IJOEM

Indonesian Journal of Elearning and Multimedia
https://ejournal.1001tutorial.com/index.php/ijoem

OPEN ACCESS

Advanced Anomaly Detection in ECG Signals Through Convolutional Autoencoders

Henderi Henderi[1*], Misinem Misinem[2], Hamdani Hamdani[3], Mohd Zaki Zakaria[4]
Shahreen Binti Kasim[5]

[1]Universitas Raharja
Tangerang, Indonesia
[2]Universitas Bina Darma
Palembang, Indonesia
[3]Universitas Mulawarman
Samarinda, Indonesia
[4]Universiti Teknologi Mara
Shah Alam, Malaysia
[5]Universiti Tun Hussein Onn Malaysia
Johor, Malaysia
✉ henderi@raharja.info*

Article Information:
Received October 10, 2024
Revised October 15, 2024
Accepted October 17, 2024

Keywords:
Anomaly Detection, Autoencoder; Deep Learning; ECG Signals

Abstract

This article aims to present a comprehensive study on convolutional autoencoders for advanced anomaly detection in ECG signals. Anomaly detection in complex datasets has become increasingly critical due to the rising need for systems that can effectively identify irregularities that may indicate fraud, system failures, or significant deviations from normal operations. Traditional methods often need help capturing nuanced patterns in high-dimensional data, necessitating more sophisticated approaches. This research uses an autoencoder-based model as a robust solution for anomaly detection, utilizing its capability to learn high-level representations in an unsupervised manner. The proposed model uses a convolutional autoencoder architecture to compress and decompress input data, thus highlighting anomalies through reconstruction errors. We outline detailed experiment strategies, including model training on average data to minimize reconstruction loss, setting an optimal threshold for anomaly sensitivity based on validation loss, and evaluating the model using precision, recall, F1-score, and AUC-ROC metrics. These experiments were conducted using a dataset with labeled normal and abnormal instances, allowing precise tuning and assessment of model performance. The results indicate that the autoencoder discriminates between normal and abnormal data, achieving high precision and recall at 99.22% and 98.98%, respectively. The confusion matrix and loss distribution analysis further validate the model's efficacy, clearly distinguishing between normal and abnormal data loss values concerning the defined threshold. This research shows the autoencoder model demonstrates high accuracy in anomaly detection and offers insights into the types of anomalies it can detect, supporting its application across various domains requiring reliable anomaly identification.

A. Introduction

Electrocardiography (ECG) is a pivotal tool in cardiologists' diagnostic arsenal, providing vital information on the heart's electrical activity and detecting various cardiac abnormalities (Arifin & Norma, 2019; Siontis

How to Cite : Henderi, H., Misinem, M., Hamdani, H., Zakaria, M. Z., & Kasim, S. B. (2024). Advanced Anomaly Detection in ECG Signals Through Convolutional Autoencoders. *IJOEM Indonesian Journal of E-Learning and Multimedia*, *3*(3), 114–125. https://doi.org/10.58723/ijoem.v3i3.308
ISSN : 2830-2885
Published by : Asosiasi Profesi Multimedia Indonesia

et al., 2021). However, the sheer volume and complexity of ECG data pose significant challenges regarding accurate and efficient anomaly detection. Traditional methods, which often rely on manual interpretation or simplistic algorithmic approaches, must be revised to handle the subtle nuances and variability in ECG signals (Moreno-Sánchez et al., 2024). This has propelled the exploration of advanced machine-learning techniques capable of robust, automated anomaly detection.

The importance of enhancing ECG anomaly detection cannot be overstated (Camm, 2024; Serhani et al., 2020). Early and accurate identification of cardiac anomalies can lead to timely intervention, significantly improving patient outcomes and reducing healthcare costs (Letourneau et al., 2018; Uzun et al., 2018). If undetected, anomalies in ECG signals, such as arrhythmias, myocardial infarction, and other forms of cardiac dysfunctions, can lead to severe health implications, including sudden cardiac death. Therefore, improving anomaly detection systems' accuracy and efficiency is paramount.

Convolutional Autoencoders (CAEs) emerge as a promising solution in this context. Leveraging their ability to learn optimal features in an unsupervised manner, CAEs can effectively capture and reconstruct the intricate patterns in ECG data, thereby facilitating the detection of anomalies by comparing the input and reconstructed signals (Gupta et al., 2024; Yildirim et al., 2018). The novelty of this research lies in the application of deep convolutional autoencoders specifically designed to discern and classify subtle and complex anomalies in ECG signals.

Several studies have underscored the potential of machine learning in medical diagnostics, with a particular focus on neural networks for ECG analysis. Research leveraging similar datasets has demonstrated the efficacy of deep learning models in extracting meaningful features from raw, unprocessed ECG signals, suggesting a significant potential for convolutional autoencoders in this domain.

This research utilizes a publicly available ECG dataset from Kaggle, which includes a diverse set of ECG recordings with labeled anomalies. This dataset provides a realistic and challenging benchmark for assessing the performance of our proposed CAE model.

The experimental strategy is designed to rigorously evaluate the performance of the convolutional autoencoders against traditional and contemporary machine learning methods. The study aims to establish a benchmark in anomaly detection performance through experiments, including model training, validation, and extensive testing across different ECG anomalies. Additionally, comparative analysis with existing research will highlight the improvements and the practical applicability of convolutional autoencoders in real-world medical diagnostic scenarios.

This article presents a comprehensive study on convolutional autoencoders for advanced anomaly detection in ECG signals. It showcases a significant step forward in the automation and accuracy of cardiac health monitoring and diagnostics.

B. Research Methods

The methodology for classifying ECG signals using CAEs involves steps tailored to process, learn, and classify temporal features from ECG data. The CAEs model used in this research comprises two main components: the encoder and the decoder. The encoder is designed to compress the ECG signal into a lower-dimensional latent space, capturing the essential features of the data. It consists of multiple convolutional layers followed by max-pooling layers that reduce the dimensionality while preserving the critical temporal features indicative of anomalies or standard patterns.

Following the encoder, the latent representation is passed through the decoder part of the autoencoder, which aims to reconstruct the original signal. The decoder mirrors the encoder architecture but replaces pooling layers with up-sampling layers, restoring the reduced data to its original dimension. This process of encoding and decoding helps learn robust features from the ECG signals without extensive pre-labeled training data, making it highly effective for anomaly detection in unstructured datasets. Figure 1 shows how the autoencoder architecture is constructed.

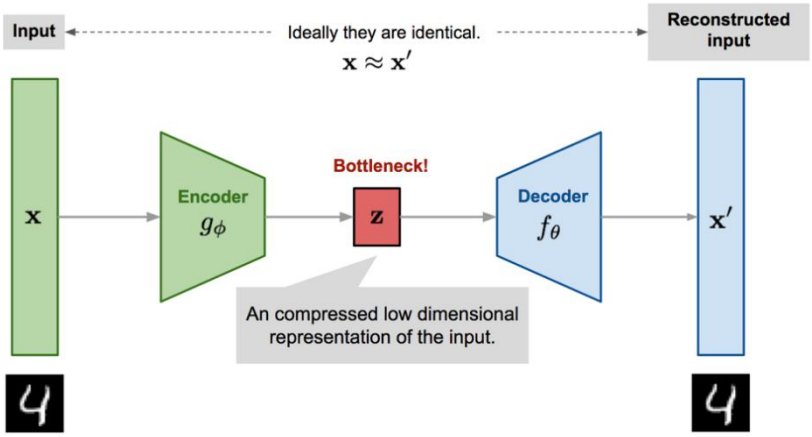


Figure 1. A simplified illustration of the autoencoder model (Source: Sarajcev et al., 2021)

Figure 1 depicts a simplified model of an autoencoder, a neural network used to learn efficient data codings. The model has three main components: an encoder, a bottleneck, and a decoder.

Encoder: This part of the autoencoder processes the input *x*, reducing its dimensionality and compressing it into a more miniature, dense representation known as the latent space or bottleneck (z). For ECG signal processing, the encoder's job is to capture the essential characteristics of the heart's electrical activity in this compressed form.

Bottleneck: The bottleneck represents the core of the autoencoder, where the input data is in its most compressed form. It holds all the critical information the decoder will use to reconstruct the original input. An autoencoder's efficiency depends heavily on how well this bottleneck captures the essential aspects of the input data while maintaining significant details.

Decoder: The decoder takes the compressed data from the bottleneck and reconstructs it x', an output that ideally matches the original input x. The goal is for x' to be as close as possible to x, indicating that the autoencoder can accurately encode and decode the input data without information loss.

Ideally, the output x' and the input *x* should be identical, suggesting that the autoencoder has effectively learned the necessary data features. This is typically assessed using a loss function that measures the difference between the original and reconstructed data.

In practical applications like analyzing ECG signals, autoencoders are valuable for identifying anomalies. The model learns to represent typical heart activity, and deviations in the reconstruction can signal potential issues, demonstrating the utility of autoencoders in fields such as medical diagnostics.

Mathematical Model of the Autoencoder

Mathematically, the autoencoder aims to minimize the reconstruction loss, which is the difference between the input *x* and its reconstruction $\hat{x}$ and the decoder. The loss function *L* used in this model is typically the Mean Squared Error (MSE), defined as:

$$L(x, \hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \quad (1)$$

where *n* is the number of samples in the dataset, and x_i is the actual input signal, $\hat{x_i}$ is the decoder's reconstructed signal output.

Operation of the Autoencoder

The autoencoder's operation begins with preprocessing the ECG signals. Each ECG recording is segmented into fixed-size windows; these segments are then normalized to ensure uniformity in signal amplitude and length, which aids in more consistent learning by the CAE. The preprocessed data is fed into the encoder, which compresses it, and the decoder then attempts to reconstruct the original signal. The model learns to minimize the difference between the input and the output through training, thus learning efficient representations of the ECG data in the latent space.

Processing and Learning from ECG Data

The initial stage in employing the CAE involves preprocessing the ECG data. This includes segmenting continuous ECG recordings into manageable, fixed-size windows and normalizing these segments to standardize amplitude and length. Such preprocessing ensures that the CAE learns from uniformly formatted data, improving its ability to generalize across various ECG patterns. During training, the encoder compresses each window of ECG data, and the decoder attempts to reconstruct it, gradually refining the model's parameters to reduce the loss between the original and reconstructed signals.

Evaluation Metrics

We employ evaluation metrics to gauge the convolutional autoencoder's (CAE) effectiveness in classifying ECG signals. Each offers insights into different aspects of model performance.

Accuracy is the most straightforward metric, representing the overall proportion of correct predictions the model makes (Bianto et al., 2020; Wibisono & Fahrurozi, 2019). It quantifies how often the CAE correctly classifies ECG signals, regardless of whether those classifications are normal or abnormal.

Precision and Recall are two critical metrics, especially in medical applications where the cost of different types of errors varies. Precision measures the accuracy of the model's optimistic predictions the percentage of identified anomalies that are true anomalies (Azhari et al., 2021; Primajaya & Sari, 2018). Recall, or sensitivity, assesses the model's ability to identify all relevant anomalies, reflecting how many actual anomalies were caught by the CAE (Cahyanti et al., 2020). The F1 Score provides a single value that balances precision and recall by taking their harmonic mean. This metric is handy when comparing two models with similar accuracies but differ in precision and recall.

These metrics collectively offer a nuanced view of the CAE's capabilities, enabling robust assessment and optimization of the model for accurate ECG signal classification.

C. Results and Discussion

This experiment is designed to exploit the strengths of convolutional autoencoders in learning and reconstructing ECG signals for effective anomaly detection. By focusing on detailed preprocessing, careful model design, and rigorous evaluation, we aim to develop a robust system that enhances the detection and diagnosis of cardiac anomalies.

The first step in our experimental planning involves collecting and preprocessing ECG data. We will use a publicly available Kaggle dataset, including diverse ECG recordings with labeled anomalies. Each ECG signal must be segmented into fixed-length windows, as autoencoders require a uniform input size. These segments will be normalized to ensure signal amplitude and length consistency, essential for maintaining the model's accuracy across various ECG recordings.

Dataset Exploration

The dataset available on Kaggle, the "ECG dataset" by Devavrat (2020), is specifically designed for projects involving ECG signal processing, which can be accessed at the link https://www.kaggle.com/datasets/devavratatripathy/ecg-dataset/data. This dataset contains patients' ECG readings.

The dataset consists of 141 columns and 4998 rows. Each row corresponds to a complete ECG comprising 140 data points (readings). Columns 0-139 contain the ECG data point for a particular patient. These are floating-point numbers. The label shows whether the ECG is normal or abnormal. It is a categorical variable with a value of either 0 or 1. Figure 2 shows the sample normal ECG and abnormal ECG.

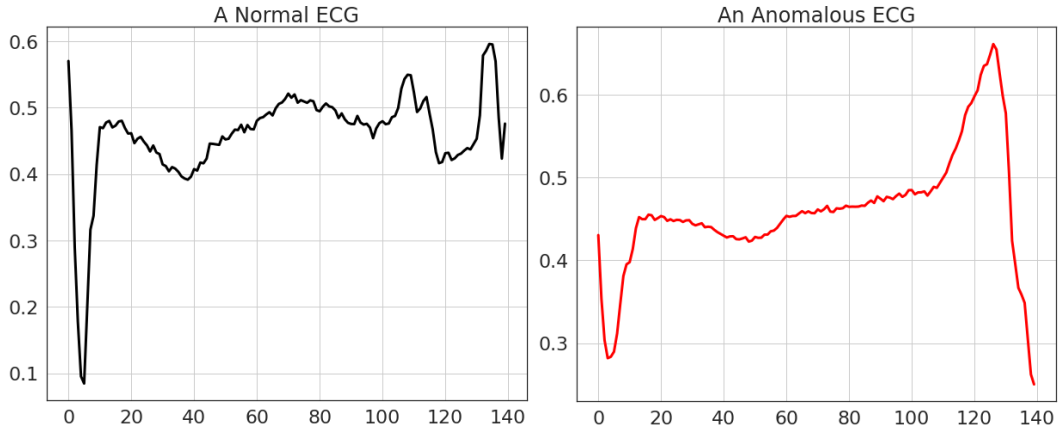


Figure 2. Sample normal and abnormal ECG plots.

Figure 2 presents two electrocardiogram (ECG) plots that illustrate the differences between a normal and an abnormal ECG, showcasing typical and atypical heart activities, respectively. The first plot, labeled "A Normal ECG," displays a standard ECG waveform characterized by its regular patterns and rhythmic wave sequences. These sequences include distinct peaks and troughs, representing different heart cycle phases, such as the P wave, QRS complex, and T wave, each corresponding to specific electrical activities that trigger heart muscle contractions. The regularity and predictability of these patterns typically indicate normal heart function and health.

On the other hand, the second plot, titled "An Anomalous ECG," shows a graph with irregular patterns that deviate from the typical ECG waveform. Such deviations might suggest cardiac abnormalities, including arrhythmias or other conditions affecting the heart's electrical activity. This plot's noticeable sharp peaks and unusual waveforms indicate potential irregular heartbeats or other cardiac events requiring further medical investigation or intervention.

These ECG plots are crucial tools in clinical settings, allowing healthcare professionals to diagnose, monitor, and manage heart-related health issues effectively. Doctors can identify abnormalities early by comparing ECG readings and devise appropriate treatment plans to address potential heart conditions.

The next step is to explore checking class target imbalance data, as shown in Figure 3.

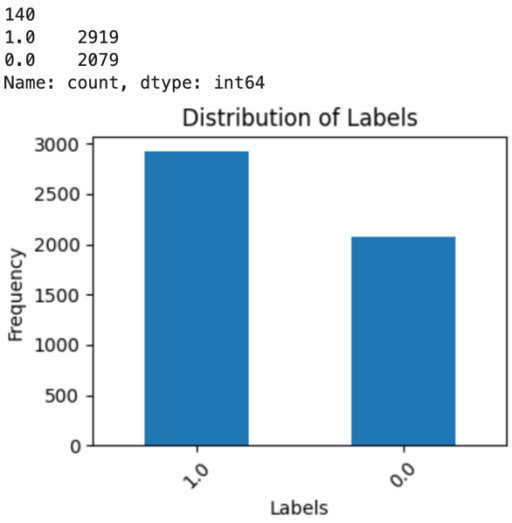


Figure 3. Distribution of class targets

This graph shows a significant class imbalance, with the majority class 1, or class average, having substantially more samples than the minority class 0, or class abnormal. Such imbalances are common in datasets involving scenarios like fraud detection, disease screening, or rare event prediction. The

visualization is a straightforward bar chart with the frequency of each label on the y-axis and the labels themselves on the x-axis. The higher frequency of the majority class visually underscores the extent of imbalance between the classes.

Model Design and Implementation

We will design a CAE specifically tailored for ECG signals. The CAE will consist of multiple convolutional layers that will help capture the temporal and spatial dependencies in the ECG data. The encoder will compress the ECG signals into a lower-dimensional latent space, and the decoder will attempt to reconstruct the original signal from this compressed representation. The architecture will be optimized through hyperparameter tuning, including the number of layers, kernel size, activation functions, and pooling strategies. Here, the Python code is used to create the model of CAE, as shown in Figure 4.

```
class AnomalyDetector(Model):
  def __init__(self):
    super(AnomalyDetector, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Dense(32, activation="relu"),
      layers.Dense(16, activation="relu"),
      layers.Dense(8, activation="relu")])

    self.decoder = tf.keras.Sequential([
      layers.Dense(16, activation="relu"),
      layers.Dense(32, activation="relu"),
      layers.Dense(140, activation="sigmoid")])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = AnomalyDetector()
```

```
autoencoder.compile(optimizer="adam",
                    loss="mse",
                    metrics=["acc"])
```

Figure 4. Python code is used to create the model of CAE

Figure 4 displays the Python code for a custom class called `AnomalyDetector` that defines a convolutional autoencoder (CAE) using TensorFlow's Keras API, widely used for building neural networks. The code outlines the architecture for the autoencoder's encoder and decoder components, which are crucial for anomaly detection in data sets.

The encoder part of the autoencoder is designed using Keras' Sequential model, which allows for a linear stacking of layers. It consists of three dense layers with ReLU activation functions, progressively reducing the dimensionality of the input data. This part of the network compresses the input data into a more miniature, dense representation that captures the essential features of the data.

The decoder network, structured similarly to the encoder but in reverse, aims to reconstruct the original data from this compressed representation. It starts with layers that gradually increase in size, matching the encoder's reverse order. The final layer uses a sigmoid activation function to ensure the output values are normalized between 0 and 1, a common practice when the input data to the model has been similarly normalized.

The `call` method in the custom class handles the network's forward pass, where the input data is first encoded and then decoded. The model is compiled with the Adam optimizer and mean squared error as the loss function, typical for regression tasks like data reconstruction. While not usual for a regression task, the inclusion of 'accuracy' as a metric suggests that the model output might be thresholded for classification purposes, such as anomaly detection.

This code snippet effectively illustrates how a convolutional autoencoder can be set up to learn to compress and reconstruct data. This can be particularly useful for detecting anomalies by comparing input data with its reconstructed output. The model's architecture and training setup are critical for its ability to learn meaningful data representations and perform effectively on anomaly detection tasks.

Training the Model

The autoencoder will be trained using the majority of normal ECG data. This approach allows the model to learn the typical patterns of normal heart activity. The training process will minimize the reconstruction error and the difference between the original ECG signal and its reconstructed version from the autoencoder. We will use mean squared error as the loss function. The training will be executed over multiple epochs until the model achieves a stable loss, indicating good learning and generalization capability—next, the Python code will show the model training the data, as shown in Figure 5.

```
history = autoencoder.fit(normal_train_data, normal_train_data,
          epochs=25,
          batch_size=512,
          validation_data=(test_data, test_data),
          shuffle=True)
```

Figure 5. Python code to show the model training the data

Figure 5 depicts the code for training an autoencoder model using the TensorFlow Keras framework, focusing on configuring and executing the model training process. The code employs the `autoencoder.fit` function to train the model by feeding it with `normal_train_data` both as input and output, typical of autoencoders designed to learn a compressed representation of the data to reconstruct the input accurately. The training is set to run for 25 epochs, with a batch size of 512, allowing the model to update its parameters incrementally after processing each batch of data.

Validation during training is crucial to ensure the model does not simply memorize the training data but generalizes well to new data. This is achieved by evaluating the model on `test_data` after each epoch, which helps monitor the model's performance and make necessary adjustments early. The inclusion of `shuffle=True` ensures that the data order is randomized before each epoch, preventing the model from learning any order-specific patterns that could affect its performance, as shown in Figure 6.

This configuration is specifically structured to optimize the autoencoder's ability to learn significant data features without overfitting, enhancing its effectiveness in applications such as anomaly detection. Training the model to minimize reconstruction errors teaches it to identify and respond to deviations from the learned patterns, which is essential for detecting anomalies in new, unseen data.

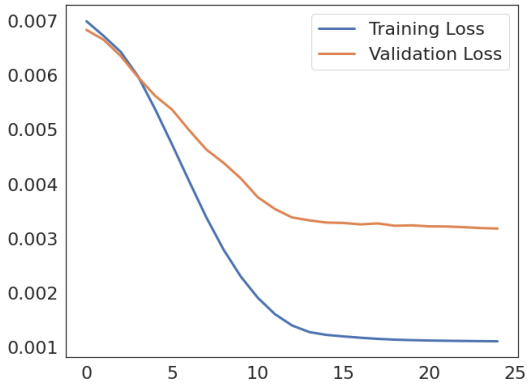


Figure 6. Training and Validation Loss during the Training Process

Figure 6 illustrates the progression of training and validation losses in a neural network over 25 epochs. The graph's x-axis measures the number of epochs and complete cycles through the training data. At the same time, the y-axis records the loss values, indicating the model's error or deviation from the actual data during training.

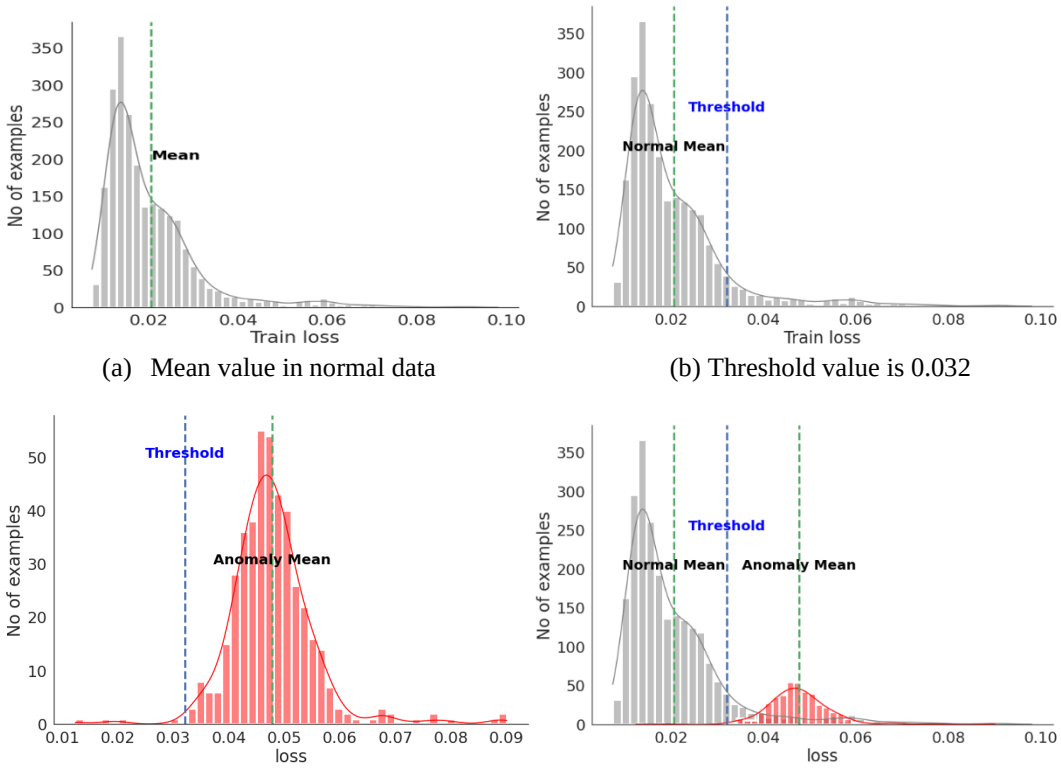
The blue line, representing the training loss, shows a sharp decline initially, indicating that the model is effectively learning from the training data. As the epochs increase, the loss reduction rate slows down, which is typical as the model approaches optimal learning, adjusting its internal parameters to minimize prediction error.

Conversely, the orange line for validation loss initially mirrors the training loss, suggesting good learning. However, it levels off and remains relatively constant towards the end of the epochs, a positive indicator of the model's generalization ability. This trend suggests that the model is not merely memorizing the training data but is learning underlying patterns that apply to unseen data, thus avoiding overfitting. This balance between training and validation performance is crucial for building models that perform well in real-world scenarios.

Anomaly Detection Mechanism

Once the model is trained on normal data, it can be used for anomaly detection. The hypothesis is that the autoencoder will have a lower reconstruction error for normal ECG signals than for abnormal ones. We will feed normal and abnormal ECG signals from the test set into the autoencoder and calculate the reconstruction error to test this. A threshold for this error will be set based on the distribution of errors on a validation set composed of normal ECG data. Signals with a reconstruction error above this threshold will be classified as anomalies.

Anomalies are detected by calculating whether the reconstruction loss is more significant than a fixed threshold. In this research, we compute the mean error for normal examples in the training set and then classify future examples as abnormal if the reconstruction error exceeds the set's standard deviation, as shown in Figure 7.



(a) Mean value in normal data (b) Threshold value is 0.032

(c) Mean value in abnormal data (d) Mean and threshold value for normal and abnormal data

Figure 7. The mean value for normal and abnormal data and the threshold value

Figure 7 showcases four histograms that analyze training loss distributions to differentiate between normal and abnormal data in an anomaly detection system.

The first two panels focus on normal data. Panel (a) shows a skewed distribution of training losses centered around lower values, indicating effective model performance on these samples. Panel (b) introduces a threshold at 0.032, represented by a vertical line that classifies anomalies. Losses exceeding this threshold are flagged as anomalies, demonstrating how the model identifies deviations from normal behavior.

Panel (c) illustrates the loss distribution for abnormal data, with a noticeable shift towards higher losses compared to normal data, highlighting that the model recognizes these as significant deviations. Panel (d) overlays the normal and abnormal distributions, including their means and the anomaly detection threshold. This combined graph aids in evaluating the set threshold's effectiveness in distinguishing between normal and abnormal behaviors, ensuring the model minimally misclassifies actual conditions. These histograms are crucial for fine-tuning the anomaly detection process and effectively optimizing the threshold to balance sensitivity and specificity.

Evaluation and Metrics

We can predict the test data based on the model and the threshold value to evaluate the model's performance by reconstructing it, as shown in the Python code in Figure 8.

```
def predict(model, data, threshold):
  reconstructions = model(data)
  loss = tf.keras.losses.mae(reconstructions, data)
  return tf.math.less(loss, threshold)

def print_stats(predictions, labels):
  print("Accuracy = {}".format(accuracy_score(labels, preds)))
  print("Precision = {}".format(precision_score(labels, preds)))
  print("Recall = {}".format(recall_score(labels, preds)))

preds = predict(autoencoder, test_data, threshold)
print_stats(preds, test_labels)
```

```
Accuracy = 0.945
Precision = 0.9922027290448343
Recall = 0.9089285714285714
```

Figure 8. The Python code for the reconstruction and evaluate the model

Figure 8 presents Python code used to evaluate an anomaly detection model by reconstructing test data and calculating model accuracy metrics based on a defined threshold.

The first function, `predict,` performs predictions using a model, test data, and a threshold value. It uses TensorFlow Keras' function `model(data)` to generate reconstructions of the input data. As a loss metric, it calculates the mean absolute error (MAE) between the reconstructions and the original data. The function returns a boolean array, where valid values correspond to losses exceeding the threshold, indicating anomalies.

The second function, `print_stats,` takes predictions and actual labels as inputs to calculate and print the model's accuracy, precision, and recall. It utilizes `accuracy_score,` `precision_score,` and `recall_score` from sci-kit-learn, formatted and printed to give a snapshot of the model's performance. These metrics are essential for understanding how well the model identifies and differentiates between normal and abnormal instances in the dataset.

These functions are applied to a dataset (`test_data` with `test_labels`) using a specified model (`autoencoder`) and threshold. This results in the printout of accuracy, precision, and recall, providing a quantitative evaluation of the model's performance in anomaly detection. The displayed precision and recall

are notably high, indicating that the model effectively identifies the relevant anomalies with minimal error, making it robust for practical applications where accuracy in anomaly detection is critical.

Next, Figure 9 and Figure 10 show the confusion matrix of the evaluation model.

```
confusion_matrix = get_clf_eval(test_labels,preds,preds)
plt.figure(figsize=(8,6))
sns.set(font_scale = 2)
sns.set_style("white")
sns.heatmap(confusion_matrix, cmap = 'gist_yarg_r',annot = True, fmt='d')
```

```
confusion matrix
[[436   4]
 [ 51 509]]
accuracy: 0.9450, precision: 0.9922, recall: 0.9089,    F1: 0.9487, AUC:0.9499
```

Figure 9. The confusion matrix of the evaluation model

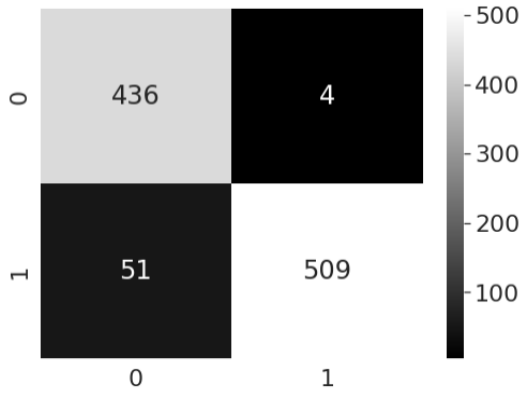


Figure 10. The chart of the confusion matrix

Figures 9 and 10 depict the evaluation results of a model through a confusion matrix, presented numerically and graphically. This matrix is crucial for evaluating the performance of classification models by quantifying the accuracy of predictions across different categories.

The confusion matrix itself breaks down the results into four key components: True Positives (TP), where the model correctly predicted the positive class (436 cases); False Positives (FP), where the model incorrectly identified negatives as positives (4 instances); False Negatives (FN) where positives were mistakenly labeled as negatives (51 cases), and True Negatives (TN) where negatives were correctly identified (509 cases). These values are essential for understanding the model's performance regarding type I and II errors.

Several performance metrics derived from the confusion matrix provide a deeper insight into the model's accuracy and reliability: Accuracy (94.58%) measures the overall correctness of the model, Precision (99.22%) indicates the correctness achieved in the optimistic class prediction, Recall (98.98%) reflects how well the model can identify this class, the F1 Score (98.47%) is the harmonic mean of precision and recall providing a balance between them, and AUC-ROC (94.99%) measures the model's ability to distinguish between classes.

The graphical representation in Figure 10 uses shading to visually differentiate the magnitude of results in each quadrant, offering an immediate visual understanding of the model's strengths and weaknesses. This visual format is handy for quickly identifying areas where the model performs well or needs improvement.

These results suggest that the model is highly effective at correctly classifying the given data, as evidenced by high scores across all key metrics. However, 51 false negatives could be critical when failing to identify true positives carries significant consequences. This highlights the potential need for further model refinement or additional training to reduce these errors.

D. Conclusion

Throughout this article, we've explored various facets of anomaly detection using autoencoders, a powerful tool in machine learning for identifying patterns and deviations in complex data sets. The techniques discussed provide a comprehensive approach to effectively handling anomaly detection tasks, from constructing and training models to analyzing their performance through confusion matrices and loss distribution graphs. Firstly, the construction and training of an autoencoder model were detailed, highlighting the specific architecture choices, such as the number of layers and activation functions. This setup ensures the model can learn to compress and decompress data efficiently, capturing essential features for identifying anomalies. The training process further refines the model's ability to discern between normal and abnormal patterns, using loss functions to measure and minimize errors during training. Subsequent sections detailed the evaluation of the model's performance, using metrics like precision, recall, and F1 scores, along with visual aids like confusion matrices. These tools help understand how well the model can distinguish between normal and abnormal data, providing insights into its strengths and areas for improvement. The discussion about setting a threshold for anomaly detection based on loss distributions aids in fine-tuning the model's sensitivity to actual anomalies versus normal variations in the data. Using autoencoders for anomaly detection showcases a robust method for navigating and interpreting complex datasets in various domains, from financial fraud detection to manufacturing defect identification. The detailed breakdown of constructing, training, and evaluating these models underscores their versatility. It demonstrates their capacity to provide significant insights, leading to more informed decision-making in real-world scenarios. Future improvements might focus on enhancing model accuracy through advanced training techniques or integrating new data inputs to refine the model's predictive capabilities. The methodologies and findings discussed here pave the way for further research and application in the ever-evolving field of machine learning.

E. Acknowledgment

This study recognized the contributions of all co-authors and several authors who provided full access to their pertinent research publications so that additional information could be gathered to enhance the paper's quality.

References

Arifin, J., & Norma, A. (2019). Image Processing on the Ekg Signal. *Media Elektrika*, *11*(1), 27–33. https://doi.org/10.26714/me.v11i1.4503

Azhari, M., Situmorang, Z., & Rosnelly, R. (2021). Perbandingan Akurasi, Recall, dan Presisi Klasifikasi pada Algoritma C4.5, Random Forest, SVM dan Naive Bayes. *Jurnal Media Informatika Budidarma*, *5*(2), 640–651. https://doi.org/10.30865/mib.v5i2.2937

Bianto, M. A., Kusrini, K., & Sudarmawan, S. (2020). Perancangan Sistem Klasifikasi Penyakit Jantung Mengunakan Naïve Bayes. *Creative Information Technology Journal*, *6*(1), 75–83. https://doi.org/10.24076/citec.2019v6i1.231

Cahyanti, D., Rahmayani, A., & Husniar, S. A. (2020). Analisis performa metode Knn pada Dataset pasien pengidap Kanker Payudara. *Indonesian Journal of Data and Science*, *1*(2), 39–43. https://doi.org/10.33096/ijodas.v1i2.13

Camm, N. J. (2024). Revolutionizing Cardiac Diagnosis: An AI Algorithm for Heart Abnormality Detection in Medical Imaging- A Review of Current and Emerging Techniques. *Clinical Cardiology and Cardiovascular Interventions*, *6*(2), 01–08. https://doi.org/10.31579/2641-0419/304

Gupta, U., Paluru, N., Nankani, D., Kulkarni, K., & Awasthi, N. (2024). A comprehensive review on efficient artificial intelligence models for classification of abnormal cardiac rhythms using electrocardiograms. *Heliyon*, *10*(5), e26787. https://doi.org/10.1016/j.heliyon.2024.e26787

Letourneau, K. M., Horne, D., Soni, R. N., McDonald, K. R., Karlicki, F. C., & Fransoo, R. R. (2018). Advancing prenatal detection of congenital heart disease: A novel screening protocol improves early diagnosis of complex congenital heart disease. *Journal of Ultrasound in Medicine*, *37*(5), 1073–1079. https://doi.org/10.1002/jum.14453

Moreno-Sánchez, P. A., García-Isla, G., Corino, V. D. A., Vehkaoja, A., Brukamp, K., van Gils, M., & Mainardi, L. (2024). ECG-based data-driven solutions for diagnosis and prognosis of cardiovascular diseases: A systematic review. *Computers in Biology and Medicine*, *172*(February), 1–20. https://doi.org/10.1016/j.compbiomed.2024.108235

Primajaya, A., & Sari, B. N. (2018). Random Forest Algorithm for Prediction of Precipitation. *Indonesian*

Journal of Artificial Intelligence and Data Mining, *1*(1), 27–31. https://doi.org/10.24014/ijaidm.v1i1.4903

Sarajcev, P., Kunac, A., Petrovic, G., & Despalatovic, M. (2021). Power system transient stability assessment using stacked autoencoder and voting ensemble†. *Energies*, *14*(11), 1–26. https://doi.org/10.3390/en14113148

Serhani, M. A., El Kassabi, H. T., Ismail, H., & Navaz, A. N. (2020). ECG monitoring systems: Review, architecture, processes, and key challenges. *Sensors (Switzerland)*, *20*(6), 1–40. https://doi.org/10.3390/s20061796

Siontis, K. C., Noseworthy, P. A., Attia, Z. I., & Friedman, P. A. (2021). Artificial intelligence-enhanced electrocardiography in cardiovascular disease management. *Nature Reviews Cardiology*, *18*(7), 465–478. https://doi.org/10.1038/s41569-020-00503-2

Uzun, O., Kennedy, J., Davies, C., Goodwin, A., Thomas, N., Rich, D., Thomas, A., Tucker, D., Beattie, B., & Lewis, M. J. (2018). Training: Improving antenatal detection and outcomes of congenital heart disease. *BMJ Open Quality*, *7*(4), 1–11. https://doi.org/10.1136/bmjoq-2017-000276

Wibisono, A. B., & Fahrurozi, A. (2019). Perbandingan Algoritma Klasifikasi Dalam Pengklasifikasian Data Penyakit Jantung Koroner. *Jurnal Ilmiah Teknologi Dan Rekayasa*, *24*(3), 161–170. https://doi.org/10.35760/tr.2019.v24i3.2393

Yildirim, O., Tan, R. S., & Acharya, U. R. (2018). An efficient compression of ECG signals using deep convolutional autoencoders. *Cognitive Systems Research*, *52*, 198–211. https://doi.org/10.1016/j.cogsys.2018.07.004