BedotProject Progress Report

School of Computing
The University of Georgia

Contents

1 Introduction

Cardiovascular diseases, particularly atrial fibrillation (AF), present a significant global health challenge, contributing to high morbidity and mortality rates worldwide. AF occurs when the heart's atria produce irregular heart rhythms and can lead to severe complications such as stroke, heart failure, and other cardiovascular events if not diagnosed and treated promptly. Early detection and accurate monitoring of AF are crucial in ensuring effective treatment and prevention of associated complications. While traditional methods for AF detection rely on electrocardiogram (ECG or EKG) recordings, which capture the heart's electrical activity, alternative approaches leveraging non-invasive and continuous monitoring techniques have gained attention. One such approach involves using Ballistocardiogram (BCG) signals, which represent the mechanical forces generated by the heart's contractions.

BCG signals offer several advantages that make them a compelling candidate for AF detection. Firstly, BCG signals enable non-invasive monitoring, making them suitable for continuous assessment in various settings, including home environments. Secondly, they provide unique insights into the mechanical aspects of cardiac function, reflecting the movements and forces associated with blood ejection during each heartbeat. This mechanical information can complement and enhance the understanding derived from electrical signals captured by ECG. Furthermore, when combined with other physiological signals or imaging modalities, such as accelerometry or imaging sensors, BCG signals can offer additional complementary information about cardiac dynamics and structure. Additionally, with the rapid advancements in wearable technology, the integration of BCG sensors into wearable devices has become feasible, allowing for real-time and remote monitoring of cardiac activity in a convenient and accessible manner.Also, BCG signals can be obtained using cushions,

mattresses, or fabrics, making them ideal for long-term monitoring.

This project aims to detect AF using BCG signals and deep learning models for automated analysis and classification. The project uses signal processing techniques, feature extraction methodologies, and deep neural networks to develop a robust and accurate AF detection system. Readers will explore various aspects, including signal processing techniques tailored for BCG signals, feature extraction methods to identify informative patterns indicative of AF presence, development and evaluation of deep learning architectures such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs) for AF detection using BCG signals as input data, and comprehensive performance evaluation metrics to assess the system's efficacy in detecting AF episodes. By integrating cardiovascular health, signal processing techniques, and deep learning methodologies, this project contributes to the advancement of innovative tools for AF detection and management, ultimately enhancing patient outcomes and healthcare practices.

2 Background Information

As part of our ongoing efforts to detect Atrial Fibrillation (AF) from Ballistocardiogram (BCG) signals, a series of experiments were meticulously designed and executed. This section aims to delve deeper into the methodologies employed and the datasets utilized in these experiments. Subsequently, the methodology section will offer a more comprehensive view of the experiments conducted.

2.1 Dataset

2.1.1 MIT-BIH Arrhythmia Dataset

The MIT-BIH Arrhythmia Database [1] serves as a cornerstone for classifying Electrocardiograms (ECGs). This extensive database comprises over 4000 extended Holter recordings gathered from the Beth Israel Hospital Arrhythmia Laboratory between 1975 and 1979. Notably, approximately 60% of these recordings were procured from hospitalized patients. Among the records, 23 were randomly selected (numbered from 100 to 124, with some numbers omitted), and an additional 25 records (numbered from 200 to 234, with some numbers missing) were specifically chosen to encompass a spectrum of uncommon yet clinically significant phenomena. These phenomena, which would not be adequately represented by a small random sampling of Holter recordings, contribute immensely to the database's utility. Each of these 48 records spans slightly over 30 minutes, with participants comprising 25 men aged 32 to 89 years and 22 women aged 23 to 89 years.

In the initial phases of this project, we employed this dataset to establish a robust model architecture and conduct preliminary experiments. Details regarding its utilization and significance in our methodology section (section 3.2)will provide a clearer understanding.

2.1.2 Synthetic Stable SCG Dataset

The connection between BCG and Seismocardiogram (SCG) is based on their mutual focus on the mechanical aspects of heart function. Both signals are useful in providing insights into cardiovascular dynamics, such as cardiac output, heart rate variability, and the coordination of heart chambers during the cardiac cycle. A steady (stable) SCG signal generally indicates normal cardiac activity without any significant irregularities or abnormalities in the detected

vibrations.

To address specific research questions regarding the peak predictive capabilities of deep learning models, we curated the Synthetic Stable SCG Dataset. This dataset was meticulously generated using the DataDemo repository's `simscg.py` file, which facilitated the creation of both stable and unstable data. We varied the noise levels (amplitude of Laplace noise) across the range [0.0, 0.2, 0.4, 0.6, 0.8, 1.0], keeping all other parameters at default settings during generation. Other hyperparameters were `smin = 90` and `smax = 180`

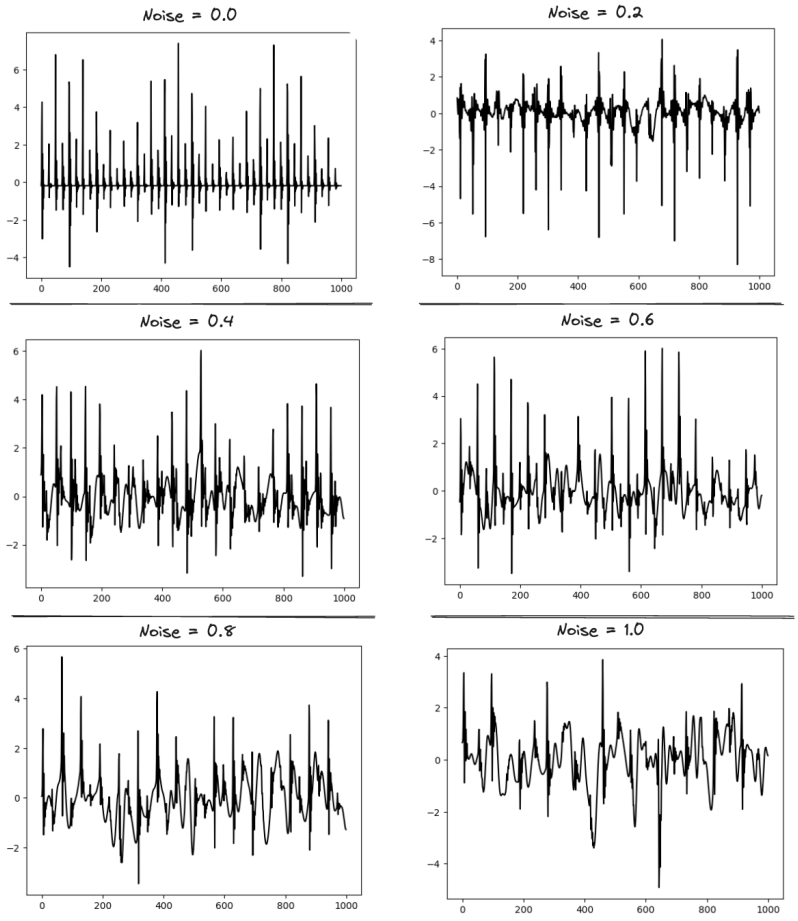

Figure 1: Visualization of Synthetic Stable SCG Signal for different noise levels.

2.1.3 Synthetic Unstable SCG Dataset

An unstable SCG (seismocardiography) signal can often indicate irregularities or abnormalities in cardiac activity. The Synthetic Unstable SCG Dataset is similar to its stable counterpart, and it was a crucial part of our investigation into the performance of deep learning models when trained on both stable and unstable synthetic SCG data. We used this dataset to evaluate the models' performance to detect peaks on unstable datasets by introducing variability through the use of noise levels ranging from 0.0 to 1.0 (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). To further increase variability during the dataset generation process, we set the systolic parameter to `systolic = random.randint(81, 180)`.

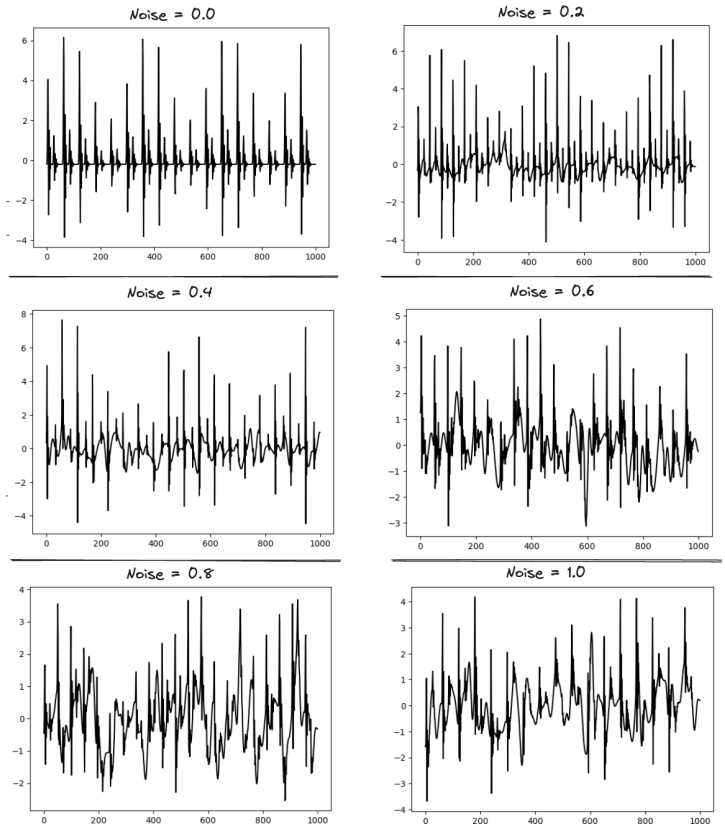

Figure 2: Visualization of Synthetic Unstable SCG Signal for different noise levels.

2.1.4 Real BCG Dataset

The Real BCG dataset is the dataset that we will use to train our model and measure its performance for AF detection. Currently, we have only acquired the Real Stable BCG dataset, and the real Unstable BCG dataset is still being acquired. The Real BCG Dataset is a valuable resource that complements our synthetic datasets. It contains real-world insights that were directly collected from patients, making it an authentic dataset and a vital benchmark for evaluating the performance and applicability of our models in clinical settings to detect AF. The Real BCG dataset is divided into train and test splits. We used this dataset to determine if deep learning models trained on synthetic data can accurately identify peaks for the Real BCG Dataset the reasoning for this will be discussed in the later sections. As a final goal for this project, we will use it to predict AF from BCG signals.

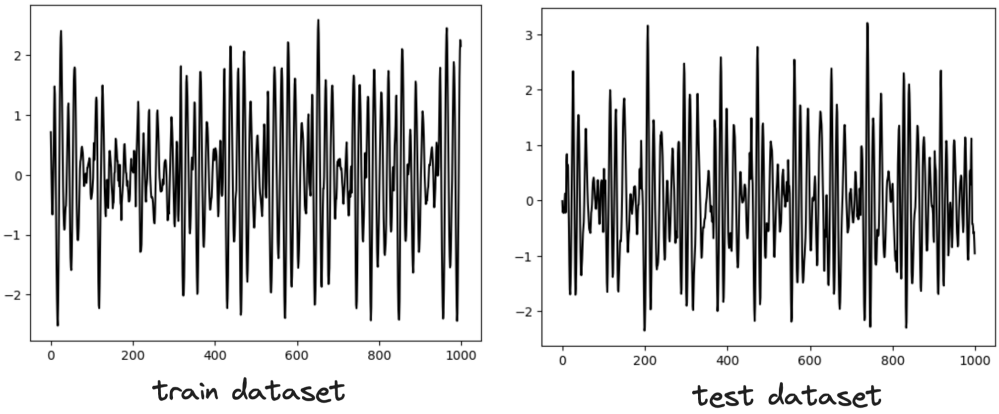

Figure 3: Visualization of Real BCG Signal.

2.2 Model Architecture and Pipeline

Figure 4 illustrates our current training pipeline, which was used for detecting Atrial Fibrillation (AF) from ECG or BCG signals. Initially, we take the ECG or BCG signal and detect

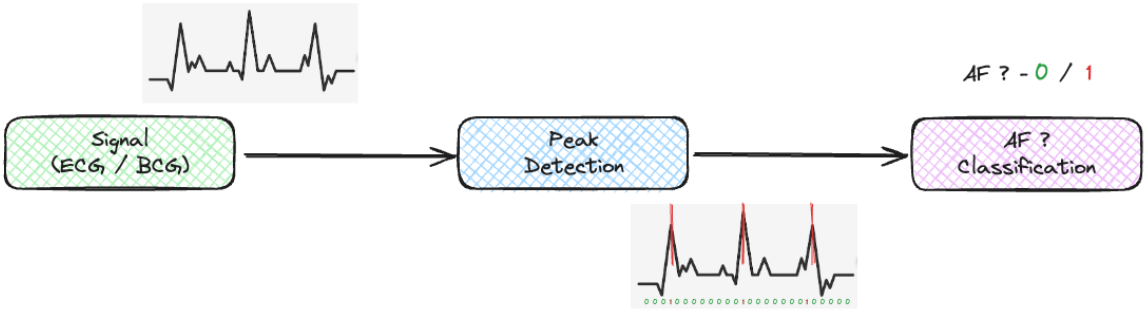

Figure 4: Pipeline for AF classification using Deep Leaning.

its peaks using a deep learning model. From these detected peaks, we predict whether or not the patient has AF using another deep-learning model. The detailed architecture for training can be found in the code-base, specifically in the `src/models/deeplean.py` file, where we have defined all the necessary components.

The primary class utilized for AF detection is `ECGClassifier`. This class first identifies peaks using the `PeakDetector`, and subsequently, it classifies them to detect AF using the `PeakClassifier`. The `PeakDetector` comprises a bidirectional Gated Recurrent Unit (GRU) and a Linear layer at the top of that to predict whether a point is a peak or not. On the other hand, the `PeakClassifier` consists of a 1-D Convolution layer and SENet (a squeeze-and-excitation neural network architecture that enhances feature re-calibration) to do the final classification of signal for AF detection.

For the loss function, we employ the Focal Loss, detailed in this link: Focal Loss. Through empirical testing, we have discovered that `FocalLoss` and Instance Normalization are crucial for effectively training deep models for peak detection.

In our codebase, specifically within the `src/models/` directory, you can locate the file

`nonparam.py`. This file houses the Rule-based non-parametric model for AF detection. Despite sharing the same components (`ECGClassifier`, `PeakDetector`, and `PeakClassifier`), this model operates solely on rules without utilizing deep learning techniques to predict AF from the input signal. We employed this model to assess the performance of AF detection without deep learning.

As discussed earlier, there are two types of data distributions: stable and unstable. Stable data indicates normal cardiac activity without significant irregularities or abnormalities, while unstable data often indicates irregularities or abnormalities in cardiac activity. Due to limited access to unstable data examples, we needed to generate synthetic data to explore whether we could train a model on well-labeled stable data and then transfer that learning to unstable data.

Our approach involved creating synthetic stable and unstable data to investigate the transfer-ability of learning from stable signals to unstable ones, especially focusing on peak detection. Given the mutual emphasis of Ballistocardiography (BCG) and Seismocardiography (SCG) on the mechanical aspects of heart function, we aimed to assess whether learning from stable SCG signals could be transferred to unstable SCG signals and also to evaluate the transfer learning between stable and unstable data distributions.

Figure 5 illustrates the tasks we undertook to study these concepts. The first dark arrow from stable SCG to unstable SCG represents our initial exploration of transferring learning from stable data to unstable data. We also trained on synthetic stable SCG data and evaluated peak detection performance on real BCG data (dark arrow from synthetic stable SCG to real stable BCG) to assess the transfer-ability between different data distributions. Our final goal, depicted by the dotted arrow, is pending completion due to ongoing acquisition

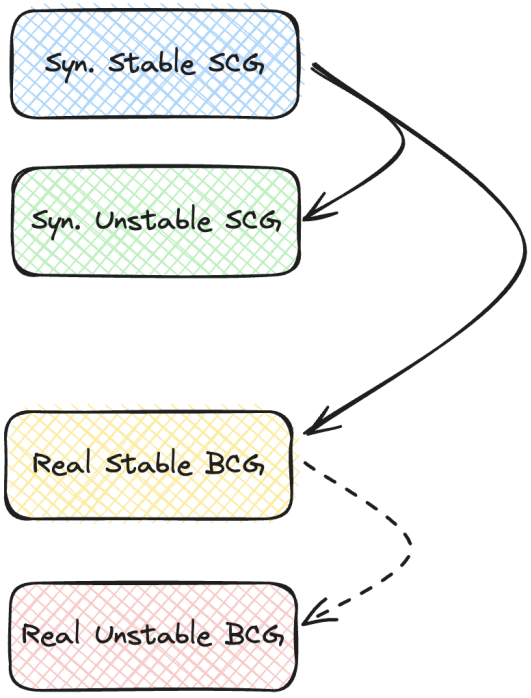

Figure 5: Illustration of tasks undertaken to study transfer learning between stable and unstable data.

of the Unstable BCG dataset.

Our ultimate goal is to detect Atrial Fibrillation (AF) from the signal data via detecting the peak positions. Figure 6 visually represents the progress made on the tasks, both those that have been completed and those that are pending. We have successfully completed the detection of AF from the Electrocardiogram (ECG) signal via peak positions. However, the detection of AF from the Ballistocardiogram (BCG) signal via peak positions is still pending, and achieving this milestone remains the final objective of this project.

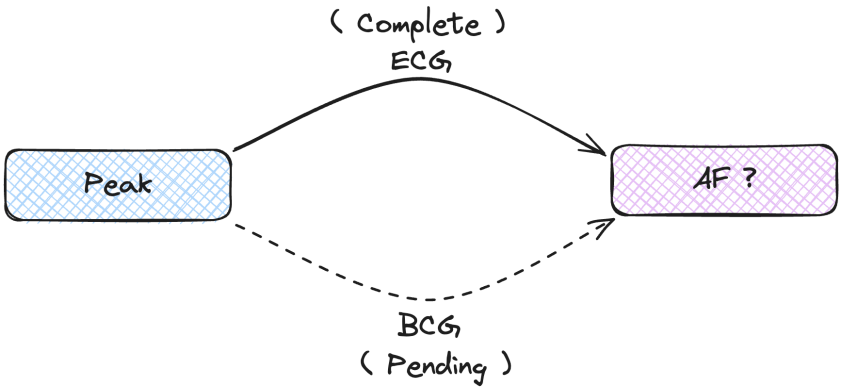

Figure 6: Progress overview of completed and pending tasks in the project.

3 Methodology and Results

3.1 AF Detection using ECG Signal on MIT-BIH Arrhythmia Dataset

We utilized the model architecture described in Section 2.2 (RNN (GRU) + CNN + SENet) to classify ECG signals from the MIT-BIH arrhythmia dataset. Specifically, we focused on the first split of 23 records (numbered from 100 to 124, with some numbers omitted), employing a split ratio of train, validation, and test datasets of `0.85:0.05:0.1`. Optimization techniques such as SGD, early_stopping, weight_decay, and lr_decay were applied to enhance the performance of the model for atrial fibrillation (AF) detection

As shown in Table 1, our model outperforms the previous CNN-based implementation [2], but it still falls short of the state-of-the-art transformer-based method [3].

We trained our model on the second split of 25 patient records (numbered from 200 to 234, with some numbers missing), with a split ratio for train, validation, and test of `16:2:3`. Our

Table 1: Performance Comparison for ECG Signal Classification

Sample Level	Model	Accuracy (%)	F1-Score
Segment	CNN-based [2]	98.17	98.17
	Transformer-based [3]	**99.23**	**99.23**
	CNN-based (ours)	98.25	98.33
Window	CNN-based (ours)	55.41	2.49

training process involved experimenting with various combinations of architecture settings including CNN, RNN, and non-parametric models as well as different inputs to predict the arrhythmia. Additionally, we integrated runtime peak detection and arrhythmia detection from these peaks into our experiments.

Table 2: Performance Comparison of ECG Models

Input	Output	Model	Accuracy (%)	F1-score
ECG Signal	Arrhythmia	CRNN-Based (new split)	64.93	72.36
		CRNN-based (hyper-tuned)	**94.50**	**96.05**
		CNN + w/o DL	79.94	85.15
ECG Peaks	Arrhythmia	RNN-based	83.55	89.52
		CNN-based	**89.78**	**93.16**
		w/o DL	71.68	83.50
ECG Signal	Arrhythmia	RNN-based (matched)	94.37	26.53
		RNN-based (40ms)	99.24	83.95
		RNN-based (60ms)	99.51	88.38
		w/o DL (40ms)	99.58	69.61
		w/o DL (60ms)	**99.58**	**91.06**

As seen in Table 2, our tuned model combining RNN and CNN architecture yielded the best performance for arrhythmia detection from ECG signals compared to other models. Additionally, the table shows results for peak detection with distances of 40ms and 60ms,

representing the minimal distance between two peaks in our ground truth samples. Notably, when the distance between peaks is set to 60ms, both our deep learning-based model and the non-parametric rule-based model exhibit superior performance compared to others.

3.2 Training deep learning models to predict peaks from the synthetic data

3.2.1 Peak Detection on the Synthetic SCG Data

We conducted a series of experiments using Synthetic SCG data to determine if deep learning models can detect peaks. Firstly, we used the Synthetic stable SCG signal with various noise settings to predict whether the input signal has a peak at the given point or not. For each noise level, the model was trained on the stable data and then tested on both the unstable data and the stable test data to evaluate its performance on the same and different distributions.

The results presented in Table 3 demonstrate the capability of our deep learning model to detect peaks, particularly when trained on stable data and applied to unstable data. It's noteworthy that the inclusion of Instance Normalization and Focal Loss is crucial for model convergence, as observed when these components were omitted, leading to non-convergence.

Figures 7 and 8 provide visual insights into the performance of our model. The red line represents the ground truth, while the blue lines depict the model's predictions at corresponding points in the signal. These visualizations illustrate the model's effectiveness when trained and tested on stable data, as well as its ability to generalize to unstable data, and vice versa.

Table 3: Performance Comparison for Peak Detection on Stable and Unstable Train for Different Noise Levels

Noise	**Model Peak Detection**	**Syn. Stable SCG**		**Syn. Unstable SCG**	
		Accuracy (%)	**F1-Score**	**Accuracy (%)**	**F1-Score**
0.0	Stable Train	99.36	81.09	99.36	81.34
	Stable Train w/o FocalLoss	Not converged		Not converged	
	Stable Train w/o Norm	Not converged		Not converged	
	Unstable Train	99.44	82.60	99.46	83.08
0.2	Stable Train	99.51	85.33	99.52	85.38
	Unstable Train	99.53	85.85	99.54	86.02
0.4	Stable Train	99.41	82.21	99.41	82.24
	Unstable Train	99.34	80.32	99.33	80.15
0.6	Stable Train	99.33	80.48	99.33	80.35
	Unstable Train	99.33	80.21	99.32	79.90
0.8	Stable Train	99.11	74.64	99.11	74.63
	Unstable Train	99.07	72.00	99.08	78.06
1.0	Stable Train	99.21	77.68	99.22	77.94
	Unstable Train	98.79	64.49	98.81	65.17

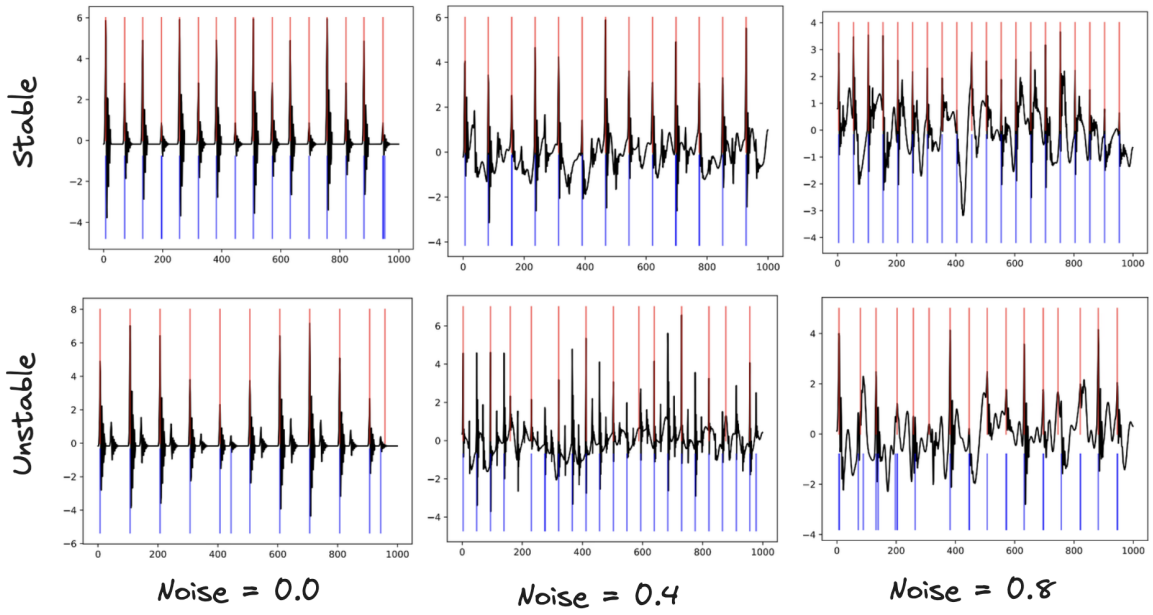

Figure 7: Visual Results of Model Trained on Stable Data

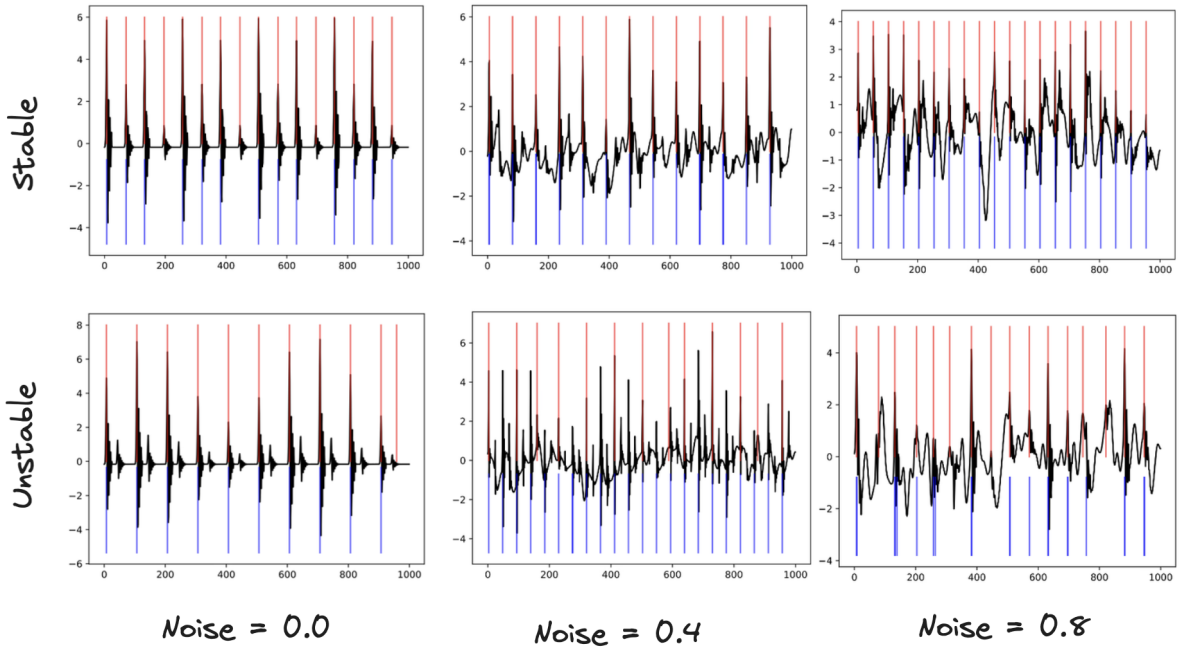

Figure 8: Visual Results of Model Trained on Unstable Data

3.2.2 Peak Detection on the Real BCG Data

As discussed in Section 3.2.1, our model exhibits the capability to be trained on synthetic stable data for peak detection and then generalize its performance to unstable synthetic data. However, a closer examination of the visualizations in Figure 1 and 2 reveals a striking similarity between the two data distributions. While this indicates success in transferring the model's knowledge from stable to unstable synthetic data, it's crucial to assess the transferability of our model in detecting peaks across different distributions.

To further investigate this transferability, we trained our model on synthetic stablee SCG data and evaluated its performance on real BCG signal data. This step is crucial as the differences between stable and unstable BCG signals may impact the model's ability to detect peaks accurately across these distinct distributions.

As shown in Figure 3, our BCG signal data is significantly noisier compared to the synthetic SCG data. Therefore, we applied smoothing to the BCG data using a bandpass filter, which helps reduce noise by allowing only the frequencies of interest (related to heart activity) to pass through, while attenuating noise outside this frequency range. For the bandpass filtering, we pre-defined the cutoff frequency as `low = 1` and `high = 15`, and set the sampling rate as `Fs = 100`. Further details can be found in the code in the file `envelope_utils/envelope_extraction.py`. This filtered BCG data is referred to as the envelope-extracted BCG signal. We also applied the same filtering to the SCG data, as our training and testing data should undergo the same signal processing process before being fed into the model.

Additionally, we conducted hyperparameter tuning on the `scipy.find_peaks()` function.

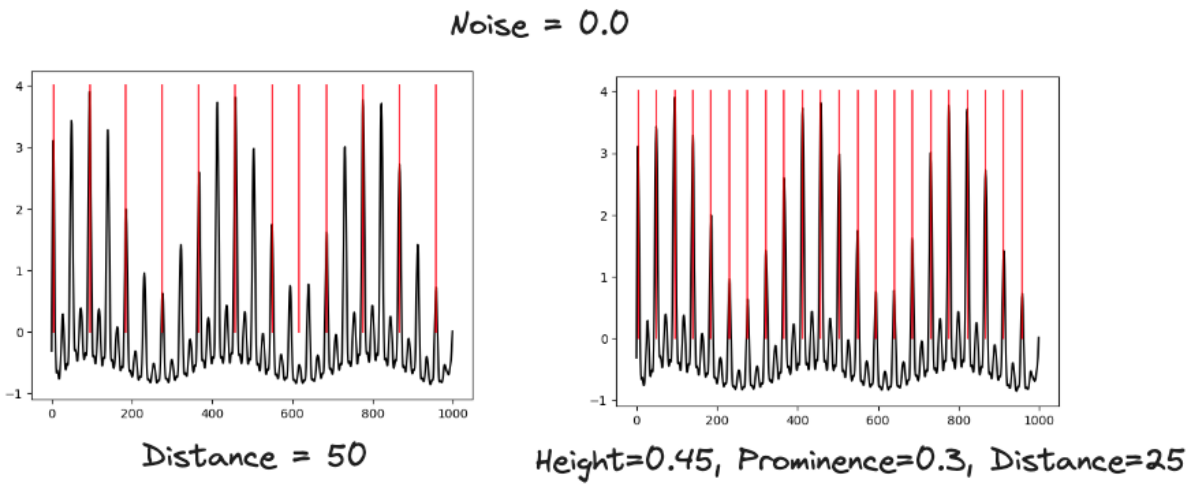

Figure 9: Visualization of the Ground Truth (red lines) for different Hyperparameters; here, only the case for Noise=0.0 for Synthetic Stable SCG is shown, but it applies to all noise levels.

As seen in Figure 9, using only `distance = 50` (indicating the minimum distance between two peaks in our ground truth) resulted in missing many peaks in the signal and incorrectly detecting non-peak signals as peaks. By setting the hyperparameter values to `height=0.45`, `prominence=0.3`, and `distance=25`, we obtained improved ground truth values for this signal, as shown in Figure 9 (right side). More details on these hyperparameters can be found in the official documentation of `scipy.find_peaks()`.

Table 4 presents the results in terms of Accuracy and F1-Score. It is evident from this table that our F1-Score was significantly impacted because the model predicted peaks that were not present in our ground truth across all noise levels. This phenomenon can be observed in the visualization of results in Figures 10 and 11. These figures show clusters of dark blue lines adjacent to each other, indicating that the model predicted peaks at locations where there were none in the ground truth for the real BCG test dataset.

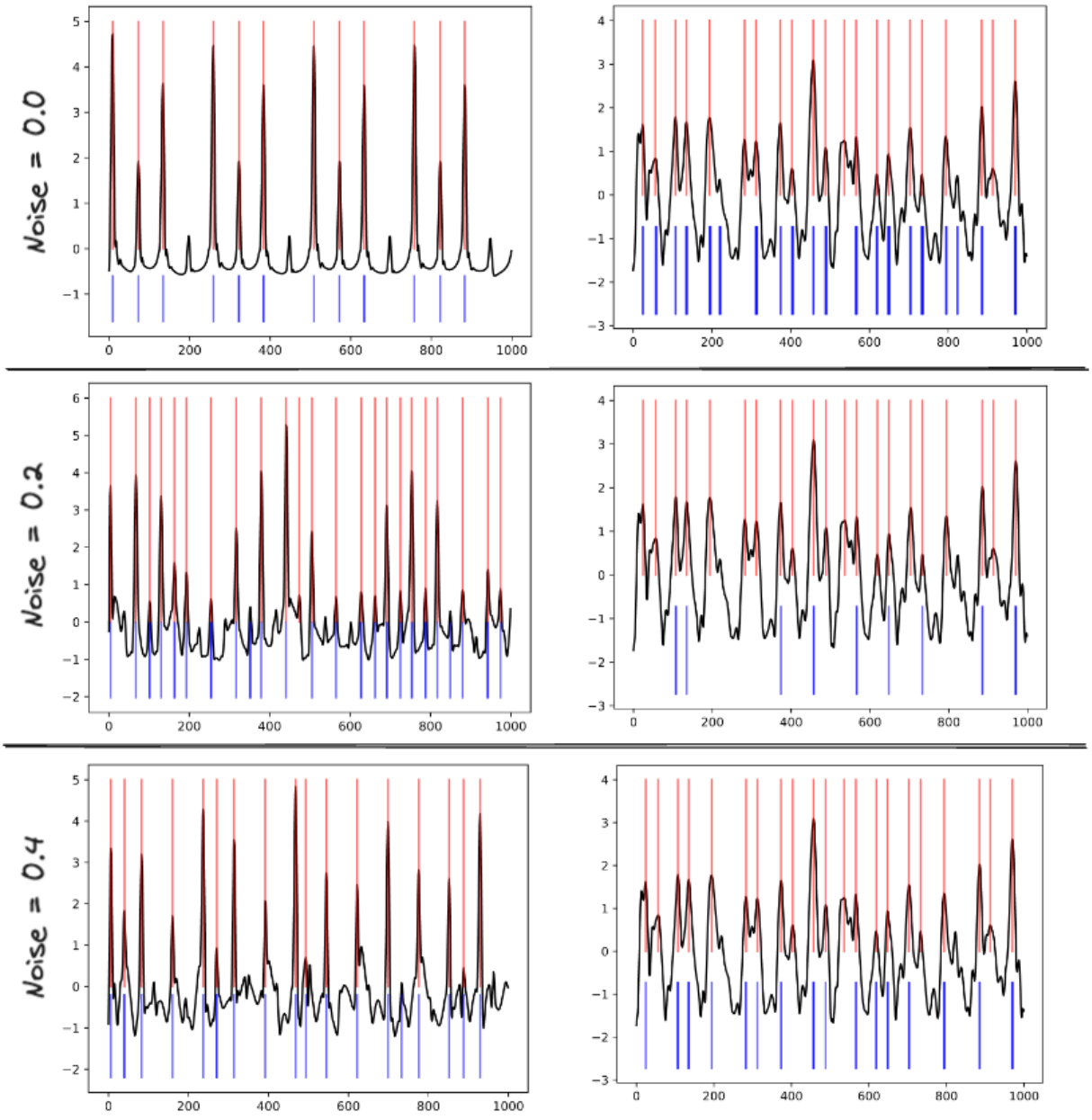

Figure 10: Visualization of results after envelope extraction method for different noise levels. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset.

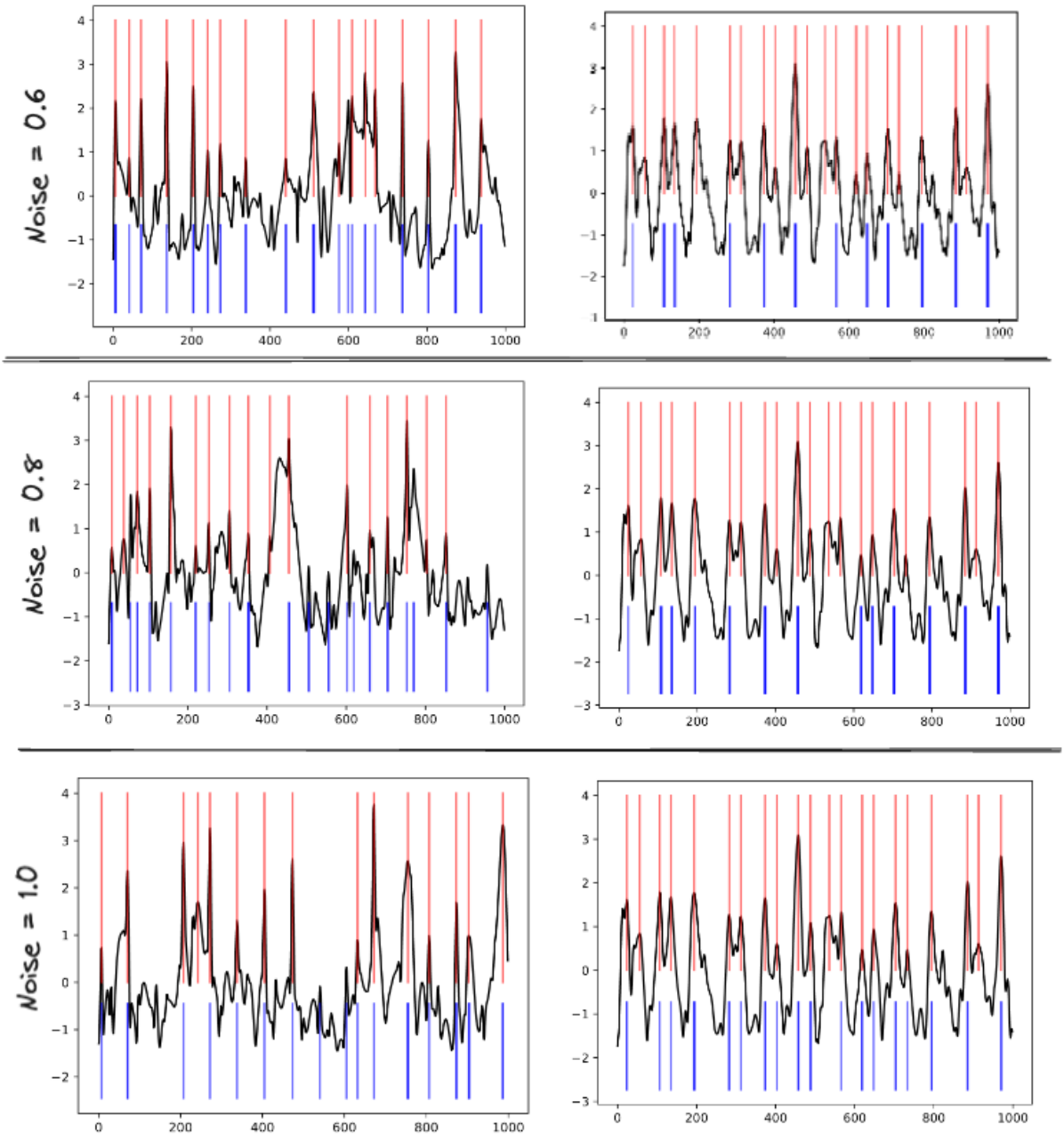

Figure 11: Visualization of results after envelope extraction method for different noise levels. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset.

Table 4: Peak detection results of training on Synthetic stable SCG and transferring that knowledge to real BCG test dataset. (Configuration: height=0.45, prominence=0.3, distance=25)

Noise	**Model Peak Detection**	**Syn. Stable SCG**		**Real BCG**	
		Accuracy (%)	**F1-Score**	**Accuracy (%)**	**F1-Score**
0.0	Stable Train	0.99.93	0.98.06	0.97.62	0.37.56
0.2		99.74	93.26	98.99	7.13
0.4		99.62	89.37	99.15	33.19
0.6		99.34	81.63	99.04	15.36
0.8		99.22	77.97	99.12	29.80
1.0		99.33	81.30	99.69	85.74

To address this issue, we conducted experiments with the window size parameter of `moving_window_integration()`. This parameter is utilized to smooth out the resultant input signal. We gradually increased the window size from 0.040 to 0.300 (specifically, [0.040, 0.080, 0.160, 0.200, 0.250, 0.300]) and reran the experiment after generating the enveloped dataset for each window size. We then trained the model on the synthetic stable SCG and assessed its transferability on the Real BCG dataset. Detailed results for all these runs can be found in the code base under the `results/` directory.

While Table 5 presents the results for all the experiments conducted with varying moving average window sizes, visualization only contains the outcomes of the two most successful runs achieved at moving average window sizes of 0.200 and 0.300.

The results presented in Table 5 demonstrate our ability to train the model on synthetic stable data and successfully transfer that learning to real BCG data. This transferability is further elucidated through visualizations in Figures 12, 13, 14, and 15. These figures provide a clearer picture of how the model performs across different noise levels and window sizes, highlighting the efficacy of our approach in capturing peak detection patterns.

Table 5: Performance of the model on peak detection with training on variable window sizes

Noise	Model Peak Detection	Syn. Stable SCG Accuracy (%)	Syn. Stable SCG F1-Score	Real BCG Accuracy (%)	Real BCG F1-Score
Window size = 0.040					
0.0	Stable Train Window size = 0.040	99.93	98.06	98.24	62.50
0.2		99.74	93.26	99.10	71.06
0.4		99.62	89.37	99.34	79.57
0.6		99.34	81.63	99.21	75.23
0.8		99.22	77.97	99.23	76.64
1.0		99.33	81.30	99.42	84.26
Window size = 0.080					
0.0	Stable Train Window size = 0.080	99.89	96.99	97.01	41.71
0.2		99.74	92.73	99.11	73.43
0.4		99.44	87.02	99.28	78.79
0.6		99.17	79.50	99.17	76.87
0.8		99.15	74.90	99.03	74.48
1.0		99.48	78.75	99.35	81.87
Window size = 0.160					
0.0	Stable Train Window size = 0.160	99.18	74.76	98.49	49.08
0.2		99.59	86.08	99.38	76.86
0.4		99.44	80.94	99.44	79.92
0.6		99.17	71.76	99.32	77.16
0.8		99.15	68.12	99.33	76.81
1.0		99.48	79.74	99.51	84.23
Window size = 0.200					
0.0	Stable Train Window size = 0.200	99.70	88.61	99.59	83.08
0.2		99.70	89.11	99.74	**90.10**
0.4		99.38	75.88	99.42	76.95
0.6		99.26	68.12	99.37	73.75
0.8		99.35	74.32	99.57	84.15
1.0		99.20	65.03	99.38	76.41
Window size = 0.250					
0.0	Stable Train Window size = 0.250	99.70	87.21	99.35	69.20
0.2		99.46	79.37	99.55	80.27
0.4		99.33	74.01	99.48	78.55
0.6		99.27	67.01	99.44	75.55
0.8		99.33	66.90	99.43	75.49
1.0		99.32	67.51	99.48	78.72
Window size = 0.300					
0.0	Stable Train Window size = 0.300	99.67	84.71	99.58	80.15
0.2		99.62	84.08	99.75	88.43
0.4		99.30	70.21	99.53	78.87
0.6		99.32	69.65	99.58	82.07
0.8		99.34	65.70	99.50	77.53
1.0		99.33	65.71	99.51	78.87

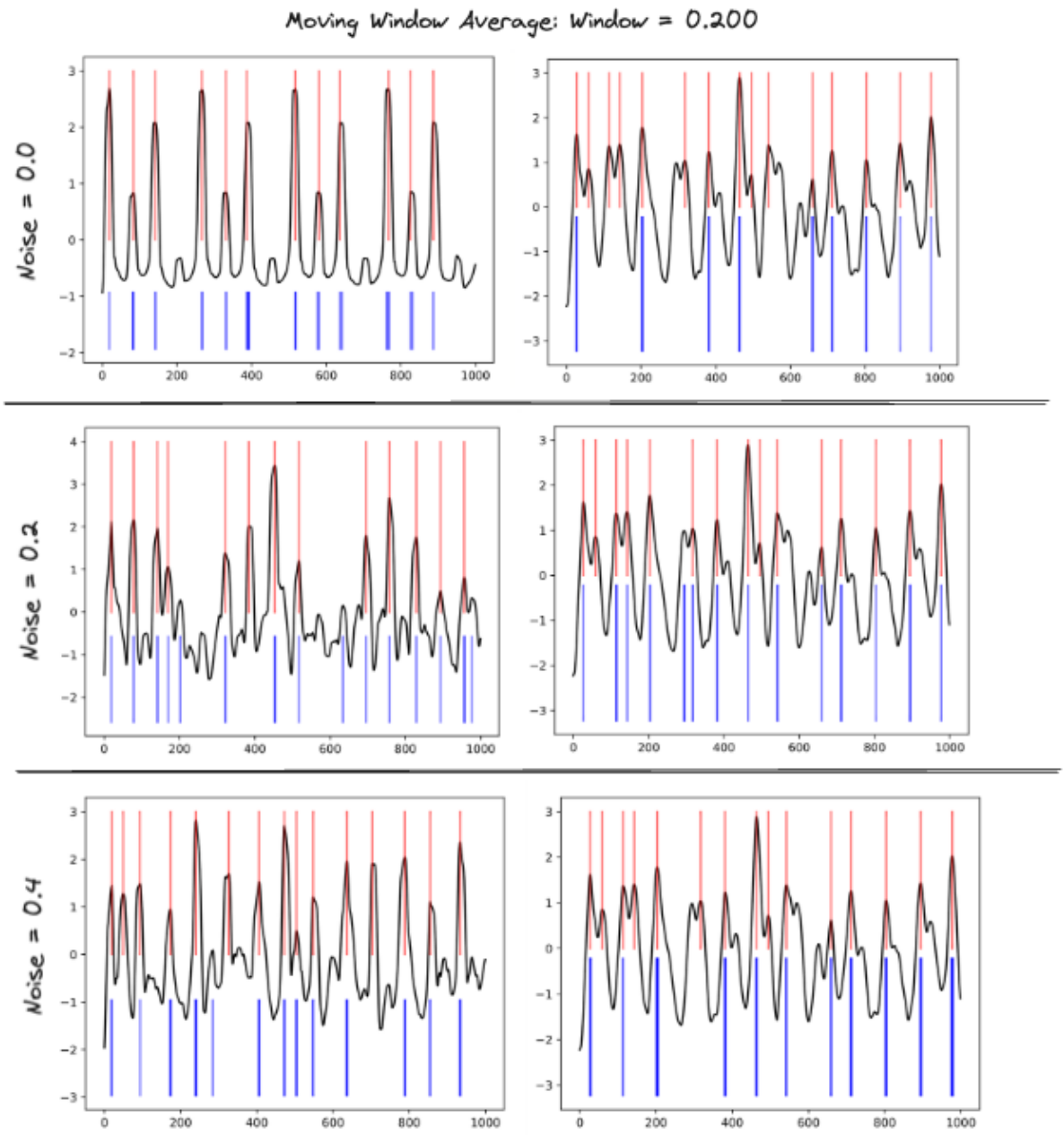

Figure 12: Visualization of results after envelope extraction method for different noise levels with moving window average of 0.200. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset after envelop extraction

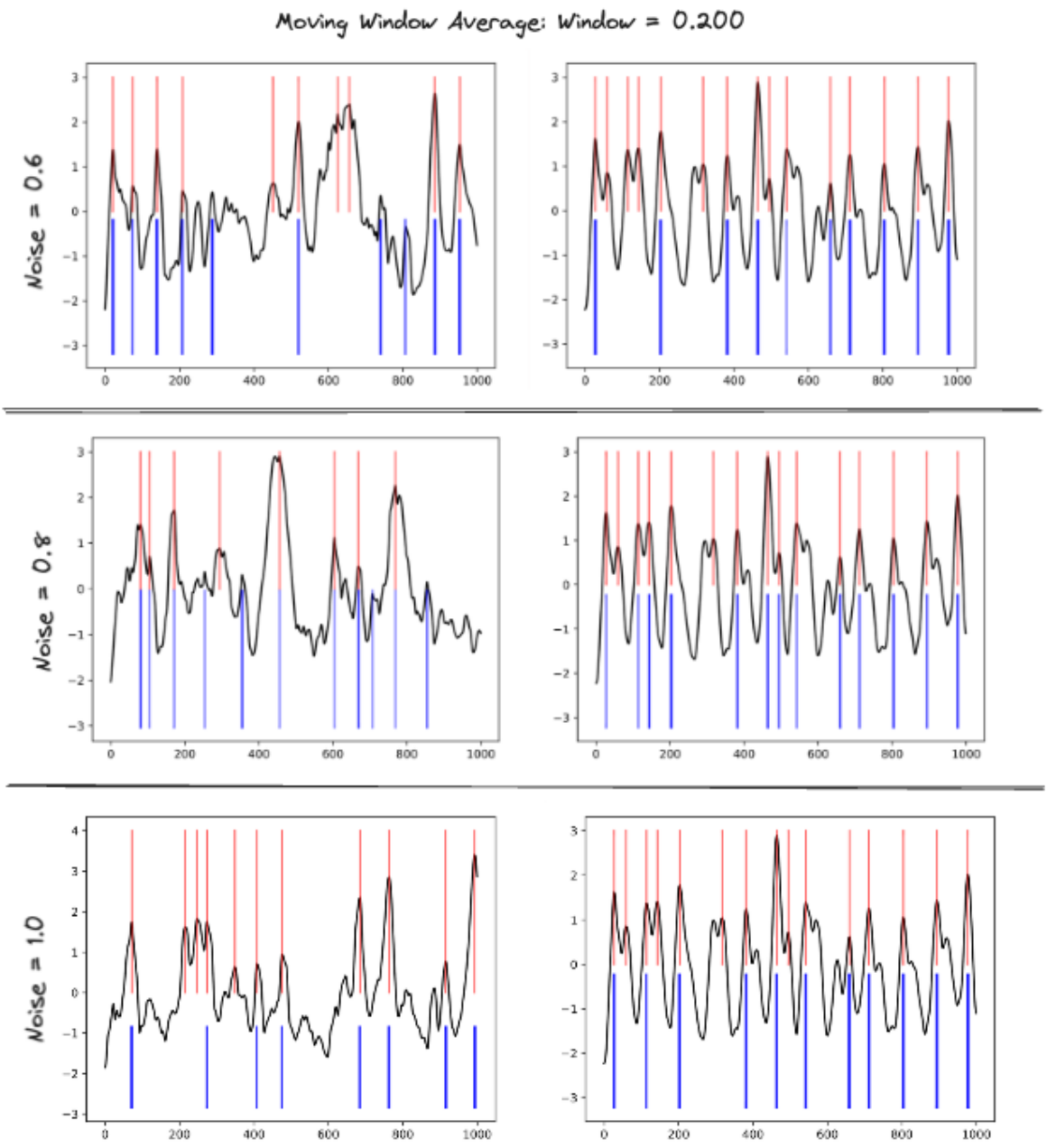

Figure 13: Visualization of results after envelope extraction method for different noise levels with moving window average of 0.200. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset after envelop extraction.

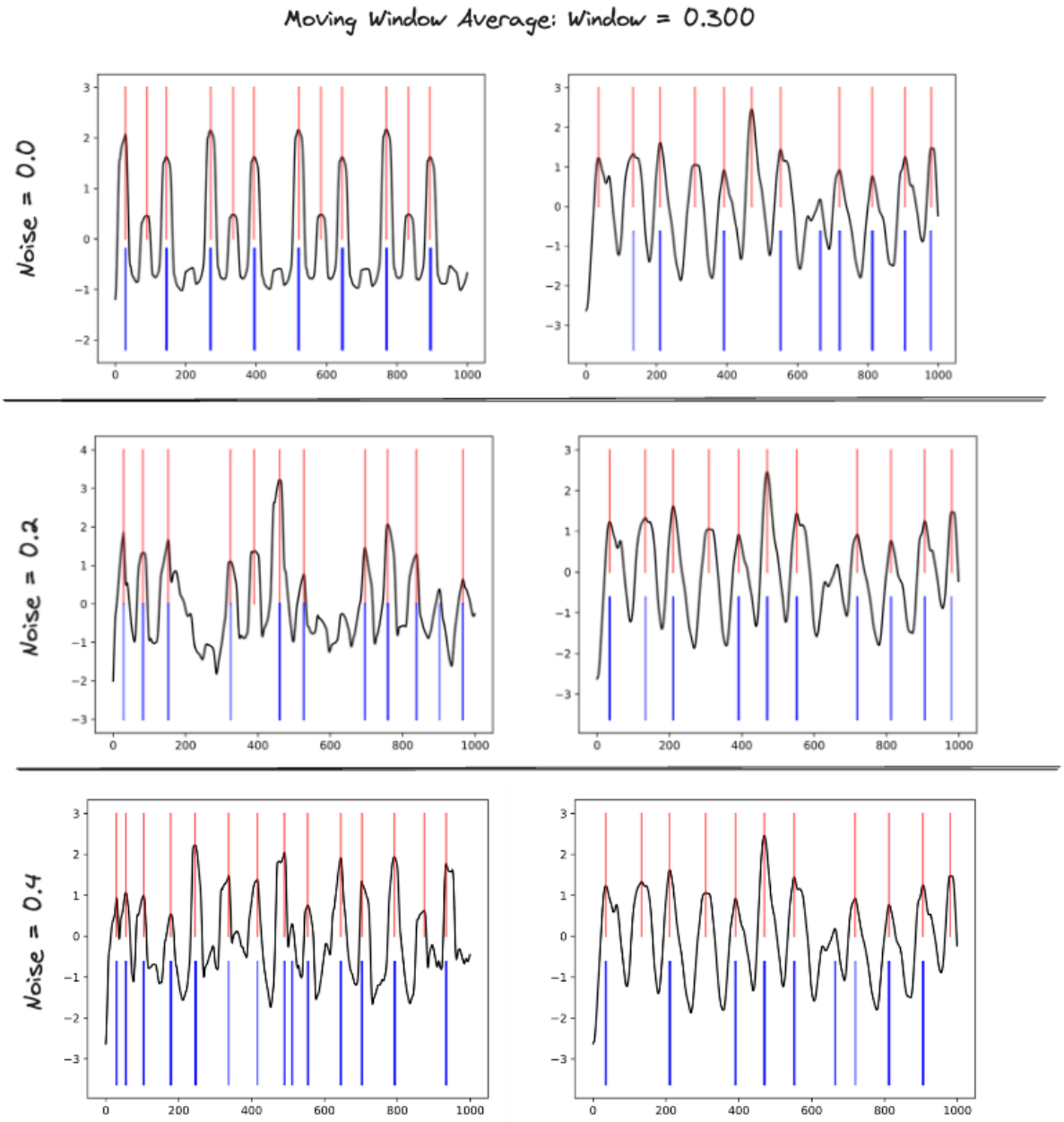

Figure 14: Visualization of results after envelope extraction method for different noise levels with moving window average of 0.300. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset after envelop extraction.

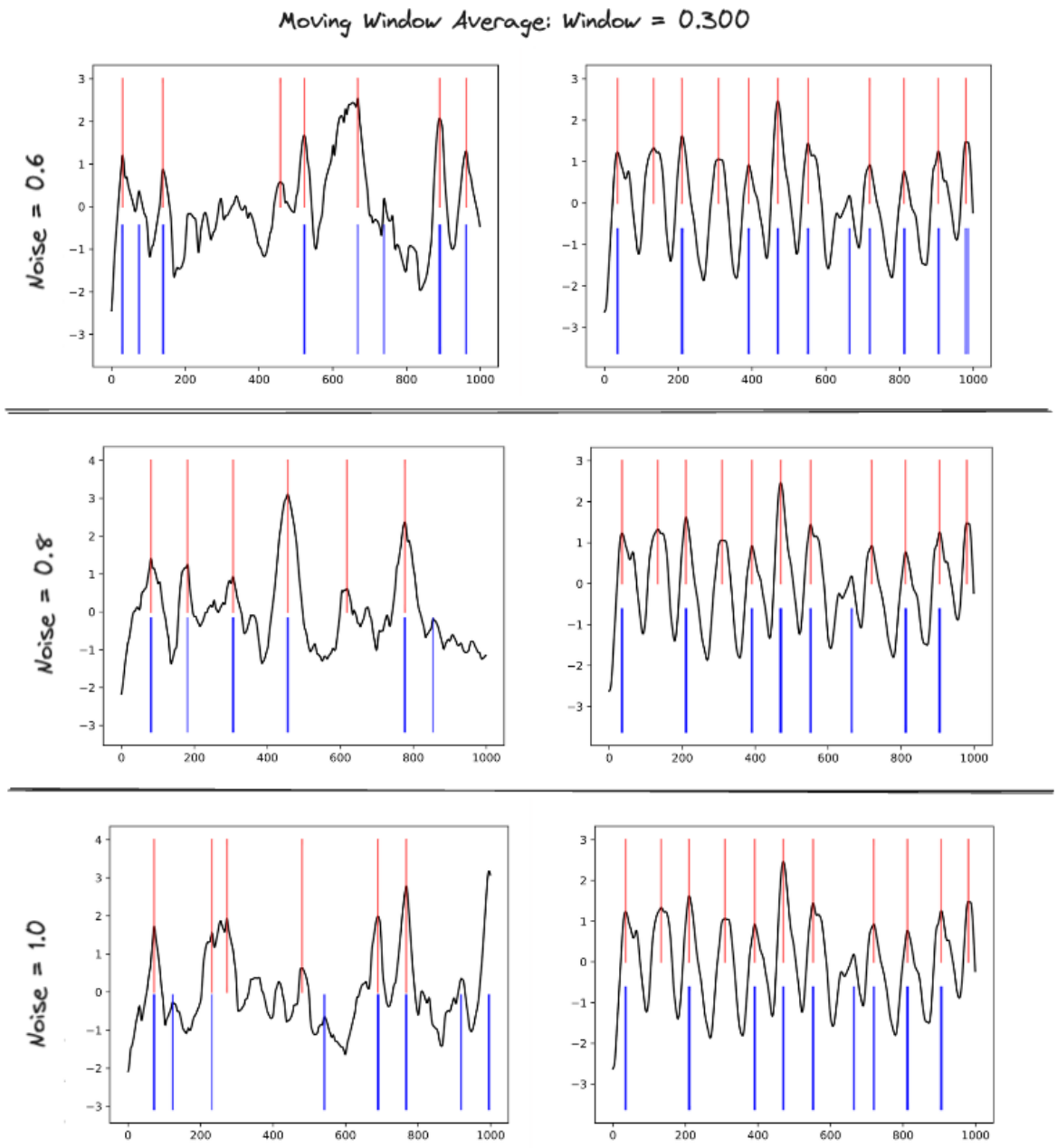

Figure 15: Visualization of results after envelope extraction method for different noise levels with moving window average of 0.300. The left side represents performance on the same distribution (Synthetic stable SCG), while the right side represents performance on the Real BCG test dataset after envelop extraction

After reviewing the results presented in Table, 5, we made the decision to train the model using a combination of window sizes corresponding to each noise level. Specifically, we merged data for noise level 0.0 across moving window average sizes of 0.200, 0.250, 0.280, 0.300, 0.320, and 0.350. We created synthetic stable SCG data for new window sizes such as 0.320, 0.350, and 0.280 based on our observation that window sizes 0.200 and 0.300 yielded superior results in the table. This additional data generation aimed to create a dataset closer to the optimal window sizes and subsequently retrain the model.

This approach was repeated for noise levels 0.2, 0.4, 0.6, 0.8, and 1.0, ensuring that the model was trained on diverse data encompassing various noise levels and window sizes. Furthermore, we aggregated data across all noise levels to further enrich our dataset, examining whether training on a diverse dataset aids the model in identifying peaks more effectively across different distributions (Real BCG Dataset).

Table 6: Peak detection results were obtained by training on synthetic stable SCG data for various noise levels, consolidated across all window sizes (0.200, 0.250, 0.280, 0.300, 0.320, 0.350)

Noise	Model Peak Detection	Syn. Stable SCG		Real BCG Window=0.200		Real BCG Window=0.250		Real BCG Window=0.300	
		Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)	F1-Score
0.0	Stable Train All Window Sizes	99.79	90.70	99.73	89.30	99.75	89.07	99.76	88.91
0.2		99.68	87.44	99.78	91.65	99.79	91.30	99.80	90.79
0.4		99.67	85.52	99.75	90.54	99.77	90.66	99.79	90.40
0.6		99.61	81.52	99.70	89.09	99.73	89.18	99.75	88.96
0.8		99.59	80.14	99.61	86.39	99.67	87.21	99.71	87.45
1.0		99.63	81.65	99.62	86.64	99.67	87.20	99.70	87.38

Table 7: Peak detection results were obtained by training on synthetic stable SCG data aggregated across all noise levels (0.0, 0.2, 0.4, 0.6, 0.8, 1.0) and window sizes (0.200, 0.250, 0.280, 0.300, 0.320, 0.350)

Noise	Model Peak Detection	Syn. Stable SCG		Real BCG Window=0.200		Real BCG Window=0.250		Real BCG Window=0.300	
		Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Accuracy (%)	F1-Score
ALL	Stable Train	99.88	94.52	99.88	95.28	99.88	95.10	99.88	94.63

Table 6 and Table 7 illustrate how enhancing the diversity of our synthetic stable SCG

dataset aids the model in accurately detecting peaks in the Real BCG dataset. Figure 16 provides a visual representation of these results, as presented in Table 7.

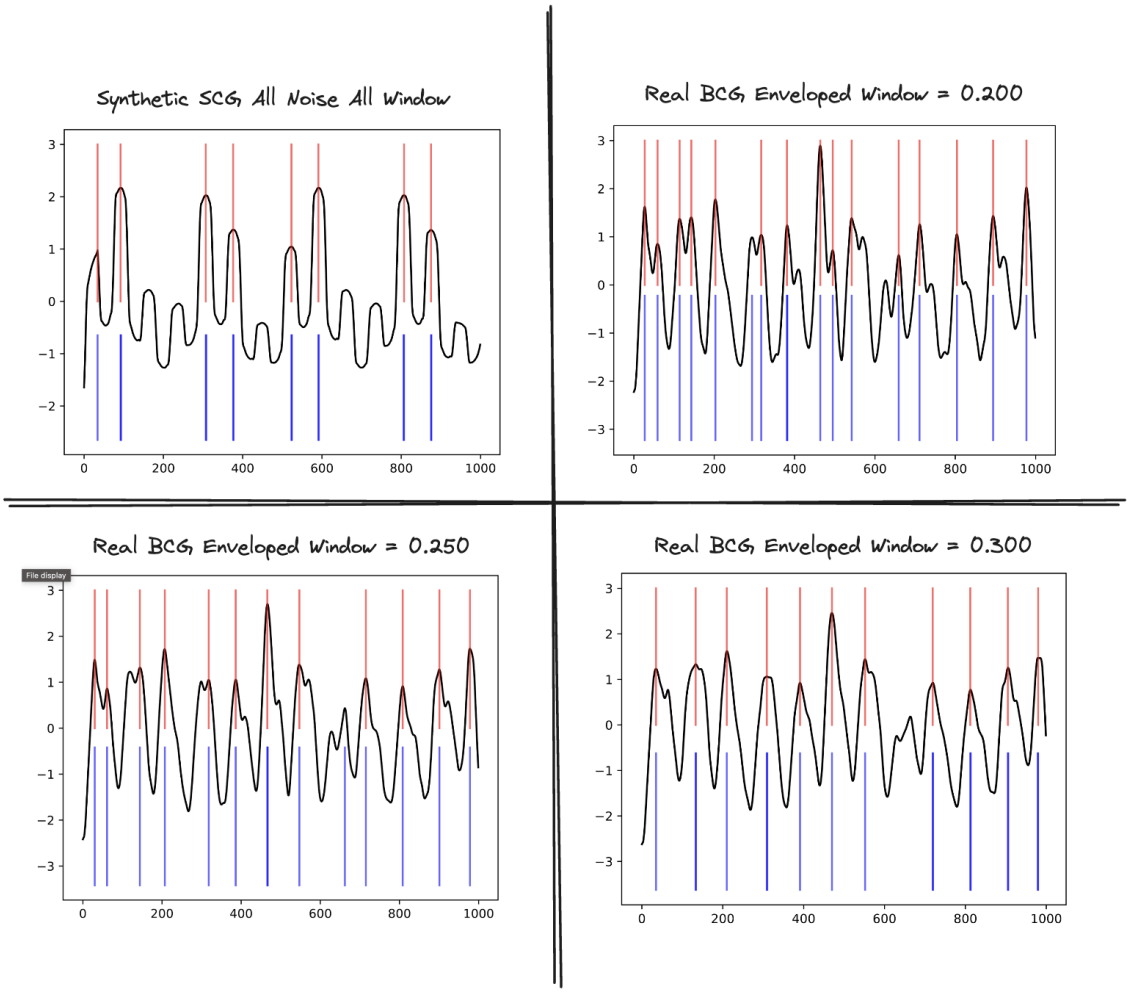

Figure 16: Visualization of results obtained by training on synthetic stable SCG data aggregated across all noise levels (0.0, 0.2, 0.4, 0.6, 0.8, 1.0) and window sizes (0.200, 0.250, 0.280, 0.300, 0.320, 0.350)

4 Conclusion

This project was centered on developing and evaluating a model for detecting peaks in heartbeat signals (ECG and BCG) and subsequently identifying atrial fibrillation (AF). A key focus was on training the model using synthetic stable data and then applying this knowledge to real BCG data. Through comprehensive experimentation involving varied window sizes and noise levels, we achieved promising results in terms of accuracy and F1-Score for peak detection tasks.

Our findings highlight the significant impact of window size selection and the benefits of aggregating information across all noise levels and window sizes. The model demonstrated robustness across different noise levels, showcasing its adaptability to real-world data variations.

Looking forward, there is potential for further enhancement by refining the labeling process to identify high-quality peak positions in stable real data more accurately. This future work can significantly contribute to improving the model's accuracy and generalizability, particularly for effective peak detection in unstable real data. Additionally, once precise peak positions are identified in both stable and unstable real data, exploring the use of these positions for predicting atrial fibrillation based on normalized peak positions in BCG signals becomes feasible.

References

[1] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database". In: *IEEE engineering in medicine and biology magazine* 20.3 (2001), pp. 45–50.

[2] Sajad Mousavi et al. "ECG Language processing (ELP): A new technique to analyze ECG signals". In: *Computer methods and programs in biomedicine* 202 (2021), p. 105959.

[3] Rui Hu, Jie Chen, and Li Zhou. "A transformer-based deep neural network for arrhythmia detection using continuous ECG signals". In: *Computers in Biology and Medicine* 144 (2022), p. 105325.