TIME SERIES FORECAST METHODS: ARMA V.S. LSTM WITH GENETIC ALGORITHM

LU JIALEI

In the field of time series analysis, forecasting holds a central role. Traditional time series models have been carefully developed to capture data patterns and predict future trends. Among these, AutoRegressive Moving Average (ARMA) models are both foundational and essential. Meanwhile, in the rapidly evolving landscape of machine learning, a wide range of algorithms has emerged, each capable of addressing the complexities of time series forecasting. Notably, Long Short-Term Memory (LSTM) models have become fundamental tools in this domain.

This semester paper embarks on a comparative analysis of these two distinct approaches. My objective is to highlight the unique strengths and limitations of each, offering a nuanced perspective on time series forecasting. Through this analysis, I aim to guide practitioners and researchers in selecting the methodology best suited to their specific needs.

KEYWORDS: Time Series Forecast, ARMA, LSTM, Genetic Algorithm.

1. INTRODUCTION

ARMA models, a foundational concept in time series analysis, were among the first principles introduced in our introductory time series course. Rooted in sophisticated mathematical and statistical theories, these models perform commendably in data fitting and forecasting, provided their underlying assumptions hold. However, their limitations become apparent when these assumptions are violated, often leading to suboptimal results. For instance, structural shifts in the data, such as changes in the time series levels, can mislead ARMA models, resulting in inaccurate fits. Additionally, if the variance of disturbances fluctuates, ARMA models struggle to adjust, further compromising their effectiveness, even after extensive data cleansing and adjustments.

In contrast, the expansive field of machine learning offers a variety of algorithms for the complex task of time series forecasting. Long Short-Term Memory (LSTM), a prominent algorithm within the recurrent neural network (RNN) family, stands out for its unique approach. Unlike ARMA models, LSTM handles time series data with robustness, even when the data deviate significantly from the assumptions required by ARMA. However, a notable drawback of LSTM and other neural network models lies in their "black-box" nature, lacking the transparency and interpretability that characterize traditional statistical models.

This paper aims to present a detailed comparative analysis of ARMA and LSTM models, highlighting their respective strengths and limitations. Additionally, I introduce an innovative approach by incorporating a Genetic Algorithm to optimize LSTM performance, demonstrating its significant advantage over ARMA models in time series forecasting.

The remainder of this paper is organized as follows: Section 2 reviews the relevant literature. Section 3 provides details on the dataset and the technical specifications of the ARMA, LSTM, and Genetic Algorithm approaches. Section 4 presents the forecasting results, while Section 5 summarizes the findings from this comparative analysis.

Lu Jialei: lujialei@stud.uni-frankfurt.de

2. LITERATURE REVIEW

In traditional econometrics, ARMA models serve as the foundational approach for fitting and forecasting time series data. This concept was initially formulated by Peter Whittle in 1951 and further refined by Box and Jenkins in the 1970s, marking a pivotal advancement in the field.

In contrast, machine learning has introduced new paradigms in time series forecasting, with Recurrent Neural Networks (RNNs) and their advanced variant, Long Short-Term Memory (LSTM), now taking center stage. RNNs have a storied history, originating in the late 1980s as documented by Werbos, and have been applied across a variety of sequence-learning tasks, including the intricate demands of time series analysis.

To address the gradient vanishing and exploding issues often encountered in RNNs, the LSTM architecture was developed. As elucidated by Karpathy, Johnson, and Feifei Li in 2016, LSTMs possess a unique capacity to store and retrieve information over extended temporal periods. This capability is achieved through the integration of gating mechanisms and a constant error carousel, establishing LSTM as a cornerstone in modern time series forecasting within the machine learning domain.

3. DATA AND METHODS

3.1. *Data*

In this paper, I will use two distinct datasets to compare the forecasting capabilities of ARMA models and LSTM in time series analysis.

First, I will generate a synthetic time series dataset spanning 1,000 periods using ARMA models, both with and without structural changes. By applying ARMA models and LSTM to fit this data, I aim to demonstrate that LSTM can perform comparably to ARMA when working with pure, artificially generated ARMA data. Additionally, I will illustrate that LSTM outperforms ARMA when the data includes structural changes.

As an illustrative example, I have selected the ARMA(2,2) model with a constant variance of 1 for the innovations, defined as follows:

$$x_t = 0.4 \cdot x_{t-1} + 0.2 \cdot x_{t-2} + u_t + 0.1 \cdot u_{t-1} + 0.5 \cdot u_{t-2}$$

Subsequently, I will introduce changes to the variance of the shocks. In this modified configuration, the variance of u_t remains at 1 for t within the range $[1, 500]$, and decreases to 0.5 for t within the range $[501, 1000]$.

For the second dataset, I will use quarterly U.S. GDP data spanning from the first quarter of 1947 to the fourth quarter of 2018. Real-world data often presents challenges for ARMA models due to frequent violations of their underlying assumptions. In contrast, LSTM demonstrates robust forecasting capabilities, even amid such complexities. I will also employ a Genetic Algorithm to optimize the selection of the window size for LSTM—a critical parameter discussed in later sections. While theory suggests that this optimization can significantly enhance LSTM's performance, I will address the practical challenges of this process, explaining why ideal results may not always be achievable.

3.2. *ARMA*

ARMA models are basic time series model. The general form of ARMA(p,q) is that:

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + u_t + \sum_{j=1}^{q} \psi_j u_{t-j}$$

Or we can use lag operator to express it:

$$(1-\sum_{i=1}^{p}\phi_i L^i)x_t = (1+\sum_{j=1}^{q}\psi_j)u_t$$

$$A(L)x_t = B(L)u_t$$

To fit an ARMA model, we need the data to meet some assumptions (Hamilton, 1994):

1. **Ergodicity:** sample mean and sample variance-covariance will converge to population mean and variance-covariance with probability when $T \to \infty$:

$$\bar{x} \equiv \frac{1}{T}\sum_{t=1}^{T} x_t \to^p E(x_t),$$

$$\frac{1}{T-j}\sum_{t=j+1}^{T}(x_t-\mu)(x_{t-j}-\mu) \to^p \gamma_j$$

2. **Week Stationary:** the expectation and variance-covariance of x_t IS NOT a function of t:

$$E(x_t) = \mu,\ E(x_t-\mu)(x_{t-j}-\mu) = \gamma_j \forall t, j$$

When applying ARMA models to real-world data, it's almost inevitable that these two assumptions are consistently violated, resulting in suboptimal forecast outcomes. I will provide concrete evidence of these violations in the subsequent sections.

3.3. *Neural Network Methods*

Long Short-Term Memory (LSTM), introduced by Hochreiter and Schmidhuber in 1997, is a type of Recurrent Neural Network (RNN). To provide a clear understanding of LSTM, I will start with a brief overview of RNNs.

3.3.1. *ANN*

In Artificial Neural Networks (ANNs)—which can be thought of as multi-layer perceptrons—the process begins by inputting data, represented as $x_1, x_2, \dots, x_k$ into the perceptron. The perceptron first computes a weighted sum of this input, then applies an activation function to produce an output. The weight vector $w_1, w_2, \dots, w_k$ can be adjusted to minimize a predefined loss function, yielding a perceptron capable of performing various forecasting tasks. This foundational understanding of ANNs sets the stage for exploring the complexities of LSTM, which I will discuss in the following sections.

When we assemble a multitude of these perceptrons into layers, we create an Artificial Neural Network (ANN) capable of significantly enhanced performance compared to a single perceptron. Within the framework of an ANN, the input layer undertakes the task of receiving initial information and transforming it into intermediary results. These intermediary results are subsequently conveyed through the hidden layers, which further process them before ultimately reaching the output layer. At the conclusion of this journey, the output layer yields the final result. The optimization of weight values across the network plays a pivotal role, as we fine-tune them to minimize the loss function. Following this adjustment and rigorous training, an ANN emerges as a powerful tool for diverse forecasting tasks.

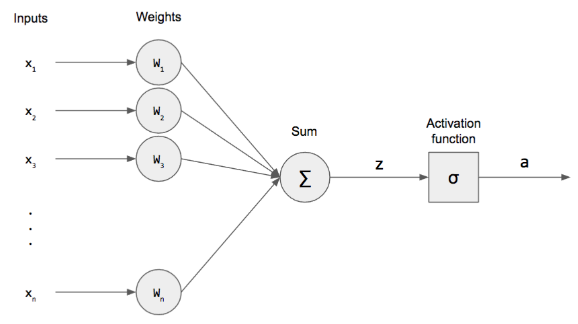


FIGURE 1.—Perceptron

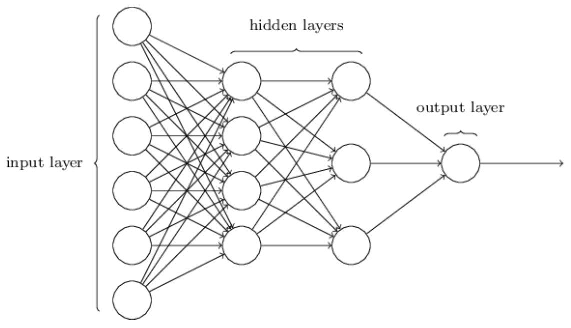


FIGURE 2.—ANN

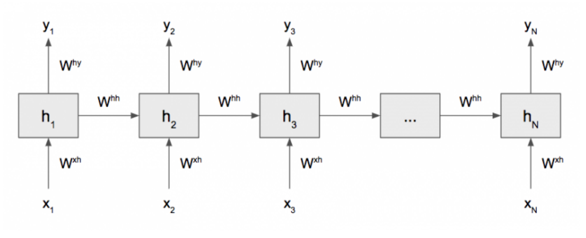


FIGURE 3.—RNN

3.3.2. *RNN*

However, ANNs possess a limitation—they are unable to capture past information, a critical component in the context of time series forecasting. To address this limitation, we turn to Recurrent Neural Networks (RNNs), which extend the capabilities of ANNs. In essence, RNNs can be viewed as a specialized variant of ANNs. Within the structure of an RNN, each perceptron in the input layer receives K inputs denoted as $x_1, x_2, \cdots, x_k, \cdots, x_K$. Among these inputs, the initial K inputs $x_1, x_2, \cdots, x_k$ originate from external sources, while the remaining $K - k$ inputs, $x_k, x_{k+1}, \ldots, x_K$, represent the output results generated by the preceding unit in the sequence. This recursive feedback mechanism is a hallmark feature of RNNs, enabling them to retain and utilize past information in their calculations,an essential attribute for effective time series forecasting.

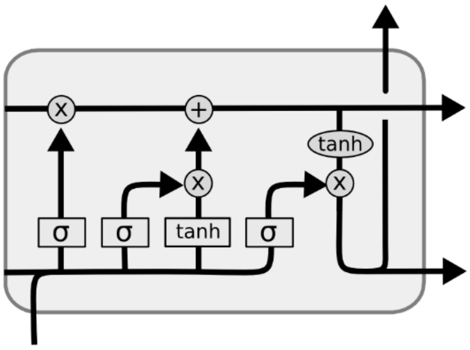


FIGURE 4.—Long-Short-Term Memory

In theory, Recurrent Neural Networks (RNNs) possess the capability to handle a wide range of time series forecasting tasks. However, practical implementation often encounters the challenge of gradient vanishing and exploding problems. These issues result in the diminishing impact of information from early time units or periods on today's forecasting. Consequently, RNNs may struggle to effectively capture long-range dependencies within the data.

3.3.3. *LSTM*

To overcome the challenges associated with training RNNs, the Long Short-Term Memory (LSTM) architecture was developed, as detailed in the work of Andrej Karpathy, Justin Johnson, and Feifei Li in 2016. LSTM introduces a gate mechanism, allowing it to selectively retain and utilize important information from the past. This innovation equips LSTM with the capacity to incorporate and leverage historical data effectively, ensuring that past observations continue to influence and enhance the accuracy of today's forecasts.

In this paper, my primary focus is not on delving into the intricate mechanisms underlying LSTM, as that topic extends beyond the scope of our current discussion. Instead, I will concentrate on how LSTM handles time series data, aligning with the central theme of this paper.

Consider a sequence of time series data denoted as $x_1, x_2, x_3, \ldots, x_{100}$. To facilitate the analysis, I can reorganize this data into a matrix format:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ x_2 & x_3 & x_4 & x_5 & x_6 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{96} & x_{97} & x_{98} & x_{99} & x_{100} \end{bmatrix}$$

Now, let's consider the following arrangement: I designate the first four elements in each row as the external inputs for each respective time period, for instance, x_1, x_2, x_3, x_4 or $x_{96}, x_{97}, x_{98}, x_{99}$. Simultaneously, I designate the last element in each row as the actual observations that we aim to fit, such as x_5, x_6, or x_{100}. Additionally, I take into account that the output of LSTM units will also serve as part of the inputs.

In this context, I treat the first four columns and the outputs from the LSTM units as the input set for a Neural Network, while the last column represents the actual observed results that the Neural Network aims to predict. It's important to note that I refer to the number 4 in this context as the "window size." After the iterative process of adjusting weight values to minimize the loss function, we derive a well-tuned LSTM model capable of achieving highly accurate forecasting results. This methodology allows us to harness the power of LSTM for effective time series forecasting within the framework of this paper.

3.4. *Genetic Algorithm*

Whether employing ARMA models or LSTM for time series forecasting, both approaches fundamentally involve the abstract process of minimizing a loss function. Numerous methodologies exist for achieving this goal, and one such approach, which I will utilize in this study, is the Genetic Algorithm.

Genetic Algorithm belongs to the category of optimization techniques renowned for their ability to intelligently explore vast and intricate solution spaces to identify values that approximate the global optimum (David J. Montana, Lawrence Davis, 1989). This algorithm draws inspiration from the principles of natural selection and evolution, employing a set of methods and processes to iteratively enhance the population's genetic makeup, ultimately leading to individuals that yield lower loss function values.

The key components of Genetic Algorithm include:

1. **Selection:**This mechanism selects individuals for reproduction based on their fitness, favoring those with characteristics conducive to achieving lower loss function values.
2. **Crossover:** The crossover process involves merging the genetic information of two individuals, creating offspring that potentially inherit beneficial traits from both parents.
3. **Mutation:** With a certain probability, mutation introduces random alterations to genetic strings, allowing for the exploration of novel genetic traits and potentially improving the population's overall fitness (Deep Malya Mukhopadhyay, Maricel O. Balitanas et al., 2009).

Through successive generations of evolution, Genetic Algorithm refines the population, gradually leading to results that provide the lowest loss function values. This methodology serves as a powerful optimization tool, particularly in complex and high-dimensional problem spaces, such as those encountered in time series forecasting.

3.5. *Genetic Algorithm on Neural Network*

Genetic Algorithm (GA), as an evolutionary optimization technique, synergizes effectively with Neural Networks, aiding in the quest for optimal model configurations and parameter adjustments to enhance their efficacy (Haruna Chiroma, Ahmad Shukri Mohd Noor et al., 2017). GA can substantially improve Neural Networks across three critical dimensions: topology optimization, weight values optimization, and independent variables selection.

Within the scope of this paper, I will employ GA to tackle two specific aspects of Neural Network enhancement:

1. **Optimal Window Size Selection:** This pertains to the realm of independent variables selection. By leveraging GA, I will identify the most suitable window size, optimizing the choice of input data periods for the LSTM model.
2. **Optimal Number of Units:** This falls under the category of basic topology optimization. Using GA, I will determine the ideal number of units or neurons within the LSTM architecture, ensuring that the network structure aligns optimally with the given time series forecasting task.

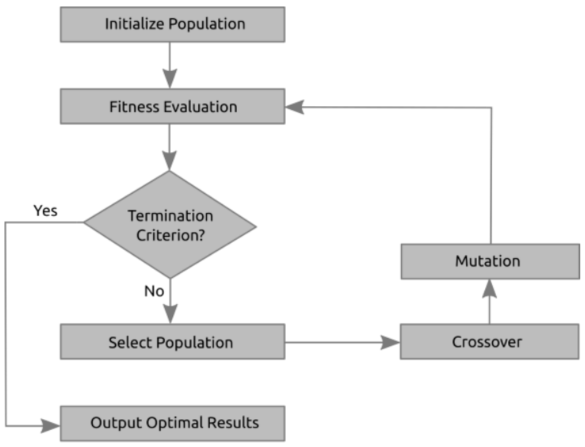


FIGURE 5.—Process of Genetic Algorithm

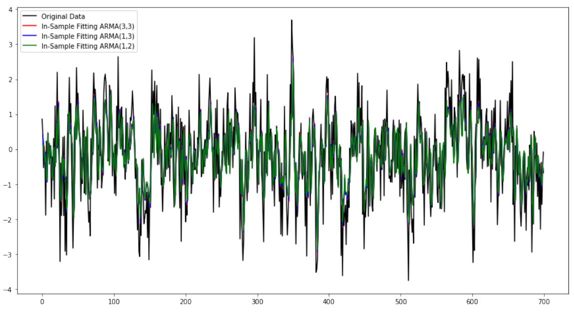


FIGURE 6.—In-Sample Fitting of ARMA Models

By harnessing the power of GA in these aspects, I aim to fine-tune the LSTM model effectively, ultimately improving its forecasting performance for the real-world time series data examined in this study.

4. RESULTS

4.1. *Generated ARMA Data without Struactual Changes*

In this study, I will employ both ARMA models and LSTM to fit and forecast generated ARMA(2,2) data. To begin, I will present the outcomes obtained from the ARMA models. After an exhaustive analysis that considers criteria such as AIC, BIC, HQIC values, as well as the examination of autocorrelation and partial autocorrelation functions (ACF and PACF), the most suitable ARMA models emerge as ARMA(3,3), ARMA(1,3), and ARMA(1,2).

Figure 6 showcases the in-sample fitting results, providing a visual representation of how well these selected ARMA models capture the underlying patterns within the generated ARMA(2,2) data. Subsequently, in Figure 7, I present the out-of-sample forecasting results, shedding light on the models' predictive capabilities when faced with data points beyond the training period. These results will serve as a crucial benchmark for evaluating the performance of LSTM in the subsequent sections.

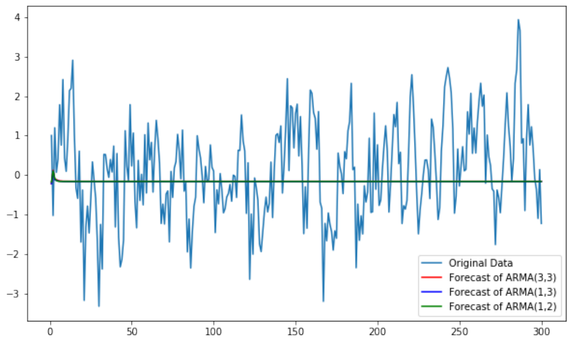


FIGURE 7.—Out-Sample Forecasts of ARMA Models

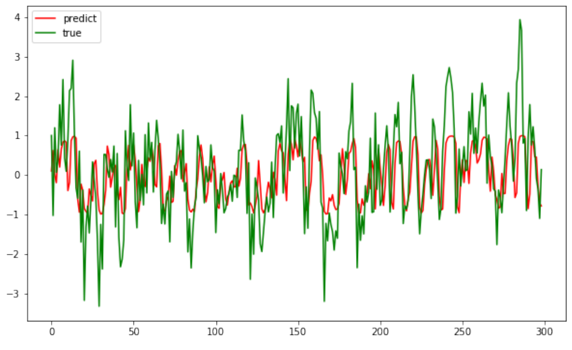


FIGURE 8.—Forecasts of LSTM Models

Upon a careful examination of the presented graphs, several noteworthy observations can be made:

1. ARMA models demonstrate commendable performance when it comes to fitting the generated data. They effectively capture and replicate the underlying patterns within the ARMA(2,2) dataset, as evidenced by the close alignment between the model predictions and the actual data points.
2. Conversely, ARMA models exhibit subpar performance in forecasting the generated data. As seen in Figure 7, the predictive accuracy of these models diminishes as they attempt to project data points beyond the training period. This discrepancy between the model's forecasted values and the actual data points highlights a notable limitation of ARMA models in handling complex time series forecasting tasks.

Moving forward, in Figure 8, I will present the results obtained from LSTM, shedding light on how this deep learning algorithm fares in comparison to the ARMA models in terms of both in-sample fitting and out-of-sample forecasting for the same ARMA(2,2) data.
From the graph, we can see that, the results from LSTM are much better than those of ARMA models.

4.2. *Generated ARMA Data with Struactual Changes*

Continuing with the analysis, I will now use both ARMA models and LSTM to fit the generated ARMA(2,2) data that incorporates structural changes, followed by forecasting.

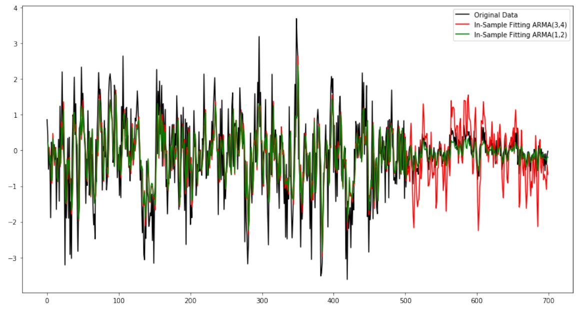


FIGURE 9.—In-Sample Fitting of ARMA Models

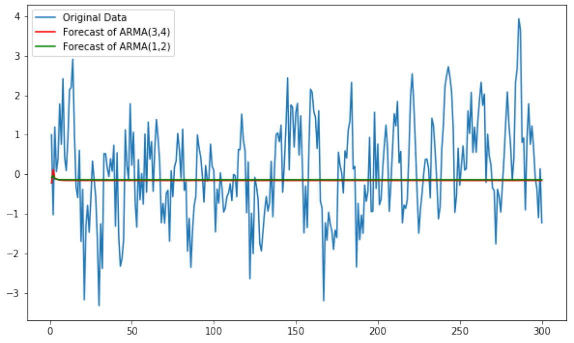


FIGURE 10.—Out-Sample Forecasts of ARMA Models

To begin, let's examine the outcomes obtained from the ARMA models. After an extensive evaluation that takes into account criteria such as AIC, BIC, HQIC values, as well as the examination of autocorrelation and partial autocorrelation functions (ACF and PACF), the most suitable ARMA models for this modified dataset are identified as ARMA(3,4) and ARMA(1,2).

Figure 9 provides an insight into the in-sample fitting results, illustrating how well these selected ARMA models adapt to the structural changes present in the generated ARMA(2,2) data. Subsequently, Figure 10 offers a glimpse into the out-of-sample forecasting results, showcasing the models' ability to predict data points in this scenario where structural changes are at play. These results serve as a crucial reference point for the forthcoming evaluation of LSTM's performance in handling this challenging dataset.

Upon a thorough analysis of the provided graphs, several key insights emerge:

1. ARMA models struggle to effectively fit the generated data that includes structural changes. This outcome aligns with expectations, as structural changes fundamentally violate the underlying assumptions of ARMA models. Consequently, the models' performance in fitting such data is notably suboptimal.
2. ARMA models also exhibit poor forecasting performance when confronted with data that incorporates structural changes. The models struggle to adapt and provide accurate predictions in this complex scenario.

Moving on to the results obtained from LSTM, as depicted in Figure 11, a stark contrast becomes evident. LSTM outperforms ARMA models significantly, offering substantially

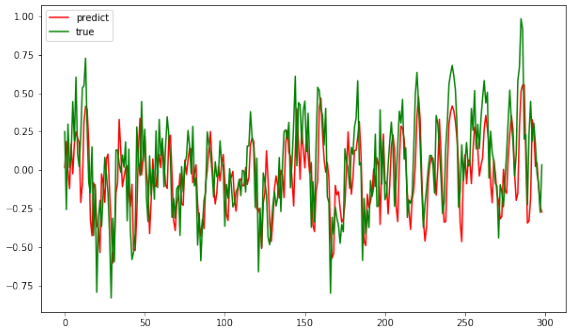


FIGURE 11.—Forecasts of LSTM Models

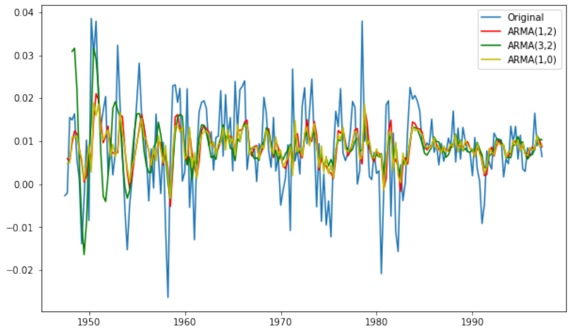


FIGURE 12.—In-Sample Fitting of ARMA

improved forecasting accuracy. LSTM's ability to incorporate information about structural changes during the forecasting process contributes to this enhanced performance.

In essence, while ARMA models attempt to find a single unified pattern to fit and forecast the data, LSTM approaches forecasting in a more adaptive manner, considering the data in a staged fashion. This enables LSTM to effectively account for the presence of structural changes and deliver superior forecasting results, as observed in the graphs.

4.3. *Real GDP Data*

Now, let's turn our attention to evaluating the performance of both ARMA models and LSTM on real GDP data. Beginning with the assessment of ARMA models, it's worth noting that three ARMA models emerge as the top candidates based on criteria such as AIC, BIC, and HQIC values: ARMA(1,2), ARMA(3,2), and ARMA(1,0). These models exhibit commendable performance in fitting the in-sample data, as demonstrated in Figure 12.
However, the real challenge arises when we move on to forecasting. Despite the root mean square errors (RMSEs) hovering around 6.6%, the forecasts generated by these ARMA models fail to provide meaningful insights, as illustrated in Figure 13.

The predictions tend to remain relatively constant, lacking the ability to capture fluctuations in the data. The forecasts struggle to accurately represent significant deviations from the expected trends.

1. After some fluctuation, the predictions keep constant.
2. The Forecasts cannot capture the large deviation well.

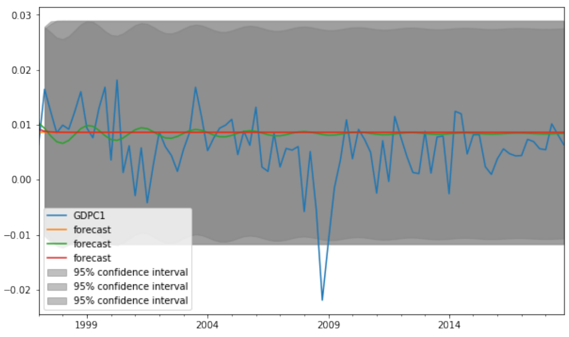


FIGURE 13.—Out-Sample Forecasts of ARMA

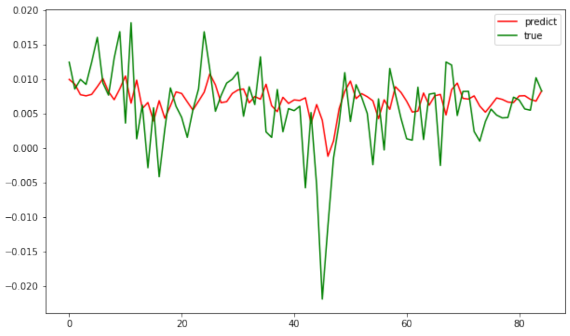


FIGURE 14.—Forecasts of LSTM

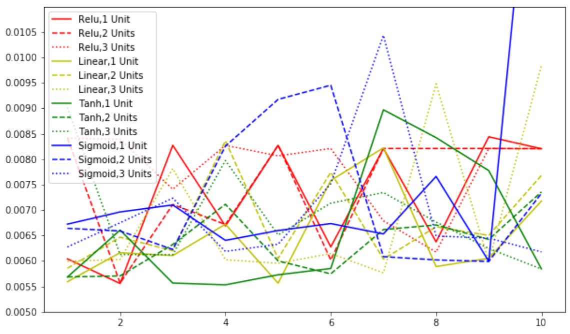


FIGURE 15.—RMSE of 120 Models

In the next phase, I explore the potential of LSTM for forecasting. To optimize LSTM's performance, I conduct experiments with various parameters, including window size, the number of LSTM units, and activation functions. There are a total of 120 models tested, and I present the best forecasting results in Figure 14, along with the RMSE values for all 120 models in Figure 15.

From the outcomes, several key conclusions can be drawn:

1. The number of LSTM units significantly impacts performance, with fewer units (N = 1 or 2) generally outperforming models with three units (N = 3).

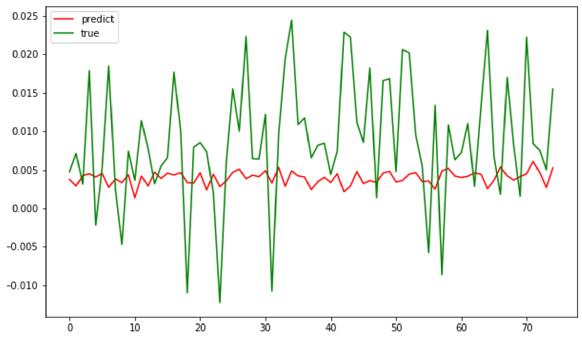


FIGURE 16.—Forecasting Result of LSTM with GA

2. Activation functions such as 'Linear' or 'Tanh' yield the best results, 'Relu' performs slightly worse, and 'Sigmoid' fares the worst.
3. The choice of window size is critical, with excessively high or low values leading to inferior results.
4. Overall, LSTM outperforms ARMA models, with the best-fitting LSTM model (Activation: Tanh, Window size = 4, Number of units = 1) not only capturing significant deviations in the original data but also tracking the underlying trends and shocks effectively.

Finally, I attempt to optimize LSTM models using Genetic Algorithm, targeting the parameters of window size and the number of units. Despite the theoretical promise of optimization, the practical results reveal a limitation of Genetic Algorithm—its computational demands. The sheer number of combinations (1021) makes it challenging to obtain the optimal result. While a larger population could theoretically guarantee finding the optimal solution, constraints on computational resources make it impractical. Therefore, a compromise is made by setting a reasonable population and increasing the number of generations, which results in a satisfactory but not necessarily optimal outcome. This highlights the trade-off between computation time and achieving the true global optimum in optimization problems.
This situation underscores one of the limitations of Genetic Algorithm, which is its high computational demand, especially in scenarios with a vast number of combinations to explore. As previously mentioned, there are a total of 1021 combinations in this case. In an ideal scenario, if the population size were higher than 1021 in the first generation, it would be possible to find the optimal result because the optimal solution, providing the highest fitness value (lowest loss), would never be eliminated.

However, practical constraints, including limited computational resources, necessitate a compromise. Typically, a reasonable population size is chosen, and the number of generations is increased to achieve a suitable result. In this specific study, I've set the population size to 50, the number of generations to 8, batch size to 10, and the number of training epochs to 10. This configuration required approximately 1 hour to yield a result. It's worth noting that obtaining the exact optimal result may require significantly more computational time and resources.

This limitation underscores the trade-off between computational efficiency and the pursuit of the true global optimum in optimization problems, a consideration that researchers and practitioners must carefully balance in their work.

5. CONCLUSION

In this comprehensive paper, I have undertaken a thorough comparison of forecasting results between ARMA models and LSTM, employing both generated ARMA data and real GDP data.

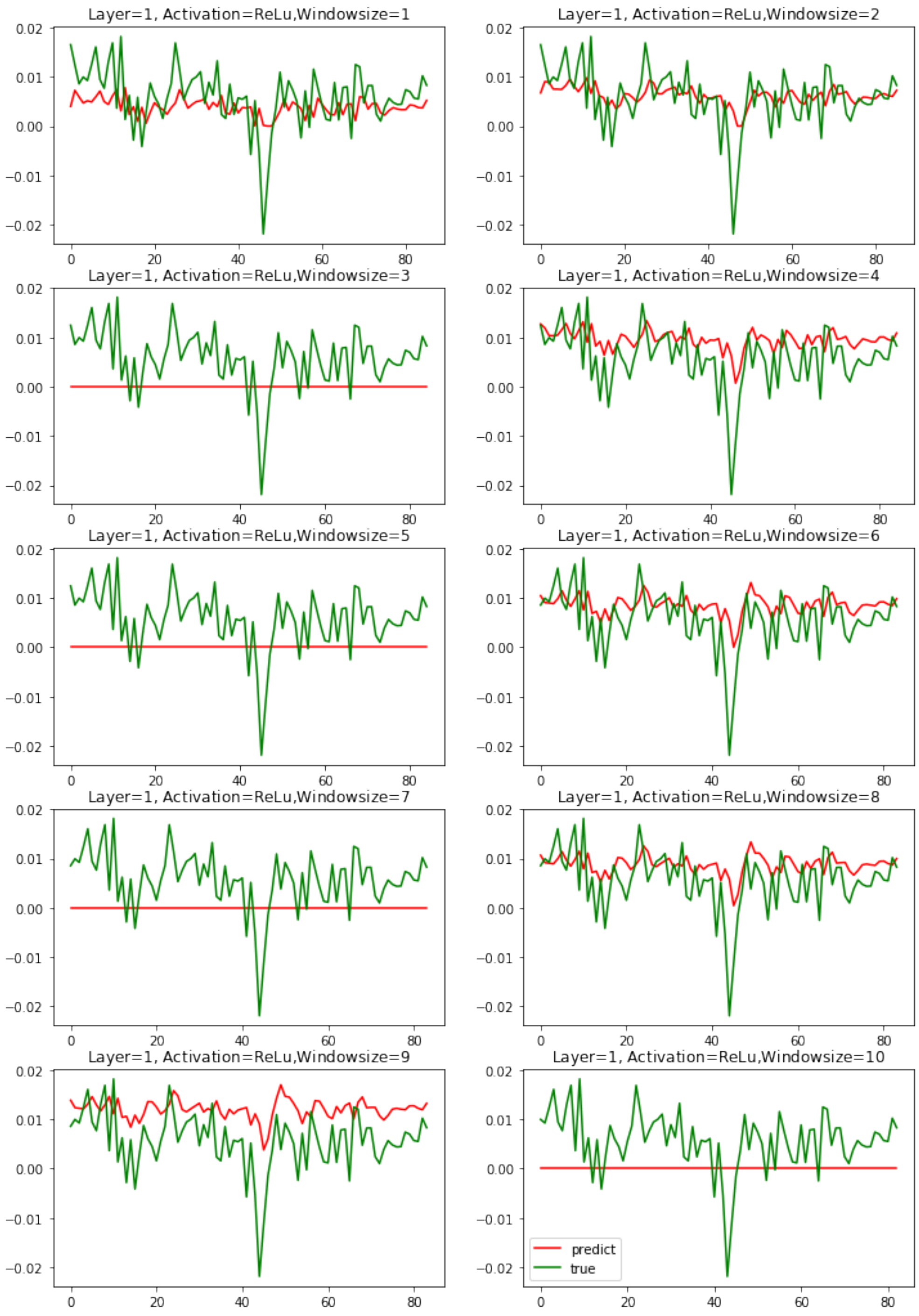
Additionally, I explored the potential of Genetic Algorithm to optimize LSTM results. Based on the outcomes obtained from this research, I have drawn the following key conclusions:

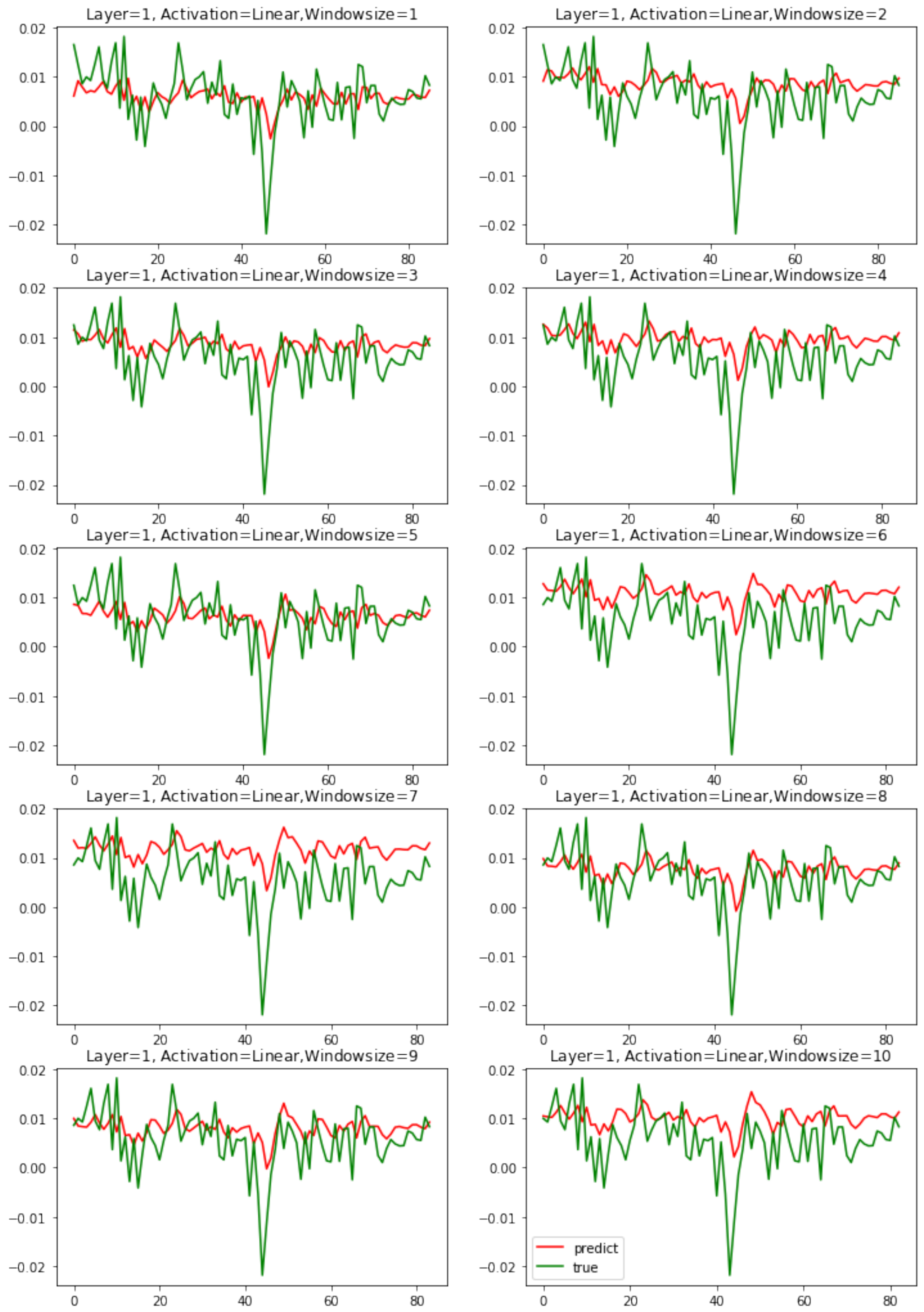
1. ARMA models are firmly grounded in mathematics and statistics, and they perform well in fitting the original data when the underlying assumptions are met. However, ARMA models fall short in the forecasting domain. Their forecasts struggle to capture significant shocks in the data and exhibit a lack of persistent fluctuation, making them less suitable for complex time series forecasting tasks.
2. LSTM, on the other hand, showcases strong forecasting capabilities. The results obtained from LSTM effectively capture large shocks in the data and maintain persistent fluctuations. Moreover, LSTM demonstrates resilience in the face of structural changes, as it conducts forecasting in a staged fashion, assigning less weight to information preceding such changes.
3. Genetic Algorithm, in theory, holds the potential to yield optimal results. However, the practical implementation of GA reveals a significant limitation—it demands substantial computational capabilities. As a result, the results obtained from GA may deviate from the true global optimum due to practical constraints on computation time and resources.

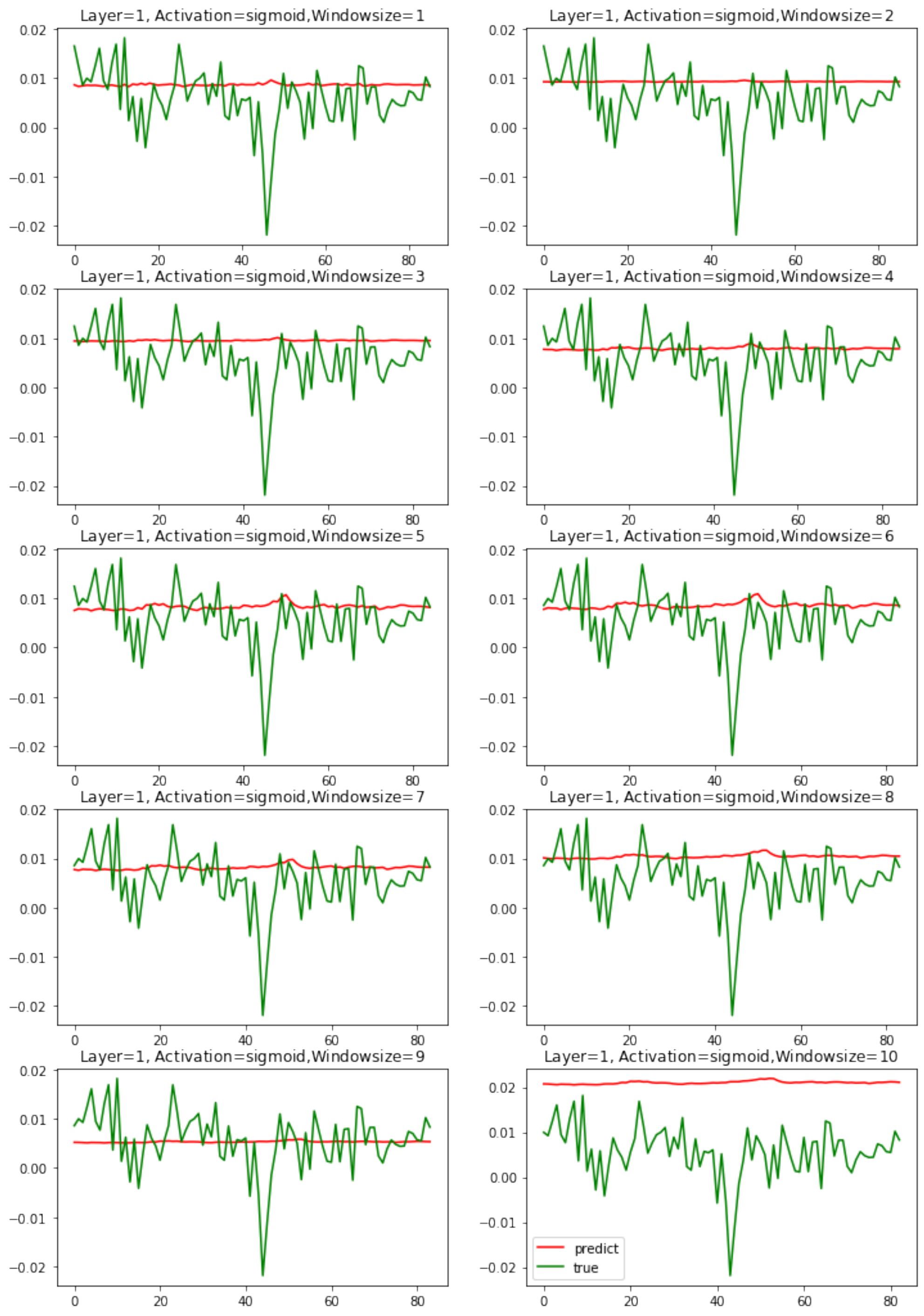
Overall, this study highlights the strengths and weaknesses of ARMA models and LSTM in the context of time series forecasting, shedding light on the potential of optimization techniques like Genetic Algorithm and emphasizing the importance of striking a balance between computational efficiency and achieving optimal solutions in real-world applications.

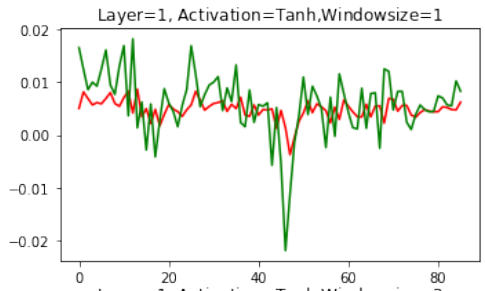
6. APPENDIX

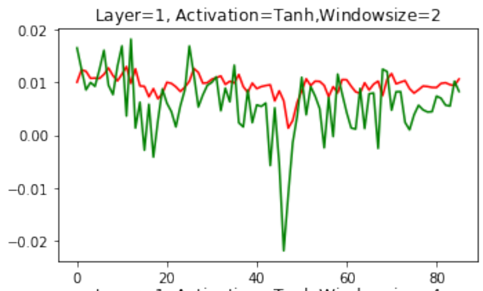
6.1. *Results of LSTM on Real GDP data*

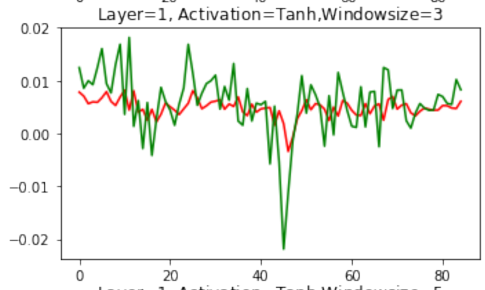


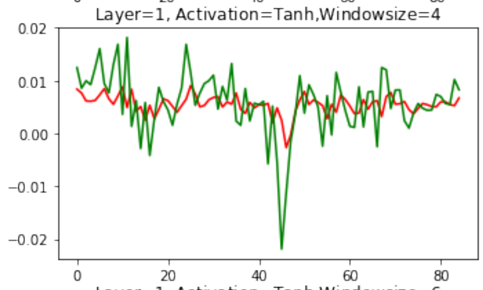

Layer=1, Activation=Linear,Windowsize=1
Layer=1, Activation=Linear,Windowsize=2
Layer=1, Activation=Linear,Windowsize=3
Layer=1, Activation=Linear,Windowsize=4
Layer=1, Activation=Linear,Windowsize=5
Layer=1, Activation=Linear,Windowsize=6
Layer=1, Activation=Linear,Windowsize=7
Layer=1, Activation=Linear,Windowsize=8
Layer=1, Activation=Linear,Windowsize=9
Layer=1, Activation=Linear,Windowsize=10
predict
true

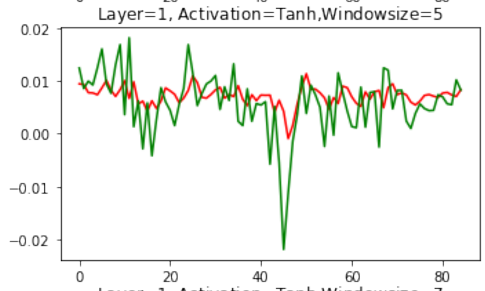

Layer=1, Activation=sigmoid,Windowsize=1
Layer=1, Activation=sigmoid,Windowsize=2
Layer=1, Activation=sigmoid,Windowsize=3
Layer=1, Activation=sigmoid,Windowsize=4
Layer=1, Activation=sigmoid,Windowsize=5
Layer=1, Activation=sigmoid,Windowsize=6
Layer=1, Activation=sigmoid,Windowsize=7
Layer=1, Activation=sigmoid,Windowsize=8
Layer=1, Activation=sigmoid,Windowsize=9
Layer=1, Activation=sigmoid,Windowsize=10
predict
true

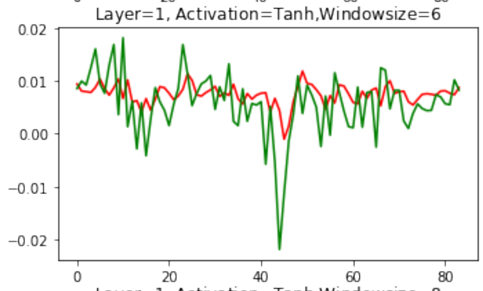

Layer=1, Activation=Tanh,Windowsize=1

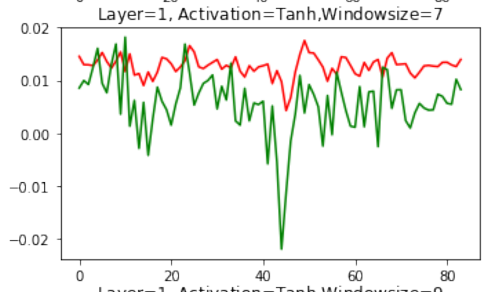

Layer=1, Activation=Tanh,Windowsize=2

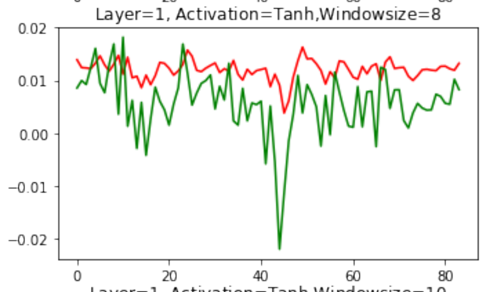

Layer=1, Activation=Tanh,Windowsize=3

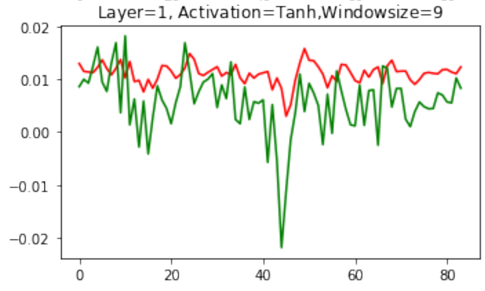

Layer=1, Activation=Tanh,Windowsize=4

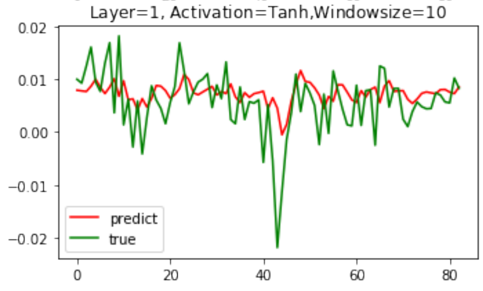

Layer=1, Activation=Tanh,Windowsize=5

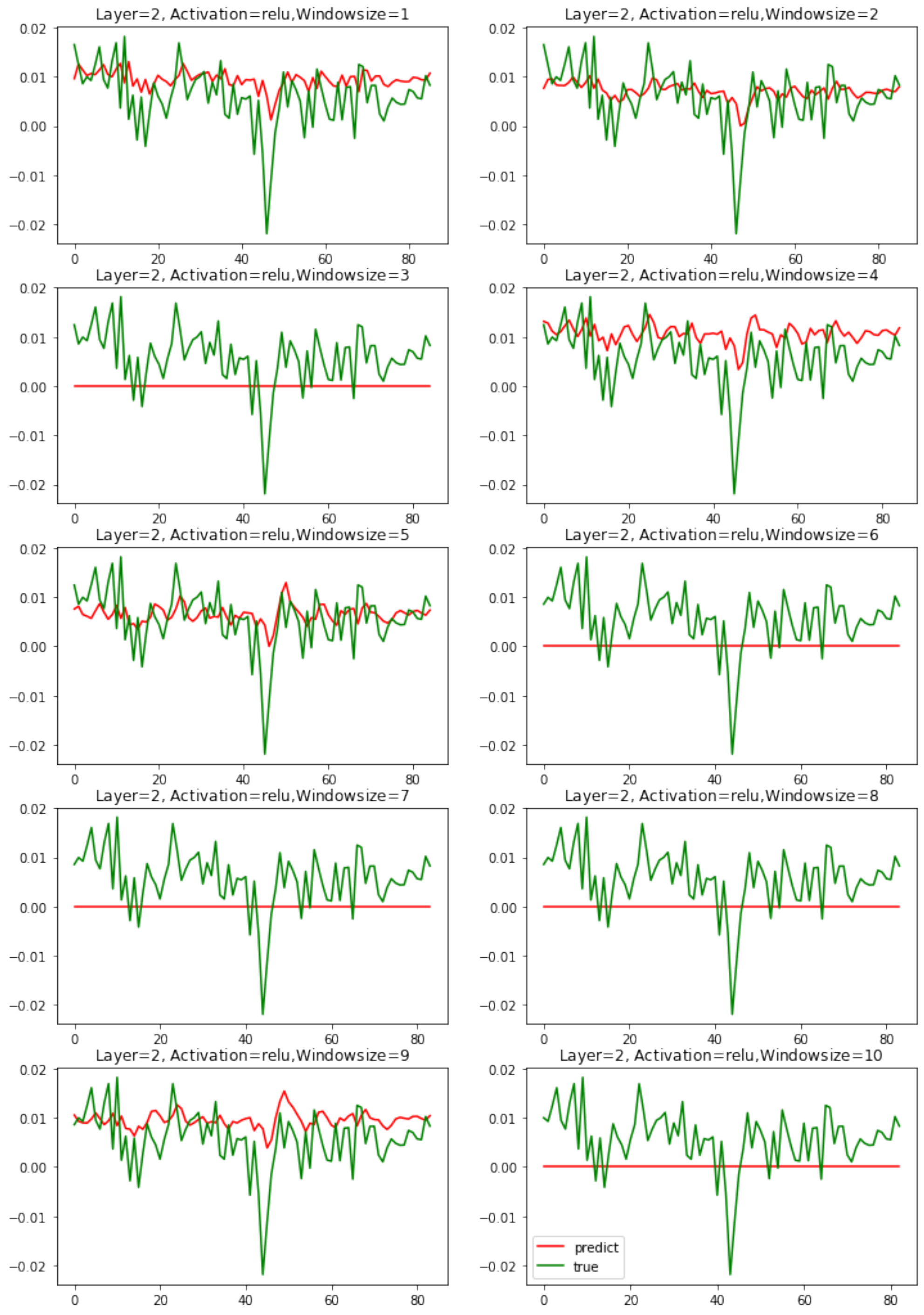

Layer=1, Activation=Tanh,Windowsize=6

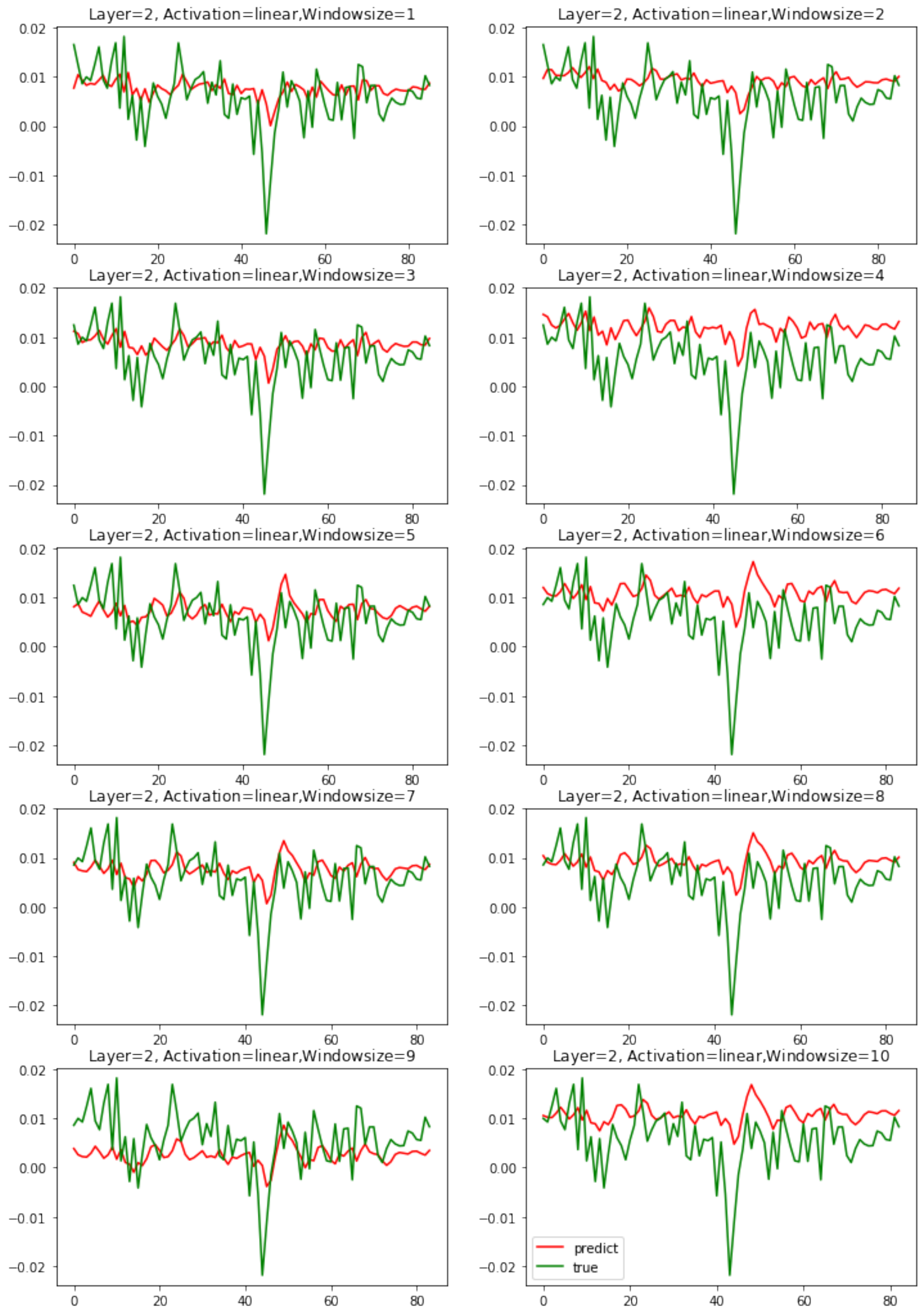

Layer=1, Activation=Tanh,Windowsize=7

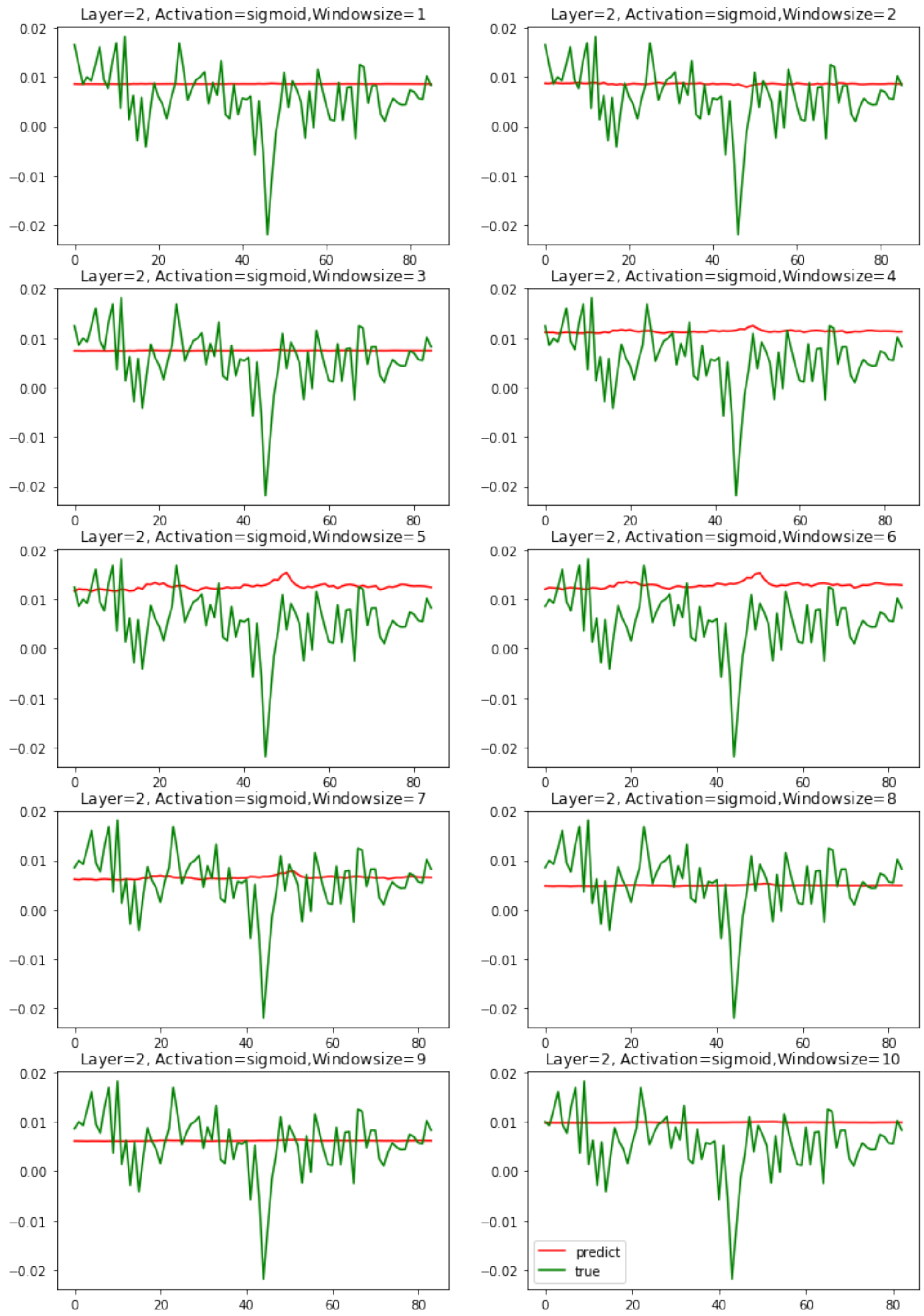

Layer=1, Activation=Tanh,Windowsize=8

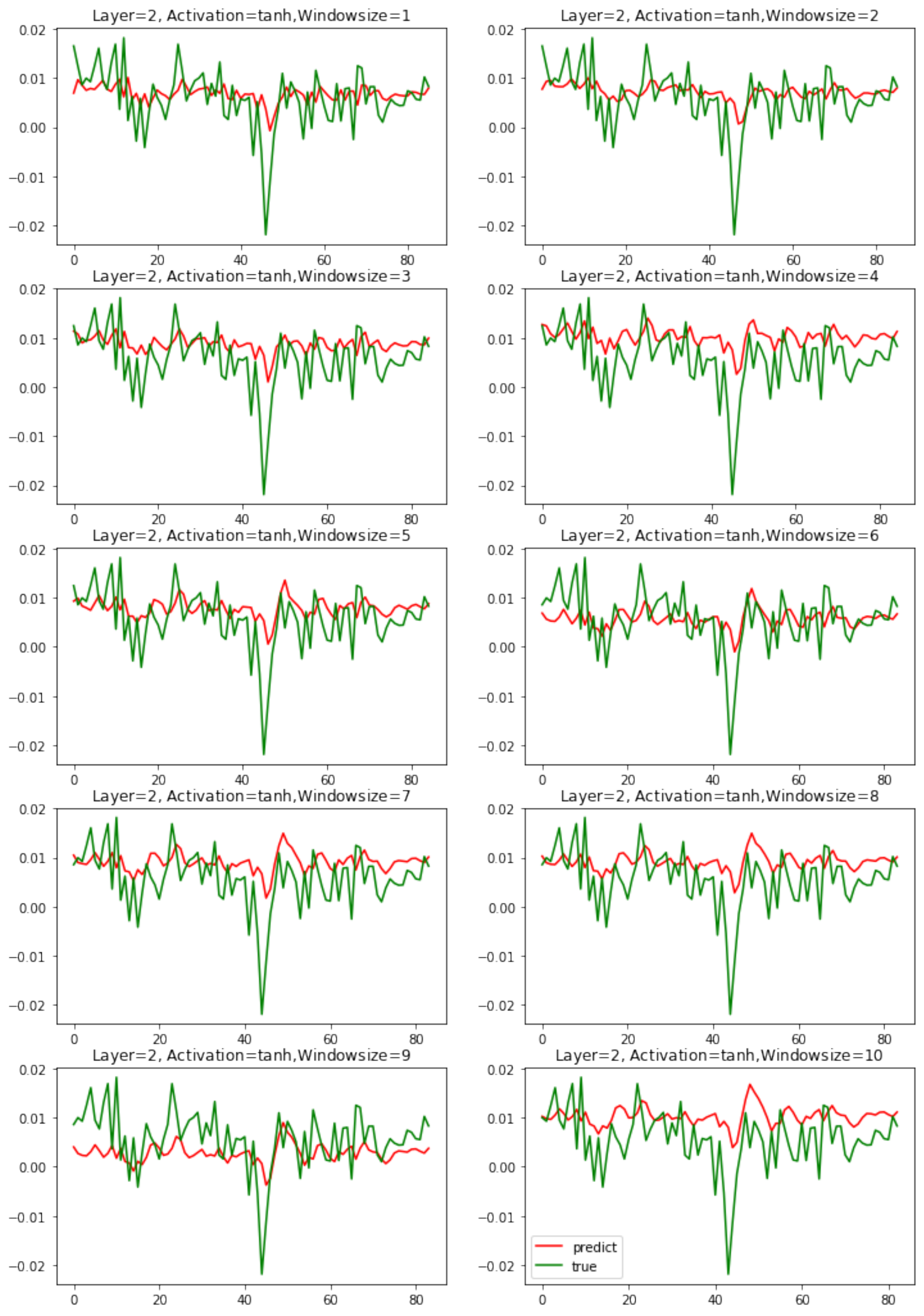

Layer=1, Activation=Tanh,Windowsize=9

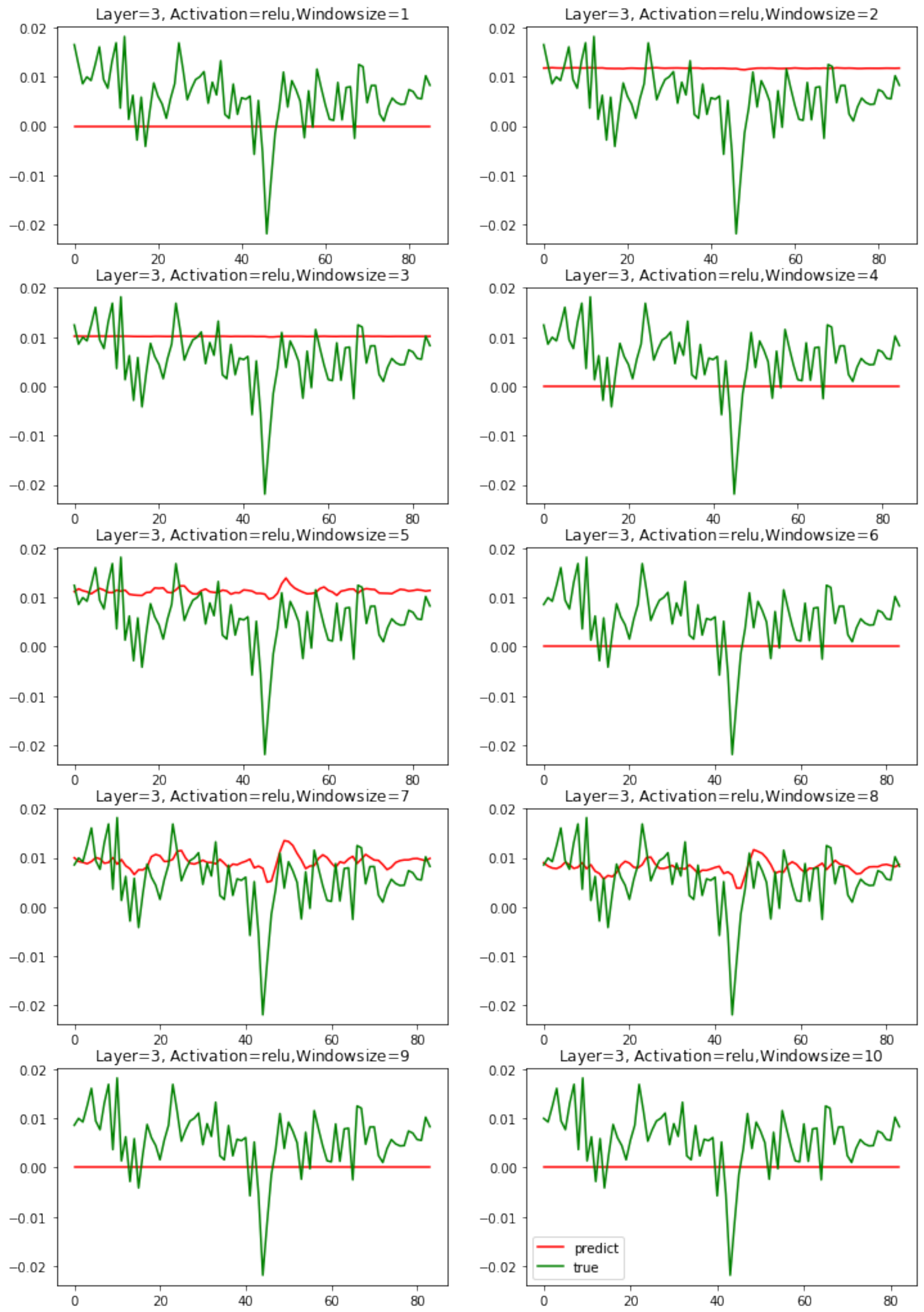

Layer=1, Activation=Tanh,Windowsize=10
predict
true

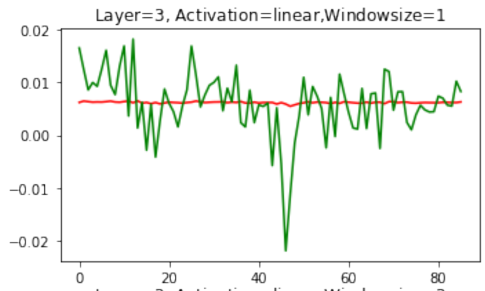

Layer=2, Activation=relu,Windowsize=1
Layer=2, Activation=relu,Windowsize=2
Layer=2, Activation=relu,Windowsize=3
Layer=2, Activation=relu,Windowsize=4
Layer=2, Activation=relu,Windowsize=5
Layer=2, Activation=relu,Windowsize=6
Layer=2, Activation=relu,Windowsize=7
Layer=2, Activation=relu,Windowsize=8
Layer=2, Activation=relu,Windowsize=9
Layer=2, Activation=relu,Windowsize=10
predict
true
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

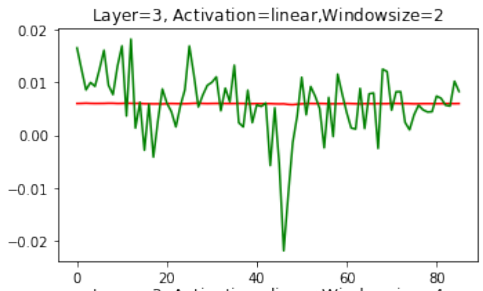

Layer=2, Activation=linear,Windowsize=1
Layer=2, Activation=linear,Windowsize=2
Layer=2, Activation=linear,Windowsize=3
Layer=2, Activation=linear,Windowsize=4
Layer=2, Activation=linear,Windowsize=5
Layer=2, Activation=linear,Windowsize=6
Layer=2, Activation=linear,Windowsize=7
Layer=2, Activation=linear,Windowsize=8
Layer=2, Activation=linear,Windowsize=9
Layer=2, Activation=linear,Windowsize=10
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80
predict
true

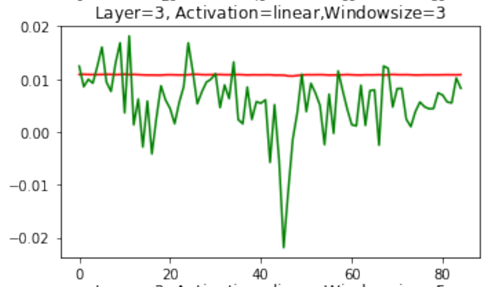

Layer=2, Activation=sigmoid,Windowsize=1
Layer=2, Activation=sigmoid,Windowsize=2
Layer=2, Activation=sigmoid,Windowsize=3
Layer=2, Activation=sigmoid,Windowsize=4
Layer=2, Activation=sigmoid,Windowsize=5
Layer=2, Activation=sigmoid,Windowsize=6
Layer=2, Activation=sigmoid,Windowsize=7
Layer=2, Activation=sigmoid,Windowsize=8
Layer=2, Activation=sigmoid,Windowsize=9
Layer=2, Activation=sigmoid,Windowsize=10
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80
predict
true

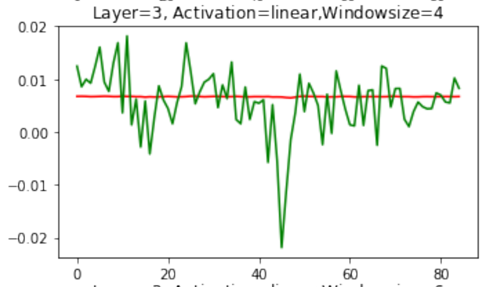

Layer=2, Activation=tanh,Windowsize=1
Layer=2, Activation=tanh,Windowsize=2
Layer=2, Activation=tanh,Windowsize=3
Layer=2, Activation=tanh,Windowsize=4
Layer=2, Activation=tanh,Windowsize=5
Layer=2, Activation=tanh,Windowsize=6
Layer=2, Activation=tanh,Windowsize=7
Layer=2, Activation=tanh,Windowsize=8
Layer=2, Activation=tanh,Windowsize=9
Layer=2, Activation=tanh,Windowsize=10
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80
predict
true

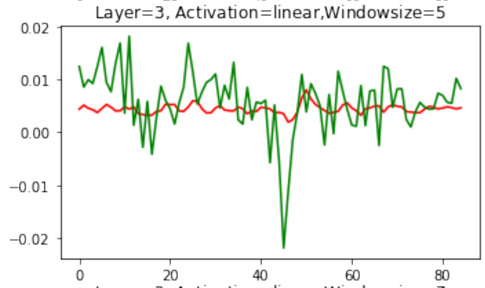

Layer=3, Activation=relu,Windowsize=1
Layer=3, Activation=relu,Windowsize=2
Layer=3, Activation=relu,Windowsize=3
Layer=3, Activation=relu,Windowsize=4
Layer=3, Activation=relu,Windowsize=5
Layer=3, Activation=relu,Windowsize=6
Layer=3, Activation=relu,Windowsize=7
Layer=3, Activation=relu,Windowsize=8
Layer=3, Activation=relu,Windowsize=9
Layer=3, Activation=relu,Windowsize=10
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80
predict
true

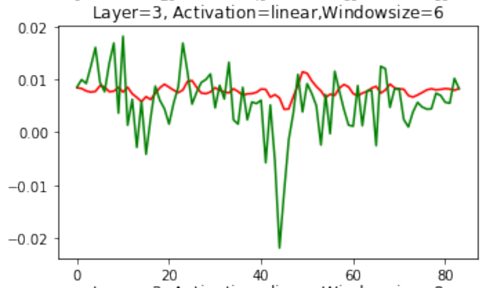

Layer=3, Activation=linear,Windowsize=1
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

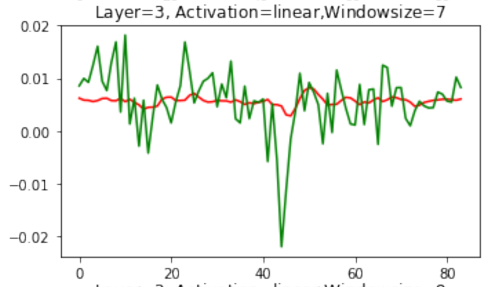

Layer=3, Activation=linear,Windowsize=2
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

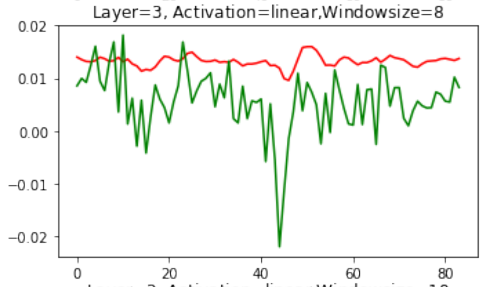

Layer=3, Activation=linear,Windowsize=3
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

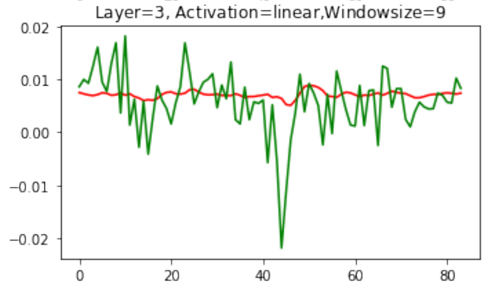

Layer=3, Activation=linear,Windowsize=4
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

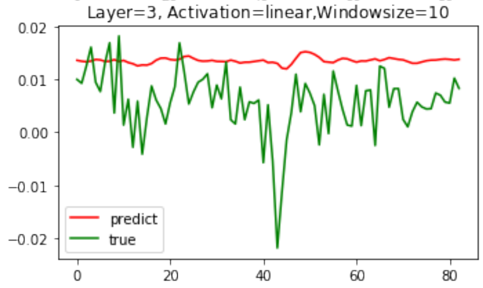

Layer=3, Activation=linear,Windowsize=5
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

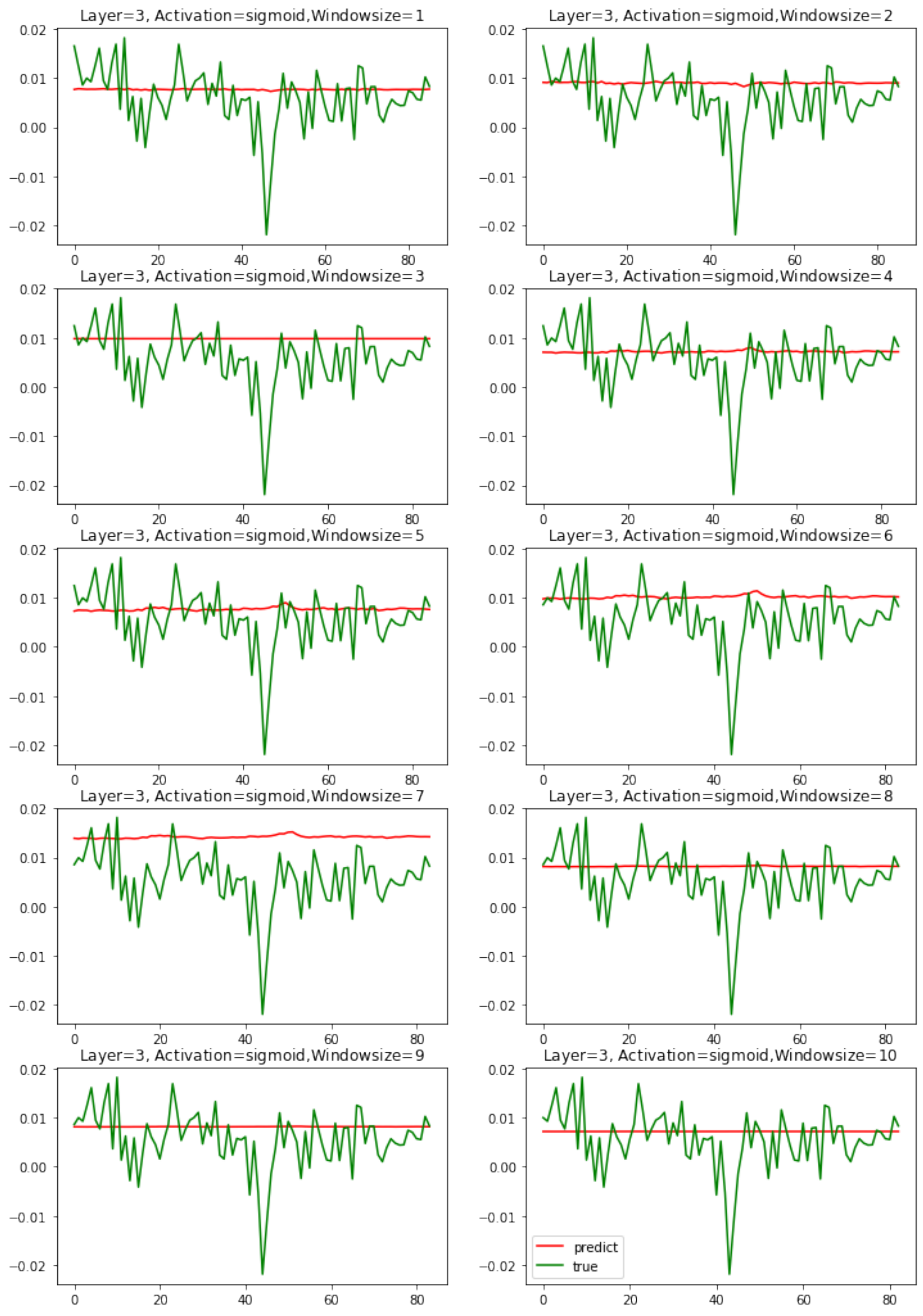

Layer=3, Activation=linear,Windowsize=6
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80

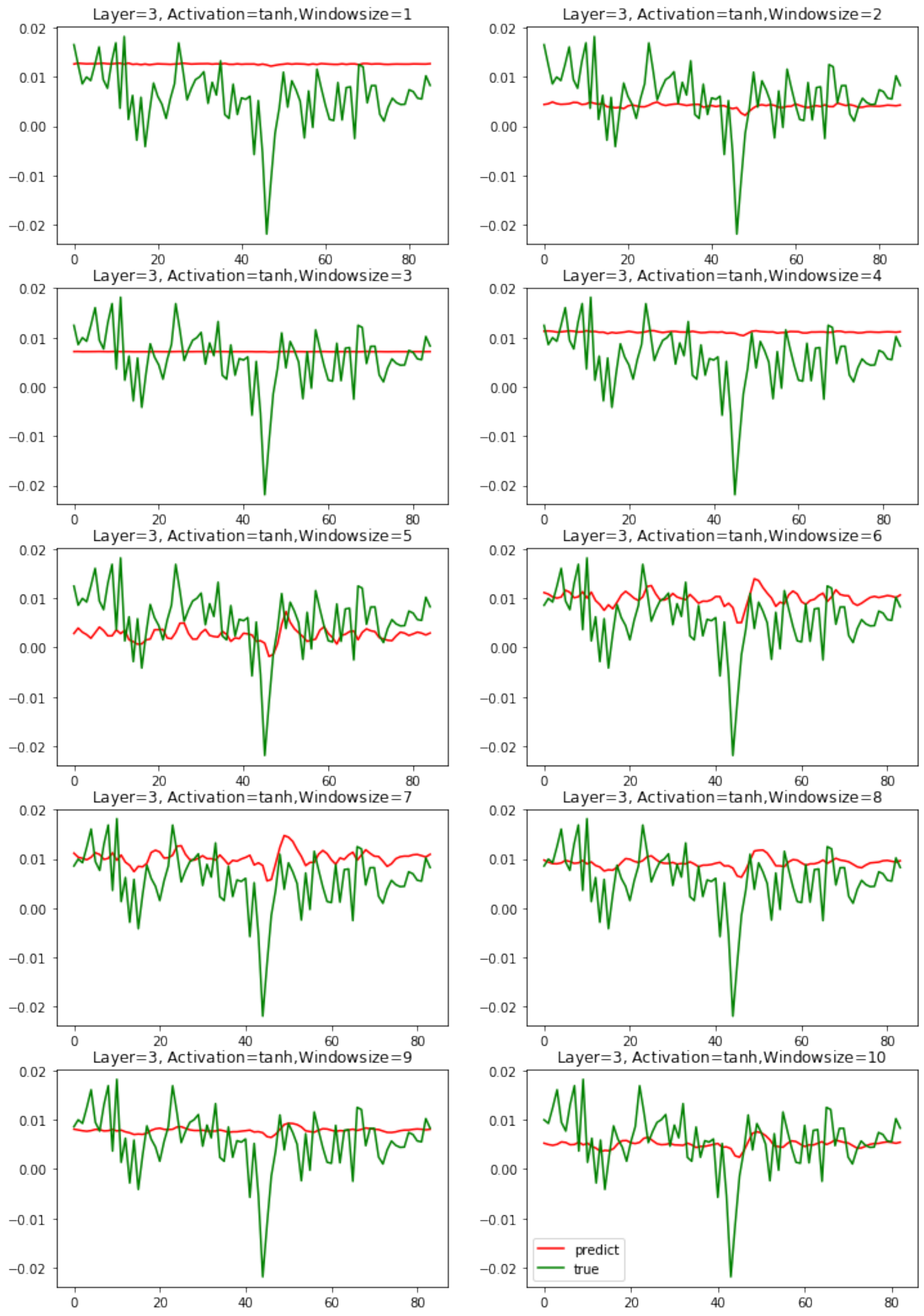

Layer=3, Activation=linear,Windowsize=7
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80


Layer=3, Activation=linear,Windowsize=8
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80


Layer=3, Activation=linear,Windowsize=9
0.02
0.01
0.00
−0.01
−0.02
0
20
40
60
80


Layer=3, Activation=linear,Windowsize=10
0.02
0.01
0.00
−0.01
−0.02
predict
true
0
20
40
60
80


Layer=3, Activation=sigmoid,Windowsize=1
Layer=3, Activation=sigmoid,Windowsize=2
Layer=3, Activation=sigmoid,Windowsize=3
Layer=3, Activation=sigmoid,Windowsize=4
Layer=3, Activation=sigmoid,Windowsize=5
Layer=3, Activation=sigmoid,Windowsize=6
Layer=3, Activation=sigmoid,Windowsize=7
Layer=3, Activation=sigmoid,Windowsize=8
Layer=3, Activation=sigmoid,Windowsize=9
Layer=3, Activation=sigmoid,Windowsize=10
predict
true


Layer=3, Activation=tanh,Windowsize=1
Layer=3, Activation=tanh,Windowsize=2
Layer=3, Activation=tanh,Windowsize=3
Layer=3, Activation=tanh,Windowsize=4
Layer=3, Activation=tanh,Windowsize=5
Layer=3, Activation=tanh,Windowsize=6
Layer=3, Activation=tanh,Windowsize=7
Layer=3, Activation=tanh,Windowsize=8
Layer=3, Activation=tanh,Windowsize=9
Layer=3, Activation=tanh,Windowsize=10
predict
true