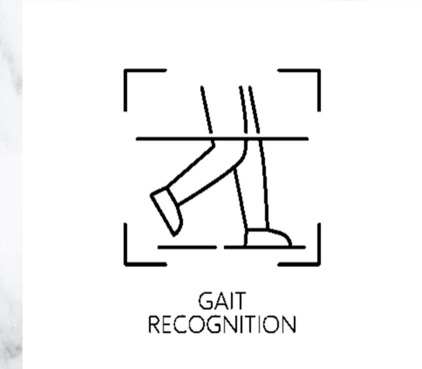
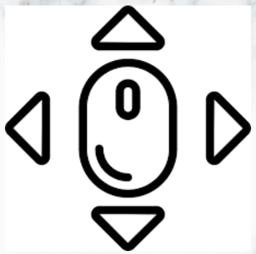
SecureAuth - Behavioral Biometrics for Enhanced Authentication Systems

24-25J-073





❑ Our Team

SUPERVISOR

Dr. Harinda Fernando

Assistant Professor

Faculty of Computing

CO-SUPERVISOR

Mr. Tharaniyawarma

Assistant Lecturer

Faculty of Computing

Madhubhashana H. N. D

IT21391668

CYBER SECURITY

Edirisinghe E.M.N

IT21340864

CYBER SECURITY

Anupama K.G. A

IT21345678

CYBER SECURITY

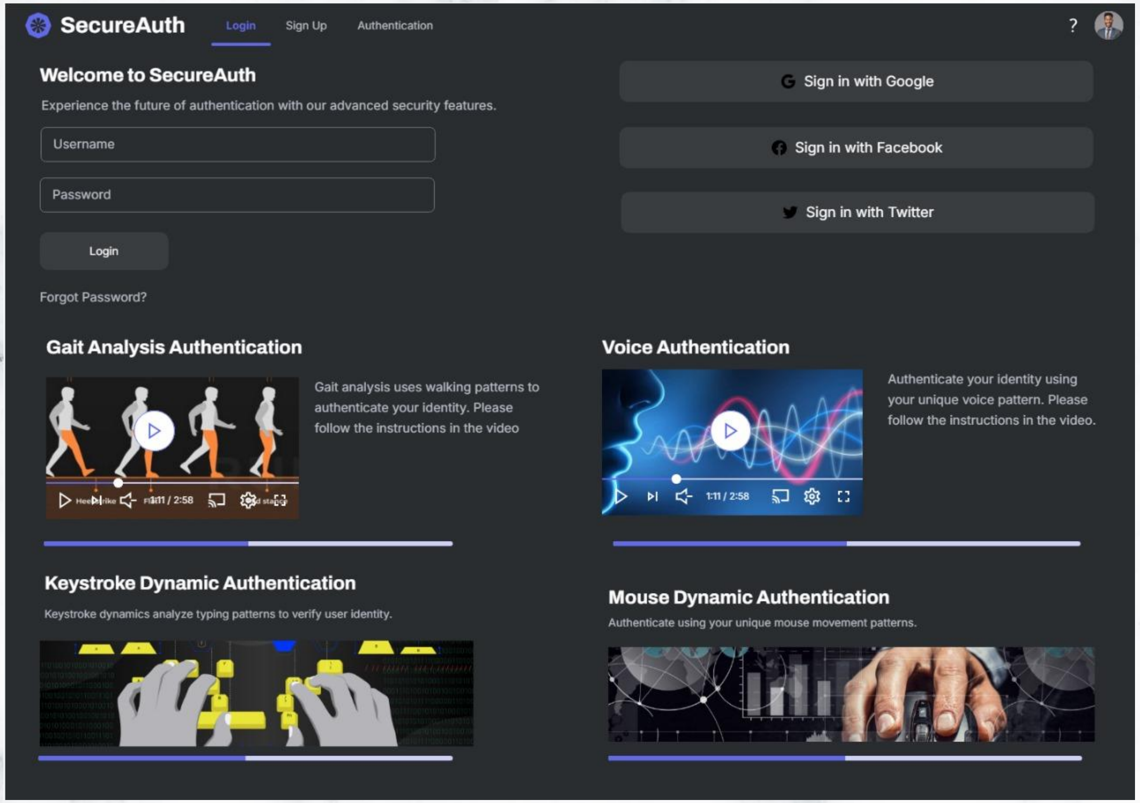
Rajapaksha R. P. K. D

IT21336072

CYBER SECURITY

SecureAuth

- **SecureAuth** is an advanced authentication solution that uses behavioral biometrics like gait, voice, typing patterns, and mouse movements to verify identity. With cutting-edge machine learning, it provides seamless, accurate, and secure access, making it ideal for protecting critical information and high-security environments.



❑ WHY **SecureAuth** IS IMPORTANT?

Enhanced Security: Protects sensitive data with multi-layered authentication.

Multi-Factor Authentication: Combines behavioral biometrics for stronger security.

Real-Time Authentication: Instant, seamless user verification.

Designed for High-Risk Environments: Ideal for government, finance, and healthcare.

Accurate & Reliable: Reduces false positives with authentication methods.

Future-Proof: Stays ahead of evolving cybersecurity threats.

❑ Research Objectives

✓ Primary Objective

To revolutionize user authentication by leveraging behavioral biometrics, offering a secure, seamless, and user-friendly alternative to traditional password systems.

❑ Research Objectives

✓ Secondary Objective

Gait Analysis
- Harness unique walking patterns to deliver an innovative and non-intrusive authentication method.

Mouse Dynamics
- Analyze natural mouse movements to enhance security without disrupting user experience.

Keystroke Dynamics
- Leverage typing patterns as an intuitive layer of identity verification.

Voice Biometric Authentication
- Utilize voice as a distinctive identifier, ensuring fast and reliable user verification..

Seamless Integration
- Provide an integrated solution that adapts to diverse environments and user needs.

Performance and Reliability
- Ensure the system performs consistently in real-world scenarios, offering high accuracy and resilience against breaches.

❑ Research Problem

Challenges with Traditional Biometric Authentication

- Vulnerable to spoofing and privacy concerns.
- Requires physical contact or proximity.

Limitations of Existing Gait Analysis Methods

- Often lack robustness and accuracy under diverse conditions.
- Need for improved feature extraction and modeling techniques.

Need for Robust and Accurate Behavioral Biometric Systems

- Behavioral biometrics offer non-intrusive and unique patterns.
- Potential to significantly enhance user authentication security.

❑ Research Question

How can behavioral biometrics, such as gait analysis, voice recognition, keystroke, and mouse dynamics, enhance the security and usability of authentication systems while maintaining user privacy and adaptability?

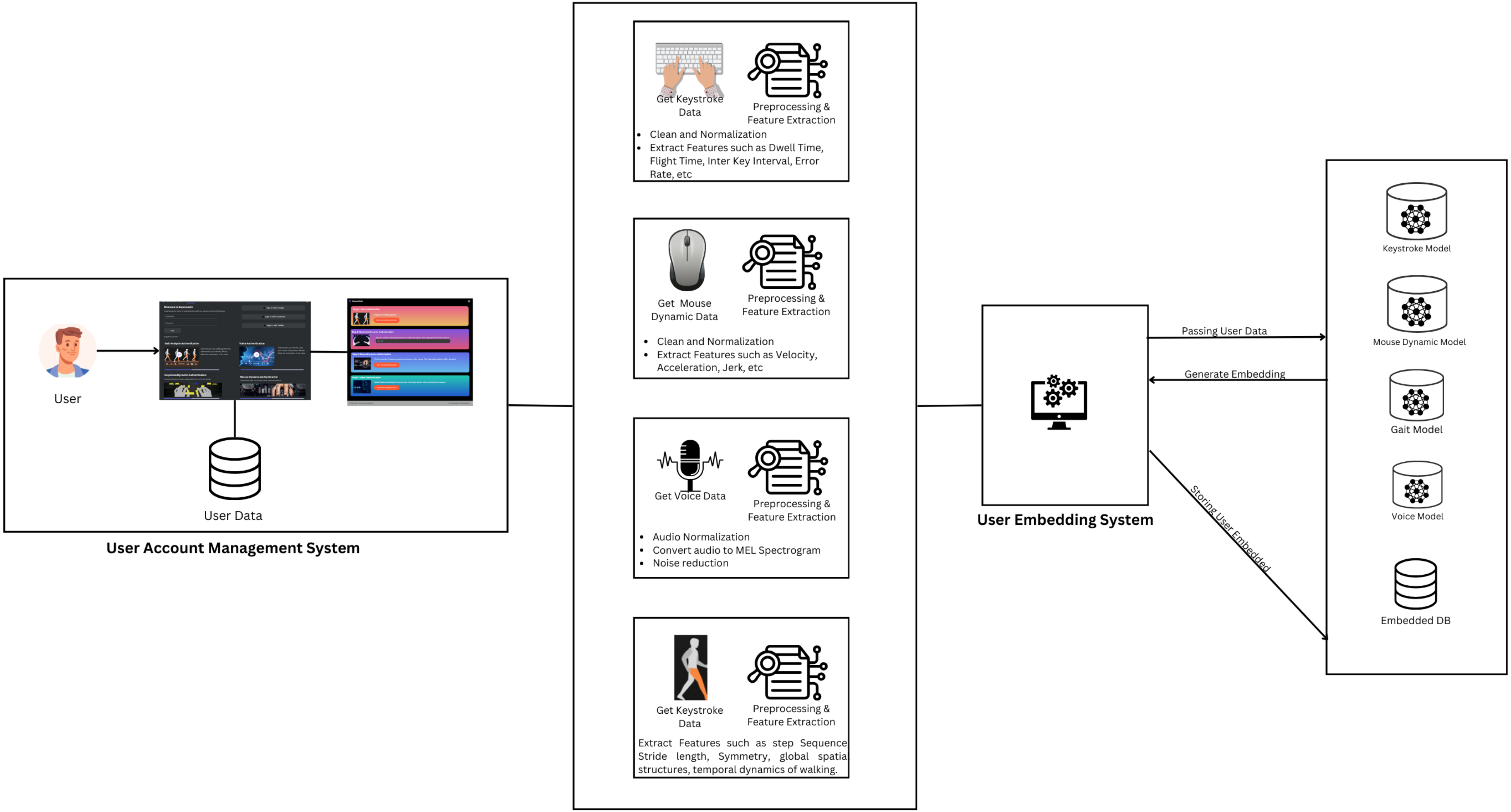
❑ Research Solution

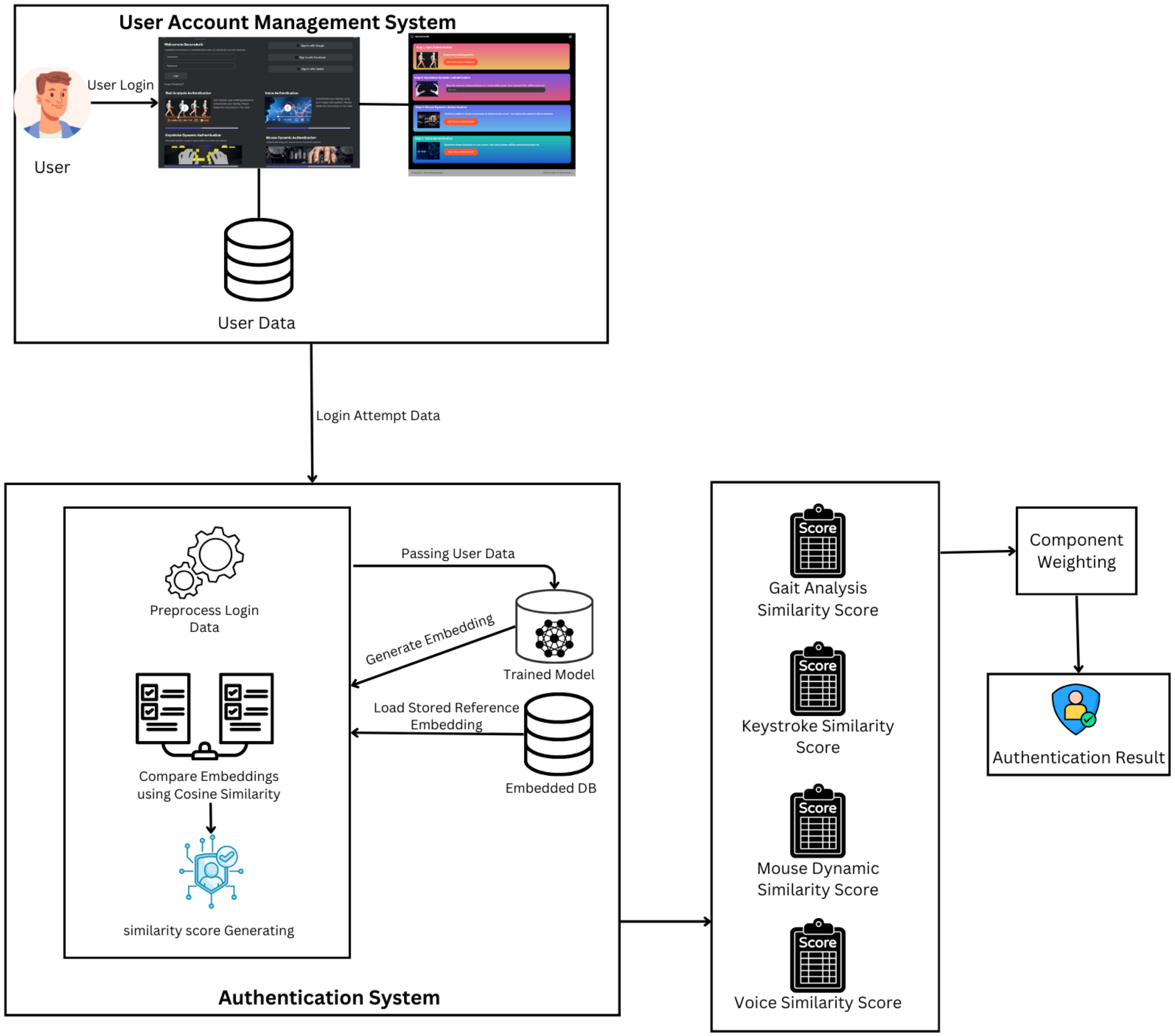
Develop **SecureAuth**, a multi-modal authentication system that combines gait, voice, keystroke, and mouse dynamics using advanced machine learning. It ensures robust security, user privacy, and seamless integration into high-security environments.

❑ Research Gap

Features/ Technologies	Scalability	Use of Online Datasets	Specialized Hardware Required
Project X	✓	✗	✗
Project Y	✓	✗	✗
Project Z	✗	✗	✓
SecureAuth	✓	✓	✗

System Diagram


User
User Data
User Account Management System
Get Keystroke Data
Preprocessing & Feature Extraction
Clean and Normalization
Extract Features such as Dwell Time, Flight Time, Inter Key Interval, Error Rate, etc
Get Mouse Dynamic Data
Preprocessing & Feature Extraction
Clean and Normalization
Extract Features such as Velocity, Acceleration, Jerk, etc
Get Voice Data
Preprocessing & Feature Extraction
Audio Normalization
Convert audio to MEL Spectrogram
Noise reduction
Get Keystroke Data
Preprocessing & Feature Extraction
Extract Features such as step Sequence Stride length, Symmetry, global spatia structures, temporal dynamics of walking.
User Embedding System
Passing User Data
Generate Embedding
Storing User Embedded
Keystroke Model
Mouse Dynamic Model
Gait Model
Voice Model
Embedded DB


User Account Management System
User Login
User
User Data
Login Attempt Data
Preprocess Login Data
Passing User Data
Generate Embedding
Trained Model
Load Stored Reference Embedding
Embedded DB
Compare Embeddings using Cosine Similarity
similarity score Generating
Authentication System
Score
Gait Analysis Similarity Score
Score
Keystroke Similarity Score
Score
Mouse Dynamic Similarity Score
Score
Voice Similarity Score
Component Weighting
Authentication Result

IT21391668 | H.N.D. MADHUBHASHANA

BSc (Hons) in Information Technology Specialising in Cyber Security

Introduction to Gait Component

- Gait authentication uses unique walking patterns for secure, non-intrusive user verification. By analyzing these patterns with a CNN model, SecureAuth adds an extra layer of security. This system ensures both accuracy and privacy with advanced encryption, offering a seamless user experience for high-security applications.

Gait Component Objectives

Develop CNN-based Model: Leverages convolutional layers to extract the spatial features of walking gait patterns.

Enhance Security: Use gait patterns for a unique, non-intrusive authentication method.

Improve Accuracy: Optimize model performance for real-world gait data.

Ensure Privacy: Maintain user data privacy with encrypted storage and processing.

Gait Component Sub-Objectives

Train Model

- Optimize the CNN-based model to achieve faster processing speeds and improved accuracy in gait recognition.

Evaluate Model Performance

- Validate the model with diverse gait data to ensure robustness.

Extract Features

- Improve techniques for extracting relevant gait features.

Data Collection

- Capture high-quality Gait Energy Images (GEIs) for analysis.

❑ Component Gap

Features/ Technologies	GEI	Use of Online Datasets	Multi model integration
Project X	✔	✔	✘
Project Y	✔	✘	✘
Project Z	✘	✘	✘
SecureAuth	✔	✔	✔

Component Question

What preprocessing ensures optimal GEI quality?

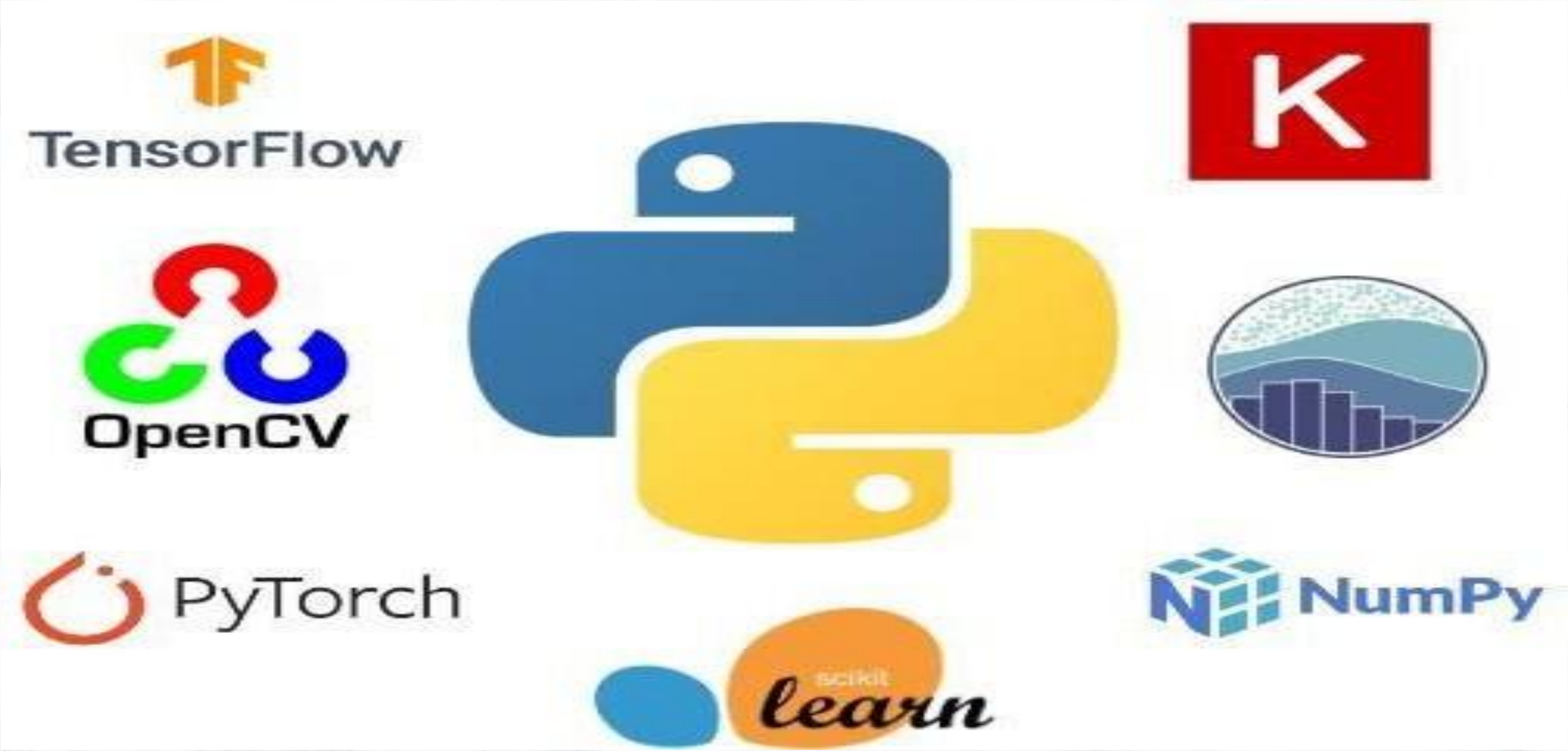
How can we enhance accuracy in distinguishing unique gait patterns?

Component Solution

Align images, normalize intensities, and extract clean silhouettes to enhance data quality.

Integrate advanced feature extraction model techniques.

Technologies



Methodology

Data Collection	Preprocessing	Model Development	Training and Validation	Evaluation
• Gather Gait Energy Images (GEIs) from online datasets.	• Normalize, resize, and augment images.	• Optimize a CNN model for fast, accurate gait recognition.	• Train on GEIs and validate with cross-validation.	• Assess performance using metrics like accuracy and F1 score.

Novelty

- The gait authentication component uses a CNN model to extract spatial features from Gait Energy Images (GEIs), ensuring a robust, non-intrusive, and efficient authentication process. This approach adapts to variations like clothing or walking speed, making it ideal for real-time, high-security authentication.

```python
def build_base_network(input_shape):
    inputs = Input(shape=input_shape)

    x = Conv2D(32, (3, 3), padding='same')(inputs)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 32)  # Enhanced feature extraction
    x = MaxPooling2D((2, 2))(x)

    x = Conv2D(64, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 64)
    x = MaxPooling2D((2, 2))(x)

    x = Conv2D(128, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 128)
    x = MaxPooling2D((2, 2))(x)

    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)

    return Model(inputs, x, name="base_network")
```

```python
# ------------------------------
# THE SIAMESE NETWORK
# ------------------------------
def build_siamese_network(input_shape):
    base_network = build_base_network(input_shape)
    input_a = Input(shape=input_shape)
    input_b = Input(shape=input_shape)

    encoded_a = base_network(input_a)
    encoded_b = base_network(input_b)

    diff = Lambda(lambda tensors: tf.abs(tensors[0] - tensors[1]))([encoded_a, encoded_b])
    outputs = Dense(1, activation='sigmoid')(diff)

    siamese_model = Model(inputs=[input_a, input_b], outputs=outputs, name="siamese_network")
    return siamese_model, base_network
```

```python
inputs = Input(shape=input_shape)
features = base_network(inputs)
x = Dense(128)(features)
x = LeakyReLU()(x)
x = Dropout(0.5)(x)

embedding = Lambda(
    lambda t: tf.math.l2_normalize(t, axis=1),
    output_shape=lambda input_shape: input_shape,
    name="embedding"
)(x)

outputs = Dense(num_classes, activation='softmax')(embedding)

classifier_model = Model(inputs, outputs, name="gait_classifier")
classifier_model.compile(optimizer=Adam(learning_rate=0.0005),
                         loss=sparse_focal_loss(gamma=2., alpha=0.25),
                         metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='accuracy', patience=3, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='loss', factor=0.5, patience=2, verbose=1)
```

Project Progress Completion

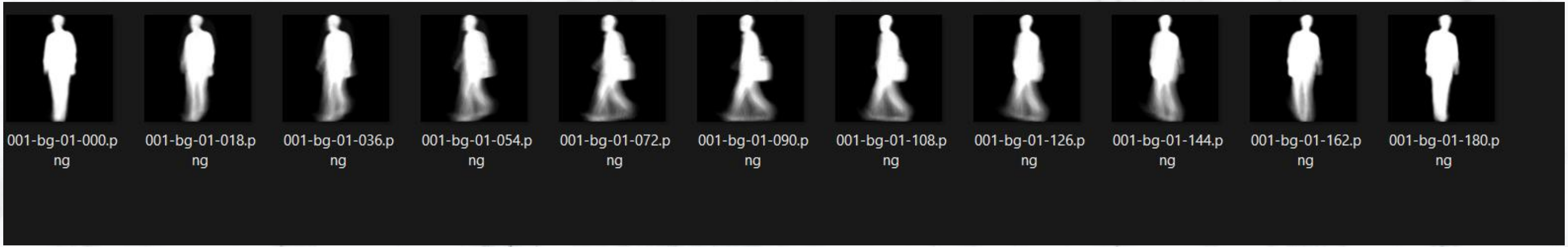
- Online Dataset

Name	Date modified	Type
001	9/25/2024 1:30 PM	File folder
002	9/25/2024 1:30 PM	File folder
003	9/25/2024 1:30 PM	File folder
004	9/25/2024 1:30 PM	File folder
005	9/25/2024 1:30 PM	File folder
006	9/25/2024 1:30 PM	File folder
007	9/25/2024 1:30 PM	File folder
008	9/25/2024 1:30 PM	File folder
009	9/25/2024 1:30 PM	File folder
010	9/25/2024 1:30 PM	File folder
011	9/25/2024 1:30 PM	File folder
012	9/25/2024 1:30 PM	File folder
013	9/25/2024 1:30 PM	File folder
014	9/25/2024 1:30 PM	File folder
015	9/25/2024 1:30 PM	File folder
016	9/25/2024 1:30 PM	File folder
017	9/25/2024 1:30 PM	File folder
018	9/25/2024 1:30 PM	File folder
019	9/25/2024 1:30 PM	File folder
020	9/25/2024 1:31 PM	File folder
021	9/25/2024 1:31 PM	File folder
022	9/25/2024 1:31 PM	File folder
023	9/25/2024 1:31 PM	File folder
024	9/25/2024 1:31 PM	File folder
025	9/25/2024 1:31 PM	File folder
026	9/25/2024 1:31 PM	File folder
027	9/25/2024 1:31 PM	File folder
028	9/25/2024 1:31 PM	File folder
029	9/25/2024 1:31 PM	File folder

bg-01	9/25/2024 1:30 PM	File folder
bg-02	9/25/2024 1:30 PM	File folder
cl-01	9/25/2024 1:30 PM	File folder
cl-02	9/25/2024 1:30 PM	File folder
nm-01	9/25/2024 1:30 PM	File folder
nm-02	9/25/2024 1:30 PM	File folder
nm-03	9/25/2024 1:30 PM	File folder
nm-04	9/25/2024 1:30 PM	File folder
nm-05	9/25/2024 1:30 PM	File folder
nm-06	9/25/2024 1:30 PM	File folder

Project Progress Completion

- Online Dataset



Project Progress Completion

- Generating Pairs

```python
def get_subject_list(root_dir):
    """Return a sorted list of subject folder names (e.g., '001', '002', ...)."""
    subjects = [d for d in os.listdir(root_dir) if os.path.isdir(os.path.join(root_dir, d))]
    subjects.sort()
    return subjects

def load_subject_images(subject_path):
    """
    Load image file paths for a subject by scanning all condition folders
    (e.g., nm-01, bg-01, etc.) within the subject's folder.
    """
    images = []
    for condition in os.listdir(subject_path):
        condition_path = os.path.join(subject_path, condition)
        if os.path.isdir(condition_path):
            for filename in os.listdir(condition_path):
                if filename.lower().endswith('.png'):
                    image_path = os.path.join(condition_path, filename)
                    images.append(image_path)
    return images

# Build a dictionary mapping subject IDs to their list of image paths
subjects = get_subject_list(data_dir)
print(f"Found {len(subjects)} subjects.")
```

```python
positive_pairs = []
for subject, images in subject_images_dict.items():
    if len(images) < 2:
        continue
    for i in range(len(images) - 1):
        positive_pairs.append((images[i], images[i+1]))

num_positive = len(positive_pairs)
print(f"Total positive pairs: {num_positive}")


negative_pairs = []
while len(negative_pairs) < num_positive:
    subj1, subj2 = random.sample(subjects, 2)
    if not subject_images_dict[subj1] or not subject_images_dict[subj2]:
        continue
    img1 = random.choice(subject_images_dict[subj1])
    img2 = random.choice(subject_images_dict[subj2])
    negative_pairs.append((img1, img2))

num_negative = len(negative_pairs)
print(f"Total negative pairs: {num_negative}")
```

Project Progress Completion

- Model Coding

```python
# THE BASE NETWORK
# -------------------------------
def build_base_network(input_shape):
    inputs = Input(shape=input_shape)

    x = Conv2D(32, (3, 3), padding='same')(inputs)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 32)  # Enhanced feature extraction
    x = MaxPooling2D((2, 2))(x)

    x = Conv2D(64, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 64)
    x = MaxPooling2D((2, 2))(x)

    x = Conv2D(128, (3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = LeakyReLU()(x)
    x = residual_block(x, 128)
    x = MaxPooling2D((2, 2))(x)

    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)

    return Model(inputs, x, name="base_network")
```

```python
# ------------------------------
# THE SIAMESE NETWORK
# ------------------------------
def build_siamese_network(input_shape):
    base_network = build_base_network(input_shape)
    input_a = Input(shape=input_shape)
    input_b = Input(shape=input_shape)

    encoded_a = base_network(input_a)
    encoded_b = base_network(input_b)

    diff = Lambda(lambda tensors: tf.abs(tensors[0] - tensors[1]))([encoded_a, encoded_b])
    outputs = Dense(1, activation='sigmoid')(diff)

    siamese_model = Model(inputs=[input_a, input_b], outputs=outputs, name="siamese_network")
    return siamese_model, base_network
```

```python
inputs = Input(shape=input_shape)
features = base_network(inputs)
x = Dense(128)(features)
x = LeakyReLU()(x)
x = Dropout(0.5)(x)

embedding = Lambda(
    lambda t: tf.math.l2_normalize(t, axis=1),
    output_shape=lambda input_shape: input_shape,
    name="embedding"
)(x)

outputs = Dense(num_classes, activation='softmax')(embedding)

classifier_model = Model(inputs, outputs, name="gait_classifier")
classifier_model.compile(optimizer=Adam(learning_rate=0.0005),
                         loss=sparse_focal_loss(gamma=2., alpha=0.25),
                         metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='accuracy', patience=3, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='loss', factor=0.5, patience=2, verbose=1)
```

Project Progress Completion

- Model Training

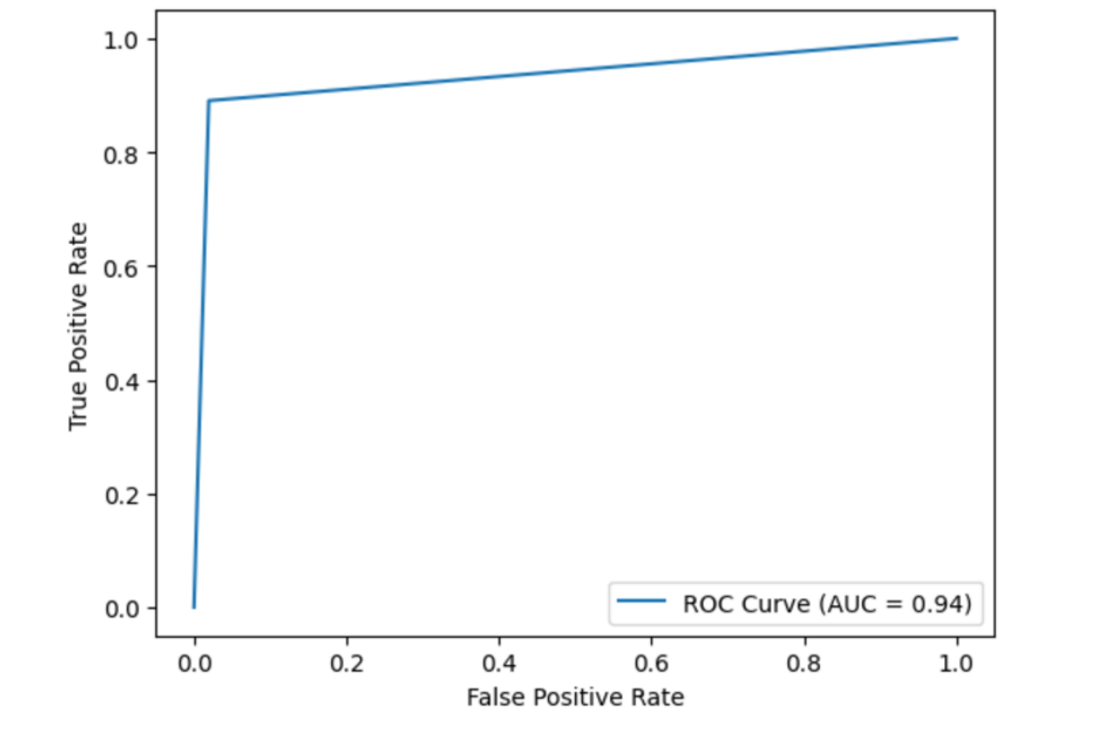
```
cm = confusion_matrix(y_true, y_pred)
report = classification_report(y_true, y_pred, target_names=class_names)

print("Confusion Matrix:")
print(cm)
print("\nClassification Report:")
print(report)
```

```
Training Siamese network...
Epoch 1/60
 88/841 ━━━━━━━━━━━━━━━━━━━━ 8:09 650ms/step - accuracy: 0.5358 - loss: 0.6950
```

Project Progress Completion

- Model Training

accuracy			0.88	13593
macro avg	0.88	0.88	0.88	13593
weighted avg	0.88	0.88	0.88	13593



Project Progress Completion

- Model Evaluation

```
query_img = preprocess_image(query_image_path)
query_embedding = embedding_model.predict(query_img, verbose=0)[0]

user_embeddings_data = np.load(NPZ_PATH, allow_pickle=True)
if user_id not in user_embeddings_data:
    raise ValueError(f"User ID {user_id} not found in embeddings.")


user_embeddings = user_embeddings_data[user_id]
similarities = np.dot(user_embeddings, query_embedding)
confidence = np.max(similarities)

print(f"{confidence:.2f}", end="")
```

```
0.57
```

Progress

PP1 – 50%

- Dataset acquired and preprocessed.
- Model architecture coded.
- Model training initiated and initial results gathered.

PP2 – 90%

- Train model to achieve acceptable accuracy and F1 score.
- Validate model performance with real-world data.

Final – 100%

- Complete frontend development and user interface.
- Finalize integration of all components.
- Compile and submit the final project report.

Future Interactions For Final Phase

Model Optimization

- Fine-tune CNN for better accuracy and F1 score.

Frontend Preparation

- Finalize frontend development.

REFERENCES

G. Giorgi, F. Martinelli, A. Saracino, and M. Sheikhalishahi, "Walking Through the Deep: Gait Analysis for User Authentication Through Deep Learning," *Inria*, [Online]. Available: https://inria.hal.science/hal-02023725/document. Accessed: Aug. 4, 2024.

I. Stylios, "Behavioral Biometrics for Continuous Authentication: Security and Privacy Issues," *ResearchGate*, Jan. 2023. [Online]. Available: https://www.researchgate.net/publication/369142299_Behavioral_Biometrics_for_Continuous_Authentication_Security_and_Privacy_Issues. Accessed: Aug. 4, 2024.

IT21340864 | E.M.N. EDIRISINGHE

BSc (Hons) in Information Technology Specializing in Cyber Security

Introduction to Keystroke Component

- SecureAuth leverages keystroke dynamics for secure, user authentication. By analyzing the unique patterns in a user's typing behavior, such as key press duration and typing speed, the system can identify and verify individuals. This method provides an additional layer of security without requiring any physical interaction, offering a seamless user experience while ensuring robust protection against unauthorized access. Keystroke biometrics integrates with other authentication techniques to enhance overall security in high-stakes applications.

Component Objectives

- **Develop Efficient One-Shot Learning**: Implement Siamese Network architecture for authentication using minimal data, ensuring quick enrollment and recognition
- **Achieve Robust and Real-Time Performance**: Design the system for fast, reliable, and scalable user authentication in live environments.
- **Ensure Privacy and Scalability**: Implement efficient user embedding storage and management for secure and scalable deployment

❑ Component Gap

Features/ Technologies	Use of Online Datasets	Bidirectional LSTM for Sequence Modeling	One-Shot Siamese Network for Authentication	Cross-User Authentication via Embedding Matching
Project X	✓	✗	✗	✗
Project Y	✓	✓	✗	✓
Project Z	✗	✗	✗	✗
SecureAuth	✓	✓	✓	✓

Component Question & Solution

How can we ensure accurate authentication with minimal user enrollment data ?

Employing a one-shot Siamese network with Bidirectional LSTMs, the system captures unique typing patterns efficiently, enabling high accuracy even with minimal user input

How does the system determine an appropriate threshold for authentication decisions ?

Analyzing the cosine similarity scores between reference embeddings and login attempt embeddings, optimizing the balance between false positives and false negatives

How is user data secured during the embedding and authentication process ?

The system employs encryption for embedding storage and transfer, ensuring data privacy.

Methodology

Data Collection

Use online datasets to collect keystroke data.

Data Preprocessing

Clean and normalize the collected data to ensure consistency and quality.

Feature Extraction

Extract Sequence Feature like Dwell time, Flight Time, Inter-Key Interval, Release Interval, etc

Preparing for Model Input

Sequence Analysis

Keystrokes are analyzed over time to capture typing rhythm, speed, and patterns

Use Advance technologies like Bidirectional LSTMs to learn the temporal dependencies between keystrokes.

Model Integration and Authentication Decision

Sequence analysis models to form a complete authentication system

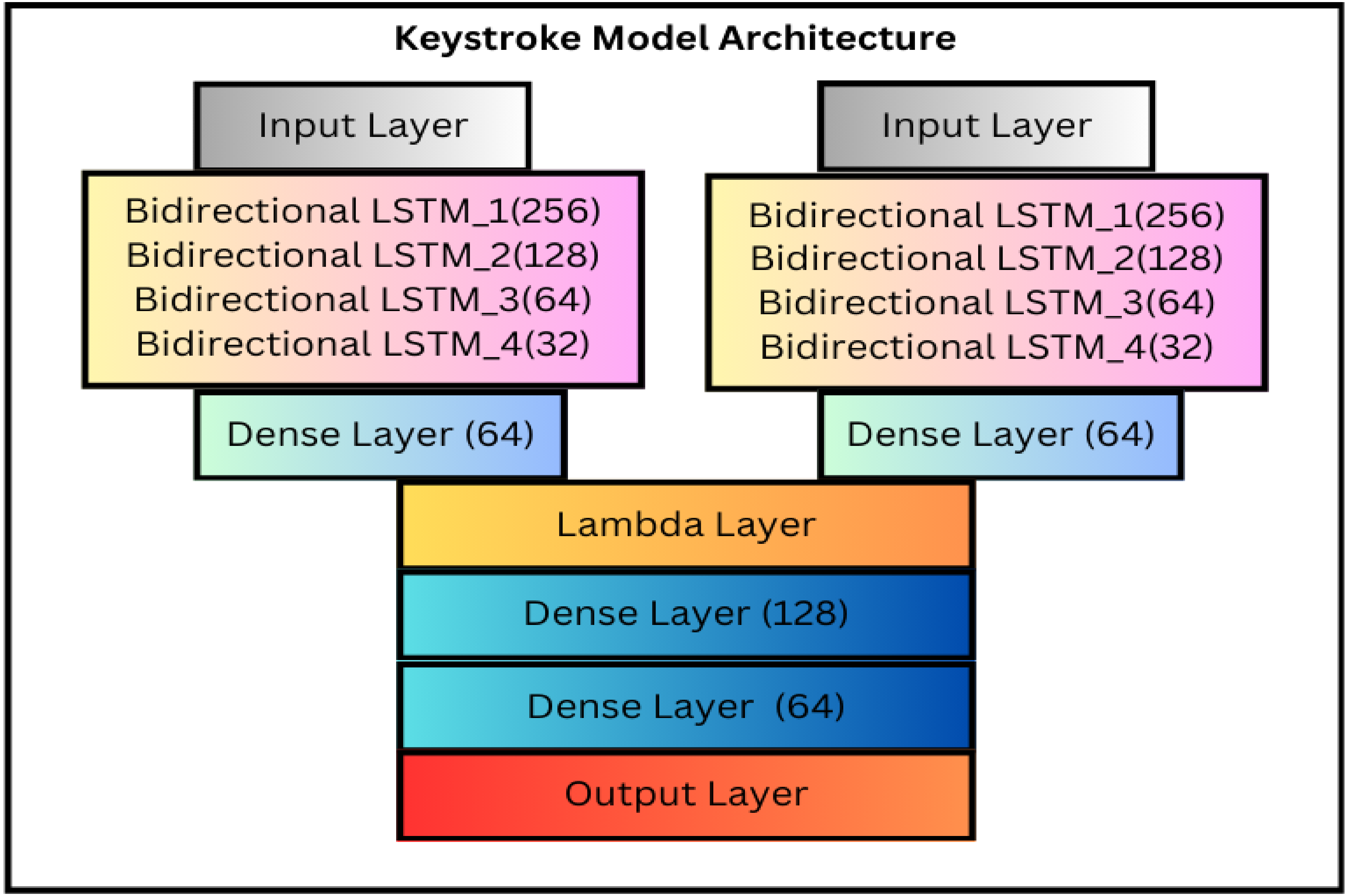
Generate embeddings for both reference user data and login attempt data.

Compare reference embeddings and login embeddings using similarity metrics like cosine similarity

Set a similarity threshold to decide whether the login attempt is authentic.

Novelty

- **Bidirectional LSTM Layers**: The use of Bidirectional LSTMs (Bi-LSTMs) helps capture both backward and forward dependencies in the keystroke sequence, making the model more adept at understanding the temporal aspects of typing behavior.
- **User-Specific Embedding Framework:** SecureAuth uses an advanced one-shot siamese network architecture to generate user-specific embeddings, ensuring personalized authentication accuracy.


Keystroke Model Architecture
Input Layer
Bidirectional LSTM_1(256)
Bidirectional LSTM_2(128)
Bidirectional LSTM_3(64)
Bidirectional LSTM_4(32)
Dense Layer (64)
Input Layer
Bidirectional LSTM_1(256)
Bidirectional LSTM_2(128)
Bidirectional LSTM_3(64)
Bidirectional LSTM_4(32)
Dense Layer (64)
Lambda Layer
Dense Layer (128)
Dense Layer (64)
Output Layer

Project Progress Completion

- Online Dataset

Name	Date modified	Type
Participant_5	12/2/2024 15:14	File folder
Participant_7	12/2/2024 15:14	File folder
Participant_23	12/2/2024 15:14	File folder
Participant_24	12/2/2024 15:14	File folder
Participant_25	12/2/2024 15:14	File folder
Participant_30	12/2/2024 15:14	File folder
Participant_31	12/2/2024 15:14	File folder
Participant_32	12/2/2024 15:14	File folder
Participant_33	12/2/2024 15:14	File folder
Participant_35	12/2/2024 15:14	File folder
Participant_36	12/2/2024 15:14	File folder
Participant_38	12/2/2024 15:14	File folder
Participant_39	12/2/2024 15:14	File folder
Participant_40	12/2/2024 15:14	File folder
Participant_45	12/2/2024 15:14	File folder
Participant_49	12/2/2024 15:15	File folder

PARTICIPANT_I	TEST_SECTION_ID	PRESS_TIME	RELEASE_TIME	LETTER	KEYSTROKE_ID
10001	106696	1.47205E+12	1.47205E+12	SHIFT	5088570
10001	106696	1.47205E+12	1.47205E+12	H	5088575
10001	106696	1.47205E+12	1.47205E+12	e	5088580
10001	106696	1.47205E+12	1.47205E+12		5088581
10001	106696	1.47205E+12	1.47205E+12	p	5088583
10001	106696	1.47205E+12	1.47205E+12	l	5088609
10001	106696	1.47205E+12	1.47205E+12	a	5088612
10001	106696	1.47205E+12	1.47205E+12	y	5088616
10001	106696	1.47205E+12	1.47205E+12	e	5088618
10001	106696	1.47205E+12	1.47205E+12	d	5088621

ERROR_RATE	AVG_WPM_15	AVG_IKI	ECPC	KSPC	ROR
3.840472674	60.8829	169.3101457	0.045317221	1.152567976	0.4332
1.612903226	33.444	319.0930581	0.041420118	1.137573964	0.1671
0.735294118	40.7928	268.541052	0.03974359	1.105128205	0.1736
1.293900185	85.3952	124.2083817	0.038817006	1.136783734	0.4083
0.170357751	37.3318	267.2398387	0.042662116	1.208191126	0.3137
1.47601476	41.989	260.6474779	0.027777778	1.109259259	0.0599
4.320987654	22.8563	466.7668385	0.054574639	1.144462279	0.033
0.36900369	80.4561	135.974997	0.035120148	1.103512015	0.2251
0.3125	77.0218	131.1405147	0.071875	1.1796875	0.4892
0.299401198	33.7949	291.7303685	0.09924812	1.239097744	0.1978
0.304414003	26.6545	375.1684623	0.083969466	1.216793893	0.0435
0.866551127	71.9805	137.5283745	0.019097222	1.076388889	0.5033

Project Progress Completion

- Feature Extraction

```python
# Iterate over each participant's data
for participant_id, group in tqdm(grouped_data, desc="Processing participants", unit="participant"):
    # Sort the group by PRESS_TIME (assuming this is the correct order)
    group = group.sort_values(by='PRESS_TIME')

    # Calculate derived features
    for i in range(len(group) - 1):
        current_row = group.iloc[i]
        next_row = group.iloc[i + 1]

        # Dwell Time (current key)
        dwell_time = current_row['RELEASE_TIME'] - current_row['PRESS_TIME']

        # Flight Time (time between releasing the current key and pressing the next key)
        flight_time = next_row['PRESS_TIME'] - current_row['RELEASE_TIME']

        # Inter-Key Interval (next PRESS_TIME - current PRESS_TIME)
        inter_key_interval = next_row['PRESS_TIME'] - current_row['PRESS_TIME']

        # Release Interval (next RELEASE_TIME - current RELEASE_TIME)
        release_interval = next_row['RELEASE_TIME'] - current_row['RELEASE_TIME']

        # Overlap Time (key1 RELEASE_TIME - key2 PRESS_TIME)
        overlap_time = current_row['RELEASE_TIME'] - next_row['PRESS_TIME']

        # Press-Release Lag (key2 PRESS_TIME - key1 RELEASE_TIME)
        press_release_lag = next_row['PRESS_TIME'] - current_row['RELEASE_TIME']
```

Project Progress Completion

- Session Separation & Balancing

```python
# Iterate through each session pair
for _, row in tqdm(pair_data.iterrows(), total=len(pair_data), desc="Processing session pairs"):
    session_1, is_short_1 = load_session_data(row['Session1_path'])
    session_2, is_short_2 = load_session_data(row['Session2_path'])

    # Increase discarded count if any session is too short
    if is_short_1 or is_short_2:
        discarded_sessions_count += 1
        continue  # Skip pairs with short sessions

    label = row['label']

    # Extract non-overlapping sequences
    session_1_windows = [session_1[i:i + sequence_length] for i in range(0, len(session_1) - sequence_length + 1, stride)]
    session_2_windows = [session_2[i:i + sequence_length] for i in range(0, len(session_2) - sequence_length + 1, stride)]

    # Ensure equal number of sequences from both sessions
    min_windows = min(len(session_1_windows), len(session_2_windows))

    session_1_data.extend(session_1_windows[:min_windows])
    session_2_data.extend(session_2_windows[:min_windows])
    labels.extend([label] * min_windows)

# Convert to NumPy arrays
session_1_data = np.array(session_1_data, dtype='float32')  # Shape (n_samples, 10, 12)
session_2_data = np.array(session_2_data, dtype='float32')  # Shape (n_samples, 10, 12)
labels = np.array(labels, dtype='float32')

#Shuffle the dataset to prevent ordering bias
session_1_data, session_2_data, labels = shuffle(session_1_data, session_2_data, labels, random_state=42)

# Return data and count of discarded sessions
return session_1_data, session_2_data, labels, discarded_sessions_count
```


```python
    label = row['label']

    # Extract non-overlapping sequences
    session_1_windows = [session_1[i:i + sequence_length] for i in range(0, len(session_1) - sequence_length + 1, stride)]
    session_2_windows = [session_2[i:i + sequence_length] for i in range(0, len(session_2) - sequence_length + 1, stride)]

    # Ensure equal number of sequences from both sessions
    min_windows = min(len(session_1_windows), len(session_2_windows))

    session_1_data.extend(session_1_windows[:min_windows])
    session_2_data.extend(session_2_windows[:min_windows])
    labels.extend([label] * min_windows)

# Convert to NumPy arrays
session_1_data = np.array(session_1_data, dtype='float32')  # Shape (n_samples, 10, 12)
session_2_data = np.array(session_2_data, dtype='float32')  # Shape (n_samples, 10, 12)
labels = np.array(labels, dtype='float32')

#Shuffle the dataset to prevent ordering bias
session_1_data, session_2_data, labels = shuffle(session_1_data, session_2_data, labels, random_state=42)

# Return data and count of discarded sessions
return session_1_data, session_2_data, labels, discarded_sessions_count
```

Project Progress Completion

- Model Coding

```python
def siamese_lstm_block(input_layer):
    """A function to define the shared Bidirectional LSTM block."""

    # First Bidirectional LSTM layer (256 units)
    x = Bidirectional(LSTM(256, return_sequences=True, dropout=0.4, recurrent_dropout=0.4))(input_layer)

    # Second Bidirectional LSTM layer (128 units)
    x = Bidirectional(LSTM(128, return_sequences=True, dropout=0.4, recurrent_dropout=0.4))(x)

    # Third Bidirectional LSTM layer (64 units)
    x = Bidirectional(LSTM(64, return_sequences=True, dropout=0.4, recurrent_dropout=0.4))(x)

    # Fourth Bidirectional LSTM layer (32 units)
    x = Bidirectional(LSTM(32, return_sequences=False))(x)

    # Return the output of the LSTM block
    return x

# Input layers for paired sequences
input1 = Input(shape=(1, 12), name="input_sequence_1")  # Sequence 1
input2 = Input(shape=(1, 12), name="input_sequence_2")  # Sequence 2

# Apply the same LSTM block to both inputs (shared weights)
output1 = siamese_lstm_block(input1)
output2 = siamese_lstm_block(input2)

# Dense layer with 64 units
dense1 = Dense(64, activation='relu')(output1)
dense2 = Dense(64, activation='relu')(output2)

# Dropout layers after Dense layers
dropout1 = Dropout(0.4)(dense1)
dropout2 = Dropout(0.4)(dense2)
```

```python
# Lambda layer to compute the absolute difference between the two embeddings
def absolute_difference(tensors):
    return K.abs(tensors[0] - tensors[1])

lambda_layer = Lambda(absolute_difference)([dropout1, dropout2])

# Fully connected layers for final classification
fc1 = Dense(128, activation='relu')(lambda_layer)
dropout3 = Dropout(0.4)(fc1)

fc2 = Dense(64, activation='relu')(dropout3)
dropout4 = Dropout(0.4)(fc2)

# Final output layer (binary classification for similarity)
final_output = Dense(1, activation='sigmoid')(dropout4)

# Define the model
model = Model(inputs=[input1, input2], outputs=final_output)
```

Project Progress Completion

- Model Training

```
# Define Exponential Decay schedule for learning rate
lr_schedule = ExponentialDecay(
    initial_learning_rate=0.001,   # Initial learning rate
    decay_steps=10000,              # Number of steps after which the learning rate decays
    decay_rate=0.96,                # Decay rate (learning rate is multiplied by decay_rate)
    staircase=True                  # If True, decay happens in discrete intervals
)


# Define early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Compile the model
optimizer = Adam(learning_rate=lr_schedule)  # Clip gradients to a max value of 1
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Train the updated model
history = model.fit(
    [X_train_1, X_train_2],
    y_train,
    validation_data=([X_test_1, X_test_2], y_test),
    epochs=50,
    batch_size=64,
    callbacks=[early_stopping],
    #verbose=1
)

# Check the training and validation accuracy
print("Training accuracy: ", history.history['accuracy'][-1])
print("Validation accuracy: ", history.history['val_accuracy'][-1])
```

Project Progress Completion

- Model Training

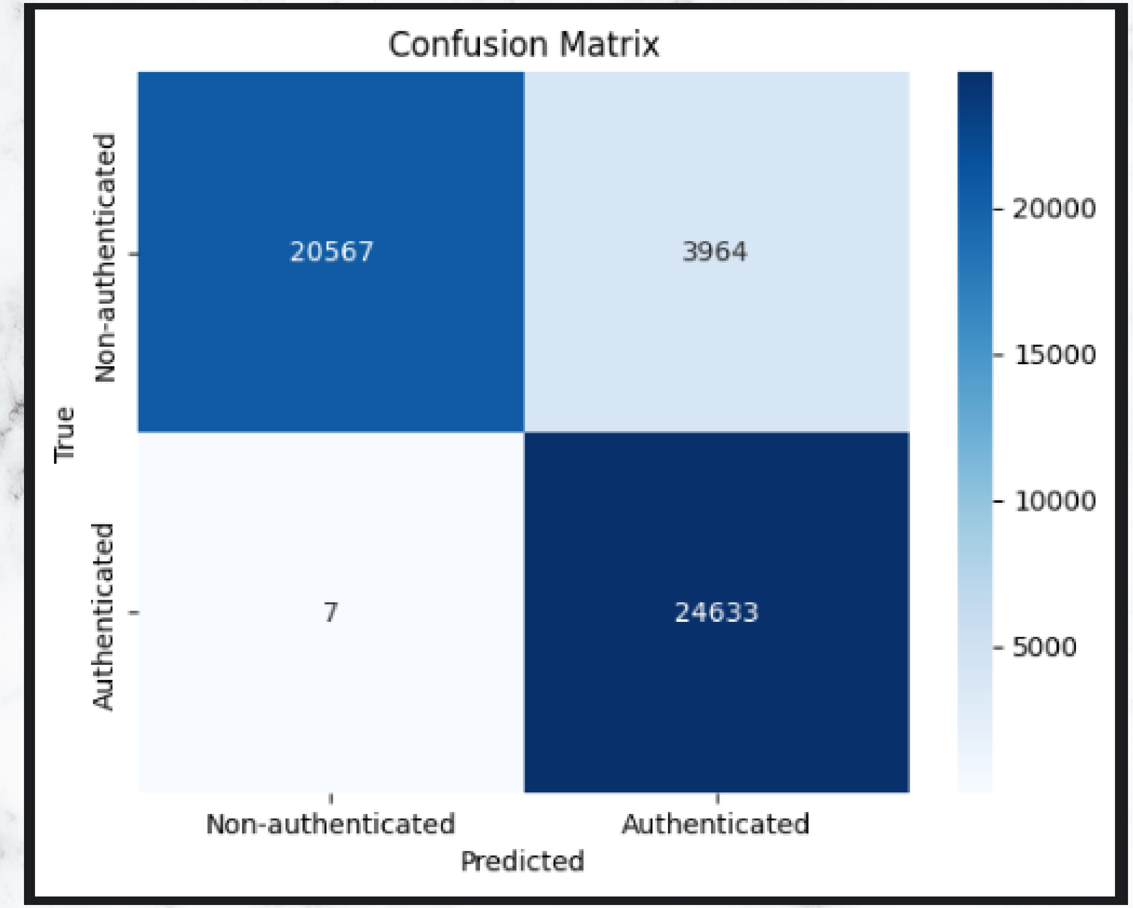
```
Epoch 1/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 529s 166ms/step - accuracy: 0.5980 - loss: 0.6382 - val_accuracy:
0.7589 - val_loss: 0.4886
Epoch 2/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 499s 162ms/step - accuracy: 0.7434 - loss: 0.5108 - val_accuracy:
0.8299 - val_loss: 0.3852
Epoch 3/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 498s 162ms/step - accuracy: 0.7827 - loss: 0.4577 - val_accuracy:
0.8550 - val_loss: 0.3448
Epoch 4/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 500s 163ms/step - accuracy: 0.8016 - loss: 0.4283 - val_accuracy:
0.8665 - val_loss: 0.3201
Epoch 5/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 488s 159ms/step - accuracy: 0.8139 - loss: 0.4084 - val_accuracy:
0.8790 - val_loss: 0.3058
Epoch 6/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 482s 157ms/step - accuracy: 0.8255 - loss: 0.3914 - val_accuracy:
0.8810 - val_loss: 0.2921
Epoch 7/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 498s 162ms/step - accuracy: 0.8286 - loss: 0.3842 - val_accuracy:
0.8825 - val_loss: 0.2905
Epoch 8/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 503s 163ms/step - accuracy: 0.8349 - loss: 0.3733 - val_accuracy:
```

```
Epoch 23/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 481s 157ms/step - accuracy: 0.8709 - loss: 0.3105 - val_accuracy:
0.9141 - val_loss: 0.2247
Epoch 24/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 484s 157ms/step - accuracy: 0.8701 - loss: 0.3121 - val_accuracy:
0.9171 - val_loss: 0.2239
Epoch 25/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 488s 159ms/step - accuracy: 0.8727 - loss: 0.3075 - val_accuracy:
0.9167 - val_loss: 0.2211
Epoch 26/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 487s 158ms/step - accuracy: 0.8729 - loss: 0.3060 - val_accuracy:
0.9171 - val_loss: 0.2216
Epoch 27/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 485s 158ms/step - accuracy: 0.8747 - loss: 0.3043 - val_accuracy:
0.9149 - val_loss: 0.2246
Epoch 28/50
3074/3074 ━━━━━━━━━━━━━━━━━━━━ 491s 160ms/step - accuracy: 0.8760 - loss: 0.3015 - val_accuracy:
0.9171 - val_loss: 0.2190
Training accuracy:  0.8755866289138794
Validation accuracy:  0.917125940322876
```

Project Progress Completion

- Model Evaluation

```
769/769 ━━━━━━━━━━━━━━━━━━━━ 29s 37ms/step - accuracy: 0.9202 - loss: 0.2169
Test Loss:  0.21872469782829285
Test Accuracy:  0.9192410111427307
1537/1537 ━━━━━━━━━━━━━━━━━━━━ 60s 38ms/step
Accuracy: 0.9192
Precision: 0.8614
Recall: 0.9997
F1 Score: 0.9254
AUC: 0.9589
              precision    recall  f1-score   support

         0.0       1.00      0.84      0.91     24531
         1.0       0.86      1.00      0.93     24640

    accuracy                           0.92     49171
   macro avg       0.93      0.92      0.92     49171
weighted avg       0.93      0.92      0.92     49171
```



Project Progress Completion

- User Embedding

```python
# Define the absolute_difference function (with explicit output_shape)
def absolute_difference(tensors):
    return K.abs(tensors[0] - tensors[1])

# Create a custom object scope to handle the Lambda layer
def absolute_difference_output_shape(input_shape):
    # The output shape of the Lambda layer will be the same as the input shape (i.e., (No
    return input_shape[0],

# Load the trained model with custom objects
model = load_model(
    '/kaggle/input/kd-final-7/tensorflow2/default/1/keystroke_authentication_model.h5',
    custom_objects={
        'absolute_difference': absolute_difference,
        'absolute_difference_output_shape': absolute_difference_output_shape
    }
)
```

```python
from tensorflow.keras.models import Model

# Redefine the model to output embeddings from the Lambda layer
embedding_model = Model(inputs=model.input, outputs=model.get_layer('lambda').output)

# Step 1: Prepare the Input Data for Batch Prediction
# Reshape new_user_data to have shape (n_keystrokes, 1, 12)
new_user_data_reshaped = np.expand_dims(new_user_data, axis=1)  # Shape: (n_keystrokes, 1, 12)

# Step 2: Generate Embeddings for All Keystrokes at Once (Batch Processing)
keystroke_embeddings = embedding_model.predict([new_user_data_reshaped, new_user_data_reshaped])
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 5s 123ms/step
```

```python
# Step 3: Aggregate Embeddings into a Single Reference Embedding
# Aggregate the embeddings (mean, for example)
reference_embedding = np.mean(keystroke_embeddings, axis=0)

# Step 4: Include User ID with the Reference Embedding
user_reference_embedding = {'user_id': user_id, 'embedding': reference_embedding}

# Print the user reference embedding
print(f"User ID: {user_reference_embedding['user_id']}")
print(f"Reference Embedding Shape: {user_reference_embedding['embedding'].shape}")
# print(f"Reference Embedding: {user_reference_embedding['embedding']}")

# Step 5: Save the Reference Embedding with User ID
np.save(f'/kaggle/working/reference_embedding_user_{user_id}.npy', user_reference_embedding)
print("Reference embedding with User ID saved!")

np.save(f'reference_embedding_user_{user_id}.npy', user_reference_embedding)
```

```
User ID: 1002
Reference Embedding Shape: (64,)
Reference embedding with User ID saved!
```

Project Progress Completion

- Authentication Process

```python
# Redefine the model to output embeddings from the Lambda layer
embedding_model = Model(inputs=model.input, outputs=model.get_layer('lambda').output)

# Function to load the reference embedding for a specific user
def load_reference_embedding(user_id):
    # Load the saved reference embedding file for the user
    try:
        reference_embedding = np.load(f'/kaggle/working/reference_embedding_user_{user_id}.npy', allow_pickle=True).item()
        print(f"Reference embedding for user {user_id} loaded successfully!")
        return reference_embedding
    except FileNotFoundError:
        print(f"Reference embedding for user {user_id} not found!")
        return None

# Function to generate the embedding for a given login attempt
def generate_login_embedding(login_data):
    # Assuming login_data is in the form of an ndarray with shape (n_keystrokes, 12 features)

     login_data_reshaped = np.expand_dims(login_data, axis=1)  # Reshape to (n_keystrokes, 1, 12)

    # Generate the embedding for the login attempt
    login_embedding = embedding_model.predict([login_data_reshaped, login_data_reshaped])

    # Aggregate the embeddings (mean, for example)
    login_embedding = np.mean(login_embedding,axis=0)

    return login_embedding
```

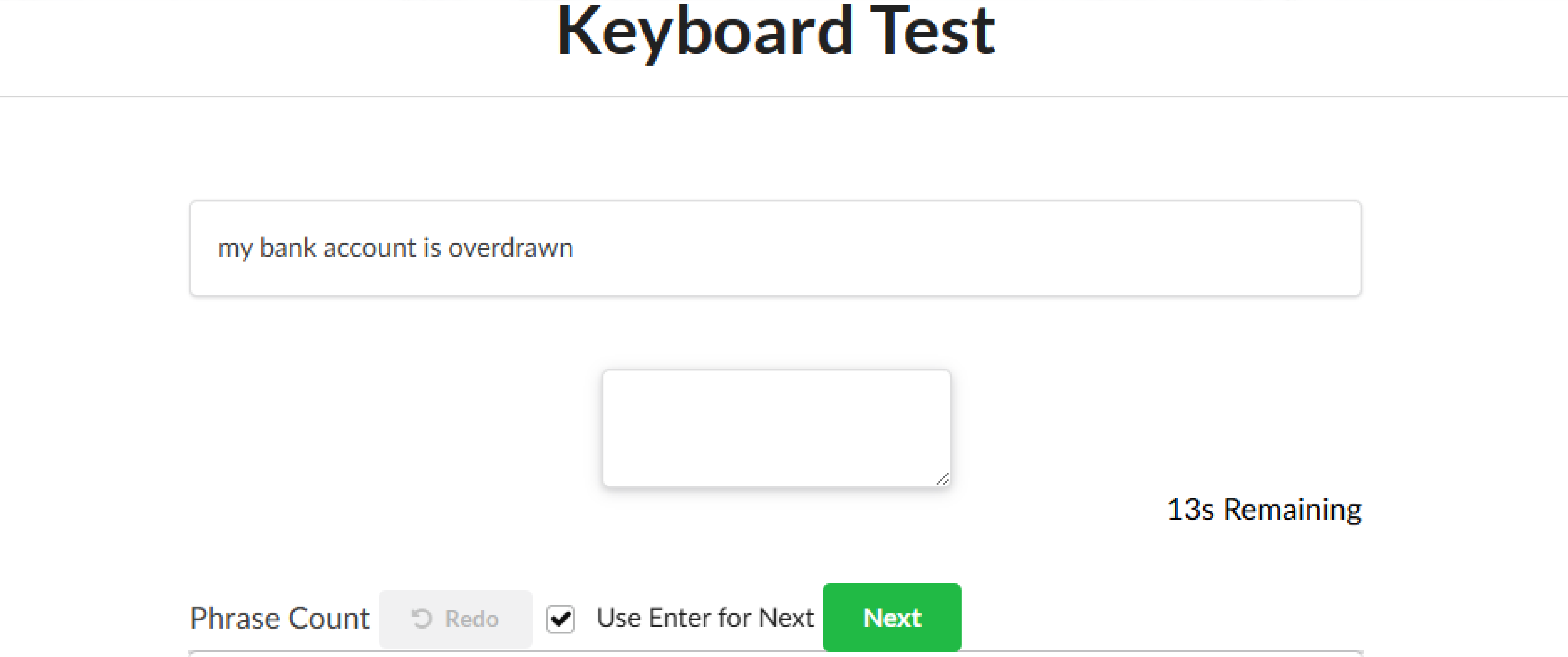
```python
# Function to compare embeddings using cosine similarity
def compare_embeddings(reference_embedding, login_embedding):
    # Print shapes for debugging
    print(f"Reference embedding shape: {reference_embedding.shape}")
    print(f"Login embedding shape: {login_embedding.shape}")

    # Check if the embeddings have the same shape
    if reference_embedding.shape != login_embedding.shape:
        print(f"Shape mismatch! Reference embedding shape: {reference_embedding.shape}, Login embedding shape: {login_embedding.shape}")

    # Compute cosine similarity
    similarity = cosine_similarity([reference_embedding], [login_embedding])[0][0]

    return similarity

# Set a similarity threshold for authentication
SIMILARITY_THRESHOLD = 0.80  # Adjust this threshold based on your validation
```

```python
    # Generate the login attempt embedding
    login_embedding = generate_login_embedding(login_data)

    # Step 3: Compare the Reference Embedding with the Login Attempt Embedding
    similarity_score,distance_score  = compare_embeddings(reference_embedding, login_embedding)
    print(f"Similarity Score: {similarity_score}\n")

    # Step 4: Authentication Decision
    if similarity_score >= SIMILARITY_THRESHOLD:
        print("Authentication successful!")
    else:
        print("Authentication failed!")
```

```
Reference embedding for user 1002 loaded successfully!
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Reference embedding shape: (64,)
Login embedding shape: (64,)
Similarity Score: 0.9685071706771851
```

Project Progress Completion

- User Registration Front End



Progress

PP1 – 50%

- Dataset acquired and preprocessed.
- Model architecture coded (Bi-LSTM).
- Model training initiated and initial results gathered (accuracy, precision, recall, F1-score, and AUC).

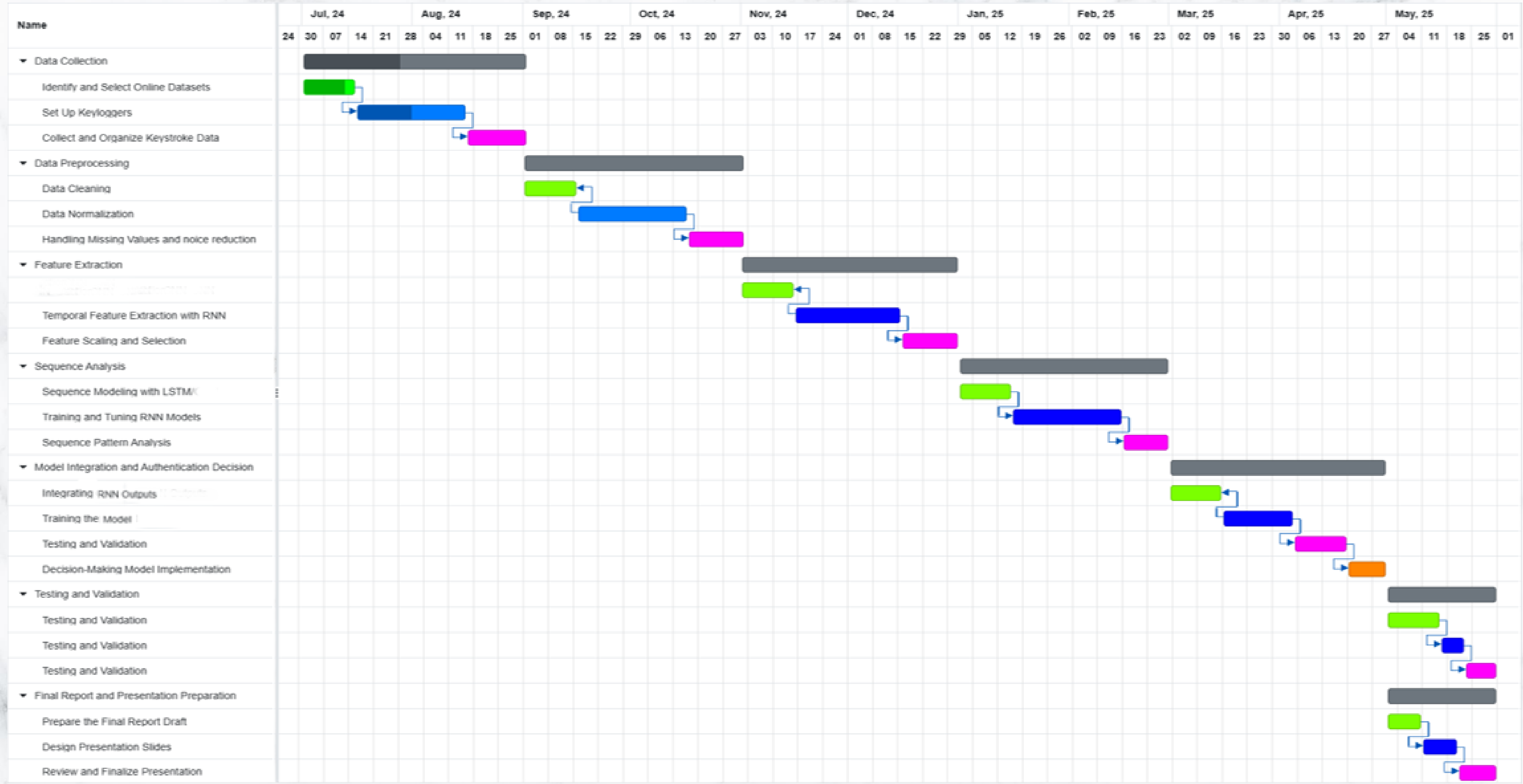
PP2 – 90%

- Train model to achieve acceptable accuracy ,F1 score and AUC.
- Validate model performance with real-world data
- User Embedding and Authentication Process Coded

Final – 100%

- Complete frontend development and user interface.
- Securely store the User embedded and User data
- Finalize integration of all components.
- Compile and submit the final project report.

Project Timeline: Gantt Chart



REFERENCES

- Aditya Arsh, Nirmalya Kar , and Subhrajyoti Deb , "Multiple Approaches Towards Authentication Using Keystroke Dynamics," 2024.
- Rashik Shadman, Ahmed Anu Wahab, Michael Manno, Matthew Lukaszewski, Daqing Hou, Faraz Hussain, “Keystroke Dynamics: Concepts, Techniques, and Applications” ,2024.
- Yutong Shi, Xiujuan Wang, Kangfeng Zheng, “User authentication method based on keystroke dynamics and mouse dynamics using HDA”, 2022.

IT21345678 |ANUPAMA K. G. A

BSc (Hons) in Information Technology Specialising in Cyber Security

Introduction to Mouse movement Component

- Mouse movement explores behavioral authentication using mouse movement patterns, aiming to enhance security by leveraging unique user behaviors. Machine learning models, including Siamese networks with one-shot learning, will be used to analyze features like velocity, acceleration, and jerk to determine if two sessions belong to the same user. This approach enables accurate authentication with minimal user data, offering a robust solution for identity verification and fraud prevention.

Component Objectives

- **Develop Efficient One-Shot Learning**: Implement Siamese Network architecture for authentication using minimal data, ensuring quick enrollment and recognition
- **Authentication System Implementation**: Develop a system to authenticate users in real-time based on their mouse movement behavior.
- **Ensure Privacy and Scalability**: Implement efficient user embedding storage and management for secure and scalable deployment

❑ Component Gap

Features/ Technologies	Use of Online Datasets	LSTM for Sequence Modeling	One-Shot Siamese Network for Authentication	Cross-User Authentication via Embedding Matching
Project X	✓	✗	✗	✗
Project Y	✓	✓	✗	✓
Project Z	✗	✗	✗	✗
SecureAuth	✓	✓	✓	✓

Methodology

Data Collection

Collect comprehensive mouse movement data from diverse users across multiple sessions.

Data Preprocessing

Clean, normalize, and label the data to ensure consistency and readiness for model input.

Feature Extraction

Extract and transform key behavioral features such as velocity, acceleration, and jerk for effective analysis.

Model Development and Training

Design and implement a Siamese network to compare user sessions and measure similarity.

Train the model using labeled session pairs, optimizing with a suitable loss function like contrastive loss or binary cross-entropy.

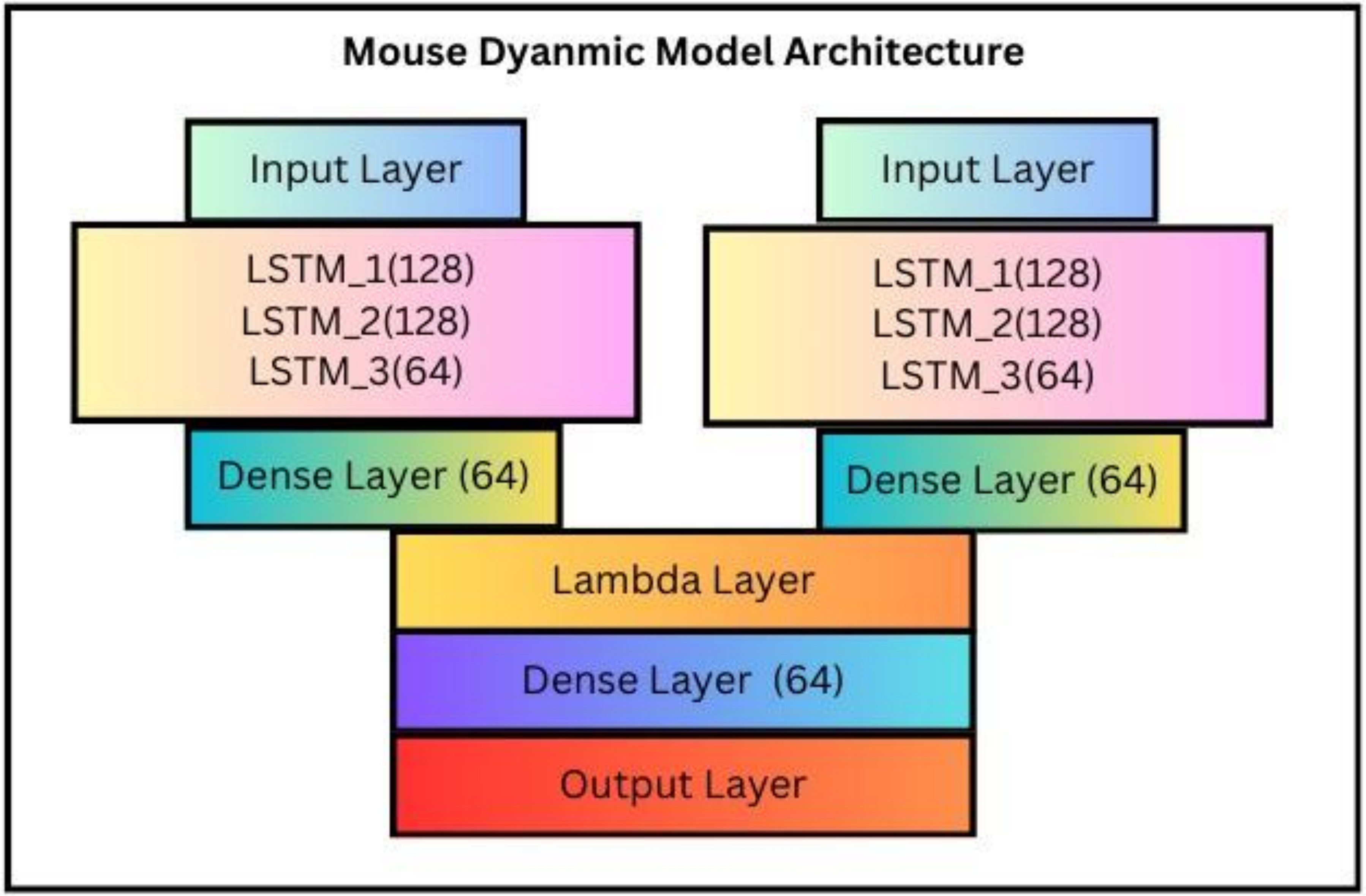
Evaluation and Implementation

Evaluate model performance using key metrics, including accuracy, precision, recall, and F1-score to ensure effectiveness.

Deploy the trained model for real-time, dynamic user authentication based on mouse movement behavior.

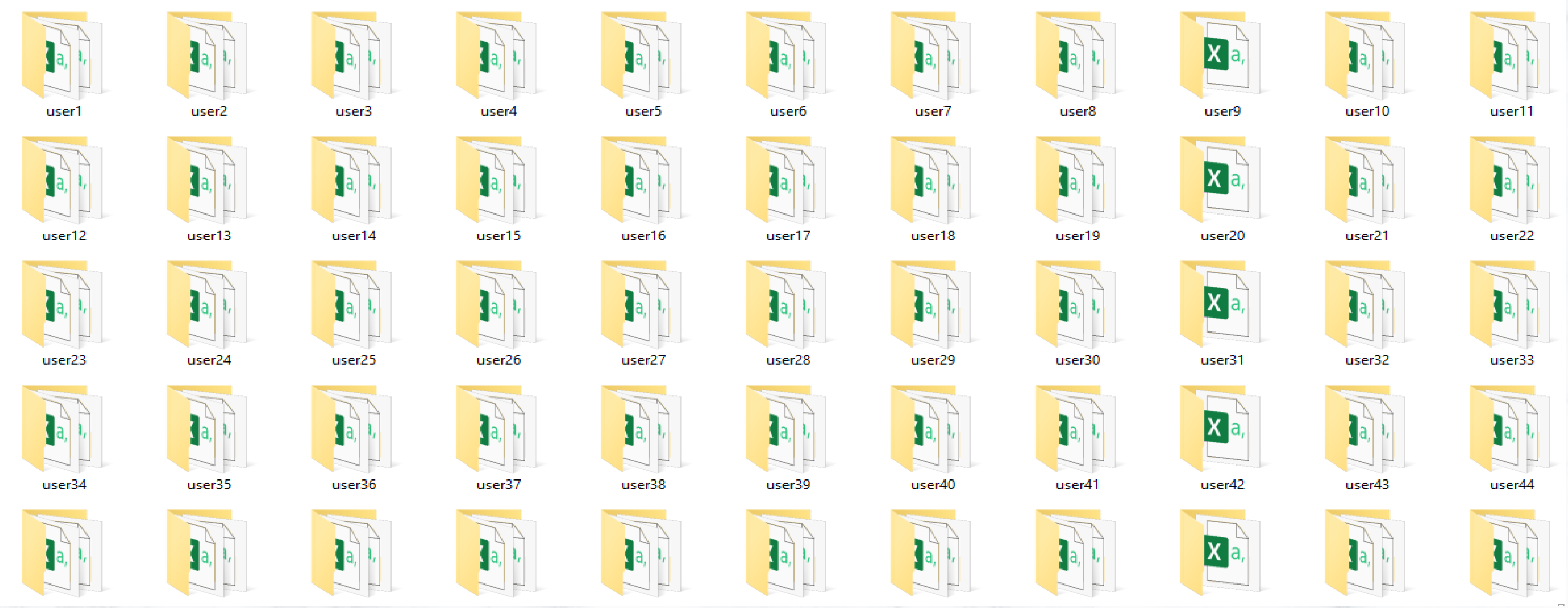
Novelty

- This Component brings a new idea by improving the one-shot Siamese model. It uses an adaptive system that adjusts the similarity threshold based on mouse movements. This helps the model make better decisions for each user, improving accuracy even with small data and changing user behavior.


Mouse Dyanmic Model Architecture
Input Layer
LSTM_1(128)
LSTM_2(128)
LSTM_3(64)
Dense Layer (64)
Input Layer
LSTM_1(128)
LSTM_2(128)
LSTM_3(64)
Dense Layer (64)
Lambda Layer
Dense Layer (64)
Output Layer

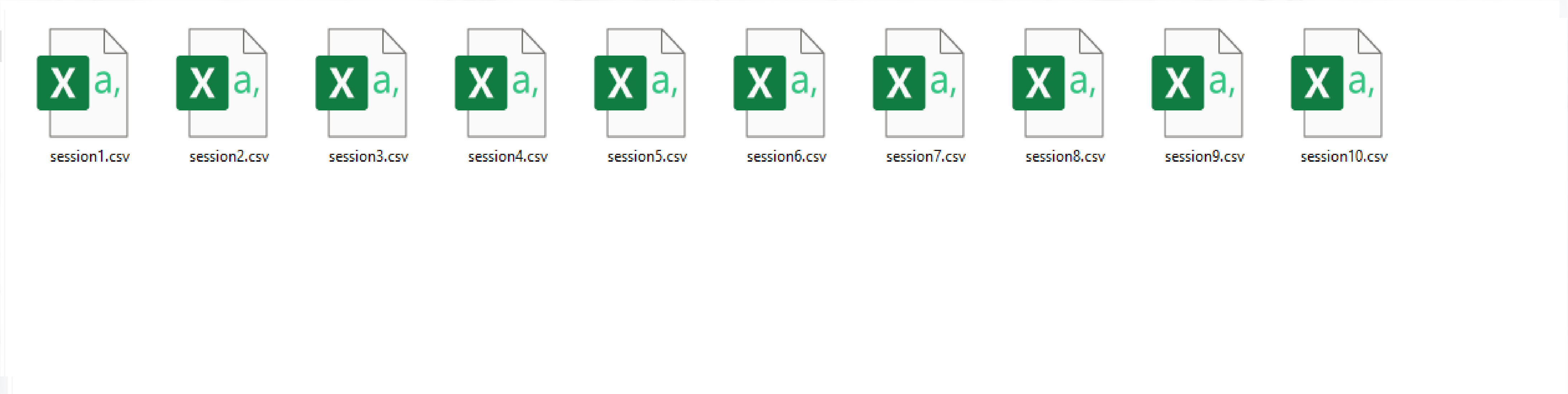
Project Progress Completion

- Dataset



Project Progress Completion

- Each user have Multiple Sessions



Project Progress Completion

- Training and Session Pairs

session_1_path	session_2_path	label
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user3\session10.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user4\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session4.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user2\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user7\session10.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session8.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session9.csv	1
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user7\session7.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session3.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session4.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session6.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user8\session5.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user2\session1.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user7\session10.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session6.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session10.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user2\session6.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user8\session1.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session2.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session2.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user5\session7.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session2.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user2\session1.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session8.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session3.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session8.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user5\session8.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user2\session8.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user10\session5.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session5.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user11\session4.csv	0
D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user1\session1.csv	D:\Workspace\OneDrive - eBEYONDS (PVT) LTD\Documents\Reasearch\Userss2\user6\session2.csv	0

Project Progress Completion

client timestamp	button	state	x	y	distance moved	velocity_x	velocity_y	velocity	acceleration	path efficiency	jerk	angle	user
30933	NoButton	Move	740	608	13.6	-0.25	-0.81	0.85	-0.71	0.99	-38.856875	-107.1	1
30950	NoButton	Move	734	591	18.03	-0.35	-1	1.06	-0.9	1	-35.81764706	-109.44	1
30968	NoButton	Move	728	573	18.97	-0.33	-1	1.05	-0.88	1	-32.88222222	-108.43	1
30983	NoButton	Move	724	555	18.44	-0.27	-1.2	1.23	-0.95	1	-38.26333333	-102.53	1
31000	NoButton	Move	720	540	15.52	-0.24	-0.88	0.91	-0.72	1	-32.68941176	-104.93	1
31233	NoButton	Move	725	532	4.47	0.36	-0.18	0.41	0.18	0.95	-48.52909091	-26.57	1
31250	NoButton	Move	739	524	16.12	0.82	-0.47	0.95	0.39	0.87	-31.27117647	-29.74	1
31267	NoButton	Move	748	516	12.04	0.53	-0.47	0.71	0.16	0.85	-30.81411765	-41.63	1
31283	NoButton	Move	753	505	12.08	0.31	-0.69	0.76	-0.13	0.86	-32.258125	-65.56	1
31300	NoButton	Move	759	488	18.03	0.35	-1	1.06	-0.25	0.87	-29.72058824	-70.56	1
31317	NoButton	Move	764	475	13.93	0.29	-0.76	0.82	-0.16	0.88	-28.71529412	-68.96	1
31333	NoButton	Move	772	453	23.41	0.5	-1.38	1.46	-0.28	0.89	-29.705	-70.02	1
31350	NoButton	Move	776	425	28.28	0.24	-1.65	1.66	-0.61	0.91	-26.68294118	-81.87	1
31366	NoButton	Move	781	386	39.32	0.31	-2.44	2.46	-0.85	0.92	-26.615625	-82.69	1
31383	NoButton	Move	789	347	39.81	0.47	-2.29	2.34	-0.54	0.93	-22.73764706	-78.41	1
31399	NoButton	Move	793	317	30.27	0.25	-1.88	1.89	-0.5	0.94	-21.71875	-82.41	1
31416	NoButton	Move	795	289	28.07	0.12	-1.65	1.65	-0.48	0.94	-18.67529412	-85.91	1
31666	NoButton	Move	786	233	6.08	-0.35	0.06	0.36	-0.32	0.92	-13.66588235	170.54	1

Project Progress Completion

- Model Coding – Convert Data into NumPy array

```python
# Load session data
def load_session_data(session_path):
    session_data = pd.read_csv(session_path)
    selected_columns = ['distance moved','velocity_x', 'velocity_y','velocity','acceleration','path efficiency', 'jerk', 'angle']
    session_data = session_data[selected_columns].values
    return session_data

# Create training data
def create_training_data(pair_csv_path, sequence_length=100):
    pair_data = pd.read_csv(pair_csv_path)
    session_1_data = []
    session_2_data = []
    labels = []

    for _, row in pair_data.iterrows():
        session_1 = load_session_data(row['session_1_path'])
        session_2 = load_session_data(row['session_2_path'])
        label = row['label']

        session_1_windows = [session_1[i:i + sequence_length] for i in range(0, len(session_1) - sequence_length + 1, sequence_length)]
        session_2_windows = [session_2[i:i + sequence_length] for i in range(0, len(session_2) - sequence_length + 1, sequence_length)]

        min_windows = min(len(session_1_windows), len(session_2_windows))
        session_1_data.extend(session_1_windows[:min_windows])
        session_2_data.extend(session_2_windows[:min_windows])
        labels.extend([label] * min_windows)

    session_1_data = tf.keras.preprocessing.sequence.pad_sequences(session_1_data, maxlen=sequence_length, dtype='float32', padding='post')
    session_2_data = tf.keras.preprocessing.sequence.pad_sequences(session_2_data, maxlen=sequence_length, dtype='float32', padding='post')
    labels = np.array(labels, dtype='float32')

    return session_1_data, session_2_data, labels
```

Project Progress Completion

- Model Coding

```python
# Define the Siamese network model
def build_siamese_model(sequence_length, feature_dim):
    """
    Builds and returns a Siamese neural network for behavioral authentication.
    """
    # Input layers for both sessions
    input_1 = Input(shape=(sequence_length, feature_dim), name="Input_Session_1")
    input_2 = Input(shape=(sequence_length, feature_dim), name="Input_Session_2")

 # Shared LSTM layers with separated dropout
    lstm_1 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_1")  # LSTM with 128 units
    lstm_2 = LSTM(128, return_sequences=True, dropout=0.4, name="LSTM_2")  # LSTM with 128 units
    lstm_3 = LSTM(64, return_sequences=False, dropout=0.4, name="LSTM_3")  # LSTM with 64 units

    dense_layer = Dense(64, activation='relu', name="Dense_Layer")

    # Process the input through each layer
    lstm_1_output_1 = lstm_1(input_1)  # Apply lstm_1 to the first input
    lstm_2_output_1 = lstm_2(lstm_1_output_1)  # Apply lstm_2 to the output of lstm_1
    lstm_3_output_1 = lstm_3(lstm_2_output_1)  # Apply lstm_3 to the output of lstm_2
    processed_1 = dense_layer(lstm_3_output_1)  # Apply dense layer to the output of lstm_3

    lstm_1_output_2 = lstm_1(input_2)  # Apply lstm_1 to the second input
    lstm_2_output_2 = lstm_2(lstm_1_output_2)  # Apply lstm_2 to the output of lstm_1
    lstm_3_output_2 = lstm_3(lstm_2_output_2)  # Apply lstm_3 to the output of lstm_2
    processed_2 = dense_layer(lstm_3_output_2)  # Apply dense layer to the output of lstm_3
```

```python
 # Lambda layer to compute absolute difference between embeddings
def absolute_difference(tensors):
    return K.abs(tensors[0] - tensors[1])

distance = Lambda(absolute_difference)([processed_1, processed_2])

 # new dense layer after the Lambda layer
dense_after_lambda = Dense(64, activation='relu', name="Dense_After_Lambda")(distance)

# Output layer
output = Dense(1, activation='sigmoid', name="Output_Layer")(dense_after_lambda)

# Build the model
model = Model(inputs=[input_1, input_2], outputs=output)
return model
```

Project Progress Completion

- Model Architecture

```
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metrics=['accuracy'])

# Print model summary
print("Model Architecture Summary:")
model.summary()

# Train the model
history = model.fit(
    [train_session_1_data, train_session_2_data], train_labels,
    validation_split=0.1,  # Optionally use part of training data for validation
    epochs=20,
    batch_size=32
)

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate([test_session_1_data, test_session_2_data], test_labels)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")

# Save the trained model
model_save_path = r"/kaggle/working/new_2.h5"
model.save(model_save_path)
print(f"Model saved as '{model_save_path}'")
```

IT21336072 | R.P.K.D RAJAPAKSHA

BSc (Hons) in Information Technology Specializing in Cyber Security

Introduction to Voice Component

- Voice authentication leverages the unique characteristics of an individual's voice for secure user identification. By analyzing features such as pitch, tone, and speaking patterns, the system can verify identities with high accuracy. Voice biometrics, combined with deep learning techniques like Triplet Networks, provides robust protection against spoofing attacks and ensures secure access to sensitive systems. By integrating voice authentication with other methods, this technology enhances security in applications demanding advanced and multifactor authentication.

Research Sub-Objectives

Data Preprocessing

- Develop an efficient pipeline for converting raw audio files into clean,denoised data suitable for analysis

Extract Features

- Improve techniques extracting relevant voice features

Model Design

- Evaluate the effectiveness of the network in distinguishing between similar and dissimilar voice samples

Pair generation

- Design an efficient mechanism for generating positive and negative spectrogram pairs for training

Research Question

- How can we handle variations in voice caused by emotional states, health conditions, or environmental factors in a CNN-based voice authentication system?

- What is the impact of using spectrograms in voice authentication compared to traditional waveform analysis for CNN-based models?

Solution

- We preprocess audio to reduce noise and standardize speech, ensuring consistency despite emotional or health variations. The CNN is trained on diverse data, using augmentation techniques to simulate different conditions, making the system robust to these variations while maintaining high accuracy.

- Spectrograms capture time-frequency features like pitch and formants, which are crucial for accurate voice identification. By using spectrograms, the CNN model learns better spatial patterns, improving performance over traditional waveform analysis, especially in noisy environments.

Methodology

Data Collection

Collect a dataset of a 3000 user voice samples

1

Data Preprocessing

Utilize Librosa or PyDub for noise reduction, normalization, and segmentation of voice data.

2

Feature Extraction

Apply CNNs to the spectrograms to extract key features that represent the unique characteristics of each speaker's voice

3

Model Development

Develop the CNN model to process the spectrograms and classify the voice data, The model consists of multiple convolutional layers followed by pooling layers to capture hierarchical features from the spectrograms

4

Training and Evaluation

Split the dataset into training, validation, and testing subsets. Use cross validation techniques to avoid overfitting and ensure the model generalizes well to unseen data

Evaluate model performance using metrics such as accuracy, F1 score.

Novelty

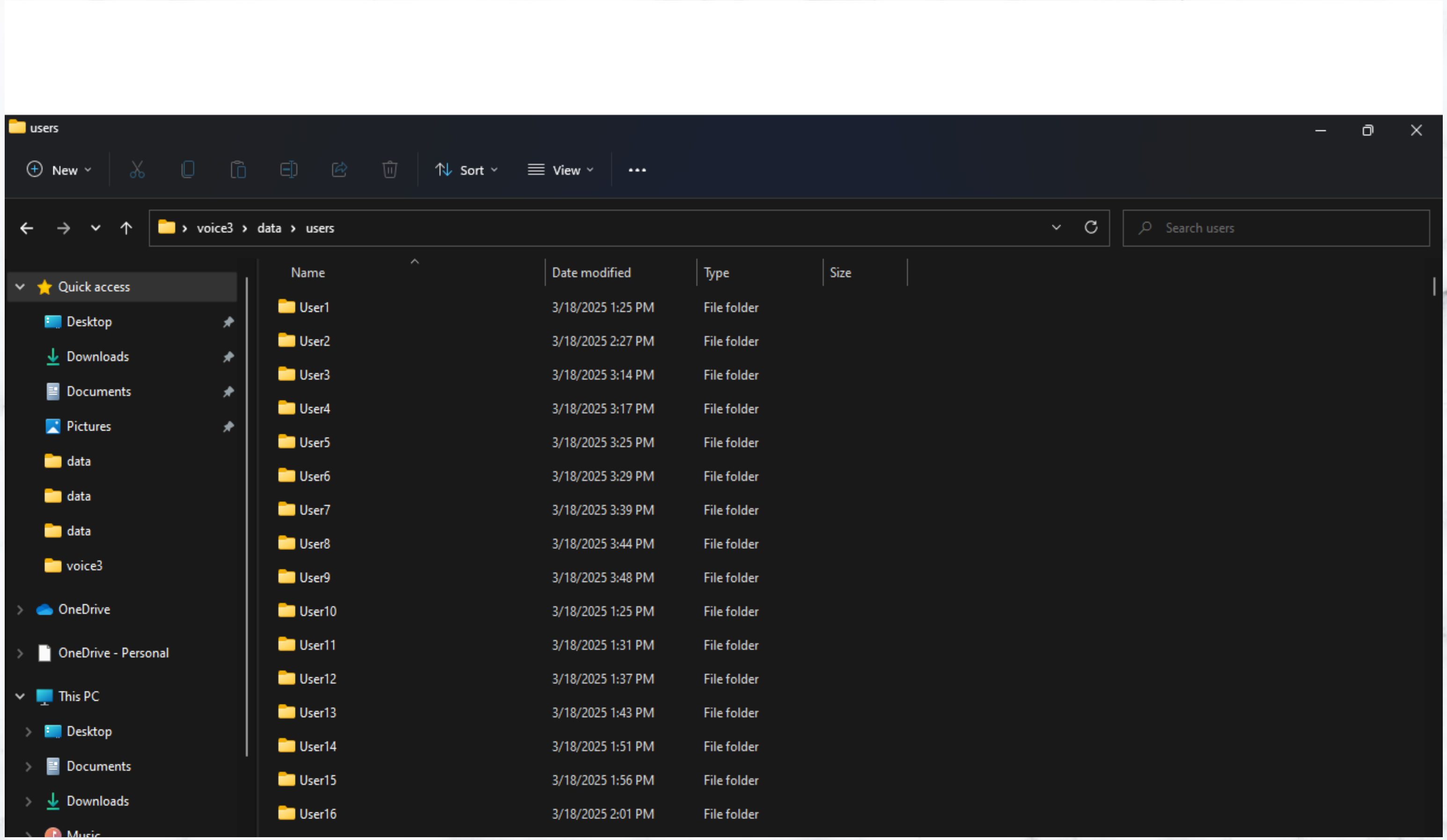
- This system combines Triplet Networks with spectrogram-based input for one-shot learning, enabling efficient voice authentication with minimal data. By analyzing detailed spatial-temporal features from spectrograms, it captures unique voice characteristics like pitch, tone, and cadence. The Triplets Network compares voice samples directly, learning to verify identity with just one sample per user, reducing training data requirements.

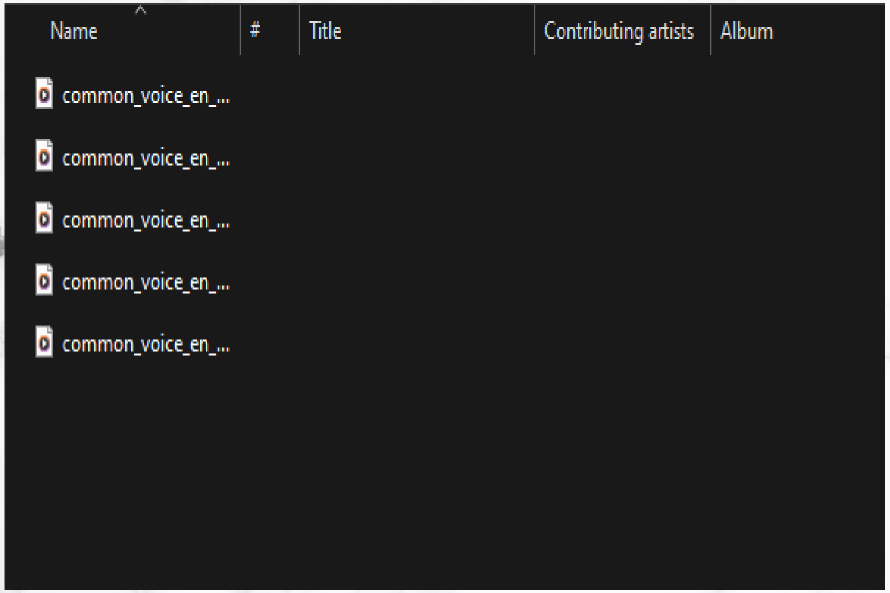
Project Progress Completion

- Online Dataset

Project Progress Completion

- Categorized Dataset





Project Progress Completion

- Create Triplets

```python
import os
import random
import csv

def generate_triplets(spectrogram_folder, output_csv):
    users = {}
    print("Scanning spectrogram folder...")

    # Collect all users and their spectrogram files
    for root, _, files in os.walk(spectrogram_folder):
        session = os.path.basename(root)
        spectrogram_files = [os.path.join(root, f) for f in files if f.endswith('.npy')]
        if spectrogram_files:
            users[session] = spectrogram_files
            print(f"Found {len(spectrogram_files)} spectrograms for {session}")

    # Check if we have enough users for triplets
    if len(users) < 2:
        print(" Not enough users to generate triplets.")
        return

    triplets = []
    print("Generating triplets...")

    # Iterate through each user and create triplets
    for user, files in users.items():
        if len(files) < 2:
            print(f"Skipping user {user}: not enough spectrograms.")
            continue

        # Select anchor and positive spectrograms
        anchor = random.choice(files)
        positive = random.choice([f for f in files if f != anchor])

        # Select a valid negative user
        valid_negative_users = [u for u in users if u != user and len(users[u]) > 0]
        if valid_negative_users:
            negative_user = random.choice(valid_negative_users)
            negative = random.choice(users[negative_user])
            triplets.append([anchor, positive, negative])
        else:
            print(f"Skipping user {user}: no valid negative pairs found.")

    # Save the generated triplets to a CSV file
    if triplets:
        with open(output_csv, 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerow(['anchor', 'positive', 'negative'])
            writer.writerows(triplets)
        print(f" Successfully saved {len(triplets)} triplets to {output_csv}")
    else:
        print(" No valid triplets generated.")

if __name__ == "__main__":
    spectrogram_folder = "C:/Users/Hp/Desktop/voice3/data/spectro"  # Update with your folder path
    output_csv = "C:/Users/Hp/Desktop/voice3/data/triplets.csv"  # Update with your output file path
    generate_triplets(spectrogram_folder, output_csv)
```

Project Progress Completion

- Model Coding

voice3 > train_model.py > ...

```python
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, Model # type: ignore

# Load triplet data and ensure all spectrograms have the same shape
def load_data(triplet_csv, target_shape=(100, 100)):
    triplets = pd.read_csv(triplet_csv)

    # Fix Windows paths by replacing `\\` with `/`
    triplets["anchor"] = triplets["anchor"].str.replace("\\", "/")
    triplets["positive"] = triplets["positive"].str.replace("\\", "/")
    triplets["negative"] = triplets["negative"].str.replace("\\", "/")

    def load_and_resize(file_path):
        array = np.load(file_path)
        if array.shape != target_shape:
            from skimage.transform import resize
            array = resize(array, target_shape, anti_aliasing=True)
        return array

    anchor = np.array([load_and_resize(file) for file in triplets["anchor"]])
    positive = np.array([load_and_resize(file) for file in triplets["positive"]])
    negative = np.array([load_and_resize(file) for file in triplets["negative"]])

    return [anchor, positive, negative]

# Triplet Loss Function
def triplet_loss(y_true, y_pred, alpha=0.2):
    anchor, positive, negative = y_pred[:, 0], y_pred[:, 1], y_pred[:, 2]
    pos_dist = tf.reduce_sum(tf.square(anchor - positive), axis=-1)
```

voice3 > train_model.py > ...

```python
        return tf.reduce_mean(loss)

# Triplet Accuracy Function
def triplet_accuracy(y_true, y_pred):
    anchor, positive, negative = y_pred[:, 0], y_pred[:, 1], y_pred[:, 2]
    pos_dist = tf.reduce_sum(tf.square(anchor - positive), axis=-1)
    neg_dist = tf.reduce_sum(tf.square(anchor - negative), axis=-1)
    return tf.reduce_mean(tf.cast(pos_dist < neg_dist, tf.float32))

# Create CNN-based Feature Extractor
def create_network(input_shape):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.GlobalAveragePooling2D()(x)
    outputs = layers.Dense(256, activation='relu')(x)
    return Model(inputs, outputs, name="FeatureExtractor")

# Train the Model
def train_model(triplet_csv, input (parameter) input_shape: Any ), batch_size=32):
    base_network = create_network(input_shape)

    anchor_input = layers.Input(shape=input_shape, name="Anchor")
    positive_input = layers.Input(shape=input_shape, name="Positive")
    negative_input = layers.Input(shape=input_shape, name="Negative")
```

Project Progress Completion

- Model Coding

voice3 > train_model.py > ...

```python
# Train the Model
def train_model(triplet_csv, input_shape=(100, 100, 1), epochs=20, batch_size=32):
    base_network = create_network(input_shape)

    anchor_input = layers.Input(shape=input_shape, name="Anchor")
    positive_input = layers.Input(shape=input_shape, name="Positive")
    negative_input = layers.Input(shape=input_shape, name="Negative")

    anchor_embedding = base_network(anchor_input)
    positive_embedding = base_network(positive_input)
    negative_embedding = base_network(negative_input)

    stacked_embeddings = layers.Lambda(lambda x: tf.stack(x, axis=1), output_shape=(3, 256))([anchor_embedding, positive_em

    model = Model(inputs=[anchor_input, positive_input, negative_input], outputs=stacked_embeddings)
    model.compile(optimizer='adam', loss=triplet_loss, metrics=[triplet_accuracy])

    # Load training data and resize spectrograms
    data = load_data(triplet_csv)
    model.fit(data, np.zeros((len(data[0]),)), batch_size=batch_size, epochs=epochs, validation_split=0.2)

    # Save the trained model
    model.save("voice_auth_model.h5")
    print("✅ Model trained and saved successfully.")

if __name__ == "__main__":
    train_model("data/triplets.csv")
```

Project Progress Completion

- Model Training

voice3 > train_model.py > ...

```python

# Train the Model
def train_model(triplet_csv, input_shape=(100, 100, 1), epochs=20, batch_size=32):
    base_network = create_network(input_shape)

    anchor_input = layers.Input(shape=input_shape, name="Anchor")
    positive_input = layers.Input(shape=input_shape, name="Positive")
    negative_input = layers.Input(shape=input_shape, name="Negative")

    anchor_embedding = base_network(anchor_input)
    positive_embedding = base_network(positive_input)
    negative_embedding = base_network(negative_input)

    stacked_embeddings = layers.Lambda(lambda x: tf.stack(x, axis=1), output_shape=(3, 256))([anchor_embedding, positive_em

    model = Model(inputs=[anchor_input, positive_input, negative_input], outputs=stacked_embeddings)
    model.compile(optimizer='adam', loss=triplet_loss, metrics=[triplet_accuracy])

    # Load training data and resize spectrograms
    data = load_data(triplet_csv)
    model.fit(data, np.zeros((len(data[0]),)), batch_size=batch_size, epochs=epochs, validation_split=0.2)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
Epoch 1/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 318s 4s/step - loss: 1.7454 - triplet_accuracy: 0.5134 - val_loss: 0.2254 - val_triplet_accuracy: 0.5576
Epoch 2/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 324s 4s/step - loss: 0.1782 - triplet_accuracy: 0.5882 - val_loss: 0.1767 - val_triplet_accuracy: 0.6431
Epoch 3/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 318s 4s/step - loss: 0.1748 - triplet_accuracy: 0.6167 - val_loss: 0.1569 - val_triplet_accuracy: 0.6803
Epoch 4/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 329s 4s/step - loss: 0.1530 - triplet_accuracy: 0.6751 - val_loss: 0.1472 - val_triplet_accuracy: 0.7072
```

Ln 86, Col 1 Spaces: 4 UTF-8 CRLF {} Python

Project Progress Completion

- Model Training

```
    # Load training data and resize spectrograms
    data = load_data(triplet_csv)
    model.fit(data, np.zeros((len(data[0]),)), batch_size=batch_size, epochs=epochs, validation_split=0.2)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

```
Epoch 19/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 280s 4s/step - loss: 0.0724 - triplet_accuracy: 0.8594 - val_loss: 0.2579 - val_triplet_accuracy: 0.6704
Epoch 20/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 280s 4s/step - loss: 0.0690 - triplet_accuracy: 0.8687 - val_loss: 0.0796 - val_triplet_accuracy: 0.8520
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consider
end using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
✅ Model trained and saved successfully.
PS C:\Users\Hp\Desktop\voice3>
```

Ln 86, Col 1 Spaces: 4 UTF-8 CRLF {} P

Progress

PP1 – 50%	PP2 – 90%	Final – 100%
• Dataset acquired and preprocessed. • Model architecture coded (CNN) • Model training initiated	• Train model to achieve acceptable accuracy and F1 score. • Validate model performance with real-world data.	• Complete frontend development and user interface. • Finalize integration of all components. • Compile and submit the final project report.

Future Interactions For 90% Phase

Model Optimization

- Fine-tune the Triplet network to improve accuracy, focusing on reducing f1 score.

Validation

- Test with real-world data and integrate with other biometric models.

Frontend Preparation

- Plan and prepare for seamless frontend development.

REFERENCES

- Kinnunen, T., & Li, H. (2010). "An overview of text-independent speaker recognition: From features to supervectors". Speech Communication, 52(1), 12-40.
- Graves, A., et al. (2013). "Speech recognition with deep recurrent neural networks". IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP).
- Sainath, T. N., et al. (2015). "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks". IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).