieve even better downstream task performance.

The specific contributions of our work are three-fold:

(1) We propose Mouse2Vec[1] – the first self-supervised method to learn semantic representations of mouse behaviour that are reusable across users and tasks. The representations jointly encode the time and frequency domain of cursor locations and click information.
(2) We show that the representations learned using Mouse2Vec capture the semantics and latent relationships of mouse behaviour, and are thus human-understandable.
(3) We demonstrate the benefit of using Mouse2Vec for data augmentation or as a feature extractor for three sample downstream tasks that are widely relevant for personalised and intelligent interactive systems.

2 RELATED WORK

We discuss related work on (1) self-supervised representation learning in HCI, (2) mouse behaviour representations, and (3) applications of mouse behaviour modelling.

2.1 Self-Supervised Representation Learning in HCI

The HCI community has recently introduced self-supervised learning methods to learn latent semantic embeddings of user interfaces [43, 65], paintings [71], visualisations [42], and speech [53, 59]. Self-supervised learning is a paradigm where models are trained solely using the input data as its own supervision, without human annotation. This way, the models can extract representations that not only capture the semantic and relationships of the input, but also are reusable in different tasks [72].

Screen2Vec [43] used an image encoder-decoder architecture to learn from UI layouts, and a pretrained Sentence-BERT [52] to learn from UI components and application descriptions. The final representation could display semantic similarities between UIs, which can be used in retrieving nearest neighbour UIs for UX designers to understand possible design solutions. Wang et al. [65] presented Screen2Words that learned embeddings of the essential information of a UI including individual components, their hierarchy structure and semantics leveraging a Transformer encoder-decoder architecture. Based on the embeddings, the authors further summarised the UI screens into natural language. Li et al. [42] proposed a contrastive learning-based method to learn representations of visualisations. The nearest neighbour query was applied on the learned representations to find visualisations that had similar visual and structural information. This provided a new way for visualisation retrieval, which can benefit large scale analyses of visualisations. Yilma et al. [71] proposed a method to learn latent embeddings of paintings using three existing representation learning methods: latent Dirichlet allocation, BERT and ResNet. They compared retrieval results of similar paintings using these embeddings, which provided insights into art recommendation. WESPER [53] is a Transformer-based method for learning representations of speech. It was trained by estimating the masked units of the input whisper speech. Based on the representations, the method further converted whisper to normal voice. Su et al. [59] proposed LipLearner, a contrastive learning-based method that extracted lipreading representations during silent speech interactions. Their representations could capture the semantic of each speech, i.e., speech content. The authors also presented that these representations could serve as effective features for a downstream task – speech command classification.

Despite the advantages and potential of self-supervised learning, no prior work has explored such approaches for mouse behaviour.

2.2 Mouse Behaviour Representations

As one of the most frequently used input devices, mouse plays a significant role in human-computer interaction. It is essential that mouse data is represented in a way that captures the characteristics of mouse behaviour, including the click events, temporal dynamics, spatial patterns, and underlying intents. After preprocessing such as normalisation [73], resampling [50, 67] and segmentation by sliding windows [36], mouse data is represented directly as original trajectories or using handcrafted features extracted from mouse traces.

Some works have used original mouse traces, including the coordinates of the mouse cursor [2, 9, 35], the time offset of each data sample [41], and mouse events [67]. These original data are usually further fed into classifiers, such as multilayer perceptrons (MLPs) [62], one-dimensional convolutional neural network (1DCNN) [9] and recurrent neural network (RNN) [41], to tackle with different tasks. Other works instead extracted a set of handcrafted features in the time domain, such as mouse movement location, angle, angle difference, velocity, acceleration, travel length, the straight distance from the origin of the trace, curvature and jerk [1, 18, 19, 23, 38, 54,

[1]Project webpage: https://perceptualui.org/publications/zhang24_chi/

56, 57]. Besides time domain features, Yildirim et al. [70] converted mouse behaviour into the frequency domain and calculated the mean and maximum of the power and frequency. They showed that these frequency domain features contributed to mouse behaviour modelling. The number of mouse clicks was also used as a feature [36]. While handcrafted features usually demonstrate better performance than using the original data, the process of extracting them is time-consuming and laborious, and requires prior knowledge of mouse behaviour characteristics. Our approach aims to address the above limitations by learning reusable mouse representations directly from the original input, offering a more efficient solution for different downstream tasks. Our representations also have the advantage of capturing semantic structures and latent relationships of the mouse data for better mouse behaviour understanding.

2.3 Applications of Mouse Behaviour Modelling

Mouse data have been used in a variety of applications including modelling users [18, 48, 61], understanding interaction contexts [36] and predicting future activities [16, 38, 74]. For example, Chuda et al. identified users during web browsing from handcrafted mouse features, such as click duration, moving velocity, and acceleration [12]. Mouse behaviour conveys information about users' cognitive states such as spatio-temporal visual attention [3] and cognitive load [54]. Identifying the current user enables personalised interfaces, e.g., rearranging layouts based on the user's interaction habit [69]. Identifying from mouse behaviour is particularly valuable in online environments where traditional passwords are vulnerable to attacks [61]. Mouse behaviour has also been used in understanding the interactive contexts. For example, Elbahi et al. recognised interactive tasks from mouse trajectories in an e-learning interface [15], while Koldijk et al. discriminated between 12 office tasks based on mouse handcrafted features [36]. Knowing which task a user is performing helps adaptive systems narrow down user needs to provide efficient assistance, as well as adjust the UIs for quicker interaction [17]. For example, when detecting the current task to be image editing, the system may suggest adding an image filter and thus make filter buttons more salient; under the task of text formatting, the system may recommend changing the font and hence increase the saliency of related interactive elements. Other researchers investigated next activity prediction, which is essential for anticipatory interactive systems. For instance, Kwok et al. predicted the next activity out of reviewing videos, self-reporting, annotating, confirming the annotation, and clicking outside the window in a crowdsourcing annotation and web search task [38]. Zhang et al. predicted the next formatting activity users intended to perform in a text editing scenario using original mouse records [74]. Proactive UIs predict and recommend users' potential next activities and can also automate tasks, which saves users' time and improves productivity [47]. In this work, we build upon these established tasks and demonstrate that our novel mouse representations can be effectively reused in different tasks that cover modelling users, interactive tasks and next activities.

3 LEARNING MOUSE BEHAVIOUR REPRESENTATIONS

Figure 1 provides an overview of the training pipeline of our method to learn representations of mouse behaviour. Mouse2Vec is trained via input reconstruction and mouse event detection, leveraging information in both the time and frequency domain. Reconstructing the input from randomly dropped or masked data has been adopted in recent self-supervised representation learning methods [29, 75]. We designed a particular mouse event detection training sub-task to encode click information, since clicks are a key characteristic of mouse behaviour [33, 36].

3.1 Mouse Data Preprocessing

Every data point in the raw mouse input is a four-tuple $(x, y, t, event)$, where x and y denote the current on-screen coordinates of the cursor, t is the timestamp, and *event* indicates whether the sample corresponds to a click (encoded as '1') or movement (encoded as '0'). We first normalised the raw x and y coordinates to the range of $[0, 1]$ to eliminate the impact of varying screen sizes, as suggested by Chong et al. [10]. Specifically, we divided the coordinates by the screen resolution if available; otherwise we used the minimum and maximum coordinate values and applied a MinMaxScaler[2]. Mouse data collection does not adhere to fixed sampling rates, i.e., data points are generated only when a mouse action happens, such as moving the cursor or clicking. To address this, we resampled the mouse data to 20 Hz based on t [50, 67] for faster computing (Appendix D). When there were no data within a sampling unit ($1\,s/20\,Hz = 0.05\,s$), we replicated the prior data point to ensure uniformity across time. Subsequently, we segmented mouse data into five-second windows, with a one-second stride [24, 32, 51].

Given that the frequency domain was shown to also provide rich information in modelling mouse behaviour [70], we encoded frequency domain information from each time window, encompassing both magnitude and phase [75]. To convert the temporal data into the frequency domain,we applied the Discrete Fourier Transform (DFT):

$$\mathcal{F}[k] = \sum_{n=0}^{N-1} T[n](\cos\frac{2kn\pi}{N} - \sin\frac{2kn\pi}{N} i), k = 0, 1, ..., N-1 \quad (1)$$

where $T = (x, y)$ is the time signal in each window; N denotes the length of T, which equals to 100 ($5\,s \times 20\,Hz$); k is the index of the frequency data; i represents the imaginary unit ($i^2 = -1$); and $\mathcal{F}$ is the frequency domain of T, comprising a sequence of complex numbers. We retained the former half of the sequence due to its symmetry following Zhang et al. [75]. For each complex number $z = a + bi$, we calculated its magnitude $M(z)$:

$$M(z) = \|z\| = \sqrt{a^2 + b^2} \quad (2)$$

and phase $P(z) \in (-\pi, \pi]$:

$$P(z) = \begin{cases} \arctan\frac{b}{a} & a > 0 \\ \arctan\frac{b}{a} + \text{Sign}(b) \times \pi & a < 0 \\ \text{Sign}(b) \times \frac{\pi}{2} & a = 0 \end{cases} \quad (3)$$

[2]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

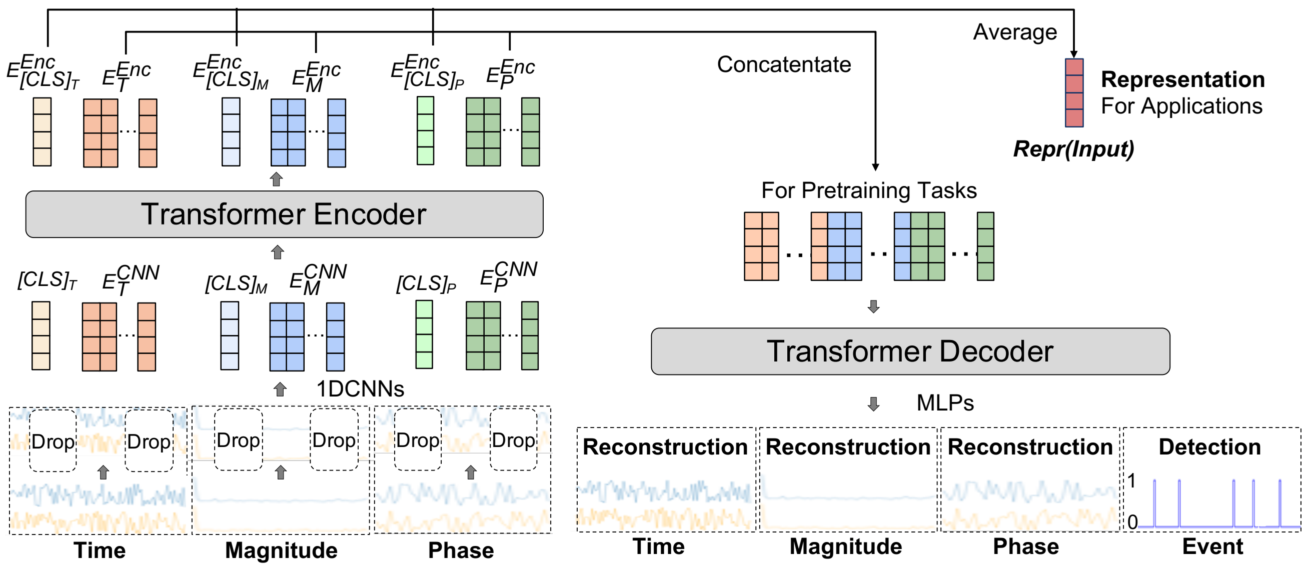


Figure 1: Pipeline of training Mouse2Vec, the proposed self-supervised mouse behaviour representation learning model. The input includes time domain T, frequency domain magnitude M and phase P, and binary $Event$ indicating clicks (1) or moves (0). Mouse2Vec is trained via reconstructing the T, M and P, and simultaneously detecting $Event$. The average of [CLS] token embeddings is the representation of the input mouse behaviour.

where $\text{Sign(b)} = \begin{cases} \frac{\|b\|}{b} & b \neq 0 \\ 0 & b = 0 \end{cases}$. After preprocessing, mouse trajectory in each window is in the form of $(T, M, P, Event)$, where T, M and P denote the original x or y sequence in the time domain, their magnitudes and phases in the frequency domain, while the binary vector $Event$ indicates a sequence of clicks or moves.

3.2 Mouse2Vec Model

The preprocessed mouse data is used as input to Mouse2Vec (see Figure 1). Mouse2Vec uses a Transformer-based encoder-decoder architecture and is trained in a self-supervised manner, relying solely on the input data, without requiring any annotations or labels. The training tasks are to reconstruct T, M and P after randomly dropping slices of the input, and detecting $Event$.

3.2.1 Dropping Input Slices. We first sliced T, M and P, and each slice had five data points. Then we randomly assigned a probability from a uniform distribution $U(0, 1)$ to each slice. Slices whose probabilities were smaller than a dropping ratio were discarded. We adopted a curriculum learning strategy [6], initially setting a low dropping ratio of 0.3 and gradually increasing it to 0.8. At each training epoch e, the dropping ratio was $max(0.3, min(0.8, \frac{e}{Epochs}))$, where $Epochs$ represents the total number of epochs. This strategy has proven to be able to enhance model robustness and convergence [75]. After dropping, we denote the remaining slices as T_{kept}, M_{kept}, and P_{kept}.

3.2.2 Architecture. Inspired by prior self-supervised representation learning models [29, 44], Mouse2Vec uses an encoder-decoder architecture based on a Transformer [64]. We chose Transformer because it can capture long-range temporal dependencies and has achieved state-of-the-art results in various time series-based applications [73, 75]. An encoder projects T_{kept}, M_{kept}, and P_{kept} into an embedding space while a decoder reconstructs the entire input (before dropping) T, M and P and simultaneously detects $Event$.

In the encoder, we first used three 1D ResNet-18 [30] based convolutional neural networks (1DCNNs) to learn local dependencies that transformed the input to E^{CNN}_T, E^{CNN}_M, and E^{CNN}_P, respectively. Subsequently, a [CLS] token was concatenated to the beginning of each E^{CNN}_i, $i \in [T, M, P]$. [CLS] tokens are commonly used to learn the representation of an entire sequence [14]. The concatenations of [CLS] tokens and E^{CNN}_i were then added with positional embeddings and three learnable domain-type embeddings, each corresponding to a domain $i \in [T, M, P]$ and having the same shape as the concatenations. The summed results were then projected by a six-layer Transformer encoder to the embeddings, consisting of the [CLS] embedding $E^{Enc}_{[CLS]_i}$, followed by the embedding of the input series E^{Enc}_i. $E^{Enc}_{[CLS]_i}$ were averaged to yield the final representation of the input mouse behaviour, denoted as $Repr(Input)$ [8, 52]. We set the dimension of the representation to 128. As such, only the encoder is needed when using our pretrained Mouse2Vec to generate mouse representations in the form of 128-dimensional embedding vectors.

The E^{Enc}_i were concatenated and fed into the decoder to perform the training task [75]. We applied a two-layer Transformer decoder, shallower than the encoder. This is because a shallower decoder can reduce training time while maintaining the quality of the learned representation [29]. The output of the decoder is then passed to MLPs, one for each training sub-task. Each MLP consisted of three linear layers. For event detection, we added a Softmax layer at the end to generate classification labels.

3.2.3 Multi-Task Training. We trained Mouse2Vec using two sub-tasks: reconstructing the entire input T, M and P based on T_{kept}, M_{kept}, and P_{kept}, and simultaneously detecting the click or move $Event$.

Sub-Task 1: Input Reconstruction. Reconstructing the entire input when the input is partially masked or dropped has become a dominant training paradigm in self-supervised representation learning [68, 75]. This task can preserve both global and local context and learn long-term dependencies [29]. We used the mean squared error between the reconstruction T', M' and P' and ground truth T, M and P as the reconstruction loss L_{Recon}.

Sub-Task 2: Mouse Event Detection. Given that clicks convey crucial information about mouse behaviour [33, 36], we introduced event detection to encode click information as a second sub-task. Since clicks are much sparser compared to mouse movements (see Section 3.3), i.e., the two classes in $Event$ are imbalanced, we employed a weighted cross entropy between the predicted $Event'$ and the ground truth $Event$ as the loss L_{Event}.

In summary, the training loss of Mouse2Vec is defined as:

$$L_{Mouse2Vec} = L_{Recon} + \beta L_{Event} + \beta' L_{MI} \tag{4}$$

where L_{MI} is the mutual information between representations originating from different inputs, which has recently become common practice in representation learning to encourage extracting more discriminative patterns [75, 76]. β and β' were set to 1 and 1e-4, respectively. The model was trained for 100 epochs with a batch size of 512 and optimised with the Adam optimiser[3]. We set the initial learning rate to 1e-4, and implemented a reduction scheme where the learning rate was reduced by 10% when the loss stopped decreasing for 20 consecutive epochs [20]. Appendix C presents the apparatus and time investment for implementing and training Mouse2Vec.

3.3 Training Datasets

We trained Mouse2Vec on two different datasets jointly: Buffalo [60] and EMAKI [73]. Buffalo is a dataset collected in a controlled laboratory environment while EMAKI is an out-of-lab dataset collected in a more natural interactive setting. Using both types of datasets provides a diverse range of mouse behaviours, which can help to learn representations that are robust and generalisable to different settings [73].

To the best of our knowledge, Buffalo is the largest publicly available laboratory dataset offering both mouse moves and clicks. It contains data from 148 participants who performed two task trials as a session, and repeated the session three times. The first task was to transcribe a piece of text. The second task was composed of two sub-tasks: writing opinions on two survey questions and describing a picture; completing routine work such as writing an email, adding attachments, and free Internet surfing. After preprocessing, Buffalo had 140 K mouse trajectories in total. 1% of the actions were clicks.

EMAKI is a publicly available, out-of-lab dataset that contains mouse movement and click data recorded during different interactive tasks. The data were collected from 39 participants who joined an online user study via their own computers to conduct three interactive tasks: writing and editing an article, drawing and editing images, and completing questionnaires about demographics and personality traits. After preprocessing, EMAKI provided 38 K mouse trajectories in total. 3% of the actions were clicks, again showing the sparsity of clicks.

[3]https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

4 ANALYSING THE REPRESENTATIONS LEARNED BY MOUSE2VEC

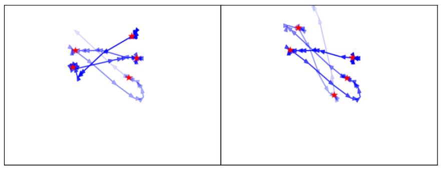
We first conducted qualitative evaluations to analyse if representations learned by our method capture meaningful relationships of mouse behaviour, i.e., similarities and differences between mouse behaviours in terms of their spatial, temporal, and interaction goals. To this end, we first clustered the mouse behaviours based on their representations. We then examined the nearest neighbours of a mouse trajectory, i.e., looking for trajectories that were the most similar. We further confirmed the utility of our representation via practical downstream tasks, which we will introduce in Section 5.

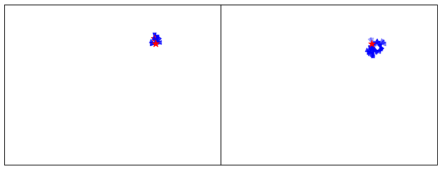
We evaluated the semantics of the representations on the EOTT dataset [49], which is publicly available and offers diverse mouse behaviours generated during various tasks by 51 participants. The interactive tasks included two artificial and two naturalistic tasks: One artificial task is a standard Fitts' Law study, where participants had to move the cursor to various target dots, arranged in a circle, following a pre-defined trace. The other artificial task is target selection, where the target moved in a 3×3 grid from the top left corner all the way to the bottom right corner of the screen. Participants had to follow and click on the target. The naturalistic tasks included web search and creative writing, in order to replicate a scenario of reading, searching for information, and typing the solution. After preprocessing, EOTT comprised 11 K mouse trajectories.

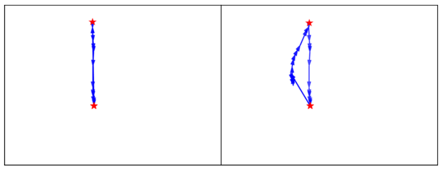
4.1 Mouse Behaviour Clusters

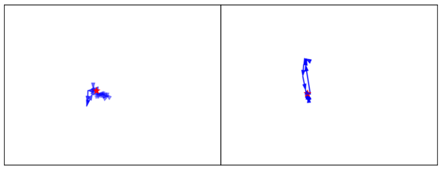
We used hierarchical density-based spatial clustering (HDBSCAN) [7] to identify clusters in the Mouse2Vec latent embedding space. HDBSCAN does not require pre-defining the number of clusters, can handle clusters with varying densities and noisy data points, and has been successfully used in representation-based clustering [71]. We used cosine similarity as the distance metric, which is a common practice in representation learning [28, 42]. On the EOTT dataset, HDBSCAN identified a total of 105 clusters. Here, we examined the nine clusters with the highest number of mouse samples, as they provided insights into the most prevalent behaviour patterns. For each cluster, we visualised the two mouse trajectories closest to its centroid and analysed their underlying interaction goals because these can be considered representative of the whole cluster [40]. As presented in Figure 2, blue lines adorned with arrows show the mouse moves, which gradually fade over time, while clicks are indicated with red stars.

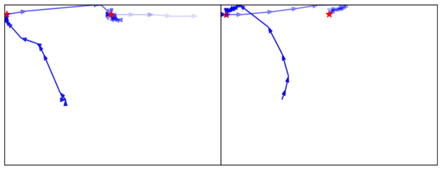
We found that most of the mouse behaviour in the largest cluster and the third largest clusters came from the standard Fitts' law task. In this task, participants navigated the mouse between fixed target points and clicked on them, and hence behaved similarly. The representative mouse trajectories in Figure 2a and Figure 2c reflected such navigation and clicks. Two clusters (Figure 2e, 2f) were formed by trajectories that all originated from the other artificial task, target selection where participants followed the dot

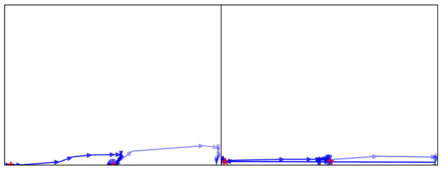

(a) Moving between targets and clicking on targets

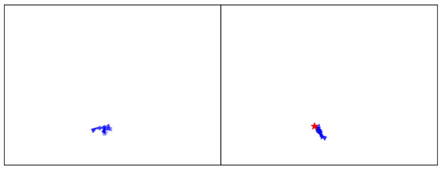

(b) Adjusting the cursor to click on a target at the top-right area

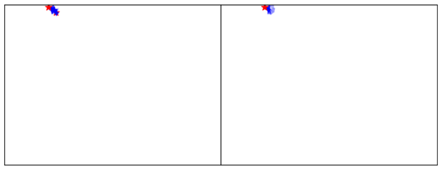

(c) Moving up to click and then vertically down to click


(d) Adjusting the cursor to click on a target at the middle area


(e) Moving to click at the top-left corner and then horizontally to the right and click again


(f) Clicking at bottom-left, moving horizontally to the right, clicking in the middle


(g) Adjusting to interact with a target at the bottom-middle area


(h) Adjusting to click on a target at the top-left area

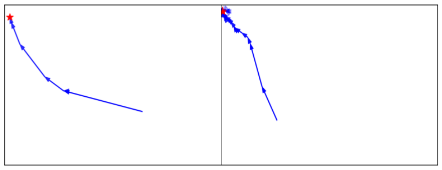
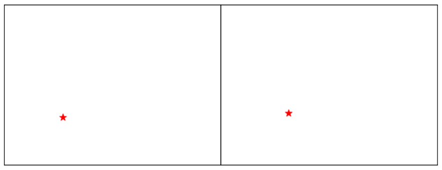
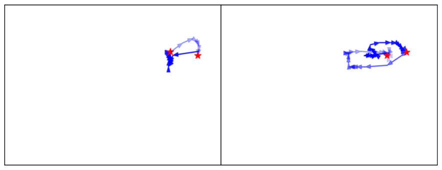

(i) Moving diagonally to click at the top-left corner

Figure 2: Two representative mouse trajectories for each of the nine largest clusters and their semantics. The two trajectories are the closest to the centroid of each cluster. The clusters are identified using HDBSCAN based on the cosine similarity between Mouse2Vec representations. The trajectories are shown in fading blue lines adorned with arrows. Each red star indicates a click.


(a) Single click


(b) Adjusting the cursor to click on a target at the top-right area

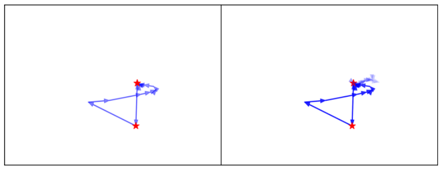

(c) Moving up to click and then vertically down to click

Figure 3: Two representative mouse trajectories for each of the three smallest clusters and their semantics.

moving from left to right. The plotted samples also showed the pattern of moving to the right and clicking on the targets. Additionally, four (Figure 2b, 2d, 2g, 2h) of nine clusters comprised short mouse movements within a small area, often accompanied by clicks. These patterns suggest a goal of adjusting the cursor to click on a target. This finding is in line with prior research indicating that users exhibit similar behaviour when interacting with a precise target [73]. The last cluster showed long and quick moves to click at the top-left corner (Figure 2i). Therefore, we can observe that similar mouse patterns, such as adjusting the mouse to click, that occur in distinct screen regions were in different clusters. This is a reasonable outcome since these behaviours may carry various meanings. For instance, mouse movements followed by clicks in the screen's top-left or top-right corner could indicate an intention to close or minimise a window, while similar movements and clicks in the top-middle area might suggest switching between tabs or selecting items from menus or tools. We also visualised the smallest three clusters in Figure 3, which are also interesting given that they reflect rare behaviours. The representative trajectories of the three clusters either consist of only one mouse action (Figure 3a) or display redundant movements (Figure 3b, 3c).

4.2 Retrieving Similar Mouse Behaviour

Clustering allows us to study the relationship between different mouse behaviours at the level of trajectory *groups*. In a second step we analysed the *instance* level, i.e. the similarity of individual mouse trajectories, using a instance retrieval approach [42, 43]. For a mouse trajectory query, we retrieved its three most similar behaviours, i.e., top-3 nearest neighbours [43]. We again used the cosine similarity between representations to measure the similarity between their corresponding original mouse trajectories. To understand if and how well our representations capture the similarity between mouse behaviour, we compared them to other representations, visualising these baselines' retrieval results.

4.2.1 Baselines. The baselines include ablations of Mouse2Vec and classical mouse representations, i.e., original data and handcrafted features as introduced in Section 2.2:

- Original data. The preprocessed mouse data.
- Handcrafted features. We incorporated a wide range of commonly used features from prior mouse behaviour modelling works resulting in 100 features, including clicks, cursor locations, trajectory angles and velocities, and frequency domain powers. The complete feature set is described in Appendix A.
- Ablations of Mouse2Vec. Given that our approach leverages time domain and frequency domain (magnitude and phase) of the on-screen locations and mouse events, i.e., the four boxes after Transformer decoder in Figure 1, we removed each of them as an ablation. As such, *Mouse2Vec w/o Event* removed event detection and thus only learns from reconstruction of the three other signals; *Mouse2Vec w/o Time*, *Mouse2Vec w/o Magnitude* and *Mouse2Vec w/o Phase* were pretrained to reconstruct the other two remaining signals and meanwhile detect mouse events.

Following prior works, we calculated both Euclidean distance Ori_{euc} [46] and cosine similarity Ori_{cos} [45] to measure the similarity between the original data, the Euclidean distance Handcrafted_{euc} to measure the similarity between handcrafted features [39], and the cosine similarity between representations for the ablations.

4.2.2 Results. Figure 4 shows an example query of mouse behaviour and the top-3 nearest neighbours retrieved by the proposed Mouse2Vec and baseline methods. The query trajectory first clicks in the middle of the screen, then moves to the top-left corner and down from the left to click at the bottom. The three trajectories retrieved based on Mouse2Vec representations have similar shape and include a click in the middle. Two of them also contain a second click at the similar, bottom location. Based on the original data, using Euclidean distance or cosine similarity retrieved the same trajectories with different ranks. However, all move down to the right of the move-up, different from the query. In addition, only one captured two clicks. None of the behaviours retrieved from handcrafted features have clicks or similar shapes to the query. Among the ablations, *Mouse2Vec w/o Phase* captured the moving trend and at least one click, while *w/o Magnitude* found similar move directions, but both were still worse than Mouse2Vec. *w/o Time* and *w/o Event* resulted in trajectories with significantly different outlines compared to the query. These observations indicate that the time, frequency domain and mouse events are all essential for the representations to capture latent similarities of mouse behaviour. Moreover, time domain and mouse events are more important than frequency domain. Figure 5 presents more examples using Mouse2Vec. Judging visually, result trajectories generally have similar outlines, moving directions, on-screen locations and clicks compared to the query, indicating that Mouse2Vec embeddings learned meaningful mouse patterns. We further evaluate the similarity from a quantitative perspective in Section 5.1.

In summary, the two qualitative analyses demonstrated that the representations learned by Mouse2Vec capture semantics of mouse

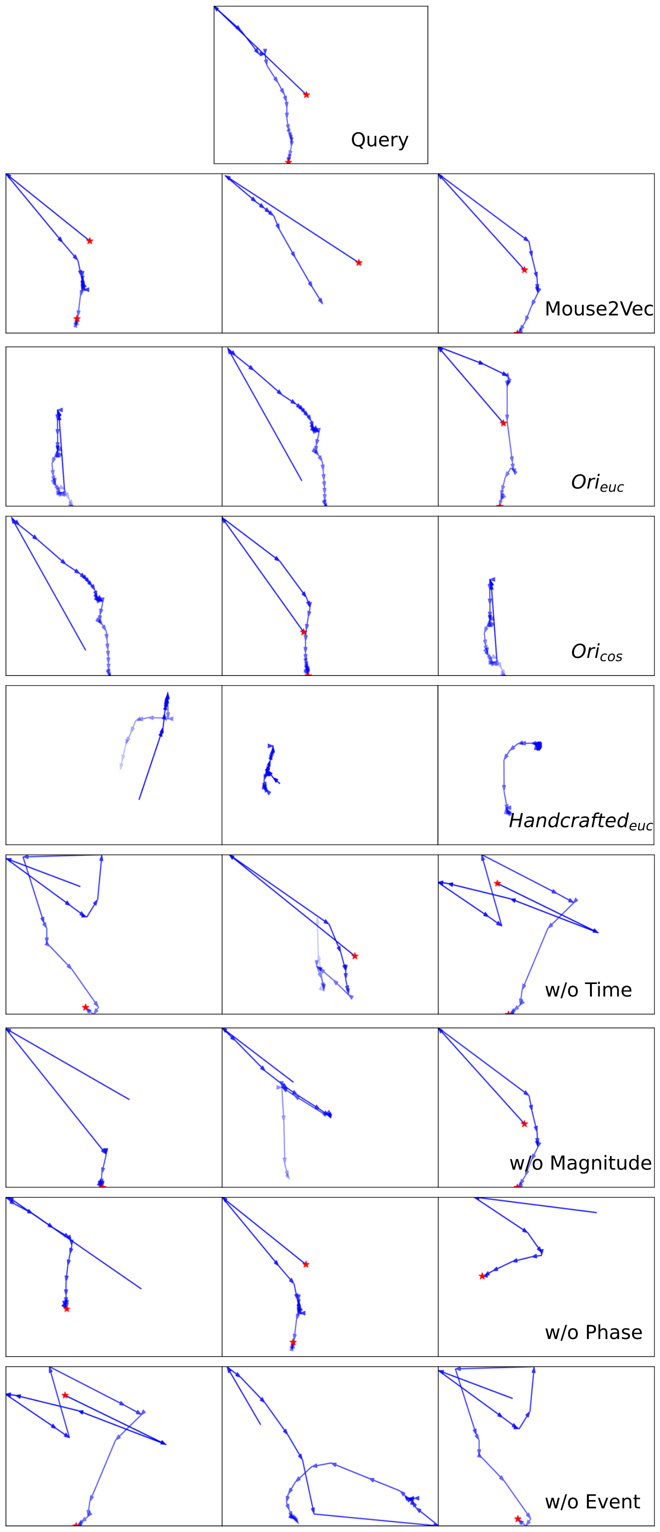


Figure 4: An example mouse trajectory as the query and the top-3 nearest neighbours retrieved using Mouse2Vec representations and baselines.

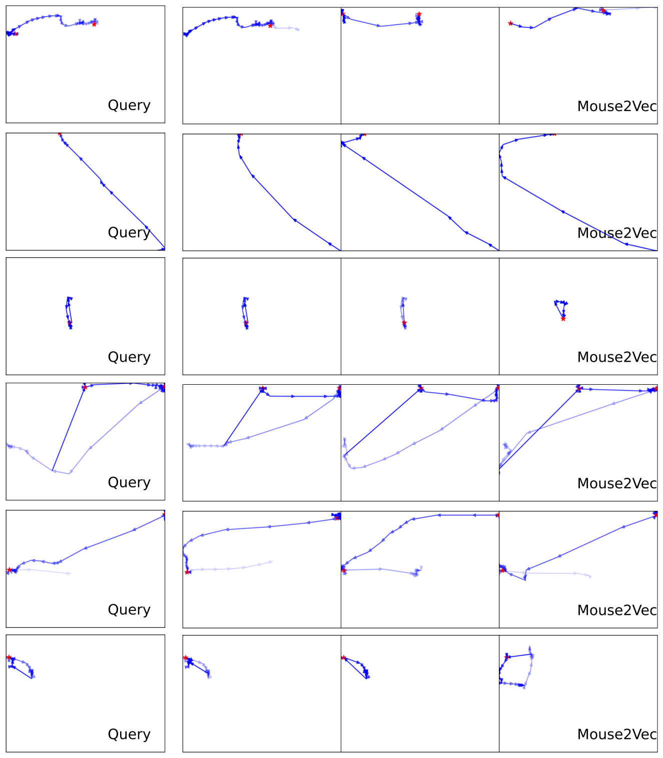


Figure 5: Six more example queries and their top-3 nearest neighbours retrieved using Mouse2Vec representations.

behaviour and can be used to retrieve mouse trajectories that have similar characteristics. Moreover, the evaluations were conducted on a dataset different from the pretraining ones, confirming that the representations are reusable.

5 EVALUATION FOR DIFFERENT DOWNSTREAM TASKS

We then conducted quantitative evaluations of Mouse2Vec to evaluate its practical usefulness for different downstream tasks. We used the representations in two ways: To augment data with limited labels using the retrieved nearest neighbours and as a generic mouse behaviour feature extractor. We evaluated these approaches for three sample downstream tasks: interactive task recognition, next activity prediction and user identification. These tasks are important for developing personalised intelligent interactive systems that can analyse user needs and actively anticipate and recommend potential activities [17, 19, 32].

For interactive task recognition, we used the EOTT dataset that included four tasks as introduced in Section 4.1. To enable next activity prediction, we used another public dataset ACTIVITY [74]. This dataset was collected from 16 participants performing text formatting tasks in a controlled laboratory setting. In each trial, participants applied a sequence of formatting activities to given "Lorem Ipsum" text snippets. There were seven candidate activities allowed by the text editor: bold, italic, underline, font size, font family, alignment, and indentation. After preprocessing, ACTIVITY had 7 K mouse trajectories, where 3% of mouse events were clicks.

We used both datasets for user identification because they both provide user IDs (51 users in EOTT and 16 users in ACTIVITY).

For interactive task recognition and next activity prediction, we performed a five-fold *user-independent* cross-validation to assess the generalisability of the method across users. Hence, we randomly split these participants into five sets. In each fold, we trained the classifier using data from four sets of participants and tested it on the remaining one. We repeated this procedure five times and averaged the accuracy across all folds as the final performance metric. For the user identification task, we performed five-fold *user-dependent* cross-validation because both the training and test sets need to have data from every user. We first randomly split each participant's data into five sets, in each fold we combined the four sets from all the participants as the training set, and used the remaining as the test set. We also repeated this five times and used the average accuracy as the performance metric.

We used MLP classifiers for the downstream tasks given that they are often used in mouse behaviour modelling [2, 57, 62]. The MLP we used had three linear layers, each with 64 hidden units. The first two linear layers were followed by ReLU activation functions, while the last layer was followed by a Softmax function to map the probability to class labels. The learning rate was initially set to 1e-3 and the classifier was trained for 50 epochs. We also used the Adam optimiser, and reduced the learning rate by 10% when the loss stopped decreasing for 20 consecutive epochs. Cross entropy between the prediction and ground-truth labels was used as the loss function.

5.1 Data Augmentation Based on Similar Behaviour

In HCI, we frequently face the challenge of having limited labelled data due to the high cost of data collection [11, 59]. The retrieved nearest neighbours to a mouse trajectory can be used to augment data when building data-driven models for practical tasks. To simulate a scenario of data scarcity, we randomly retained only 10% of the training data. For each training sample, we performed data augmentation using its nine nearest neighbours, which restored the training set to its original size. We queried for similar behaviours on the pretraining datasets (Buffalo and EMAKI) because they contain a variety of mouse behaviours. Specifically, we performed the following steps for each original mouse data sample in the 10% training set [66]:

(1) Use the pretrained Mouse2Vec to generate representations of the query sample and the candidate samples from the pretraining datasets;
(2) Calculate the cosine similarities between the query's representation and candidates' representations;
(3) Find the nine nearest neighbours from the candidates;
(4) Add these neighbours to the training set and assign them the same label as the query.

We then fed the augmented training data into the MLP classifier.

Table 1 shows the accuracies achieved with and without data augmentation for all three tasks. As can be seen from the table, data augmentation consistently improved the accuracy across tasks and datasets. For example, the accuracy of recognising four interactive tasks on EOTT increased by 6.79%, and the accuracy of identifying

Augmentation		ACTIVITY Dataset		EOTT Dataset	
		Next Activity Prediction	User Identification	Interactive Task Recognition	User Identification
w/o		44.27±2.36	10.22±0.90	65.81±3.81	4.37±0.40
Ori_{euc}		45.70±2.35	13.11±0.68	70.86±4.27	4.50±0.59
Ori_{cos}		45.01±0.93	13.42±0.71	65.29±2.58	4.28±0.36
Handcrafted		43.87±0.67	9.04±1.16	64.82±6.78	3.62±0.61
Mouse2Vec	w/o Time	44.71±0.70	10.29±0.84	60.94±3.87	4.71±0.21
	w/o Magnitude	<u>46.90±1.42</u>	13.96±0.83	<u>71.65±4.46</u>	4.86±0.41
	w/o Phase	46.52±1.43	<u>14.05±1.02</u>	65.29±2.58	**5.13±0.57*****
	w/o Event	45.45±1.38	12.63±1.17	70.87±4.23	4.78±0.51
	Full	**47.09±2.03***	**14.34±0.67***	**72.60±4.03***	<u>5.02±0.38</u>

Table 1: Accuracies (Mean±Std, in percentage) achieved in three tasks evaluated on two datasets, by using only 10% of the original mouse training data (w/o), and augmented with the nine nearest neighbours retrieved based on different representations. Best results are shown in bold and the second-best results are <u>underlined</u>. The three tasks are interactive task recognition (evaluated on EOTT), next activity prediction (evaluated on ACTIVITY) and user identification (both datasets). Stars mark the significance levels of difference between accuracies ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

16 users on the ACTIVITY dataset increased by 4.12%. We further performed a Wilcoxon signed-rank test and confirmed that the differences between the results were statistically significant ($p < .05$). Mouse2Vec and its ablated versions consistently achieved the best two results on the different downstream tasks and datasets. The full version of Mouse2Vec performed the best in most cases, showing that time and frequency domain as well as mouse events are all useful. Among the ablations, *w/o Phase* and *w/o Magnitude* outperformed *w/o Time* and *w/o Event*, illustrating that the time and event information is more important than magnitude and phase. We can also see that augmenting data via handcrafted features obtained worse accuracy than using the original 10% data. This is likely because augmenting data with dissimilar trajectories introduces data and label noise, while Section 4.2 also visually presents that handcrafted features are not ideal to retrieve similar mouse data.

5.2 Mouse2Vec as a Reusable Feature Extractor

Pretrained models can be reused on new datasets to extract features. For example, the VGG model [58] was first trained on natural images but later directly used or fine-tuned to extract features also from other types of images. Inspired by this, we evaluated if our pretrained Mouse2Vec could be used to extract features for different tasks. We also evaluated the two ways of extracting features: 1) Directly applying the pretrained Mouse2Vec, i.e., freezing the encoder of Mouse2Vec and only updating the classifier; and 2) fine-tuning the pretrained Mouse2Vec for each downstream task, i.e., starting from the pretrained encoder parameters and updating both the Mouse2Vec encoder and the classifier. The former case allows (novice) users to quickly and easily apply our model for their dataset or task without changing Mouse2Vec; whereas the latter targets at users who have experience in training deep learning models.

We compared Mouse2Vec with the baselines introduced in Section 4.2.1. While in data augmentation, the input to the classifier was always (augmented) original data, here the input was different: For representations learned by Mouse2Vec and its ablations, the input of the classifier was the embedding vector generated by the encoder. For handcrafted features, the input to the classifier was the feature vector. For original mouse data, the input was the preprocessed mouse sequence. We used all data from each dataset.

The results from these experiments are summarised in Table 2. Directly using the frozen Mouse2Vec already outperforms other representations on all tasks and both datasets. Mouse2Vec achieved an accuracy of 75.84% when recognising four interactive tasks and 62.69% when predicting the next activity (out of seven) with an improvement of 4.90% compared to using handcrafted features. For user identification on ACTIVITY (16 individuals), Mouse2Vec obtained an accuracy of 19.59%, and 9.43% when identifying 51 individuals on EOTT dataset. In most cases, handcrafted features obtained better results than the original mouse data, indicating the effectiveness of these features used in prior works. Among the ablations of Mouse2Vec, *w/o Time* and *w/o Event* brought the largest performance drop. This shows that encoding the time domain and mouse events into the representation is essential. We further performed a Wilcoxon signed-rank test between the results achieved by Mouse2Vec (both freezing and fine-tuning) and the best results obtained by the two classical representations. The test confirmed that Mouse2Vec significantly outperformed original mouse data and handcrafted features. These results demonstrate that the pretrained Mouse2Vec can be transferred to other datasets to extract mouse behaviour features that are useful for three different tasks.

Given that the full version of Mouse2Vec outperformed the other representations, we evaluated fine-tuning using the full version and found it further boosted the performance on all the downstream tasks. Fine-tuning increased the accuracy by 18.03% when predicting the next activity, 15.76% and 11.43% when identifying users from ACTIVITY and EOTT dataset, and 7.78% when recognising interactive tasks, from the best-performing classical representation. Compared to the frozen Mouse2Vec encoder that were

Representation		ACTIVITY Dataset		EOTT Dataset	
		Next Activity Prediction	User Identification	Interactive Task Recognition	User Identification
Original		47.45±0.92	15.58±4.85	72.97±3.30	5.86±0.15
Handcrafted		57.79±3.17	16.43±1.54	73.55±2.89	5.69±0.18
Mouse2Vec	w/o Time	53.59±2.14	15.11±0.81	73.25±4.21	6.34±0.33
	w/o Magnitude	59.02±2.58	16.03±0.62	73.35±3.53	7.59±0.35
	w/o Phase	59.76±1.81	17.84±1.58	74.16±3.24	8.34±0.44
	w/o Event	57.87±1.82	15.66±0.99	73.28±2.94	6.40±0.41
	Full	<u>62.69±3.57</u>***	<u>19.59±0.60</u>**	<u>75.84±3.39</u>*	<u>9.43±0.42</u>***
	Full (FineTune)	**75.82±3.88*****	**32.19±0.83****	**81.33±3.05****	**17.29±1.10*****

Table 2: Accuracies (Mean±Std, in percentage) on interactive task recognition (EOTT dataset), next activity prediction (ACTIVITY dataset) and user identification (both datasets). We compare results achieved using representations obtained by Mouse2Vec with using original data, handcrafted features, or representations learned using the ablation of Mouse2Vec. The bottom row presents the results when fine-tuning our pretrained Mouse2Vec. Best results are shown in bold, and the second-best results are <u>underlined</u>. Stars mark the significance levels of difference between Mouse2Vec and the best classical representation ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

trained according to the distribution of the pretraining datasets, fine-tuning can adapt the model towards the domain of the downstream datasets. Additionally, fine-tuning allows updating more parameters of the model, i.e., the model has a larger capacity than directly deploying the frozen Mouse2Vec.

We also compared Mouse2Vec against the approaches that were specifically geared to each downstream task from prior works (see Appendix F). Results showed that directly using the pretrained, frozen Mouse2Vec outperformed these methods in most cases, while fine-tuning Mouse2vec always led to the best results. Given that Mouse2Vec was trained on two datasets, we also evaluated its performance when trained only on one dataset (See Appendix B). Results show that training on both datasets outperformed training on only one dataset, confirming the effectiveness of our training strategy.

6 DISCUSSION

6.1 On Performance

In this work, we proposed Mouse2Vec – the first self-supervised method to learn reusable semantic representations of mouse behaviour. We evaluated the learned representations on three sample downstream tasks with practical value for the development of intelligent user interfaces. As shown in Table 2, the representations learned using Mouse2Vec perform significantly better than using original mouse behaviour data (15.24% improvement) and handcrafted features that are widely used in mouse behaviour modelling (4.90% improvement). These improvements are consistent across these different tasks as well as the two datasets investigated in this work (ACTIVITY and EOTT). These results are highly promising, in particular in light of other benefits that our self-supervised methods offers compared to the widely used approach of manual feature engineering. It is often unknown which features will be useful for a specific task beforehand, and thus requires either domain expertise to pick or invent features, or implementing many candidate features, e.g., over one hundred features [1], which is time-consuming and tedious. As such, self-supervised learning also significantly lowers the barrier for interactive behaviour modelling in HCI.

Beyond these advantages, Mouse2Vec is even more beneficial for experienced users who know how to train or fine-tune deep learning models. Fine-tuning updates parameters of both the encoder and classifier and hence, offers improved adaptability to various settings, tasks, or users. As we have shown in Table 2, fine-tuning Mouse2Vec results in even larger performance improvements for downstream tasks, reaching up to 28.37% improvement compared to using the original data and 18.03% compared to handcrafted features.

As shown in Table 1, the representations learned using Mouse2Vec is also effective for augmenting data required to train supervised methods. We also showed that Mouse2Vec achieved comparable performance to baseline representations when using only 50% or even 20% of annotated training data (Appendix E). It is a common problem in many computational behaviour modelling tasks that collecting labelled training data is costly and thus undesirable. When used for data augmentation, Mouse2Vec can achieve significant performance improvements of up to 6.79% when trained on only 10% of real data and the remaining 90% generated via nearest neighbour retrieval. Despite these improvements, the performance remains slightly lower than when trained on the full amount of original data (cf. row "Original" in Table 2). This is likely because the augmented data were gathered from the pretraining datasets and assigned labels according to query samples, which introduced data and label noise. In contrast, training with the full amount was done using data from the same dataset and genuine labels.

We also showed that leveraging two datasets to train Mouse2Vec obtained better representations than using only one dataset. This suggests that the performance of self-supervised (mouse) behaviour modelling can be further improved, if additional datasets can be released in the future.

6.2 On the Method

In this work, we described one possible implementation of Mouse2Vec that uses a Transformer-based encoder-decoder architecture. This choice was motivated by the recent success of Transformer-based models across a wide range of tasks and fields in the computer and engineering sciences, including HCI [52, 65, 73]. While the encoder-decoder architecture has been generally used in many works, we introduced two design decisions that make the method geared to mouse behaviour representation learning. The key characteristic and contribution of Mouse2Vec is that it jointly encodes the time and frequency domain (magnitude and phase) of continuous on-screen cursor locations, as well as discrete mouse events (clicks vs. moves). As can be seen from the ablation experiments, these design decisions have proven to be crucial to achieve the aforementioned performance: Removing any of time, magnitude, phase or event encoding resulted in a performance drop compared to the full model in most cases in data augmentation and feature extraction. Specifically, *Mouse2Vec w/o Event* and *w/o Time* performed the worst, indicating that mouse events and the temporal sequence of on-screen locations contain important information and are most essential for mouse behaviour modelling. This finding echoes prior works that have found mouse clicks [33, 70] and temporal information to be important in mouse data-driven tasks [54, 69]. Our evaluations also provided qualitative evidence to support this finding. For example, by visualising the similar mouse behaviours, we found that removing the frequency domain information still roughly capture the similarity in terms of location and shape. However, removing the time or event information resulted in a loss of this information (Figure 4).

With continuing advances in machine learning, specifically methods suited for temporal sequence modelling tasks, it will be interesting in future work to explore other Mouse2Vec implementations that either use entirely new underlying architectures or other ways to encode spatio-temporal mouse behaviour data.

6.3 On Representation Learning of Mouse Behaviour

In this work, we explore a method that may represent a paradigm shift in computational user behaviour modelling – specifically mouse behaviour as studied here, but also beyond. In the past, computational models of behaviour were developed for each individual interactive task or application, which limited their generalisability and reusability. Self-supervised representation learning, in contrast, has the potential to establish an entirely new paradigm in which rich representations of user behaviour can be learned without manual annotations. These representations can also be shared, built upon, and improved further by training on additional datasets – thus benefiting a wide range of tasks and applications at the same time. As we have shown through a cluster analysis of the representations (see Section 4.1) as well as a nearest neighbour retrieval experiment (see Section 4.2), the similarity of latent embeddings reflects the similarity of the original mouse behaviours. This suggests that the learned representations are able to capture underlying structure and semantics of mouse behaviour. Mouse2Vec can identify mouse behaviours for similar interaction goals, such as adjusting the cursor to click on a precise target, or move the cursor quickly across the screen. By doing so, our method identifies trajectories that are generally more similar to the query than several baselines. Most importantly, we demonstrate that these clusters are human-understandable and encode semantics (see Figure 2 and Figure 3). This is no small feat in light of the well-known limitations with respect to interpretability and the black-box nature of many machine learning methods. As such, the representations learned by Mouse2Vec can not only be used to improve performance or enable new applications, but also have potential as a tool to analyse, better understand, and characterise user behaviour in a data-driven fashion. This understanding can then be used to improve future interactive systems.

6.4 Limitations and Future Work

The aforementioned properties make our method promising for a number of potential future applications and use cases: For example, the clusters could be used to automatically annotate trajectories in a long mouse data recording, which saves time compared to human annotation. Based on the sizes of clusters, UX designers could analyse users' common or rare mouse behaviours to improve the usability of an interface. This could be used for UI optimisation, e.g., moving important information to the most frequent areas or by optimising layouts to minimise mouse travel time. The system could also retrieve other users with similar mouse behaviours and who have successfully completed a task, and then provide guidance to users who struggle or get stuck.

While the mouse is widely used and one of the most important input devices, one limitation of our work is that we have not studied other devices, such as the keyboard. Given that mouse and keyboard are usually used together in daily interaction scenarios, it will be interesting to see how our method could be extended to such multi-modal representation learning – a task that is now attracting increasing attention in HCI research [4, 65]. Another modality that we have not explored in this work, given that this may raise privacy concerns, is the user interface itself. However, future research could also learn multi-modal UI-mouse representations, e.g., by integrating models like Screen2Vec [43]. This would allow us to analyse the interplay between interface and mouse behaviour, and thus further improve mouse behaviour modelling.

7 CONCLUSION

In this work, we proposed Mouse2Vec – a novel self-supervised method to learn representations of mouse behaviour. In contrast to existing mouse behaviour modelling methods that are application-specific, our approach yields representations that can be reused across users and interactive tasks. Our results demonstrated that the representations learned by Mouse2Vec capture latent semantics and relationships of mouse behaviours in the form of interpretable clusters and allow to retrieve mouse trajectories with similar characteristics. Extensive experiments for three practical sample downstream tasks and different datasets, illustrated that Mouse2Vec is effective in improving performance when used for data augmentation or as a feature extractor. We believe its reusability and data efficiency make our approach particularly appealing as an alternative to tedious and time-consuming data collection and annotation that are common in HCI research so far. More generally, our results underline the potential of applying self-supervised methods for computational user and behaviour modelling in HCI.

ACKNOWLEDGMENTS

The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting G. Zhang. A. Bulling was funded by the European Research Council (ERC; grant agreement 801708). Z. Hu was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016. M. Bâce was funded by the Swiss National Science Foundation (SNSF) through a Postdoc. Mobility Fellowship (grant number 214434) while at the University of Stuttgart. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). We would like to thank Yuxuan Cheng for her technical support, and thank anonymous reviewers for their helpful feedback.

REFERENCES

[1] Margit Antal and Elöd Egyed-Zsigmond. 2019. Intrusion detection using mouse dynamics. *IET Biometrics* 8, 5 (2019), 285–294.

[2] Margit Antal, Norbert Fejér, and Krisztian Buza. 2021. SapiMouse: Mouse dynamics-based user authentication using deep feature learning. In *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, Timisoara, 61–66.

[3] Ioannis Arapakis and Luis A Leiva. 2020. Learning efficient representations of mouse movements to predict user attention. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 1309–1318.

[4] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. 2021. Uibert: Learning generic multimodal representations for ui understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*. IJCAI, Montreal, 1–8.

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, Montreal, 41–48.

[7] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, Gold Coast, 160–172.

[8] Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. In *2020 25th International conference on pattern recognition (ICPR)*. IEEE, Milano, 5482–5487.

[9] Penny Chong, Yuval Elovici, and Alexander Binder. 2019. User authentication based on mouse dynamics using deep neural networks: A comprehensive study. *IEEE Transactions on Information Forensics and Security* 15 (2019), 1086–1101.

[10] Penny Chong, Yi Xiang Marcus Tan, Juan Guarnizo, Yuval Elovici, and Alexander Binder. 2018. Mouse authentication without the temporal aspect–what does a 2d-cnn learn?. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, San Francisco, 15–21.

[11] Jeremy Chu, Dongsheng An, Yan Ma, Wenzhe Cui, Shumin Zhai, Xianfeng David Gu, and Xiaojun Bi. 2023. WordGesture-GAN: Modeling Word-Gesture Movement with Generative Adversarial Network. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg, 1–15.

[12] Daniela Chudá and Peter Krátky. 2014. Usage of computer mouse characteristics for identification in web browsing. In *Proceedings of the 2014 International Conference on Computer Systems and Technologies*. ACM, Ruse, 218–225.

[13] Qingfeng Dai, Yongkang Wong, Guofei Sun, Yanwei Wang, Zhou Zhou, Mohan S Kankanhalli, Xiangdong Li, and Weidong Geng. 2023. Unsupervised Domain Adaptation by Causal Learning for Biometric Signal based HCI. *ACM Transactions on Multimedia Computing, Communications and Applications* 20, 2 (2023), 1–18.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*. ACL, Minneapolis, 4171–4186.

[15] Anis Elbahi, Mohamed Ali Mahjoub, and Mohamed Nazih Omri. 2013. Hidden markov model for inferring user task using mouse movement. In *Fourth International Conference on Information and Communication Technology and Accessibility (ICTA)*. IEEE, Hammamet, 1–7.

[16] Anis Elbahi and Mohamed Nazih Omri. 2015. Web user interact task recognition based on conditional random fields. In *Computer Analysis of Images and Patterns: 16th International Conference, CAIP 2015, Valletta, Malta, September 2-4, 2015 Proceedings, Part I 16*. Springer, Valletta, 740–751.

[17] Anis Elbahi, Mohamed Nazih Omri, Mohamed Ali Mahjoub, and Kamel Garrouch. 2016. Mouse movement and probabilistic graphical models based e-learning activity recognition improvement possibilistic model. *Arabian Journal for Science and Engineering* 41, 8 (2016), 2847–2862.

[18] Paul Freihaut, Anja S Göritz, Christoph Rockstroh, and Johannes Blum. 2021. Tracking stress via the computer mouse? Promises and challenges of a potential behavioral stress marker. *Behavior Research Methods* 53, 1 (2021), 1–21.

[19] Eugene Yujun Fu, Tiffany CK Kwok, Erin You Wu, Hong Va Leong, Grace Ngai, and Stephen CF Chan. 2017. Your mouse reveals your next activity: towards predicting user intention from mouse interaction. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, Turin, 869–874.

[20] Tatiana Gabruseva, Dmytro Poplavskiy, and Alexandr Kalinin. 2020. Deep learning for automatic pneumonia detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. IEEE, Virtual, 350–351.

[21] V. Gael. 2014. hmmlearn. https://github.com/hmmlearn/hmmlearn

[22] Krzysztof Z Gajos, Katharina Reinecke, Mary Donovan, Christopher D Stephen, Albert Y Hung, Jeremy D Schmahmann, and Anoopum S Gupta. 2020. Computer mouse use captures ataxia and parkinsonism, enabling accurate measurement and detection. *Movement Disorders* 35, 2 (2020), 354–358.

[23] Hugo Gamboa and Ana Fred. 2004. A behavioral biometric system based on human-computer interaction. In *Biometric Technology for Human Identification*, Vol. 5404. SPIE, Orlando, 381–392.

[24] Daniel Garabato, Jorge Rodríguez García, Francisco J Novoa, and Carlos Dafonte. 2019. Mouse Behavior Analysis Based on Artificial Intelligence as a Second-Phase Authentication System. *Multidisciplinary Digital Publishing Institute Proceedings* 21, 1 (2019), 29.

[25] Eric J Gonzalez and Sean Follmer. 2023. Sensorimotor Simulation of Redirected Reaching using Stochastic Optimal Feedback Control. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg, 1–17.

[26] Julien Gori and Olivier Rioul. 2020. A feedback information-theoretic transmission scheme (FITTS) for modeling trajectory variability in aimed movements. *Biological cybernetics* 114, 6 (2020), 621–641.

[27] Auejin Ham, Junsu Lim, and Sunjun Kim. 2021. Do We Need a Faster Mouse? Empirical Evaluation of Asynchronicity-Induced Jitter. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, Virtual, 743–753.

[28] Lei Han, Alessandro Checco, Djellel Difallah, Gianluca Demartini, and Shazia Sadiq. 2020. Modelling user behavior dynamics with embeddings. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. ACM, Virtual, 445–454.

[29] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New Orleans, 16000–16009.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, Las Vegas, 770–778.

[31] Zhiming Hu, Andreas Bulling, Sheng Li, and Guoping Wang. 2021. FixationNet: Forecasting Eye Fixations in Task-Oriented Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 27, 5 (2021), 2681–2690. https://doi.org/10.1109/TVCG.2021.3067779

[32] Zhiming Hu, Andreas Bulling, Sheng Li, and Guoping Wang. 2022. EHTask: recognizing user tasks from eye and head movements in immersive virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 4 (2022), 1992–2004.

[33] Amit Kumar Jaiswal, Prayag Tiwari, and M Shamim Hossain. 2020. Predicting users' behavior using mouse movement information: An information foraging theory perspective. *Neural Computing and Applications* 35, 1 (2020), 1–14.

[34] Markus Klar, Florian Fischer, Arthur Fleig, Miroslav Bachinski, and Jörg Müller. 2023. Simulating Interaction Movements via Model Predictive Control. *ACM Transactions on Computer-Human Interaction* 30, 3 (2023), 1–50.

[35] Agata Kołakowska. 2013. A review of emotion recognition methods based on keystroke dynamics and mouse movements. In *2013 6th international conference on human system interactions (HSI)*. IEEE, Poland, 548–555.

[36] Saskia Koldijk, Mark Van Staalduinen, Mark Neerincx, and Wessel Kraaij. 2012. Real-time task recognition based on knowledge workers' computer activities. In *Proceedings of the 30th European Conference on Cognitive Ergonomics*. ACM, Edinburgh, 152–159.

[37] M. Korobov. 2019. sklearn-crfsuit. https://github.com/TeamHG-Memex/sklearn-crfsuite

[38] Tiffany CK Kwok, Eugene Yujun Fu, Erin You Wu, Michael Xuelin Huang, Grace Ngai, and Hong-Va Leong. 2018. Every little movement has a meaning of its own: Using past mouse movements to predict the next interaction. In *23rd International Conference on Intelligent User Interfaces*. ACM, Berlin, 397–401.

[39] Dmitry Lagun, Mikhail Ageev, Qi Guo, and Eugene Agichtein. 2014. Discovering common motifs in cursor movement data for improving web search. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, New York, 183–192.

[40] Friedrich Leisch. 2006. A toolbox for k-centroids cluster analysis. *Computational statistics & data analysis* 51, 2 (2006), 526–544.

[41] Luis A Leiva, Ioannis Arapakis, and Costas Iordanou. 2021. My mouse, my rules: Privacy issues of behavioral user profiling via mouse tracking. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*. ACM, Canberra, 51–61.

[42] Haotian Li, Yong Wang, Aoyu Wu, Huan Wei, and Huamin Qu. 2022. Structure-aware visualization retrieval. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans, 1–14.

[43] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Virtual, 1–15.

[44] Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. 2023. Deep Graph Representation Learning and Optimization for Influence Maximization. In *International Conference on Machine Learning*. PMLR, Honolulu, 21350–21361.

[45] Hechen Liu and Markus Schneider. 2012. Similarity measurement of moving object trajectories. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on GeoStreaming*. ACM, Redondo Beach, 19–22.

[46] David S March and Lowell Gaertner. 2021. A method for estimating the time of initiating correct categorization in mouse-tracking. *Behavior Research Methods* 53, 1 (2021), 1–11.

[47] Christian Meurisch, Cristina A Mihale-Wilson, Adrian Hawlitschek, Florian Giger, Florian Müller, Oliver Hinz, and Max Mühlhäuser. 2020. Exploring user expectations of proactive AI systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–22.

[48] Christopher Murphy, Jiaju Huang, Daqing Hou, and Stephanie Schuckers. 2017. Shared dataset on natural human-computer interaction to support continuous authentication research. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, Denver, 525–530.

[49] Alexandra Papoutsaki, Aaron Gokaslan, James Tompkin, Yuze He, and Jeff Huang. 2018. The eye of the typer: a benchmark and analysis of gaze behavior during typing. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, Warsaw, 1–9.

[50] Phillip T Pasqual and Jacob O Wobbrock. 2014. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto, 743–752.

[51] Md Muhaimenur Rahman and Sarnali Basak. 2021. Identifying user authentication and most frequently used region based on mouse movement data: A machine learning approach. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, Virtual, 1245–1250.

[52] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. https://doi.org/10.48550/ARXIV.1908.10084

[53] Jun Rekimoto. 2023. WESPER: Zero-shot and Realtime Whisper to Normal Voice Conversion for Whisper-based Speech Interactions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg, 1–12.

[54] Hansol Rheem, Vipin Verma, and D Vaughn Becker. 2018. Use of mouse-tracking method to measure cognitive load. In *Proceedings of the human factors and ergonomics society annual meeting*. SAGE Publications Sage CA: Los Angeles, CA, Los Angeles, 1982–1986.

[55] Diogo Aranha Ribeiro, Danton Diego Ferreira, Daniel Augusto Pereira, Roberto Alves Braga Junior, and Rodrigo Dantas Nunes. 2019. Mechanical fault detection in electric motors measured by a digital signal processing device in an optical mouse. *Measurement* 138 (2019), 350–355.

[56] Sergio Salmeron-Majadas, Ryan S Baker, Olga C Santos, and Jesus G Boticario. 2018. A machine learning approach to leverage individual keyboard and mouse interaction behavior from multiple users in real-world learning scenarios. *IEEE Access* 6 (2018), 39154–39179.

[57] Nyle Siddiqui, Rushit Dave, Mounika Vanamala, and Naeem Seliya. 2022. Machine and deep learning applications to mouse dynamics for continuous user authentication. *Machine Learning and Knowledge Extraction* 4, 2 (2022), 502–518.

[58] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition.

[59] Zixiong Su, Shitao Fang, and Jun Rekimoto. 2023. LipLearner: Customizable Silent Speech Interactions on Mobile Devices. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg, 1–21.

[60] Yan Sun, Hayreddin Ceker, and Shambhu Upadhyaya. 2016. Shared keystroke dataset for continuous authentication.

[61] Yan Sun, Hayreddin Ceker, and Shambhu Upadhyaya. 2016. Shared keystroke dataset for continuous authentication. https://doi.org/10.1109/WIFS.2016.7823894

[62] Yi Xiang Marcus Tan, Alfonso Iacovazzi, Ivan Homoliak, Yuval Elovici, and Alexander Binder. 2019. Adversarial attacks on remote user authentication using behavioural mouse dynamics. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Budapest, 1–10.

[63] Emanuel Todorov and Michael I Jordan. 2002. Optimal feedback control as a theory of motor coordination. *Nature neuroscience* 5, 11 (2002), 1226–1235.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett (Eds.), Vol. 30. Curran Associates, Inc., Long Beach, 1–11. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[65] Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile UI summarization with multimodal learning. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, Virtual, 498–510.

[66] Ziqi Wang, Yuexin Wu, Frederick Liu, Daogao Liu, Le Hou, Hongkun Yu, Jing Li, and Heng Ji. 2023. Augmentation with Projection: Towards an Effective and Efficient Data Augmentation Paradigm for Distillation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, Kigali, 1–20. https://openreview.net/pdf?id=kPPVmUF6bM_

[67] Datong Wei, Chaofan Yang, Xiaolong Zhang, and Xiaoru Yuan. 2021. Predicting mouse click position using long short-term memory model trained by joint loss function. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Virtual, 1–6.

[68] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. 2021. Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. ACM, Coimbra, 220–233.

[69] Pingmei Xu, Yusuke Sugano, and Andreas Bulling. 2016. Spatio-temporal modeling and prediction of visual attention in graphical user interfaces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose, 3299–3310.

[70] Metehan Yildirim and Emin Anarim. 2022. Mitigating insider threat by profiling users based on mouse usage pattern: ensemble learning and frequency domain analysis. *International Journal of Information Security* 21, 2 (2022), 239–251.

[71] Bereket A Yilma and Luis A Leiva. 2023. The Elements of Visual Art Recommendation: Learning Latent Semantic Representations of Paintings. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, Hamburg, 1–17.

[72] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. 2019. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF international conference on computer vision*. IEEE, Seoul, 1476–1485.

[73] Guanhua Zhang, Matteo Bortoletto, Zhiming Hu, Lei Shi, Mihai Bâce, and Andreas Bulling. 2023. Exploring Natural Language Processing Methods for Interactive Behaviour Modelling. In *The Proceeding of 2023 IFIP TC13 Conference on Human-Computer Interaction (INTERACT)*. IFIP, York, 1–18.

[74] Guanhua Zhang, Susanne Hindennach, Jan Leusmann, Felix Bühler, Benedict Steuerlein, Sven Mayer, Mihai Bâce, and Andreas Bulling. 2022. Predicting Next Actions and Latent Intents during Text Formatting. In *Proceedings of the CHI Workshop Computational Approaches for Understanding, Generating, and Adapting User Interfaces*. ACM, New Orleans, 1–6.

[75] Wenrui Zhang, Ling Yang, Shijia Geng, and Shenda Hong. 2023. Self-Supervised Time Series Representation Learning via Cross Reconstruction Transformer. *IEEE Transactions on Neural Networks and Learning Systems* 1, 1 (2023), 1–11.

[76] Chen Zhao, Le Wu, Pengyang Shao, Kun Zhang, Richang Hong, and Meng Wang. 2023. Fair Representation Learning for Recommendation: A Mutual Information Perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI, Washington DC, 4911–4919.

A COMPLETE FEATURE SET

Table 3 shows the 100 handcrafted features used in our work as a baseline mouse representation. These features were commonly used in prior mouse behaviour modelling works as introduced in Section 2.2.

B BUFFALO AND EMAKI

Since our Mouse2Vec was trained on two datasets, Buffalo and EMAKI, we examined if they both contribute useful information to the representation. Therefore, we compared the accuracies achieved on downstream tasks using Mouse2Vec pretrained on both datasets, on only Buffalo or only EMAKI. Table 4 shows the accuracies achieved on three downstream tasks on two datasets. We can see

Statistic	Description	Reference
Total number	Clicks	[36]
Mean, Median, Maximum, Minimum, Standard deviation	X and Y coordinate	[23, 38]
	Travel distance	[1, 18, 19, 23, 56]
	Straight distance	[19]
	X, Y, angular and total speed	[1, 18, 19, 23, 38, 54, 56]
	X, Y and total acceleration	[1, 18, 19, 23, 38, 56]
	Angle, angle difference	[1, 18, 19, 23, 38, 56]
	Jerk	[1]
	Curvature	[1, 23]
Mean frequency, Mean power, Peak power, Peak power frequency	Speed, Acceleration, Jerk, Angular velocity, Curvature, Curvature difference	[55, 70]

Table 3: A collective mouse feature set used in prior mouse behaviour modelling methods.

Training Dataset(s)	ACTIVITY Dataset		EOTT Dataset	
	Next Activity Prediction	User Identification	Interactive Task Recognition	User Identification
Buffalo	59.69±2.82	18.15±0.88	73.63±4.56	8.26±0.70
EMAKI	54.81±1.87	15.33±0.99	71.56±3.25	6.38±0.33
Buffalo + EMAKI	**62.69±3.57*****	**19.59±0.60****	**75.84±3.39***	**9.43±0.42*****

Table 4: Accuracy (Mean±Std, in percentage) on three downstream tasks achieved by Mouse2Vec pretrained on different datasets. Best results are shown in bold. Stars mark the significance levels of difference between Mouse2Vec and the second best results ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

that pretraining on Buffalo dataset outperformed on EMAKI. This is likely because the Buffalo dataset has a much larger amount of data compared to EMAKI (about 4 times). This means that the Buffalo dataset can encode richer information about mouse behaviour. However, pretraining on both datasets achieved the highest accuracies. This indicates that both datasets contribute to the representation, which is consistent with previous research [5] and validates the effectiveness of our training strategy.

C APPARATUS AND TIME CONSUMPTION OF MOUSE2VEC

We trained and evaluated Mouse2Vec on a server, which has Intel Xeon Platinum 8260 CPU and NVIDIA Tesla V100-32GB GPU. Dealing with a batch of mouse data (512 mouse trajectories) required ~3.87 GB of RAM. Mouse2Vec works on both GPU and CPU. Table 5 presents the time consumption of processing one batch on GPU or CPU to: train Mouse2Vec, extract representation using the pretrained Mouse2Vec, train an MLP-based classifier after the frozen Mouse2Vec, and fine-tune Mouse2Vec together with the classifier. For each scenario, we calculated the average time across all the batches. It can be seen that on GPU, our Mouse2Vec is promising to be used in real-time applications.

D HIGHER RESAMPLING RATES

We evaluated Mouse2Vec when resampling mouse data at 20 Hz [50, 67] in the main paper. We also tested Mouse2Vec at higher sampling rates, i.e., 100 Hz, 250 Hz or 500 Hz [22, 27] for downstream tasks. As shown in Table 6, the performances are similar among these sampling rates (<2% difference). To save compute time and resources, we chose 20 Hz.

E PROPORTIONS OF TRAINING SET

We trained classifiers for downstream tasks on a fraction (10%, 20% or 50%) of the annotated data, based on the classical representations or Mouse2Vec features (freezing or fine-tuning). As shown in Figure 6, Mouse2Vec consistently outperforms the baselines on all the subsets. When trained on 50% data, the frozen Mouse2Vec gets the accuracy comparable to classical representations that were trained on the entire dataset. The performance improvements become even more significant when fine-tuning Mouse2Vec – using 20% of the training set already shows higher accuracies than baselines. Our result is promising given that it points towards a future in which only fractions of training data actually will have to be collected and annotated manually, potentially saving 50% or even 80% of the annotation effort; and will be complemented with synthetic data to improve performance of user behaviour models (Section 5.1).

	GPU	CPU
Self-supervised training of Mouse2Vec	0.15	2.89
Extract representation with Mouse2Vec	0.05	0.67
Frozen Mouse2Vec + Classifier	0.12	3.31
Finetune Mouse2Vec + Classifier	0.15	3.92

Table 5: Average time consumption (in second) processing one batch (512 mouse trajectories) in different scenarios using GPU or CPU.

Sampling Rate (Hz)	ACTIVITY Dataset		EOTT Dataset	
	Next Activity Prediction	User Identification	Interactive Task Recognition	User Identification
20	62.69±3.57	**19.59±0.60**	75.84±3.39	9.43±0.42
100	**62.99±1.06**	18.50±1.08	75.10±5.16	8.95±0.70
250	61.70±1.58	18.06±1.21	74.93±3.51	**9.75±0.52**
500	62.00±3.04	18.89±0.56	**76.21±3.43**	9.44±0.75

Table 6: Accuracies (Mean±Std, in percentage) on interactive task recognition (EOTT dataset), next activity prediction (ACTIVITY dataset) and user identification (both datasets). We compare results achieved by pretrained Mouse2Vec when using different sampling rates of mouse data. Best results are shown in bold.

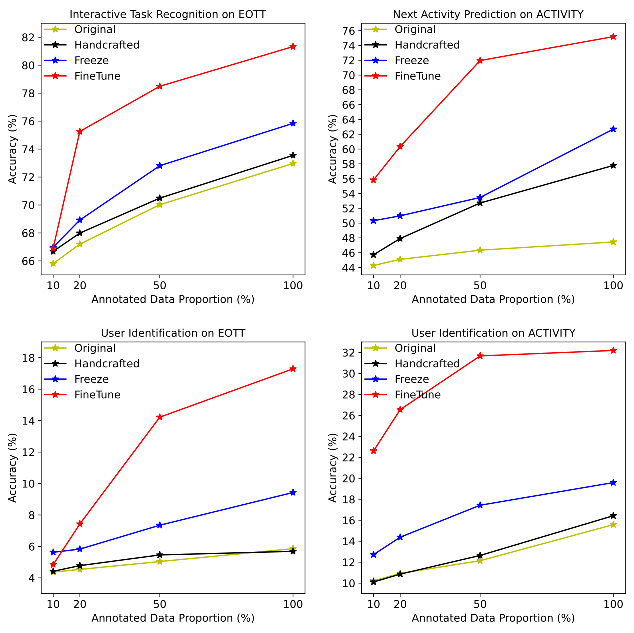


Figure 6: Accuracy achieved by original mouse data, handcrafted features, and freezing (Freeze) or fine-tuning (FineTune) the pretrained Mouse2Vec using different proportions of training set (annotated data).

F COMPARISON WITH PRIOR TASK-SPECIFIC METHODS

We compared Mouse2Vec with mouse-based data-driven methods that have been proposed by prior works specifically geared towards each downstream task. We evaluated both freezing and fine-tuning Mouse2Vec as in Section 5.2. Between each of them and the best performing prior method, we conducted a Wilcoxon signed-rank test to examine the significance of the results.

F.1 Interactive Task Recognition

The following are the state-of-the-art (SOTA) methods specifically proposed for interactive task recognition:

- *Original + HMM*: Hidden Markov model (HMM) was used by Elbahi et al. [15, 17] for interactive task recognition. We tried different types of HMMs (GaussianHMM, GMMHMM, MultinomialHMM, PoissonHMM and VariationalGaussianHMM) from the Scikit-Learn hmmlearn package [21] and reported the best performance.
- *Original + CRF*: Inspired by prior works from Elbahi et al. [16, 17] that employed conditional random field (CRF), we trained a CRF model using the Sklearn-crfsuite package [37]. We tuned hyperparameters c1 and c2 in an exponential space[4] and reported the best performance.
- *Features + NB*, *Features + KStar*, *Features + DT* and *Features + MLP*: Koldijk et al. [36] extracted handcrafted features and trained naive Bayes (NB), KStar, decision tree (DT) and MLP as classifiers. Accordingly, we trained these four classifiers on handcrafted features (Table 3) using the implementation suggested by the authors.

Table 7 shows the results of these evaluations for each method. Freezing Mouse2Vec already outperforms all SOTA methods, while fine-tuning it leads to a better performance, with an accuracy of 81.33% (7.78% higher than the best performing SOTA *Features+MLP*). The Wilcoxon test validated the significance of the outperformance. Among the SOTAs, the best method obtained a minor accuracy increment of 1.23%, indicating the difficult of improving performance for this downstream task.

Method		Accuracy
Original + HMM		42.24±13.42
Original + CRF		49.70±3.55
Features + NB		58.47±5.15
Features + Kstar		70.22±1.51
Features + DT		72.32±1.10
Features + MLP		73.55±2.89
Mouse2Vec	Freeze	<u>75.84±3.39</u>*
	FineTune	**81.33±3.05****

Table 7: Interactive task recognition accuracy (Mean±Std, in percentage) achieved by directly applying Mouse2Vec (Freeze), fine-tuning Mouse2Vec (FineTune) and the different baselines inspired by existing methods [15–17, 36]. Best results are shown in bold, and the second best results are <u>underlined</u>. Stars mark the significance levels of difference between Mouse2Vec and the best SOTA method ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

Method		Accuracy
Features & Activities+SVM		68.46±4.43
Features & Activities+RF		69.15±8.10
Features & Activities+LSTM		71.78±5.10
Mouse2Vec	Freeze	62.69±3.57**
	FineTune	<u>75.82±3.88</u>*
Mouse2Vec & Activities	Freeze	75.19±2.28*
	FineTune	**76.79±1.96****

Table 8: Next activity prediction accuracy (Mean±Std, in percentage) achieved by directly applying Mouse2Vec (Freeze), fine-tuning Mouse2Vec (FineTune) using or not activity history, and the different baselines inspired by existing methods [19, 38, 74]. Best results are shown in bold, and the second best results are <u>underlined</u>. Stars mark the significance levels of difference between Mouse2Vec and the best SOTA method ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

F.2 Next Activity Prediction

The following prior methods were proposed for next activity prediction:

- *Features & Activities + SVM*: Fu et al. used a support vector machine (SVM) on the fusion of handcrafted features and four history activities to predict the next activity [19]. We fused handcrafted features with past four activities and implemented the SVM classifier using Scikit-Learn optimising hyperparameters[5] $C \in [1, 10, 100, 1000], \gamma \in [0.001, 0.0001]$.
- *Features & Activities + LSTM*: Kwok et al. [38] trained two separate LSTM models for two types of input – handcrafted features and five history activities. Then the weighted fusion of the predictions from the two models was used to generate final predictions. Following the original implementation, we tuned the weight α between the two inputs in the range of 0.1 to 0.9 with a step size of 0.1 as well as the learning rate in [1e-3, 1e-4], and reported the best result.
- *Features & Activities + RF*: Zhang et al. [74] used a two-stream random forest (RF) to predict next activities. While one stream processed handcrafted features, the other one handled past seven activities. We implemented RF using Scikit-Learn, tuned the weight α following the original implementation as well as the number of tree in [10,100,1000], and reported the best result.

[4]https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html
[5]https://scikit-learn.org/stable/modules/grid_search.html

Given that all the methods used the history of activities as an additional input, we also built a variant of our method that used this information. Specifically, we concatenated the Mouse2Vec representation with the seven most recent activities and used the resulting vector as the input to the classifier, denoted as *Mouse2Vec & Activities*. As Table 8 shows, using mouse behaviour only, fine-tuning Mouse2Vec outperformed SOTA methods that used activities. After adding history activities, the frozen Mouse2Vec gained a significantly higher accuracy. It improved the accuracy from the best-performing prior method (3.41% higher than *Features & Activities + LSTM*) more than this method improved from the second-best prior method (2.63% higher than *Features & Activities + RF*), indicating that the improvement of our method is meaningful.

F.3 User Identification

Only one prior work conducted user identification from mouse data:

- *Features + kNN*: Chuda et al. used a kNN classifier on the handcrafted features to identify users [12]. We implemented the classifier using Scikit-Learn and optimised the number of nearest neighbours in [5, 10, 50, 100].

As Table 9 shows, freezing Mouse2Vec achieved significantly higher accuracies on both datasets, while fine-tuning Mouse2Vec improved the performance even more, e.g., reaching 32.19% accuracy identifying 16 users from ACTIVITY.

Method		**ACTIVITY**	**EOTT**
Features + kNN		15.91±0.53	5.97±0.48
Mouse2Vec	Freeze	19.59±0.60*	9.43±0.42**
	FineTune	**32.19±0.83*****	**17.29±1.10****

Table 9: User identification accuracy (Mean±Std, in percentage) achieved by directly applying Mouse2Vec (Freeze), fine-tuning Mouse2Vec (FineTune) and the different baselines inspired by existing methods [12]. Best results are shown in bold. Stars mark the significance levels of difference between Mouse2Vec and the SOTA method ($^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).