

ML-Fall24-Team7

Detecting User Actions from Mouse Events

Introduction

This project aims to classify a user's current activity (e.g., web browsing, chatting, watching videos, reading) based solely on their mouse events. This approach enhances productivity by enabling features like sharing status updates on platforms like Slack or activating Focus mode, while preserving user privacy by avoiding more invasive monitoring methods such as screen captures or keystroke logging.

Problem Statement

- **Problem:** Automatically identifying user activity purely based on mouse events, which avoids privacy concerns.
- **Motivation:** Enhancing user productivity and privacy by providing non-invasive activity recognition.

Objectives

- Develop a model capable of distinguishing user activity with high accuracy.
- Preserve user privacy by using non-invasive data collection methods.
- Enable real-time inference so that current action can be predicted without any delays.

Literature Review

Previous research has attempted to identify individuals based on their mouse usage patterns [3] or predict the next mouse event of the user [2]. However, we are approaching the problem by classifying the current user's high-level activity in real time. Kuric et al. [1] have suggested several features (such as clicks, velocity, acceleration, etc.) that can be used to classify current actions, which can support our project.

Methodology

Data collection

Data was collected from 5 team members who each performed mouse events for 10 minutes. Mouse events included chatting, playing chess, making ppt, reading, web browsing, and watching youtube videos. We extracted features for each time window from our recordings.txt and turned into csv file. For each dataset, it had 2998 samples each representing interaction instance and 28 features capturing various characteristics of user actions.

Data Preprocessing Method Implemented

- **Feature Engineering:** Extracting additional insights like scroll patterns, click patterns, idle times, or mouse velocity.

Mouse Movement-Inferred Feature Overview:

Table 1

Overview of mouse-movement-inferred features (mouse dynamics) across literature. Conceptually related metrics that appear with varied mutations are grouped together, aggregates and variants listed.

Feature	Description	Aggregates	Variants
<i>Temporal features</i>			
Velocity	Change in distance over time	Min, max, mean, sd, min-max difference	x-axis, y-axis, smoothed
Acceleration	Change in velocity over time	Min, max, mean, sd, min-max difference	x-axis, y-axis, positive, negative
Jerk	Change in acceleration over time	Mean, sd	Positive, negative
Snap	Change in jerk over time	Mean, sd	Positive, negative
Angular velocity	Change of angle over time	Mean, sd	–
Movement duration	Time between clicks	–	Total
Reaction time	Time until response	–	–
Initiation time	Time until mouse movement	–	–
Pause	Idle cursor time	Count, sum	–
<i>Spatial features</i>			
Movement distance	Path length	–	Smoothed, total
Straightness	Path length ratio to ideal path	Mean, sd	–
Path deviation	Perpendicular distance to ideal path	Max	AUC
Jitter	Smoothed to real path length ratio	–	–
Angle	Current movement direction	–	–
Flips	Movement directional changes	Count	x-axis, y-axis
Path crossings	Intersections of path and ideal path	Count	–
Curvature	Change of angle over distance	Mean, sd	–
Inflection points	Curvature changes, flex points	Count	–
<i>Stimuli interaction features</i>			
Clicks	Number of stimuli presses	Count	–
Hold time	Stimuli press duration	–	–
Click error	Stimuli center and press distance	–	x-axis, y-axis, absolute
Time to click	Stimuli hover to press time	–	–
Scroll	Page scrolling	–	Horizontal, vertical

- **Dimensionality Reduction (PCA and t-SNE):** PCA: Reduced high-dimensional data while retaining key features, minimizing overfitting and enhancing computational efficiency. t-SNE: Used for 2D visualization to reveal data structure and cluster distribution, helping to better understand relationships between classes.
- **Windowing:** Divided the data into windows of varying sizes and intervals to capture both short-term and long-term patterns, enabling effective analysis of time-sequence data. We also measured the effect of changing the window sizes.
- **Oversampling:** Addressed class imbalance by duplicating minority class samples. RandomOverSampler was used to balance class distribution, allowing the model to learn each class more effectively.

- **Scaling** As each feature would have different distribution in their own space, we normalize each feature using the Standard scaler. We've used the code from the scikit-learn library.
- **Null Frame Filling:** As idle mouse events can occur, we fill in the empty frames with null frames so that ML/DL models acknowledge the empty frames as idle mouse events.

ML Algorithms/Models Implemented

Supervised Learning

In our project, we implemented both supervised and unsupervised learning methods for user behavior detection based on mouse movement data. We split the training and testing data into approximately 8:2 ratio as the dataset became larger. For supervised learning, we trained 3 ML models (**LightGBM**, **Random Forest**, **KNN**), and also the **LSTM** model. We also trained the models with varying window size of 50, 100, and 300 to examine the effect of window size in terms of accuracy. Alongside accuracy we also analyzed the inference time among the models.

Unsupervised Learning

For unsupervised learning, we used **K-Means** clustering to identify hidden patterns and groups in the data. To do this, we used PCA and t-SNE as dimensionality reduction techniques to visualize the characteristics of the data and increase the performance of the clustering. The raw features extracted from the preprocessing algorithm are used as the target data for the clustering.

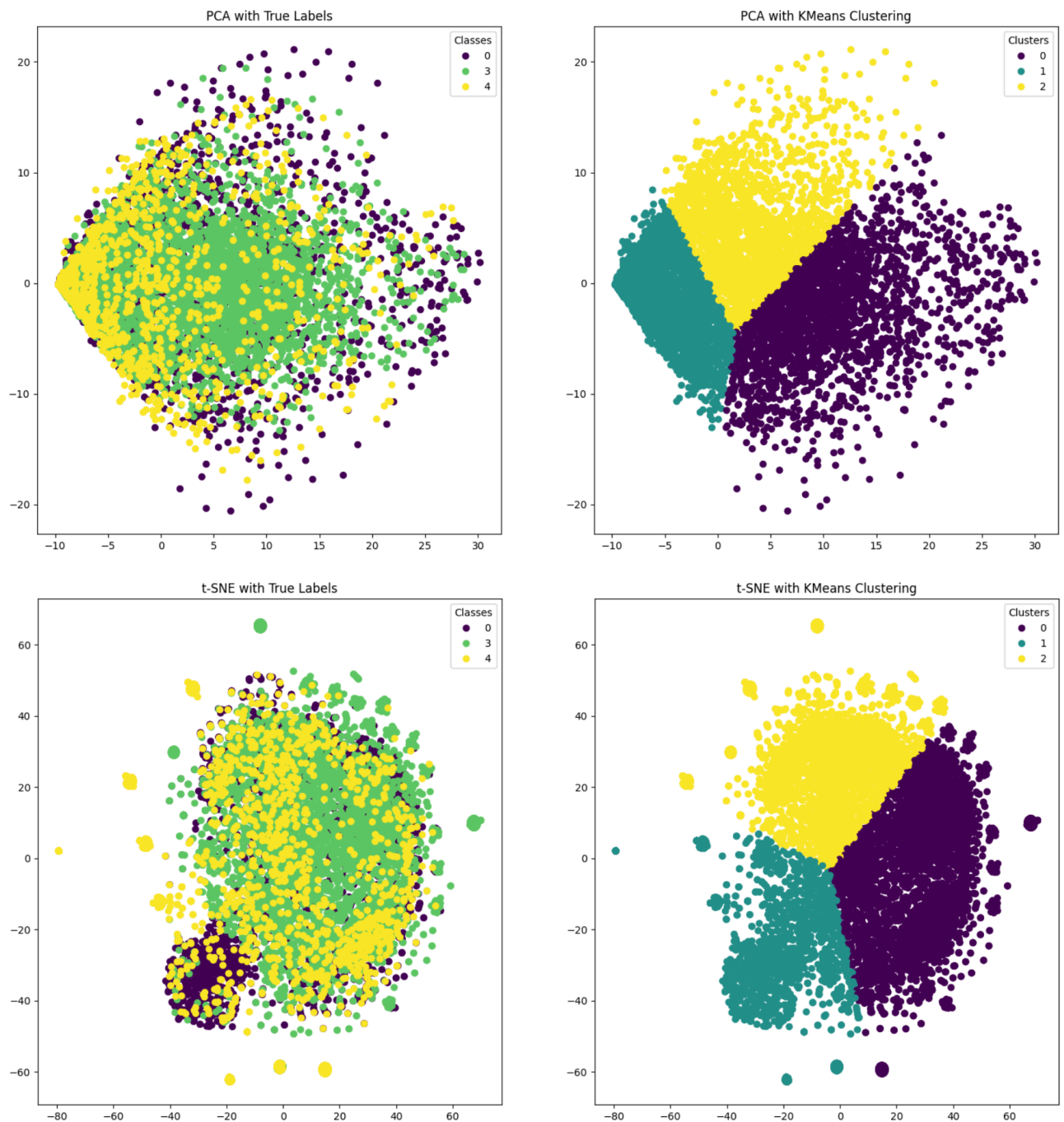
Relevant Courses and Methods

- **CS 7641:** Unsupervised and Supervised Learning focusing on machine learning approaches for plain data.
- **CS 7643:** Deep learning models and algorithms, including time-series analysis.

Results and Discussion

- **Visualizations:**
 - **Unsupervised Learning Method: K-means**

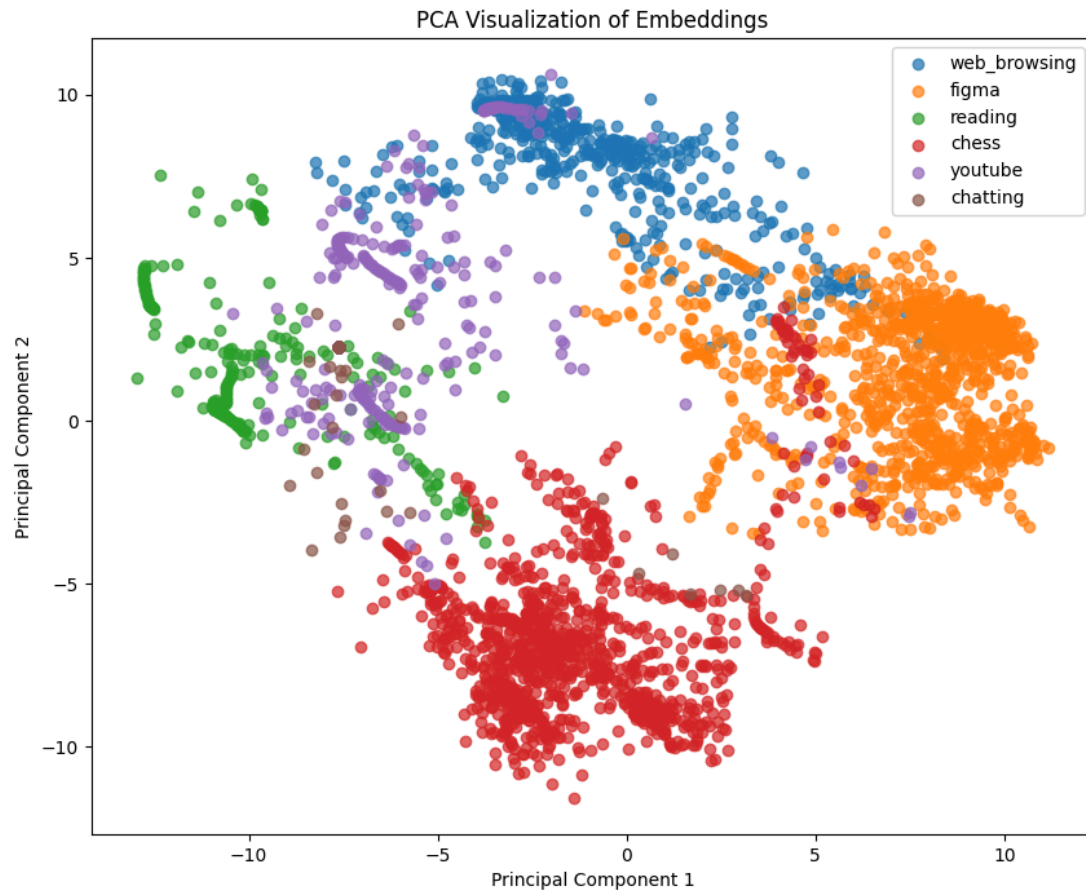
In our project, we used a variety of visualizations to analyze the mouse movement data. We leveraged PCA and t-SNE to reduce the dimensionality so that we could visually interpret the high-dimensional data, and visualized the K-Means clustering results to more clearly see the boundaries of the clusters. The visualization of the actual labels versus the clustered results helped us understand if the model was detecting patterns in the data well.



- Supervised Learning Method (Embeddings): LSTM with 30 seconds (300 frames) window size

We visualized the embedding of the trained LSTM model to observe whether the classification worked well in the latent vector space. As the image implies, the classes are

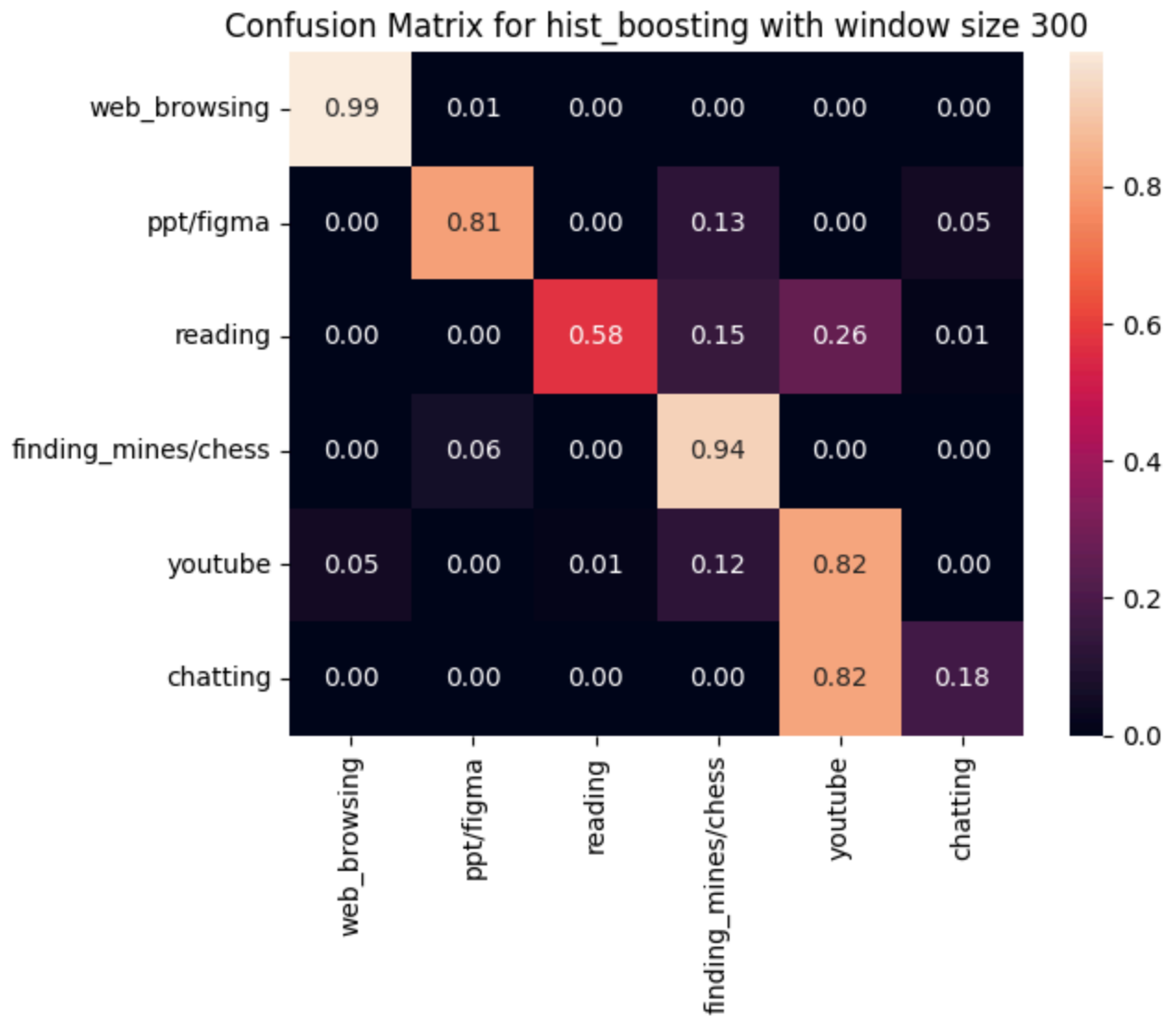
well-clustered even though a linear dimensionality reduction method (PCA) is used.



- Supervised Learning Method (Confusion Matrix): LightGBM with 30 seconds (300 frames) window size

We also visualized the confusion matrix, which represents the performance of the LightGBM model, to check the prediction accuracy for each class. The result looks promising for some of the classes like "web-browsing" and "playing games", which reaches over 94% of accuracy. However, some classes like "chatting" and "reading" has relatively poor classification accuracy below 60%. We might be able to handle this

problem through class merging like “playing games” or add more training data.



- Quantitative Analysis:

The following table shows the test accuracy for each model and window size. We derive insights from the table's results.

Models \ Window Size	300	100	50
Gradient Boosting	0.85	0.75	0.69
Random Forest	0.71	0.67	0.64
KNN	0.22	0.29	0.31
LSTM	0.85	0.77	0.75

- LSTM performs better than other ML models because of strong non-linearity which extracts complex patterns from the diverse mouse events. The capability of capturing the

time sequence information better than other models allow it to perform better than other models on smaller windows.

- ii. Gradient Boosting performs almost the same when the feature size is large (window size is large), indicating that large window size allows lighter ML models to learn complex patterns. However, the performance slightly drops as the feature size (window length) decreases.
 - iii. Random forest performs slightly worse than gradient boosting, indicating that boosting algorithms (regression trees) are better at discovering non-linear patterns than pure combination of decision trees.
 - iv. Knn performs so bad that it's almost same as random guess. Strong non-linearity is required to solve this classification task.
- **Comparison between LSTM and Gradient Boosting:** The LightGBM model used in the project was chosen because it is lightweight, efficient, and has excellent characteristics in terms of learning speed and performance. This was represented on the highest accuracy in 300-window. In terms of LSTM, it is also known to be lightweight but more effective on capturing the sequence information from the training data. This is represented as continuously high accuracy in 100 and 50-sized windows. Therefore, we can suggest applying different models based on the corresponding usages. For example, if fast inference is required and the window size should be small (shorter observing time), then LSTM with lower window size would be the best choice. However, only for the higher accuracy, I believe Gradient boosting model can perform well due to fast training and inference on the CPU.
 - **Analysis of the Confusion Matrix:** As the confusion matrix shows, web browsing and game is the easiest task among the 5 categories. However, chatting and reading papers are shown as very difficult task. This is due to the motion overlap among the tasks. For example, the motion of reading can be included or be seen as web browsing, and chatting can be interpreted as youtube watching as it rarely includes mouse movement. Therefore, adding new features that can assist differentiating these actions would further enhance the model's classification performance.
 - **Inference Time Analysis:**
For the fair comparison, we analyzed how much time the model costs to inference single data sample. We have tested 100 times total and averaged the time cost to minimize external impacts.

Models	Time Cost(sec)
Gradient Boosting	0.02
Random Forest	0.017

Models	Time Cost(sec)
LSTM (CPU)	0.003
LSTM (GPU)	0.005

The inference time for Gradient Boosting algorithm was the slowest as it follows the large tree which is built during the training process. To learn the complex pattern of the data, the tree had to be large, leading to slower inference time. On the other hand, LSTM with CPU was the fastest. We can further improve the LSTM model's inference speed with the following approaches. First, mini-batching can be properly utilized to make real-time inference as semi-real-time while enhancing the speed and lowering the delay. Also, GPU with higher performance can be used to further increase the speed, but the memory copy overhead should be properly addressed to make the real-time inference faster than the CPU.

- **Next Steps:**

As the extension of this project to have higher accuracy and performance, we would have to focus on two directions. First, large amount of clean data should be collected. Currently we only have 5 participants' data of less than 1 hour each. This strongly limits the model's size and architecture, forcing us to use ML models instead of state-of-the-art deep learning models. Secondly, better model and hyperparameters can be selected. Due to the limited time, we only have tested couple of ML/DL models and architectures with fixed hyperparameters. If combined with larger datasets, we would be able to train complex but more powerful models like Attention or CNN-based models. Overall, larger datasets and model capacity with proper hyperparameters would further enhance the classification performance.

Project Timeline

Below is the Gantt chart outlining each group member's responsibilities for the project.

[Gantt Chart](#)

[illegible]

Each group member's specific contributions to the project proposal are outlined in the table below. The preparation for mid-term/final reports and presentation are done together during the team meeting.

Name	Proposal Contributions
Ji Min Park	Set up evaluation metrics (accuracy, F1-score, time cost), visualizations, and analysis of model performance and feature importance.
Hyunju Ji	Implemented feature engineering, including velocity, acceleration, scroll patterns, and other mouse dynamics features.
Woohyun Noh	Conducted literature review and contributed to data processing and feature engineering, focusing on feature extraction and integration.
Jungwoo Park	Implemented data preprocessing methods(Oversampling, Scaling, Feature Extraction PCA t-SNE).
Minsuk Chang	Developed and trained ML/DL models (e.g., LightGBM, LSTM, K-means), and experimented with different model configurations.

1. E. Kuric, P. Demcak, M. Krajcovic, and P. Nemcek, "Is mouse dynamics information credible for user behavior research? An empirical investigation," *Computer Standards & Interfaces*, vol. 90, p. 103849, 2024.
2. E. Y. Fu et al., "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, pp. 869–874, 2017.
3. J. J. Matthiesen and U. Brefeld, "Assessing user behavior by mouse movements," in *HCI International 2020-Posters*, pp. 68–75, 2020.