Neural Network Training Loss Optimization Utilizing the Sliding Innovation Filter

Naseem Alsadi*[a], Waleed Hilal[a], Onur Surucu[a], Alessandro Giuliano[a], Stephen A. Gadsden[a], John Yawney[b], Mohammad Al-Shabi[c]
[a]McMaster University, 1280 Main St W, Hamilton, ON L8S 4L8 ; [b]Adastra Corporation, 8500 Leslie St #600, Thornhill, ON L3T 7M8; [c]University of Sharjah, Sharjah, UAE

ABSTRACT

Artificial feedforward neural networks (ANN) have been traditionally trained by backpropagation algorithms involving gradient descent algorithms. This is in order to optimize the network's weights and parameters in the training phase to minimize the out of sample error in the output during testing. However, gradient descent (GD) has been proven to be slow and computationally inefficient in comparison with studies implementing the extended Kalman filter (EKF) and unscented Kalman filter (UKF) as optimizers in ANNs. In this paper, a new method of training ANNs is proposed utilizing the sliding innovation filter (SIF). The SIF by Gadsden *et al.* has demonstrated to be a more robust predictor-corrector than the Kalman filters, especially in ill-conditioned situations or the presence of modelling uncertainties. In this paper, we propose implementing the SIF as an optimizer for training ANNs. The ANN proposed is trained with the SIF to predict the Mackey-Glass Chaotic series, and results demonstrate that the proposed method results in improved computation time compared to current estimation strategies for training ANNs while achieving results comparable to a UKF-trained neural network.

Keywords: Optimization, Estimation Theory, Kalman Filter, Gradient Descent, Smooth Variable Structure Filter, Sliding Modes, State-Space Methods, Deep Learning, Machine Learning, Neural Networks

BRIEF INTRODUCTION

Gradient descent (GD) is an algorithmic optimization technique that minimizes a function iteratively. In the field of machine learning and deep learning, gradient descent has been widely recognized as the most popular optimization algorithm for the training of models due to its simplicity and ease of implementation. Deep learning models rely on the existence of data to train the model to modify its parameters such as weights and biases. Subsequently, the model is trained with the subset of the existing data by determining the cost function, which represents the error between the predicted output and the actual training data output. The optimization algorithm works to minimize this cost function as much as possible by adjusting the weights and biases of the learning model.

Artificial neural networks (ANN) can approximate any function with arbitrary precision [1]. Du *et al.* proved that for over-parameterized deep neural networks, gradient descent converges to a global minimum of a cost function and achieves zero training loss despite the non-convexity of the objective function [2, 3]. Their analysis was extended to residual convolutional neural networks to prove similar convergence. These findings were also confirmed in Zhang *et al.*, which showed that the stochastic gradient descent manages to efficiently train a convolutional neural network used for image classification and easily fits randomized labelling of the data [4].

The use of gradient descent has also extended to other architectures of neural networks, such as radial basis neural networks which arose due to the slow convergence of the gradient descent algorithms developed for feedforward neural networks [5, 6]. Furthermore, Wijnhoven *et al.* explored the use of gradient descent in object detection instead of support vector machine (SVM) when analyzing very large training datasets as the existing models were computationally complex [7]. The studies showed that gradient descent converges quickly to the overall optimum, requires only one iteration over the training set, and is faster than SVM by two to three orders of magnitude without loss of classification performance [7].

*Naseem Alsadi, Waleed Hilal and Onur Surucu equally contributed to this publication.

Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV, edited by Tien Pham, Latasha Solomon, Proc. of SPIE Vol. 12113, 121131Z
 · 0277-786X · doi: 10.1117/12.2619029

It was shown by Puskorius *et al.* that applications of artificial neural networks in non-linear dynamic control were sparse and that systems were predominantly static network architectures that are trained by gradient descent [8, 9]. However, in recurrent neural networks (RNN), gradient descent has been widely overlooked due to practical difficulties such as being slow and also ineffective in converging to a solution [8, 10]. Additionally, in the case of random initialization, gradient descent generally performs poorly with deep neural networks [11]. These issues with gradient descent have resulted in the exploration of other training methods for neural networks such as the Kalman filter (KF) and Extended Kalman filter (EKF) [8].

Gradient descent remains the most prominent training loss optimization algorithm, however, it does not necessarily converge to the global minima. In numerous cases, gradient descent struggles to find global minima, only to achieve local convergence [12]. When employing a gradient descent-based optimization methodology for training a neural network, there are two common outcomes. The first of which is the nonconvex nature of the training criterion and the second is the magnitude of nonglobal local
minima [13]. As a direct consequence of these two outcomes, the potential for local convergence becomes a seemingly indispensable characteristic of gradient descent.

The issue of local convergence has become known as the local minima problem, and there has been extensive work conducted on methods to avoid this complication when training neural networks [14]. Some authors have attempted to build upon gradient descent while introducing novel methods to the greater neural network architecture. However, there is a steep cost associated with the reduction of local minima. Lo *et al.* advised that a methodology grounded in convexification will ultimately lead to the lowest cost and greatest utility. The authors propose convexification of the sum of squared errors to solve the problem at hand [15].

Additionally, researchers have taken steps to replace gradient descent by employing variant optimization functions. Ahmad *et al.* proposes the utilization of genetic algorithms to replace gradient descent, with the aim of locating a global optimal solution in a complex space [16]. The authors explored the potential advantages of using variant optimization algorithms in numerous situations. The results show that the introduction of genetic algorithms yields a smaller classification error than gradient descent in specific applications [16]. There is plenty of literature discussing numerous substitutes for gradient descent [17, 18].

Of interest to the core topic, is the advantages of using estimation strategies to augment model performance. Various authors posit the use of KF to reduce the chance of local convergence. Literature suggests that whilst classical gradient descent is susceptible to adopting a local minima, the KF can be utilized to avoid this issue. Benromdhane *et al.* postulated that using an EKF over gradient descent will enable escaping the 'local minima' problem. The authors report that the proposed method converged to a satisfactory minima under various initial conditions, however, led to a substantial computational cost [19]. Katić *et al.* advance a method to maintain convergence properties while reducing computational complexity. The authors suggest that applying the EKF to all network parameters is inherently computationally complex. Therefore, the authors opt for the decomposition of the network into variant layers, leaving the cumulative network optimization process to be conducted layer by layer. The authors report that the proposed method converged much faster than its gradient descent-based counterpart, while also offering an excellent compromise between the computation complexity and convergence properties [20].

Li *et al.* propose the implementation of an EKF-based training loss optimization algorithm. The authors discover that the EKF produced improved convergence properties and a quicker training speed than its gradient descent-based counterpart [21].

Darojah *et al.* conducted a comparative study between the EKF and a classical gradient descent-based backpropagation algorithm. The authors use the EKF as a method of state estimation for a nonlinear dynamic system, where the state vector is a cumulative representation of the network's synaptic weights. The authors reported lower computation time and higher performance rates when using the EKF [22].

As asserted previously, when using the EKF to train a neural network, the process becomes that of nonlinear state estimation. The literature shows that when dealing with nonlinear state estimation, the unscented Kalman filter (UKF) yields better results than the EKF [23]. Various authors, however, postulate that the superiority of the UKF not only lies within performance characteristics but rather within the implementation itself. LaViola *et al.* pointed to an advantage, namely that when dealing with an UKF, there is no requirement for linearizing with the Jacobian matrices [24]. The authors

allude to the fact that the UKF will utilize a minimum set of sample points to locate the mean and covariance estimates. Furthermore, the paper affirms that the UKF is a superior alternative to the EKF for numerous applications, including neural networks [24]. In addition to improved performance and noncomplex implementation, the UKF does not increase computational complexity, rather it can operate at the same level of computational complexity as the EKF [25].

In another study, Darojah *et al.* compare the results achieved in the previously mentioned paper, in which the EKF was used, to an UKF implementation [26]. The authors report that although for this specific implementation there was no significant variance in performance, the advantage held by the UFK was the lack of requirement for calculating the Jacobian [24, 26].

Zhan *et al.* suggest that while the EKF provides a straightforward implementation, several drawbacks can be combated with the use of the UKF [27]. Some of these drawbacks include a biased nature of estimates, instability to linearization, and costly calculation of the Jacobian matrices. The authors show that when dealing with neural network training, an UKF implementation can outperform an EKF implementation [27].

Contemporary advancements have progressed the condition of nonlinear state estimation methodologies. Alternatively, sliding mode observers (SMO) were presented to estimate the true states by utilizing variable structure techniques [28]. Based on these two techniques, the smooth variable structure filter (SVSF) was formed to estimate the true states as an alternative method [29]. An enhanced alternative, the sliding innovation filter (SIF) was proposed as a predictor-corrector estimator for linear systems with a similar principle as the SVSF. However, the SIF achieves more robust results against disturbances by utilizing a simpler gain structure [29].

Nowadays, neural networks are the most used learning algorithms to model non-linear and complex relationships [30]. Optimizers help to adjust the weights and the learning rates to converge a faster result. The EKF neural network training technique was first proposed in 1989 by Singhal *et al.* [31]. According to their paper, the EKF is a faster optimal estimator compared to the standard backpropagation (SBP). Also, the KF based techniques achieve the global optimal solution in fewer iterations with less training effort [32]. The SVSF was also utilized to tune the network weights as a novel training method, and it achieved guaranteed stability with a great static input-output mapping [33]. Since the SIF accomplishes a greater accuracy compared to the SVSF, greater performance is expected from the implementation of the SIF as an optimization technique [29]. Therefore, the implementation of the SIF in neural network training is proposed as an improvement to current techniques of optimization.

In this paper, Section II provides an overview of neural networks and the equations involved in forward propagation. Section III summarizes the process and equations of the UKF, followed by the SIF equations in Section IV. In Section V, the proposed work and experimental simulations are stated followed by an analysis and discussion of the results. Finally, concluding statements and remarks are presented in Section VI.

I. NEURAL NETWORKS

Neural Networks (NN) are computer algorithms that mimic the working principle of the human brain [34]. The processing elements, known as neurons, consist of weights and biases. The network structure consists of an arbitrary number of neurons placed inside three main layers: the input layer, the hidden layers and the output layer [35]. The network is densely connected, which means that each input neuron is connected to every hidden layer neuron, which are also in turn connected to every neuron in the output layer. Feedforward neural networks (FFNN) are the simplest form among neural networks, moving only in one direction (forward) from the input layer, through the hidden layers, to the output layer [36]. Each neuron inside the layers is fired by an activation function [34]. There are a vast majority of activation functions that can be used, but due to the restricted size of weights and biases the rectified linear (ReLu) is the most practical one for our case [37]. The function returns itself if the output is positive, otherwise it returns zero.

$$ReLu(x) = \max(x, 0) \qquad (1)$$

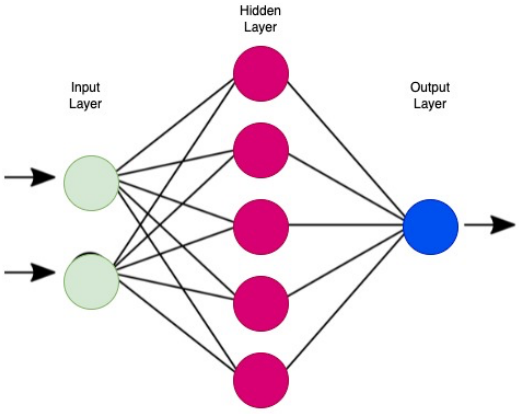

Fig. 1. Illustration of an artificial neural network with an input layer, one hidden layer and an output layer.

Let us assume that w represents the weights of the NN, b represents the biases, x refers to the input, with the output referred to as z, n is the total number of neurons inside the previous layer and each neuron is assigned with i. The equation below (2) provides the sample calculation of a neuron's output in a NN [34, 36, 38]:

$$z = w^T x + b \tag{2}$$

An activation function is then applied to each output of every neuron within the hidden layer. The training "learning" process involves tuning the network's synaptic weights iteratively to build an accurate prediction from given inputs. A common way to attack this non-convex problem is using gradient descent (GD) to find local minima of the function [2]. Stochastic gradient descent (SGD) is preferable for smaller datasets due to its less significant computation complexity [38]. Let us assume that y is the true value, $f(w, b)$ denotes the loss function, noting that $\nabla_{w,b}$ is the gradient of the loss function. The objective is to minimize the loss function by tuning the weights and the biases so that the overall network produces an output that results in the smallest cost, or error, as possible. The cost function shows how precise is our model at predicting the output, and is shown in equation (3) follows [39]:

$$f(w, b) = \frac{1}{N} \sum_{i=1}^{n} (y_i - (wx_i + b))^2 \tag{3}$$

In order to find the slope pointing in the direction of the global minima, the gradient parameter is calculated by using partial derivatives in equation (4) as follows [39]:

$$\nabla_{w,b} = \begin{bmatrix} \frac{df}{dw} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum_{i=1}^{n} -2w_i(y_i - (wx_i + b)) \\ \frac{1}{N} \sum_{i=1}^{n} -2(y_i + (wx_i + b)) \end{bmatrix} \tag{4}$$

After calculating the gradient, new weights and biases are computed with equations (5) and (6) [38, 40]. The adjustment of the step magnitude is expressed with α which determines the step size of the GD. Then, the whole process repeats itself recursively until the global minima is reached.

$$w_{k+1} = w_k - \alpha \nabla_{w,b} f(w, b) \tag{5}$$

$$b_{k+1} = b_k - \alpha \nabla_{w,b} f(w, b) \tag{6}$$

II. THE UNSCENTED KALMAN FILTER

The UKF was a method proposed by Julier *et al.* to generalize the KF to nonlinear systems based on a weighted statistical linear regression strategy, linearizing the model statistically [41, 42]. The UKF is considered superior to the EKF in terms of accuracy and is much simpler to implement. A set of weighted samples or 'sigma-points' is generated from the states' projected probability distribution around the mean and covariance [42, 43]. The sigma-points are subsequently propagated through nonlinear functions leading to an approximation of the desired estimate's mean and covariance [42, 43]. A deterministic sampling technique known as the unscented transform is used by the UKF, which has been shown to provide a more accurate *a priori* estimate of the mean and covariance in comparison to its predecessor, the EKF [42].

These strategies involved in the UKF allow it to eliminate the need for linearization of the system model by determining the Jacobian matrices, which is considered cumbersome and the limiting factor for the EKF's performance [42, 43, 44]. For applications with significantly nonlinear models, there is evidence that approximating the linear model by evaluating the Jacobian introduces significant error when determining the estimate of the covariance matrices, further highlighting the superiority of the UKF [44].

An overview of the equations involved in the UKF as well as a summary of the process is detailed as follows [42, 43]. First, $(2n + 1)$ sigma points are generated to approximate the n-dimensional random variable x_k which has a mean $\hat{x}_{k|k}$ and covariance $P_{k|k}$. The calculation of these sigma points' initial samples and weights is as follows:

$$X_{0,k|k} = \hat{x}_{k|k} \tag{7}$$

$$W_0 = k/(n + k) \tag{8}$$

Calculating subsequent n sigma points is then possible by the following equations:

$$X_{i,k|k} = \hat{x}_{k|k} + \left(\sqrt{(n + k)P_{k|k}}\right)_i \tag{9}$$

$$W_i = 1/[2(n + k)] \tag{10}$$

Similarly, the rest of the remaining n sigma points are determined as follows:

$$X_{i+n,k|k} = \hat{x}_{k|k} - \left(\sqrt{(n + k)P_{k|k}}\right)_i \tag{11}$$

$$W_{i+n} = 1/[2(n + k)] \tag{12}$$

In the sigma point initialization equations, k is a design parameter that is usually much smaller than 1. The weight associated with the ith sample point is denoted as W_i [42]. Next, the sigma points are propagated through the system's nonlinear model (13) and with the weight for each respective point, a state estimate prediction is calculated (14) as follows:

$$\hat{X}_{i,k+1|k} = f(X_{i,k|k}, u_k) \tag{13}$$

$$\hat{x}_{k+1|k} = \sum_{i=0}^{2n} W_i \hat{X}_{i,k+1|k} \tag{14}$$

Then, the predicted state error covariance can be determined from the previous two equations by:

$$P_{k+1|k} = \sum_{i=0}^{2n} W_i (\hat{x}_{i,k+1|k} - \hat{x}_{k+1|k})(\hat{x}_{i,k+1|k} - \hat{x}_{k+1|k})^T \tag{15}$$

After being successfully propagated through the system's nonlinear model, the sigma points must then undergo propagation through the nonlinear measurement model (16) in order to arrive at a predicted measurement (17) as follows:

$$\hat{Z}_{i,k+1|k} = h(\hat{X}_{i,k+1|k}, u_k) \tag{16}$$

$$\hat{z}_{k+1|k} = \sum_{i=0}^{2n} W_i \hat{Z}_{i,k+1|k} \tag{17}$$

Now it is possible to calculate the innovation (measurement) covariance (18) and the cross-covariance between the states and measurements (19) as follows:

$$P_{zz,k+1|k} = \sum_{i=0}^{2n} W_i (\hat{Z}_{i,k+1|k} - \hat{z}_{k+1|k})(\hat{Z}_{i,k+1|k} - \hat{z}_{k+1|k})^T \tag{18}$$

$$P_{xz,k+1|k} = \sum_{i=0}^{2n} W_i (\hat{X}_{i,k+1|k} - \hat{x}_{k+1|k})(\hat{Z}_{i,k+1|k} - \hat{z}_{k+1|k}) \tag{19}$$

The Kalman gain can now be calculated from the innovation and cross-covariance previously as follows in (20):

$$K_{k+1} = P_{xz,k+1|k} P_{zz,k+1|k}^{-1} \tag{20}$$

Finally, the updated a posteriori state estimate and state error covariance can be calculated by (21) and (22):

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - \hat{z}_{k+1|k}) \tag{21}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} P_{zz,k+1|k} K_{k+1}^T \tag{22}$$

The complexity of the UKF at first glance may seem to be a computational burden, however this is offset by the fact that it does not require linearization of the nonlinear function through Jacobian matrices [26]. For increasingly nonlinear systems, the UKF is considered a very accurate estimator [42, 43].

III. THE SLIDING INNOVATION FILTER

The SIF is a method by Gadsden *et al.* which is based on variable structure techniques and sliding mode observers. The benefit of the SIF over the UKF is its simplicity and ability to maintain robustness in the presence of uncertainties and disturbances [29]. Like the KF and UKF, the SIF is a predictor-estimator estimation method and utilizes a correcting term known as the SIF gain, similar to the Kalman gain [29].

A. Linear Systems and Measurements Case

The structure of the SIF's prediction stage is very similar to the KF and involves the calculation of *a priori* state estimates $\hat{x}_{k+1|k}$, state error covariance $P_{k+1|k}$ and innovation $\tilde{z}_{k+1|k}$ shown respectively as follows [29]:

$$\hat{x}_{k+1|k} = A\hat{x}_k + Bu_k \tag{23}$$

$$P_{k+1|k} = AP_{k|k}A^T + Q_k \tag{24}$$

$$\tilde{z}_{k+1+k} = z_{k+1} - C\hat{x}_{k+1|k} \tag{25}$$

In (23) through to (25) the system matrix, input gain matrix and measurement matrix are represented by A, B and C respectively. The system's input is represented as u_k and the system and measurement covariance matrices are Q and R

respectively, with the assumption that the system and measurement noises are white, meaning zero mean and Gaussian [29].

For the update stage, the SIF gain K_{k+1} is calculated along with the *a posteriori* state estimates $\hat{x}_{k+1|k+1}$, and updated state error covariance $P_{k+1|k+1}$ respectively as follows:

$$K_{k+1} = C^{+}\overline{sat}(\left|\tilde{z}_{k+1|k}\right|/\delta) \tag{26}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}\tilde{z}_{k+1|k} \tag{27}$$

$$P_{k+1|k+1} = (I - K_{k+1}C)P_{k+1|k}(I - K_{k+1}C)^{T} \ldots \\ \ldots + K_{k+1}R_{k+1}K_{k+1}^{T} \tag{28}$$

From the previous equations, in (26) the term C^{+} indicates to the pseudoinverse of the measurement matrix, whilst $\overline{sat}$ refers to the saturation value (value between -1 and +1) of the diagonal of that term, $\left|\tilde{z}_{k+1|k}\right|$ is innovation's absolute value and δ signifies the width of the sliding boundary layer. In equation (28), I signifies the identity matrix with dimensions $n \times n$ where n refers to the number of states [29].

The KF and SIF strategies differ in terms of their gain structure and is considered the key difference between the two filters. Optimality is expected by the KF since the gain is derived as a function of the state error covariance, however the SIF gain relies on the measurement matrix, innovation term and uniquely, a sliding boundary layer term [29].

B. Nonlinear System and Measurements Case

The SIF is like the KF in the sense that it can be extended to nonlinear systems and measurements by determining their respective Jacobian matrices to find their linearized form [29]. This proposed method is referred to as the extended sliding innovation filter (ESIF). The Jacobian matrices are found by evaluating the partial derivatives of the nonlinear system and measurements functions, $f(\hat{x}_{k|k}, u_k)$ and $h(\hat{x}_{k+1|k})$ as follows:

$$F_{k+1} = \left.\frac{\partial f}{\partial f}\right|_{\hat{x}_{k|k}, u_k} \tag{29}$$

$$H_{k+1} = \left.\frac{\partial h}{\partial f}\right|_{\hat{x}_{k+1|k}} \tag{30}$$

The prediction stage involves determining the *a priori* state estimates like the linear case and can be summarized as follows:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \tag{31}$$

$$P_{k+1|k} = F_{k+1}P_{k|k}F_{k+1}^{T} + Q_{k+1} \tag{32}$$

$$\tilde{z}_{k+1|k} = z_{k+1} - h(\hat{x}_{k+1|k}) \tag{33}$$

In (31), f denotes the nonlinear system function, F_k is the linearized version after the corresponding Jacobian matrix has been evaluated at time k [29]. Similarly, h denotes the nonlinear measurement function whilst H_k is the linearized version.

The update stage consists of three more equations which can be summarized as follows:

$$K_{k+1} = H_{k+1}^{+}\overline{sat}(\left|\tilde{z}_{k+1|k}\right|/\delta) \tag{34}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}\tilde{z}_{k+1|k} \tag{35}$$

$$P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^{T} \\ \ldots + K_{k+1}R_{k+1}K_{k+1}^{T} \tag{36}$$

The term H_{k+1}^{+} in (34) is the pseudoinverse of the linearized Jacobian of the measurement matrix at time $k+1$, and in (36) the term H_{k+1} is the linearized Jacobian of the measurement matrix at time $k+1$ [29].

To summarize, we can see that partial derivatives of the system and measurement matrix are evaluated to determine the linearized Jacobian in (29) and (30). From there, an initial prediction for the states is calculated in (31), as well as a prediction for the state error covariance and innovation in (32) and (33). From the innovation determined, it is possible to calculate the SIF gain from (34) which is used to update the state estimates and state error covariance as shown in (35) and (36) respectively.

IV. PROPOSED METHOD

Existing literature has demonstrated that the UKF is an accurate and efficient optimizer for feedforward neural networks, and unlike training with gradient descent, does not require the calculation of derivatives [45]. Not requiring any linearization, the UKF is considered more suitable for systems with unknown linearities or when computing derivatives proves cumbersome [45].

The novel SIF estimation strategy is considered more robust than the UKF in the presence of modeling uncertainties or disturbances [29]. In this work, the SIF has been implemented into a single hidden layer feedforward neural network to observe the performance of this estimation method in comparison to the UKF and GD as an optimizer for training neural networks. The SIF will train the network's weights, treating them as a state vector with the main goal of minimizing the cost function stated in equation (3).

A. Experimental Setting

In this section, the experimental evaluation of the three different optimizers is discussed and the procedures involved. To present a detailed comparison, the optimizers implemented into the NN predicted non-linear time series data. The simulation was coded in Python.

The Mackey-Glass non-linear time delay differential equation was used to generate the dataset for the NN with 1000 samples. For the purposes of this experiment, the differential equation produces a one dimensional periodic and chaotic data series. The parameter for chaos induces mild chaos at 17 and moderate chaos at 30. The chaos parameter is a measure of the disorder of the produced data but is based on the principle that simple deterministic laws can exhibit complex external behaviour [46]. This is the behaviour we are trying to predict with SIF-trained NN.

The NN was constructed with a single hidden layer consisting of 13 neurons. The initial weights were determined using the open-source neural network Python library, Keras. As an optimizer, the SIF and UKF were coded and implemented, whereas the SGD was executed by using a Python deep learning library. The structure of the UKF was inspired by *Wan et al.* with the initial parameters set as $P_0 = 0.1I$, $Q = 0.001I$, $R = 0.001$ where I is the identity matrix of dimensions $n \times n$ where n is the number of states [47]. The formulation of the SIF was obtained by Gadsden *et al.* and reconstructed as a NN optimizer. The covariances were initialized the same way as the UKF was in its respective network (P_0, Q, R). Other parameters include $\eta = 0.01$ and δ, which was generated via a pseudo-random number generator which selected values between 0.0009 and 0.9, both of which were determined through manual tuning of the network.

However, we note that instability during training was quite high, to reduce stability we integrate stage selection for both non-linear models, where a batch of weight predictions are made every step during the training procedure and the set of weights that leads to the lowest loss is selected for the next step.

V. RESULTS AND DISCUSSION

The proposed model was evaluated based on selected metrics, the first of which is the mean squared error (MSE), which is the average squared variance between the actual and estimated value. The proposed SIF-trained model yielded an average MSE of 0.0095, with a minimum of 0.006 after 10 trials. An example of the model output is plotted against true validation data in Fig. 2. In addition to examining the MSE, the computational cost was analyzed through execution time. The total training time taken for convergence was an average of 1.55 minutes.

For comparison purposes, the UKF and SGD implementations were tested using the same validation set. The UKF implementation's estimates are illustrated in Fig. 3. This network resulted in an average MSE of 0.009 and an average training time of 4.93 minutes over 10 trials**.** The SGD implementation achieved an average MSE of 0.008, with a minimum of 0.002 over 10 trials with an average training time of 8.08 seconds. The SGD trained neural network's output can be seen in Fig. 4.

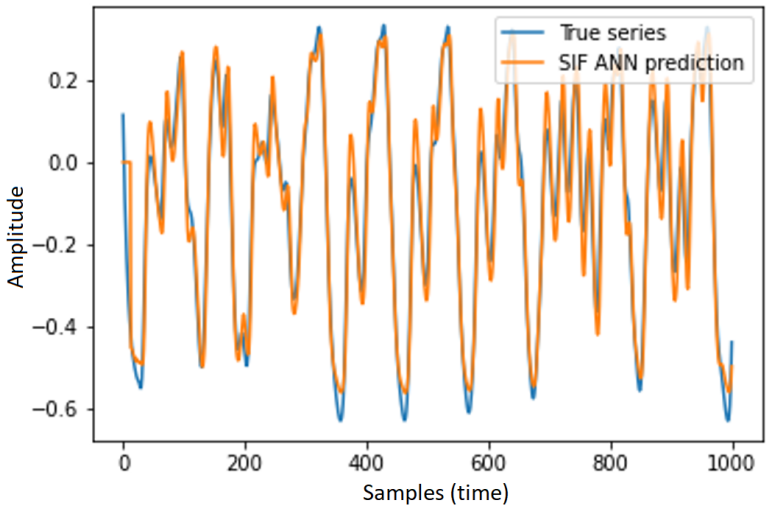

Fig. 2. True series and estimates of the Mackey-Glass chaos function from a SIF-trained neural network.

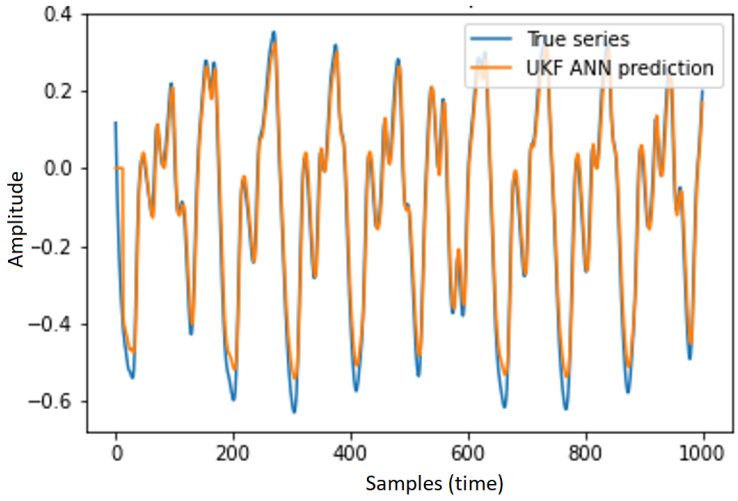

Fig. 3 True series and estimates of the Mackey-Glass chaos function from a UKF-trained neural network.

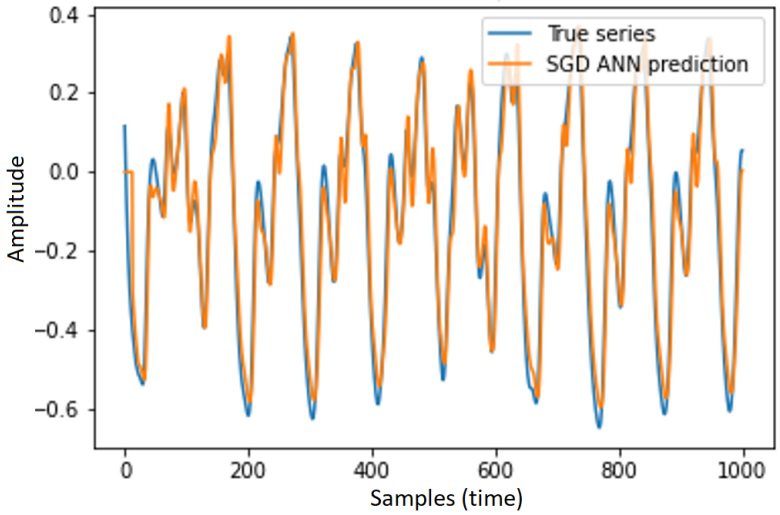

Fig. 4. True series and estimates of the Mackey-Glass chaos function from a SGD-trained neural network.

TABLE I
COMPARISON OF RESULTS

Training model	MSE	Average Computation Time
SIF	0.0095	1.55 min
UKF	0.009	4.93 min
SGD	0.008	8.08 s

The examination of the results indicates an overall excellent performance on behalf of the model. The proposed model was able to achieve accurate predictive capabilities while maintaining practical execution time. However, the UKF on average slightly outperformed the proposed model in terms of accuracy. Despite a somewhat lower accuracy, the proposed model generated a lower computational cost than the UKF implementation. Furthermore, the results of the SGD indicate a higher average accuracy of the model with a lower computational cost. The results show that there is a vivid trade-off between minimizing computational cost and increasing model accuracy with the non-linear estimation models. The SGD implementation had an overall better performance than both non-linear estimation models. While the SIF did not behave completely as expected, the results were promising and indicate that the SIF can be reliably utilized for such purposes.

VI. CONCLUSION

A novel method for training neural networks using an estimation strategy known as the sliding innovation filter (SIF) was proposed in this paper. The SIF was proven to be superior to the unscented Kalman filter (UKF) in terms of estimation robustness and performs better in the presence of modelling uncertainties, disturbances or error. Neural networks have been predominantly trained by gradient descent (GD) algorithms, and there have been successful efforts which utilize the UKF for training as well. Based on these facts, the SIF was implemented as an optimizer to train the neural network's parameters such as the weights and biases. The neural network was trained on the Mackey-Glass Chaotic time series in order to determine the model's performance in predicting the output. Results showed that the SIF yielded an estimate that was slightly less accurate than the UKF as a neural network optimizer. However, the SIF-trained network was able to converge much faster than the UKF-trained network. The SIF did not display improvement over SGD for training networks, however the results from this research prove that it has promising potential for applications other than just an estimation strategy.

VII. REFERENCES

[1] E. Sitompul, "Adaptive Neural Networks for Nonlinear Dynamic Systems Identification," in *Fifth International Conference on Computational Intelligence, Modelling and Simulation*, Seoul, South Korea, 2013.

[2] S. S. Du, J. D. Lee, H. Li, L. Wang and X. Zhai, "Gradient Descent Finds Global Minima of Deep Neural Networks," in *36th International Conference on Machine Learning*, Long Beach, California, 2019.

[3] Y. Li and Y. Liang, "Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data," in *Advances in Neural Information Processing Systems 31*, Montreal, Canada, 2018.

[4] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Interactive Computer Aided Learning*, Budapest, Hungary, 2017.

[5] N. B. Karayiannis, "Reformulated Radial Basis Neural Networks Trained by Gradient Descent," *IEEE Transactions on Neural Networks,* vol. 10, no. 3, pp. 657-671, 1999.

[6] D. J. Lary and H. Y. Mussa, "Using an extended Kalman filter learning algorithm for feed-forward neural networks to describe tracer correlations," *Atmospheric Chemistry and Physics,* vol. 4, no. 3, pp. 3563-3667, 2004.

[7] R. G. J. Wijnhoven and P. H. N. de With, "Fast Training of Object Detection using Stochastic Gradient Descent," in *20th International Conference on Pattern Recognition*, Istanbul, Turkey, 2010.

[8] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 279-297, 1994.

[9] S. Chen and S. A. Billings, "Neural networks for nonlinear dynamic system modelling and identification," *International Journal of Control,* vol. 56, no. 2, pp. 319-346, 1992.

[10] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks,* vol. 5, no. 2, pp. 157-166, 1994.

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010.

[12] B. Cetin, J. Burdick and J. Barhen, "Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks," in *IEEE International Conference on Neural Networks*, San Francisco, 1993.

[13] J. L. Brierton, "Techniques for avoiding local minima in gradient-descent-based ID algorithms," in *Proc. SPIE 3066, Radar Sensor Technology II*, Orlando, 1997.

[14] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14, pp. 76-86, 1992.

[15] J. T.-H. Lo, "Solving the Local-Minimum Problem in Training Deep Learning Machines," in *International Conference on Neural Information Processing*, Guangzhou, China, 2017.

[16] F. Ahmad, "Performance Comparison of Gradient Descent and Genetic Algorithm Based Artificial Neural Networks Training," in *10th International Conference on Intelligent Systems Design and Applications*, Cairo, Egypt, 2010.

[17] V. Ranganathan and S. Natarajan, "A New Backpropagation Algorithm without Gradient Descent," *arXiv,* 2018.

[18] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein and W. Maass, "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets," *arXiv Neural and Evolutionary Computing,* 2019.

[19] S. Benromdhane, "Convergence to Satisfactory Minima of the Kalman Filter Algorithm for Supervised Learning," in *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, US, 1995.

[20] D. Katic and S. Stankovic, "Fast learning algorithms for training of feedforward multilayer perceptrons based on extended Kalman filter," in *Proceedings of International Conference on Neural Networks*, Washington, DC, USA, 1996.

[21] S. Li, "Comparative analysis of backpropagation and extended Kalman filter in pattern and batch forms for training neural networks," in *International Joint Conference on Neural Networks*, Washington, DC, USA, 2001.

[22] Z. Darojah and E. S. Ningrum, "The extended Kalman filter algorithm for improving neural network performance in voice recognition classification," in *International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Lombok, Indonesia, 2016.

[23] H. E. U. H. H. P. C. Yanling Hao College of Automation, Z. Xiong, F. Sun and X. Wang, "Comparison of Unscented Kalman Filters," in *International Conference on Mechatronics and Automation*, Harbin, 2007.

[24] J. J. J. LaViola, "A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion," in *American Control Conference*, Denver, 2003.

[25] R. v. d. Merwe and E. A. Wan, "Efficient Derivative-Free Kalman Filters for Online Learning," in *European Symposium on Artificial Neural Network*, Bruge, 2001.

[26] Z. Darojah, E. S. Ningrum and D. S. Purnomo, "The Training of Feedforward Neural Network Using The Unscented Kalman Filter for Voice Classification Application," in *International Electronics Symposium on Knowledge Creation and Intelligent Computing*, Surabaya, 2017.

[27] R. Zhan and J. Wan, "Neural Network-Aided Adaptive Unscented Kalman Filter for Nonlinear State Estimation," *IEEE Signal Processing Letters,* pp. 445-448, 2006.

[28] M. C. Turner and D. G. Bates, Mathematical Methods for Robust and Nonlinear Control, Leicester: Springer, 2007.

[29] S. A. Gadsden and M. Al-Shabi, "The Sliding Innovation Filter," vol. Volume: 8, pp. 96129 - 96138, 19 May 2020.

[30] G. I. Webb, Naive Bayes, Boston, MA: Springer US, 2010, pp. 713--714.

[31] S. Singhal and W. Lance, "Training Multilayer Perceptron with the Extended Kalman Algorithm," pp. 133-140, 1988.

[32] C. M. Takenga, K. R. Anne, K. K. and J. C. Chedjou, "Comparison of gradient descent method, Kalman filtering and decoupled Kalman in training neural networks used for fingerprint-based positioning," Los Angeles, 2004.

[33] S. A. Gadsden, R. Ahmed, M. El Sayed and J. Tjong, "Artificial neural network training utilizing the smooth variable structure filter estimation strategy," pp. 537-548, 2015.

[34] D. Svozil, V. Kvasnicka and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems,* vol. 29, pp. 43 - 62, 1997.

[35] N. K. Manaswi, "Multilayer Perceptron," in *Deep Learning with Applications Using Python : Chatbots and Face, Object, and Speech Recognition With TensorFlow and Kera*, Berkeley, 2018, pp. 45-56.

[36] U. Michelucci, "Feedforward Neural Networks," in *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*, Berkeley, Apress, 2018, pp. 83-136.

[37] S. Bak, "Improved Geometric Path Enumeration for Verifying ReLU Neural Networks," in *Computer Aided Verification*, Springer International Publishing, 2020, pp. 66-96.

[38] N. Ketkar, Deep Learning with Python: A Hands-on Introduction, Bangalore: apress, 2017.

[39] Y. Lecun, L. Bottou, G. B. Orr and K.-R. Muller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, Berlin, Springer Berlin Heidelberg, 1998, pp. 9-50.

[40] C. C. Aggarwal, Linear Algebra and Optimization for Machine Learning, Yorktown Heights: Springer, 2020.

[41] Y. Zhang, F. Gao and L. Tian, "INS/GPS Integrated Navigation for Wheeled Agricultural Robot Basd on Sigma-point Kalman Filter," in *7th International Conference on System Simulation and Scientific Computing*, Beijing, China, 2008.

[42] A. Gadsden, S. Habibi, D. Dunne and T. Kirubarajan, "Nonlinear Estimation Techniques Applied on Target Tracking Problems," *Journal of Dynamic Systems, Measurement, and Control,* vol. 134, pp. 054501 (1-13), 2012.

[43] S. Julier, J. Uhlmann and H. F. Durrant-Whyte, "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators," *IEEE Transactions on Automatic Control,* vol. 45, no. 3, pp. 477-482, 2000.

[44] J. T. Ambadan and Y. Tang, "Sigma-Point Kalman Filter Data Assimilation Methods for Strongly Nonlinear Systems," *Journal of the Atmospheric Sciences,* vol. 66, no. 2, pp. 261-285, 2009.

[45] X. Zhao, J. Yu and F. Li, "UKF-Based Training Algorithm for Feed-Forward Neural Networks with Application to XOR Classification Problem," in *8th International Conference of Natural Computation*, Chongqing, China, 2012.

[46] M. Farzad, H. Tahersima and H. Khaloozadeh, "Predicting the Mackey Glass Chaotic Time Series," in *SICE-ICASE International Joint Conference*, Busan, South Korea, 2006.

[47] E. A. a. v. d. M. R. Wan, "The Unscented Kalman Filter," in *Kalman Filtering and Neural Networks*, John Wiley & Sons, Ltd, 2002, pp. 221-280.

[48] D. Ruck, "Comparative analysis of backpropagation and the extended Kalman filter for training multilayer perceptrons," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 14, no. 6, pp. 686-691, 1992.

[49] C. K. Chui and G. Chen, Kalman Filtering with Real-Time Applications, Springer Series in Information Sciences, 2009.

[50] Y. Kim and H. Bang, "Introduction to Kalman Filter and Its Application," in *Introduction and Implementations of the Kalman Filter*, IntechOpen, 2018.

[51] R. Williams, "Training recurrent networks using the extended Kalman filter," in *International Joint Conference on Neural Networks*, Baltimore, MD, USA, 1992.

[52] W. Pietruszkiewicz, "A Comparison of Nonlinear Kalman Filtering Applied to Feed–forward Neural Networks as Learning Algorithms," in *IEEE 9th International Conference on Cyberntic Intelligent Systems*, Reading, UK, 2010.