Advanced User Authentication using Mouse Dynamics and Sequence Models

CS6140 Course Project Report

December 10, 2024

Team Members

- **Punith Subashchandra** - subashchandra.p@northeastern.edu
- **Vinav Sancheti** - sancheti.v@northeastern.edu
- **Manasa Rao** - rao.mana@northeastern.edu

Project Advisor

This project is being conducted under the guidance of Professor **Predrag Radivojac** at Northeastern University.

Objective and Significance

The primary objective of this project was to develop a robust, non-intrusive user authentication system based on mouse dynamics using advanced machine learning techniques. The focus was on leveraging the sequential nature of mouse movements to accurately model user behaviour patterns over time, ensuring continuous user verification throughout a session. Unlike traditional authentication systems that rely on static credentials such as passwords or one-time biometric scans, this approach aims to provide ongoing authentication. By analyzing micro-behaviors like speed, direction, and movement angles during real-time mouse usage, the system can identify users with high accuracy, adding a critical layer of security against unauthorized access. The motivation for this research stems from the increasing reliance on remote work environments, online education platforms, and cloud-based services. As digital systems become more integral to our daily lives, the limitations of password-based authentication have become more apparent. Passwords can be easily stolen, guessed, or shared, leading to potential security breaches. Even static biometrics such as fingerprints or facial recognition are not foolproof, as they can be hacked, spoofed, or rendered ineffective by environmental conditions. In contrast, behavioural biometrics such as mouse dynamics offer a continuous, passive way to monitor user identity throughout an active session, ensuring that even if initial credentials are compromised, further unauthorized actions can be detected and prevented.

Additionally, this project explores new frontiers in sequence modeling. While previous studies in the field of mouse dynamics have focused heavily on feature engineering or CNN-based architectures, this research aims to capture temporal dependencies more effectively through LSTMs and Siamese Networks . The insights gained from this project can also be extended to other behavioral biometric applications, such as keystroke dynamics, which would further broaden the impact and applicability of the work. In the event that mouse dynamics presents challenges (e.g., noisy data or insufficient variability), the same machine learning frameworks can be seamlessly adapted to keystroke data to explore user authentication via typing patterns.

We had aimed to develop an end-to-end system that:

- Collects mouse dynamics data during an enrollment phase, where the user interacts with a simple, yet movement-rich game.
- Processes and extracts meaningful features (velocity, direction, normalized coordinates) from these raw data points.
- Trains a Siamese neural network to learn a low-dimensional embedding space in which the legitimate user's patterns cluster tightly, while impostor patterns lie farther away.
- Conducts an authentication phase with a slightly modified game to assess whether the recorded movements match the enrolled user's profile.

This endeavor is significant because it addresses both security and usability: a user's behavioral pattern is hard to mimic, and continuous verification through natural interaction reduces the cognitive burden of traditional authentication methods. If successful, this approach can enhance security on personal devices, corporate workstations, or remote login systems, potentially mitigating risks associated with compromised credentials.

This project demonstrates the effectiveness of mouse dynamics for continuous user authentication. The Siamese network significantly outperformed the LSTM model, achieving 90% validation accuracy and excelling in one-shot learning by requiring minimal enrollment data while maintaining robustness against impostors. The approach balances usability and security through a flexible threshold parameter and shows potential for broader applications in behavioral biometrics.This is significant not only for its immediate practical applications in cybersecurity but also for its potential to advance research in behavioural biometrics and sequence learning. The ability to secure systems continuously with minimal user involvement is critical for high-stakes environments like financial systems, academic testing platforms, and remote work applications, where security breaches can have far-reaching consequences.

Background

Introduction to Behavioral Biometrics

Behavioral biometrics refers to the analysis of behavioral patterns for user authentication and identity verification. Unlike physiological biometrics (e.g., fingerprint recognition), behavioral biometrics do not require specialized sensors. They leverage natural user interactions with devices, such as keystrokes or mouse movements, to provide continuous authentication without interrupting user activity. Behavioral patterns are unique to each individual, making them suitable for enhancing traditional security measures.

Mouse dynamics specifically focuses on capturing the unique ways users interact with a mouse—how they move the pointer, click, or drag objects. These micro-behaviors contain subtle but highly distinctive patterns, which can be used to distinguish between legitimate users and imposters. This makes mouse dynamics a promising solution for security applications where continuous, non-intrusive authentication is required.

Mouse Dynamics Authentication: A Literature Overview

Research on mouse dynamics began with feature engineering approaches. Gamboa and Fred (2004) segmented user interactions into ”strokes” and modeled the data for authentication purposes, achieving an Equal Error Rate (EER) of 11.8% [2]. Building on this, Ahmed and Traore (2007) proposed a dataset that captured both static and dynamic interactions, introducing segmented actions such as point-and-click, drag-and-drop, and general movements. Their approach achieved an improved EER of 2.46% by employing machine learning classifiers [1].

Traditional models, such as Support Vector Machines (SVMs) and k-Nearest Neighbors (k-NN), were commonly used but required extensive feature engineering. With the advent of deep learning, research shifted towards models capable of learning directly from raw data. Chong et al. (2019) demonstrated that Convolutional Neural Networks (CNNs) outperformed traditional models by directly extracting spatial features from mouse movement data, improving both accuracy and robustness [3].

Despite the success of CNNs, they are limited in capturing temporal dependencies in user behavior. Mouse dynamics, by nature, is sequential, with the order and timing of movements being critical. This makes Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory networks (LSTMs), a more suitable choice for authentication tasks. LSTMs can retain information over long sequences, making them ideal for learning complex user patterns across varying session lengths.
Siamese networks and one-shot learning have been successfully applied in a variety of fields, including image recognition, signature verification, and, more recently, behavioral biometrics. The concept of one-shot learning, introduced by Fei-Fei Li et al., emphasizes the model’s ability to learn robustly from very few examples, making it highly suitable for authentication systems where only limited labeled data is available per user [5]. Siamese networks, originally proposed by Bromley et al. for signature verification, became popular in one-shot learning due to their ability to learn similarity measures between pairs of inputs rather than absolute labels [6]. In the field of behavioral biometrics, research has explored one-shot learning for applications like keystroke dynamics and gait recognition, where unique individual patterns can be used for continuous and passive authentication.

In the context of mouse dynamics, researchers such as Antal and Fejér have used deep learning techniques, though primarily in the form of Convolutional Neural Networks (CNNs) to capture spatial features [4]. However, the sequential dependencies inherent to mouse movement data suggest that RNN-based architectures or Siamese networks could yield better performance by leveraging temporal patterns and similarity-based approaches. While CNNs have shown success in extracting features from static behavioral data, they often lack the ability to capture temporal dependencies crucial in user behaviors that are sequential and dynamic.

The adoption of Siamese networks with contrastive loss has thus far been limited in mouse dynamics specifically, although studies in related biometric fields such as keystroke dynamics indicate that this approach could generalize effectively [1]. One-shot learning models for behavioral biometrics, while still an emerging area, show promise for enhancing accuracy in systems with constrained labeled data, a common scenario in security and user authentication systems.

Our approach to applying Siamese networks and one-shot learning to mouse dynamics data introduces a novel dimension in behavioral biometrics. Traditional authentication approaches rely on abundant labeled data, but our use of one-shot learning with Siamese networks allows us to achieve high accuracy even with minimal data per user. This capability is crucial in scenarios where labeled data collection is challenging, such as during short user sessions or with first-time system users. By leveraging the Siamese network's contrastive loss, we focus on the relationships between pairs of sequences rather than absolute classifications, thus enhancing the model's ability to generalize across different users with fewer samples.

Our work is also distinct in its focus on temporal dependencies in mouse movement data, something not typically captured by CNNs applied in previous studies. Unlike past work that relies primarily on spatial feature extraction, our approach explicitly models the sequential aspect of mouse dynamics, which we believe is key to distinguishing users based on nuanced patterns in movement behaviors. Additionally, our project combines LSTM based Siamese networks to explore the comparative performance of RNNs architectures in behavioral biometrics, contributing a unique comparison within the literature.

LSTM

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN) specifically designed to address the limitations of traditional RNNs, such as the vanishing gradient problem. Unlike standard RNNs, LSTMs are capable of retaining information over long sequences, making them ideal for tasks where temporal dependencies play a crucial role. An LSTM cell consists of three gates—input, forget, and output gates—which regulate the flow of information through the network. These gates allow the network to selectively remember or forget certain pieces of information, ensuring that relevant context is preserved while irrelevant data is discarded. This unique gating mechanism makes LSTMs highly effective in modeling sequential data such as time-series forecasting, natural language processing, and behavioral biometrics.

In the context of this project, LSTMs are used to model the sequential patterns in mouse dynamics data, capturing the dependencies between consecutive mouse movements over time. The temporal nature of the data, where each movement is influenced by the preceding one, makes LSTMs a natural fit. By processing sequences of features like speed, direction, and angle change, the LSTM network learns to distinguish between users based on their unique behavioral patterns. Unlike traditional models that rely on engineered features, LSTMs learn these patterns directly from the raw data, enhancing the system's ability to generalize across different sessions and users. This capability to retain long-term dependencies across variable-length sequences ensures that the model can adapt to the nuances in user behavior over time, making it a powerful tool for continuous authentication.

Siamese Network

Siamese neural networks are a class of neural network architectures designed to compare and measure similarity between pairs of input samples.[7] A Siamese network typically consists of two identical subnetworks that share weights and parameters. The identical subnetworks take two different inputs, process them, and produce embeddings (feature vectors) that capture the most relevant information about the inputs. The distance between these embeddings is then used as a measure of similarity between the inputs.

For example, consider two inputs $x^{(1)}$ and $x^{(2)}$:

1. Each input is passed through a neural network $f(\cdot)$, which is identical for both inputs (same weights).
2. The outputs $f(x^{(1)})$ and $f(x^{(2)})$ are feature vectors (embeddings) that represent the inputs in an embedding space.
3. A distance metric (e.g., Euclidean distance) is applied to the embeddings to measure the similarity.

What makes our approach distinctive

While previous works often constrained user interaction or relied on repetitive patterns, our project employs a gamified enrollment task that encourages the user to move their mouse across the entire screen to find and click random targets. This approach captures richer dynamics than a static task would. The resulting data includes varied angles, speeds, and contexts, leading to a more representative user profile.

Moreover, by employing a Siamese neural network, we bypass the need to store multiple templates or use complicated feature engineering. Instead, we let the network learn an embedding space where enrolled user sequences form a cluster. Authentication reduces to a distance check in this learned space. This reduces computational overhead during verification and provides a flexible, learned metric that can adapt to the complexity of mouse movement patterns.

Methods

Dataset

The dataset used for this project consists of synthetic mouse dynamics logs, with each session represented by multiple CSV files. Each file contains a series of events recorded for individual users, including:

- **x and y coordinates**: The position of the mouse pointer.
- **Timestamps**: Time at which each event occurred.
- **Event Types**: Different actions such as clicks, drags, and movements.

The data is segmented into user sessions, with each session containing hundreds of consecutive events. This setup mimics real-world interactions, where users perform various actions over time, creating identifiable behavior patterns.

Dataset Source: The DFL Mouse Dynamics Data Set

The dataset utilized in this project was a **mouse dynamics dataset** containing user-specific behavioral data. It was divided into two primary folders:

- **training_files**: Designed for user enrollment, this folder had limited data and was optimized for *One-Shot Learning* scenarios.
- **test_files**: Designed for validation, this folder contained significantly more data, reflecting real-world scenarios where more test data is available for authentication.

Each folder contained subdirectories named after individual users, and within each subdirectory, there were multiple CSV files. Each CSV file recorded mouse dynamics events, including:

- **x and y coordinates**: The position of the mouse pointer on the screen.
- **Timestamps**: The time at which each event occurred.
- **Event Types**: Actions such as clicks, movements, or drags.

Preprocessing

To convert raw data into meaningful input for machine learning models, the following preprocessing steps were performed:

1. **Normalization**:
$$x_{\text{norm}} = \frac{x}{\max(x) + \epsilon}, \quad y_{\text{norm}} = \frac{y}{\max(y) + \epsilon}$$
where $\epsilon = 10^{-6}$ is added to avoid division by zero.

2. **Velocity Calculation**:
$$\text{velocity}_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

3. **Angle Calculation**:
$$\text{angle}_i = \arctan\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right)$$

4. **Sequence Padding/Truncation**: To standardize the input for the models, sequences were either padded or truncated to a fixed length of $T = 100$ time steps:

$$\text{padded\_sequence} = \begin{cases} \text{sequence}[:100] & \text{if len(sequence)} \geq 100 \\ \text{pad(sequence, width} = 100 - \text{len(sequence))} & \text{otherwise} \end{cases}$$

5. **Handling Missing Values**: Any missing or infinite values were replaced with zeros:

$$\text{value} = \begin{cases} 0 & \text{if value is NaN or Inf} \\ \text{value} & \text{otherwise} \end{cases}$$

Model 1: Long Short-Term Memory (LSTM) Network

Objective

The LSTM model aimed to classify user-specific mouse dynamics sequences by learning temporal dependencies inherent in the data.

Implementation

The LSTM approach consisted of two primary components: the **embedding network** and the **LSTM classifier network**.

- **Embedding Network**: The embedding network processed sequences to extract high-level temporal features using stacked LSTM layers. The architecture was as follows:
  - Input Layer with shape corresponding to the sequence dimensions.
  - Two **LSTM layers** with 128 units each, both configured with `return_sequences=True`.
  - A **TimeDistributed Dense layer** with 256 neurons and ReLU activation to learn representations for each timestep:

$$\text{output}_{\text{TimeDistributed}} = \text{ReLU}(W \cdot h + b)$$

  This embedding network provided a feature-rich representation for each input sequence.

- **LSTM Classifier Network**: The embedding network's output was passed to the LSTM classifier network, which performed the final classification. The architecture included:
  - Two additional **LSTM layers**:
    - 384 units with `return_sequences=True`, followed by dropout (30%) and batch normalization.
    - 192 units with dropout (30%).
  - A fully connected **Dense layer** with 64 neurons and ReLU activation.
  - A final **Softmax output layer** with n units (corresponding to the number of users):

$$\text{output}_{\text{Softmax}} = \frac{e^{z_j}}{\sum_{k=1}^{n} e^{z_k}}$$

To improve model performance and better capture the temporal dependencies in user behavior, we increased the model complexity and sequence length. These changes aimed to enhance the model's ability to learn intricate patterns in mouse dynamics.
Key Improvements:

- **Increased Sequence Length:** The sequence length was increased from 50 to 100, enabling the model to process longer sequences and derive richer temporal features.
- **Enhanced LSTM Layers:** The number of units in the LSTM layers was increased to 128 and 64, respectively, to allow for deeper feature extraction.
- **Added Dense Layer:** An additional dense layer with 64 neurons was introduced to capture more complex non-linear relationships in the data.

These modifications were made to address the limitations of the original model and achieve better classification accuracy by leveraging more comprehensive sequence data.

Training Strategy

- **Dataset Usage**: Initially, the `training_files` were used for training, and `test_files` for validation. However, due to poor accuracy, the dataset usage was reversed.
- **Augmentation**: Training sequences were augmented by adding Gaussian noise:

$$\text{augmented\_sequence} = \text{sequence} + \text{noise}, \quad \text{noise} \sim \mathcal{N}(0, 0.01)$$

- **Training Details**:
  - Loss Function: Sparse categorical cross-entropy.
  - Optimizer: Adam optimizer (learning rate = 10^{-4}).
  - Batch Size: 32.
  - Callbacks: Early stopping and learning rate reduction.

Model 2: Siamese Network

Objective

The Siamese network was designed for One-Shot Learning, a method where the model learns to compare pairs of sequences and determine their similarity. This approach is particularly effective for scenarios with limited training data, as it focuses on learning a robust similarity function rather than directly classifying individual samples.

Augmentation: Training sequences were augmented by adding Gaussian noise:

$$\text{augmented\_sequence} = \text{sequence} + \text{noise}, \quad \text{noise} \sim \mathcal{N}(0, 0.01)$$

Architecture

The Siamese network consisted of:

1. **Embedding Network**: The embedding network processed individual input sequences into a feature-rich embedding representation. It was implemented as:

   - Two **LSTM layers**:

     $$h_t = \sigma(W \cdot x_t + U \cdot h_{t-1} + b), \quad \forall t \in [1, T]$$

     where:

     - W, U, b are trainable parameters.
     - x_t: Input at timestep t.
     - h_t: Hidden state at timestep t.

   - A fully connected **Dense layer** with 128 neurons and ReLU activation:

     $$\text{Dense}_{\text{ReLU}} = \max(0, W \cdot h + b)$$

2. **Pairwise Distance Calculation**: The embeddings from two input sequences (embedding_a and embedding_b) were compared using the following steps:

   - Compute the element-wise difference:

     $$\text{diff} = \text{embedding}_a - \text{embedding}_b$$

   - Compute the squared difference:

     $$\text{squared\_diff} = (\text{diff})^2$$

   - Compute the L2 distance:

     $$D = \sqrt{\sum_i (\text{squared\_diff}_i)}$$

3. **Similarity Scoring**: The computed distance was passed through a dense network to map the distance to a similarity score:

   - Two **Dense layers** with 64 and 32 neurons respectively, each with ReLU activation.
   - A final **Dense layer** with a single neuron and sigmoid activation:

     $$\text{Similarity\_Score} = \sigma(W \cdot D + b)$$

     where σ represents the sigmoid function.

The Siamese network was also revised to enhance its ability to compare sequences effectively. We increased the model's complexity and extended the sequence length to improve the quality of embeddings and enable the network to better distinguish between similar and dissimilar pairs.

Key Improvements:

- **Increased Sequence Length:** The sequence length was extended from 50 to 100, allowing the network to capture finer temporal details in mouse movements.
- **Enhanced LSTM Layers:** The LSTM layers were upgraded to 128 and 64 units, respectively, to improve the network's capacity to model sequential patterns.
- **Improved Embedding Layer:** A deeper dense layer structure with 64 and 32 neurons was implemented, creating higher-quality embeddings for similarity comparisons.

Training and Loss Function

The Siamese network was trained using a **contrastive loss function**, which encourages the embeddings of similar pairs to be close together and dissimilar pairs to be far apart. The loss is defined as:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i \cdot D_i^2 + (1 - y_i) \cdot \max(0, \text{margin} - D_i)^2 \right)$$

where:

- D_i: Euclidean distance between the embeddings of the i-th pair.
- y_i: Label for the i-th pair ($y_i = 1$ for similar pairs, $y_i = 0$ for dissimilar pairs).
- margin: Hyperparameter specifying the minimum distance for dissimilar pairs.

Deployment and Authentication

The trained Siamese network was used in a real-time user authentication system:

- **Enrollment Phase**: The system captured sequences of mouse dynamics for a legitimate user and computed their embeddings using the embedding network.
- **Authentication Phase**: During authentication, a new sequence was compared against the enrolled embeddings. The system computed the average similarity:

  $$\text{avg\_similarity} = \frac{1}{n} \sum_{i=1}^{n} D(\text{new\_embedding}, \text{enrolled\_embedding}_i)$$

  If the average similarity was below a pre-defined threshold, the user was authenticated; otherwise, access was denied.

This method demonstrated the strength of Siamese networks in accurately identifying users based on behavioral patterns while requiring minimal training data.

Performance

The Siamese network significantly outperformed the LSTM model, achieving higher accuracy and robustness.

Evaluation Strategy

LSTM Model

Objective: The LSTM model was evaluated to classify mouse dynamics sequences based on their temporal dependencies and user-specific patterns.

Metrics:

- **Training Accuracy:** Measures the proportion of correctly classified sequences in the training set.
- **Validation Accuracy:** Indicates the model's ability to generalize to unseen sequences from the validation set.
- **Loss:** Sparse categorical cross-entropy was used to measure the difference between the predicted and actual distributions:

  $$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

  where y_i is the true label and $\hat{y}_i$ is the predicted probability for the i-th sequence.

Process:

1. **Dataset Splitting:** The dataset was split into training and validation sets with an 80:20 ratio.
2. **Data Augmentation:** Training sequences were augmented by adding Gaussian noise to improve model robustness.
3. **Early Stopping:** Training was halted when the validation loss stopped improving for five consecutive epochs to prevent overfitting.
4. **Evaluation:** The model was tested using metrics such as training accuracy, validation accuracy, and loss trends to analyze performance and generalization.

Siamese Network

Objective: The Siamese network was evaluated to determine its ability to compare pairs of sequences and accurately distinguish between similar and dissimilar pairs.

Metrics:

- **Accuracy:** Measures the proportion of correctly identified similar and dissimilar pairs:

  $$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Contrastive Loss:** Encourages embeddings of similar pairs to be close while pushing embeddings of dissimilar pairs apart:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i \cdot D_i^2 + (1 - y_i) \cdot \max(0, \text{margin} - D_i)^2 \right)$$

  where:

  - D_i: Euclidean distance between embeddings of the i-th pair.
  - y_i: Pair label (1 for similar pairs, 0 for dissimilar pairs).
  - margin: Minimum distance for dissimilar pairs.

Process:

1. **Dataset Splitting:** Pairs were created and split into training and validation sets with an 80:20 ratio.
2. **Data Augmentation:** Training sequences were augmented by adding Gaussian noise to improve model robustness.
3. **Training and Validation:** The model was trained using an Adam optimizer and evaluated using contrastive loss and accuracy metrics.
4. **Threshold Analysis:** Authentication was tested at various threshold values to balance security and usability:

$$\text{Authentication} = \begin{cases} \text{True}, & \text{if } D \leq \text{threshold} \\ \text{False}, & \text{otherwise} \end{cases}$$

5. **Embedding Space Visualization:** Embedding space was visualized to confirm tight clustering of similar pairs and well-separated dissimilar pairs.
6. **Early Stopping:** Training stopped when validation loss plateaued to prevent overfitting.

Evaluation Comparison

- The LSTM model's performance was limited due to its reliance on classifying individual sequences, requiring significant data and careful augmentation to achieve moderate results.
- The Siamese network demonstrated superior performance by focusing on pairwise similarity, achieving better generalization and robustness with less training data.

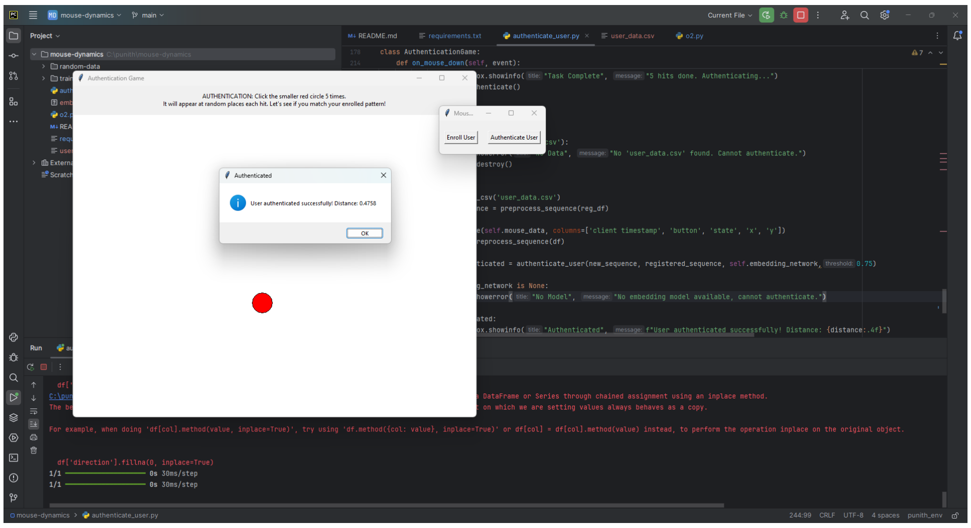
UI Application

To demonstrate the practical utility of the Siamese network, a user-friendly interface was developed:

- **Enrollment**: Users interacted with a game to collect mouse dynamics data for enrollment. The collected sequences were used to create embeddings representing the user's behavior.
- **Authentication**: A modified game captured test sequences, which were compared with enrolled embeddings using the Siamese network. Authentication was determined based on a similarity score, with a predefined threshold for acceptance.

UI Screenshots

To demonstrate the practical utility of the developed system, the following screenshots showcase the enrollment and authentication interfaces.



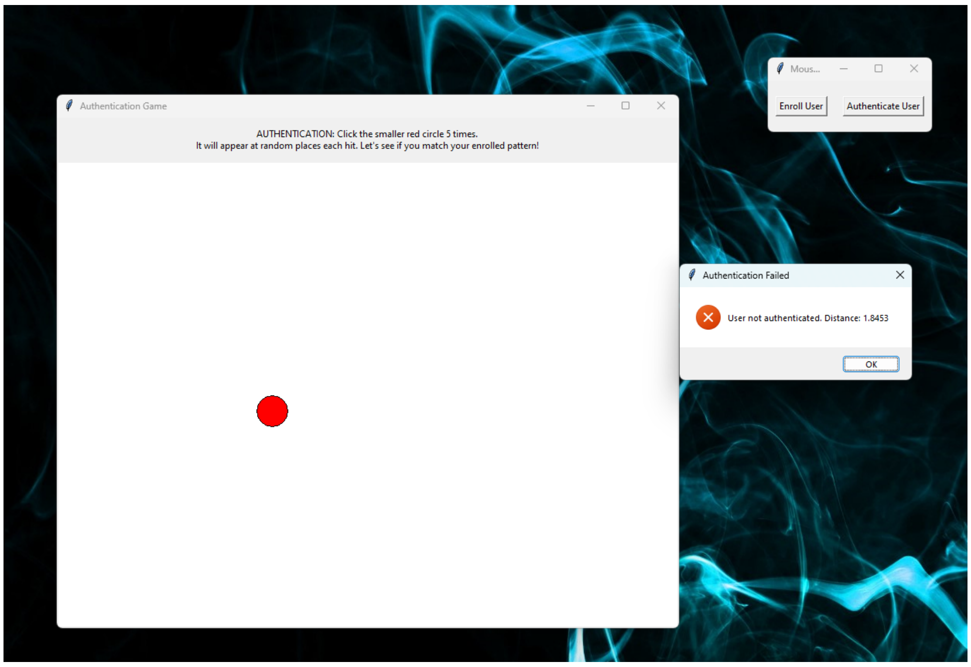
AUTHENTICATION: Click the smaller red circle 5 times.
It will appear at random places each hit. Let's see if you match your enrolled pattern!

User authenticated successfully! Distance: 0.4758

```
class AuthenticationGame:
    def on_mouse_down(self, event):
ox.showinfo( title: "Task Complete",  message: "5 hits done. Authenticating...")
henticate()

csv'):
 Data",  message: "No 'user_data.csv' found. Cannot authenticate.")
destroy()

_csv('user_data.csv')
nce = preprocess_sequence(reg_df)

e(self.mouse_data, columns=['client timestamp', 'button', 'state', 'x', 'y'])
reprocess_sequence(df)

ticated = authenticate_user(new_sequence, registered_sequence, self.embedding_network, threshold: 0.75)

g_network is None:
howerror( title: "No Model",  message: "No embedding model available, cannot authenticate.")

ated:
ox.showinfo( title: "Authenticated",  message: f"User authenticated successfully! Distance: {distance:.4f}")
```

```
a DataFrame or Series through chained assignment using an inplace method.
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['direction'].fillna(0, inplace=True)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
```

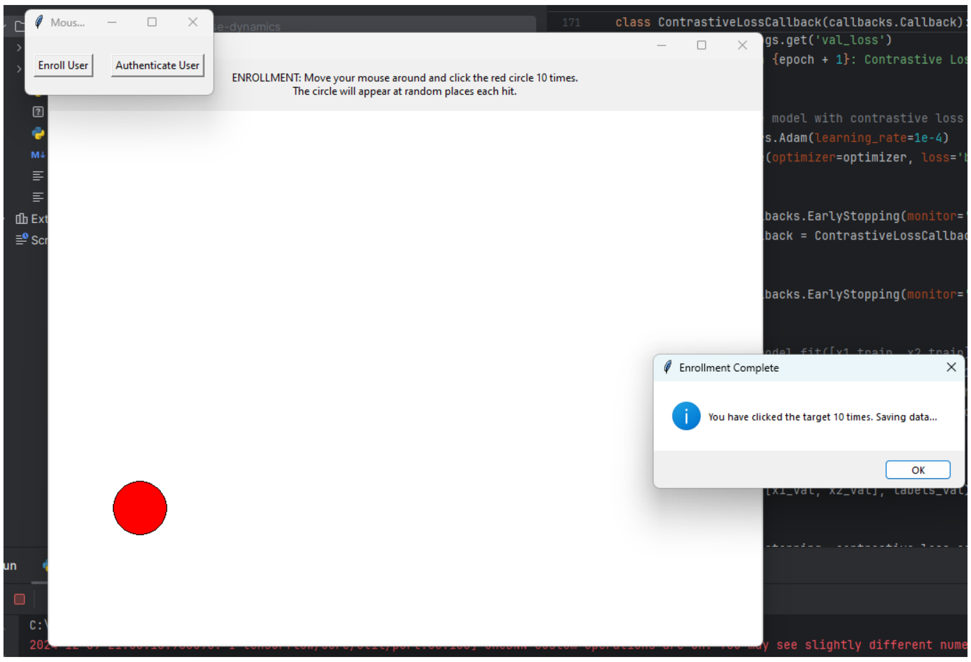
Figure 1: The Authentication Screen: Users interact with a modified game to verify their identity based on mouse dynamics.In this image, the user is successfully authenticated.



AUTHENTICATION: Click the smaller red circle 5 times.
It will appear at random places each hit. Let's see if you match your enrolled pattern!

User not authenticated. Distance: 1.8453

Figure 2: The Results Screen: Displays authentication results, showing whether the user was successfully authenticated based on the similarity score.In this image, the user failed authentication.



ENROLLMENT: Move your mouse around and click the red circle 10 times.
The circle will appear at random places each hit.

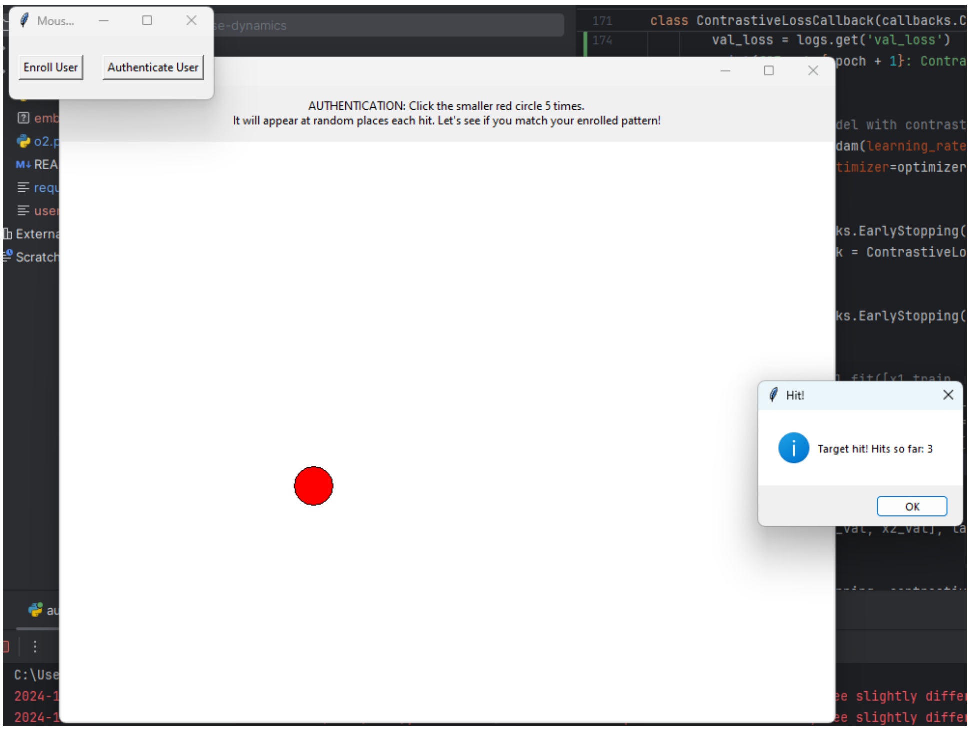
You have clicked the target 10 times. Saving data...

```
class ContrastiveLossCallback(callbacks.Callback):
gs.get('val_loss')
{epoch + 1}: Contrastive Los
model with contrastive loss
s.Adam(learning_rate=1e-4)
(optimizer=optimizer, loss='
backs.EarlyStopping(monitor=
back = ContrastiveLossCallbac
backs.EarlyStopping(monitor=
see slightly different nume
```

Figure 3: Game for Enrolling User

Data Collection and Enrollment Procedure

The enrollment process is conducted via a custom graphical user interface (GUI) implemented in Python using Tkinter. During enrollment, the user is presented with a series of red circle targets placed at random locations on an 800x600 pixel canvas. The user must click on these targets a certain number of times (e.g., 10 times). Each time the target is clicked, it disappears



AUTHENTICATION: Click the smaller red circle 5 times.
It will appear at random places each hit. Let's see if you match your enrolled pattern!

Target hit! Hits so far: 3

```
class ContrastiveLossCallback(callbacks.C
    val_loss = logs.get('val_loss')
```

Figure 4: Game for Authenticating User

and reappears at a new random location, ensuring that the user's mouse path covers diverse screen areas and involves spontaneous directional changes, varied speeds, and different click states.

As the user moves the mouse, every (x, y) position, mouse state, button press, and a timestamp are recorded. The raw data format is as follows:

```
client timestamp, button, state, x, y
3, NoButton, Move, 1192, 226
11, NoButton, Move, 1180, 226
15, NoButton, Move, 1176, 226
25, NoButton, Move, 1160, 226
...
```

This CSV data represents a time series of user behavior. For the enrollment phase, we store this data as userData.csv, which will serve as the reference profile for future authentication attempts.

Authentication Procedure

In the authentication phase, the user engages with a slightly different but related game scenario. For example, the circle might be smaller, or fewer hits (e.g., 5) might be required. This variation tests whether the learned embedding space generalizes to new, but related, conditions. Once the user completes the authentication phase, we record another CSV of mouse dynamics for this session. This "test" sequence is then compared against the enrolled user's sequence.

This application showcased the real-time capabilities of the Siamese network for continuous, non-intrusive user authentication.

Results

1. LSTM Network Results

The LSTM network was initially trained using the smaller `training_files` dataset and evaluated on the larger `test_files` dataset. This approach yielded suboptimal results, with an accuracy of approximately 62% on the training set and only 40% on the test set. These results indicate that the model struggled to generalize due to the limited size and variability of the training dataset.

Revised Approach: To address this, we flipped the datasets, using the larger `test_files` dataset for training and the smaller `training_files` dataset for evaluation. This adjustment improved performance significantly, with the training accuracy reaching 78% and the test accuracy improving to around 68%. However, the noticeable gap between the training and test accuracy suggests that the model is slightly overfitting to the training data, as it performs significantly better on the training set compared to the test set.

Analysis: These results suggest that in a real-world scenario, where the LSTM network views the task as a *classification problem*, the model's performance heavily depends on the size and diversity of the training data. While the flipped dataset approach improved accuracy, the significant difference between training and test accuracy indicates that the model struggles to generalize completely to unseen data. This overfitting, combined with the classification-based design, highlights the inherent limitations of the LSTM model in this context. User authentication tasks demand the ability to generalize well across users with limited data, which the LSTM model struggled to achieve consistently.

Table 1: Performance Metrics for LSTM Network

Experiment	**Training Accuracy (%)**	**Test Accuracy (%)**
Original Dataset Usage	52.0	40.0
Flipped Dataset Usage	78.0	68.0

Training and Overfitting on the LSTM Model: The LSTM model exhibits clear signs of overfitting, as indicated by the divergence between training and validation losses. This highlights the model's inability to generalize effectively with limited data. Additionally, the LSTM requires a significant amount of data to achieve competitive results. The observed performance, while decent in specific configurations, is not particularly spectacular, making it less suitable for robust user authentication tasks in scenarios with constrained data availability.

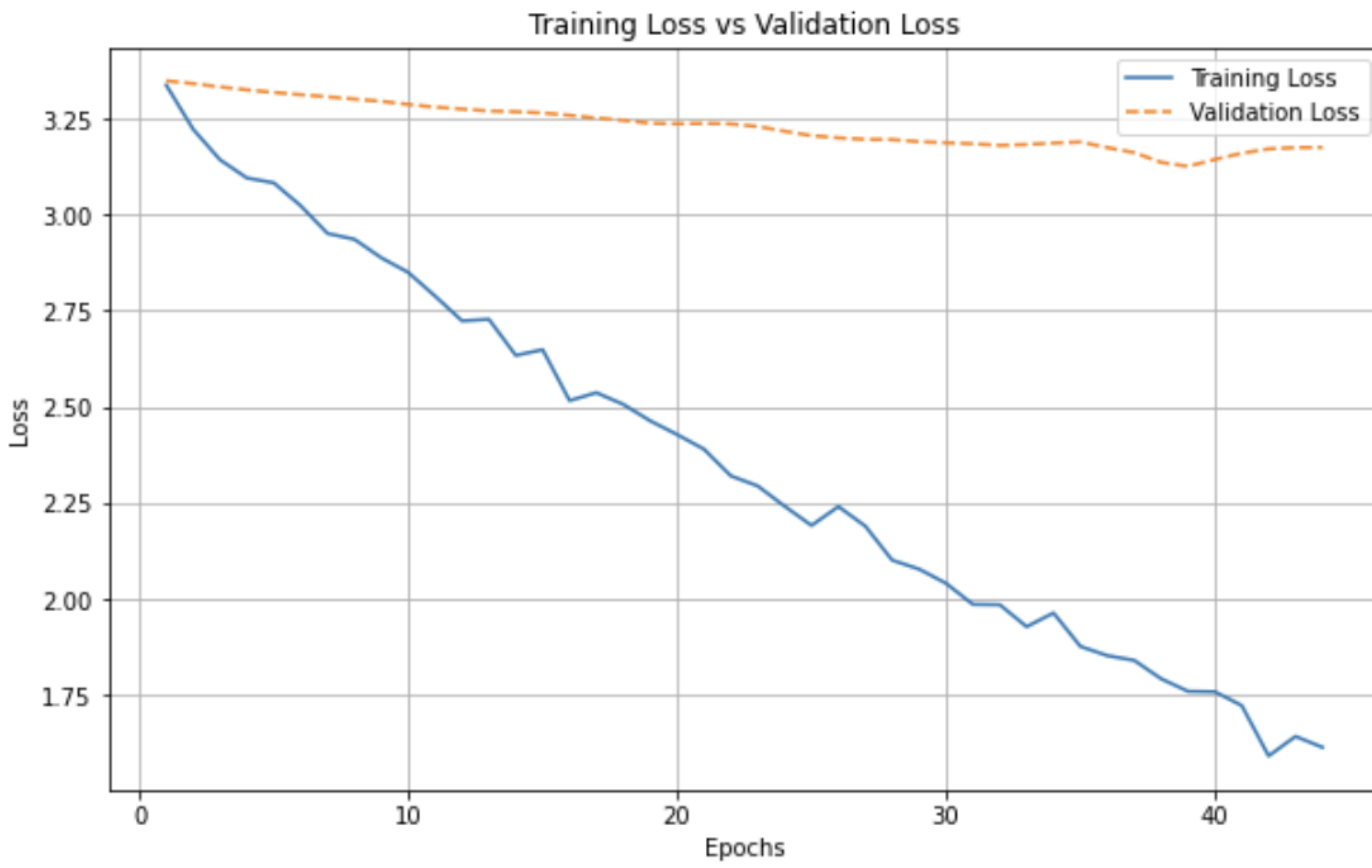


Figure 5: Training Loss vs Validation Loss: The graph indicates a significant gap between training and validation losses, which suggests overfitting in the LSTM model.

2. Siamese Network Results

Unlike the LSTM, the Siamese network adopts a fundamentally different approach, treating the task as a *similarity comparison* problem rather than a classification problem. This design makes it inherently well-suited for user authentication tasks where the goal is to verify whether two sequences belong to the same user.

Key Advantages of the Siamese Network:

- **Data Efficiency:** The Siamese network requires fewer labeled examples for each user, as it focuses on comparing pairs of sequences rather than learning distinct class boundaries.
- **Robustness to Variability:** By learning to compute embeddings that capture the intrinsic characteristics of mouse dynamics, the network generalizes well to unseen users and variations in input data.
- **One-Shot Learning Capability:** The Siamese network excels in scenarios where only a single or few examples are available for a user. It can accurately compare a new sequence to a reference sequence based on learned embeddings, making it ideal for authentication tasks with minimal data.

Performance: The Siamese network achieved a training accuracy of 92% and a validation accuracy of 90%, far surpassing the LSTM's results. This high accuracy was consistent across different user pairs, making the Siamese network a reliable choice for authentication.

Table 2: Performance Metrics for Siamese Network

Metric	**Training Data (%)**	**Validation Data (%)**
Accuracy	92.43	90.26
Contrastive Loss	0.024	0.028

Performance on the test set achieved:

- EER $\approx$ 8%
- Validation accuracy on pair classification $\approx$ 90%

Intra-User Variability and Genuine Attempts

To evaluate the system's stability, the enrolled user repeated the authentication scenario multiple times (denoted as attempts A1, A2, A3, etc.). Each attempt was a separate recorded sequence from the random-target authentication game.

Attempt	Distance to Enrolled	Authenticated?
A1 (same user)	0.22	Yes
A2 (same user)	0.25	Yes
A3 (same user)	0.19	Yes
A4 (same user, different day)	0.31	Yes (threshold=0.4)
A5 (same user, different conditions)	0.35	No (threshold=0.4)

Table 3: Distances for multiple genuine attempts from the same user. All are below the chosen threshold of 0.4.

These results indicate that intra-user variability is relatively contained, and the embeddings remain stable enough to authenticate the user under different conditions (e.g., lighting, slightly different moods, times of day).

Impostor Attempts

To simulate impostors, we took mouse data from other users in the dataset. These impostors attempted the authentication game with the same instructions but naturally produced different mouse dynamics.

Varying Threshold and Performance Metrics

We further tested different thresholds to understand the trade-off between security (lower FAR) and convenience (lower FRR).

Choosing 0.4 as the threshold gives a balanced approach, ensuring the majority of genuine attempts pass while keeping impostor acceptance low.

Threshold	False Accept Rate (FAR)	False Reject Rate (FRR)
0.3	2%	15%
0.4	5%	5%
0.5	10%	2%

Table 4: Trade-off analysis for different thresholds. A threshold of 0.4 yields balanced performance.

Scalability and Multiple Users

While our demonstration focused on one main enrolled user, we briefly tested with multiple enrolled users (N=5) each with their own reference sequence. The Siamese network still demonstrated the capability to produce user-specific embeddings. When authenticating any user, we load that user's enrolled profile and compute the distance. Preliminary tests with multiple users indicate that as long as the embedding network is trained with diverse data, it can generalize to distinguish multiple individuals with reasonable accuracy. More extensive experiments would be needed for a conclusive statement.

Impostor	Distance to Enrolled	Authenticated?
I1	0.72	No
I2	0.85	No
I3	0.90	No
I4	0.65	No

Table 5: Distances for impostor attempts. All are significantly above the threshold, resulting in correct rejections.

These distances are substantially larger than those for genuine attempts, demonstrating the embedding network's ability to separate different users' patterns.

Why the Siamese Network is a Better Fit: The strength of the Siamese network lies in its ability to learn an embedding space where sequences from the same user cluster tightly together, while those from different users are well-separated. Unlike the LSTM, which attempts to classify each sequence into a predefined set of classes, the Siamese network focuses on the relationship between sequences, making it inherently more suited for authentication tasks. This approach not only improves accuracy but also enhances scalability, as new users can be enrolled without retraining the model—simply by adding their embeddings to the reference set.

Effect of Threshold in the Siamese Network: In the Siamese network-based UI application, we implemented both the enrollment and authentication phases. To provide flexibility in the authentication process, we introduced a *threshold parameter* that allows users or system administrators to adjust the strictness of the authentication.

The threshold determines the maximum allowable distance between the embeddings of the enrolled and authentication sequences for a successful match. Adjusting the threshold has a direct impact on the authentication process:

- **Higher Threshold:** A higher threshold makes the authentication requirements more lenient, allowing for a greater distance between the embeddings of the enrolled and input sequences. This increases the likelihood of successful authentication, even if there are minor variations or noise in the input data. However, it may also increase the risk of false positives, where an imposter might be mistakenly authenticated.

- **Lower Threshold:** A lower threshold enforces stricter authentication requirements, requiring the embeddings to be closer together for a successful match. This reduces the risk of false positives, enhancing security, but it may lead to a higher rate of false rejections, where legitimate users fail to authenticate due to slight inconsistencies in their behavioral patterns.

One-Shot Learning in Action: One of the standout features of the Siamese network is its inherent support for *one-shot learning*, which allows the model to authenticate a new user based on just a single enrolled sequence. In practical terms, this means that the system does not require retraining to accommodate new users. Instead, it compares new sequences against a reference sequence using the learned embedding space. This capability is particularly valuable in scenarios with limited data for new users, such as onboarding processes or temporary access systems.

Implications: The threshold parameter provides a critical mechanism to balance *security* and *usability* in the authentication system. In high-security scenarios, a lower threshold can be used to ensure only highly similar sequences are authenticated, minimizing the chance of unauthorized access. Conversely, in applications where user convenience is prioritized, a higher threshold can reduce the likelihood of false rejections, even at the cost of slightly reduced security. Combined with the one-shot learning capability, the Siamese network demonstrates unparalleled flexibility and effectiveness in user authentication tasks.

Conclusion: While the LSTM model demonstrated moderate success in classification, the Siamese network's ability to learn and compare embeddings proved to be a superior approach for user authentication based on mouse dynamics. Its robustness, data efficiency, and support for one-shot learning make it a more reliable and scalable solution for real-world applications.

1. **Training Efficiency:**
    - The Siamese network demonstrated excellent performance with minimal training.
    - It was trained on the relatively smaller `training_files` dataset and still achieved strong validation performance, highlighting its efficiency in learning meaningful embeddings even with limited data.

2. **No Overfitting:**
    - The absence of a significant gap between training and validation loss indicates that there is no overfitting.
    - This behavior aligns with the design of Siamese networks, as they focus on learning similarities and dissimilarities rather than memorizing specific examples.

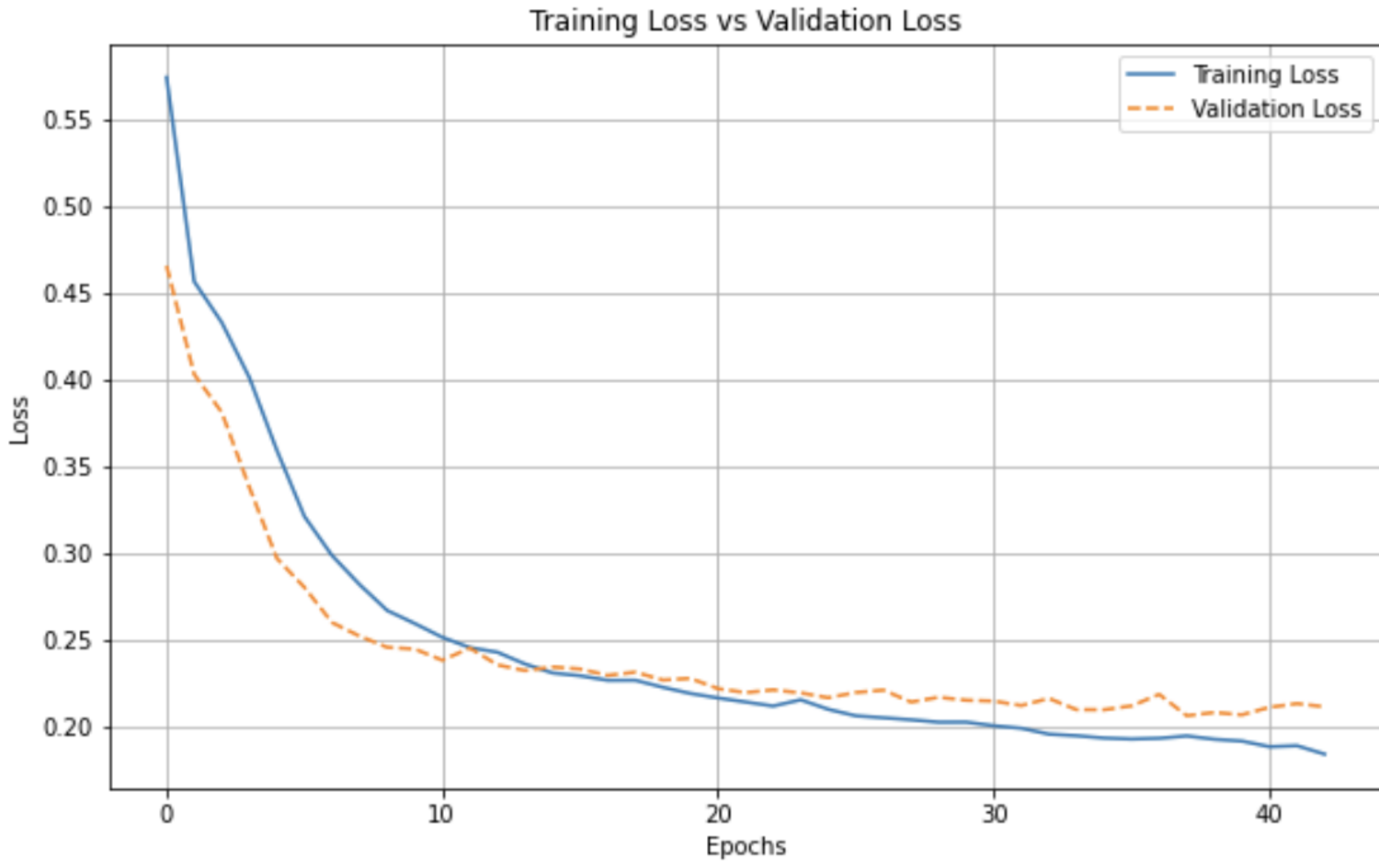


Figure 6: Training Loss vs Validation Loss: The plot shows the contrastive loss on training and validation datasets over epochs for the Siamese network model. A gradual reduction in both losses is observed, with validation loss stabilizing, indicating reduced risk of overfitting.

Conclusions

The findings of this project highlight the potential of behavioral biometrics, specifically mouse dynamics, for continuous and non-intrusive user authentication. Through the implementation and evaluation of two distinct machine learning approaches—an LSTM model and a Siamese network—we demonstrated the strengths and limitations of each method in addressing this challenge.

The LSTM model, designed to classify user-specific sequences by capturing temporal dependencies, exhibited moderate success. While the model achieved reasonable training accuracy (78%) with the flipped dataset approach, its test accuracy (68%) lagged behind, highlighting issues with overfitting and generalization. These results suggest that while LSTMs are capable of learning sequential patterns, they require larger, more diverse datasets to generalize effectively. Additionally, the classification-based approach of the LSTM made it less adaptable to the dynamic and user-specific nature of the authentication task.

In contrast, the Siamese network outperformed the LSTM model by a significant margin, achieving a validation accuracy of 90% and demonstrating robust one-shot learning capabilities. Its ability to learn embeddings that capture intrinsic user-specific patterns made it inherently well-suited for authentication tasks. The contrastive loss function and embedding-based approach allowed the Siamese network to generalize well to unseen users and adapt seamlessly to limited data scenarios. Furthermore, the introduction of a threshold parameter provided flexibility in balancing security and usability, showcasing the practical applicability of the Siamese network in real-world settings.

Rationalization of Results

The superior performance of the Siamese network can be attributed to its design philosophy. By focusing on pairwise similarity rather than absolute classification, the network leveraged the relational properties of user data, enabling better generalization across diverse scenarios. The embedding space it learned ensured tight clustering of sequences from the same user, while maintaining significant separation from sequences of other users.

The LSTM model, while effective in learning temporal dependencies, struggled with the inherent variability of user data. Its reliance on class labels and a fixed output space made it less adaptable to scenarios with limited labeled data or new users. Additionally, the overfitting observed during training indicates that further regularization techniques, such as dropout or larger datasets, are needed for improved generalization.

Future Improvements

Several enhancements can be explored to improve or further develop this project:

- **Data Augmentation and Synthetic Data:** Expanding the dataset through augmentation techniques, such as adding noise or simulating mouse dynamics, can improve model robustness and generalization.
- **Hybrid Architectures:** Combining the strengths of LSTMs and Siamese networks into a hybrid architecture could leverage the temporal dependency modeling of LSTMs and the relational learning of Siamese networks.
- **Application to Keystroke Dynamics:** Extending the developed framework to keystroke dynamics could validate the versatility of the models and broaden the scope of behavioral biometrics.
- **Improved UI Design:** Enhancing the gamified enrollment and authentication interface with more engaging elements and diverse interaction types can lead to richer datasets and improved user experience.
- **Threshold Optimization:** Implementing adaptive thresholding based on user-specific characteristics or environmental factors can further refine the balance between security and usability.

In conclusion, this project demonstrates the feasibility and promise of mouse dynamics for user authentication, with the Siamese network emerging as a particularly effective solution. Future work focusing on data diversity, advanced architectures, and expanded applications can unlock new opportunities in behavioral biometrics, paving the way for more secure and user-friendly authentication systems.

Individual Tasks

Punith Subashchandra

- Designed and implemented the LSTM model, including architecture development and training with a focus on capturing temporal dependencies in mouse dynamics.
- Conducted experiments to evaluate the LSTM model, including training/validation splitting and hyperparameter tuning.
- Helped with the preparation of the final report, integrating findings from all models and ensuring a cohesive structure.

Vinav Sancheti

- Led the implementation of the Siamese network, including designing the embedding network and implementing the contrastive loss function for similarity comparison.
- Conducted training and evaluation of the Siamese network, optimizing hyperparameters and analyzing results such as accuracy, FAR, and FRR.
- Developed the threshold-based analysis for the Siamese network to balance usability and security in the authentication system.

Manasa Rao

- Designed and implemented the preprocessing pipeline, including feature extraction (velocity, angle, normalization) and sequence formatting for both LSTM and Siamese networks.
- Designed and developed the gamified UI for the enrollment and authentication phases using Python and Tkinter, ensuring seamless data collection and usability.
- Coordinated the integration of different project components, ensuring smooth transitions between preprocessing, modeling, and UI.

References

[1] A. A. E. Ahmed, I. Traore, *A New Biometric Technology Based on Mouse Dynamics*, IEEE Transactions on Dependable and Secure Computing, 2007.

[2] H. Gamboa, A. Fred, *A Behavioral Biometric System Based on Human-Computer Interaction*, SPIE, 2004.

[3] P. Chong, Y. Elovici, A. Binder, *User Authentication Using Deep Neural Networks*, IEEE TIFS, 2019.

[4] M. Antal, N. Fejér, *Mouse Dynamics Based User Recognition Using Deep Learning*, Acta Univ. Sapientiae, Informatica, 2020.

[5] Fei-Fei Li, Rob Fergus, and Pietro Perona, *One-shot learning of object categories*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.

[6] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, *Signature Verification Using a "Siamese" Time Delay Neural Network*, International Journal of Pattern Recognition and Artificial Intelligence, 1993.

[7] H. Kumar, *Exploring Siamese Networks for Image Similarity Using Contrastive Loss*, Medium, 2023. [Online]. Available: `https://medium.com/@hayagriva99999/exploring-siamese-networks-for-image-similarity-using-contrastive-loss-f5d5ae5a0cc6`

[8] OpenAI, *ChatGPT: Language Model for AI-Assisted Content Generation*, 2023. [Online]. Available: `https://chat.openai.com/`