K-Fold Cross-Validation Enhanced Convolutional Neural Networks for Robust Detection of Abnormal ECG Signals

Abstract:

In pursuit of an accurate classification model, a convolutional neural network (CNN) was trained and tested on a substantial dataset of ECG heartbeats, with a particular focus on the model's ability to handle the prevalent class imbalance between normal and abnormal heartbeats. The analysis was thorough, employing k-fold cross-validation to mitigate overfitting and provide a robust estimate of model generalization. The findings are significant: the model achieved an outstanding training accuracy of 100%, with a very low training loss of 0.0012. Meanwhile, the validation loss was moderately higher at 0.0429, with validation accuracy at a commendable 98.86%. The average accuracy across all folds was consistent, approximating 98.49%, underscoring the model's reliable performance. Furthermore, class weighting strategies were integrated to enhance the model's sensitivity to the minority class, ensuring a balanced learning process. Collectively, the results validate this CNN model's robustness in generalizing well to unseen data, making it a promising tool for the accurate detection of abnormal ECG signals, which is paramount in clinical diagnostic procedures. The model's remarkable accuracy in identifying normal versus abnormal heartbeats positions it as an invaluable asset in advancing cardiac care and automated diagnosis.

Introduction:

This study explores the crucial statistical question of whether a Convolutional Neural Network (CNN), augmented with k-fold cross-validation, can accurately differentiate between normal and abnormal electrocardiogram (ECG) signals. It aims to deploy a deep learning CNN model to predict abnormal ECG signals from single heartbeats using both labeled training data and unlabeled test data. This approach tests the model's ability to generalize learned patterns to new, unseen data, mirroring real-world scenarios in healthcare diagnostics characterized by numerous uncertainties and unseen variables.

Data Analysis:

Data Set:

The dataset used in the CNN model originated from the PTB Diagnostic ECG Database

(Ralf-Dieter Bousseljot Published: Sept. 25, 2004. Version: 1.0.0).

Data Objects:

For the purpose of our study, the data is encapsulated within the 'ptb.Rdata' file which consists of several objects formatted for use in R(version 4.3.0, 2023-04-21 ucrt). This data is summarized in Table 1.

Table 1: PTB Database Data Frame Objects Implemented in R.

Data Frame	Description
`X_train`	Matrix of dimensions 10552 by 187; each row corresponds to an individual heartbeat signal
`X_test`	• Matrix with the dimensions of 4000 by 187, consisting of heartbeat signals to be used for model testing. • Unlike `X_train`, `X_test` does not come with associated labels, rendering it a blind test set intended to evaluate the model's predictive capabilities on unseen data.
`y_train`	• Vector of length 10552 that contains the labels for each heartbeat signal. • The labels are binary coded, with 0 representing a 'normal' heartbeat and 1 signifying an 'abnormal' heartbeat. • This labeling convention allows us to frame our task as a binary classification problem, where the predictive model will be trained to differentiate between normal and abnormal heartbeats based on the input signal.

The absence of diagnostic labels in the test set (`X_test`) is a deliberate design to simulate real-world scenarios where the true outcome is unknown and to objectively assess the performance of the predictive model developed from the training data (`X_train` and `y_train`). On this same rationale, there is no separate vector designated as 'y_test' for this study. This setup challenges the model to generalize its learned patterns to new, unlabeled data, providing a stringent test of its diagnostic accuracy.

Software Specifications and Libraries:

The CNN model and its graphical output summary were generated using R (version 4.3.0, dated 2023-04-21, UCRT edition). This process was facilitated within an R Integrated Development Environment (IDE) through the utilization of the `reticulate` package, which integrates Python (version 3.9.13) into R. This integration allowed for the seamless use of TensorFlow (version 2.13) and Keras (version 2.13.1) libraries, both of which are essential Python libraries for constructing and working with the CNN architecture.

Data Preprocessing:

The data matrices 'X_train' and X_test' underwent essential pre-processing steps such as and normalization scaling, and reshaping into arrays steps as part of the analysis pipeline, ensuring optimal conditions for the machine learning algorithms employed. The vector'y_train' remained as vector containing non-zero variance values.

Class Imbalance (Reduction in Majority Class Bias):

Prior to training, in the 'y_train' vector, it was noted that because there was a significant class imbalance between normal and abnormal heartbeats, zeros and ones, respectively. distribution of the classes in the training dataset, showing 2046 instances of the normal ECG class and 8506 instances of the abnormal ECG class. Thus, an additional step of applying weights to the minority class of zero to reduce bias in accounting for the majority abnormal class. This was implemented using the 'keras' package; the formulae used to calculate the weights for each class are summarized in Table 2.

Table 2: Class Imbalance in Outcome Training Vector

Class	Weight formula	Approximate Value
0 (Normal ECG)	(1/Count of class 0)×(Total number of samples/2.0)=(1/2046)×(10552/2.0)	2.5787.
1 (Abnormal ECG)	(1/Count of class 1)×(Total number of samples/2.0)=(1/8506)×(10552/2.0)	0.6203

CNN Architecture:

In brief, the CNN was built using the keras library in R. It begins with a 1D convolutional layer with 64 filters, a kernel size of 5, and ReLU activation, tailored for input data of shape 186 by 1. This is followed by a max pooling layer with a pool size of 2 to reduce dimensionality and a flattening layer to transform the data into a vector. It includes two dense layers; the first has 100 units with ReLU activation for learning non-linear combinations of features, and the second is the output layer with a single unit and sigmoid activation function for binary classification. The model uses the Adam optimizer and binary crossentropy as the loss function, with accuracy as the metric for performance evaluation.

Results:

Figure 1 graphically summarizes and presents the outcomes of training and validation losses, along with accuracy metrics, throughout the model's learning process.

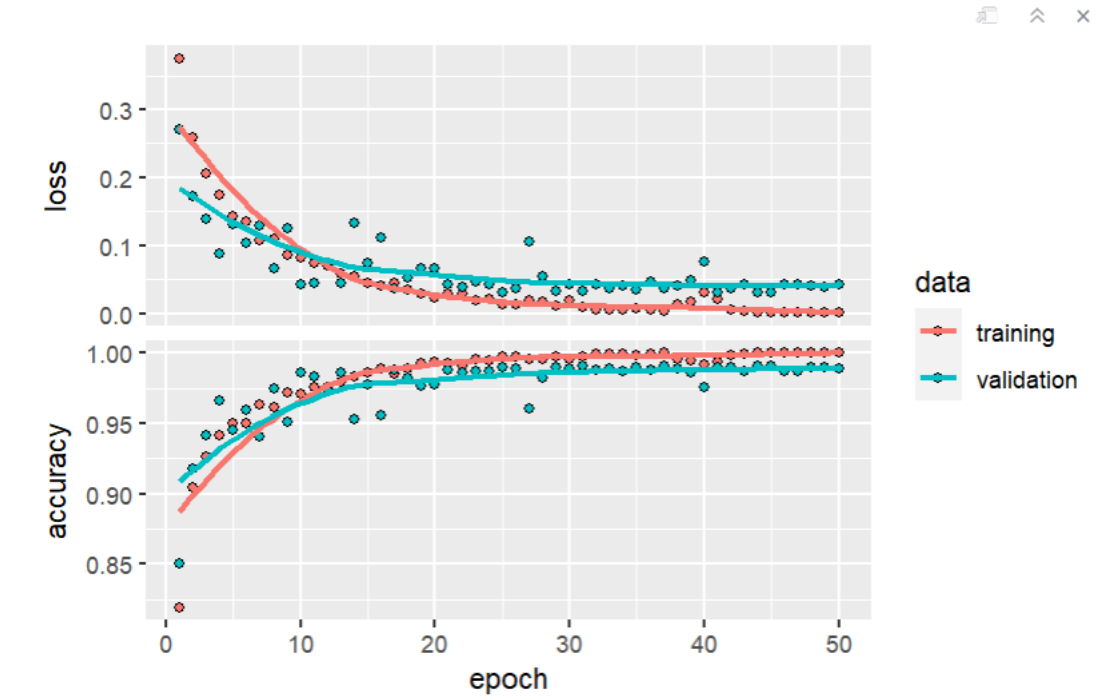


Figure 1: Training and Validation Loss and Accuracy of ECG Classification CNN over Epochs
The plot illustrates the CNN model's training and validation loss and accuracy for ECG signal classification over 50 epochs, showing rapid improvement initially and then stabilizing, indicating effective learning and potential slight overfitting. The training loss levels off near zero and the validation loss stabilizes at a slightly higher level, while both training and validation accuracy increase sharply at first and then plateau, with training accuracy nearing perfection. These observations suggest good model fit to training data and reasonable generalization, with the plot chosen for display due to its close accuracy measurements to those of a k-fold cross-validated model.

***Training Phase Metrics*:**

Loss: Represents the model's error or the difference between the predicted values and the actual values. A lower loss indicates better model performance. The loss decreased significantly from 0.3746 in the first epoch to 0.0012 in the 50th epoch, showing the model learned effectively from the training data.

Accuracy: Measures the percentage of correctly predicted instances out of all predictions. The training accuracy improved from 81.95% in the first epoch to 100% in the final epoch, indicating the model perfectly learned to classify the training data.

Validation Phase Metrics:

val_loss: This is the model's loss (error) on the validation dataset. Unlike the training loss, the validation loss is not used for training the model but to evaluate its performance on unseen data. It's crucial for detecting overfitting. The validation loss started at 0.2712 and had fluctuations throughout the training but decreased overall, reaching 0.0429 by the 50th epoch.

val_accuracy: Represents the accuracy of the model on the validation set. This metric started at 85.03% and increased to 98.86% by the end of training. High validation accuracy suggests that the model generalizes well to new, unseen data.

***Observations and Implications**:*

The model's training performance improved consistently, reaching perfect accuracy, which suggests it has effectively learned the training dataset. The validation accuracy also increased significantly, indicating good generalization. However, the fluctuations in validation loss across epochs, especially noticeable drops and rises (e.g., a drop at epoch 10 followed by an increase in subsequent epochs), might suggest the learning process encountered some variability in how well the model predictions matched the validation data. This is normal in the training process, especially with complex datasets.

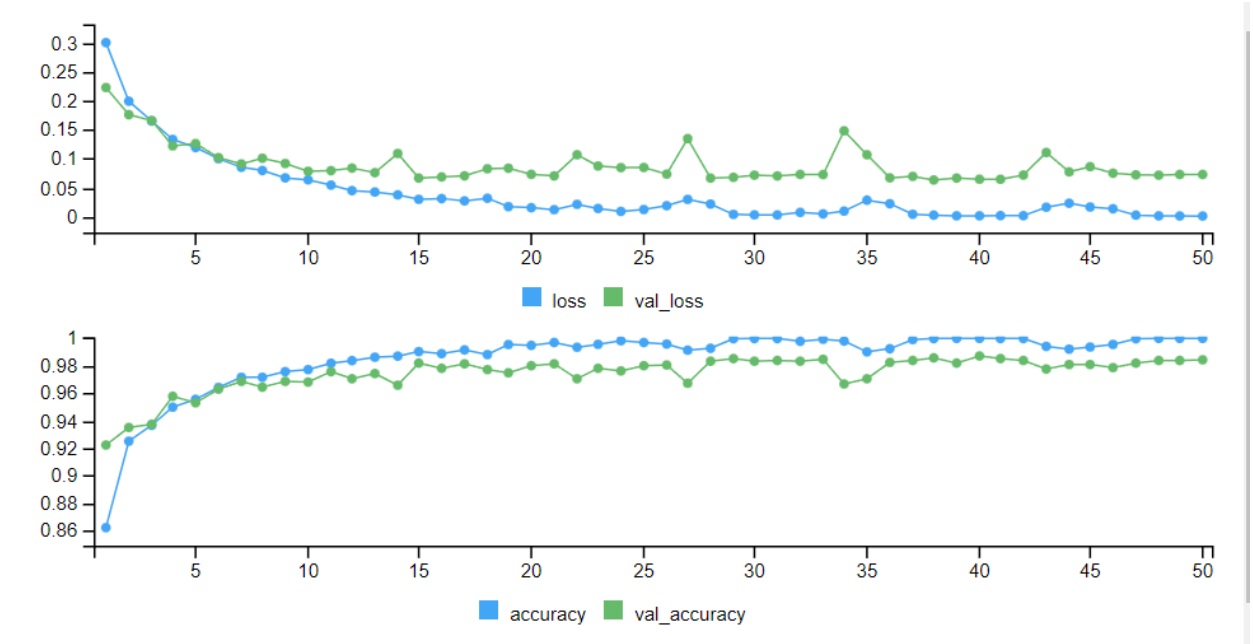
Reaching a high level of accuracy and a low level of loss in both training and validation suggests a well-fitted model. However, achieving 100% training accuracy might also raise concerns about overfitting, where the model learns the training data too well, including its noise and outliers, which can negatively impact its performance on new data. The validation metrics are crucial here; as long as they continue to improve or remain stable, the model is likely to generalize well.

In practice, monitoring both training and validation metrics is essential for diagnosing model behavior, such as underfitting (both accuracies are low) or overfitting (training accuracy is high, but validation accuracy is much lower). The metrics provided indicate a successful training process with effective learning and generalization up to the last epoch reported.

K-Fold Cross Validation:

High performance on a single validation set is not always sufficient to guarantee that the model will perform well across all possible unseen datasets. Further cross-validation can help to ensure that the model's performance is consistent across different subsets of the data. In k-fold cross-validation, the data is divided into k subsets (in this case, 5 subsets were used), and the model is trained k times, each time using a different subset as the validation set and the remaining data for training. This process can provide a more robust estimate of the model's performance and can help to detect if the model's performance is overly dependent on the choice of the validation set. Figure 2 encapsulates the performance trends of the k-fold cross-validated CNN, showcasing the convergence of loss and the stabilization of accuracy over 50 epochs in classifying ECG signals.

Figure 2: Training Dynamics of k-Fold Cross-Validated CNN for ECG Signal Classification



These plots summarize the training process of your k-fold cross-validated CNN model over 50 epochs. The upper plot shows the loss for both training (loss) and validation (val_loss), decreasing sharply initially and then plateauing, which indicates the model is learning and not improving significantly after a certain point. The lower plot illustrates the accuracy of the model, showing that both the training (accuracy) and validation accuracy (val_accuracy) increase rapidly at the start and then level off, with the training accuracy reaching higher levels compared to the validation accuracy. There's a small, consistent gap between training and validation accuracy, suggesting a slight overfitting but good generalization overall.

Average Accuracy Across All Folds:

The k-fold cross-validated CNN model reported the average accuracy obtained across all folds of the cross-validation process as approximately 98.49%. The Average Accuracy across all folds is calculated by averaging the accuracy scores obtained from each of the k validation sets. An accuracy score is a measure of the model's performance, representing the proportion of correct predictions made by the model over all predictions. This score of 0.98493173122406 (or 98.49%) is exceptionally high and suggests that the model performs very well across different subsets of the data, indicating strong generalization capabilities. This means that the model not only learned the underlying patterns in the training data but was also able to apply this knowledge effectively to unseen data, making correct predictions with a very high success rate. In practical terms, this means the model you've developed is likely to perform very well on similar data outside of the data it was trained and validated on, assuming the new data is drawn from the same distribution as the training and validation sets. The k-fold cross-validation's average accuracy, despite being computationally expensive, provides a more comprehensive and robust measure of the model's ability to generalize, as it encompasses multiple training and validation cycles across different segments of the data. The slight drop in accuracy compared to the single validation set method is

not necessarily indicative of poorer performance but rather a more conservative and potentially reliable estimate of how the model will perform in real-world applications.

Bias and Variance Trade off (Caveats):

The bias-variance trade-off is a fundamental concept that explains the balance between two main sources of error in machine learning models: bias, which results from erroneous assumptions in the learning algorithm, and variance, which occurs due to the model's sensitivity to small fluctuations in the training set. In the context of this Convolutional Neural Network (CNN) model, especially when dealing with a dataset exhibiting class imbalance and aiming for generalization to unseen data, the trade-off becomes critically important.

Initially, the CNN model might have a high bias if it's too simplistic. This was evident from the initial loss and accuracy metrics; the model did not perform perfectly at the start, indicating it was unable to capture all the underlying patterns in the data. However, as training progressed and the model learned from the data (evidenced by the decrease in loss and increase in accuracy), the bias decreased. In datasets with class imbalance, such as this one with different numbers of normal and abnormal ECG readings, a model with high bias might oversimplify the problem and underperform by not adequately learning the minority class features.

Achieving 100% accuracy on the training data suggests the model has learned the training data exceptionally well, perhaps too well, leading to overfitting. Overfitting is a sign of high variance, where the model is overly complex and learns the noise in the training data as if it were a true signal, which can degrade its performance on unseen data. The validation metrics serve as a check against overfitting by providing insight into how the model performs on data it hasn't seen during training. Although the validation accuracy and loss indicate good generalization (98.86% accuracy and a stabilized loss), careful monitoring is required to ensure that the model does not become too tailored to the training data, neglecting its ability to generalize.

The ideal balance in the bias-variance trade-off is achieved when both bias and variance are minimized, allowing the model to generalize well without underfitting or overfitting. For datasets with class imbalance, applying appropriate techniques like class weights or oversampling the minority class can help reduce bias towards the majority class. Additionally, validation methods like k-fold cross-validation, are crucial in controlling variance and ensuring the model generalizes well to unseen data. In this case, the application of k-fold cross-validation further aids in assessing the model's performance more robustly across different subsets of data, ensuring the high performance is consistent and not a result of data split. In summary, the model's training phase metrics and validation phase metrics, along with the use of k-fold cross-validation, illustrate an effective approach to managing the bias-variance trade-off. This ensures that the model not only learns effectively from the training data, with careful attention to class imbalance, but also maintains the ability to generalize to new, unseen data, mitigating the risks of overfitting while addressing underrepresentation in the data.

Conclusion:

In conclusion, the CNN developed and analyzed in this study demonstrates exceptional capability in classifying ECG signals, showing high accuracy and robust generalization as evidenced by both the training metrics and k-fold cross-validation results. By effectively addressing the prevalent class imbalance and employing k-fold cross-validation, the model has not only achieved a commendable classification performance but also indicated strong potential for reliable application in clinical diagnostic procedures. The success in handling class imbalance with weighted strategies, achieving near-perfect training accuracy, and maintaining high validation accuracy underscores the model's utility in advancing cardiac care through automated diagnosis, even when faced with the inherent uncertainties of real-world unseen data.