ECG Classification based on Time-Frequency Analysis and Deep Learning

By: Mosab Aboidrees Altraifi Yousif and Abdelrahman Mohammed (Group 12)

Deployment

1. Overview

The deployment phase of this machine learning project focuses on making the ECG Classification System based on time-frequency analysis and deep learning available for real-world use. The system analyzes ECG signals using a Continuous Wavelet Transform (CWT) and a pre-trained Convolutional Neural Network (CNN) model to classify them into ARR (Arrhythmia), CHF (Congestive Heart Failure), or NSR (Normal Sinus Rhythm) [1].

For now, the system was deployed locally as a web application using **Streamlit [2]**, which provides an interactive interface for users to upload ECG signals, visualize them, and obtain real-time predictions from the model.

2. Model Serialization

The trained CNN model was serialized using **HDF5** format (`.h5`), a standard format for storing neural network models in TensorFlow and Keras. This format was chosen due to its efficiency in handling large amounts of data, portability, and compatibility with many deep learning frameworks [3].

Key considerations during serialization included:

- Ensuring the model's architecture and weights were preserved.
- Minimizing storage size while maintaining accuracy.
- Compatibility with deployment platforms.

The serialized model is loaded during runtime for making predictions on new ECG signals uploaded by the user.

3. Model Serving

The serialized CNN model is served using **Streamlit**, which enables real-time interaction with the model. When a user uploads an ECG signal, the application loads the serialized model and processes the signal by applying CWT to generate a scalogram. This scalogram is passed through the CNN model to make predictions [2].

The choice of **Streamlit** as a deployment platform was based on its simplicity, ease of use for creating interactive web applications, and ability to handle machine learning model integration without complex backend development [2].

4. API Integration

Although the current deployment does not involve a separate API for accessing the model, the web application acts as an interface for users to interact with it [4]. If API integration were required, the following considerations would be made:

- **Endpoints**: An endpoint would be created to accept ECG signals in `.npy` format, perform predictions, and return the classification result.
- **Input Formats**: The API would accept ECG signals as NumPy arrays or JSON payloads containing serialized signal data.
- **Response Formats**: The API would return predictions in JSON format [4], including the predicted class (ARR, CHF, or NSR) and confidence scores.

5. Suggested Security Considerations (To Be Implemented)

In the current version of the deployment, security considerations are minimal. However, suggested methods for enhancing security before the presentation include:

- **Authentication**:
  - **Method**: Implement Google OAuth or another third-party OAuth provider for user authentication. This will allow users to securely log in using their existing credentials (e.g., Google, GitHub) [5].
  - **Benefit**: Protects the application from unauthorized access, ensuring only authenticated users can upload files and run predictions [5].
- **Authorization**:
  - **Method**: Implement role-based access control (RBAC), where certain features (e.g., file upload or model use) are restricted to specific user roles (e.g., admin vs. regular user) [6].
  - **Benefit**: This ensures that different users can access appropriate features based on their roles [6].
- **Encryption**:
  - **Method**: Ensure all client and server communication is encrypted using **SSL/TLS** (HTTPS). In Streamlit this is handled by default [7].
  - **Benefit**: Prevents sensitive data, such as ECG signals and prediction results, from being exposed during transmission over the network [7].

6. Suggested Monitoring and Logging (To Be Implemented)

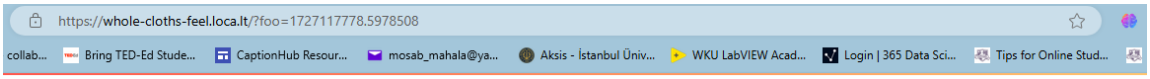
Monitoring the model and logging critical events is crucial to ensure the system's performance and reliability. Suggested methods include:

- **User Activity Logging**:
  - **Method**: Implement logging of user activities such as login attempts, file uploads, and predictions [8].
  - **Benefit**: Provides an audit trail of user interactions, enabling developers to identify any unauthorized access or misuse of the system [8].
- **External Monitoring Tools**:
  - **Method**: Integrate cloud-based monitoring tools such as **AWS CloudWatch**, **Datadog**, or **Prometheus** to track resource usage (e.g., CPU, memory), application health, and performance [9].
  - **Benefit**: These tools provide real-time insights into the system's health and allow for proactive scaling or issue resolution [9].

Implementing these suggested security and monitoring methods will better protect the application against unauthorized access, and the model's performance can be tracked and maintained over time.

Appendix:

Screenshots after deployment:



ECG Classification based on Time-Frequency Analysis and Deep Learning

By: Mosab Aboidrees Altraifi Yousif and Abdelrahman Mohammed (Group 12)

Overview

This system analyzes ECG signals using Continuous Wavelet Transform (CWT) and a pre-trained Convolutional Neural Network (CNN) model. It allows you to upload an ECG signal file in `.npy` format, visualize the signal, generate its time-frequency representation (scalogram) using CWT, and classify the signal into one of the following categories:

- **ARR**: Arrhythmia
- **CHF**: Congestive Heart Failure
- **NSR**: Normal Sinus Rhythm

How to Use:

1. **Upload the Signal**: Click the 'Browse files' button below to upload an ECG signal in `.npy` format from your computer.
2. **View the Signal**: Once uploaded, the first 1000 samples of the ECG signal will be plotted.
3. **Apply CWT**: The system will apply Continuous Wavelet Transform (CWT) to the entire ECG signal and display a scalogram.
4. **Predict the Class**: The system will use a pre-trained CNN model to predict whether the ECG signal belongs to the ARR, CHF, or NSR class.

https://whole-cloths-feel.loca.lt/?foo=1727117778.5978508

collab... Bring TED-Ed Stude... CaptionHub Resour... mosab_mahala@ya... Aksis - İstanbul Üniv... WKU LabVIEW Acad... Login | 365 Data Sci... Tips for Online S

1. Upload the Signal

Choose an ECG signal file (.npy)

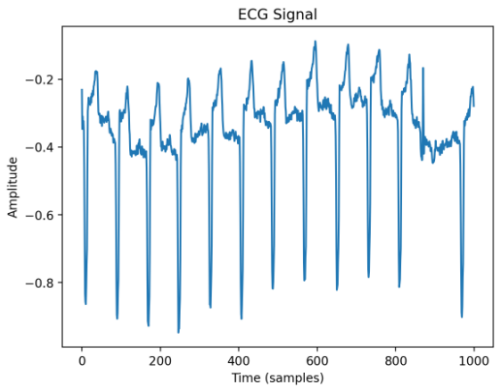
Drag and drop file here
Limit 200MB per file • NPY
Browse files

signal_label_1_index_3.npy 0.5MB

Loading ECG signal...

2. Plotting the ECG Signal

The following plot shows the first 1000 samples of the ECG signal:



https://whole-cloths-feel.loca.lt/?foo=1727117778.5978508

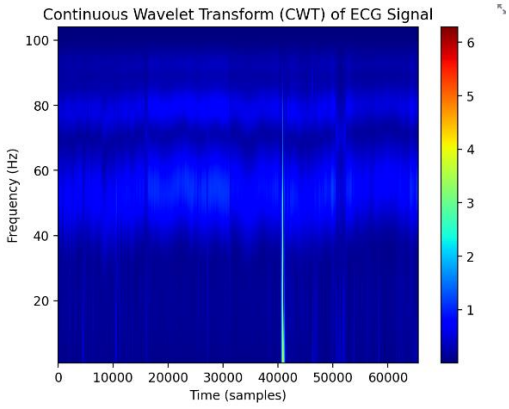
collab... Bring TED-Ed Stude... CaptionHub Resour... mosab_mahala@ya... Aksis - İstanbul Üniv... WKU LabVIEW Acad... Login | 365 Data Sci... Tips for Online Stud

3. Applying Continuous Wavelet Transform (CWT)

Continuous Wavelet Transform (CWT) is a powerful tool for analyzing time-frequency characteristics of signals like ECG. By using a wavelet (in this case, the 'morl' wavelet), we can decompose the ECG signal into various frequency components while retaining the time information. This results in a **scalogram**, a 2D time-frequency representation of the signal.

Applying Continuous Wavelet Transform to the entire ECG signal...

The following scalogram represents the time-frequency distribution of the ECG signal:



4. CNN Prediction of ECG Signal Class

The system uses a pre-trained Convolutional Neural Network (CNN) model to classify the ECG signal. The CNN has been trained on similar ECG data and can predict whether the signal belongs to one of the following categories:

References:

[1] Moody, G. B., and R. G. Mark. "The impact of the MIT-BIH Arrhythmia Database." IEEE Engineering in Medicine and Biology Magazine. Vol. 20. Number 3, May-June 2001, pp. 45–50. (PMID: 11446209)

[2] Richards, T. (2023). Streamlit for Data Science: Create interactive data apps in Python. Packt Publishing Ltd.

[3] Ferguson, M., Jeong, S., Law, K. H., Levitan, S., Narayanan, A., Burkhardt, R., ... & Lee, Y. T. T. (2019, August). A standardized representation of convolutional neural networks for reliable deployment of machine learning models in the manufacturing industry. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 59179, p. V001T02A005). American Society of Mechanical Engineers.

[4] Bloch, J. (2006, October). How to design a good API and why it matters. In Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (pp. 506-507).

[5] Darwish, M., & Ouda, A. (2015, October). Evaluation of an OAuth 2.0 protocol implementation for web server applications. In 2015 International Conference and Workshop on Computing and Communication (IEMCON) (pp. 1-4). IEEE.

[6] Ferraiolo, D., Cugini, J., & Kuhn, D. R. (1995, December). Role-based access control (RBAC): Features and motivations. In Proceedings of 11th annual computer security application conference (pp. 241-48).

[7] Das, M. L., & Samdaria, N. (2014). On the security of SSL/TLS-enabled applications. Applied Computing and informatics, 10(1-2), 68-81.

[8] Gonen, Y., & Gudes, E. (2011, March). Users tracking and roles mining in web-based applications. In Proceedings of the 2011 Joint EDBT/ICDT Ph. D. Workshop (pp. 14-18).

[9] Giamattei, L., Guerriero, A., Pietrantuono, R., Russo, S., Malavolta, I., Islam, T., ... & Panojo, F. S. (2023). Monitoring tools for DevOps and microservices: A systematic grey literature review. Journal of Systems and Software, 111906.