engineering proceedings

MDPI

Proceeding Paper

Improving the Predictive Power of Historical Consistent Neural Networks †

Rockefeller Rockefeller [1,2,*], Bubacarr Bah [1,2,‡], Vukosi Marivate [3,‡] and Hans-Georg Zimmermann [4,‡]

[1] African Institute for Mathematical Sciences, Cape Town 7945, South Africa; bubacarr@aims.ac.za
[2] Department of Mathematical Sciences, Stellenbosch University, Cape Town 7945, South Africa
[3] Department of Computer Sciences, University of Pretoria, Pretoria 0028, South Africa; vukosi.marivate@cs.up.ac.za
[4] Fraunhofer Society, 200703 Munich, Germany; hans.georg.zimmermann@iis.fraunhofer.de
* Correspondence: rockefeller@aims-senegal.org
† Presented at the 8th International Conference on Time Series and Forecasting, Gran Canaria, Spain, 27–30 June 2022.
‡ The first author did most of the work as the student, which was jointly supervised by the other authors.

Abstract: The Historical Consistent Neural Networks (HCNN) are an extension of the standard Recurrent Neural Networks (RNN): they allow the modeling of highly-interacting dynamical systems across multiple time scales. HCNN do not draw any distinction between inputs and outputs, but model observables embedded in the dynamics of a large state space. In this paper, we propose to improve the predictive power of the (Vanilla) HCNN using three methods: (1) HCNN with Partial Teacher Forcing, (2) HCNN with Sparse State Transition Matrix, and (3) a Long Short Term Memory Formulation of HCNN. We investigated the effect of those long memory improvement methods on three chaotic time-series mathematically generated from the Rabinovich–Fabrikant, the Rossler System and the Lorenz system. To complement our study, we compared the accuracy of the different HCNN variants with well-known recurrent neural networks methods such as Vanilla RNN and LSTM for the same prediction tasks. Overall, our results show that the Vanilla HCNN is superior to RNN and LSTM. This is even more the case if you include the above long memory extensions (1), (2) and (3). We demonstrate that (1) and (3) are superior for the modeling of our chaotic dynamical systems. We show that for these deterministic systems, the ensembles are narrowed.

Keywords: recurrent neural networks; historical consistent neural networks; time series forecasting; chaotic dynamical systems

Citation: Rockefeller, R.; Bah, B.; Marivate, V.; Zimmermann, H.-G. Improving the Predictive Power of Historical Consistent Neural Networks. *Eng. Proc.* **2022**, *18*, 36. https://doi.org/10.3390/engproc2022018036

Academic Editors: Ignacio Rojas, Hector Pomares, Olga Valenzuela, Fernando Rojas and Luis Javier Herrera

Published: 27 June 2022

1. Introduction

Over the recent years, data-driven approaches, including deep learning techniques have played an instrumental role in the way we model, predict, and control dynamical systems [1]. Thanks to the help of modern mathematical methods, the availability of data and computational resources, Neural Networks have been increasingly used to understand complex systems (non linear and high dimensional systems) [2]. In 1989, Hornik, Stinchcombe, and White proved through the Universal Approximation theorem that Multi-layer feedforward Networks are Universal Approximators [3]. The Universal Approximation for RNN is stated in [2]. Recurrent neural networks (or RNN) are a family of Neural networks designed for processing sequential data. They are increasingly being used to understand, analyze and forecast the evolution of complex dynamical systems due to the explicit modeling of time and memory they offer [4]. RNN fulfill the universal approximation properties and allow the identification of dynamical systems in form of high dimensional, non-linear state space models [5]. Simple in architectures with sophisticated learning algorithms, they have emerged as one of the first class candidates for modeling dynamical systems [6]. The benefits they offer to deal with the typical challenges associated with forecasting in

general, make them suitable for learning non-linear dependencies from observed time series data [7].

Throughout the training process, RNN rely on external inputs, which make them suitable for the modeling of open dynamical systems. However, they assume constant environmental conditions as from present time on, which make them temporarily inconsistent [7]. HCNN do not have this inconsistency between past and future modeling. Introduced by [8], HCNN is based upon the assumption that dynamical systems must be seen in a context of large systems in which various (non-linear) dynamics interact with each other in time. HCNN do not only model the individual dynamics of interest, but also the external drivers in the same manner, by embedding them in the dynamics of a large state space [8].

This paper brings about three contributions: Firstly, we modeled three well-known chaotic dynamical systems by the use of Vanilla HCNN: namely the Lorenz, the Rabinovich–Fabrikant and the Rossler Systems. Secondly, we improved the forecast accuracy and the length of the forecast horizon of the Vanilla HCNN using three methods: HCNN with Partial Teacher Forcing, HCNN with Sparse constraint on state transition matrix, and a Long Short Term Memory Formulation of HCNN. Thirdly, we ran a comparative analysis between those different strands of HCNN and well-known deep learning neural network models such as Vanilla Recurrent Neural Networks (RNN) and long-short term memory based model (LSTM). The rest of this paper is organized as follows.

The Sections 2 and 3 provide respectively a review of the mathematical description of RNN and an architectural description of HCNN as well as its learning algorithm. In Section 4, we discussed the intuition behind and the architecture of the different HCNN improvement methods. The focus of Section 5 is on the data generation of the three chaotic dynamical systems. Section 6 shows the different results and comparative analysis between the pre-cited methods and the existing well-known recurrent neural networks namely RNN and LSTM. In Section 7, we demonstrate that our results are reproducible for different HCNN instances. Finally, we present the conclusion and future work in the Section 8.

2. Reminder of Recurrent Neural Networks for Dynamical Systems

Let us consider, as in Figure 1, a dynamical system driven by an external signal u_t.

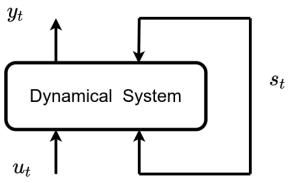

Figure 1. RNN Identification of a (folded) dynamical system using a discrete time description.

$$s_t = f(s_{t-1}, u_t) \quad (1)$$
$$y_t = g(s_t) \quad (2)$$

Let us assume that, at each time t, an output y_t is recorded. A dynamical system can be described for discrete time grids by a state space model, consisting respectively of a state transition and an output equation. The recursive Equation (1) describes the current state of the system s_t with respect to the previous state of the system s_{t-1} and the external signals u_t. The expected output y_t is computed as a function of the current state of the system (2). Key in the success of RNN, is their ability to generalized well, due to the fact that it is trained using parameter sharing [9]. Without loss of generality, we can approximate the state space model with the state transition (3) and the related output Equation (4):

$$s_t = \tanh(As_{t-1} + Bu_t) \quad (3)$$
$$y_t = Cs_t \quad (4)$$

where A, B and C are the weight matrices, respectively, for hidden-to-hidden, input-to-hidden and hidden-to-output connections. This makes a simple Recurrent Neural Network (RNN) with recurrent connections between hidden units across the whole time range [4]. By performing a finite unfolding in time, we transform the temporal equations above into the spatial architecture as shown in the Figure 2 above [8].

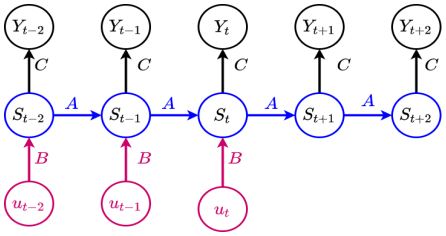

Figure 2. Vanilla RNN architecture.

The Vanilla RNN explains the dynamics observed on the y_t at each time point by splitting its complexity into two parts: the external driven part represented by the external influences u_t and the autonomous driven part (or hidden dynamics) represented by the internal states s_t. If the internal states of the system play an important role into understanding the dynamics of the observables, then an overshooting extends the autonomous part of the system several steps in the future and enable a reliable forecast of the y_t. For a given sequence of u_t and computed y_t values, we pair the corresponding observed values y_t^d and find the optimal set of shared parameters (the matrices A, B and C) by solving the following optimization problem in the Equation (5) here after [7]:

$$\min_{A,B,C} \frac{1}{T} \sum_{t=1}^{T} ||y_t - y_t^d|| \tag{5}$$

The training of the RNN can be conducted using the error-back-propagation-through-time (BPTT) algorithm. This is a natural extension of standard back-propagation that performs gradient descent on a network unfolded in time. More details on BPTT are given in [4]. However, despite its success over the past years, and especially on short term forecasts, the missing external inputs u_t in the future (which can be interpreted as a constant environment, i.e., $u_t \simeq 0$), makes a vanilla RNN temporarily inconsistent [7].

3. Historical Consistent Neural Networks

The HCNN were designed to address the temporal inconsistency in RNN. A dynamical system is often viewed in the context of large systems in which various (non-linear) dynamics interact with one another in time. However, we can only measure/observe a small subset of those variables. Therefore, HCNN reconstruct (at least part of) the hidden variables in order to understand the dynamics of the whole system [7,10]. Here the input and output variables are combined and termed as observables ($Y_t := (u_t, y_t), (Y_t \in \mathbb{R}^N)$). Together with the hidden variables, they form the state of the system at each time τ (see Figure 3) and are treated by the model in the same manner. The corresponding state transition Equation (6) and output Equation (7) are also provided below.

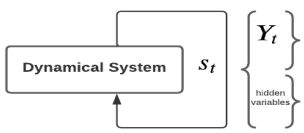

Figure 3. HCNN identification of a (folded) dynamical system using a discrete time description.

$$s_t = A\tanh(s_{t-1}) \quad (6)$$
$$Y_t = [Id, 0]s_t \quad (7)$$

The joint dynamics for all observables is characterized in the HCNN by the sequence of states s_t. The observables $(i = 1, \ldots, N)$ are arranged on the first N state neurons of s_t and followed by non-observable (hidden) variables as subsequent neurons. The object $[Id, 0]$ is a fixed matrix that reads out the observables from the state vector s_t [7]. At the initial time, the state s_0 is described as a bias/random vector and the matrix A contains the only free parameters [7].

Similar to a standard RNN, HCNN fulfils the universal approximation theorem, as highlighted in [7,10]. However, the lack of any input signals and an unfolding across the complete data makes it difficult to train in practice [8]. As proposed by [4], the models that have recurrent connections from their outputs leading back into the model may be trained with teacher forcing.

This is a procedure that emerges from the maximum likelihood criterion. It makes the best possible use of the data from the observables and therefore accelerates the training of the HCNN [2,10]. Throughout the fitting procedure, the teacher forcing mechanism introduces a hidden layer r_t that is a copy of the internal state s_t, with the exception that its first N components which correspond to the computed expected values Y_t are replaced with the observed values Y_t^d as shown in the Equation (8).

$$\begin{aligned} Y_t &= [Id, 0]s_t \\ r_t &= s_t - [Id, 0]^\top (Y_t - Y_t^d) \quad (8) \\ s_{t+1} &= A\tanh(r_t) \quad (9) \end{aligned}$$

From the temporal equations above, we can also derive its spatial representation, through the resulting network architecture of the HCNN with integrated teacher forcing mechanism as illustrated in the Figure 4 below.

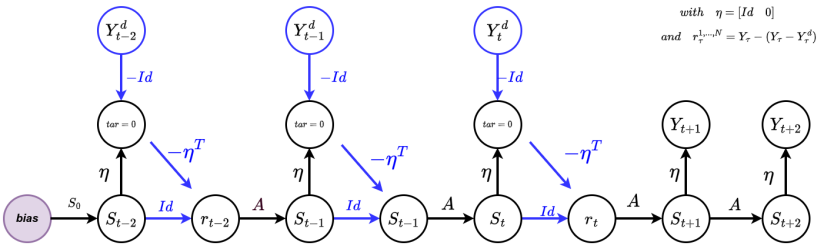

Figure 4. HCNN identification of a (folded) dynamical system using a discrete time description.

At each time t during training, the output layer of the HCNN is replaced by a cluster that is given a fixed target value of zero. This forces the HCNN to create the expected values Y_t at each time t, to compensate for the negative observed values $-Y_t$ coming from the top node [8]. The content of that cluster, i.e., $(Y_t - Y_t^d)$, with a minus symbol is transferred to the upper part (the first N neurons) of the hidden layer r_t. Furthermore, a copy of the state s_t is also transferred to the intermediate hidden layer r_t on a component-by-component basis. As a result of that, the expected values Y_t on the first N components of the hidden layer r_t are replaced by the observed values [8] and the subsequent state s_{t+1} is computed using the state transition Equation (9).

4. Long-Term Memory Improvement Methods

To improve the long-term memory of the Vanilla HCNN model, three different improvement methods have been designed: HCNN with Partial Teacher Forcing, with Large

Sparse State Transition Matrices and a Long Short-Term Memory Formulation. The intuitions behind each of the methods are shown below.

4.1. HCNN with Partial Teacher Forcing

To enforce the long-term learning of the HCNN, we endow the output layers of the HCNN with a dropout filter, guided by a probability, as illustrated in the Equation (10).

$$\text{dropout}_{(p)}(x_i) = \begin{cases} 0 & \text{if dropout}_{(p)} \\ x_i & \text{if no dropout}_{(1-p)} \end{cases} \tag{10}$$

At time t, when the filter is activated (which means for a probability p), the HCNN randomly suppress elements in the time series that come from the cluster containing the difference between expectations and observations $(Y_t - Y_t^d)$. Thus, in the upper part (the first N components) of the r_t vector, the network is enforced to replace the observations with its internal expectations. The architecture is represented in the Figure 5 below.

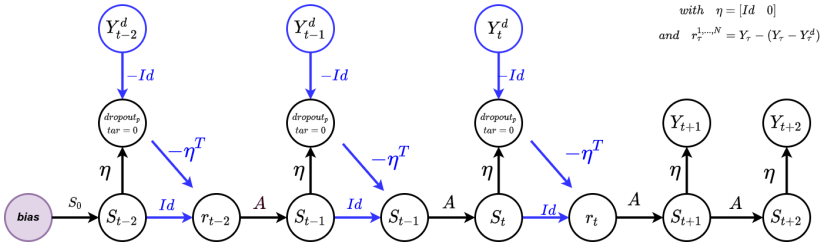

Figure 5. HCNN architecture with partial teacher forcing mechanism.

4.2. HCNN with Large Sparse State Transition Matrix

HCNN may often use large state vectors to model large dynamical systems (number of observables > 100). During training time, the iteration with a fully connected state transition matrix A could cause an information overload, leading to two risks:

- The matrix–vector computation between A and $\tanh(s_t)$, which includes the addition of randomly generated (and learned) scalar values will likely blow up to infinity (∞).
- The superposition of additional information brought in by the large dimensionality of A could destroy the longer memory information acquired throughout.

In order to overcome that, we can choose to set the transition matrix A sparse to a chosen degree. As a result of that, the spread of the information peak through the network is damped by the sparsity too. Another approach, as proposed by [7], consists of a heuristic approach that represents the sparsity of A, as inversely proportional to the state dimensionality of the system, as shown in the Equation (11) below:

$$\text{Sparsity (A)} \quad = \min(1, \frac{50}{\dim(s)}) \tag{11}$$

4.3. HCNN with LSTM Formulation

The third approach to improve the long-term memory of the vanilla HCNN is through an exponential smoothing embedding of the HCNN with a learnable diagonal matrix [11], subject to the following constraints $0 \leq D_{ii} \leq 1$. The resulting state transition Equation (12) and output Equation (13) are provided below:

$$\begin{aligned} s_t &= (\text{Id} - D)s'_{t-1} + DA\tanh(s'_{t-1}) \\ &= s'_{t-1} + D(A\tanh(s'_{t-1}) - s'_{t-1}) \qquad (12) \\ Y_t &= [Id, 0]s_t \qquad (13) \end{aligned}$$

where $s'_{t-1} = TeacherForcing(s_{t-1})$

Built upon the ideas of the LSTM formulation of RNN [12], the resulting architecture of the LSTM formulation of HCNN is provided in the Figure 6 below.

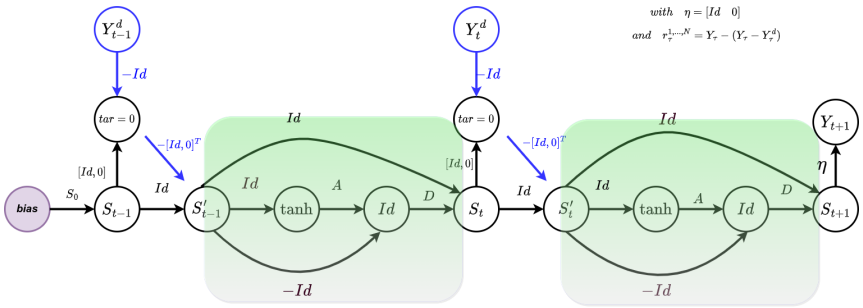

Figure 6. Architectural description of HCNN with an LSTM Formulation.

The next section will focus on the different dynamical systems that will be used to generate the data for the fitting procedure of the HCNN models.

5. Experimental Setup

The experiments we carried out in this work aimed at forecasting the dynamics of three fully observable, chaotic and deterministic systems, namely the Rabinovich–Fabrikant, Rossler System and the Lorenz system. Their chaotic properties come from the fact that they are very sensitive to their initial conditions, which smallest changes completely modify the respective trajectories. They are also well known to show aperiodic behaviour which is apparently random and unpredictable [1,13].

5.1. The Rabinovich–Fabrikant System

The Rabinovich–Fabrikant system is represented by the system of Equation (14) below:

$$\begin{cases} \frac{dx}{dt} & = y(z - 1 + x^2) + \gamma x \\ \frac{dy}{dt} & = x(3z + 1 - x^2) + \gamma y \\ \frac{dz}{dt} & = -2z(\alpha + xy) \end{cases} \tag{14}$$

where we set $\alpha = 0.2, \gamma = 0.1$. Introduced by Mikhail Rabinovich and Anatoly Fabrikant, the set of equations describes the stochasticity arising from the modulation instability in a non-equilibrium dissipative medium [14]. The corresponding attractor is shown in the Figure 7 below.

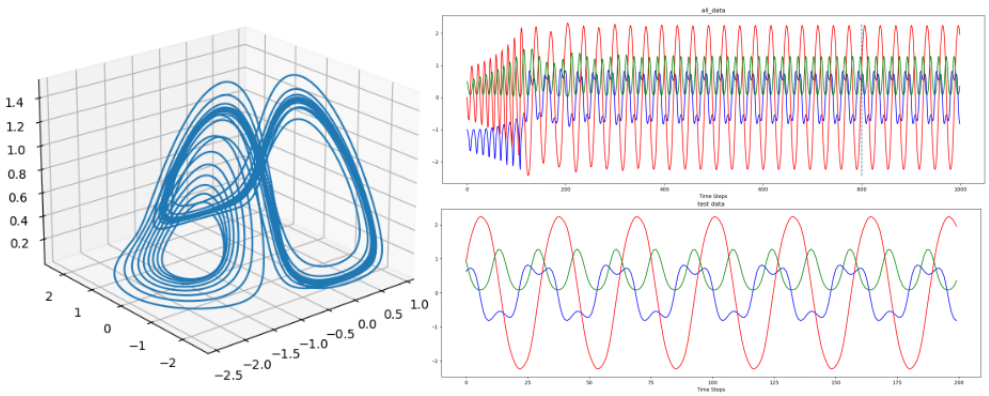

Figure 7. The Rabinovich–Fabrikant attractor and the corresponding time series, split into training and test data.

5.2. The Rossler System

The Rossler system is represented by the system of Equation (15) below:

$$\begin{cases} \frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + ay \\ \frac{dz}{dt} &= b + z(x - c) \end{cases} \tag{15}$$

where we set $a = 0.2$, $b = 0.2$ and $c = 6.3$. Studied first by Otto Rössler in the 1970s, these non-linear ordinary differential equations define a continuous-time dynamical system that exhibits chaotic dynamics associated with the fractal properties as it is shown by the corresponding attractor in the Figure 8 below [15].

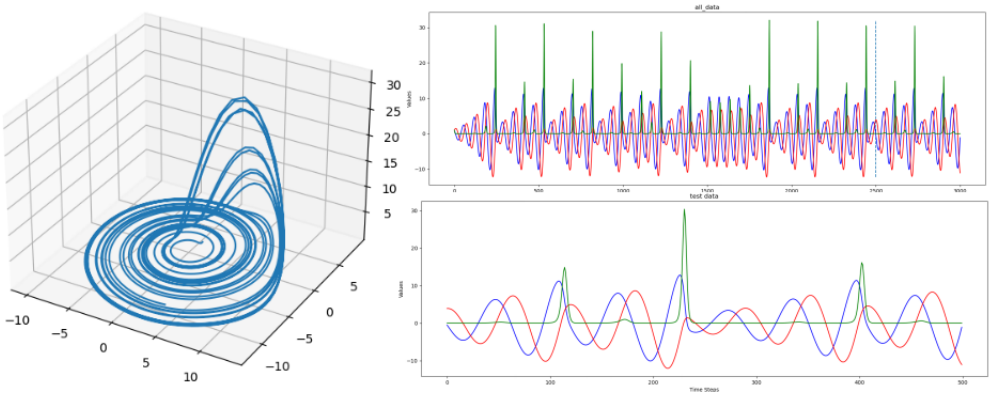

Figure 8. The Rossler attractor and the corresponding time series, split into training and test data.

5.3. The Lorenz System

The Lorenz system is represented by the system of Equation (16) below:

$$\begin{cases} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= (\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{cases} \tag{16}$$

where we set $\rho = 10$, $\sigma = 28$ and $\beta = 2.667$. The equations above describe the rate of change of three quantities, namely the rate of convection, the horizontal temperature variation and the vertical temperature variation. First studied by Edward Lorenz, the Lorenz system is a simplified mathematical model for the atmospheric convection [16,17]. The graphical representation of its attractor is provided in the Figure 9 below.

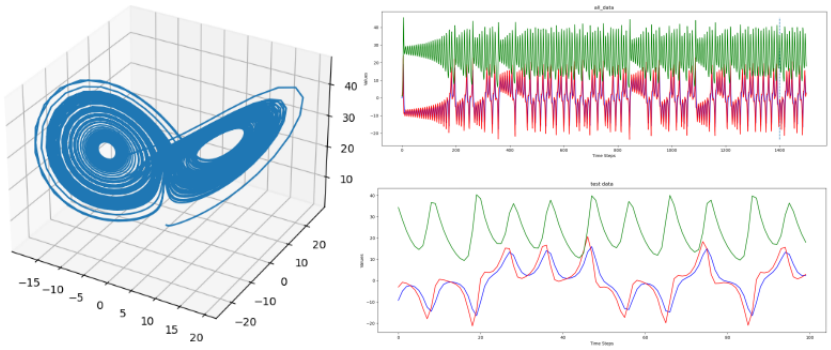

Figure 9. The Lorenz attractor and the corresponding time series, split into training and test data.

5.4. Traning Strategies

We solved each of the systems above by numerical approximation, with the configurations summarized in the Table 1 below.

Table 1. Configuration of the training data for each of the chaotic dynamical systems.

System	Initial Conditions	Sample Size	Step Size	Truncation Parameter	Training Size	Test Size
Rossler	$(1, 1, 1)$	10,000	0.01	5	1800	200
Lorenz	$(0, 1, 1.05)$	12,000	0.01	8	1400	100
Rab-Fabrikant	$(-1, 0, 0.5)$	20,000	0.01	10	1800	200

Here, the truncation parameter refers to the down-sampling that has been applied on the initial sample size. On the Lorenz time series for instance, we recorded values only every 8th time step and obtained a new sample size of 2000 observations, which has been divided into 1400 for the training and 100 observations for the forecast period as it is shown in the Figure 9 above. The truncation parameter was chosen carefully to make sure that for each system, from a graphical view point, the shapes of the corresponding attractors are preserved as exemplified in the Figures 7–9 above.

The different HCNN models were instantiated with 20 variables. Three accounting for the observables and 17 hidden variables, modeled in the same manner, to explain the dynamics of the different chaotic dynamical systems at hand. For the HCNN model with Sparse constraints, we chose the state dimensionality value as 100 ($\dim(s) = 100$), which implies the transition matrix A will be half sparse as stated in the Equation (11) above. For The HCNN with Partial Teacher Forcing, the dropout filter was set with a incremental probability to ensure that the dropout probability will reach 25% at the end of the training. The diagonal matrix D of the LSTM Formulation of HCNN was constrained between 0 and 1 at each stage of the training. For comparison purpose, we also instantiated a Recurrent Neural Network model and an LSTM formulation of it with 20 hidden states each.

5.5. Evaluation Metrics

As evaluation metric, we chose the Logarithm of Hyperbolic Cosine as our Loss Function, represented by the equation below:

$$\text{LogCosh Loss} \quad = \quad \frac{1}{T}\sum_{t=1}^{T}\frac{1}{p}\ln\cosh\left(p\left(Y_t - Y_t^d\right)\right) \tag{17}$$

6. Results and Analysis

The different models were trained and the results are shown in the different plots below.

6.1. On the Rabinovich-Fabrikant System

The plots below consist of both the actual observations and the predicted values of the three trajectories along the forecast period for each of the models as shown in pairs below the Vanilla HCNN and Vanilla RNN models (Figure 10), the HCNN with partial Teacher Forcing and with Sparse Constraints models (Figure 11), the LSTM Formulation of HCNN and of RNN models (Figure 12).

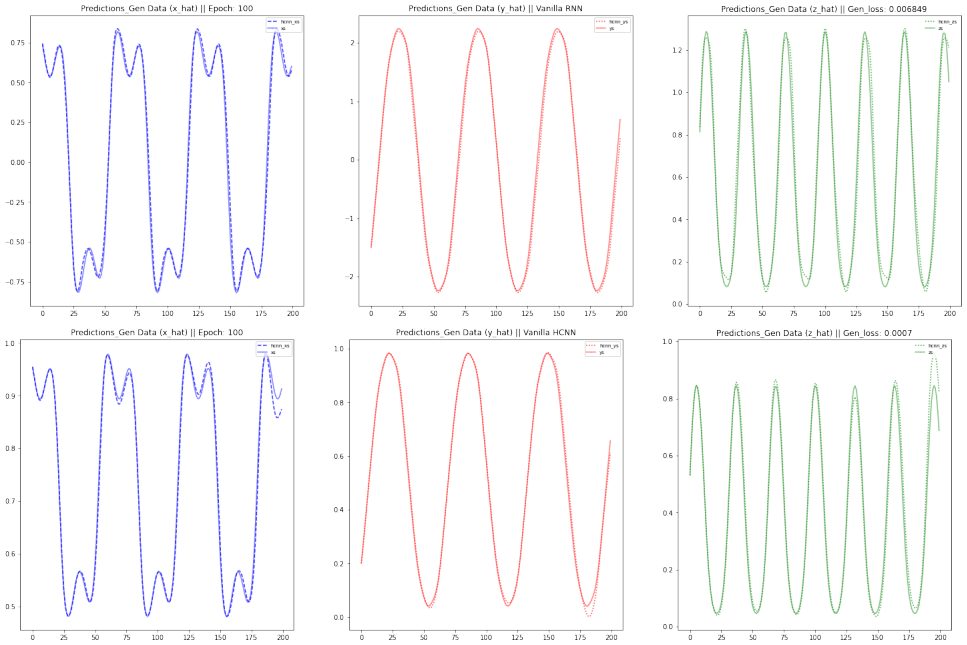

Figure 10. Generalization: Vanilla HCNN (**top**) and Vanilla RNN (**bottom**).

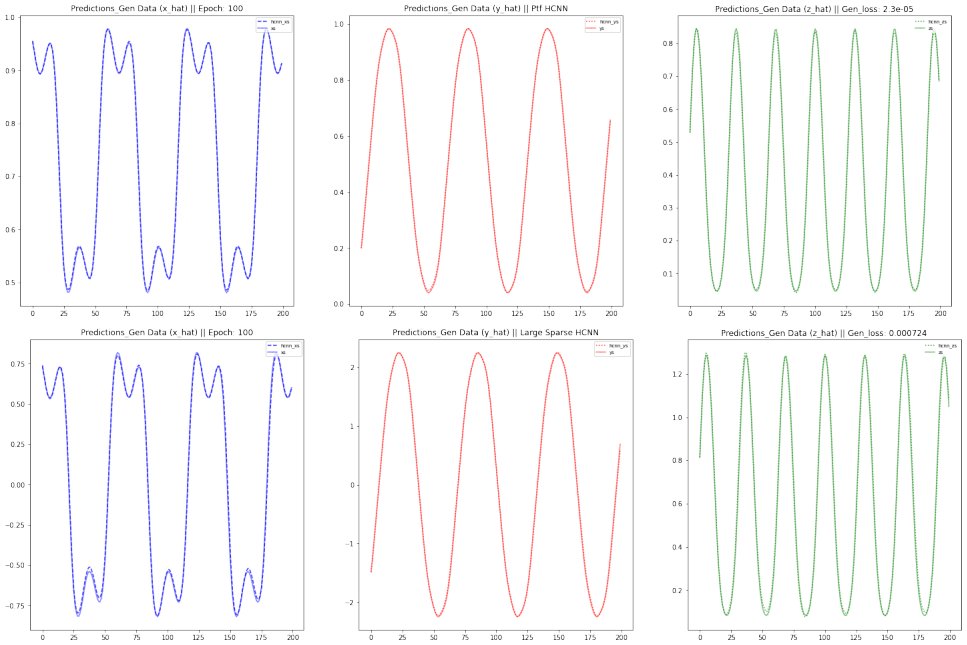

Figure 11. Generalization: HCNN pTF (**top**) and HCNN Large Sparse (**bottom**).

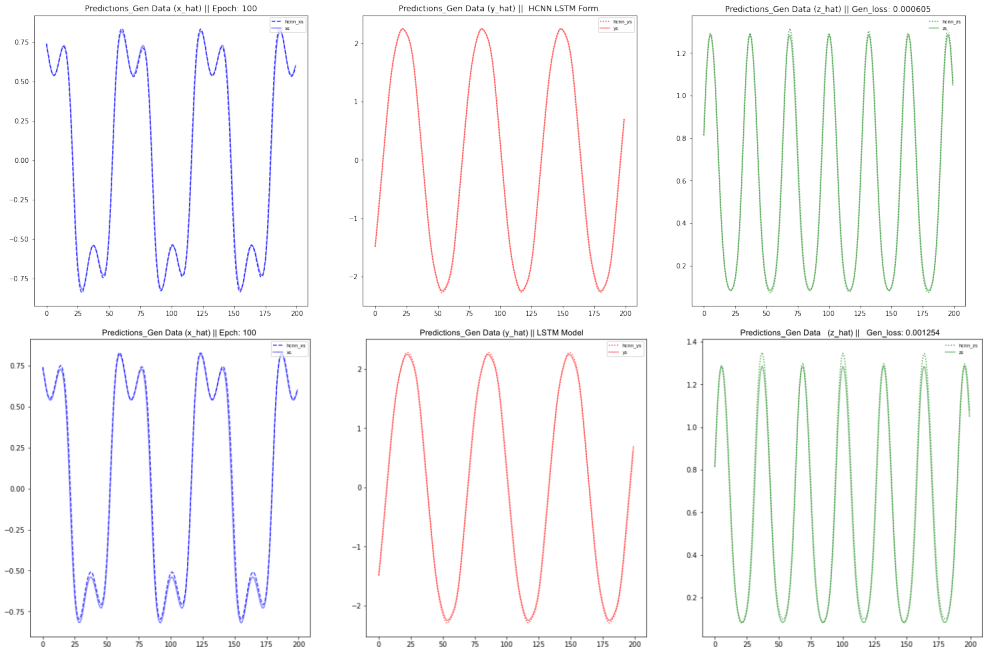

Figure 12. Generalization: HCNN with LSTM Form (**top**) and LSTM Based Model (**bottom**).

6.2. On the Rossler System

The plots below consist of both the actual observations and the predicted values of the three trajectories along the forecast period for each of the models as shown in pairs below the Vanilla HCNN and Vanilla RNN models (Figure 13), the HCNN with partial Teacher Forcing and with Sparse Constraints models (Figure 14), the LSTM Formulation of HCNN and of RNN models (Figure 15).

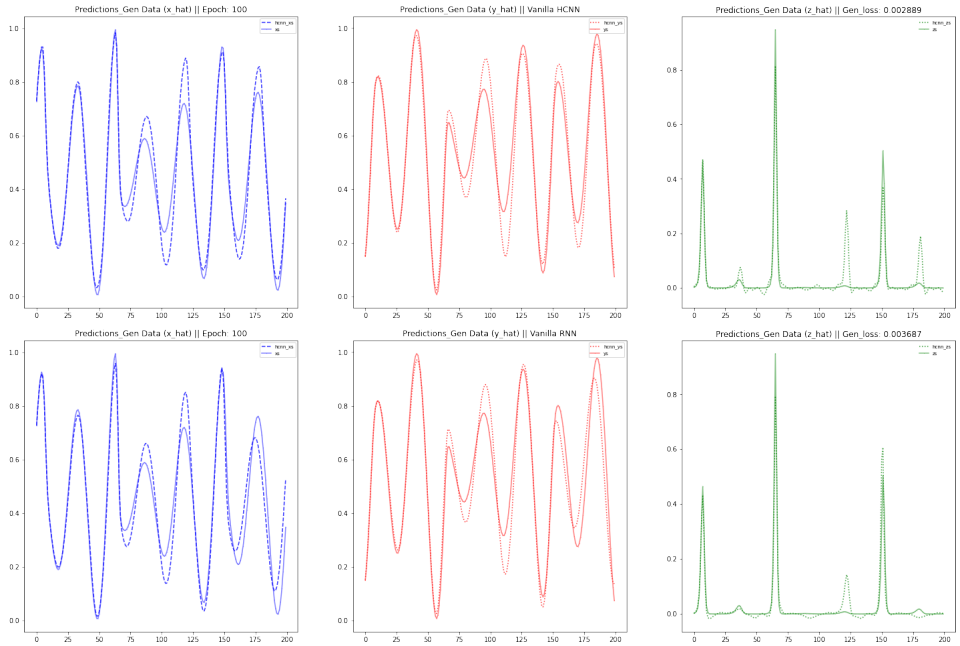

Figure 13. Generalization: Vanilla HCNN (**top**) and Vanilla RNN (**bottom**).

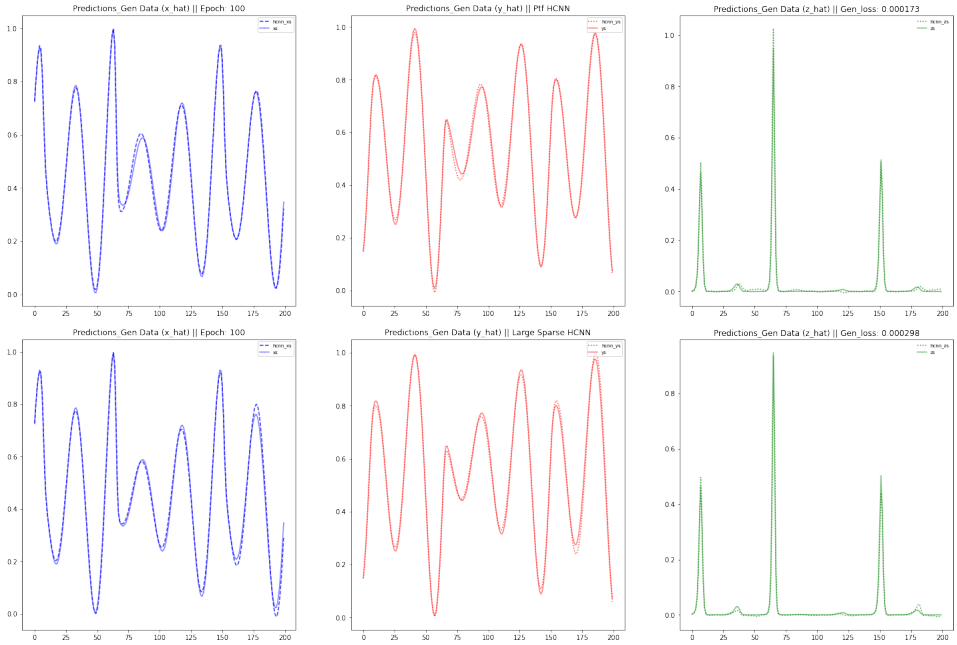

Figure 14. Generalization: HCNN pTF (**top**) and HCNN Large Sparse (**bottom**).

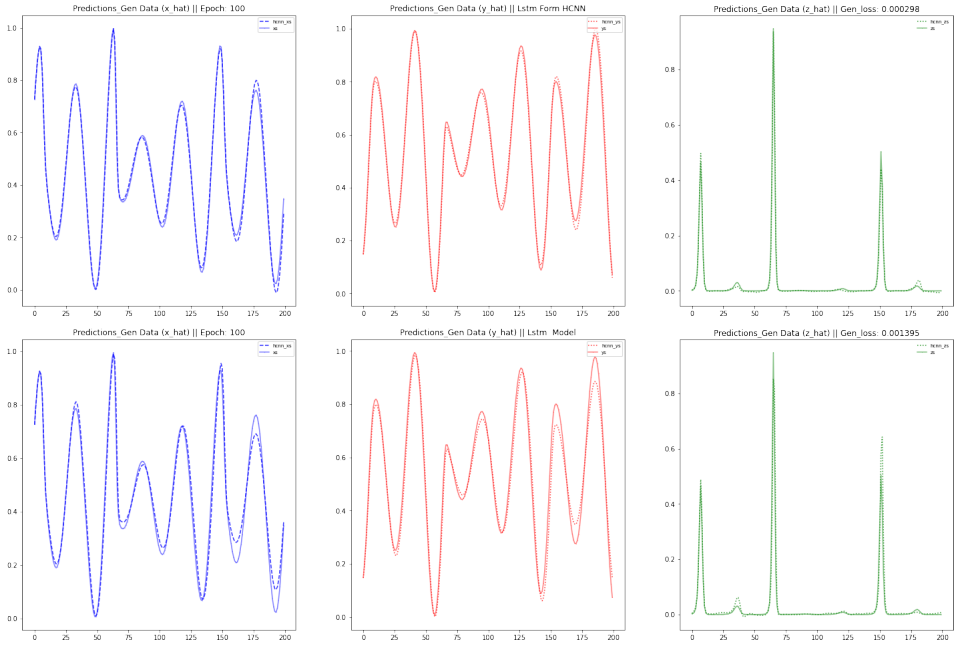

Figure 15. Generalization: HCNN with LSTM Form (**top**) and LSTM Based Model (**bottom**).

6.3. On the Lorenz System

The plots below consist of both the actual observations and the predicted values of the three trajectories along the forecast period for each of the models as shown in pairs below the Vanilla HCNN and Vanilla RNN models (Figure 16), the HCNN with partial Teacher Forcing and with Sparse Constraints models (Figure 17), the LSTM Formulation of HCNN and of RNN models (Figure 18).

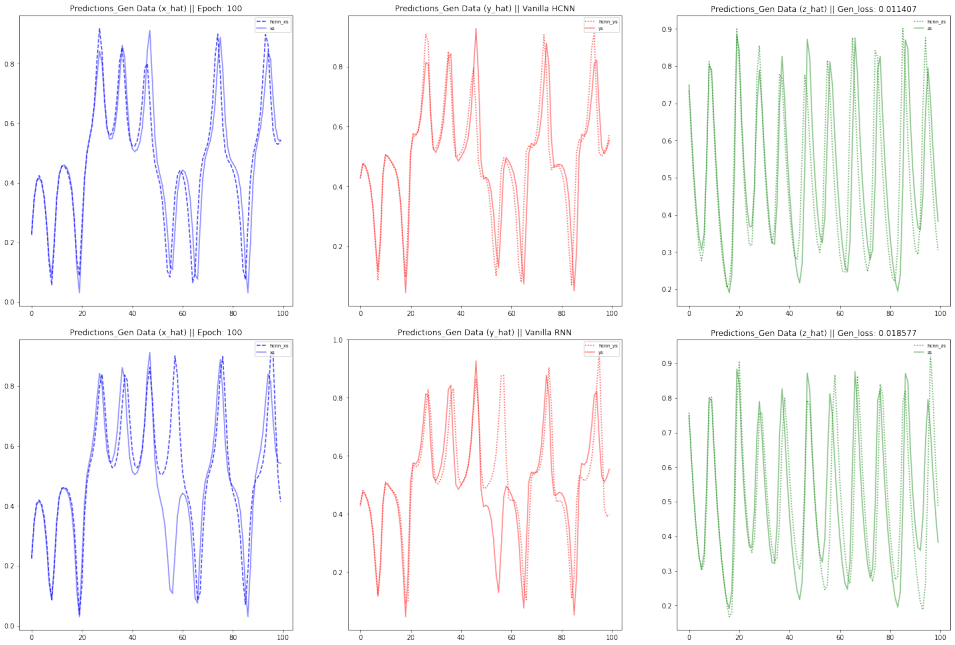

Figure 16. Generalization: Vanilla HCNN (**top**) and Vanilla RNN (**bottom**).

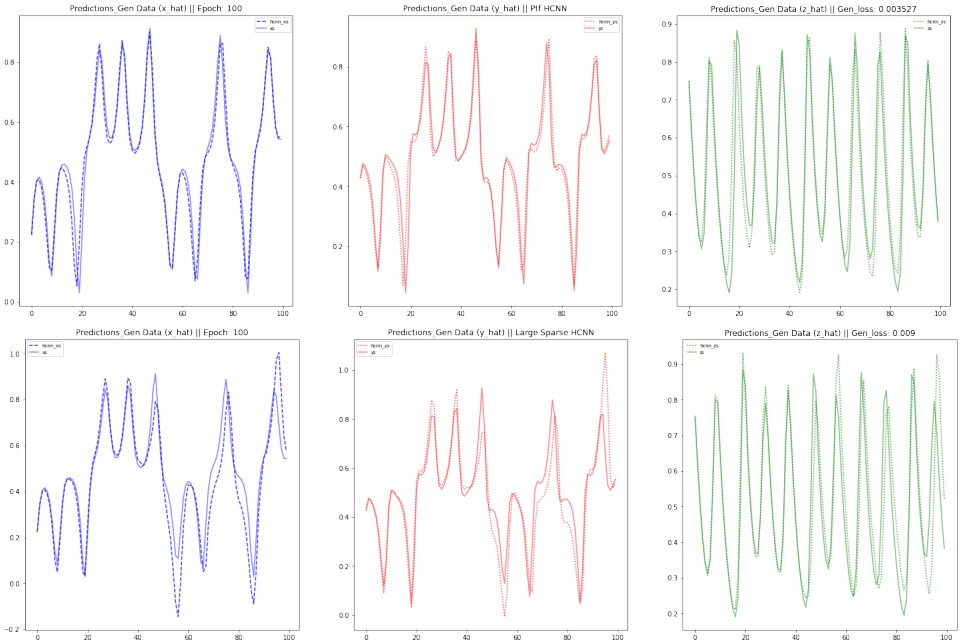

Figure 17. Generalization: HCNN pTF (**top**) and HCNN Large Sparse (**bottom**).

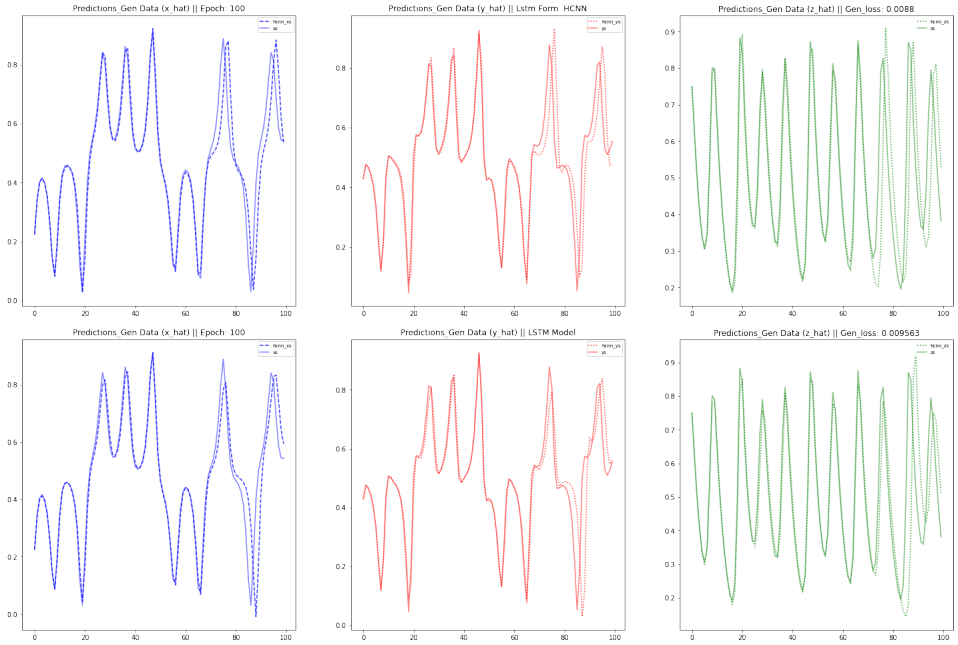

Figure 18. Generalization: HCNN with LSTM Form (**top**) and LSTM Based Model (**bottom**).

For each of the experiments, HCNN with partial Teacher Forcing is the superior method. To summarize our findings, we grouped the results for every model in the Table 2 below.

Table 2. Forecast error made by the different models on the different chaotic dynamical systems.

Model	**Rabi-Fabrikant** $(\times 10^{-3})$	**Rossler** $(\times 10^{-3})$	**Lorenz** $(\times 10^{-3})$
Vanilla HCNN	0.7	2.88	3.52
Vanilla RNN	6.84	3.68	18.5
HCNN p-Teacher Forcing	**0.023**	**0.17**	**0.173**
HCNN Lar-Sparse Tran. Mat	0.072	0.29	9.01
HCNN with LSTM Form.	0.6	0.22	8.07
RNN with LSTM Form.	1.25	1.39	9.56

As a general remark, the vanilla HCNN model outperformed the Vanilla RNN model on the three forecasting exercises. Out of the three HCNN improvement methods, not only the HCNN with partial Teacher Forcing is the superior one, it has also improved the forecast error made by the existing well-known RNN and LSTM models (provided by the PyTorch library).

7. Ensemble Computations

To ensure that our results are reproducible, we instantiate an ensemble of HCNN with 10 members and train them simultaneously to forecast the dynamics of the Lorenz system. The Figure 19 below consists of both the actual observations and the 10 instances of the HCNN model, and median of the ensembles.

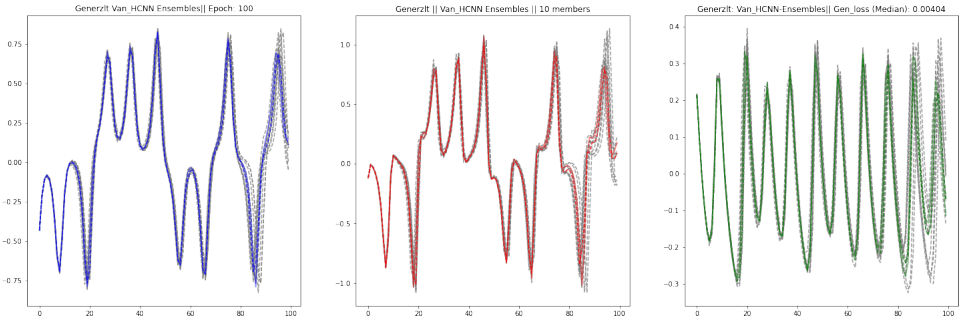

Figure 19. Vanilla HCNN ensemble computations.

Looking at the picture above, we can see that for the 10 different instances of Vanilla HCNN, the shape of the three time series are preserved. At each time step, we computed the median and average out of each the 10 forecasts. The result we obtained show a balance between both generalization errors with the values being respectively 4.04×10^{-3} for the ensemble's median forecast and 4.09×10^{-3} for the ensemble's average forecast. This shows that HCNN are able to model High dimensional non linear systems in a consistent way and that both the ensemble's median and average forecast are also reliable candidates for the forecast the dynamics of such systems. .

8. Conclusions

Throughout this paper, we have seen that HCNNs are able to model High dimensional deterministic non linear systems in a consistent way. The different improvement methods have been instrumental in enforcing a long term learning of the dynamics of the multivariate time-series generated from the Lorenz, the Rossler and the Rabinovich-Fabrikant Systems. Among the three improvement methods, measuring by the generalization error, the Partial Teacher Forcing method is the superior way to improve both the long memory and the extent of the forecast horizon. The Large Sparse HCNN method is mostly aligned to biological methods. The LSTM Formulation Method has a linear sub-structure to overcome the vanishing/exploding gradient problems which sometimes creates numerical instability. The results obtained from the ensemble computation graphically show that the shape of the three time series are near each other throughout the whole forecast horizon. Hence, this shows that for different instances of HCNN, the results are ensured to be reproducible. Moving forward, it is worth noting that the datasets were generated mathematically. This gives to those dynamical systems the attribute of fully observables. Since the final goal of this ongoing research is on the analysis of climate data, those are rather high-dimensional and noisy measurements: such kind of system are called partially observables.There is currently a work in progress to extend these techniques to wind forecasting as a basis for wind turbine control. Analysing such systems will require transferring the long memory insights gained from this experiment to that new task. Furthermore, the identification of the optimal HCNN meta-parameters and the formulation of additional improvement techniques for the learning of HCNN will also be probable directions to look at. On another hand, the ensemble forecasts (median and average of the ensemble forecasts) seem to be a promising direction to navigate into, for such forecasting exercise.

Author Contributions: Conceptualization, R.R., B.B., H.-G.Z. and V.M.; methodology, R.R., B.B., H.-G.Z. and V.M.; software, R.R.; validation, R.R., B.B., H.-G.Z. and V.M.; formal analysis, R.R.; investigation, R.R.; resources, R.R.; data curation, R.R.; writing—original draft preparation, R.R.; writing—review and editing, R.R., B.B., H.-G.Z. and V.M.; visualization, R.R.; supervision, R.R., B.B., H.-G.Z. and V.M.; project administration, R.R., B.B., H.-G.Z. and V.M.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was carried out with the aid of a grant from the International Development Research Centre, Ottawa, Canada, and with financial support from the Government of Canada, provided through Global Affairs Canada (GAC).

Acknowledgments: A special thanks to the African Institute for Mathematical Sciences, AIMS South Africa, which is my host institution, through the Next Einstein Initiative.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brunton, S.L.; Kutz, J.N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*; Cambridge University Press: Cambridge, UK, 2022.
2. Schäfer, A.M.; Zimmermann, H.G. Recurrent neural networks are universal approximators. In Proceedings of the 16th International Conference, Athens, Greece, 10–14 September 2006; Springer: Berlin/Heidelberg, Germany, 2006.
3. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
4. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, **2017.**
5. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
6. Haykin, S.; Network, N. A comprehensive foundation. *Neural Netw.* **2004**, *2*, 41.
7. Zimmermann, H.G.; Tietz, C.; Grothmann, R. Forecasting with recurrent neural networks: 12 tricks. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 687–707.
8. Zimmermann, H.G.; Grothmann, R.; Tietz, C.; Jouanne-Diedrich, H.V. Market modelling, forecasting and risk analysis with historical consistent neural networks. In *Operations Research Proceedings 2010*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 531–536.
9. Mvubu, M.; Kabuga, E.; Plitz, C.; Bah, B.; Becker, R.; Zimmermann, H.G. On Error Correction Neural Networks for Economic Forecasting. In Proceedings of the 2020 IEEE 23rd International Conference on Information Fusion (FUSION), Rustenburg, South Africa, 6–9 July 2020.
10. Zimmermann, H.G.; Neuneier, R.; Grothmann, R. Multi-agent modelling of multiple FX-markets by neural networks. *IEEE Trans. Neural Netw.* **2001**, *12*, 735–743. [CrossRef] [PubMed]
11. Danny, P.; Allon, J. Multivariate exponential smoothing: Method and practice. *Int. J. Forecast.* **1989**, *5*, 83–98.
12. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
13. Serrano-Pérez, J.D.; Fernández-Anaya, G.; Carrillo-Moreno, S.; Yu, W. New results for prediction of chaotic systems using deep recurrent neural networks. *Neural Process. Lett.* **2021**, *53*, 1579–1596. [CrossRef]
14. Rabinovich, M.I.; Fabrikant, A.L. Stochastic self-modulation of waves in nonequilibrium media. *J. Exp. Theor. Phys.* **1979**, *77*, 617–629.
15. Rössler, O.E. An equation for continuous chaos. *Phys. Lett. A* **1976**, *57*, 397–398. [CrossRef]
16. Curry, J.H. A generalized Lorenz system. *Commun. Math. Phys.* **1978**, *60*, 193–204. [CrossRef]
17. Edward, O.; Sauer, T.; Yorke, J.A. *Coping with Chaos. Analysis of Chaotic Data and the Exploitation of Chaotic Systems*; Wiley Series in Nonlinear Science; John Wiley and Sons: Hoboken, NJ, USA, 1994.