Predicting Chaotic Time Series using Machine Learning Techniques

Henry Maathuis*, Luuk Boulogne*, Marco Wiering, and Alef Sterk

University of Groningen, Groningen, The Netherlands,
maathuishenry@gmail.com, lhboulogne@gmail.com, m.a.wiering@rug.nl and a.e.sterk@rug.nl

Abstract. Predicting chaotic time series can be applied in many fields, e.g. in the form of weather forecasting or predicting stocks. This paper discusses several neural network approaches to perform a regression prediction task on chaotic time series. Each approach is evaluated on its sequence prediction ability on three different data sets: the intermittency map, logistic map and a six-dimensional model. In order to investigate how well each regressor generalizes, they are compared to a 1 Nearest Neighbor baseline. In previous work, the Hierarchical Mixture of Experts architecture (HME) has been developed. For a given input, this architecture chooses between specialized neural networks. In this work, these experts are Multilayer Perceptrons (MLPs), Residual MLPs, and Long Short-Term Memory neural networks (LSTMs). The results indicate that a Residual MLP outperforms a standard MLP and an LSTM in sequence prediction tasks on the logistic map and the 6-dimensional model. The standard MLP performs best in a sequence prediction task on the intermittency map. With the use of HMEs, we successfully reduced the error in all the above mentioned time series prediction tasks.

Keywords: Dynamical Systems, Neural Networks, Hierarchical Mixture of Experts

1 Introduction

Time series prediction is a task that involves the use of historical information to predict the future states of a system. Such tasks are often partitioned in one-step predictions and the prediction of a sequence of states. Many disciplines are concerned with time series prediction. These disciplines range from weather stations forecasting days or even weeks ahead to stock brokers and traders predicting the course of stocks. Most problems in nature, including prediction problems, deal with nonlinear or chaotic data, which poses a challenge. In the field of Machine Learning, a large quantity of work on predicting time series that involves using Neural Networks (NNs) such as Multi-Layer Perceptrons (MLP) and Radial Basis Function NNs (RBF-NN) has already been done, including work on

*These authors contributed equally to this work.

weather forecasting [1, 2]. Other research has been conducted with other Supervised Learning models such as Support Vector Machines or Recurrent Neural Networks [3–5].

To measure how well a system performs on a prediction task it is often compared to a baseline. 1 Nearest Neighbor regression is used as a baseline in this research. This baseline is a powerful tool when large amounts of data are available (see Section 4.1). However, since temporal data of any kind might not always be available in abundance, it is interesting to see whether an NN can generalize better than the baseline.

Three types of chaotic time series are discussed in Section 2. Different kinds of NNs: MLPs, Residual MLPs and Long Short-Term Memory (LSTM), are considered in this work and their predictive performance on chaotic data is investigated. These NNs are also embedded within a larger architecture, the Hierarchical Mixture of Experts (HME). This architecture allows expert NNs to specialize in certain input regions so that the problem complexity is reduced for the separate experts.

The nature of the dynamical systems with which the data sets for this work are produced makes the behavior of these systems state dependent. Allowing for an HME system where agents specialize in certain types of states could improve the system performance. In this research we are interested to see whether the HME architecture is useful for improving the prediction accuracy on chaotic time series.

2 Data Sets

For the convenience of the reader we first give a general overview of dynamical systems. Next, we give the concrete examples used in our machine learning experiments.

2.1 Dynamical Systems

Dynamical systems are mathematical models for everything that evolves in time. Simple examples are springs and pendulum clocks. More complicated examples are the celestial bodies comprising the solar system or the atmosphere which produces everyday weather. These systems are deterministic in the sense that the present state of the system completely determines its future. In other words, probability does not play a role in the evolution of a system. See [6] for an extensive account.

Many dynamical systems arise in the form of *iterated maps*. Let D be some domain and consider a map $f : D \to D$. For an initial condition $x_0 \in D$ we can iterate the map f by setting $x_{n+1} = f(x_n)$. This gives the following time series:

$$x_0, f(x_0), f(f(x_0)), f(f(f(x_0))), \ldots$$

A weather forecasting model, for example, fits in this framework. If an initial condition x_0 represents today's weather, then by solving the governing differential equations of atmospheric physics we can compute a prediction for tomorrow's weather $f(x_0)$.

Since the seminal work of the mathematician and meteorologist E.N. Lorenz [7] it is well known that deterministic systems can be unpredictable: small errors in the initial condition x_0 may lead to large errors in predictions for the future. This phenomenon, which is colloquially known as *chaos*, hampers long-term weather forecasts and stimulated the development of mathematical research on nonlinear dynamics and chaos theory.

2.2 The Logistic Map and Intermittency Map

The most familiar, and perhaps simplest, example of a dynamical system with chaotic dynamics is the *logistic map* which is given by:

$$f : [0,1] \to [0,1], \quad f(x) = rx(1-x), \tag{1}$$

where $0 < r \leq 4$ is a parameter. This map is a simple model for population growth which also takes overpopulation into account. Increasing the parameter r leads to a period doubling cascade, and for $r = 4$ the dynamics have been proven to be chaotic.

Another example of a 1-dimensional dynamical system is the so-called *intermittency map* [8] which is given by:

$$f : [0,1] \to [0,1], \quad f(x) = \begin{cases} x(1+(2x)^{\alpha}) & \text{if } 0 \leq x \leq \frac{1}{2}, \\ 2x-1 & \text{if } \frac{1}{2} < x \leq 1, \end{cases} \tag{2}$$

where $0 < \alpha < 1$ is a parameter. This map has a neutral fixed point at $x = 0$, which causes the time series to spend long times near $x = 0$. This effect becomes stronger when α tends to 1. For the experiments in our paper, this value is fixed to 0.5.

2.3 Atmosphere Data

A classical problem in the theory of atmospheric circulation is the characterization of the recurrent flow patterns observed at midlatitudes in northern hemisphere winters. The prime motivation for studying this phenomenon is to understand the persistence and predictability of atmospheric motion beyond the time scales of baroclinic synoptic disturbances (2 to 5 days). It is expected that insight in the nature of this so-called low-frequency variability will lead to significant progress in extended range weather forecasting [9].

Classical theories associate recurrent large-scale flow patterns with stationary states of the atmospheric circulation, which correspond to equilibria in the dynamical equations of atmospheric motion [10]. Small-scale weather acts then as a random perturbation inducing fluctuations around equilibria and transitions between states. From the perspective of nonlinear dynamical systems this scenario has been explained in terms of *intermittency* which means that a system alternates between regimes of chaotic and regular behavior such as nearly steady or periodic dynamics. The intermittency map given by Equation (2) exhibits this type of dynamics. Intermittency is observed in various forms in atmospheric models [11, 12]; also see [13] for an overview.

The simplest model for the midlatitude atmospheric circulation is the barotropic vorticity equation for flow over an orography profile (i.e., mountains):

$$\frac{\partial}{\partial t}\Delta\psi = -J(\psi, \Delta\psi + \beta y + \gamma h) - C\Delta(\psi - \psi^*), \tag{3}$$

where ψ is the stream function which describes the atmospheric velocity field, ψ^* is the forcing, β controls the Coriolis force, and h is the orography profile. The differential operators Δ and J are defined as $\Delta f = f_{xx} + f_{yy}$ and $J(f, g) = f_x g_y - f_y g_x$, respectively. For parameter settings and boundary conditions, see [11, 14].

A *low-order model* can be derived from Equation (3) by means of spectral discretisation. The idea is to expand the stream function ψ in a truncated Fourier series with time-dependent coefficients. An orthogonal projection then gives a system of ordinary differential equations. A particular low-order model consisting of 6 ordinary differential equations was studied in [11] who found intermittent transitions between two states representing a westerly and a blocked flow respectively which resemble the patterns found in the real atmosphere. The dynamics consists of three recurrent episodes: (i) transitions from westerly to blocked flows, typically taking 30 days, (ii) transitions from blocked to westerly flows, typically taking 40–80 days, and (iii) spiraling behavior around the westerly regime, typically lasting more than 200 days. The dynamics of the intermittency map given by Equation (2) can be seen as a prototype for this more complicated scenario.

From the 6-dimensional model derived from Equation (3), see [11, 14] for explicit equations, we generated a discrete time series by numerical integration. We sampled the continuous time series by intervals of 6 hours.

2.4 Splitting the Data Sets

The total length of the resulting discrete time series for each type of dynamical system is 12,001 data points. The first 12,000 are used as input and the last 12,000 as the corresponding output. The resulting data sets are divided into three segments of equal length, which are used respectively as training, test and validation sets for our systems. For the atmosphere data, the training, validation and test sets each translate to approximately 3 years of data.

3 System Design

In this work, two different Neural Network (NN) regressors are considered to learn the behavior of the dynamical systems described in Section 2. These are the well established feedforward Multi-Layered Perceptron (MLP) and the recurrent

Long Short-Term Memory NN (LSTM). Each of these NNs are evaluated on their ability to predict sequences and they are compared to a simple but robust baseline (1 Nearest Neighbor).

To obtain good performance, we allow a regressor to express different behavior for different parts of the data set domain by using the structure of the Hierarchical Mixture of Experts [15] ensemble technique (HME) for our regressor, explained in Section 3.3.

3.1 (Residual) Multilayer Perceptron

The MLP is a very popular NN used in a lot of different tasks in which a nonlinear problem is to be solved. An MLP consists of an input layer, output layer and an arbitrary amount of hidden layers. An MLP maps an input vector to an output vector: $f : R^{input} \rightarrow R^{output}$. More specifically, the output is obtained by a combination of the input and the weights associated between the neurons in the NN. After obtaining the output for each neuron, the result is transformed by feeding it to an activation function. This allows the system to learn nonlinear decision boundaries. The parameters of the model are learned by the use of an optimizer. An optimizer describes how the weights in an NN are updated according to the gradient obtained by using the backpropagation algorithm. Backpropagation uses a loss function with respect to the weights to compute the gradient of the output error and propagates this back through the NN. Such a loss function $l(d, y)$ computes a similarity measure between the desired output d and the actual output y.

A closely related NN is the MLP that utilizes residual learning blocks. This NN uses residual learning blocks instead of the default hidden layers. Apart from this learning block, this NN is identical to the plain MLP. Residual learning blocks allow the system to learn not only from the output of the layer itself, but also from its input. In a standard setup the output of an NN can be represented as:

$$y = f(x). \tag{4}$$

Here x is the input and y is the mapped output. However in the case of a residual learning block, the output is calculated as:

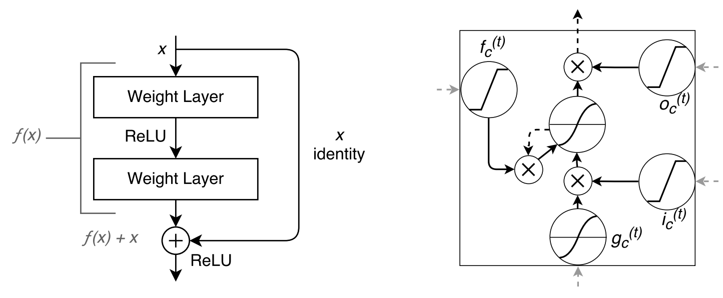
$$y = f(x) + x. \tag{5}$$

Empirical evidence has shown that residual learning blocks can reduce the error of the NN and allow for easier optimization [16]. Figure 1(a) shows a schematic overview of the residual learning block.

3.2 Long Short-Term Memory

In MLPs a layer of neurons only contains forward connections to subsequent layers. Layers in Recurrent Neural Networks (RNNs) however, also have neurons that maintain connections to themselves and preceding layers. When predicting the next time step in a series, these connections allow the past to be taken into account. This is because the activation of neurons is not only dependent on the input at the current time step, but also on input from earlier time steps [17]. An issue with RNNs is that the error gradients vanish after being propagated back through many layers or time steps. [18]. To overcome this problem, a Long Short-Term Memory NN (LSTM) has been introduced [19, 18]. LSTMs can be implemented by replacing the nodes in one or more hidden layers of an NN with so-called memory cells. These cells contain an internal state that is maintained using gates. The memory cell is depicted in Figure 1(b). Here, we give a brief description of the memory cell. A more extensive explanation of an LSTM and the memory cell is presented in [17].

A gate in a memory cell is defined as a neuron with a sigmoid activation function. In a gated connection from neuron A to neuron B, the activation of a gate is multiplied with the activation of A to obtain the input for B. Since the activation of a gate lies within the interval (0,1), it can be viewed as indicating the percentage of activation of A that flows through to B.



(a) (b)

Fig. 1: (a): A residual learning block as described in [16]. (b): The LSTM memory cell extended with forget gates [20], after [17]. On the neurons, the corresponding activation function (hard sigmoid [21] or hyperbolic tangent) is depicted. The multiplication of the output of nodes is depicted as a circle with the multiplication symbol. All arrows can be regarded as connections with a weight fixed at one. Filled arrow: Connection within one time step. Black dashed arrow: Connection to the next time step. Gray dashed arrow: Input connection. This is a connection from the previous time step to the current one.

Including the forget gate, which is described in [20], a memory cell contains three gates. For memory cell c at time step t, these are an input gate $i_c^{(t)}$, a forget gate $f_c^{(t)}$ and an output gate $o_c^{(t)}$. Besides gates, the cell has an input neuron $g_c^{(t)}$ and an internal state $s_c^{(t)}$. It produces an activation $h_c^{(t)}$, which is the activation of c in the hidden layer that is used as input for subsequent layers. We compute a forward pass through a hidden layer of LSTM cells with input $x^{(t)}$, as described by:

$$\begin{aligned}
\boldsymbol{g}^{(t)} &= \tanh(W^{gx}\boldsymbol{x}^{(t)} + W^{gh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_g), \\
\boldsymbol{i}^{(t)} &= \hat{\sigma}(W^{ix}\boldsymbol{x}^{(t)} + W^{ih}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i), \\
\boldsymbol{f}^{(t)} &= \hat{\sigma}(W^{fx}\boldsymbol{x}^{(t)} + W^{fh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f), \\
\boldsymbol{o}^{(t)} &= \hat{\sigma}(W^{ox}\boldsymbol{x}^{(t)} + W^{oh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o), \\
\boldsymbol{s}^{(t)} &= \boldsymbol{g}^{(t)} \odot \boldsymbol{i}^{(t)} + \boldsymbol{s}^{(t-1)} \odot \boldsymbol{f}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{s}^{(t)}) \odot \boldsymbol{o}^{(t)},
\end{aligned} \tag{6}$$

after [17]. Here, element-wise multiplication is indicated with the symbol $\odot$. Furthermore, W^{ij} denotes the weight matrix from j to i, $\boldsymbol{b_i}$ denotes the bias for i, tanh denotes the hyperbolic tangent and $\hat{\sigma}$ denotes a variant of the hard sigmoid [21], described by:

$$\hat{\sigma}(x) = \max(0, \min(1, 0.2x + 0.5)). \tag{7}$$

3.3 Hierarchical Mixture of Experts

The Hierarchical Mixture of Experts (HME) is a tree-structured architecture for Supervised Learning that was coined by Jordan et al. [15]. Their system, incorporates the idea of Mixture of Experts (MoE) also originating from the authors in earlier research. MoE follows a divide-and-conquer principle in which the problem space is partitioned into local regions.

Each of these local regions are assigned to individual experts and allows the experts to specialize in certain regions. In this architecture an expert could complete any specific task such as a classification or prediction task. In [22] a Mixture of Multi-Layer Perceptron Experts is implemented to forecast the Tehran stock exchange. The Mixture of Experts is also widely used in classification and has seen applications in domains such as gender, ethnic origin, and pose of human faces classification [23].

Empirical evidence has shown that HME has several advantages over the plain MoE architecture. HME often outperforms MoE which is attributed to the fact that a HME partitions the data both locally and globally, providing different resolution scales.

The MoE architecture consists of a manager (or gate) and a variable amount of experts. The manager network maintains a softmax output where the amount of output neurons equals the amount of experts. Each output unit represents how much the output of that corresponding expert should contribute to the final prediction. Using a softmax output allows us to divide the contribution of each expert and force the total contribution given each expert to partition unity. For an HME architecture with a depth of two, the i^{th} weighted output in the bottom layer is described by:

$$\mu_i = \sum_j g_{j|i}\mu_{ij}, \tag{8}$$

where μ_{ij} is the output of the j^{th} expert that contributes to μ_i and $g_{j|i}$ is the weight given to μ_{ij}. The output of the whole system μ is computed using the outputs in the bottom layer weighted by the top layer manager:

$$\mu = \sum_i g_i \mu_i. \tag{9}$$

The HME architecture consists of a variable amount of managers which each manage either multiple experts or the output of multiple linearly combined expert blocks. This definition allows the system to obtain an arbitrary recursion depth as seen in Equations (8) and (9). Figure 2 shows a general HME model of depth two. In this figure, x denotes the input which is identical for every manager and expert network. The individual gates and experts are trained end-to-end with backpropagation.

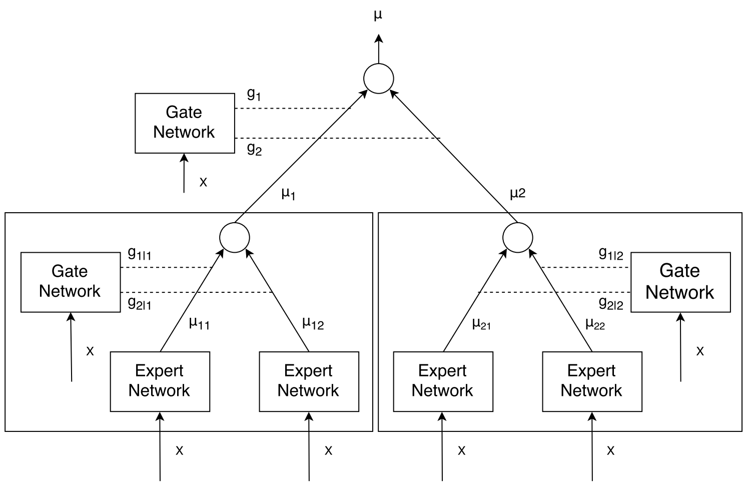


Fig. 2: Hierarchical Mixture of Experts model of depth two. Reprinted from [15].

.

Expert Networks. In the HME structure, multiple expert networks are present. These networks are full fledged regressors that are trained dependently of each other. In our setup the expert types that are considered are MLPs, Residual MLPs and LSTMs. For each of these types the hyperparameters associated with the network that yielded the lowest validation loss are used.

Gating Networks. The strength of the HME architecture lies in the managers that determine to what extent the individual agents contribute to the final prediction. In this research, MLPs, Residual MLPs and LSTM are tested as managers using the hyperparameters that yielded the lowest validation loss in the single learner tasks. LSTMs are notorious to learn from experience and therefore we hypothesize that LSTM managers could provide better results than MLP or Residual MLP managers.

4 Results

4.1 1 Nearest Neighbor

1 Nearest Neighbor (1NN) is used as a baseline measure for the systems described in Section 2. The baseline matches an input with the training input it received earlier and returns the corresponding training output. Starting at this corresponding training output, it follows the trajectory already present in the training space. This follows from the way the training set is constructed, see Section 2.4. The more training inputs are fed to the system, the likelihood of finding a similar training instance to an arbitrary input increases. This means that this baseline can perform well if the system has seen a lot of training instances before. The error of the 1NN baseline reduces over the amount of training instances used.

4.2 Training

The weights of the LSTM memory cells were initialized with the method described in [24]. All other weights were initialized with the method described in [25]. For weight optimization, we used Adam [26]. Training samples were presented to the models with a batch size of four.

Early Stopping. During training, once an NN obtains a better validation accuracy, this NN is saved. To reduce training time, early stopping is implemented by monitoring the validation loss. Training is terminated when either the maximum number of 1000 epochs has been reached or the validation loss did not decrease over 50 epochs.

Hyperparameters. In order to obtain a well performing NN architecture, preliminary tests to determine hyperparameters for the MLP, Residual MLP and LSTM were performed. For each type of NN, the hyperparameters were selected. The selection was done based on the ability to, given a time step in a validation set s_t, make an accurate prediction s'_{t+1} of the next time step s_{t+1}. The performance measure used was the Mean Squared Error, hereafter referred to as loss.

Each of the NNs are tested with different sets of parameters. For all NNs, a parameter sweep was performed over the hidden layer sizes, the learning rate and the learning rate decay. For the MLP and Residual MLP the sweep also included the number of hidden layers and activation functions. Table 1 shows the validation loss for each type of NN and data set together with their best corresponding parameters settings.

Table 1: The best found hyperparameters, which were selected using the validation loss of 1-step prediction. For each data type, the best performing NN type is shown in bold. The loss function is the Mean Squared Error.

Data	NN type	h. layers	h. l. sizes	activation	lr	lr decay	loss
Intermittency	**MLP**	**3**	**50**	**PReLU**	**0.01**	**0.001**	$\mathbf{1.60 \times 10^{-4}}$
	R. MLP	1	200	sigmoid	0.001	0.0001	4.83×10^{-4}
	LSTM	-	50	-	0.01	0.001	4.19×10^{-3}
Logistic	MLP	3	200	PReLU	0.01	0.0001	1.53×10^{-8}
	R. MLP	**3**	**200**	**sigmoid**	**0.01**	**0.0001**	$\mathbf{6.00 \times 10^{-9}}$
	LSTM	-	50	-	0.01	0.0001	2.82×10^{-3}
Atmosphere	MLP	3	200	PReLU	0.001	0.0001	8.12×10^{-8}
	R. MLP	**1**	**200**	**PReLU**	**0.001**	**0.0001**	$\mathbf{2.29 \times 10^{-8}}$
	LSTM	-	200	-	0.001	0.0001	6.99×10^{-6}

Final NN Architectures. For each dynamical system, we trained 10 NNs of each type with the best found hyperparameters (all rows in Table 1). For each dynamical system, we also trained one type of HME. The NN types and hyperparameters of the best performing NNs (bold rows in Table 1) were used for the experts and managers in these HMEs. The newly constructed models were trained end-to-end. This is needed for the different experts to learn state dependent behavior. We used four experts and three managers as illustrated in Figure 2. This resulted into a total of 40 trained NNs for each of the three dynamical systems.

4.3 Testing

Sequence Prediction Evaluation Method. The 120 NNs and the three baselines, from now on collectively referred to as regressors, were evaluated on their ability to predict the behavior of a dynamical system, given the starting state in the test set s_0. The sequence that describes the behavior of the dynamical system is generated by first making the regressor predict the next time step s'_1 given s_0. Subsequent time steps at time $t+1$ are recursively predicted using the prediction made at time t.

For the logistic and intermittency maps, the length l of the predicted sequences is 100 data points. For the atmosphere data, $l = 600$, which corresponds to around five months of data.

To obtain more reliable results, the regressors are evaluated given multiple starting states. Given the test or validation set S the first 4,000 $- \, l$ (the sets consist of 4,000 data points) elements are used exactly once as starting state when evaluating a regressor. The resulting 4,000 $- \, l$ evaluations are averaged elementwise to obtain a single sequence of error measures for each regressor a. This sequence of error measures e_a^S is the mean loss of a, given S.

NN Selection. We define the performance of a regressor a on the validation set val as the mean of all elements of e_a^{val}. Using this performance measure, we selected the single best NN for each dynamical system. Table 2 shows these results.

Table 2: This table shows the lowest mean validation loss on sequence prediction as described in section 4.3 of the single best NN of each NN type for each data set. The best performing NNs are shown in bold.

Data	NN type	Lowest mean loss
Intermittency	**HME**	**0.158152**
	Residual MLP	0.327154
	MLP	0.160673
	LSTM	0.164677
Logistic	**HME**	**0.219681**
	Residual MLP	0.220254
	MLP	0.222794
	LSTM	14.86825
Atmosphere	**HME**	**0.008987**
	Residual MLP	0.010553
	MLP	0.010731
	LSTM	0.027812

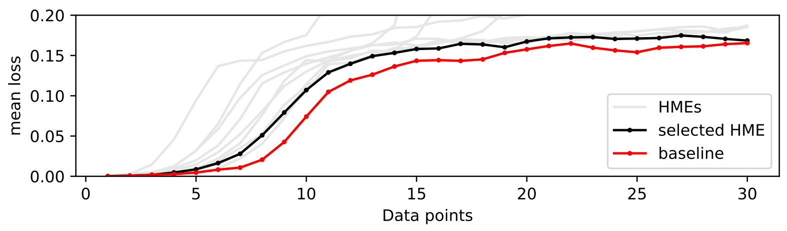
When taking for a each NN of the best NN type and each baseline for every dynamical system, we computed e_a^{test}. Here $test$ denotes the test set corresponding to the dynamical system on which a was trained. The losses of the predicted sequences are shown in Figure 3.

5 Discussion

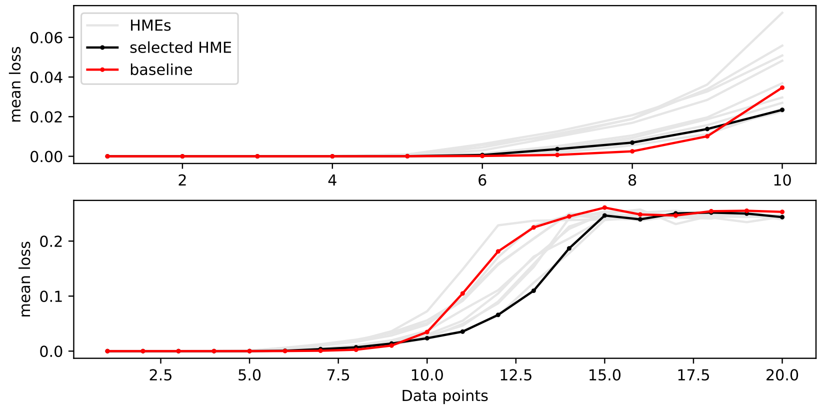
Intermittency Map. The results obtained with the intermittency data show that the sequence prediction performance of the NNs are worse than the baseline regressor. Figure 3(a) shows that the NNs are able to capture the general behavior of the dynamical system, but the predicted sequence deviates a lot from the ground-truth. The loss of the HME rapidly increases after the 7th data point.

Logistic Map. Figure 3(b) shows the sequence prediction ability of the NNs and the baseline on the logistic map. An interesting pattern arises in the domain [6, 9]. The baseline performance exceeds the performance of the selected Neural Network within this interval. However one should note that the selected HME performs better in the range [10, 15]. After data point 15 there is no evidence that either the baseline or the HME outperforms one another.

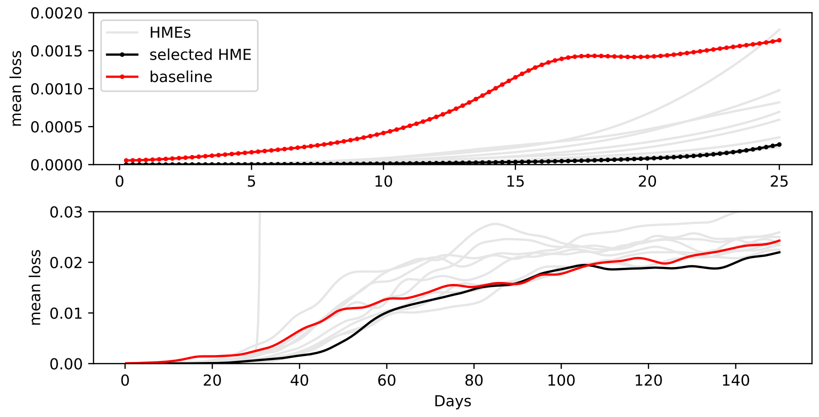
Atmosphere Data. The most interesting results are found in the 6-dimensional model. We found that out of all dynamical systems considered, relative to the



(a)



(b)



(c)

Fig. 3: Sequence prediction on intermittency map (a), logistic map (b) and atmosphere (c) test data.

baseline, the NNs performed best on the atmosphere data. On sequence prediction, an HME with Residual MLPs for managers and experts reliably outperforms the baseline of 1NN for about 24 days (96 time steps) as can be seen in Figure 3(c). Furthermore, the loss remains fairly small during the first 20 days. The HME that performed best on the validation data outperforms the baseline for several months. Figure 4 shows that HMEs are able to capture the general behavior of the atmospheric dynamical system.

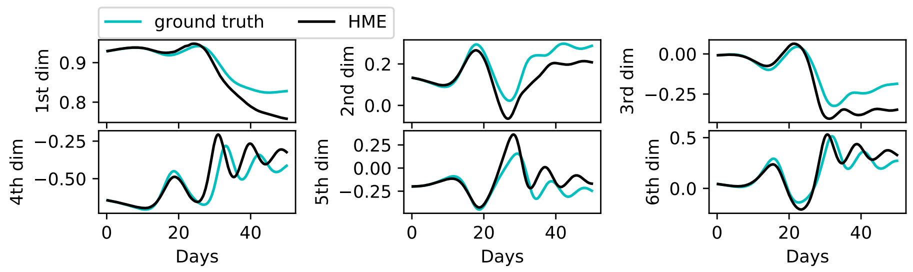


Fig. 4: Example of atmosphere data prediction by an HME over 50 days. The starting state has been chosen from the test set at random.

Relative to the baseline, the NNs perform better on the atmosphere data than on the other data sets. This could be because of the larger dimensionality of the atmosphere data. Because the same number of data points are sampled from each data set, the larger dimensionality causes the atmosphere data to be relatively sparse. This might allow the better interpolation capabilities of the NNs with respect to the baseline to become more apparent on the atmosphere data than on the other data sets.

As can be seen in Figure 3(c) where one of the grey lines becomes almost vertical around day 30, the mean loss of one of the HMEs suddenly becomes enormous. This is probably because this HME made a prediction slightly outside of the domain of the training set and predicted the next datapoint even further outside of the domain. This type of extrapolation causes the error to snowball.

6 Conclusion and Future Work

Conclusion. This work provides an overview of Neural Networks (NNs) in combination with a Hierarchical Mixture of Experts (HME) architecture applied to three different dynamical systems: the logistic map, the intermittency map and a 6-dimensional model which contains patterns that are found in the real atmosphere.

When testing Multi-Layer Perceptrons (MLPs), Residual MLPs and Long Short-Term Memory NNs (LSTMs) on these data sets it was observed that the best results were obtained with the Residual MLP for the logistic map and the atmosphere data. The best results for the intermittency map were obtained with an MLP.

Compared to the 1-Nearest Neighbor baseline, the NNs used in this work are not suitable for dynamical systems such as the intermittency and logistic map. On the contrary, we found that the time series prediction test on the 6-dimensional data does give promising results. In the first couple of months of the predicted sequence, the baseline is clearly outperformed by the HME.

The results indicate that the HME architecture helps in reducing the generalisation error of dynamical system predictions, since for every data set tested, better results were obtained with using this architecture than without.

Future Work. Although we obtained promising results, there are several ways in which they could be improved. Future research can focus on the use of deeper HME architectures or more extensive hyperparameter studies. Other types of NNs or sequence prediction techniques might also prove useful for the problem at hand. A final suggestion is to investigate different ensemble techniques that might reduce the generalization error.

A drawback of predicting the behavior of a dynamical system step by step as described in Section 4.3 is that, when making a prediction of a time step, the system only uses one preceding step. For feedforward NNs, there is thus no countermeasure for the accumulation of error over time. Performance might thus be increased by using multiple previous predictions as input to the NNs.

Furthermore, although LSTMs did not perform well on these data sets, other types or combinations of recurrent connections might also help to counter this problem. Future research could also indicate whether, for the different data sets, varying the amount of data used influences the performance of NNs with respect to the baseline. It would also be interesting to see how well NNs can make predictions based on noisy training data and whether the models resulting from this research could be used as pre-trained models for training and testing on real world data.

References

1. Maqsood, I., Khan, M.R., Abraham, A.: An ensemble of neural networks for weather forecasting. Neural Computing & Applications **13**(2) (2004) 112–122
2. Taylor, J.W., Buizza, R.: Neural Network Load Forecasting With Weather Ensemble Predictions. IEEE Transactions on Power Systems **17**(3) (2002) 626–632
3. Gardner, M.W., Dorling, S.: Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. Atmospheric environment **32**(14) (1998) 2627–2636
4. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and computing **14**(3) (2004) 199–222
5. Maier, H.R., Dandy, G.C.: Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environmental modelling & software **15**(1) (2000) 101–124
6. Broer, H., Takens, F.: Dynamical Systems and Chaos. Volume 172 of Applied Mathematical Sciences. Springer (2011)

7. Lorenz, E.: Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences **20** (1963) 130–141
8. Pomeau, Y., Manneville, P.: Intermittent Transition to Turbulence in Dissipative Dynamical Systems. Communications in Mathematical Physics **74** (1980) 189–197
9. Reinhold, B.: Weather Regimes: The Challenge in Extended-Range Forecasting. Science **235** (1987) 437–441
10. Charney, J., DeVore, J.: Multiple Flow Equilibria in the Atmosphere and Blocking. Journal of the Atmospheric Sciences **36** (1979) 1205–1216
11. Crommelin, D., Opsteegh, J., Verhulst, F.: A Mechanism for Atmospheric Regime Behavior. Journal of the Atmospheric Sciences **61** (2004) 1406–1419
12. Sterk, A., Vitolo, R., Broer, H., Simó, C., Dijkstra, H.: New nonlinear mechanisms of midlatitude atmospheric low-frequency variability. Physica D: Nonlinear Phenomena **239** (2010) 702–718
13. Broer, H., Vitolo, R.: Dynamical systems modelling of low-frequency variability in low-order atmospheric models. Discrete and Continuous Dynamical Systems B **10** (2008) 401–419
14. Sterk, A., Holland, M., Rabassa, P., Broer, H., Vitolo, R.: Predictability of extreme values in geophysical models. Nonlinear Processes in Geophysics **19** (2012) 529–539
15. Jordan, M.I., Jacobs, R.A.: Hierarchical Mixtures of Experts and the EM Algorithm. Neural computation **6**(2) (1994) 181–214
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
17. Lipton, Z.C., Berkowitz, J., Elkan, C.: A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv preprint arXiv:1506.00019 (2015)
18. Hochreiter, S.: The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **6**(02) (1998) 107–116
19. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural computation **9**(8) (1997) 1735–1780
20. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to Forget: Continual Prediction with LSTM. IET Conference Proceedings (1999) 850–855
21. Courbariaux, M., Bengio, Y., David, J.P.: BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In: Advances in Neural Information Processing Systems. (2015) 3123–3131
22. Ebrahimpour, R., Nikoo, H., Masoudnia, S., Yousefi, M.R., Ghaemi, M.S.: Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange. International Journal of Forecasting **27**(3) (2011) 804–816
23. Gutta, S., Huang, J.R., Jonathon, P., Wechsler, H.: Mixture of Experts for Classification of Gender, Ethnic Origin, and Pose of Human Faces. IEEE Transactions on neural networks **11**(4) (2000) 948–960
24. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. (2010) 249–256
25. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: Proceedings of the IEEE international conference on computer vision. (2015) 1026–1034
26. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980 (2014)