Data Augmentation for Wearables Activity Data

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute in partial

fulfillment of the requirements for the

Degree in Bachelor of Science in

Computer Science

By

____

Jai C. Patel

____

Caleb C. Talley

and in Computer Science and Data Science

By

____

Natasha R. Ussrey

Sponsoring Organization:

MIT Lincoln Laboratory

Dr. Shakti Davis

Daniel Hooks

Dr. Kajal Claypool

Date: 10/13/2023

Project Advisor:

Professor Xiaozhong Liu

Abstract

To aid early illness detection research, we developed machine learning models to augment realistic physiological data using MIT Lincoln Laboratory's Super Computer. We applied validation metrics to the generated data to compare its accuracy against the real data. We found that all five models we developed were capable of generating realistic heath data. The models' ability to augment realistic "healthy" data can improve the ongoing efforts of early illness detection.

Executive Summary

From the plague to the flu, the emergence of widespread illness is more common than not, especially with the recent COVID-19 pandemic. Early illness detection aims to detect widespread illnesses in subjects before they become symptomatic. Through early detection of an infectious disease, outbreak can be contained at the local level thereby reducing adverse effects on populations (Steele, 2020). This can be done by finding specific patterns of a person's "healthy" state to compare to their "unhealthy" state. In this project, the physiological data was collected in a research study amongst 9381 United States Department of Defense (DoD) personnel wearing Garmin and Oura fitness tracking devices (Conroy et al., 2022). The devices provided a comfortable process for subjects while collecting a range of physiological data including heart rate, heart rate variability, pulse oximetry, respiratory rate, skin temperature, step count, and sleep time (Conroy et al., 2022).

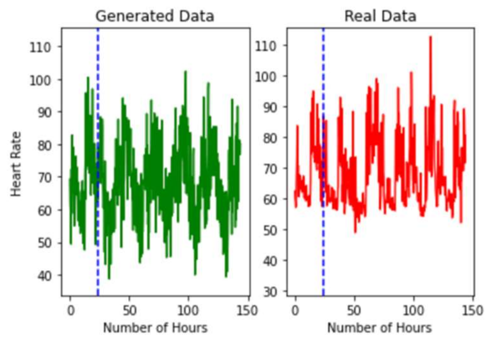
While this current form of data collection is beneficial, it can be costly, time-consuming, and raise privacy concerns amongst test subjects. To mitigate these concerns, we hope to provide an alternative way to predict early illness through data augmentation. Data augmentation is a set of techniques used to artificially increase the amount of available data by generating new data points from existing data (Mumuni et al., 2022). By creating new physiological data, we remove privacy concerns from test subjects and the time it takes to collect real data. Our project aimed to use data augmentation on wearables activity data by using a Large Language Model (LLM) with MIT Lincoln Laboratory's supercomputer (LLSC). Our project was broken down into the following sequential objectives:

1. Preprocess and visualize the raw physiological health data provided by MIT-LL
2. Create multiple machine learning models to generate healthy physiological data

3. Test and train the machine learning models using various evaluation metrics

Using various Python libraries, we created visualizations for heart rate and step count data as they were strongly correlated to each other. We did this in order to gain a better understanding of trends in the raw data before preprocessing. To prepare the data for our model, we combined the largest files of data for each feature into a dataframe. We reformatted the data itself to speed up training time of the model. Gaps in the data were likely caused from taking the watch off or turning it off. To remove these gaps, we utilized padding techniques to minimize the amount of error and training time.

Deep learning architectures such as Long Short-Term Memory models (LSTMs), Gated Recurrent Unit models (GRUs), LSTM's and GRU's with Multi-Headed Attention, and Transformers were developed using PyTorch to generate our new data. We chose these models due to their ability to learn long-term dependencies. By running the data through each of these models, we saw which architecture performed the best. We examined performance metrics within each model, such as feature selection, training time, and realistic data generation. By applying regression techniques on our models, we ensured that the generated data is a realistic representation of the inputted raw data. The regression technique we used for all five models was mean squared error (MSE).



The figure above shows the generated heart rate data (left) from one of the models in comparison to the real heart rate data (right). Ultimately, all five models created were capable of generating realistic physiological data. However, we determined that the Transformer model generated the worst data due to the large number of parameters, lack of features inputted, and the small amount of data inputted. This work of implementing data augmentation will aid researchers in early illness detection, and can be scaled up over time to generate larger amounts of data.

Acknowledgements

We would like to extend our gratitude to everyone who has assisted us in this project. We would like to thank Dr. Shakti Davis and Daniel Hooks of MIT Lincoln Laboratory for their mentorship throughout the project, meeting with us frequently to discuss our progress, and helping us become acclimated to the lab. We would like to thank Professor Xiaozhong Liu of WPI for providing constant encouragement and guidance throughout our weeks at the lab to develop meaningful results for this project. Lastly, we would like to offer our sincere appreciation to Group 23 (Biological and Chemical Technologies), specifically Dr. Kajal Claypool, for welcoming us to the team and ensuring we had an educative yet enjoyable experience at MIT Lincoln Laboratory.

Table of Contents

Table of Figures

1. Introduction

The COVID-19 pandemic's global impact demonstrated the power of infectious diseases and the amount of research needed to slow them down. A large component of slowing down the pandemic was the distribution of vaccines and quarantining. Individuals who tested positive were forced to isolate for weeks at a time. In addition, close-contacts such as family, friends, and coworkers typically isolated for a period of time in case they were infected by the positive individual.

Currently, research on early illness detection is taking place, which aims to detect illnesses in subjects before they become symptomatic. With the COVID-19 pandemic, if individuals were informed that they were going to test positive for COVID-19 before becoming symptomatic, they could have quarantined sooner. Through early detection of an infectious disease, the outbreak can be contained at the local level thereby reducing adverse effects on populations (Steele, 2020).

The problem, however, is that there is a vast amount of raw physiological data that is necessary to train detection algorithms. With small- and medium-sized datasets, detection algorithms will not provide accurate and representative predictions. The current data collection efforts can be costly, time-consuming, and raise privacy concerns amongst test subjects. In addition, some physiological data collected prior to the pandemic may be outdated, as the general populations' physiology has been impacted by the use of vaccines and changes in peoples' lifestyles.

In the ongoing early illness detection research, amounts of physiological data have been collected using wearable fitness trackers such as Garmin watches, FitBits, and Oura Rings. Fitness trackers have risen in popularity over the last decade, with a recent study reporting that

21% of Americans regularly use fitness trackers or smartwatches (Abdelhamid, 2021). These devices provide a seamless and comfortable process for the participants while collecting a wide range of physiological data, such as:

- Heart Rate
- Heart Rate Variability
- Pulse Oximetry
- Respiratory Rate
- Skin Temperature
- Step Count
- Sleep Time

This paper focuses on data augmentation, which is a set of techniques to artificially increase the amount of available data by generating new data points from existing data (Mumuni et al., 2022). The existing data that we are working with was collected in a research study amongst 9381 United States Department of Defense (DoD) personnel wearing Garmin and Oura fitness tracking devices. The size of the data totals over 599,174 user-days of service and over 200 million hours of recorded data (Conroy et al., 2022).

In order to efficiently augment the data, we will begin by preprocessing the data, followed by developing machine learning models. We will test the accuracy and efficiency of these models by applying validation metrics on the generated data.

2. Background

The project aims to augment wearables activity data by using a Large Language Model (LLM) with MIT Lincoln Laboratory's supercomputer (LLSC). We will first preprocess and visualize the data using common Python libraries. Deep learning architectures, such as recurrent neural networks (RNNs), long short-term memory models (LSTMs), gated recurrent unit models (GRUs), and Transformer models will be developed to generate the new data. We will conduct training and testing by applying mean absolute error (MAE), mean squared error (MSE), coefficient of determination (R^2), and other techniques on the data augmentation model to ensure that the generated data is a realistic representation of the inputted raw data.

2.1 Data Augmentation

The prediction accuracy of deep learning models is largely dependent on the amount of diverse data available to the model. For this project, increasing the amount of wearables fitness data can directly improve the accuracy of the early illness detection models.

Data augmentation has a variety of applications, such as audio data for speech recognition models, text data for natural language processing, and image data for healthcare. However, it is important to note the difference between augmented data and synthetic data. Augmented data refers to artificially-generated data that is driven from an original dataset(s); synthetic data refers to data that was created from scratch by using Deep Neural Networks (DNNs) to generate more data (Mumuni et al., 2022). In the case of this project, we will use the original data from the Rapid Analysis of Threat Exposure (RATE) study to train the models and augment the data.

2.2 RATE Study

The research study mentioned previously, RATE, collected wearables fitness data from military personnel. The research specifically contributed towards detecting COVID-19 in an active workforce such as the military. Of the 9381 participants, 98.6% wore Garmin smartwatches and 89.8% wore Oura rings, with 88.4% wearing both (Conroy et al., 2022). The participants were instructed to wear their devices for the entire study, unless the devices restricted them in their workplace, needed to be charged, or became uncomfortable.

The devices recorded a wide variety of physiological data, including heart rate, heart rate variability, respiratory rate, step count, sleep time, body battery, metabolism, and blood oxygenation. We will utilize a total of 1,050 participants' data collected from the RATE study.

2.3 Preprocessing Health Data

Data preprocessing refers to the process of transforming data before inputting it into an algorithm or machine learning model (Benhar et al., 2018). This transformation takes raw data and changes it into an improved dataset through cleaning, normalization, integration, and other formatting techniques. Applying these techniques to physiological data is very important to data augmentation. Normalization must be applied to almost all diverse datasets due to the different scales of health data. For example, the average heart rate is between 60-200 beats per minute, whereas the average respiratory rate is 12-18 breaths per minute (Laskowski, 2020). Cleaning must also be applied to many health datasets; noise and missing data are two of the prime challenges in medical datasets (Benhar et al., 2018). Thus, removing the outliers and noise can result in better predictive modeling.

2.4 Deep Learning Architectures

A neural network is a deep learning model that uses interconnected nodes/neurons in a layered structure to process data. The input layer holds the input data, the hidden layers process the input data, and the output layer produces and/or decodes the predicted result of the whole network. Weights and biases connect a given vector input to the given network (Amazon Web

Services, 2023). We will use multiple variations of neural networks that are specifically designed for time-series data and generative usage.

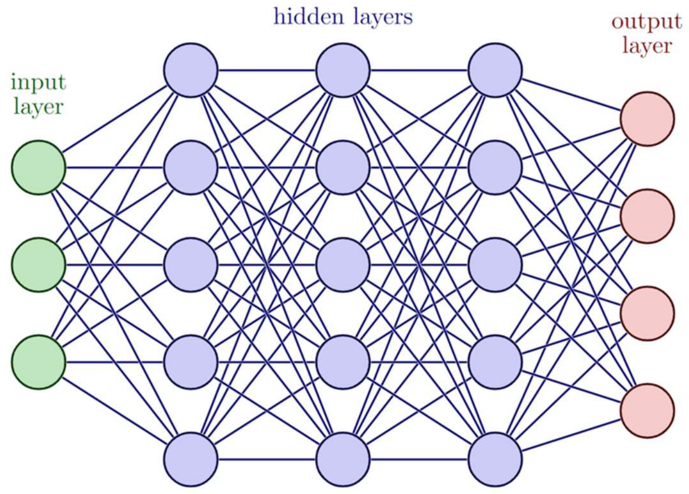


Figure 1: Simple Neural Network

2.4.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a variation of neural networks designed for time-series and sequential data. RNN modules process input data sequentially by having each cell of the module process one individual batch of the input(s). The number of cells in the RNN module are determined by the length of the input. The connected cells keep past information by maintaining a hidden state, which serves as a short-term memory for the model as represented by *h* in the figure below. (Amidi and Amidi, 2019).

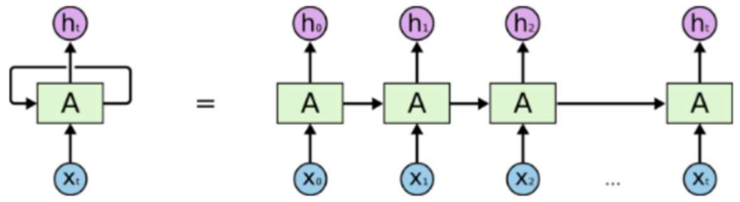


Figure 2: Recurrent Neural Network (RNN) Module (Olah, 2015)

Long Short-Term Memory models (LSTMs) are a variation of RNNs that are capable of learning long-term dependencies through a more-thorough learning process. LSTM cells have a cell state, which serve as the primary long-term memory for the model. An LSTM cell takes in the previous cell's input data, hidden state, and cell state. The LSTM cell will return a new hidden state and an updated cell state, which can then be used in the model's next module (Olah, 2015). To update both the hidden state and the cell state, a module utilizes a forget gate, input gate, and output gate. The forget gate uses the previous hidden state and input to determine what information in the cell state is relevant and should stay stored. The input gate determines what information from the hidden state and input will be stored in the updated cell state. The output gate uses the previous hidden state and input along with the updated cell state to create a new hidden state. From here, the same process is repeated with the new hidden state, the updated cell state, and a new input (Olah, 2015).

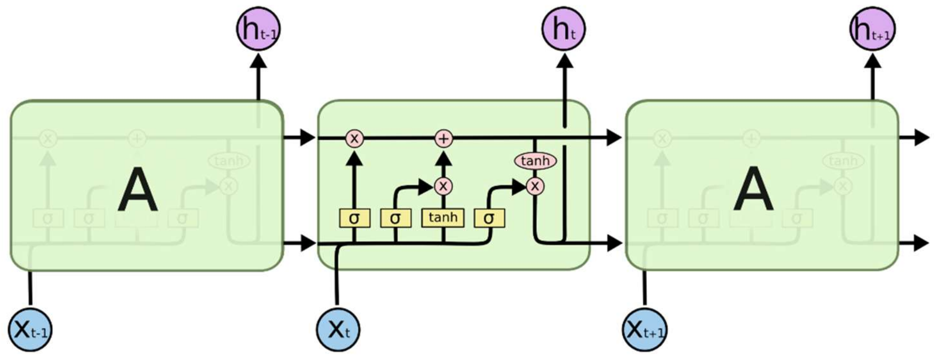


Figure 3: Long Short-Term Memory (LSTM) Module (Olah, 2015)

Gated Recurrent Unit models (GRUs) are a variation of the LSTM model. This architecture combines the forget gate and input gate to create what is called the "update gate" (Olah, 2015). This gate helps determine what previous information needs to be passed on to the next cell. The "reset gate" determines how much of the previous information to forget. The GRU model has just one continuous memory as opposed to both a short-term memory and a long-term memory.

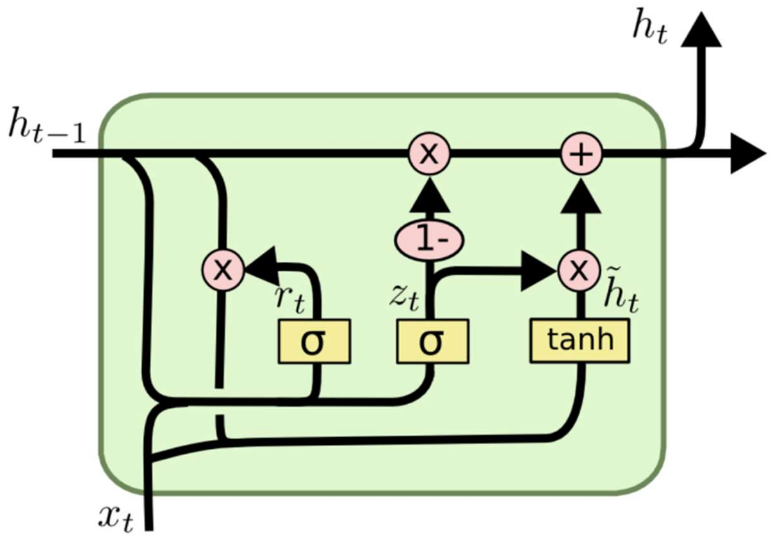


Figure 4: Gated Recurrent Unit (GRU) Cell (Olah, 2015)

One of the main drawbacks for RNNs is their difficulty to use previous data to connect the information stored. This happens when there is a large gap in sequential data where the relevant information is needed in context (Olah, 2015). Variations of RNNs, such as LSTMs and GRUs, successfully remedy this limitation. Another drawback of RNNs, even with its variations such as LSTMs and GRUs, is that they can only take in data sequentially. This inherently prevents parallelization which becomes critical with longer sequence lengths, as memory constraints limit batching across samples (Vaswani et al, 2017). Machine learning mechanisms, such as self-attention, were created to remedy these limitations of RNNs.

2.4.2 Self-Attention

Self-attention mechanisms relate different positioned inputs of a sequence to create a robust, generic numerical representation. This will help the model to focus on the most relevant information from the input when generating an output. Self-attention mechanisms can process all parts of the input simultaneously, which is far more efficient than models that sequentially take in information (e.g. RNNs, LSTMs, GRUs). Multi-head attention uses parallelized, separate self-attention mechanisms as opposed to one large self-attention mechanism. This allows the model to learn various contextual relationships in a computationally-efficient manner.

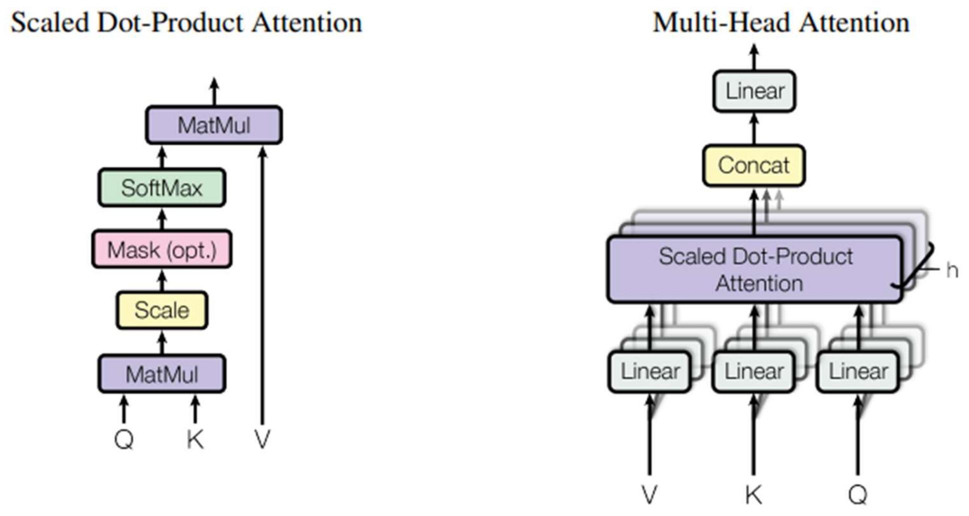


Figure 5: Self-Attention Architecture (Vaswani et al, 2017)

Self-attention can either be added to existing deep learning architectures, such as LSTMs or GRUs, or can be used as a basis for an entire deep learning model, such as Transformers.

2.4.3 Transformers

The Transformer model is a neural network that relies on self-attention mechanisms rather than recurrence to create global relationships for the input sequence. The Transformer uses these relationships to generate a linguistic output (Vaswani et al, 2017). This model has two main processing steps: encoding the information and decoding the information.

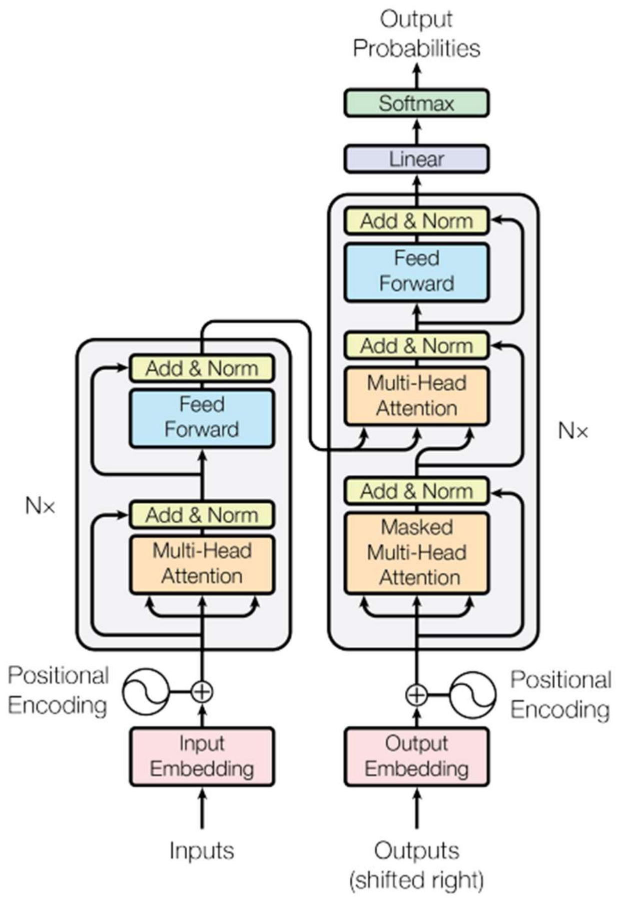


Figure 6: Transformer Architecture (Vaswani et al, 2017)

To encode the incoming information, Transformers have an encoder. The encoder takes in an input, and outputs an encoded vector representation given after performing self-attention. Transformers use a decoder to decode the encoded vector representation, then to recursively generate output predictions of similar type to the original input.

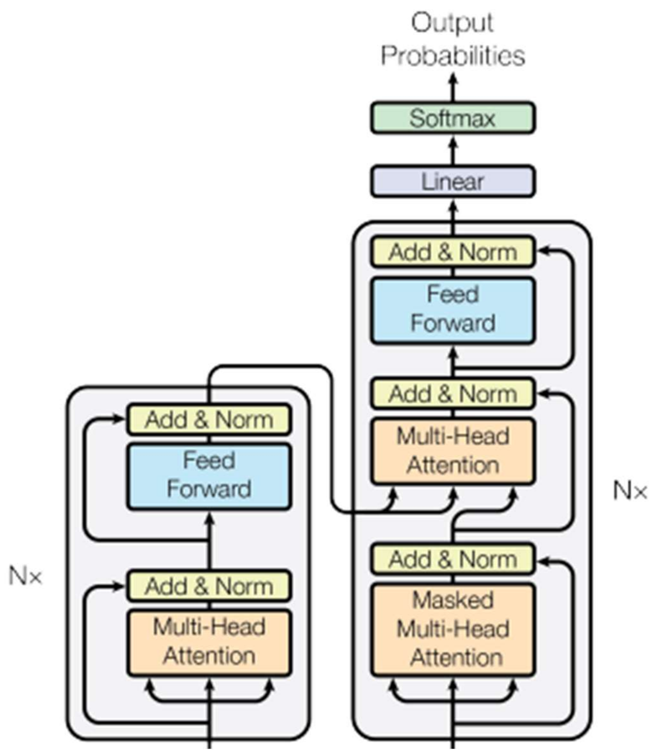


Figure 7: Transformer Encoder (Left) and Decoder (Right) (Vaswani et al, 2017)

Transformer encoders and decoders are frequently used to create extremely powerful AI systems and applications. For example, Open AI's Generative Pre-training Transformer (GPT), which is used to create ChatGPT, is made with multi-layered Transformer decoders.

2.5 Evaluation Metrics

In machine learning, evaluation metrics are used to determine the quality of a given model by measuring its performance. These metrics use various statistical functions to determine a model's accuracy, compare models to one another, and assist in improving the model's predictive power through training and testing. The type of evaluation metrics we are using are for regression, which is generally used to predict the numerical relationship of the input data. We will be using mean absolute error (MAE), mean squared error (MSE), and coefficient of determination (R^2). MAE uses the average distance between the predicted and original values of the data to determine the amount of error the model provides. MSE determines the amount of error the model provides in the same way as MAE, except it takes the square of the average distance between the predicted and original data values. The MSE metric is used alongside MAE to determine the larger errors in our model. R^2 is the amount of predicted variance explained by the model divided by the actual variation explained by the real data. The higher the R^2 metric is, the greater the model will fit the data.

3. Methodology

This project aimed to augment wearables activity data by using a Large Language Model (LLM) with MIT Lincoln Laboratory's supercomputer (LLSC). The project was broken down into the following three sequential objectives:

1. Preprocess and visualize the raw physiological health data provided by MIT Lincoln Laboratory
2. Create multiple machine learning models to generate realistic, healthy physiological data
3. Test and train the machine learning models using various evaluation metrics

We predicted that the Transformer model would have the overall best accuracy. Transformers, as opposed to LSTMs and GRUs, take in the data simultaneously as opposed to sequentially, which we believe will allow for a better overall representation of the data. The LSTM and GRU models were predicted to have better training time due to their ability to train with a smaller number of parameters and data. We added Multi-Head Attention to LSTMs and GRUs to get the benefits of both RNNs and Transformers, which we predict will have the best balance of training time and accuracy.

3.1 Visualization

We created visualizations such as line graphs, bar graphs, and histograms to get a better understanding of trends in the raw data before preprocessing. Of the many types of physiological health data collected, we visualized only heart rate and step data as they are strongly correlated to each other. With both the heart rate and step data, we analyzed different sizes and types of samples from each. All of the visualizations were done using Python's Matplotlib library.

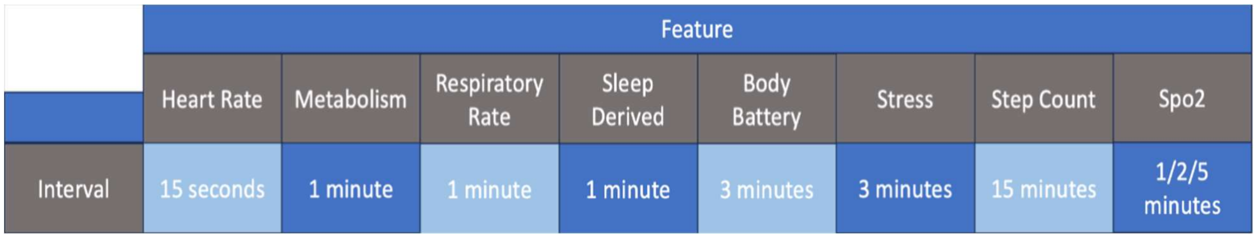
First, we plotted histograms of just one person's data, so we could see the amount of data collected per hour of the day. We were able to see hours of activity and increased data records versus hours where the wearables were turned off or the user was asleep. Next, we increased the size of the histograms by plotting 10 people's data, followed by 20. Increasing the size of the plotted data allowed us to visualize a more representative distribution of the data. We utilized Python's NumPy library, specifically the numpy.random.seed() to randomly pick data.

Next, we ordered all of the heart rate and step data by file size to identify the individuals with the smallest and largest amount of recorded data. By ordering the data, we were able to store the head, which is the five smallest datasets, and the tail, which is the five largest datasets. We then created histograms: one head histogram and one tail histogram. By visualizing the head and tail graphs, we were able to see the trends for individuals with a large set of data and those with a small set of data.

Following the histogram visualizations, we decided to plot line graphs of the data, with the x-axis being the hour of the day and the y-axis being the average heart rate per hour/average step count per hour of a given individual. Similar to the histograms, this allowed us to visualize trends in average heart rate and step count for users depending on the hour of the day.

3.2 Preprocessing

The data we used was stored in folders, with each folder grouped by their respective feature. Each physiological features' data was recorded at different time intervals as shown below:

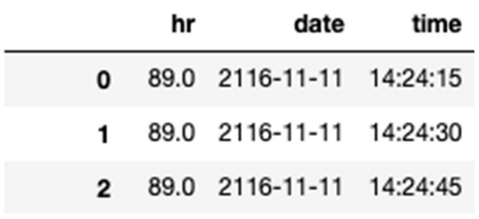


	Feature							
	Heart Rate	Metabolism	Respiratory Rate	Sleep Derived	Body Battery	Stress	Step Count	Spo2
Interval	15 seconds	1 minute	1 minute	1 minute	3 minutes	3 minutes	15 minutes	1/2/5 minutes

Figure 8: Features vs. Time Intervals

Within each folder, the data was stored in parquet files which each contained an individual's physiological data recorded over months of time. To prepare the data for the model, we first needed to convert the data stored in the parquet files into tensors.

The first step of the conversion was to sort all of the heart rate file paths by size in order, from smallest to largest. We then separately stored the last 100 file paths, which gave us access to the largest datasets. These 100 largest datasets will be referred to as the tail. We wanted to use the largest datasets to get the best possible representation of the general population who used Garmin watches. These people who had the most data made sure to remove the Garmin watch while not in use, which could have produced incorrect heart rate readings of 0 bpm. Once the tail paths were stored, we loaded in all of the data, and appended it to a dataframe. We modified columns in the dataframe by dropping irrelevant ones such as "id_event" and "id_device". Then, we reformatted the "collection_timestamp" column into two separate columns, "date" and "time", as shown below:



	hr	date	time
0	89.0	2116-11-11	14:24:15
1	89.0	2116-11-11	14:24:30
2	89.0	2116-11-11	14:24:45

Figure 9: Data Columns in Dataframe

Next, we decided to reformat the data by changing the time intervals from every 15 seconds to every 15 minutes to accommodate to the LLSC's memory usage. Reformatting the data in this way sped up the time to train the model, as changing the interval from every 15 seconds to 15 minutes decreases the amount of redundant data. For every 15-minute period, the heart rate value was equal to the mean of all the heart rate values of each 15-second interval within that period.

Once the data was reformatted, we went through the 100 users' data and attempted to gather a week's worth of consecutive heart rate data for each individual with no gaps. For those with gaps in their data, we utilized padding techniques. The padding was only applied on users with at least 85% of a week's worth of data to minimize the amount of padding necessary. The gaps in data were likely caused from taking the watch off, turning it off, or a misreading by the wearable's sensors. To pad the data, we first created a function that replaced each gap with a NaN value. While these values worked when computing the mean and standard deviation, they were not usable for the model's calculations and metrics. We also tried padding the data using the mean value of all the data, but that skewed the model significantly. Instead of padding the data with the overall mean value, we padded the data using the local mean for each gap. This eliminated any calculation difficulties and significant skewing. We substituted the gap values with the local mean by first identifying the empty values in the dataset, finding the nearest values to its left and right, then using the mean of the two values to replace the gap. This process was repeated until all gaps in the dataset were filled. This resulted in a dataframe filled with 81 peoples' three weeks' worth of consecutive heart rate data.

Once the 81 individuals' heart rate data were gathered and normalized, we defined the length of the set of intervals we want for each row of the data. For example, if we want to have a day's worth of data for each row, and the data is recorded in 15-minute intervals, then we would have 96 intervals per row (4 time-intervals per hour * 24 hours per day).

After figuring out how many intervals we want per row, we split the data into X (data), and Y (label) pairs. X consisted of all of the weeks' data for the individuals, except for the last set of intervals. If each set of intervals is equal to a day's worth of data, the last day of data would be omitted because there is no access to data beyond the last given day. Y consisted of all the data except for the first set of intervals. If each set of intervals is equal to a day's worth of data, the first day of data would be omitted because we do not have access to data previous to the first day.

We now have the X and Y pairs for a set length of time. With this, we vertically sliced and appended the pairs such that they were equal length to the number of periods we were predicting. This allows us to randomly select the necessary sets of data more efficiently.

Once all of the data formatting and splitting was completed with the X and Y pairs, we reorganized the rows for X and Y randomly and created training, validation, and testing sets of data. 60% of the data was for training, 20% was for validation, and 20% was for testing.

3.3 Models

The five deep learning models we implemented to augment the data were Long Short-Term Memory (LSTMs), Gated Recurrent Unit (GRUs), Transformer, LSTM with Multi-Head Attention, and GRU with Multi-Head Attention. All of these models were developed using PyTorch, a machine learning framework for Python. We chose these models due to their ability to learn long-term dependencies. By running the same data through each of these models, we were able to see how well the models performed on the test set, which would indicate how well the models learned the patterns of the data. We also saw the training speed for each of these models, which indicated the tractability and efficiency of the models.

3.3.1 LSTM and GRU

LSTM and GRU were the first two deep learning models we implemented, as they are popular variations of RNNs and are simple compared to Transformers. The main tradeoff between these two models is that LSTMs are generally more powerful due to their greater number of gates and a separated long-term and short-term memories; GRUs are a simpler architecture and are faster to train.

The LSTM and GRU models we created took in one time interval of data for each cell. This led to the number of cells for each module to be equal to the number of time intervals per row. To get the predictions, we first got the final cell's hidden state. We then applied a fully-connected layer to the hidden state to produce the final predictions.

3.3.2 Transformer

The Transformer was the next model we decided to implement. Compared to RNNs, such as LSTMs or GRUs, Transformers are seen as more versatile, meaning they are better at making long term dependencies. However, Transformers require more training data compared to RNNs to obtain their best performance, which can lead to a worse performance by the model on smaller datasets. Then, each interval of the training data was inputted into the Transformer model simultaneously. This data was taken in once by the encoder and once by the decoder.

3.3.3 LSTM and GRU Models with Multi-Headed Attention

In order to create a model that has a level of complexity between LSTMs, GRUs, and Transformers, we created models that use an RNN architecture with a multi-head attention mechanism. After the RNN module returns its final hidden state, we used three separate neural networks to create query, key and value tensors. These query, key, and value tensors are placed into a multi-head attention module. The output of the multi-head attention module is then placed into a fully-connected layer which produces the output predictions. This process is shown below in Figure 10.

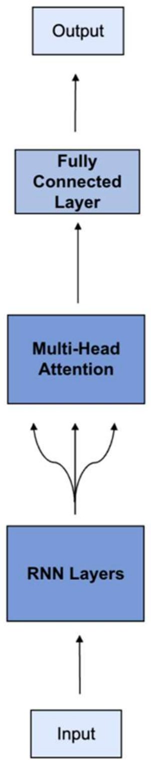


Figure 10: LSTM/GRU model with Multi-Head Attention Architecture

3.4 Hyperparameters

Hyperparameters are parameters that control the learning process of the models, which must use optimal hyperparameters in order to learn the data in the best way possible. The first hyperparameter selected was the number of epochs for training the models. Each epoch passes the training set through the models once, thus having an adequate number of epochs was important to train the models. We started with a single epoch, and slowly increased the number of epochs to find the optimal value. A number of epochs that is too high could result in overfitting, where the models become too familiar with the training set and would not perform well on the test data. A number of epochs that is too low could result in underfitting, where the models are unable to capture patterns of the data.

The next hyperparameter selected was the learning rate, which is the value that controls the rate at which each model updates and learns parameters. A higher learning rate could result in unstable predictions due to overshooting the gradient. A lower learning rate could increase the training time greatly.

The final hyperparameter we selected was batch size. The batch size specifies how many samples are processed by a model before altering the model's parameters. While testing smaller batch sizes, we found that the training time increased greatly. While testing larger batch sizes, the test loss increased due to the models not detecting particular patterns in the data.

3.4.1 Model-Specific Hyperparameters

For the LSTM and GRU models, we focused on the size of the hidden state and the number of layers. For the Transformer model, we focused on the number of heads in the Multi-Head Attention layers and the number of sub-encoder layers in the encoder. For the LSTM and GRU models with Multi-Head Attention, we focused on the size of the hidden state, the number of layers, the number of heads in the Multi-Head Attention layers, and the embed dimension of the model.

3.4.2 Hyperparameter Tuning

In order to find adequate hyperparameters for our models, we performed a grid search across all of the hyperparameters, using every possible configuration of a discrete set of hyperparameter values. We trained the model with each hyperparameter configuration, then evaluated which configuration was better by using the validation dataset. We performed this hyperparameter tuning for each of the models.

3.5 Data Augmentation

After tuning the hyperparameters for the models with a grid search and training the model, we generated data using a bootstrapped sample. We first randomly selected a person's first set of intervals from the original dataset. We then used that set of intervals to generate the next set of intervals. We then used our newly-generated set of intervals to generate the next set of intervals. After creating all of the data, we denormalized this data in order to have comprehensive visualizations. By using this process indefinitely, we were able to create any amount of data we want.

3.6 White Noise

The augmented data should reflect seasonality based on various cycles, such as awake vs. asleep and weekend vs. weekday. However, we want to avoid the cycles becoming repetitive and almost identical. To achieve this, we applied white noise to the data using a normal distribution. This process added random values between an absolute integer bound (e.g. between –1 and 1) to the respective heart rate values. By doing so, more variability was added to the cycle of the data, making it more realistic.

3.7 Evaluation Metrics

The models take in a number of intervals' worth of data and try to predict the next consecutive set of intervals while training, validating, and testing. These predictions were compared to the actual next consecutive set of intervals in the data. We then evaluated the models' performance based on the difference between the predictions and the actual values. This was done by using the Mean Squared Error (MSE) metric.

To measure how realistic the augmented data was, we took a bootstrapped person's data from the full dataset. By applying mean absolute error (MAE), mean squared error (MSE), or coefficient of determination (R^2), we statistically measured the amount of variance between the real data and the generated data from the individual's first set of intervals. This measurement tells us how reliable the model's prediction is, which reflects its realism. We can tell if the data is reliable if the evaluation metrics are under a certain upper bound (e.g. if the MAE <= 10).

4. Results

4.1 Visualizations

As mentioned in the Methodology, we began the project by visualizing both heart rate and step data. The first visualizations we created were histograms of 1, 10, and 20 peoples' data collected per hour of the day, as shown below.

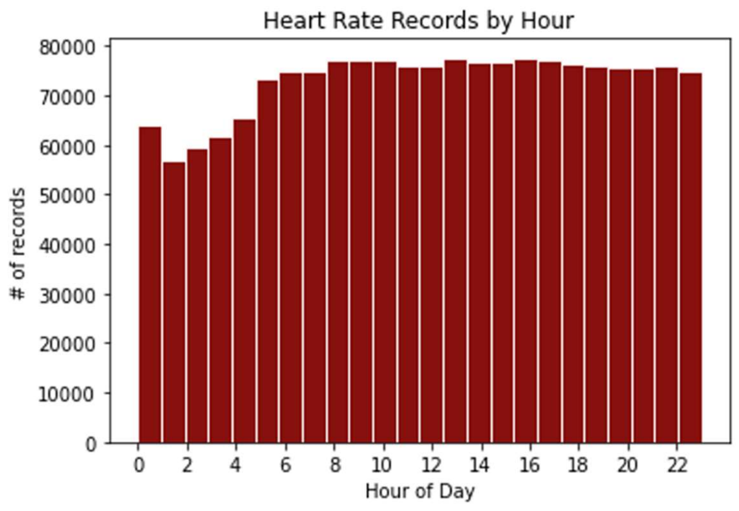


Figure 11: Heart Rate Record by Hour for 1 Person

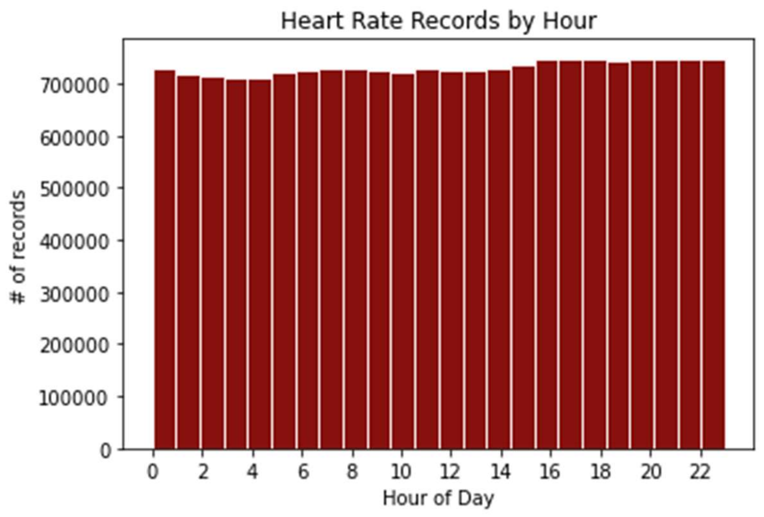


Figure 22: Heart Rate Record by Hour for 10 People

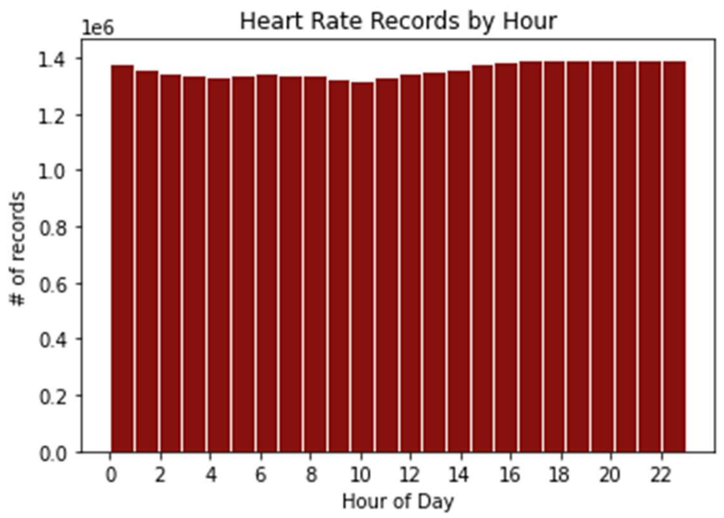


Figure 33: Heart Rate Record by Hour for 20 People

The next visualizations we created were histograms of the head (5 smallest datasets) and the tail (5 largest datasets). Each figure contains 5 subplots, each representing one dataset from the head/tail respectively.

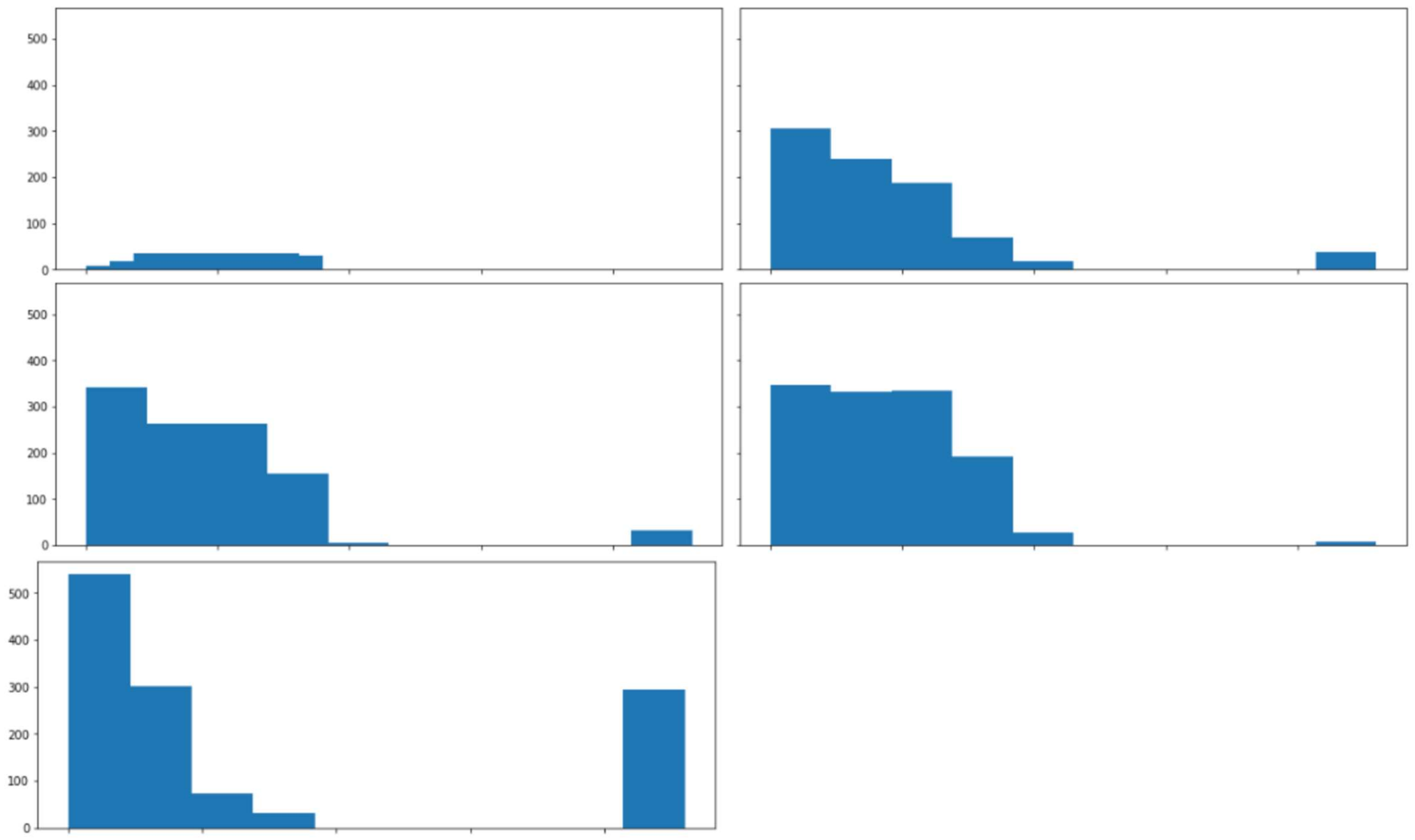


Figure 44: 5 Smallest Heart Rate Datasets

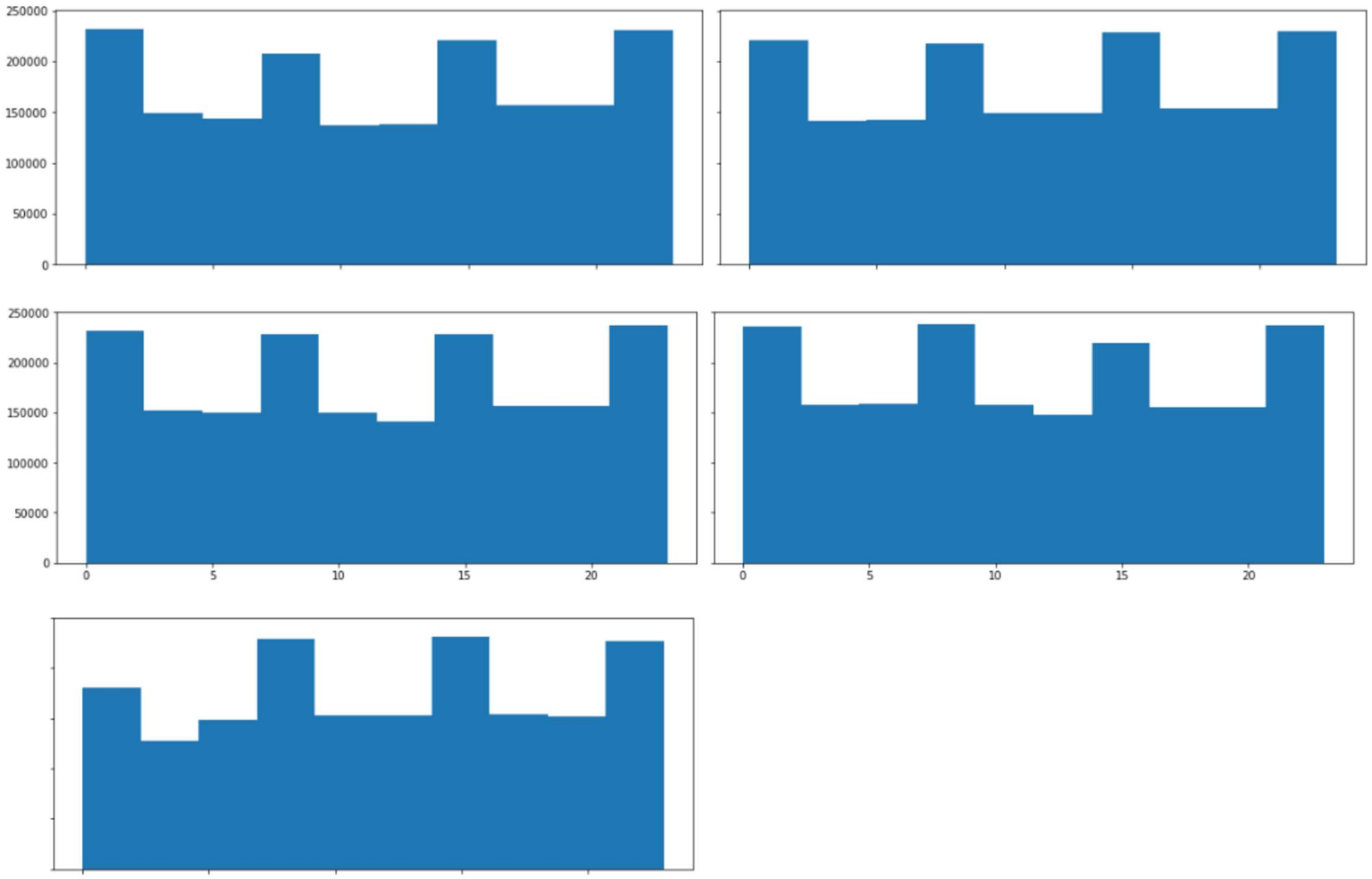


Figure 55: 5 Largest Heart Rate Datasets

The last preliminary visualization we created was an "Average Heart Rate per Hour for a Day" line graph to understand the heart rate trends for a single random user over a day's time (red line). This allows for comparison between the day's continuous heart rate values and the overall average heart rate value (blue line).

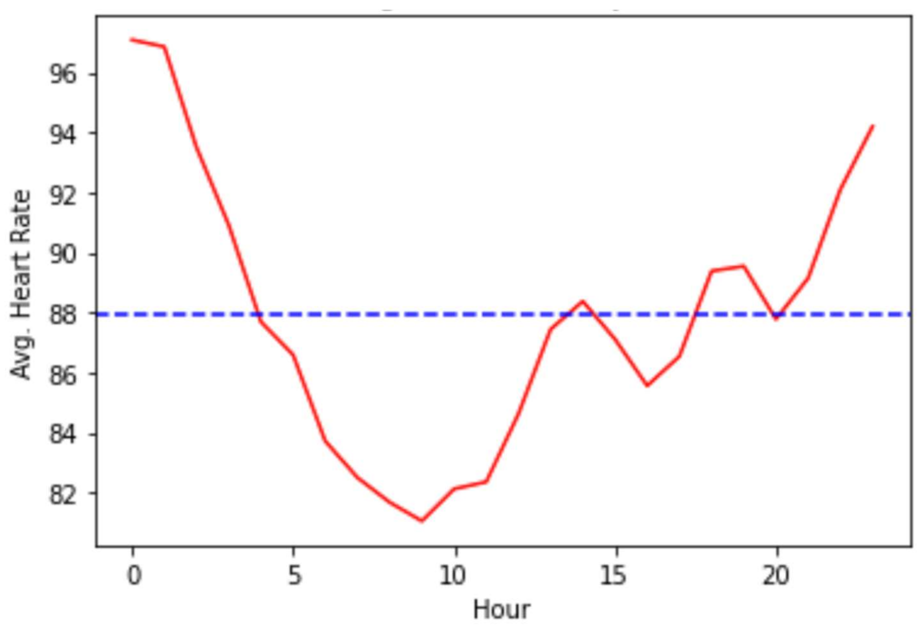
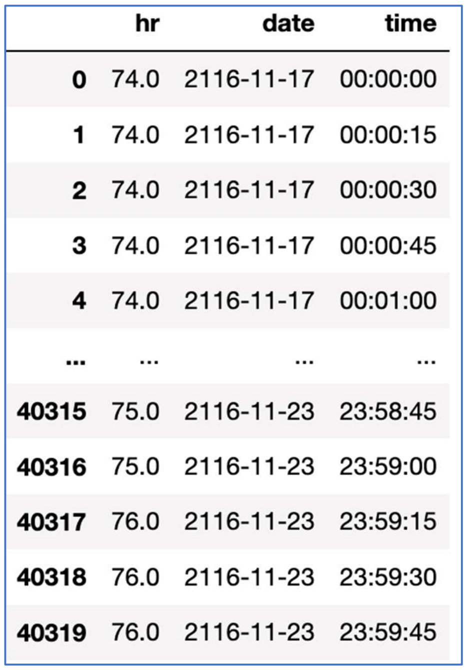


Figure 66: Average Heart Rate per Hour for a Day

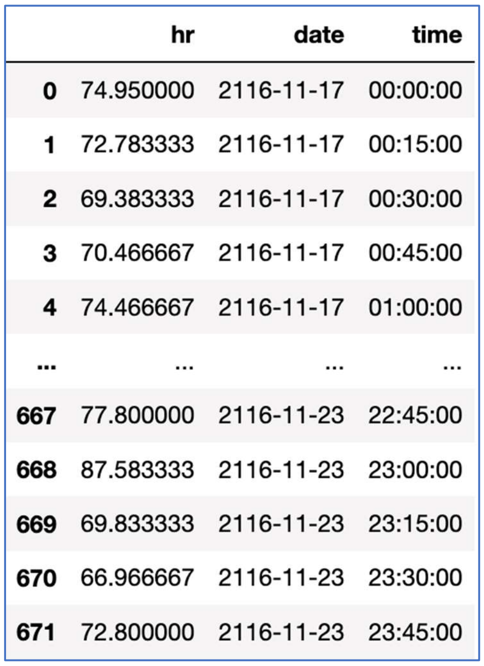
4.2 Preprocessing

The primary change made to the data when preprocessing was modifying the time intervals of the heart rate data from 15 seconds to 15 minutes. Below shows a side-by-side comparison of the first 5 and last 5 entries of one individual's week worth of data. The left (Figure 17a) is with 15-second intervals (40,320 data points), and the right (Figure 17b) is 15-minute intervals (672 data points).



	hr	date	time
0	74.0	2116-11-17	00:00:00
1	74.0	2116-11-17	00:00:15
2	74.0	2116-11-17	00:00:30
3	74.0	2116-11-17	00:00:45
4	74.0	2116-11-17	00:01:00
...	...	...	...
40315	75.0	2116-11-23	23:58:45
40316	75.0	2116-11-23	23:59:00
40317	76.0	2116-11-23	23:59:15
40318	76.0	2116-11-23	23:59:30
40319	76.0	2116-11-23	23:59:45

Figure 87a: 15-Second Intervals



	hr	date	time
0	74.950000	2116-11-17	00:00:00
1	72.783333	2116-11-17	00:15:00
2	69.383333	2116-11-17	00:30:00
3	70.466667	2116-11-17	00:45:00
4	74.466667	2116-11-17	01:00:00
...	...	...	...
667	77.800000	2116-11-23	22:45:00
668	87.583333	2116-11-23	23:00:00
669	69.833333	2116-11-23	23:15:00
670	66.966667	2116-11-23	23:30:00
671	72.800000	2116-11-23	23:45:00

Figure 77b: 15-Minute Intervals

4.3 Hyperparameter Configuration

After performing a grid search, we decided to choose the following hyperparameter values for all of the models:

- Learning Rate: 0.005005
- Batch Size: 15
- Number of Epochs: 5
- Hidden Layer Size: 1000
- Number of Multi-Head Attention Layers for Transformers: 1
- Number of Multi-Head Attention Layers for LSTM/GRU with Attention: 4
- Number of Sub-Encoder Layers: 12
- Number of LSTM/GRU Layers: 1
- Embed Dimension for Multi-Head Attention Layers: 64

4.4 Model Performances

To compare the efficiency of different models, we calculated the training time and test loss for each model.

Model	**Training Time (Avg. 1 Epoch)**	**Model MSE Test Loss**
LSTM	0.81 seconds	1.0046
GRU	0.65 seconds	1.2039
Transformer	1.78 seconds	0.9951
LSTM + Attention	0.91 seconds	1.0762
GRU + Attention	0.76 seconds	1.1771

Figure 18: Model Comparison between Training Time and MSE Test Loss for Heart Rate in 15-Minute Intervals

After we trained the models, we generated five periods of data using a bootstrapped person's first period of the day. For example, with a period of one day, we created five days of generated data. After data generation, we applied Gaussian noise to the generated data, including the bootstrapped period to negate possible seasonality. We compared the generated data to the person's actual data for the same periods with MAE, MSE, and R^2.

Model	**MAE**	**MSE**	**R^2**
LSTM	6.52	104.24	-0.6
GRU	7.96	125.5	-0.31
Transformer	6.52	103.69	-0.67
LSTM + Attention	6.56	102.94	-0.5
GRU + Attention	16.69	442.88	-0.19

Figure 19: Model Comparison between Validation Metrics for Heart Rate in 15-Minute Intervals

Below shows a model comparison based exclusively on MAE loss. We decided to exclusively use MAE loss for the remainder of the model comparisons due to MAE's robustness to outliers.

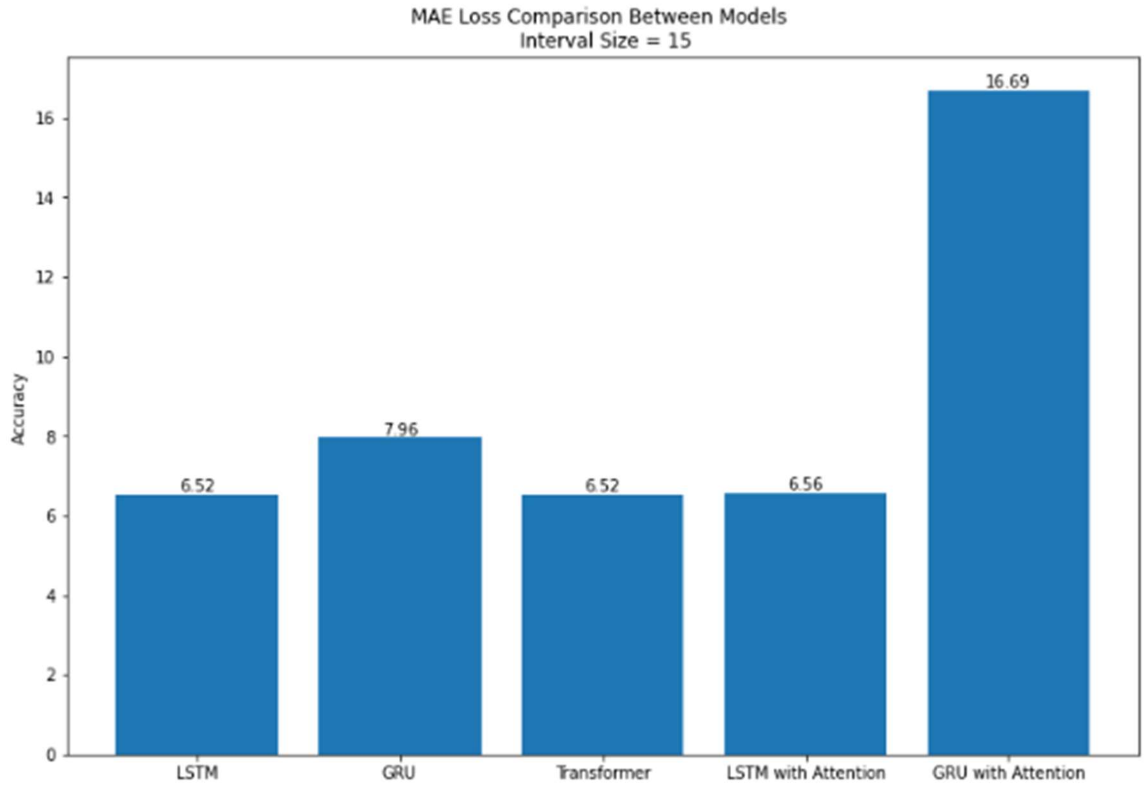


Figure 20: MAE Loss Comparison Between Models

After visualizing the MAE loss for the different models, we decided to adjust the interval sizes of the dataset to evaluate the robustness of the models. For each of the following visualizations, we predicted the next five sets of intervals. Each person had three weeks' worth of data, with a maximum of 10% padding. There are 51 people's data used to train each model, and the white noise is between the values –20 and 20.

We picked 12-, 24-, 42-, and 63-hour long sets of intervals because three weeks (504 hours) is evenly divisible by these four values, making it possible to evenly reshape the sets into separate rows. Appendices A, B, and C show MAE validation at different interval sizes and different random seeds (1, 2 and 3 respectively) without white noise. Appendices D, E, and F show MAE validation at different interval sizes and different random seeds (also 1, 2 and 3 respectively) with white noise between the values –20 and 20.

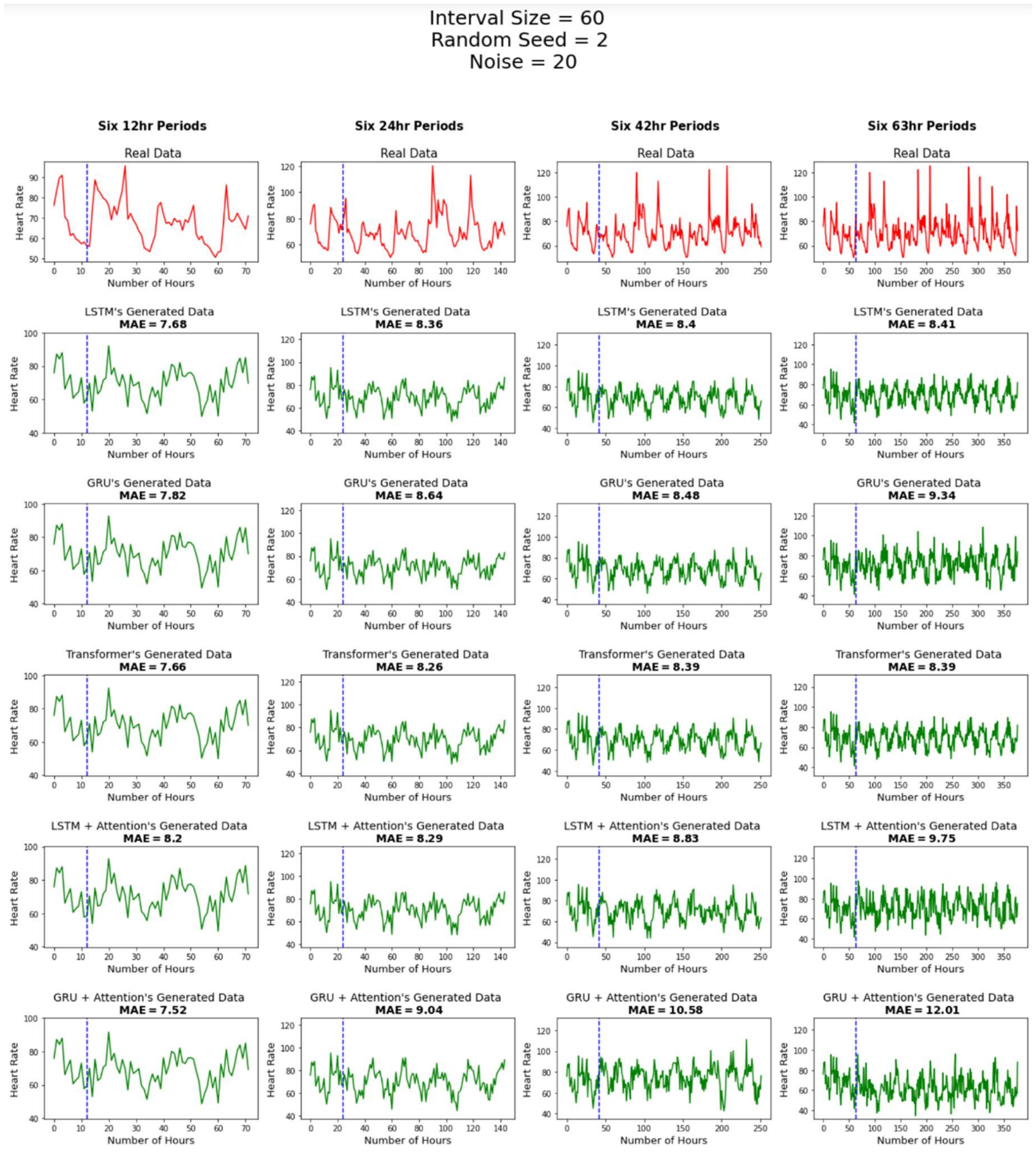
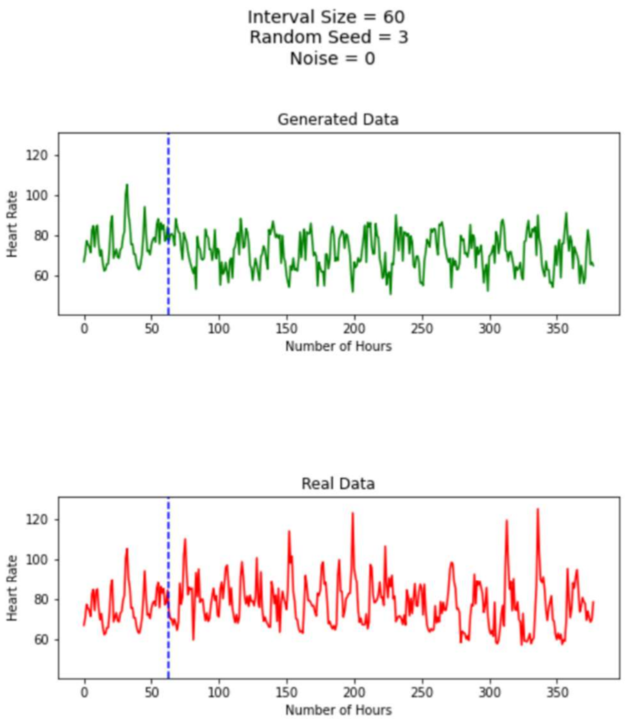
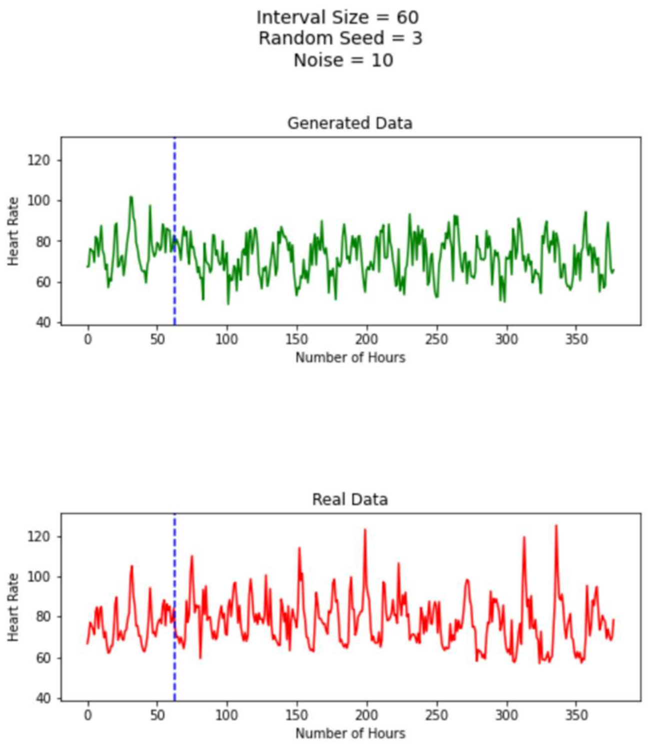
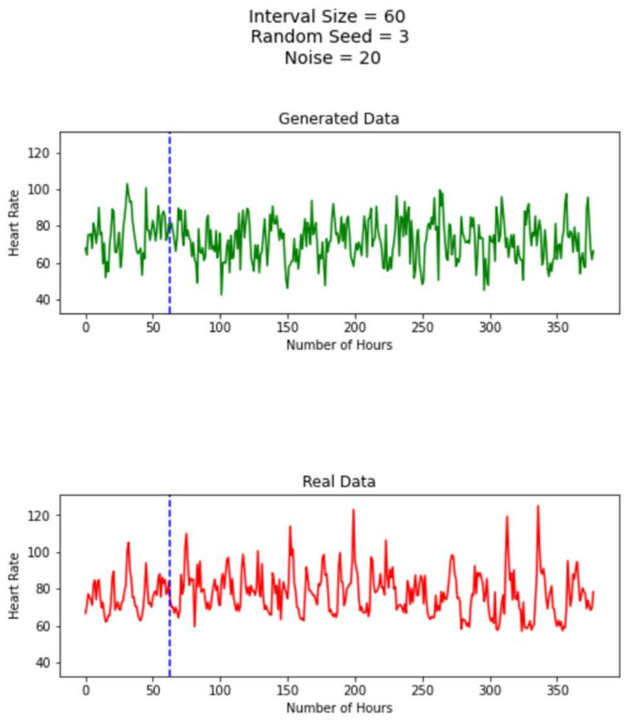


Figure 21: MAE Validation at Interval Size = 60

Next, we compared different Gaussian noises that we applied to the generated data. The blue dotted line in the graphs shows the point at which the bootstrapped real data is used to create the generated data.







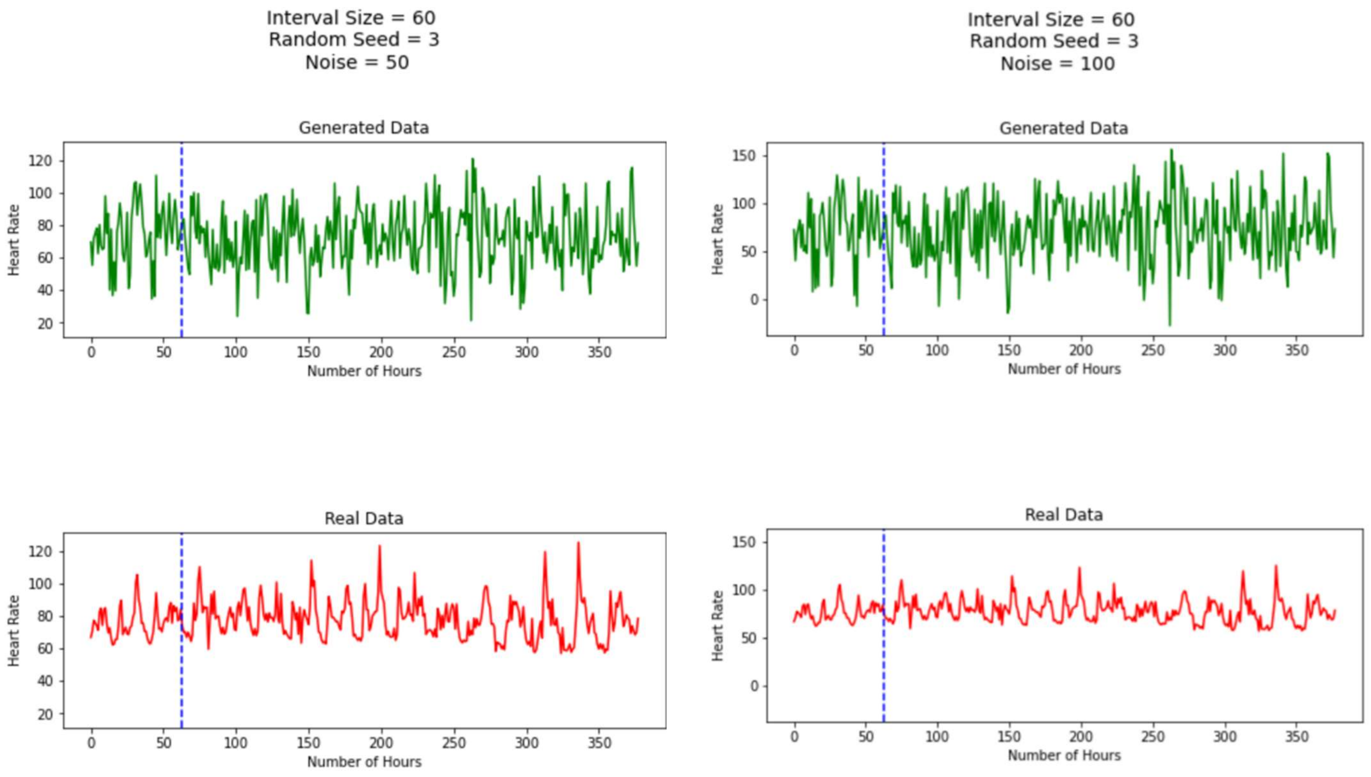


Figure 22: LSTM Models with varying noise levels

The next visualizations we created were used for evaluating the model's performance in generating a certain number of days. We evaluated the model's ability to generate 5-, 10-, 15-, and 20-days' worth of data in advance to validate its accuracy.

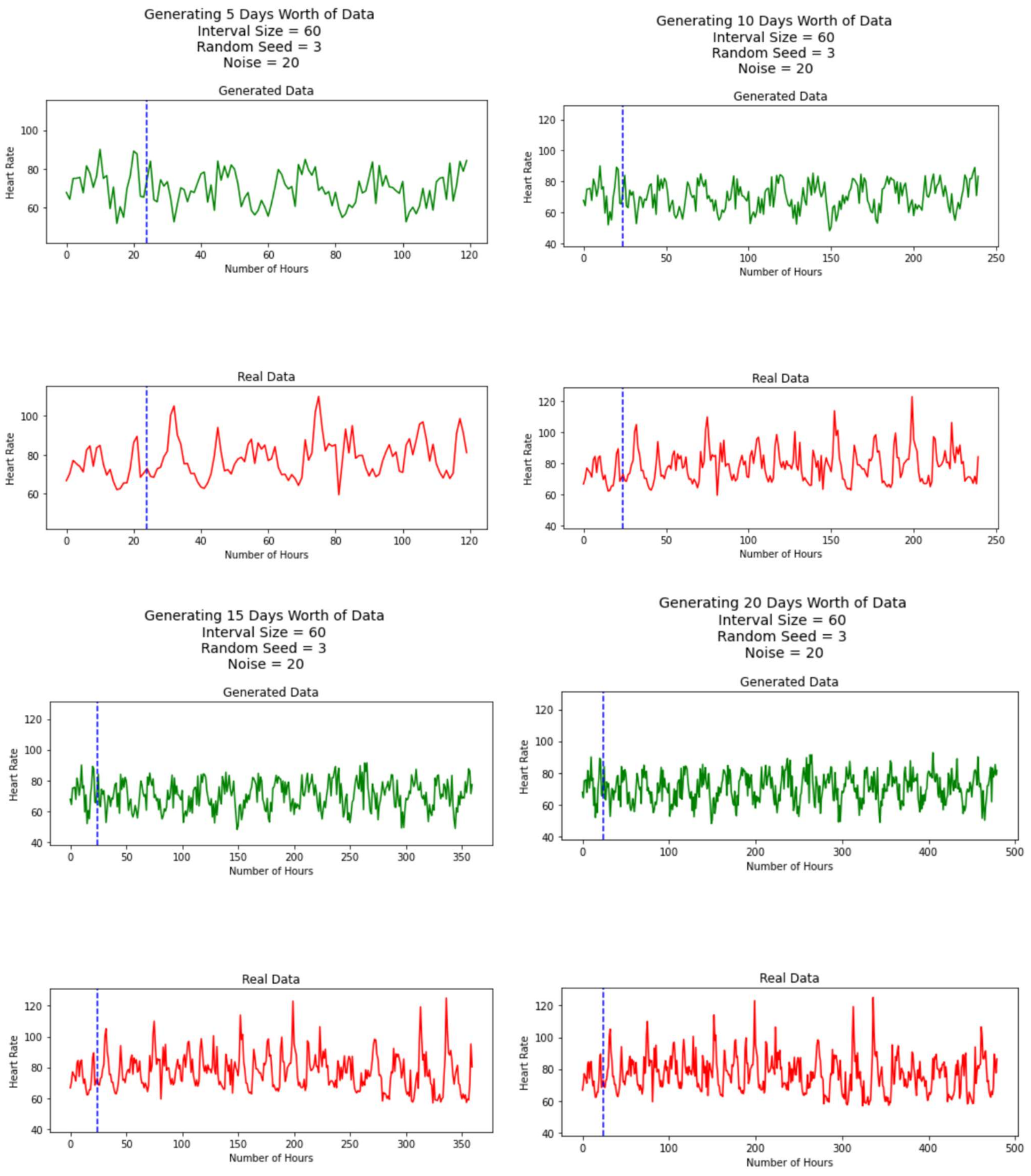


Figure 23: LSTM model predicting varying amount of days

Next, we evaluated the robustness of the model against the amount of padding we applied. Datasets with lower percentages of padding allowed for more samples to be utilized for training.

# Of Samples	**Padding %**	**MAE loss**
11	*5%*	10.2995
51	*10%*	9.6680
81	*15%*	11.7775

Figure 24: MAE loss for each Padding % for Heart Rate in 15-Minute Intervals

5. Discussion

5.1 Analysis

From a thorough analysis of our results, our original prediction that the Transformer model would have the best performance was rejected. Transformer models have a high number of parameters and a high complexity, meaning they perform best with a vast amount of data. While this caused the Transformer model to have a viable test loss, the high parameter count caused the model to have the slowest training time.

The goal of this project was to augment realistic physiological data, meaning we do not want to outright predict the real data values present. Because of this, a MSE test loss too close to zero may mean the given model is generating redundant data, which indicates poor performance. However, having a MSE test loss in a generally-low range is beneficial in evaluating the model's performance. This logic also applies to the MAE loss when comparing the generated data to the respective real data.

To compare the generated data to the real data, we applied the MAE, MSE, and R^2 evaluation metrics. Of these three metrics, we found MAE to be the most representative of the models' performances. The MAE metric showed significantly lower loss and a more interpretable range than the MSE metric, while the R^2 metric is difficult to interpret for data containing seasonality and trends. Thus, we decided to use MAE for all other evaluations.

The first comparisons made were between MAE loss values and the generated data at different interval sizes for each model. In order to effectively analyze the graphs at different interval sizes, we utilized the following criteria:

1. Similar, non-constant seasonality to data
2. Similar density to real data
3. Similar range to real data

The "inconsistent seasonality" criterion refers to the ability of the generated data to not repeat itself, and to have different trends in the data over a period of time. The "similar density to real data" criterion refers to the heart rate records being close in proximity to one another as they are in the real data. Lastly, the "similar range to real data" criterion refers to the generated data not having peaks of high and low amplitudes compared to the real data. The following graphs compare real data with generated data that meet a certain criterion and generated data that does not meet the specific criterion, respectively.

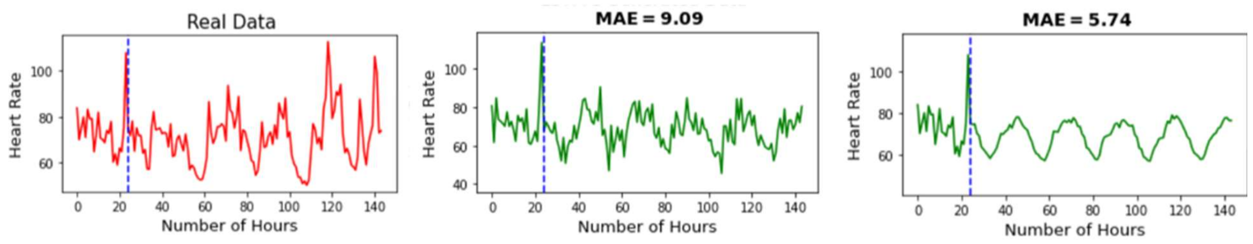


Figure 25: Real vs Generated Data with Seasonality Comparison

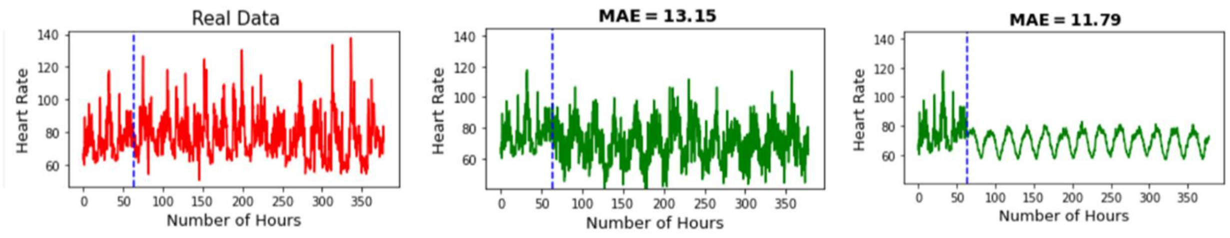


Figure 26: Real vs Generated Data with Range Comparison

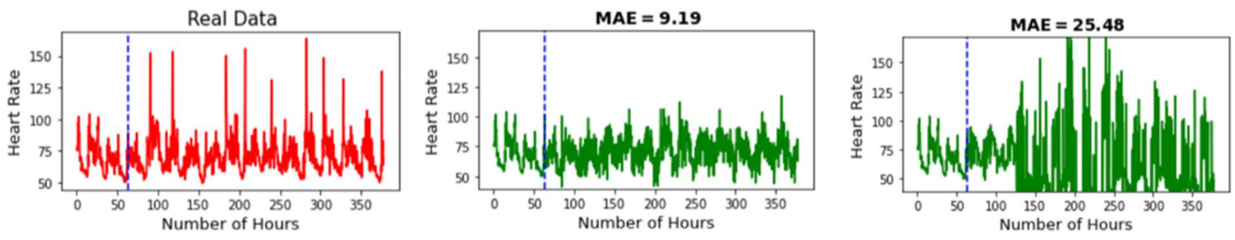


Figure 27: Real vs Generated Data with Density Comparison

Using this analysis strategy, we came across several key findings. First, all five models generated realistic data at different interval sizes, with some performing better than others. The GRU and GRU with Attention models tended to have similar seasonality, range, and density compared to the real data. Meanwhile, the Transformer model would typically show repetitive patterns in its generated data, indicating poor performance and realism in terms of seasonality and range. Next, we discovered that the generated data typically met our criteria the best at the largest interval size, 60. When analyzing the graphs with an interval size of 10 or 15, the RNN with Attention models performed poorly, regardless of the amount of noise applied. The visualizations showed unreasonable ranges and densities (see all Appendices with interval sizes 10 and 15 for model comparison).

When different levels of Gaussian noise were applied, there was a clear trend in the data where the MAE loss grew higher as the noise increased. Noise is an important piece in representing the realistic seasonality of the data. However, the randomized nature of noise can increase the model's variability, thus increasing the MAE loss.

The comparison of MAE loss values at different padding percentages for the generated data showed that MAE loss tends to vary when different levels of padding are applied. Padding is essential to filling time gaps, but filling in gaps of time with the local mean can result in greater amounts of repetitive data. In addition, padding smaller amounts of data provides a smaller number of samples, as there are less individuals that are only missing smaller amounts of data.

Ultimately, all of these findings are relative to the original problem at hand. Each of the models created were capable of generating realistic heart rate data. Although some of the models performed better and faster than others, each of them was able to achieve the basic goal of data augmentation – using data to generate artificial data. The ability to augment data can open up the ongoing early illness detection research; the more data available to researchers, the better illness detection can get.

5.2 Limitations

One of the limitations faced during the project was the amount of data we were able to run through the models. When working with 15-second intervals for heart rate data, we had approximately 5,760 data points for a single day. With this much data, the LLSC's GPUs ran out of allocated memory in Jupyter Notebook. Thus, we decreased the amount of data substantially by converting the 15-second intervals into 15-minute intervals. This hindered the performance of the models, such as the Transformer model, as they perform better with more data.

When examining the various features of the physiological data provided to us, we noticed the time intervals varied between each feature. For example, heart rate was recorded every 15 seconds whereas step count was recorded every 15 minutes. There was also one feature, heart rate variability (SpO2), that had multiple time intervals of different values, each being 1 minute, 2 minutes, or 5 minutes. Along with these differences in time, there was also a difference in units of measure between features. For example, respiratory rate was recorded in breaths per minute whereas body battery was recorded in percentage. These differences between features made it difficult to run multiple features through the models at once and difficult to combine and compare features.

5.3 Future Work

Although the models perform well, there are some improvements we believe could lead to useful insights, efficient data generation, and enhanced prediction ability. One way to improve the models is by incorporating various time intervals, both shorter and longer (eg. minutes, hours, days), to train the model. By doing so, the models can predict specific times of day or days in the week, enabling it to capture routine-related patterns (eg. day time vs night time, weekends vs weekdays). Additionally, we could separate the data into larger periods, such as one month, six months, or one year, which would provide the models with a more comprehensive understanding of each individual. This would allow the models to capture how a person's physiological features vary across different seasons, holidays, and other special occasions. Another way to improve the models is experimenting with different features. This would be beneficial to observe how the models' prediction accuracy changes and compare the results across different features. Feature engineering techniques can also be applied to correlated features in the data, such as heart rate and step count, to enhance the model's predictive capabilities for a given individual.

6. Conclusion

Although widespread illnesses such as COVID-19 occur rarely, there is a need to detect the spread of such diseases early to limit economic impacts. We developed machine learning models to augment data, which will provide researchers with more realistic, healthy physiological data for their work. We applied evaluation metrics on the generated data to compare its accuracy against the real data. We found that all five models were capable of generating realistic physiological health data with low training times and test losses. This project's result of generating realistic physiological data is a step in the right direction for early illness detection research to progress in the near future.

7. References

Abdelhamid, M. (2021, November 16). Fitness Tracker Information and Privacy Management: Empirical Study. Journal of Medical Internet Research, 23(11), e23059. https://doi.org/10.2196/23059

Amazon Web Services (2023). What is a Neural Network? AI and ML Guide - AWS. Amazon Web Services, Inc. https://aws.amazon.com/what-is/neural-network/

Benhar, H., Idri, A., & Fernández-Alemán, J. L. (2018, May 17). Data Preprocessing for Decision Making in Medical Informatics: Potential and Analysis. Advances in Intelligent Systems and Computing, 1208–1218. https://doi.org/10.1007/978-3-319-77712-2_116

Conroy, B., Silva, I., Mehraei, G., Damiano, R., Gross, B., Salvati, E., Feng, T., Schneider, J., Olson, N., Rizzo, A. G., Curtin, C. M., Frassica, J., & McFarlane, D. C. (2022, March 8). Real-time infection prediction with wearable physiological monitoring and AI to aid military workforce readiness during COVID-19. Scientific Reports, 12(1), 3797. https://doi.org/10.1038/s41598-022-07764-6

Laskowski, E. (2020, October 2). What's a normal resting heart rate? Mayo Clinic. https://www.mayoclinic.org/healthy-lifestyle/fitness/expert-answers/heart-rate/faq-20057979#:~:text=A%20normal%20resting%20heart%20rate%20for%20adults%20ranges%20from%2060

Mumuni, A., Mumuni, F. (2022, November 15). *Data augmentation: A comprehensive survey of modern approaches*. Array, Volume 16. https://www.sciencedirect.com/science/article/pii/S2590005622000911?via%3Dihub

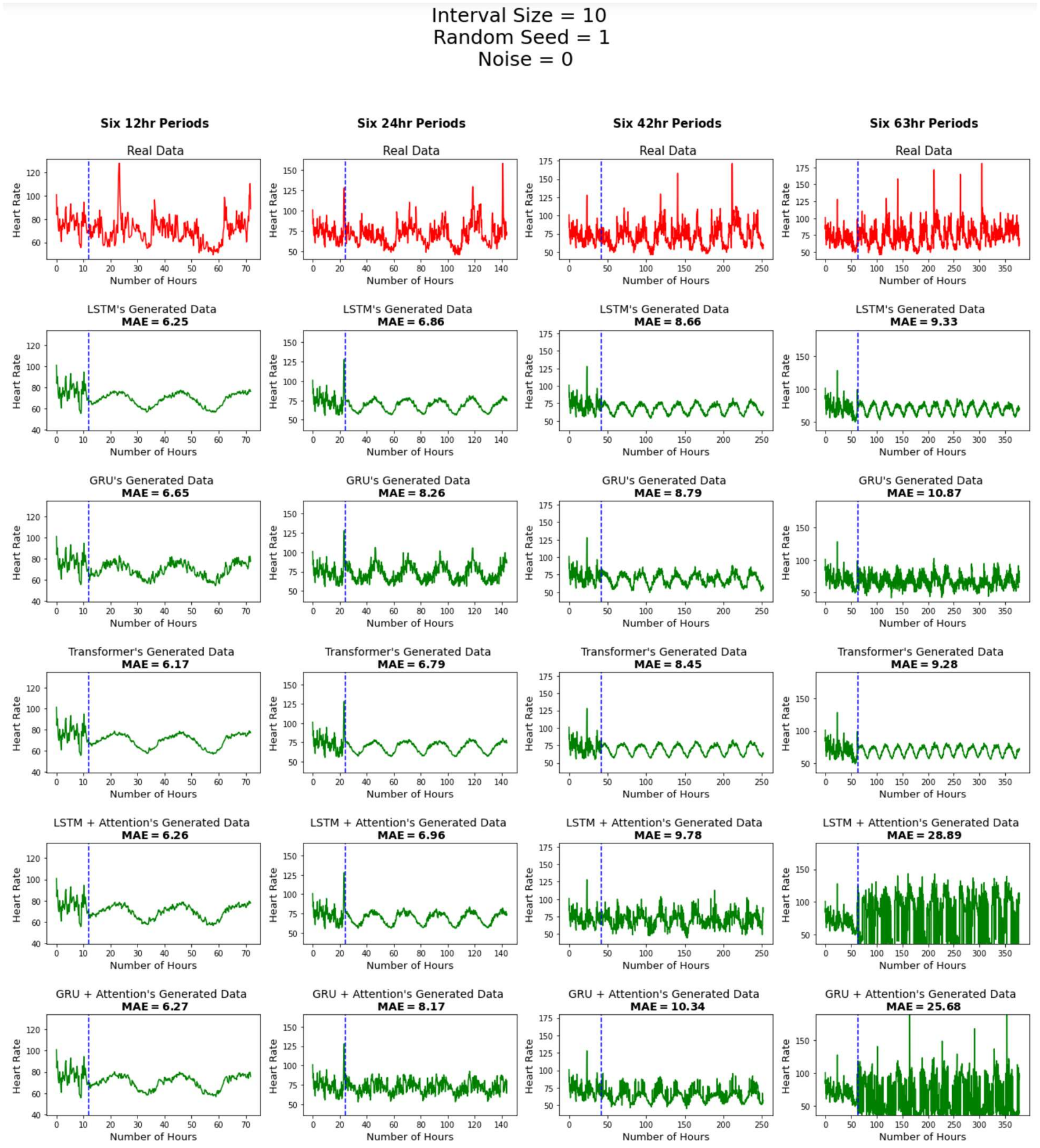
Olah, C. (2015, August 27). Understanding LSTM Networks. Github.io. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

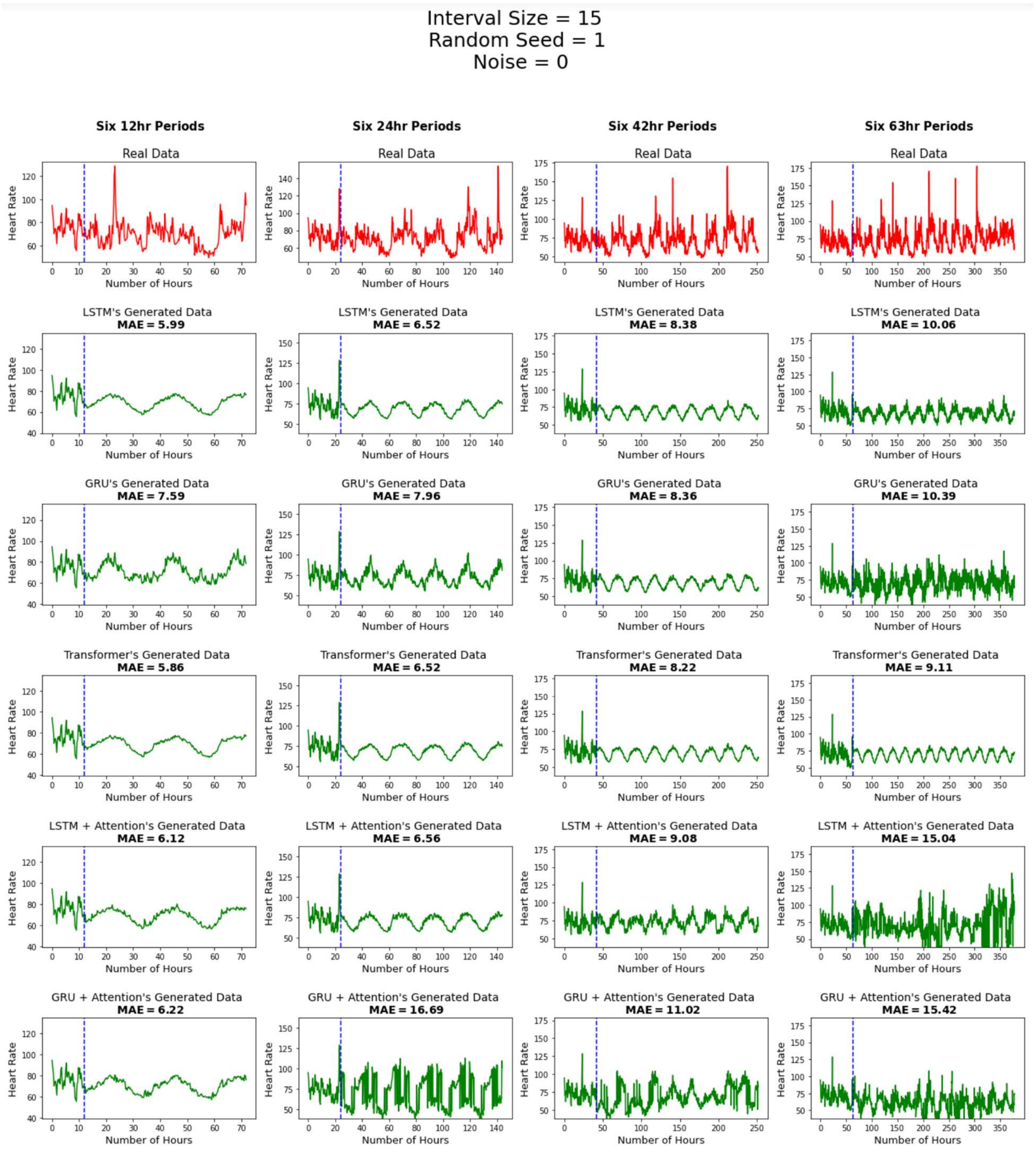
Steele, L., Orefuwa, E., Bino, S., Singer, S. R., Lutwama, J., & Dickmann, P. (2020, September 11). Earlier Outbreak Detection—A Generic Model and Novel Methodology to Guide Earlier Detection Supported by Data From Low- and Mid-Income Countries. Frontiers in Public Health, 8. https://doi.org/10.3389/fpubh.2020.00452

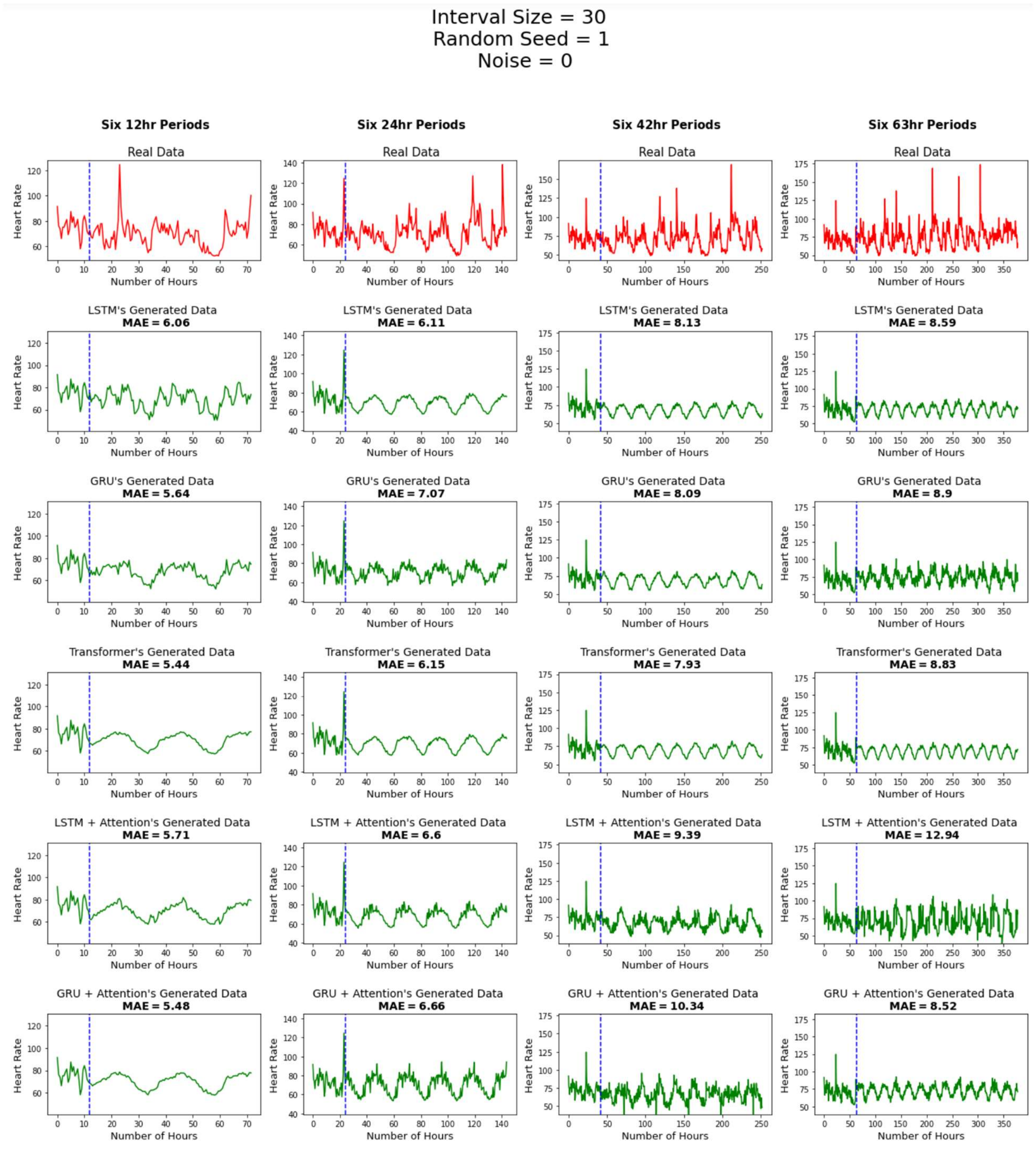
Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

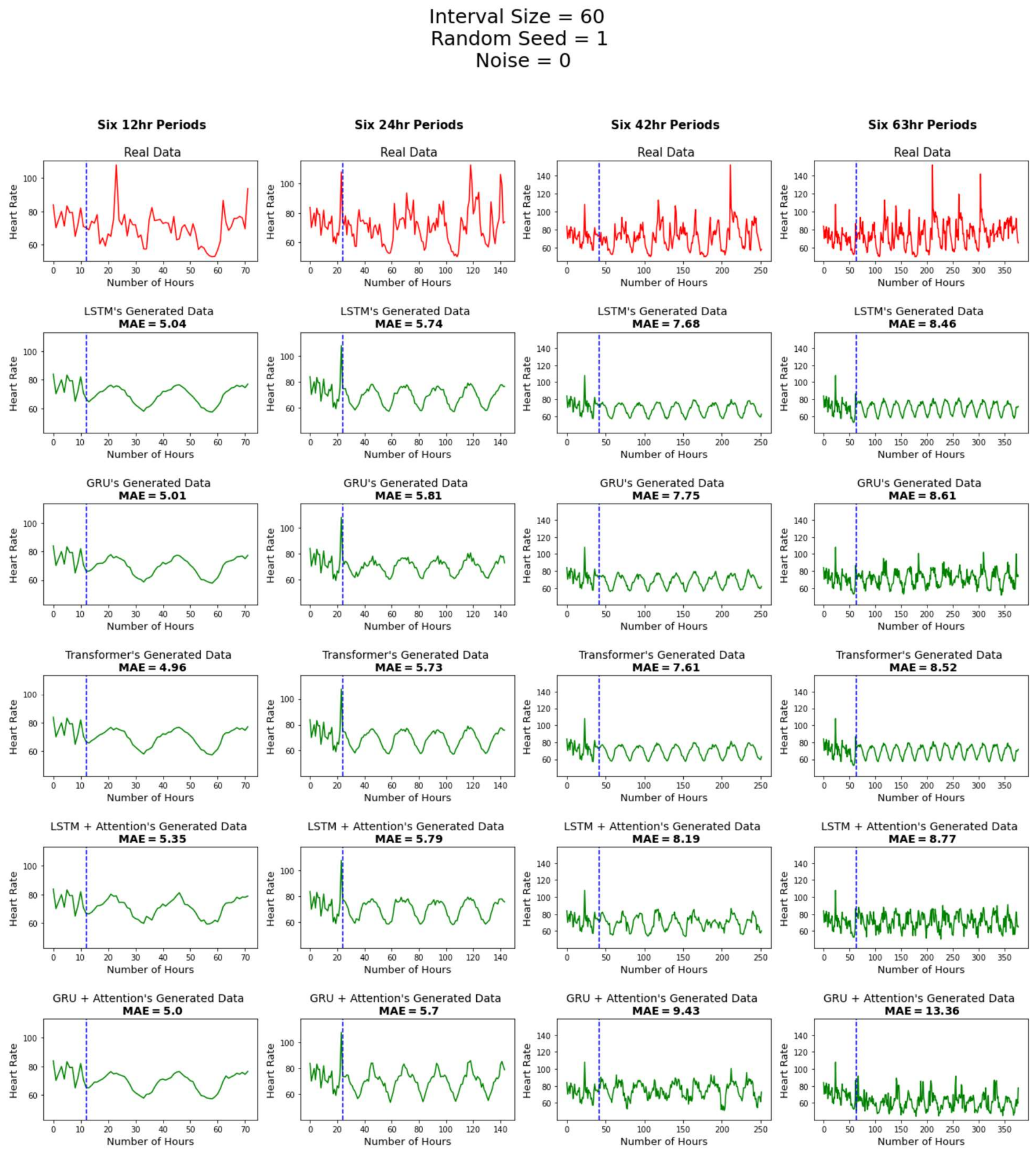
8. Appendices

8.1 Appendix A. MAE Validation at Different Interval Size (Random Seed = 1, Nose = 0)

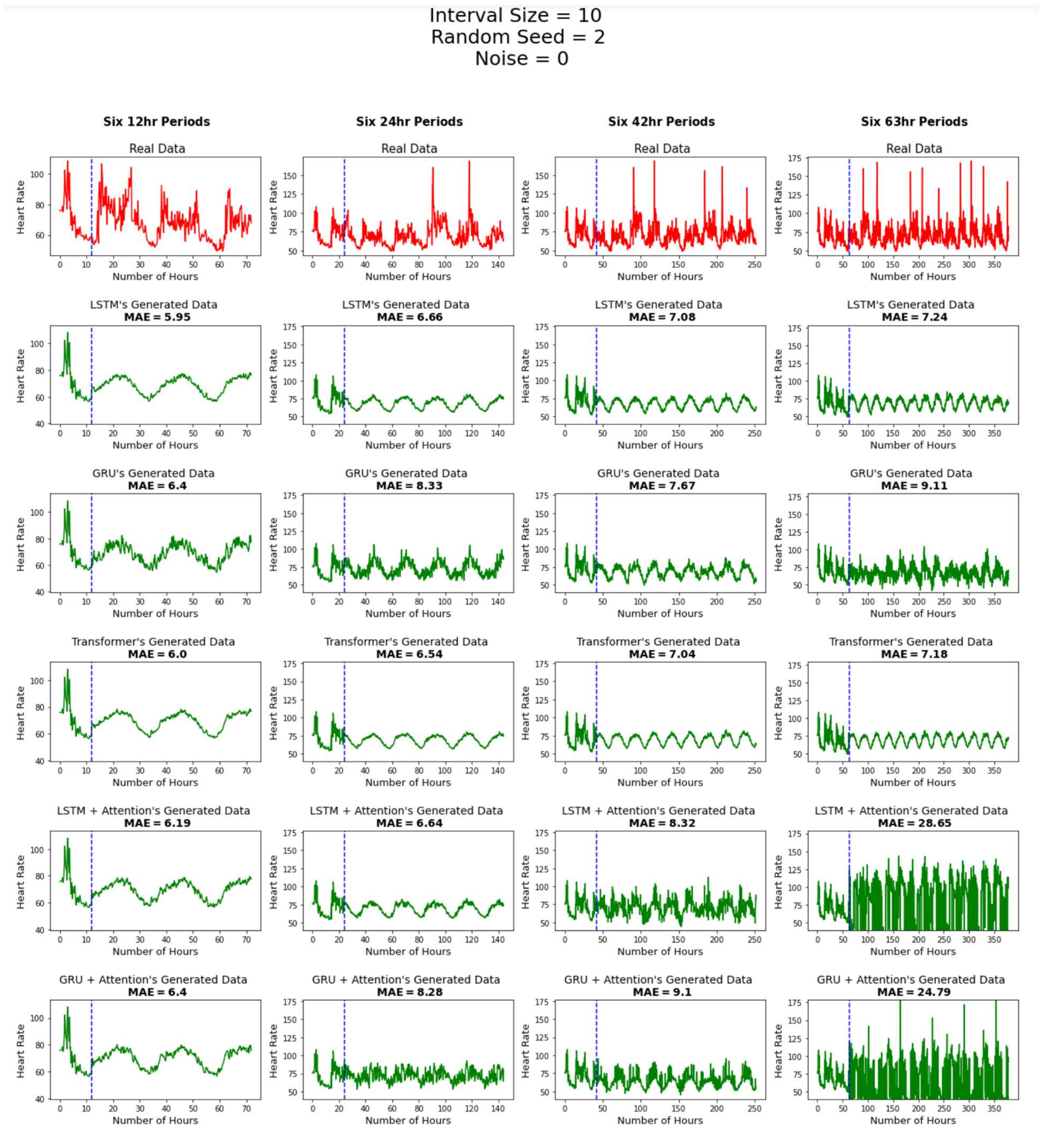


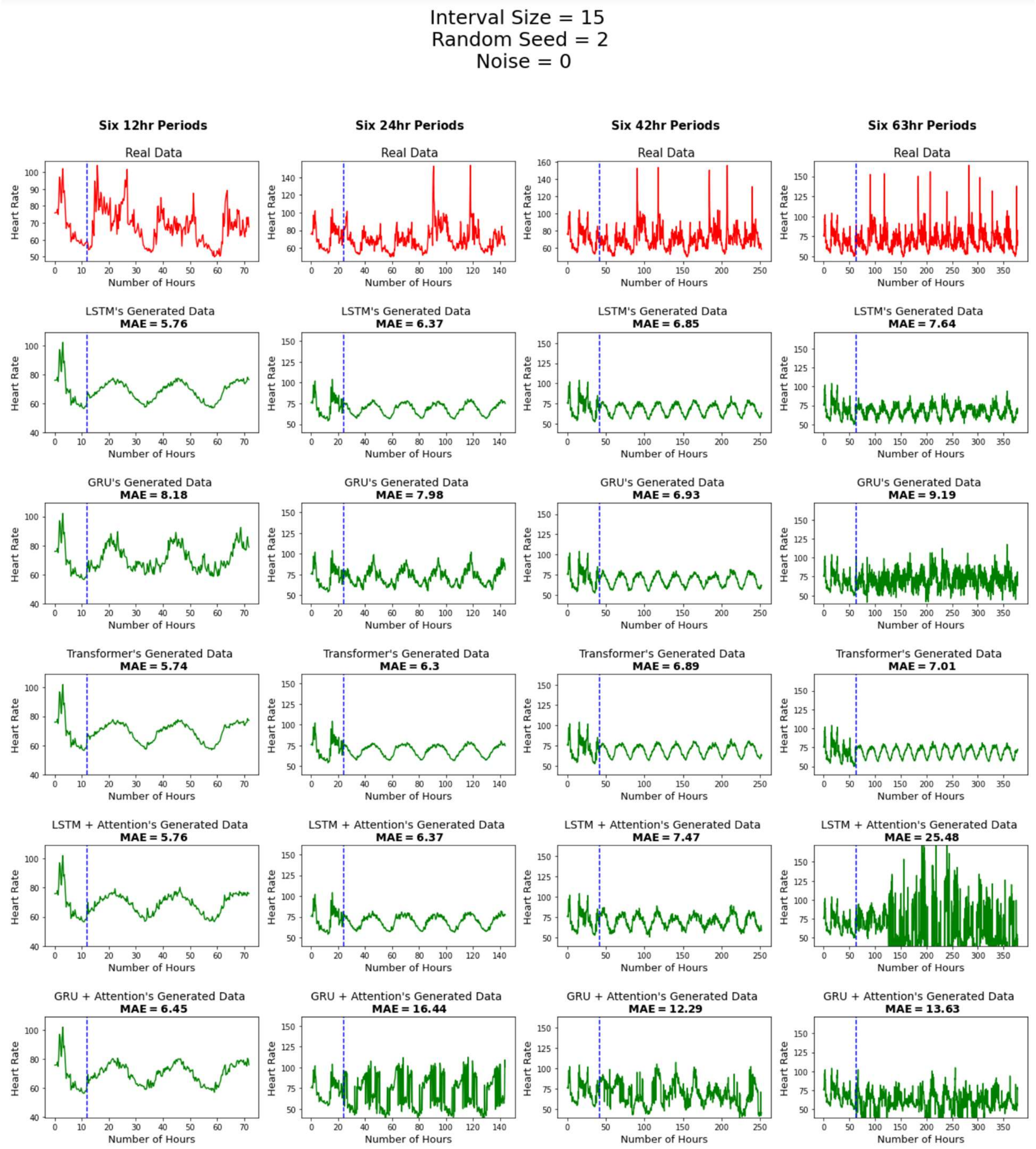

Interval Size = 15
Random Seed = 1
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 5.99
MAE = 6.52
MAE = 8.38
MAE = 10.06
GRU's Generated Data
MAE = 7.59
MAE = 7.96
MAE = 8.36
MAE = 10.39
Transformer's Generated Data
MAE = 5.86
MAE = 6.52
MAE = 8.22
MAE = 9.11
LSTM + Attention's Generated Data
MAE = 6.12
MAE = 6.56
MAE = 9.08
MAE = 15.04
GRU + Attention's Generated Data
MAE = 6.22
MAE = 16.69
MAE = 11.02
MAE = 15.42
Heart Rate
Number of Hours

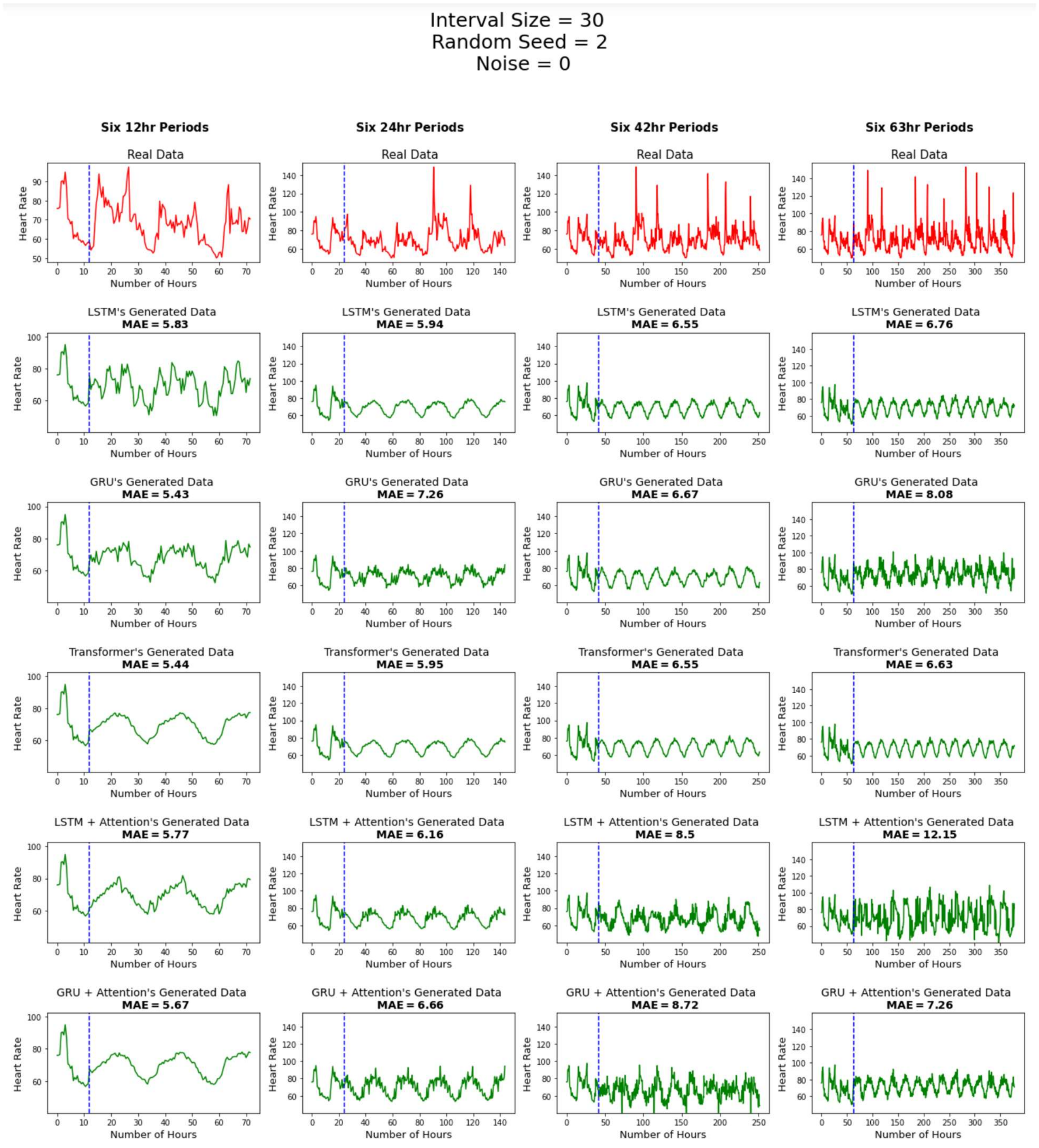

Interval Size = 30
Random Seed = 1
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 6.06
MAE = 6.11
MAE = 8.13
MAE = 8.59
GRU's Generated Data
MAE = 5.64
MAE = 7.07
MAE = 8.09
MAE = 8.9
Transformer's Generated Data
MAE = 5.44
MAE = 6.15
MAE = 7.93
MAE = 8.83
LSTM + Attention's Generated Data
MAE = 5.71
MAE = 6.6
MAE = 9.39
MAE = 12.94
GRU + Attention's Generated Data
MAE = 5.48
MAE = 6.66
MAE = 10.34
MAE = 8.52
Heart Rate
Number of Hours

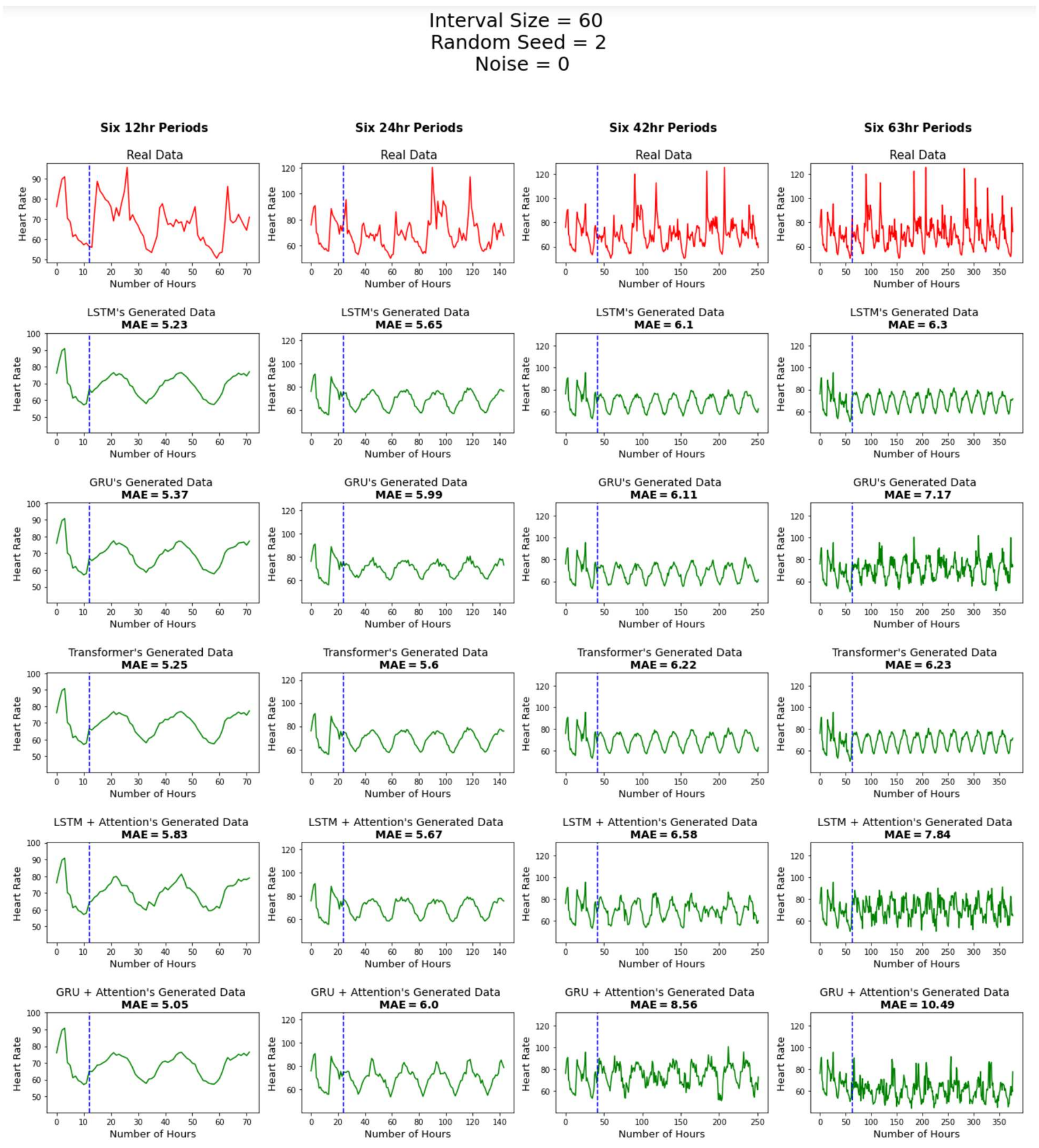

Interval Size = 60
Random Seed = 1
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 5.04
MAE = 5.74
MAE = 7.68
MAE = 8.46
GRU's Generated Data
MAE = 5.01
MAE = 5.81
MAE = 7.75
MAE = 8.61
Transformer's Generated Data
MAE = 4.96
MAE = 5.73
MAE = 7.61
MAE = 8.52
LSTM + Attention's Generated Data
MAE = 5.35
MAE = 5.79
MAE = 8.19
MAE = 8.77
GRU + Attention's Generated Data
MAE = 5.0
MAE = 5.7
MAE = 9.43
MAE = 13.36
Heart Rate
Number of Hours

8.2 Appendix B. MAE Validation at Different Interval Size (Random Seed = 2, Nose = 0)

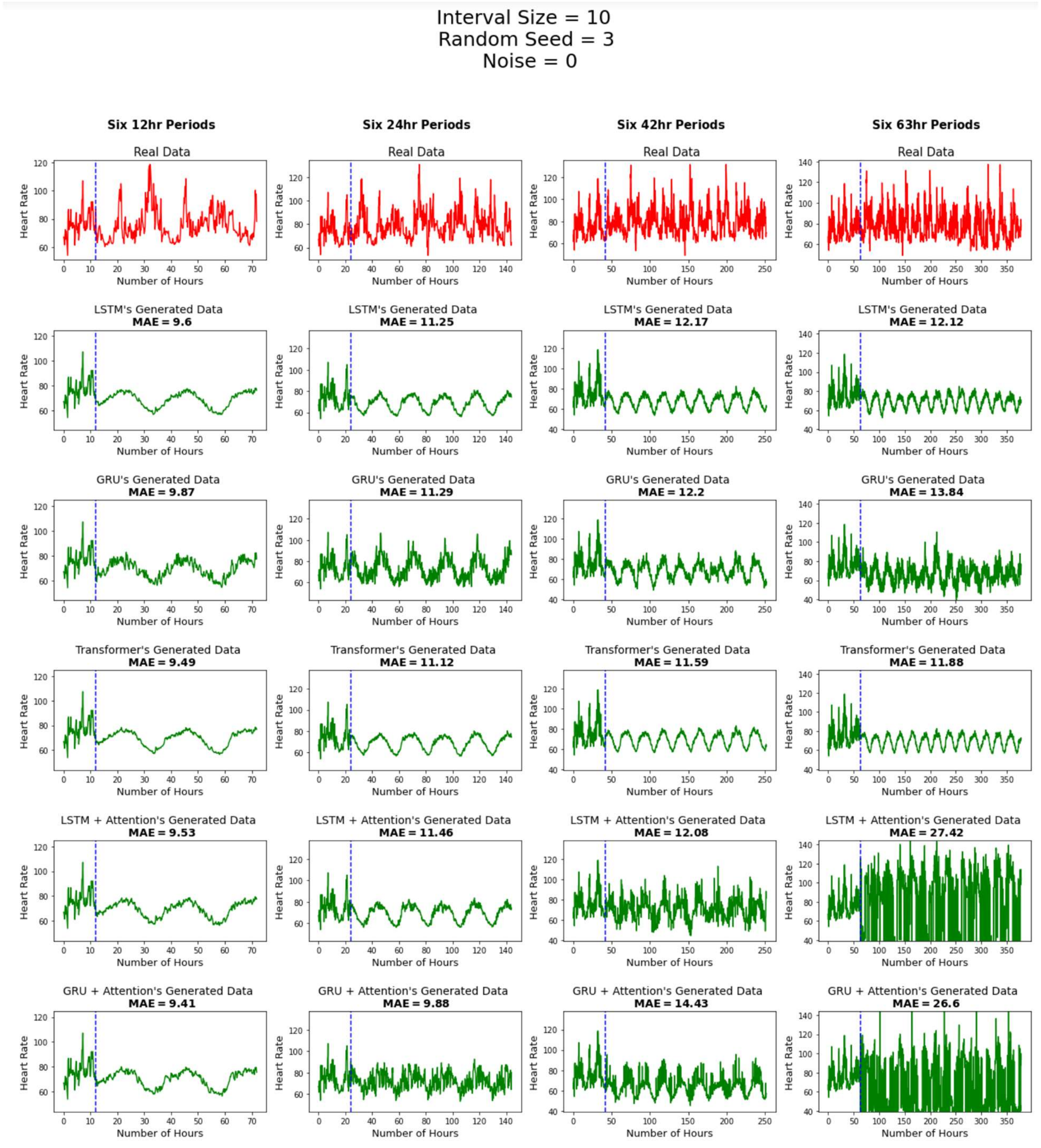


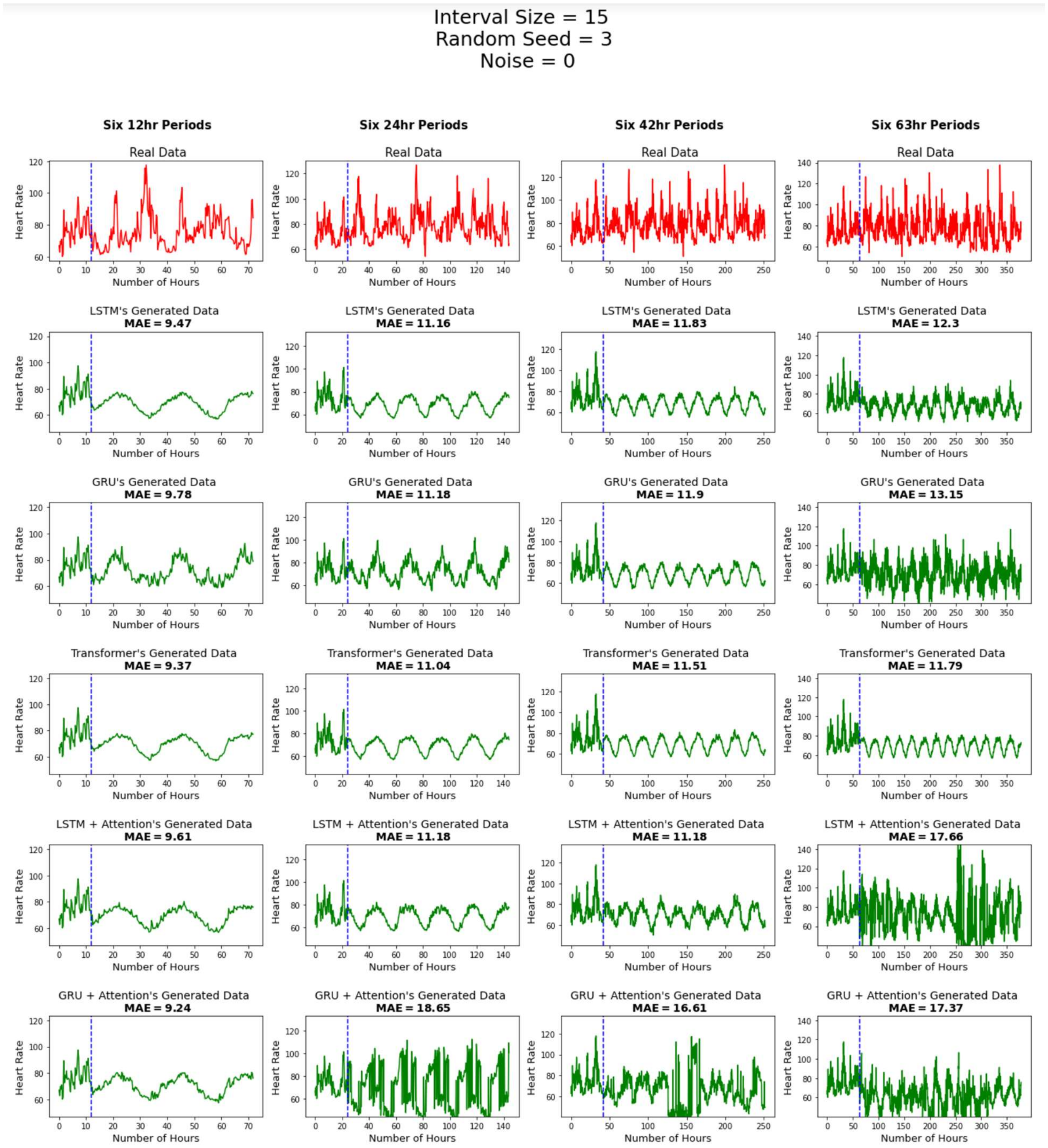

Interval Size = 15
Random Seed = 2
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 5.76
MAE = 6.37
MAE = 6.85
MAE = 7.64
GRU's Generated Data
MAE = 8.18
MAE = 7.98
MAE = 6.93
MAE = 9.19
Transformer's Generated Data
MAE = 5.74
MAE = 6.3
MAE = 6.89
MAE = 7.01
LSTM + Attention's Generated Data
MAE = 5.76
MAE = 6.37
MAE = 7.47
MAE = 25.48
GRU + Attention's Generated Data
MAE = 6.45
MAE = 16.44
MAE = 12.29
MAE = 13.63
Heart Rate
Number of Hours

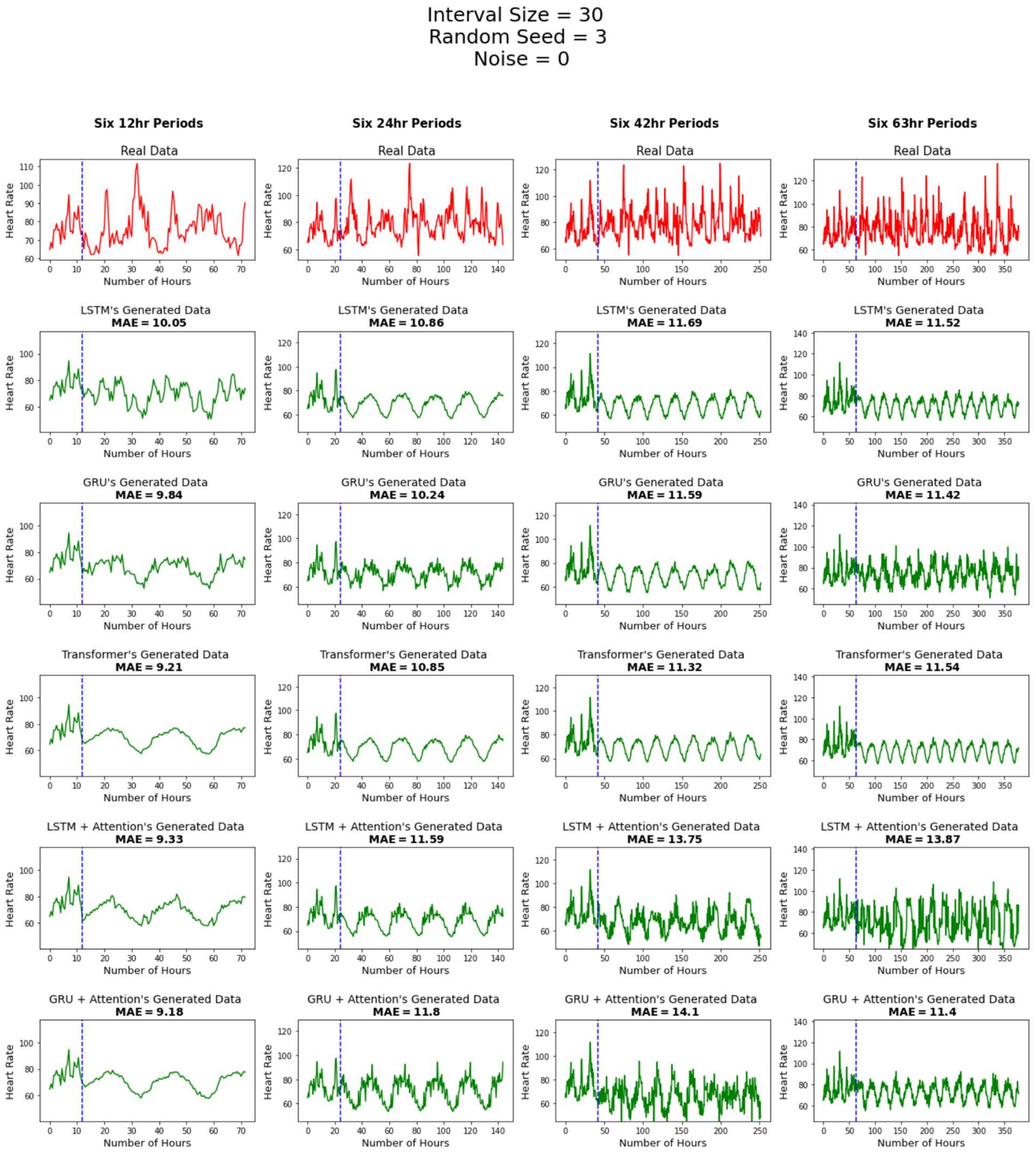

Interval Size = 30
Random Seed = 2
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 5.83
MAE = 5.94
MAE = 6.55
MAE = 6.76
GRU's Generated Data
MAE = 5.43
MAE = 7.26
MAE = 6.67
MAE = 8.08
Transformer's Generated Data
MAE = 5.44
MAE = 5.95
MAE = 6.55
MAE = 6.63
LSTM + Attention's Generated Data
MAE = 5.77
MAE = 6.16
MAE = 8.5
MAE = 12.15
GRU + Attention's Generated Data
MAE = 5.67
MAE = 6.66
MAE = 8.72
MAE = 7.26
Heart Rate
Number of Hours

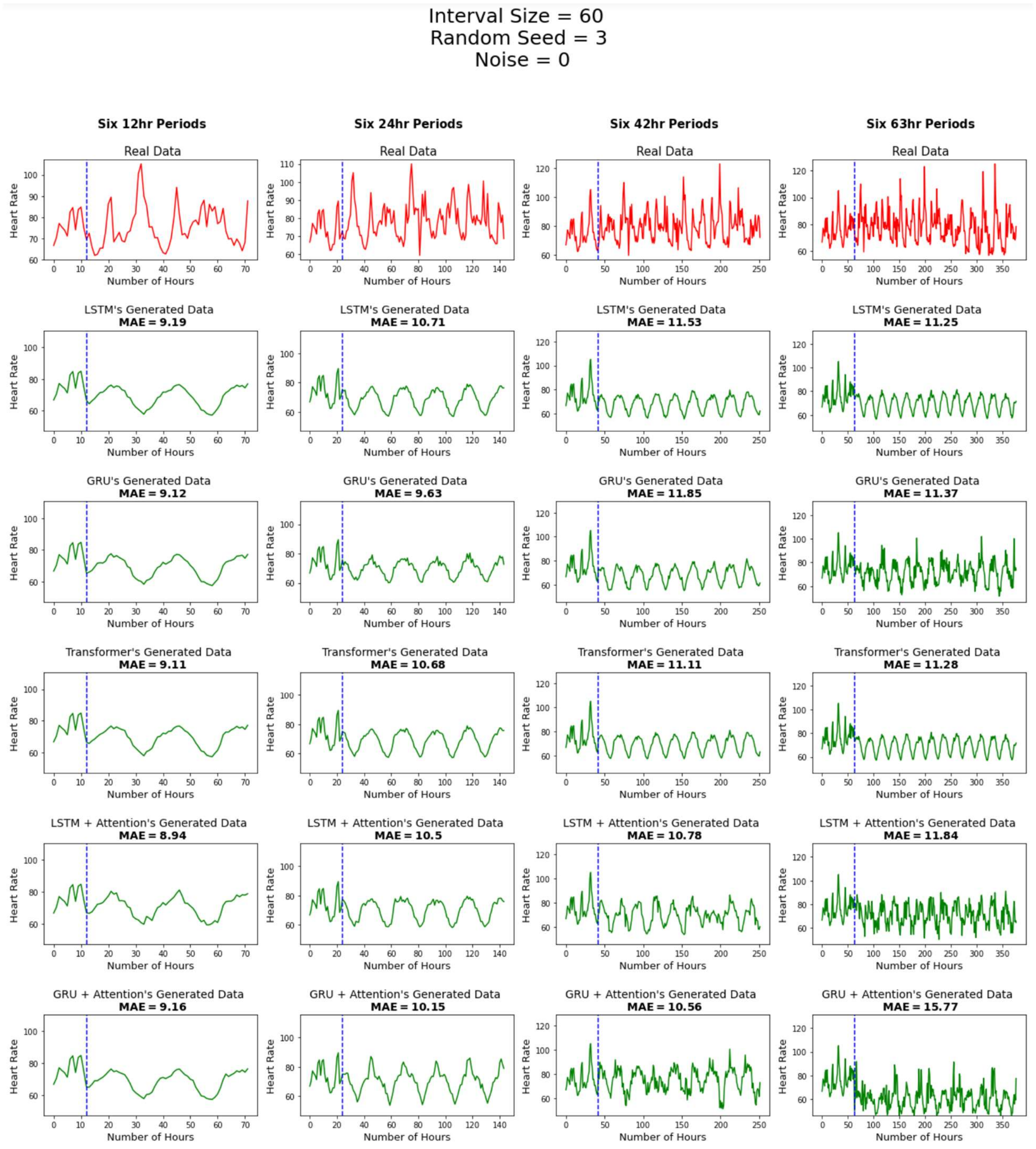

Interval Size = 60
Random Seed = 2
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 5.23
MAE = 5.65
MAE = 6.1
MAE = 6.3
GRU's Generated Data
MAE = 5.37
MAE = 5.99
MAE = 6.11
MAE = 7.17
Transformer's Generated Data
MAE = 5.25
MAE = 5.6
MAE = 6.22
MAE = 6.23
LSTM + Attention's Generated Data
MAE = 5.83
MAE = 5.67
MAE = 6.58
MAE = 7.84
GRU + Attention's Generated Data
MAE = 5.05
MAE = 6.0
MAE = 8.56
MAE = 10.49
Heart Rate
Number of Hours

8.3 Appendix C. MAE Validation at Different Interval Size (Random Seed = 3, Nose = 0)

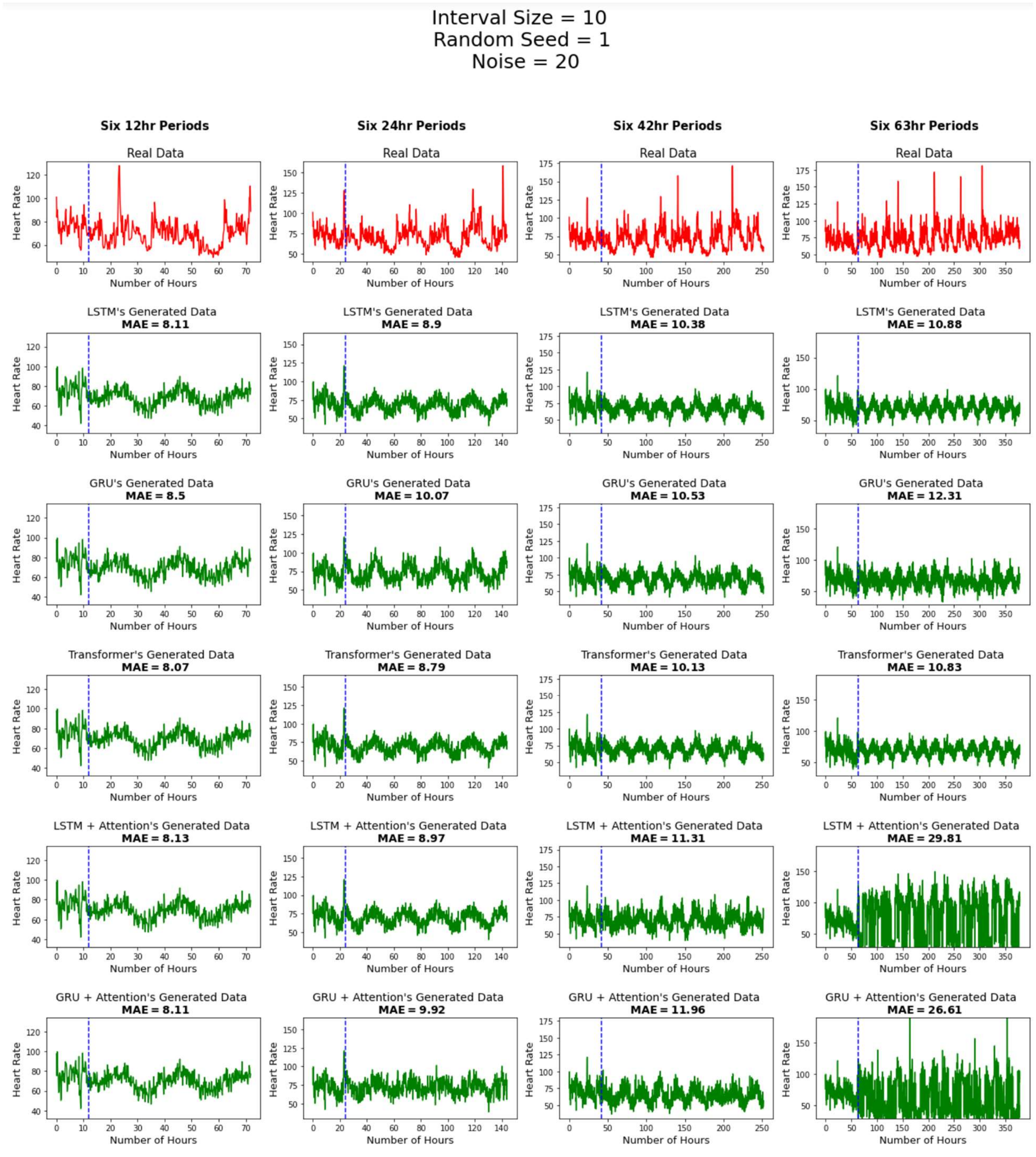


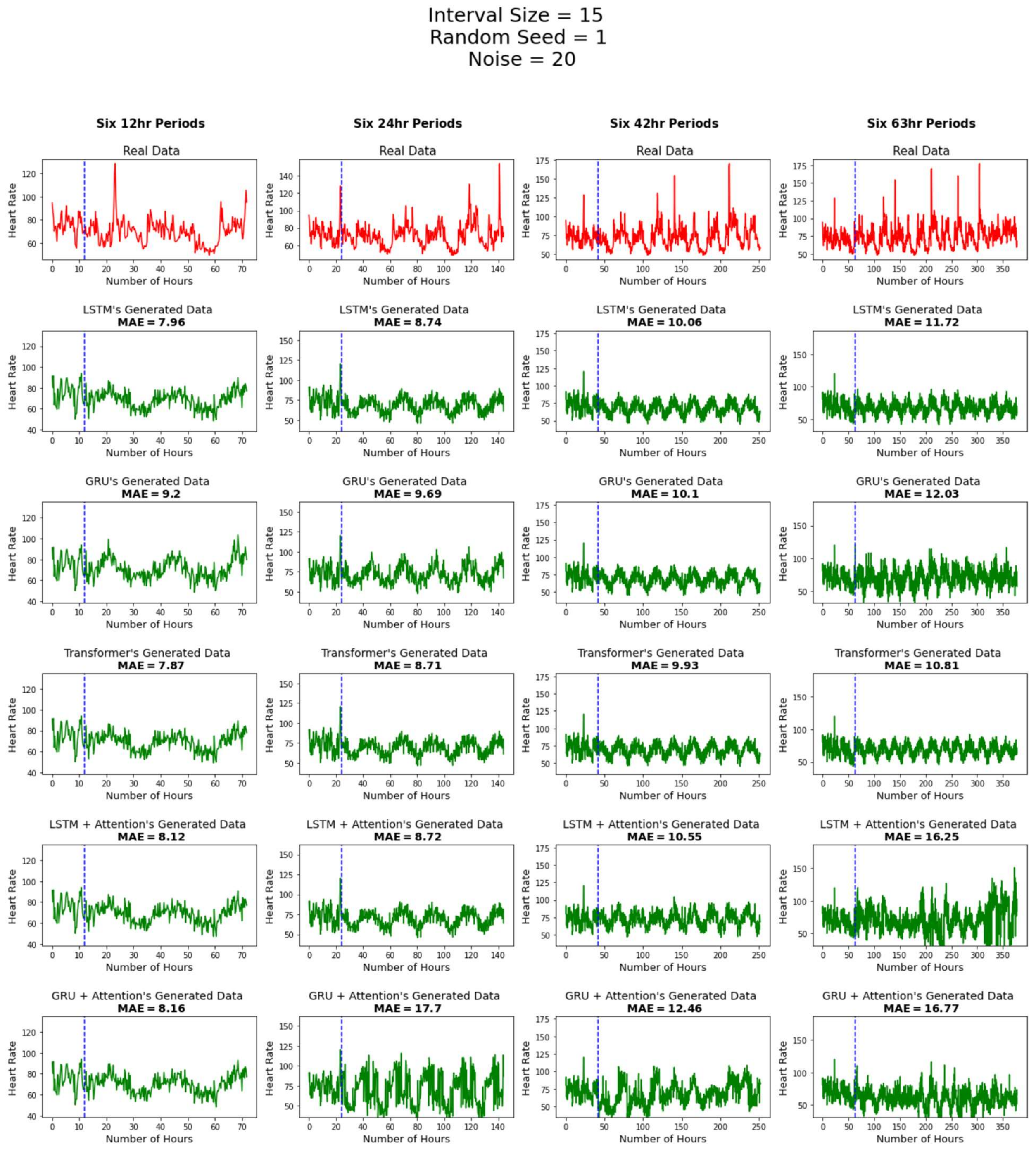

Interval Size = 15
Random Seed = 3
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 9.47
MAE = 11.16
MAE = 11.83
MAE = 12.3
GRU's Generated Data
MAE = 9.78
MAE = 11.18
MAE = 11.9
MAE = 13.15
Transformer's Generated Data
MAE = 9.37
MAE = 11.04
MAE = 11.51
MAE = 11.79
LSTM + Attention's Generated Data
MAE = 9.61
MAE = 11.18
MAE = 11.18
MAE = 17.66
GRU + Attention's Generated Data
MAE = 9.24
MAE = 18.65
MAE = 16.61
MAE = 17.37
Heart Rate
Number of Hours

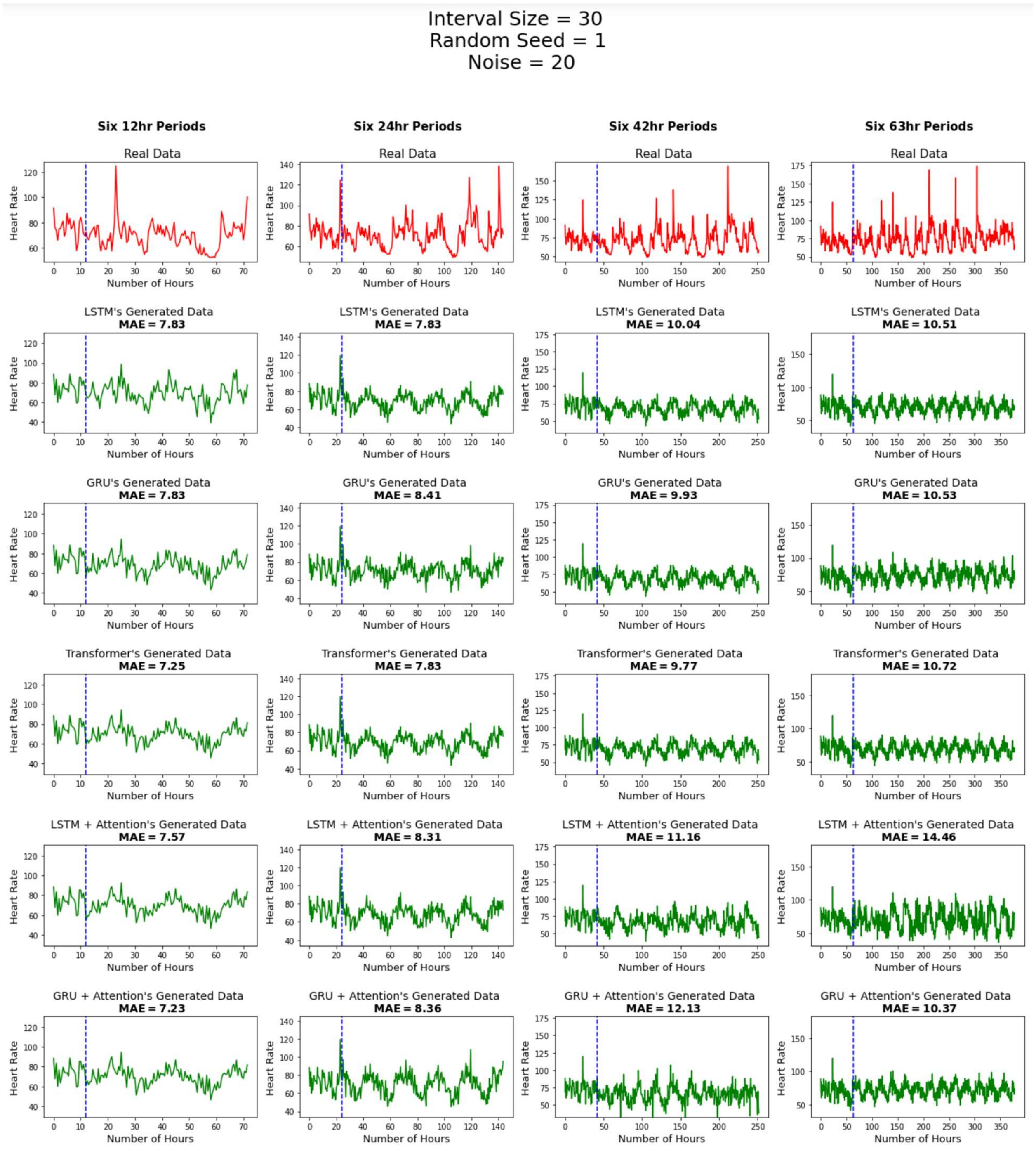

Interval Size = 30
Random Seed = 3
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 10.05
MAE = 10.86
MAE = 11.69
MAE = 11.52
GRU's Generated Data
MAE = 9.84
MAE = 10.24
MAE = 11.59
MAE = 11.42
Transformer's Generated Data
MAE = 9.21
MAE = 10.85
MAE = 11.32
MAE = 11.54
LSTM + Attention's Generated Data
MAE = 9.33
MAE = 11.59
MAE = 13.75
MAE = 13.87
GRU + Attention's Generated Data
MAE = 9.18
MAE = 11.8
MAE = 14.1
MAE = 11.4
Heart Rate
Number of Hours

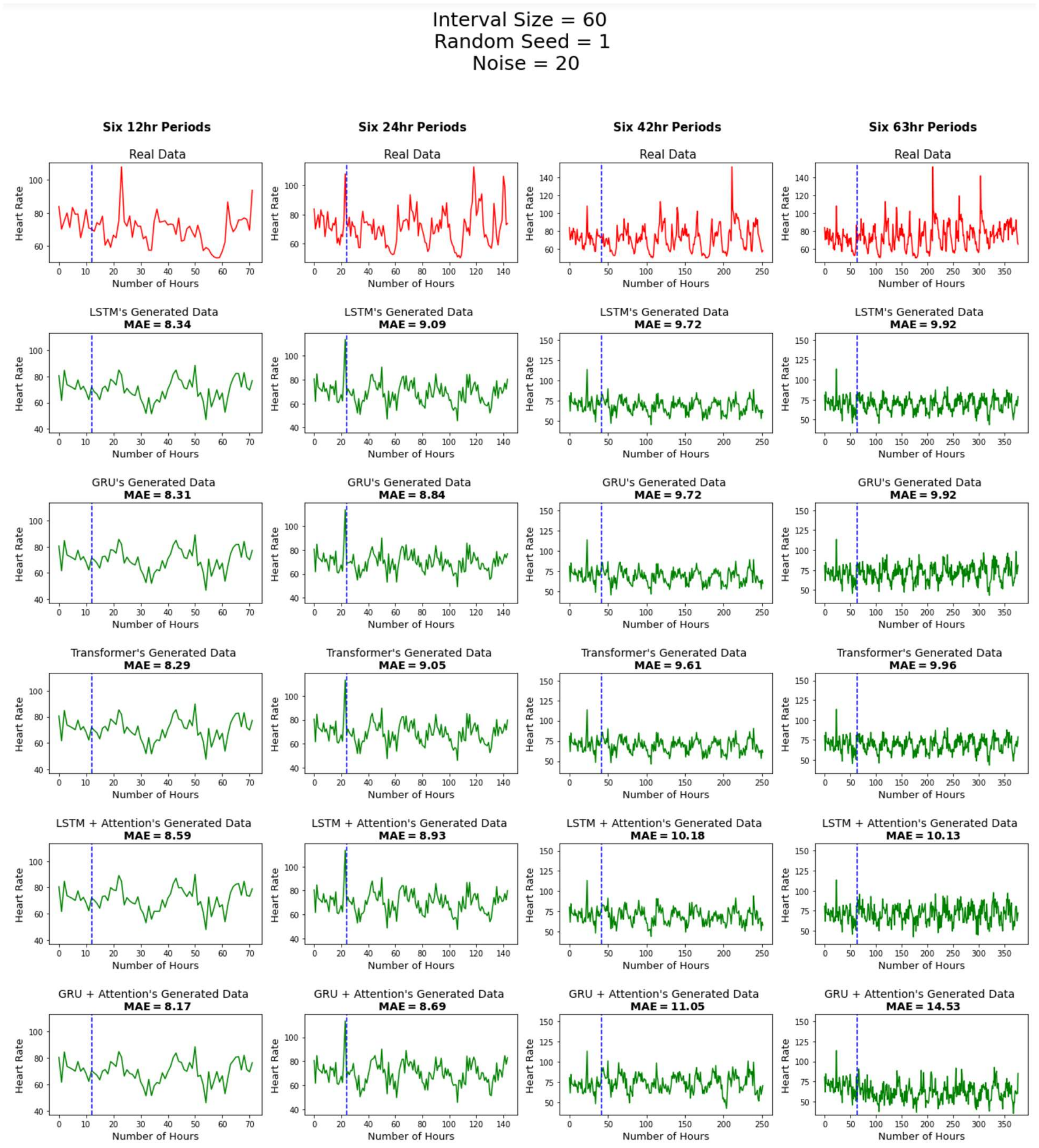

Interval Size = 60
Random Seed = 3
Noise = 0
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 9.19
MAE = 10.71
MAE = 11.53
MAE = 11.25
GRU's Generated Data
MAE = 9.12
MAE = 9.63
MAE = 11.85
MAE = 11.37
Transformer's Generated Data
MAE = 9.11
MAE = 10.68
MAE = 11.11
MAE = 11.28
LSTM + Attention's Generated Data
MAE = 8.94
MAE = 10.5
MAE = 10.78
MAE = 11.84
GRU + Attention's Generated Data
MAE = 9.16
MAE = 10.15
MAE = 10.56
MAE = 15.77
Heart Rate
Number of Hours

8.4 Appendix D. MAE Validation at Different Interval Size (Random Seed = 1, Nose = 20)

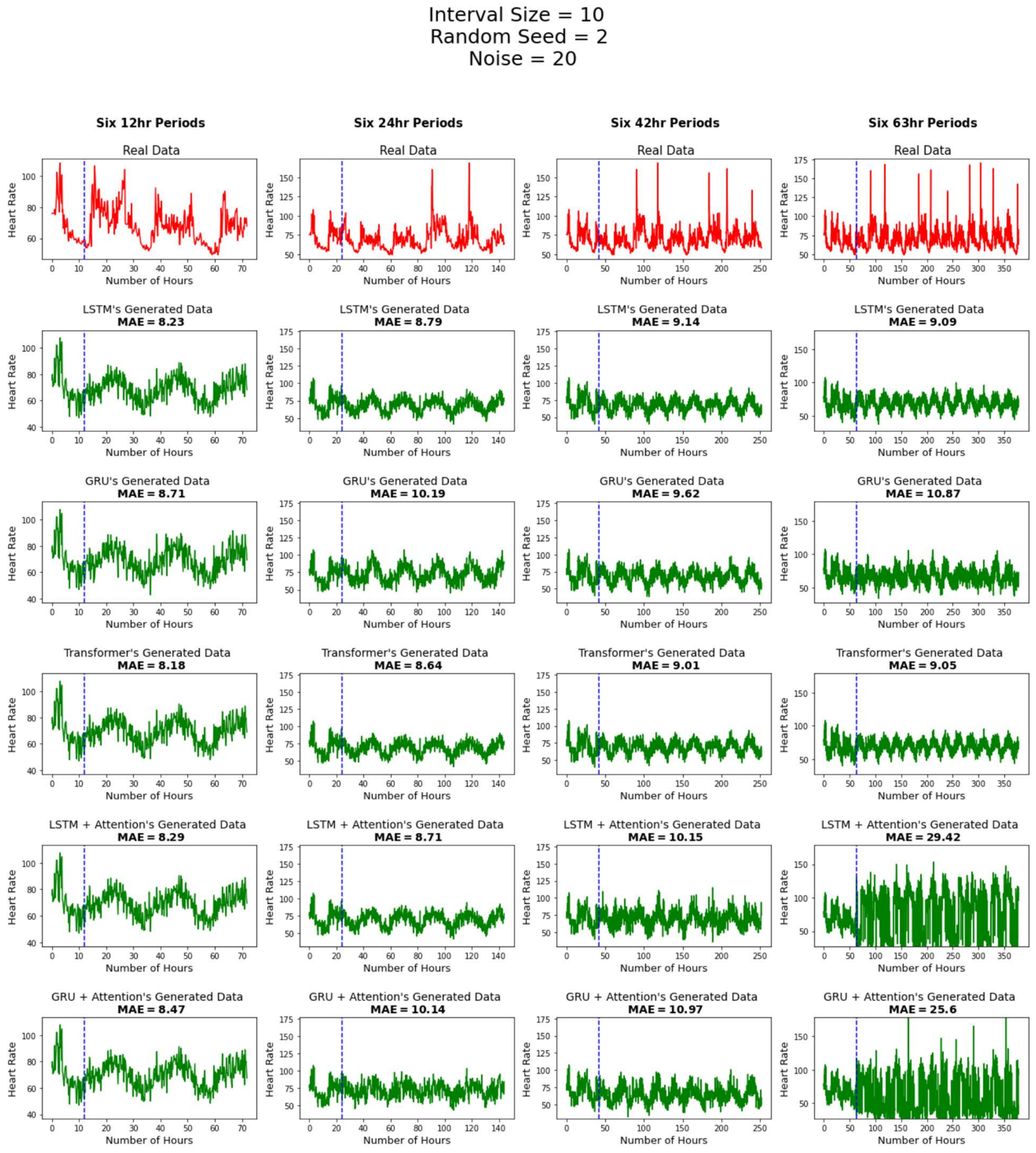



Interval Size = 15
Random Seed = 1
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
Real Data
Real Data
Real Data
LSTM's Generated Data
MAE = 7.96
LSTM's Generated Data
MAE = 8.74
LSTM's Generated Data
MAE = 10.06
LSTM's Generated Data
MAE = 11.72
GRU's Generated Data
MAE = 9.2
GRU's Generated Data
MAE = 9.69
GRU's Generated Data
MAE = 10.1
GRU's Generated Data
MAE = 12.03
Transformer's Generated Data
MAE = 7.87
Transformer's Generated Data
MAE = 8.71
Transformer's Generated Data
MAE = 9.93
Transformer's Generated Data
MAE = 10.81
LSTM + Attention's Generated Data
MAE = 8.12
LSTM + Attention's Generated Data
MAE = 8.72
LSTM + Attention's Generated Data
MAE = 10.55
LSTM + Attention's Generated Data
MAE = 16.25
GRU + Attention's Generated Data
MAE = 8.16
GRU + Attention's Generated Data
MAE = 17.7
GRU + Attention's Generated Data
MAE = 12.46
GRU + Attention's Generated Data
MAE = 16.77
Heart Rate
Number of Hours

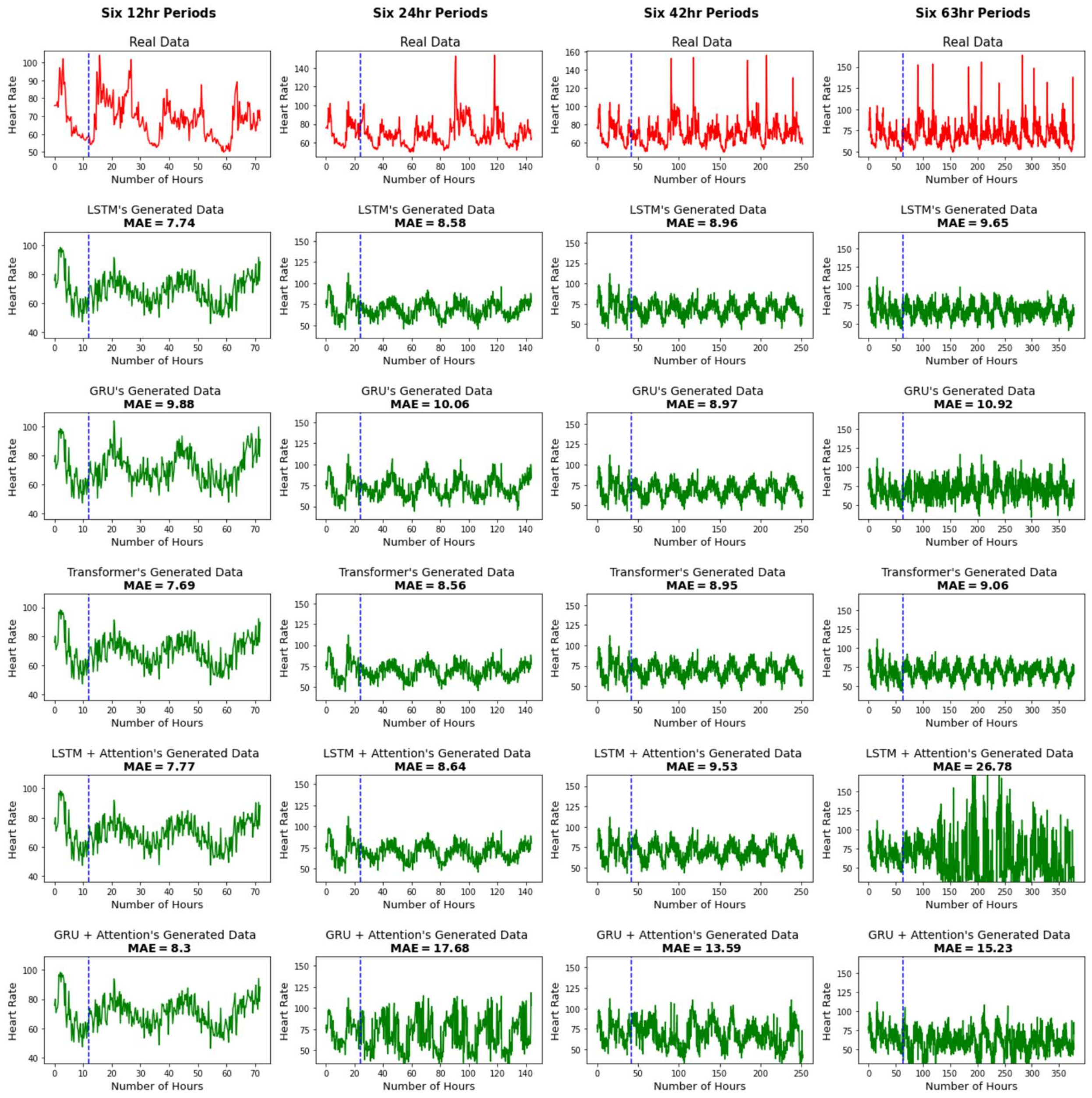

Interval Size = 30
Random Seed = 1
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 7.83
MAE = 7.83
MAE = 10.04
MAE = 10.51
GRU's Generated Data
MAE = 7.83
MAE = 8.41
MAE = 9.93
MAE = 10.53
Transformer's Generated Data
MAE = 7.25
MAE = 7.83
MAE = 9.77
MAE = 10.72
LSTM + Attention's Generated Data
MAE = 7.57
MAE = 8.31
MAE = 11.16
MAE = 14.46
GRU + Attention's Generated Data
MAE = 7.23
MAE = 8.36
MAE = 12.13
MAE = 10.37
Heart Rate
Number of Hours

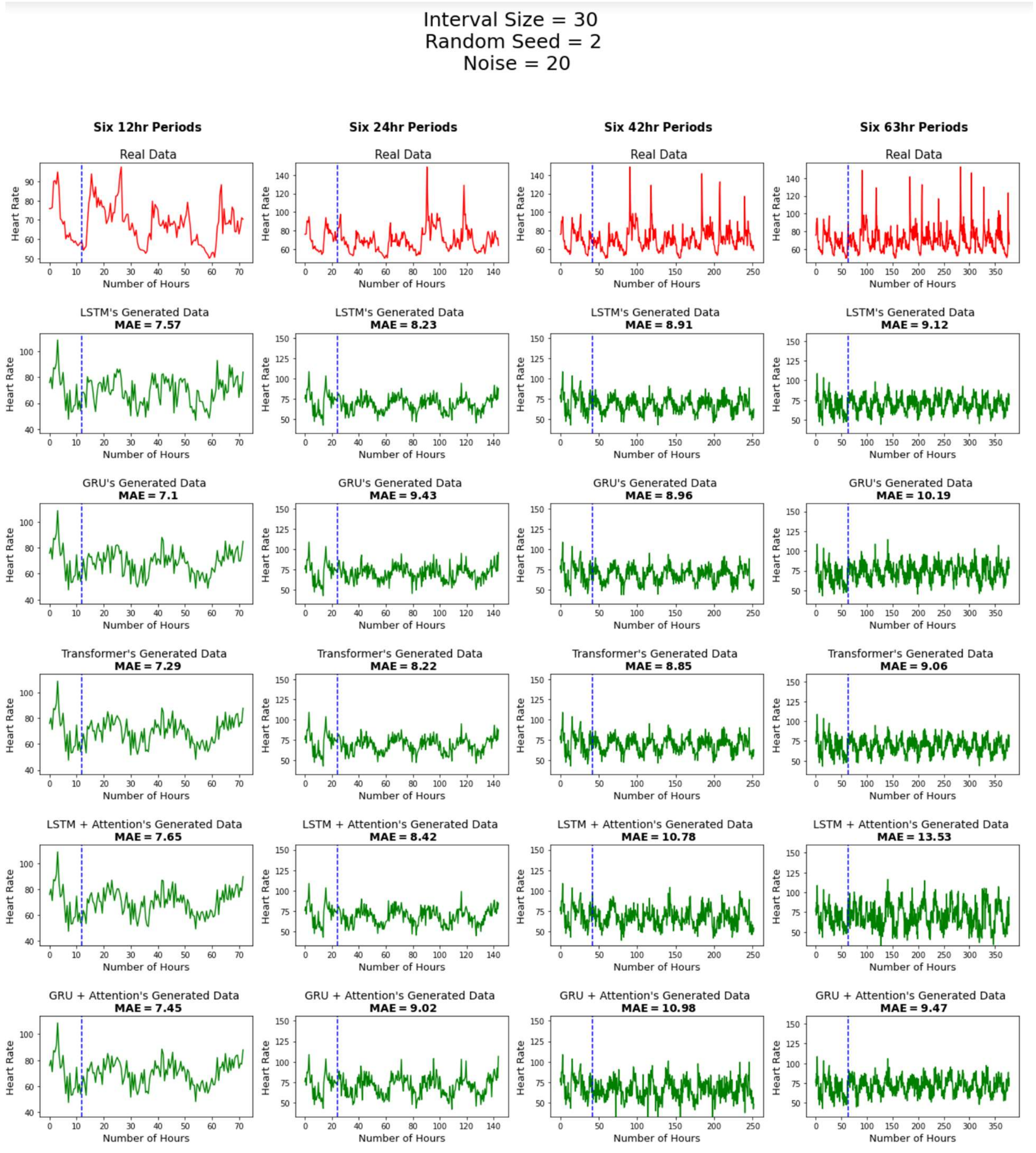

Interval Size = 60
Random Seed = 1
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
GRU's Generated Data
Transformer's Generated Data
LSTM + Attention's Generated Data
GRU + Attention's Generated Data
MAE = 8.34
MAE = 9.09
MAE = 9.72
MAE = 9.92
MAE = 8.31
MAE = 8.84
MAE = 9.72
MAE = 9.92
MAE = 8.29
MAE = 9.05
MAE = 9.61
MAE = 9.96
MAE = 8.59
MAE = 8.93
MAE = 10.18
MAE = 10.13
MAE = 8.17
MAE = 8.69
MAE = 11.05
MAE = 14.53
Heart Rate
Number of Hours

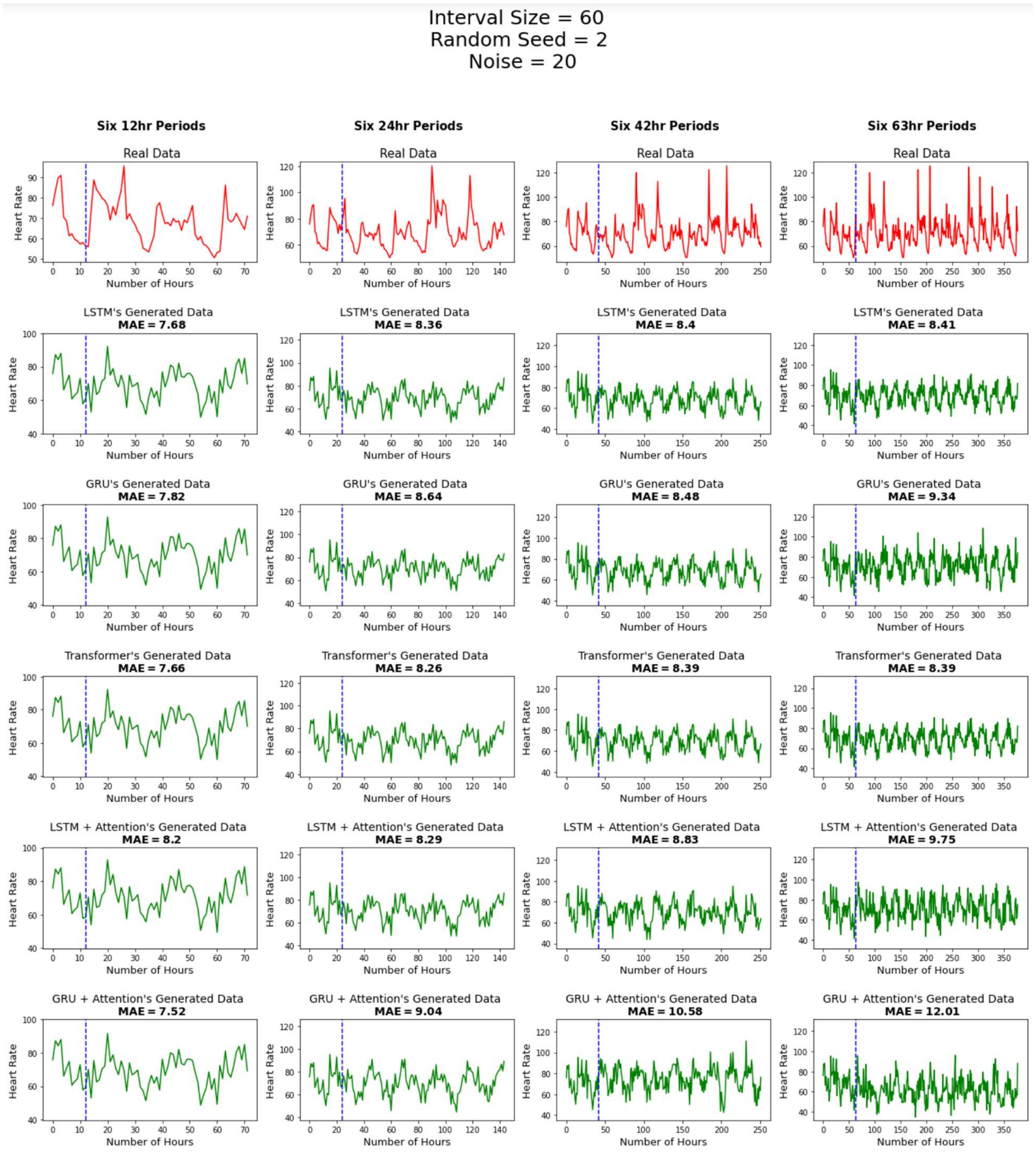
8.5 Appendix E. MAE Validation at Different Interval Size (Random Seed = 2, Nose = 20)



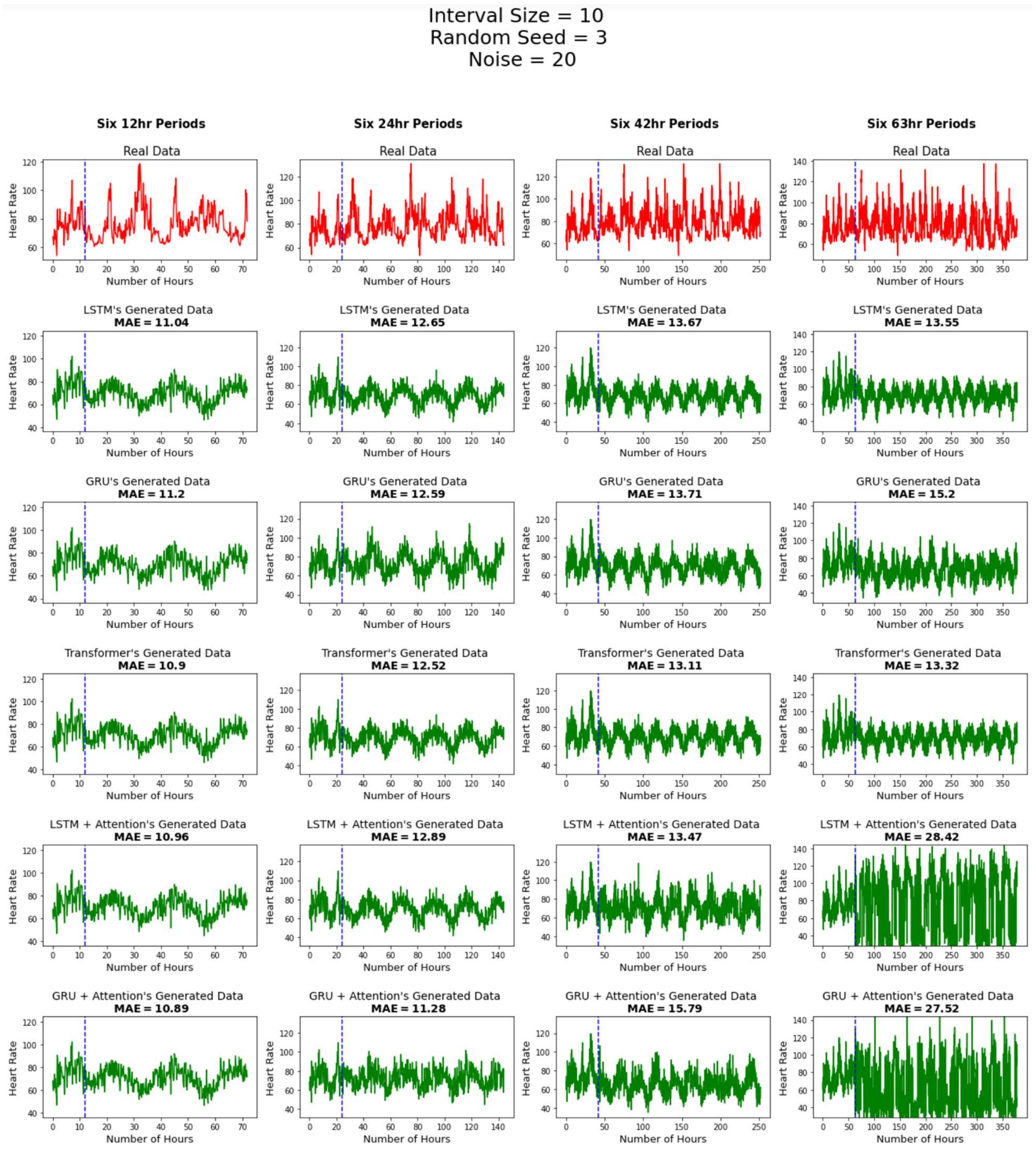
Interval Size = 15
Random Seed = 2
Noise = 20

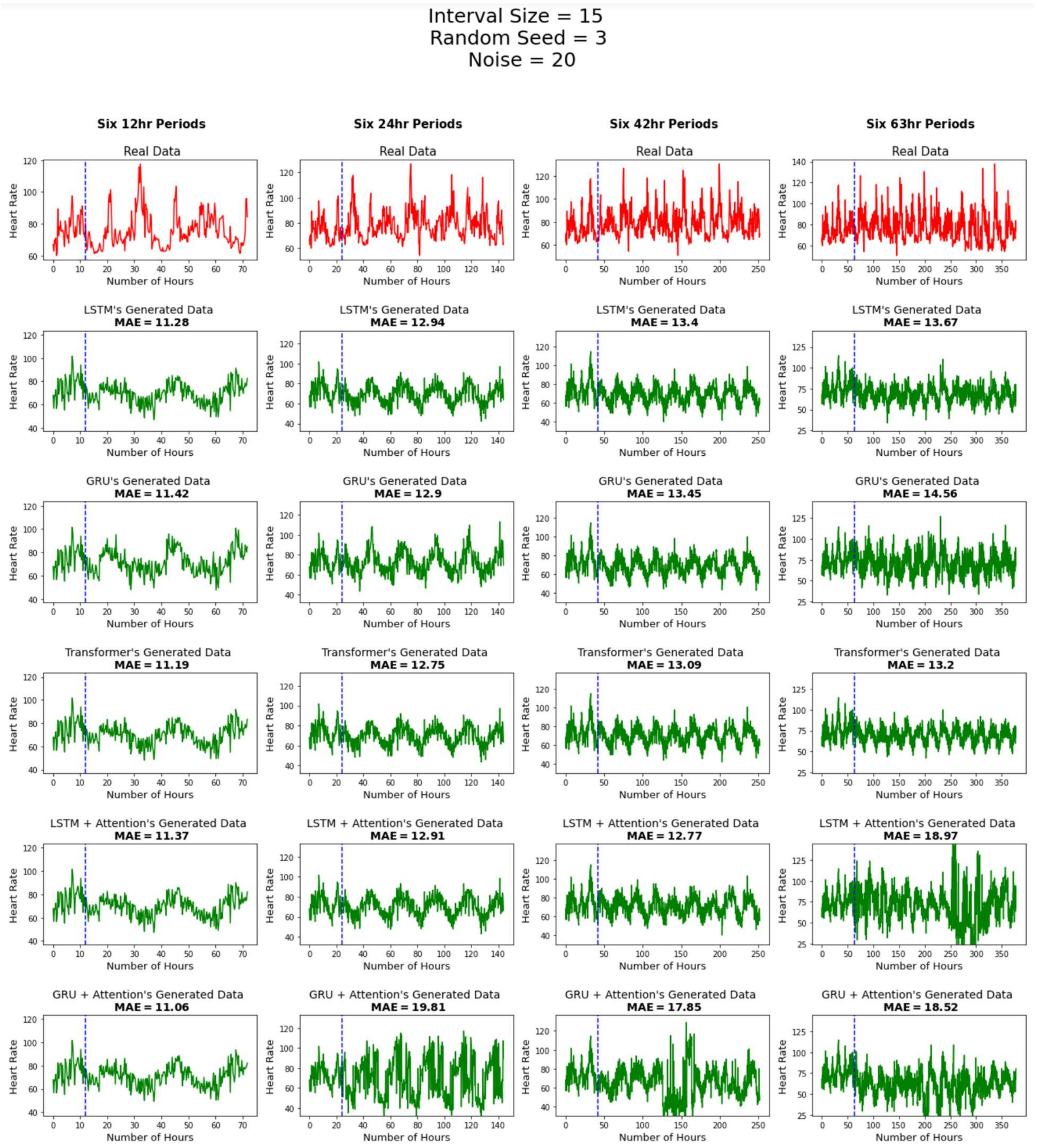


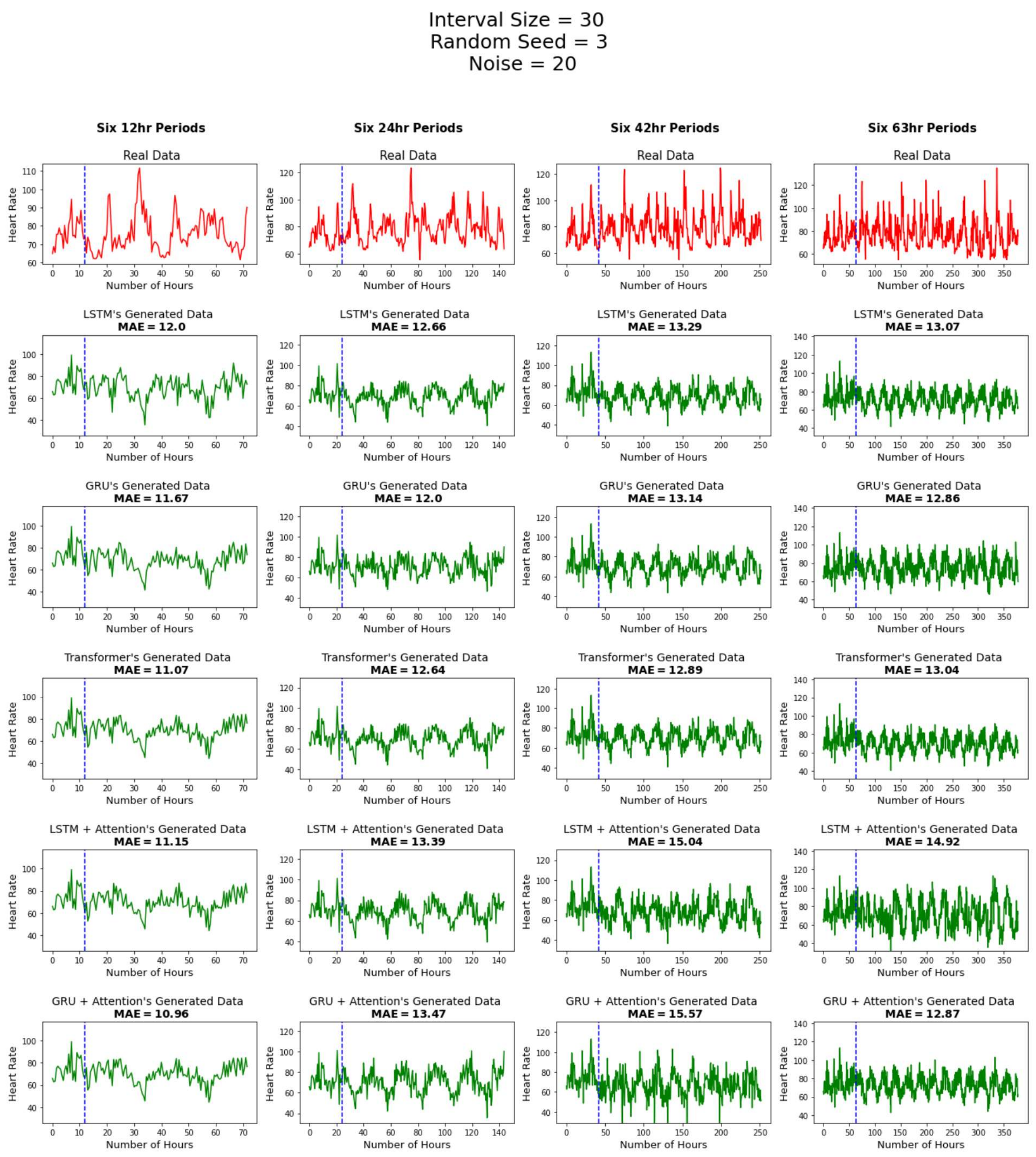

Interval Size = 30
Random Seed = 2
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data MAE = 7.57
LSTM's Generated Data MAE = 8.23
LSTM's Generated Data MAE = 8.91
LSTM's Generated Data MAE = 9.12
GRU's Generated Data MAE = 7.1
GRU's Generated Data MAE = 9.43
GRU's Generated Data MAE = 8.96
GRU's Generated Data MAE = 10.19
Transformer's Generated Data MAE = 7.29
Transformer's Generated Data MAE = 8.22
Transformer's Generated Data MAE = 8.85
Transformer's Generated Data MAE = 9.06
LSTM + Attention's Generated Data MAE = 7.65
LSTM + Attention's Generated Data MAE = 8.42
LSTM + Attention's Generated Data MAE = 10.78
LSTM + Attention's Generated Data MAE = 13.53
GRU + Attention's Generated Data MAE = 7.45
GRU + Attention's Generated Data MAE = 9.02
GRU + Attention's Generated Data MAE = 10.98
GRU + Attention's Generated Data MAE = 9.47
Heart Rate
Number of Hours

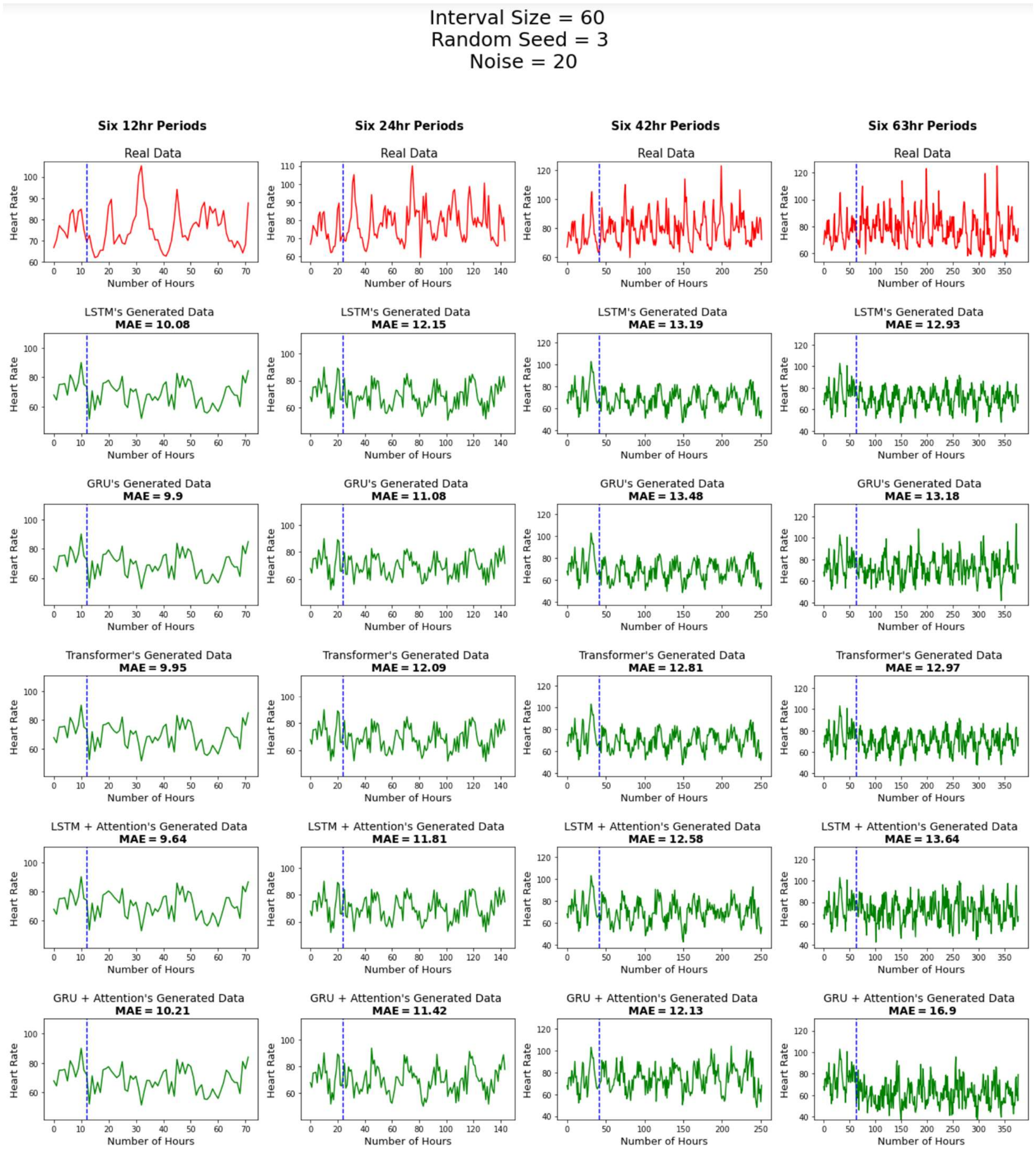

Interval Size = 60
Random Seed = 2
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 7.68
MAE = 8.36
MAE = 8.4
MAE = 8.41
GRU's Generated Data
MAE = 7.82
MAE = 8.64
MAE = 8.48
MAE = 9.34
Transformer's Generated Data
MAE = 7.66
MAE = 8.26
MAE = 8.39
MAE = 8.39
LSTM + Attention's Generated Data
MAE = 8.2
MAE = 8.29
MAE = 8.83
MAE = 9.75
GRU + Attention's Generated Data
MAE = 7.52
MAE = 9.04
MAE = 10.58
MAE = 12.01
Heart Rate
Number of Hours

8.6 Appendix F. MAE Validation at Different Interval Size (Random Seed = 3, Nose = 20)




Interval Size = 15
Random Seed = 3
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 11.28
MAE = 12.94
MAE = 13.4
MAE = 13.67
GRU's Generated Data
MAE = 11.42
MAE = 12.9
MAE = 13.45
MAE = 14.56
Transformer's Generated Data
MAE = 11.19
MAE = 12.75
MAE = 13.09
MAE = 13.2
LSTM + Attention's Generated Data
MAE = 11.37
MAE = 12.91
MAE = 12.77
MAE = 18.97
GRU + Attention's Generated Data
MAE = 11.06
MAE = 19.81
MAE = 17.85
MAE = 18.52
Heart Rate
Number of Hours


Interval Size = 30
Random Seed = 3
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 12.0
MAE = 12.66
MAE = 13.29
MAE = 13.07
GRU's Generated Data
MAE = 11.67
MAE = 12.0
MAE = 13.14
MAE = 12.86
Transformer's Generated Data
MAE = 11.07
MAE = 12.64
MAE = 12.89
MAE = 13.04
LSTM + Attention's Generated Data
MAE = 11.15
MAE = 13.39
MAE = 15.04
MAE = 14.92
GRU + Attention's Generated Data
MAE = 10.96
MAE = 13.47
MAE = 15.57
MAE = 12.87
Heart Rate
Number of Hours


Interval Size = 60
Random Seed = 3
Noise = 20
Six 12hr Periods
Six 24hr Periods
Six 42hr Periods
Six 63hr Periods
Real Data
LSTM's Generated Data
MAE = 10.08
MAE = 12.15
MAE = 13.19
MAE = 12.93
GRU's Generated Data
MAE = 9.9
MAE = 11.08
MAE = 13.48
MAE = 13.18
Transformer's Generated Data
MAE = 9.95
MAE = 12.09
MAE = 12.81
MAE = 12.97
LSTM + Attention's Generated Data
MAE = 9.64
MAE = 11.81
MAE = 12.58
MAE = 13.64
GRU + Attention's Generated Data
MAE = 10.21
MAE = 11.42
MAE = 12.13
MAE = 16.9
Heart Rate
Number of Hours