Using a Neural Network to Identify ECG Anomalies

By Jacob Thieret: S01984343 - Undergraduate

Introduction

Cardiovascular diseases are a significant health concern due to their potential severity and high prevalence. Among these, arrhythmias—which denote irregular heartbeats—can often be elusive to detect and diagnose. The accurate detection of these anomalies in electrocardiogram (ECG) data is therefore a critical task with substantial implications for patient outcomes.

This report focuses on utilizing the power of machine learning, particularly deep learning, to detect such anomalies in ECG data. We employ Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), as the cornerstone of our anomaly detection system. We also demonstrate a popular anomaly detection algorithm called Isolation Forrest, to see how it can compare to the LSTM network.

Motivation and Significance

Cardiovascular diseases (CVDs) are the number one cause of death globally, taking an estimated 17.9 million lives each year according to the World Health Organization (WHO) [1]. A large number of these deaths can be prevented through early detection and management of cardiac arrhythmias, anomalies in heart rhythms that may be indicative of an underlying heart condition.

The Electrocardiogram (ECG) is a fundamental tool used in the medical field to monitor the electrical heart activity and detect such anomalies. However, the manual interpretation of ECG recordings can be time-consuming, requiring expertise that may not be readily available, especially in regions that have fewer medically trained professionals on hand. Additionally, the sheer volume of data that needs to be processed presents another challenge. Heart diseases like Myocardial Infarction, AV Block, Ventricular Tachycardia and Atrial Fibrillation could be diagnosed from ECG signals, in which nearly 300 million ECG’s being recorded annually (Hedèn et al., 1996)[3]. Also, with 24-hour monitor tests now becoming more common, more and more data is being recorded and it is becoming exceedingly difficult to manually process these recordings.

Automated anomaly detection in ECG data has the potential to revolutionize healthcare outcomes. Early detection of cardiac arrhythmias could allow for prompt and proactive management, potentially reducing the incidence of severe cardiac events and associated mortality. This not only would result in better patient outcomes but could also substantially reduce the healthcare costs associated with the treatment of advanced CVDs.

Methodology

Data Acquisition and Preprocessing

Data serves as the foundation of any machine learning project. In this study, we utilized the MIT-BIH Arrhythmia Database—a widely used resource in ECG studies. The database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, each providing an abundance of ECG signal data. The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10-mV range. Two or more cardiologists independently annotated each record, with approximately 110,000 annotations in total (Moody and Mark)[4].

The ECG signals are preprocessed by reshaping them into windows of 3600 samples each. Subsequently, we annotate each of these windows based on the presence of any abnormal beats as indicated by the expert annotations. The processed ECG data and their corresponding labels are then compiled into two large NumPy arrays for further analysis.

Data Normalization

Before feeding this data to the model, we performed data normalization. This process involved standardizing the ECG signals to have zero mean and unit variance. The normalization helps make the model more efficient and ensures that the range of the values in the data does not negatively impact the training process.

```
mean = np.mean(data, axis=(0, 1))
std = np.std(data, axis=(0, 1))
data = (data - mean) / std
```

Data split

Class imbalance, where one class of data vastly outnumbers the other, is a common issue in machine learning tasks. In our case, the majority of the ECG segments are normal, with only a small portion showing anomalies. This imbalance can bias the model towards the majority class, reducing its ability to detect anomalies.

To address this, we employed Synthetic Minority Over-sampling Technique (SMOTE) to oversample the minority class in the training data, resulting in a balanced dataset. It's important to note that SMOTE was only applied to the training data to prevent information leak from the validation/test sets to the training set.

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2,
random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25,
random_state=1)
smote = SMOTE()
X_train_resampled, y_train_resampled = smote.fit_resample(X_train.reshape(X_train.shape[0], -
1), y_train)
X_train_resampled = X_train_resampled.reshape(-1, X_train.shape[1], X_train.shape[2])
```

Why LSTMs for ECG Anomaly Detection?

ECG data is essentially a time series, with each heartbeat following the previous one, influenced by its past states and influencing the ones yet to come. LSTM networks are designed to handle such time series data effectively by considering the temporal sequence of the data. They achieve this by leveraging internal states and 'gates' to control the flow of information, thus capturing long-term dependencies that traditional RNNs might miss.

LSTMs can learn to recognize complex patterns over time, which means it can potentially learn the patterns associated with normal heartbeats and thus become proficient in identifying heartbeats that deviate from the norm.

This is a model that consists of stacked bidirectional long short-term memory (LSTM) layers, dropout layers for regularization, and a time-distributed dense layer. The below architecture is a common one for sequence-to-sequence prediction problems, such as time series forecasting, natural language processing tasks, etc.

```
model = Sequential([
    Bidirectional(LSTM(64, return_sequences=True), input_shape=(None, 2)),
    Dropout(0.2),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.2),
    TimeDistributed(Dense(1))
])
model.compile(optimizer='adam', loss='mse')
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
bidirectional_6 (Bidirectional)	(None, None, 128)	34304
dropout_6 (Dropout)	(None, None, 128)	0
bidirectional_7 (Bidirectional)	(None, None, 128)	98816
dropout_7 (Dropout)	(None, None, 128)	0
time_distributed_3 (TimeDistributed)	(None, None, 1)	129

Total params: 133,249
Trainable params: 133,249
Non-trainable params: 0

- Bidirectional wrapper is used to apply the LSTM layer forwards and then backwards, helping the LSTM to learn long-range dependencies in the sequence data.
    - The LSTM layer itself has 64 units
    - input_shape=(None, 2) argument indicates that the model can take sequences of arbitrary length (None) with two features per time-step (2)
- Dropout 20% to prevent overfitting
- Another bidirectional LSTM layer, again with 64 units and returning sequences
- Another dropout layer, again dropping out 20% of its inputs
- The output layer of the model. The TimeDistributed wrapper is used to apply the Dense layer to every time-step of the input sequence independently. The Dense layer itself has a single unit and, by default, no activation function (which means it's a linear activation).
- The model is compiled with the Adam optimizer and mean squared error loss

Training method

The model was trained on the resampled training data for up to 50 epochs using a batch size of 32, validated against a separate dataset. An early stopping mechanism was implemented, halting training if there was no improvement in validation performance for 5 consecutive epochs. Additionally, the Model Checkpoint callback was utilized, ensuring the model weights at their best performance were saved.

This strategy, including the use of bidirectional LSTM layers, is anchored on LSTM's proficiency in managing time-series data. LSTMs, by learning and remembering patterns over extended sequences, are ideal for sequence prediction tasks like ECG anomaly detection. Coupled with thorough data preprocessing and resampling to address class imbalance, the method aims to build a robust model, with early stopping and model checkpointing mechanisms mitigating overfitting and ensuring retention of the optimally performing model. The model's true capability to detect unseen ECG anomalies is then assessed using a separate test set.

```
callbacks = [
    EarlyStopping(patience=5),
    ModelCheckpoint('model.h5', save_best_only=True)
]
history = model.fit(X_train_resampled, y_train_resampled, validation_data=(X_val, y_val),
epochs=50, batch_size=32, callbacks=callbacks)
```

Other Techniques Utilized

Isolation Forrest

The second methodology for the ECG anomaly detection project employed an unsupervised learning approach using the Isolation Forest algorithm. This algorithm works by isolating observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

```
# Calculate statistical features (mean and standard deviation) for each window
features = np.array([np.mean(window, axis=0) for window in data])
features = np.hstack((features, np.array([np.std(window, axis=0) for window in data])))
# Standardize the features to have 0 mean and unit variance
scaler = StandardScaler()
features = scaler.fit_transform(features)
# Create an Isolation Forest model
clf = IsolationForest(contamination=0.01)
# Fit the model on the features
clf.fit(features)
# Predict the anomalies in the data
pred = clf.predict(features)
# Check the anomaly score of each window
score = clf.decision_function(features)
anomalies = np.argwhere(pred == -1)
```

This unsupervised learning methodology, using the Isolation Forest algorithm, provides a powerful alternative for ECG anomaly detection. It does not require a balanced dataset like supervised learning methods and can be more robust to changes in the type and structure of the anomalies in the ECG signal.

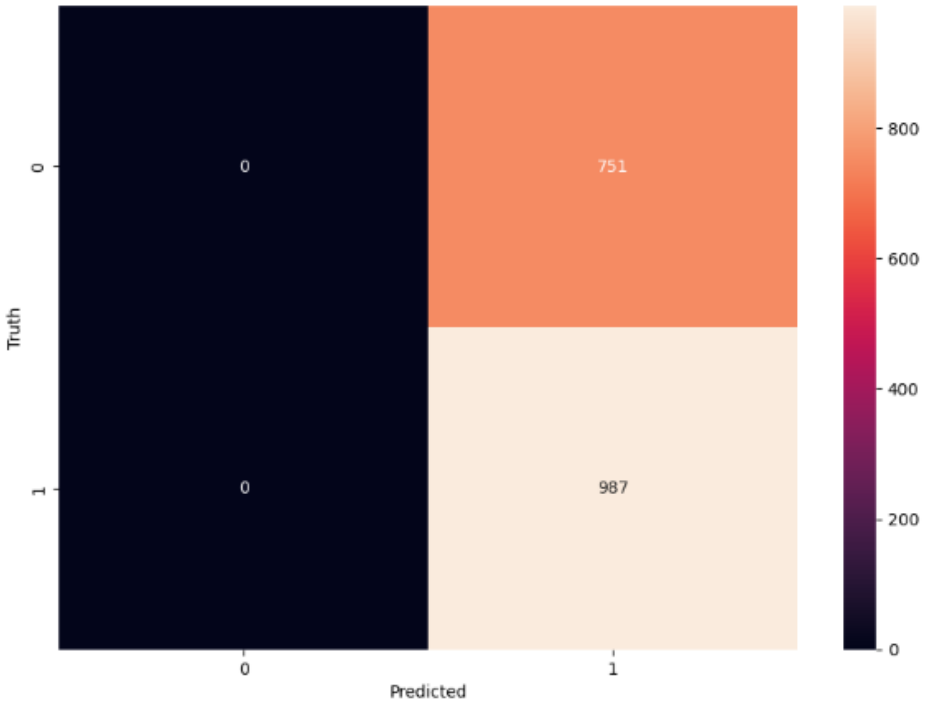
Results from the LSTM Model:

Model Evaluation

```
y_pred = model.predict(X_test)
# The model's output is continuous, but we need binary predictions for the metrics.
y_pred = y_pred.max(axis=1).flatten()
# We can choose a threshold (e.g., 0.5) and classify all instances with an output
above this threshold as anomalies.
y_pred_bin = (y_pred > 0.5).astype(int)
```

Confusion Matrix

```
# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred_bin)
# Plot confusion matrix
plt.figure(figsize=(10,7))
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```



True Negatives (0, 0): These are the cases where the model correctly predicted the negative class (i.e., the absence of an anomaly). Here, this number is 0, which suggests that there were no actual negative cases, or the model failed to correctly predict any negative cases.

False Positives (0, 1): These are the cases where the model incorrectly predicted the positive class (i.e., the presence of an anomaly). Here, this number is 751, which is quite high. This means that the model has incorrectly flagged a lot of instances as anomalies.

False Negatives (1, 0): These are the cases where the model incorrectly predicted the negative class. Here, this number is 0, which suggests that the model did not miss any actual anomalies.

True Positives (1, 1): These are the cases where the model correctly predicted the positive class. Here, this number is 987, which suggests that the model has done well in correctly identifying the anomalies.

Accuracy: This is not applicable here because it assumes an equal cost of false positives and false negatives, which is rarely the case in real-world scenarios. The dataset seems to have too many anomalies per regular heartbeats to train on, seems to be leading to misleading accuracy.

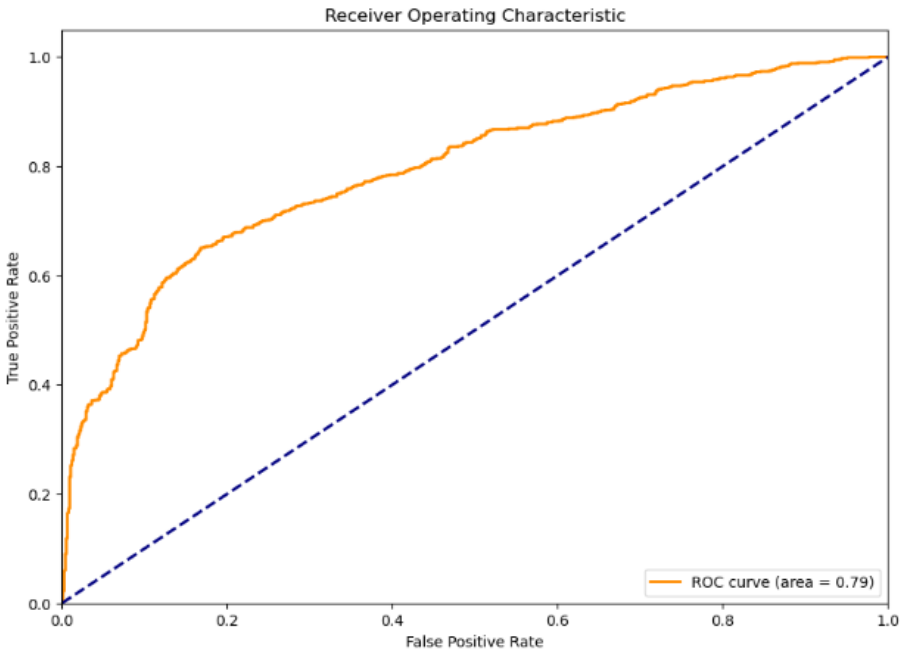
Precision (Positive Predictive Value): This is the ratio of true positives to the sum of true and false positives. Here, it would be 987 / (987 + 751) = 0.57. T Here, it would be 987 / (987 + 751) = 0.57. A precision score of 57% is relatively low for our application, as correctly identifies an anomaly only about 57% of the time.

F1 Score: This is the harmonic mean of precision and recall, and it tries to find the balance between them. Here, the F1 Score would be 2 * (0.57 * 1) / (0.57 + 1) = 0.72. This is a moderate F1 score.

ROC Curve:

```
# Calculate the ROC curve points
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
# Calculate the AUC (area under the ROC curve)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(10,7))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' %
roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

ROC-AUC: This is the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve is created by plotting the true positive rate (recall) against the false positive rate (ratio of false positives to the sum of false positives and true negatives) at various threshold settings. The AUC is the area under this curve, and it ranges from 0 to 1. An AUC of 0.5 suggests no discrimination (i.e., ability to classify correctly), 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding. Here, an AUC of 0.79 suggests acceptable discrimination.



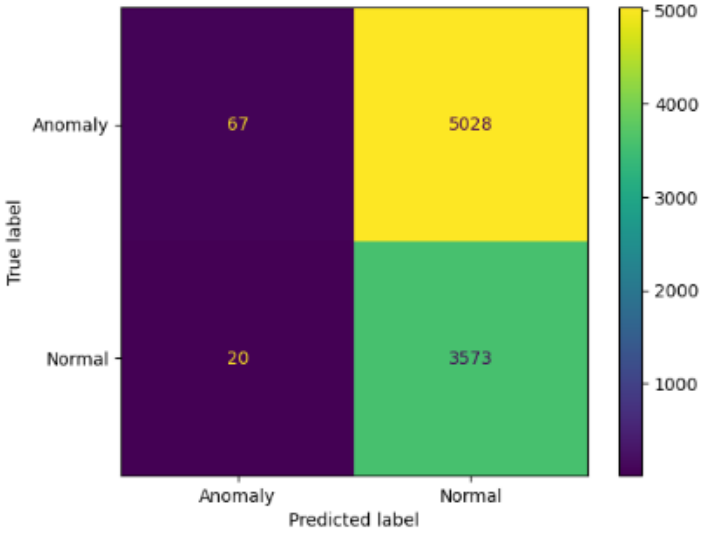
In summary the model seems to perform well at identifying actual anomalies (high recall) but not so good at confirming its predictions (moderate precision). It results in a high number of false positives, which ultimately renders this model useless in a healthcare application. False positives completely defeat the purpose of automatic heart irregularity detection. Professionals need to be able to be confident in the models' warnings. If there are too many alerts to anomalies, then the alerts will not be acknowledged with any sincerity.

Results from the Isolated Forrest Model:

Model Evaluation

```
# Predict the anomalies in the data
pred = clf.predict(features)
# Check the anomaly score of each window
score = clf.decision_function(features)
anomalies = np.argwhere(pred == -1)
```

Confusion Matrix



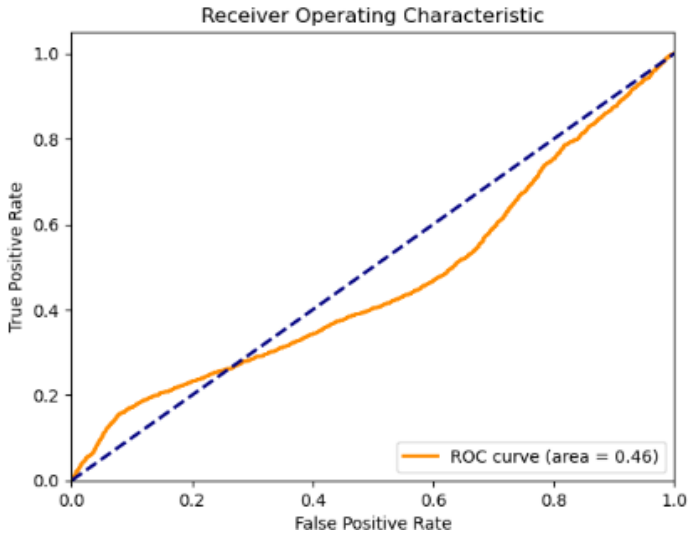
- True Negatives (0, 0): These are the cases where the model correctly predicted the negative class (i.e., the absence of an anomaly). Here, this number is 67, which suggests that the model correctly predicted normal behavior 67 times.
- False Positives (0, 1): These are the cases where the model incorrectly predicted the positive class (i.e., the presence of an anomaly). Here, this number is 5028, which is extremely high. This means that the model has incorrectly flagged a lot of instances as anomalies that were normal behavior.
- False Negatives (1, 0): These are the cases where the model incorrectly predicted the negative class. Here, this number is 20, which suggests that the model missed 20 actual anomalies.
- True Positives (1, 1): These are the cases where the model correctly predicted the positive class. Here, this number is 3573, which suggests that the model has done well in correctly identifying some of the anomalies.

Precision (Positive Predictive Value): This is the ratio of true positives to the sum of true and false positives. Here, it would be 3573 / (3573 + 5028) = 0.415. This means when the model predicts an anomaly, it's correct only about 41.5% of the time.

Recall (Sensitivity, Hit Rate, or True Positive Rate): This is the ratio of true positives to the sum of true positives and false negatives. Here, it would be 3573 / (3573 + 20) = 0.994. This means the model identifies 99.4% of all anomalies.

F1 Score: This is the harmonic mean of precision and recall, and it tries to find the balance between them. Here, the F1 Score would be 2 * (0.415 * 0.994) / (0.415 + 0.994) = 0.586. This is a moderate F1 score, which suggests that the model is not particularly good at classifying correctly.

ROC Curve:

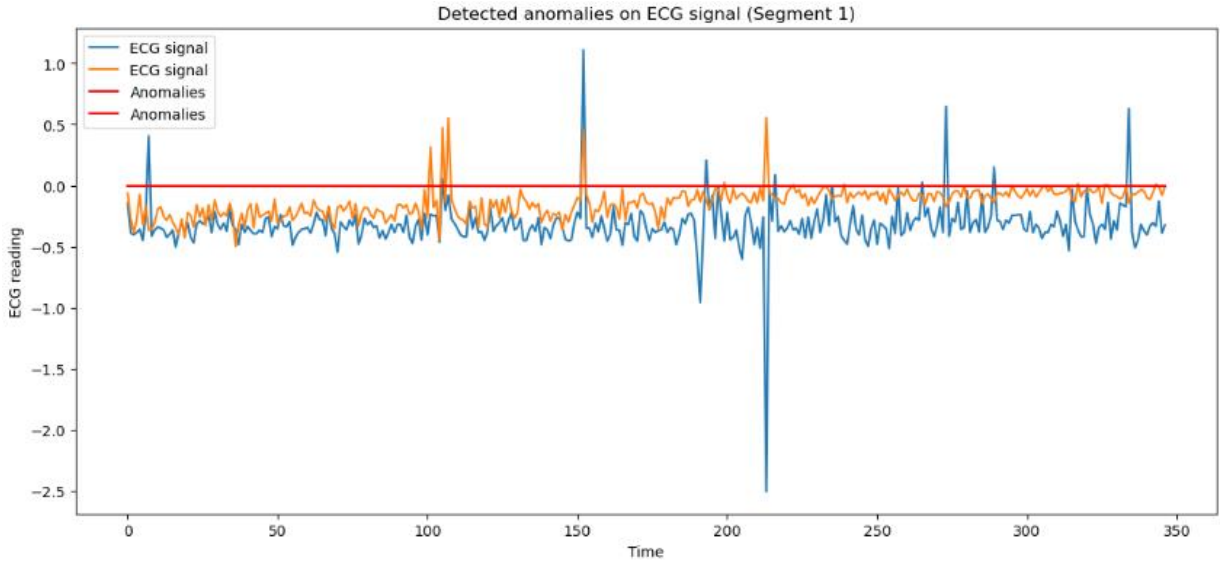


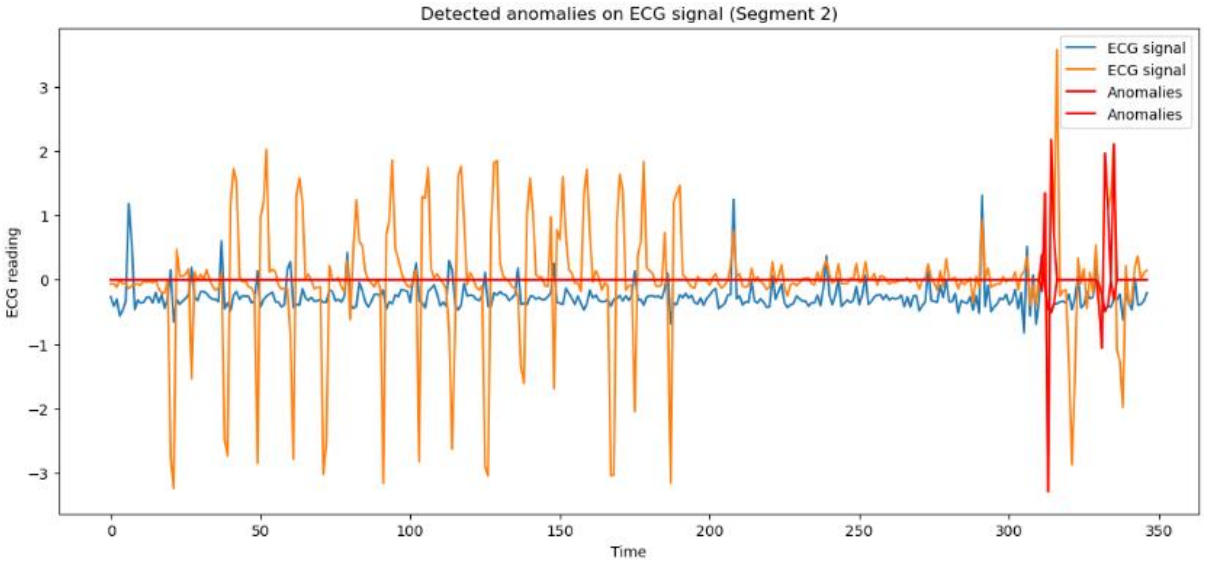
ROC-AUC: This is the area under the Receiver Operating Characteristic (ROC) curve. The AUC is the area under this curve, and it ranges from 0 to 1. An AUC of 0.5 suggests no discrimination (i.e., ability to classify correctly), 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding. Here, an AUC of 0.45 suggests that the model has poor discrimination ability. This means the model is not good at distinguishing between positive (anomaly) and negative (normal) cases.

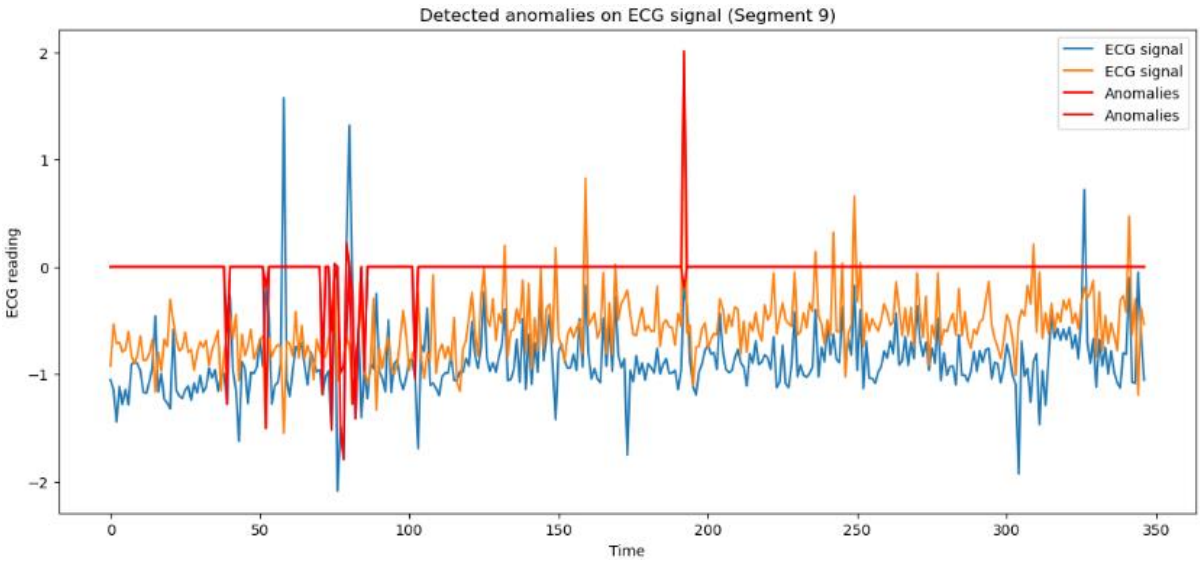
In conclusion, although the model identifies most anomalies (high recall), it incorrectly flags many normal instances as anomalies (low precision), leading to many false alarms. The model's poor discrimination ability is further confirmed by the low ROC-AUC score. This model may require further tuning to better balance its recall and precision. The high number of false positives suggests that the model may be too sensitive to variations in the data that are not actually indicative of anomalies.

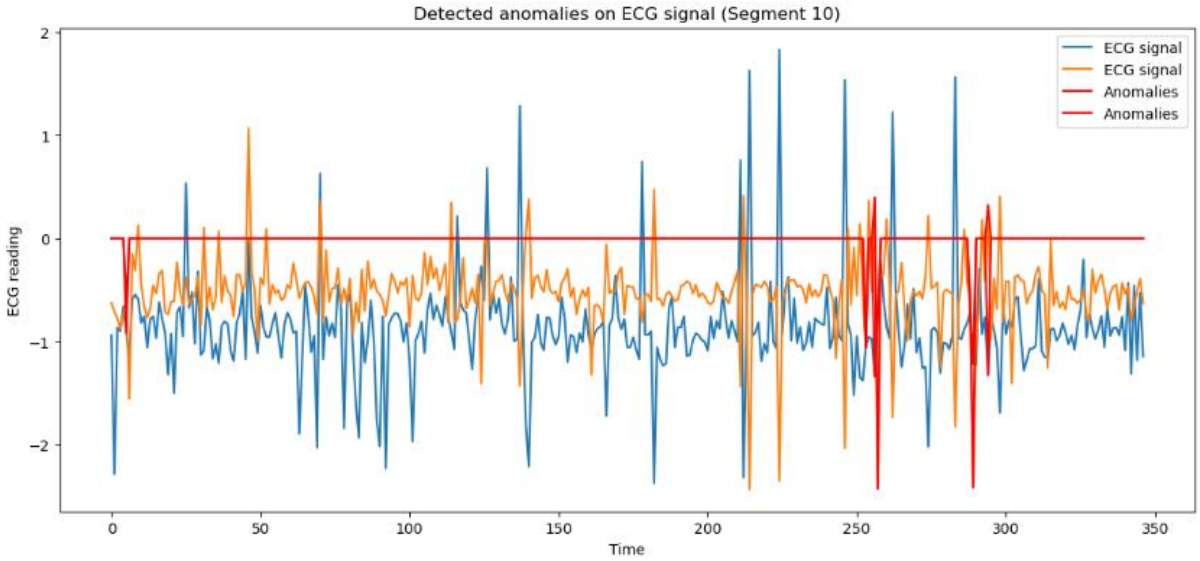
Additional graphs displaying the anomalies detected overtime.
(I did not have enough time to figure out how to get something like this to display for the other model)

I will note that I am not a cardiologist, shocker, so I don't have any reliable means to verify these results, but I feel as though these are interesting to look at and are effective at visualizing the model's performance.




Detected anomalies on ECG signal (Segment 2)
ECG signal
ECG signal
Anomalies
Anomalies
ECG reading
Time


Detected anomalies on ECG signal (Segment 9)
ECG signal
ECG signal
Anomalies
Anomalies
ECG reading
Time


Detected anomalies on ECG signal (Segment 10)
ECG signal
ECG signal
Anomalies
Anomalies
ECG reading
Time

Discussion

In this project, we used a Long Short-Term Memory (LSTM) network and an Isolation Forest, which both are particularly proficient for anomaly detection in time-series data. The LSTM model has been able to correctly identify a substantial number of anomalies (high recall). This is significant as failing to detect actual anomalies can lead to severe consequences in healthcare applications. The Isolation Forest model also showed an impressive capability in recognizing many anomalies.

These models offer an improvement over traditional time-series analysis methods that generally require a priori knowledge of the data's characteristics. Here, both models can learn patterns from the data autonomously, making them suitable for a wider range of applications.

Limitations and Potential Areas for Improvement:

While both models have strengths, there are also some significantly debilitating limitations. Both the LSTM and Isolation Forest models had issues with precision, resulting in many false positives. These could lead to unnecessary alerts and warnings, potentially causing undue stress or leading to alarm fatigue, where too many alerts lead to professionals ignoring the warnings. This can be a significant issue in healthcare settings where accurate predictions are crucial.

The LSTM model also had no true negatives. This may suggest that the model or the data might not be correctly distinguishing between normal and abnormal cases. This will require some additional attention in further research, as it is necessary to properly structure the data in a way that the model can easily understand and learn from. Unfortunately, I believe I may have made some errors in my programming when selecting data from the dataset to validate the anomalies on. This could also be due to the nature of the data in that it is too abundant with irregular arrythmias and not enough regular arrythmias leading to an imbalance in the dataset, skewing the effectiveness of the model.

Potential Future Work:

Based on our experience and these results, several potential future research avenues arise:

Training the model on normal heartbeats

Our results suggest the model may not be correctly distinguishing between normal and abnormal heartbeats. By first training the model on ECG data from normal heartbeats, the model might better learn what constitutes a regular arrythmia, and better identify anomalies. Utilizing a larger dataset: The size and diversity of the dataset used for training is arguably the most significantly influence model performance. A larger and more varied dataset could improve the model's ability to generalize and accurately predict unseen data. The model used in the paper (Rajpurkar et al.)[2], utilized a much larger and diverse dataset, and that accompanying a more robust CNN model than our fairly simplistic LSTM model, has produced much more optimistic

results in anomaly detection with the model outperforming the experts in some cases. This proves that this is not a fruitless adventure, and a more sophisticated approach is required for real world applications.

Optimizing model parameters

Further tuning of model parameters might lead to improved performance. Techniques such as grid search or randomized search can help find optimal parameters. I attempted to utilize first a Bayesian hyperparameter optimization, which is a sequential design strategy for global optimization of black-box functions that does not assume any prior knowledge about the objective function. However, this approach took an immense amount of computing resources, and I was not able to train the model in time. The Bayesian Optimization approach builds a probability model of the objective function (in this case, it could be model accuracy or F1-score, or any other metric we're trying to optimize) and uses it to select the most promising hyperparameters to evaluate in the true objective function. The key idea is that it uses past evaluation results to choose the next input values.

An approach I took to mitigate the frustratingly lengthy wait time was implementing CUDA and cuDNN in my python environment, which in conjunction with TensorFlow's library makes it easy to train models on your systems GPU rather than the slower CPU. This was effective for a time. This was a nice boost in performance; however my system still was not able to perform all the calculations fast enough.

Given that I had not tuned the hyperparameters of the models too many times, some variant of Bayesian Optimization, or Grid or Random Search would possibly yield better results. If I had the computing resources though, Baysian Optimization would be my first next step because it is usually more efficient than other optimization methods like Grid Search and Random Search, because it uses previous evaluation results to inform the next set of hyperparameters to try.

Trying different models

While LSTM and Isolation Forest models have shown some promise, other models, such as Convolutional Neural Networks (CNNs), may perform better. CNNs have shown strong performance in time-series data and could be a promising alternative.

Conclusion

In our study, we attempted to address a global critical health concern—cardiovascular diseases, focusing particularly on the detection of arrhythmias through the analysis of electrocardiogram (ECG) data. We utilized Long Short-Term Memory (LSTM) networks and Isolation Forest machine learning algorithms, for detecting anomalies in ECG data. The LSTM network, well-suited to time series data like ECG, served as our primary model, while the Isolation Forest served as a comparison.

This area of research is important because cardiovascular diseases cause a lot of deaths worldwide. Early and accurate detection of these diseases can lead to better treatment and management, potentially preventing severe outcomes. Automated anomaly detection can be a game-changer in this regard, especially considering the increasing volume of ECG data that needs to be interpreted. Additionally, with 24-hour scans becoming more available, it is safe to assume the recording technology to continue to improve, and 24/7 heart monitors are not too far from being reality, thus some type of automated analysis of our rhythms will become necessary at some point in the future.

LSTM networks are well-suited for analyzing ECG data since they can recognize patterns over time. These networks can learn to spot normal heart rhythms and, in turn, pick out the ones that are not normal. Our LSTM model showed moderate success, correctly identifying anomalies but also predicting a lot of false positives. In a real-world healthcare setting, this high number of false alarms is problematic and potentially decrease the trust in the model.

On the other hand, we also used the Isolation Forest algorithm, which is another method for anomaly detection. It does not need balanced data like LSTM and can be better at identifying different types of irregular heart rhythms. However, this model ended up having a high number of false positives as well and was also not very successful in confirming its anomaly predictions similar to the LSTM model, rendering both models effectively useless in a real-world application.

In conclusion, both LSTM networks and Isolation Forest can be used for ECG anomaly detection, but they both have their own strengths and weaknesses. While they can recognize anomalies, they also predict a significant number of false positives. In the future, more research and improvements are needed to make these methods more reliable and effective for real-world applications. This study really put into perspective how difficult the problem is and how simple and effective of a solution deep learning neural networks are in this type of research. AI is not useful for just the application of detecting heart anomalies. It’s capacity to learn will prove invaluable in many areas of research, and clearly has the potential to be the largest disruptive technology and human has seen in their lifetime. We are excited to have been born in a time with such impressive leaps in technology, and even more so that we get to experience cutting edge technology. We are about to witness the world completely change, and I am grateful to have the opportunity to contribute to these advances, or at least witness what other people can create with this technology.

Works Cited:

[1] “Cardiovascular Diseases (Cvds).” *World Health Organization*, 11 June 2021, www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[2] Rajpurkar, Pranav, et al. “Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks.” *arXiv.Org*, 6 July 2017, arxiv.org/abs/1707.01836.

[3] Hedèn, Bo, Ohlsson, Mattias, Holst, Holger, Mj¨oman, Mattias, Rittner, Ralf, Pahlm, Olle, Peterson, Carsten, and Edenbrandt, Lars. Detection of frequently overlooked electrocardiographic lead reversals using artificial neural networks. The American journal of cardiology, 78(5):600–604, 1996.

[4] Moody, George, and Roger Mark. “MIT-BiH Arrhythmia Database.” *MIT-BIH Arrhythmia Database v1.0.0*, 24 Feb. 2005, www.physionet.org/content/mitdb/1.0.0/.