Digital Signal Processing 135 (2023) 103966

Contents lists available at ScienceDirect

Digital Signal Processing

journal homepage: www.elsevier.com/locate/dsp

Intelligent estimation: A review of theory, applications, and recent advances

Naseem Alsadi [a], S. Andrew Gadsden [a,*], John Yawney [b]

[a] Department of Mechanical Engineering, McMaster University, Hamilton, ON, Canada
[b] Adastra Corporation, Toronto, ON, Canada

A R T I C L E I N F O

Article history:
Available online 10 February 2023

Keywords:
Estimation theory
Artificial intelligence
Machine learning
Deep learning
Kalman filter

A B S T R A C T

Recent developments in the field of deep learning have led to the widespread integration of artificial neural networks in various domains of application. Prominent contemporary artificial neural network training techniques are based on first-order gradient computation. The emphasis on algorithmic performance has driven the emergence of variant artificial neural network training methodologies. Estimation theory, traditionally considered a sub-field of statistics and signal processing, has been explored by various researchers for the development of non-gradient based training methods. Articles published with the aim of utilizing estimation-based artificial neural network training techniques have shown promising results. We identify the integration of estimation theory within the artificial neural network training procedure as intelligent estimation. In this paper, the field of intelligent estimation is analyzed in greater depth with emphasis on the algorithmic performance of novel implementations. Intelligent estimation with applications in the professional domain is also considered, and will help lay the foundation for future research in the literature.

1. Introduction

Estimation theory is a sub-section of the wide-ranging field of statistics and signal processing. Estimation methods attempt to predict, filter, or smooth the value of particular states of interest based on prior empirical data. The integration and application of these methods have spanned numerous fields, including signal processing, control systems and machine learning. These applications are exceptionally accurate and reliable, facilitating their utilization in high-risk environments, such as air traffic control [1]. To estimate system states and dynamics, estimation models are utilized [2–4]. The performance of these models when dealing with practical applications can be augmented with the introduction of non-linear flexibility [5] and measurement noise robustness [6–8]. The Kalman filter is a prominently utilized estimation method, used for various applications, especially target tracking problems. Since the development of the Kalman filter in the 1960s, its integration into complex optimization problems has increased, with recent developments in the field of artificial intelligence and machine learning [9].

With recent advances in deep learning, a major machine learning strategy, the employment of algorithms capable of precisely learning from prior data is occurring at an unprecedented rate. Deep learning algorithms utilize selected data to optimize network parameters to reduce output error [10,11]. Various factors influence the quality of the learning phase. The volume of training data and network architecture complexity play a crucial part in shaping the model learning characteristics [12]. The training algorithm, which is applied to optimize network parameters, is critical to regulating algorithmic computational complexity and storage requirements [13].

Network training algorithms are employed to locate model parameters that yield the minimum possible output error within the shortest time duration possible. These algorithms can be classified and assessed based on their generalization characteristics, which indicates how well the algorithm can perform provided variant datasets and model architectures, and their convergence rate. Algorithmic convergence characteristics can profoundly influence training time, positively and negatively. Contemporary research has been conducted on developing training algorithms that aim to deliver exceptional generalization and convergence characteristics.

Network training methods can be dichotomized into gradient-based and non-gradient based approaches [14]. Gradient-based approaches compute the impact that slight variations in the parameter value have on the output error. To do such, gradient-based

* Corresponding author.
E-mail addresses: alsadin@mcmaster.ca (N. Alsadi), gadsden@mcmaster.ca (S. Gadsden), john.yawney@adastragrp.com (J. Yawney).

https://doi.org/10.1016/j.dsp.2023.103966

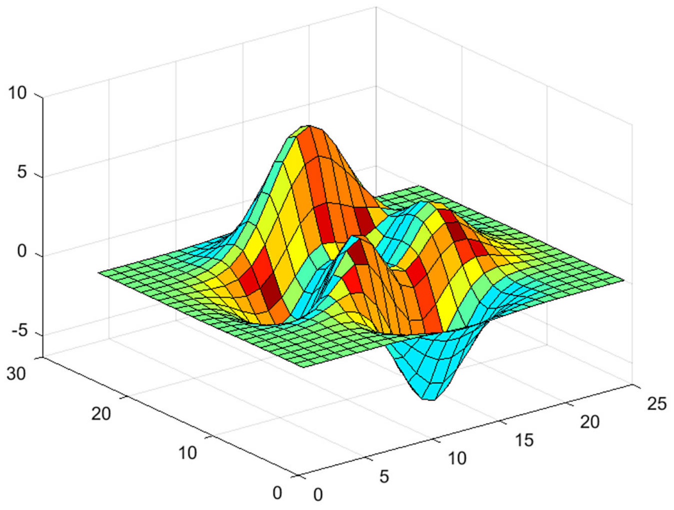

Fig. 1. Non-Convex Loss Function.

methods employ the gradient of the loss function with respect to the selected parameters of interest. The computation can be carried out in parallel, therefore speeding up execution time. Non-gradient based approaches avoid gradient calculations, such as genetic algorithms and The Nelder-Mead Method [15–17].

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 [18–20]. When employing a gradient descent-based optimization methodology for training a neural network, there are two common outcomes. The first of which is the non-convex nature of the training criterion and the second is the magnitude of non-global local minima [21,22]. These issues are visualized in the non-convex loss function displayed in Fig. 1. 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 [23]. 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 [24].

Additionally, researchers have taken steps to replace gradient descent by employing variant optimization functions. F. Ahmad et al. propose the utilization of genetic algorithms to replace gradient descent, with the aim of locating a global optimal solution in a complex space [25]. 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. There is plenty of literature discussing numerous substitutes for gradient descent [26,27].

Researchers have endeavoured to explore the use of non-gradient estimation-based training methods as a possible replacement for gradient-based methods [14]. Estimation-based network training techniques have proven to produce accurate and reliable models [28]. These techniques have been tested across various applications, achieving similar or better results than other tested training algorithms.

The fundamental motivation behind this literature review is the exploration and analysis of the various heterogeneous estimation techniques applied to neural network training as an efficient alternative to common contemporary methods, such as gradient descent. Over the last three decades, researchers have explored employing classical estimation techniques, like the Kalman filter and its variations, within the feedforward and recurrent neural networking training process.

This paper discusses the recent developments in intelligent estimation which includes any estimation-based techniques for the training of neural networks. Furthermore, contemporary trends in research and applications surrounding estimation theory and deep learning are considered. This paper provides a comprehensive survey of relevant research in the literature, and will serve as a foundation for future work. Section 2 gives a brief introduction to estimation techniques and backpropagation. Section 3 presents a full review of estimation-based network training techniques. The paper concludes with a discussion of the advantages and disadvantages of utilizing estimation-based training methods along with potential future applications and research.

2. Background

After systematically assessing the existing literature in the field of Intelligent Estimation, which entails the employment of estimation methods in the process of training neural networks, it is clear that there is a set of central concepts and topics that lay the foundation for the majority of developments in the field. We identify these as the basic Kalman filter, feedforward neural network and recurrent neural network. Therefore, to provide the necessary background information, we introduce these topics in detail in the sections below. Note that the section on the Kalman filter also discusses why Kalman filter variations and other estimation methods have also been utilized by researchers.

2.1. Kalman filter

The Kalman filter (KF) is a prominent state and parameter estimation algorithm developed by Rudolf E. Kalman, which aims at utilizing a series of measurements, obtained across a known temporal distance, to estimate the value of some unknown variables. The KF is commonly known as the optimal solution to various tracking and data prediction tasks [29]. An introduction to the formalization and utilization of the KF is presented below.

The KF is used to estimate states on the basis of linear dynamic systems in state-space format under the presence of white noise. The linear system can be represented with a state vector and a sensor yielded linear measurement matrix. The linear state vector and measurement matrix can be formalized as [30]:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{1}$$

$$z_{k+1} = Cx_{k+1} + v_{k+1} \tag{2}$$

where x_k is the system state at time step k, A is a constant linear system matrix, B is the input gain matrix, C is the linear measurement matrix. u_k is the system input at time step k, w_k is the system noise at time step k, and v_k is the measurement noise at time step k.

A nonlinear system can also be represented with the following state vector and measurement matrix [31]:

$$x_{k+1} = f(x_k, u_k) + w_k \tag{3}$$

$$z_{k+1} = h(x_{k+1}) + v_{k+1} \tag{4}$$

where f and h are nonlinear system and measurement models, respectively.

Filtering is the process of estimating the current or future state of a specified dynamic system, which includes the process of filtering out unwanted system noise [32]. The KF procedure is comprised of recursive stages, shaped in a predictor-corrector fashion. The first stage of the KF procedure is named the prediction stage, where the system states are estimated using a system model. The estimate achieved during the prediction stage is an a priori estimate, due to being yielded prior to any system measurement. The second stage of the KF is the Update stage, where the states are updated based on the measurement error. Estimates obtained during the Update stage will be a posteriori state estimates as a result of being produced after attaining system observations [33].

The KF is developed on a set of three core assumptions, namely a linear system model, known initial state values, and noise with zero mean Gaussian noise. The prediction stage can be formalized with the following a priori state estimate and error covariance [34]:

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{5}$$

$$P_{k+1|k} = AP_{k|k}A^T + Q_k \tag{6}$$

After the state prediction and error covariance matrix are produced during the respective Prediction stage, the update stage can be conducted with the following set of equations [35]:

$$K_{k+1} = P_{k+1|k}C^T(CP_{k+1|k}C^T + R_{k+1})^{-1} \tag{7}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - C\hat{x}_{k+1|k}) \tag{8}$$

$$P_{k+1|k+1} = (I - K_{k+1}C)P_{k+1|k}(I - K_{k+1}C)^T + K_{k+1}R_{k+1}K_{k+1}^T \tag{9}$$

where P is the approximate error covariance matrix, Q is the covariance of the process noise and R is the covariance of the observation noise.

The procedure is repeated iteratively, flowing from the prediction to the update stage. If the fundamental assumptions listed prior are not met, the KF becomes sub-optimal and may even become unstable. However, expecting a perfect linear system in a pragmatic implementation is unreasonable, especially when dealing with real physical systems [36]. Variations of the KF have been developed to produce better results when dealing with nonlinear systems, such as the extended Kalman filter (EKF) and unscented Kalman filter (UKF) [37–39,31,40]. Other methods have been developed to improve KF-based methods in terms of overall accuracy or robustness to disturbances and uncertainties. These methods include the H-infinity filter [41], particle filter [42], and variable structure methods such as the smooth variable structure filter (SVSF) [43] and sliding innovation filter (SIF) [44]. The Kalman filter has been employed for a variety of purposes and it's reliability and efficiency have made it a popular choice for it's integration into the field of machine learning, which will be shown throughout the remainder of this paper. For instance, Gaussian process regression models, which has made large waves in the field of machine learning, can be reformulated as a linear-Gaussian state space model, therein allowing them to be solved with a Kalman filter. The advantage of doing such is a dramatic reduction in the computational complexity of the model in comparison with direct Gaussian process regression methods [45–49].

2.2. Feedforward neural networks

Neural networks are algorithms which are developed to establish a relationship between a selected set of data input and output. These algorithms are designed to mimic the biological neural connections found in the human brain [13]. The most fundamental variation of the neural network is composed of three layers, namely a input, hidden and output layer with a unidirectional flow of data. This form of neural network, displayed in Fig. 2, is termed the feedforward neural network (FFNN) [50].

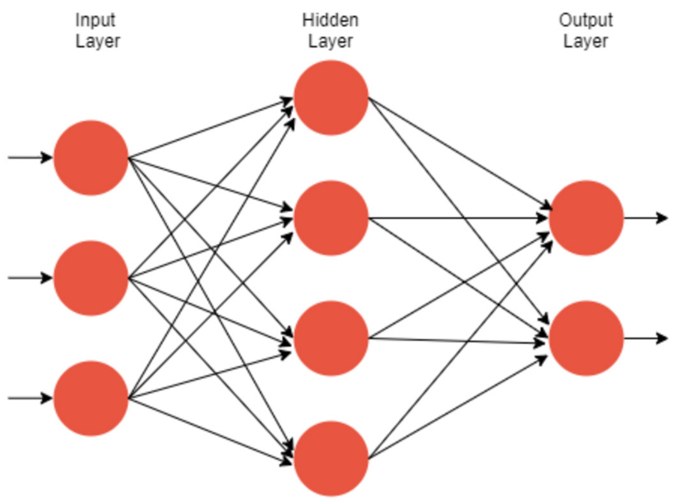

Fig. 2. A Typical Feedforward Neural Network.

The prominent feedforward neural network training procedure, gradient descent, utilizes a gradient based approach to perform iterative updates to the network weights. Each iteration aims at improving network output and minimizing error. The training procedure begins with the initialization of network weights and biases. This random initialization process decreases the probability of early local convergence. Subsequently, a network output signal is produced and an output error is computed. Local gradients are then calculated and the network weights are respectively updated. This process repeats and the weights are continuously modified [51,52]. The cumulative feedforward neural network training procedure is detailed below.

Each layer of the network architecture is composed of various neurons. These neurons are fired with a selected activation function [53,54], which will produce a lower limit output when the input does not meet the threshold and an upper limit output when the input does. The rectified linear activation function, ReLU, is one of the most prominently utilized functions. The ReLU function is an identity function if the input is positive, otherwise it returns zero [55]:

$$ReLU = max(x, 0) \tag{10}$$

Each neuron will yield a preliminary output, z, which is a function of the input, x, achieved via the network's weights and the connecting neuron's output values and bias, b. The preliminary output of the neuron, which will be the input to the selected activation function can be formalized as such:

$$z = w^T x + b \tag{11}$$

The learning procedure is then employed to locate the optimal set of network weights with the aim of reducing the identified cost function. The gradient-based training algorithm, stochastic gradient descent, which may be the most widely-used network training algorithm, will update the weights on each iteration of the algorithm on the basis of the average gradient on a mini-batch of m samples [11].

$$g = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)}; w), y^{(i)}) \tag{12}$$

The computed gradient is then multiplied by a learning rate. ϵ, which regulates the degree of change in the current weight values. The learning rate is widely held to be the most important hyperparameter [11]. Selecting the optimal learning rate can pose various difficulties. A learning rate which is too small will slow down the learning phase, and a learning rate too high might skip over a desired local minima. The a priori computation of the optimal learning rate is not generally possible, further complicating the selection process [56]. A few methods can be used to attempt to locate an optimal learning rate. Manually tuning the learning rate according to obtained results is a method commonly utilized. Manually performing tuning, however, is time consuming and inefficient. Another method commonly utilized to locate an optimal learning rate is grid search. Grid search involves the use of a finite set of hyperparameter values, each of which will be used to train separate models. Subsequently, the performance of the models will be compared and the learning rate which generated the best performance is selected. Typically the finite set of learning rate values is produced on a logarithmic scale, encompassing a range of values such as 10^{-5} to 10^{-1} [11].

$$w_{k+1} = w_k - \epsilon g \tag{13}$$

The described learning procedure is repeated until a stopping criteria is met, whether that be a selected number of iterations, epochs, or a specified threshold metric [57].

2.3. Recurrent neural networks

Recurrent neural networks are a category of artificial neural networks where a sequential connection is establish among nodes. Nodes will be connected to other nodes within variant layers in a unidirectional fashion. The flow of data is unidirectional, therefore the recurrent neural networks will contain input, hidden, and output layers. The temporal flow of data from node to node allows previous outputs to be utilized as input for successive nodes. As a result, information from prior input is compiled and transferred to subsequent nodes, allowing for the model to dynamically learn from the past [58–60].

Classical neural networks work well on the presumption that the input and output are directly independent of each other, however, this is not always the case. This is crucial to the implementation of the proposed method and will be discussed in greater detail below [61–64].

Fig. 3 shows a labeled recurrent neural network diagram. At each time step t, a new input x_t is provided to the network. The hidden section of the network, which compiles and stores all previous data, is calculated as [59]:

$$h_t = f(W_x x_t + W_h h_{t-1}) \tag{14}$$

Conclusively, the output y_t is sequentially generated at each node with:

$$y_t = g(W_y) \tag{15}$$

where $g(x)$ represents the selected activation function.

3. Estimation and feedforward neural network training

The first estimation method included in the artificial neural networks training process was in 1989, and the employment of various heterogeneous estimation methods directly followed. Kalman filters were the first estimation method to be utilized in the neural network training process. Since then, the H-Infinity, Smooth Variable Structure, Particle, and Sliding Innovation filters have been used. In addition, diverse variations of these filters were utilized to further augment the method's performance given the specific task of neural network training. A cumulative diagram of all the estimation methods employed in the field is seen in Fig. 4.

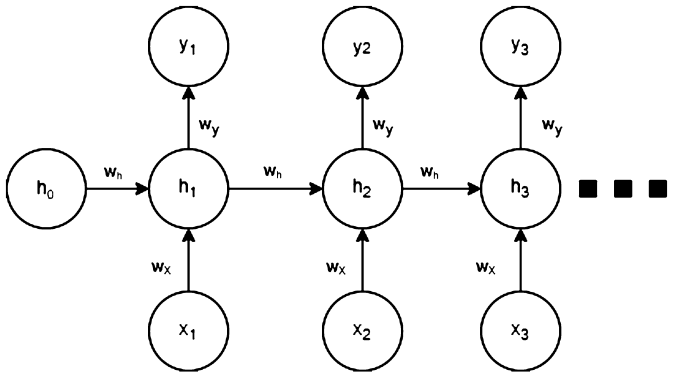

Fig. 3. A Typical Recurrent Neural Network Setup.

3.1. Kalman filters

In the late 1980s, Singhal and Wu presented an estimation-based feedforward network training technique as an effective method for circumventing the issues encountered by numerous other training approaches (Table 1), in specific backpropagation [65]. In their seminal paper, the authors postulate that although backpropagation as a training algorithm has become widespread, complexity will ultimately bound convergence performance. According to them, the cumulative learning procedure can be modeled as an "identification problem for a nonlinear dynamic system which can be solved using the extended Kalman algorithm." Their paper states that introducing the EKF can decrease overall convergence time. They conduct a comparative study on the EKF and classical backpropagation algorithm based on their convergence characteristics, reporting faster convergence and lower average error for the EKF. The training procedure can then be defined as locating the minimum mean square error (MMSE) of the state w utilizing previous observations. The training problem can therefore be stated as [65]:

$$\hat{w}_{k+1} = \hat{w}_k + K_k E_k \tag{16}$$

$$P_{k+1} = P_k - K_k H_k^T P_k \tag{17}$$

$$K_k = P_k H_k [R_k + H_k^T P_k H_k]^{-1} \tag{18}$$

$$E = y_k - \hat{y}_k \tag{19}$$

where $\hat{w}_k$ represents the network weights at update step at k. K is the Kalman gain matrix. y_k is the target vector, $\hat{y}_k$ is the networks output vector and E_k is the error vector. P_k is the approximate error covariance matrix. H_k is the gradient matrix.

A five step procedure is reported during each training epoch. The input vector is initially propagated through the network and the respective output vector is produced. Subsequently, the derivative matrix H_k is calculated using backpropagation. Next, the Kalman gain matrix K_k is computed utilizing the approximate error covariance matrix P_k, measurement covariance matrix R_k and the calculated derivative matrix H_k. The state vector is updated with the Kalman gain matrix K_k, error vector E_k and current state vector w_k. Lastly, the approximate error covariance matrix is updated as a function of Kalman gain matrix K_k, the derivative matrix H_k and current approximate error covariance matrix P_k. The signal flow diagram for the training procedure is shown in Fig. 5.

To evaluate the performance of the algorithm, Singhal and Wu develop a four quadrant classification problem and compared the training results of the proposed algorithm with standard backpropagation. The outcome of the experiment displayed the superior performance of the model by achieving a lower average output error than standard backpropagation.

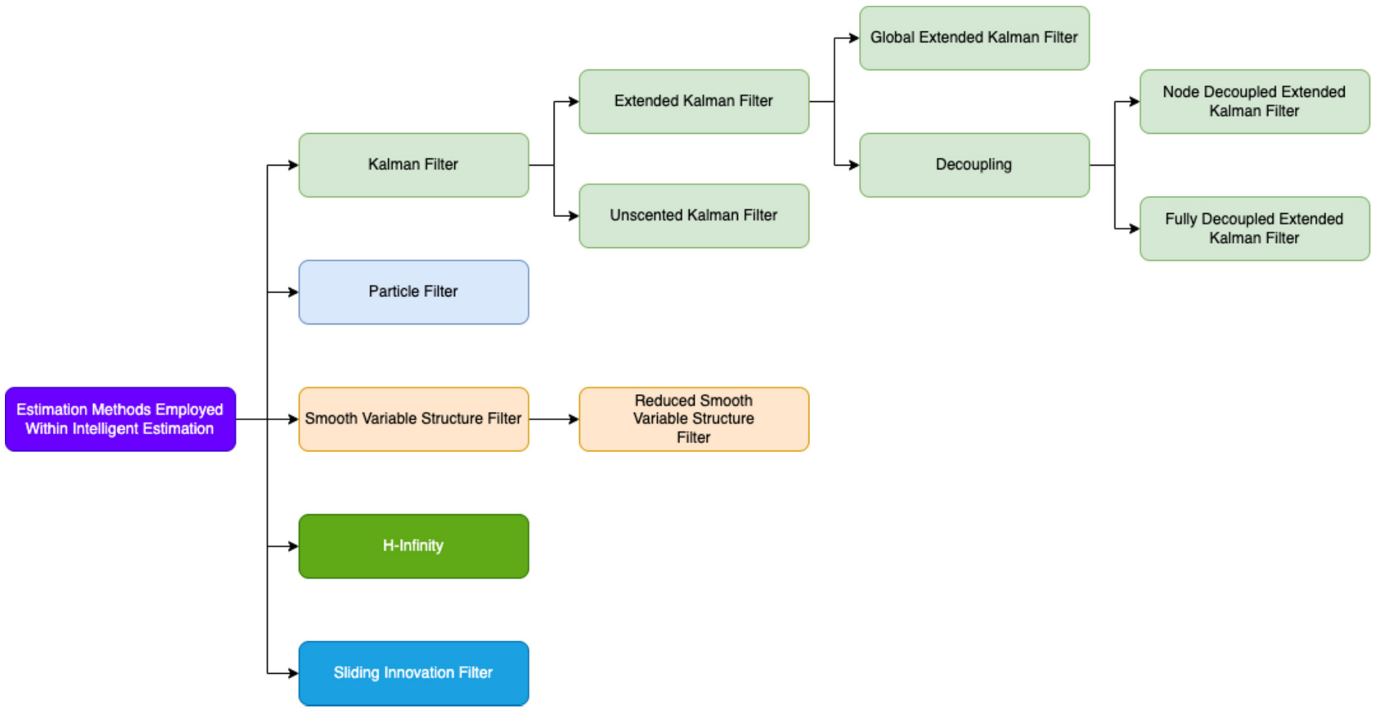
Estimation Methods Employed Within Intelligent Estimation
Kalman Filter
Extended Kalman Filter
Global Extended Kalman Filter
Decoupling
Node Decoupled Extended Kalman Filter
Fully Decoupled Extended Kalman Filter
Unscented Kalman Filter
Particle Filter
Smooth Variable Structure Filter
Reduced Smooth Variable Structure Filter
H-Infinity
Sliding Innovation Filter

Fig. 4. Summary of Intelligent Estimation Methods: Common Estimation Strategies used for Training of Neural Networks.

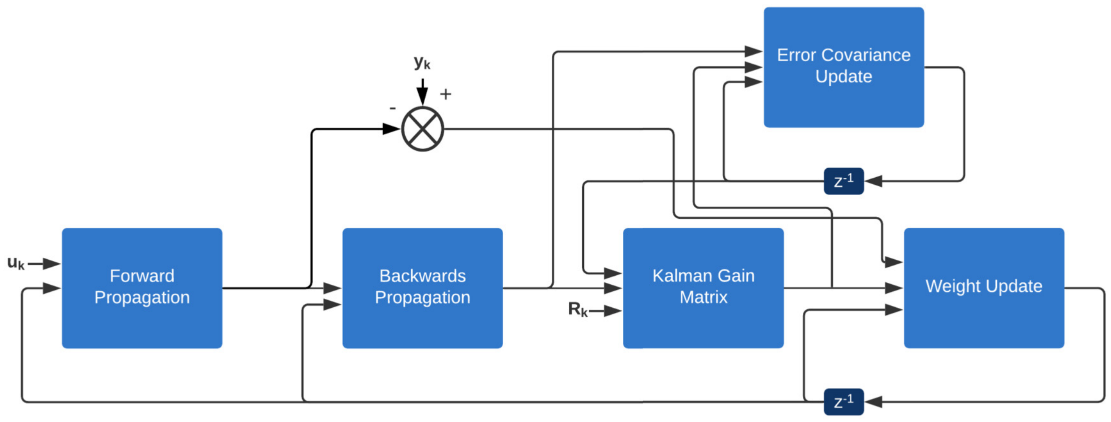

Fig. 5. Signal Flow Diagram for the Global Extended Kalman Filter.

The algorithm introduced by Singhal and Wu, termed the global extended Kalman filter (GEKF), yielded significant advantages in convergence speed; however, algorithmic computational complexity limited practical implementation. The computational complexity and storage requirement of the GEKF is directly associated to the number of model weights utilized in the architecture. Therefore, when implementing the GEKF, a decrease in model performance is expected as model architecture expands. Contemporary neural network applications will typically require deep and extensive model architectures, which, as discussed prior, yield a non-ideal environment for the employment of the GEKF. Consequently, researchers have endeavoured to modify the GEKF to avoid the aforementioned complexity and storage setbacks [66].

The GEKF entails a global update of the network weights. While being a crucial feature of the algorithm, this global update has led to an augmentation of algorithmic computational complexity. To solve this problem, Shah et al. suggest decomposing the global update policy into a collection of sub-tasks or sub-problems. The authors established a series of algorithms to update each neuron's weights independently at each time step to minimize the overall cost [67].

Shah et al. developed the neuron-level extended Kalman algorithm (NEKA) approach based on the critical observation that weights belonging to the same neuron displayed a higher correlation than weights that do not [68]. NEKA simplifies the GEKF into a set of decoupled algorithms. Each neuron can thus be assumed to be a sub-problem of the cumulative learning procedure. The NEKA approach can be formalized as:

$$\hat{w}^i_{k+1} = \hat{w}^i_k + K^i_k E_k \tag{20}$$

$$P^i_{k+1} = P^i_k - K^i_k (H^i_k)^T P^i_k \tag{21}$$

$$K^i_k = P^i_k H^i_k [I + (H^i_k)^T P^i_k H^i_k]^{-1} \tag{22}$$

$$E = y_k - \hat{y}_k \tag{23}$$

where w^i_k represents the ith group's network weights at update step at k. K is the Kalman gain matrix. y_k is the target vector, $\hat{y}_k$

Table 1
Notable Approaches to Intelligent Estimation.

Authors	Publication Year	Fundamental Estimation Technique
S. Singhal and L. Wu **(GEKF)** [65]	1989	Extended Kalman Filter
G. V. Puskorius and L. A. Feldkamp **(DEKF)** [66]	1991	Extended Kalman Filter
S. Shah, F. Palmieri, and M. Datum **(NEKA/MEKA)** [67]	1992	Extended Kalman Filter
R. Zhan and J. Wan [69]	2006	Unscented Kalman Filter
He-Sheng Tang, Songtao Xue and Tadanobu Sato [70]	2007	H-infinity Filter
R. Ahmed, M. Sayed, S. A. Gadsden, J. Tjong and S. Habibi [71]	2016	Smooth Variable Structure Filter
Z. Darojah, E. S. Ningrum and D. S. Purnomo [72]	2017	Unscented Kalman Filter
M. Nazari and S. Shamshirband [73]	2020	Particle Filter
N. Alsadi, W. Hilal, O. Surucu and S. A. Gadsden **(SIFNN)** [74]	2022	Sliding Innovation Filter

Table 2
Comparison of Computational Complexity Per Iteration and Network Storage.

Algorithm	Operation Count	Storage Cost
BP	184	33856
GEKF	280996	1784
NEKA	22070	1784
MEKA	8074	1784

is the networks output vector and E_k is the error vector. P_k is the approximate error covariance matrix and H_k is the gradient matrix.

When employing the NEKA approach, each neuron will contain its own Kalman gain matrix and covariance matrix. While the approach improves on the issues faced by the GEKF, each neuron, at every time step, will be required to perform an LxL matrix inversion, where L is the output vector dimension. This process is inherently computationally expensive. The multiple extended Kalman algorithm (MEKA) is an algorithm advanced by Shah et al. that builds upon a local approach to the EKF. The MEKA is quite similar to the NEKA; however, it does not require a matrix inversion at each neuron. The MEKA is given by:

$$(q_k^i)^T = y_k^i(1-y_k^i)u_k^i \tag{24}$$

$$\hat{w}_{k+1}^i = \hat{w}_k^i + K_k^i E_k \tag{25}$$

$$P_{k+1}^i = P_k^i - K_k^i (q_k^i)^T P_k^i \tag{26}$$

$$K_k^i = P_k^i q_k^i [1 + (q_k^i)^T (H_k^j)^T P_k^i q_k^i]^{-1} \tag{27}$$

$$E = y_k - \hat{y}_k \tag{28}$$

Shah et al. compared the performance of the GEKF, NEKA and MEKA. The authors report that the GEKF achieved the lowest error in all test cases; however, it yielded a considerably large computation and storage cost compared to the other algorithms, evident in Table 2. The MEKA achieved the overall best results by substantially reducing computation and storage cost and obtaining a lower error than NEKA. Shah et al. conclude, perhaps with some bias, that MEKA is "among the most reliable algorithms for supervised learning." However, they admit that while MEKA has achieved lower computational cost than the other algorithms, there are significant opportunities for improvement.

The comparative study conducted by Shah et al. vividly depicts the improvements in computation and storage cost due to the global update policy's partitioning, be it NEKA or MEKA. Yet when it came to analyzing output error, the partitioning of the global update policy did not fare too well. Both the NEKA and MEKA had higher average output error than the GEKF. The partitioning process itself may be a causative reason for the augmentation in output error. Puskorius et al. point to this observation and propose that the treatment of individual neurons as subsystems, while improving computational complexity, can ultimately lead to inadequate model output performance [75].

As discussed previously, the fundamentally computationally expensive aspect of the GEKF lies within the set of network weights' consistent and uniform treatment. This uniform treatment was the core aspect that Shah et al. desired to build on to circumvent computational and storage issues encountered by the GEKF. Puskorius et al. propose another strategy, comparable to the algorithms propounded by Shah et al., named the decoupled extended Kalman filter (DEKF). The core feature of the DEKF, which distinguishes it from alternative approaches, is its capability to group network weights. The grouping, or decoupling, of network weights, permits the disregarding of distinct elements in the error covariance matrix, namely elements that stem from weights in separate groups. As a directly inherent consequence of decoupling, the larger the number of groups utilized, the larger the decline in computation and storage cost. Thus, the DEKF can be said to incorporate the dynamism and capacity for complexity regulation [76–78].

In their 1991 paper, Puskorius et al. present two variations of the DEFK algorithm. The first of the two is the node decoupled EKF (NDEKF), where the grouping of network weights is conducted based on node association. The second variation of the DEKF put forth by Puskorius et al. is the fully decoupled EKF (FDEKF), where each group comprises an individual network weight. The DEKF algorithm can be expressed with [79]:

$$\hat{w}_{k+1}^i = \hat{w}_k^i + K_k^i E_k \tag{29}$$

$$P_{k+1}^i = P_k^i - K_k^i (H_k^i)_k^T P_k^i \tag{30}$$

$$K_k^i = P_k^i H_k^i [R_k + \sum_{j=1}^{N} (H_k^j)^T P_k^j H_k^j]^{-1} \tag{31}$$

$$E = y_k - \hat{y}_k \tag{32}$$

where N is the quantitative representation of the degree of decoupling. For the NDEKF approach, N is equal to the number of nodes and the FDEKF approach, N is set to the number of weights in the model. Note that when N is set to 1, the DEKF is reduced back to the GEKF. From this angle, the DEKF can be viewed as a unique instance of the GEKF.

It is crucial to note that unlike the local approaches discussed prior, NEKA and MEKA, the DEKF computes a global scaling matrix. The global scaling matrix is calculated for all weight groups rather then uniquely for each weight group. The non-local approach, however, does not negatively impact the overall computational complexity of the DEKF and the weight group updates can be conducted in parallel. As discussed prior, the utilization of the DEKF will allow for the reduction in the error covariance matrix. Fig. 6 shows the representation of the error covariance matrix as a result of using the GEKF and DEKF variations. The computational complexity (Table 3) of the GEKF has been reduced by a factor of 8 as a result of implementing the NDEKF.

The authors compare the performance of the GEKF, NDEKF, FDEKF and standard backpropagation, utilizing variant classification tasks. Fig. 7 illustrates how the algorithms performed on Singhal and Wu's original 4-quadrant classification problem. The training result graph indisputably shows the superior performance

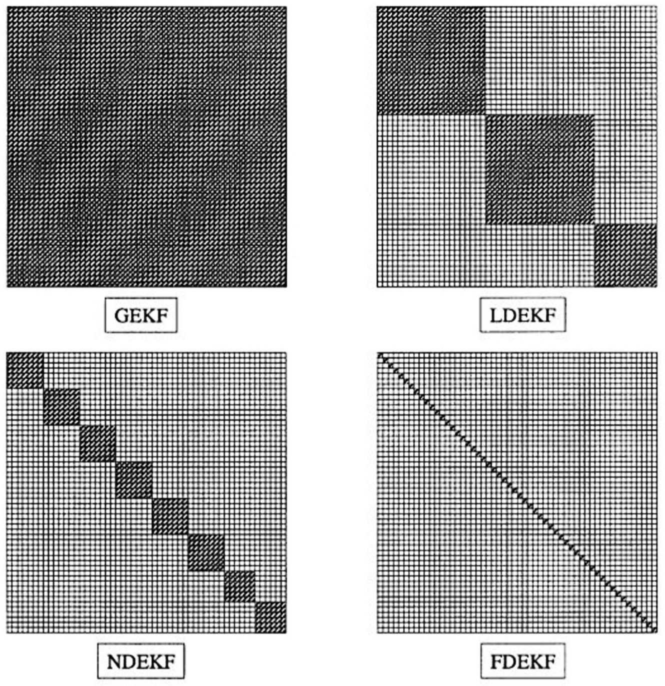

Fig. 6. Block-diagonal Representation of the Approximate Error Covariance Matrix for Variant Levels of Decoupling. The computational complexity of the GEKF has been reduced by a factor of 8 as a result of implementing the NDEKF [29].

Table 3
Comparison of Computational Complexity.

Algorithm	Computational Complexity
BP	$\mathcal{O}(MS)$
GEKF	$\mathcal{O}(M^2S^2)$
NEKA	$\mathcal{O}(M^2S)$
MEKA	$\mathcal{O}(M^2S)$
NDEKF	$\mathcal{O}(S^2N+\sum_{i=1}^{N} m_i)$
FDEKF	$\mathcal{O}(S^2N+M)$

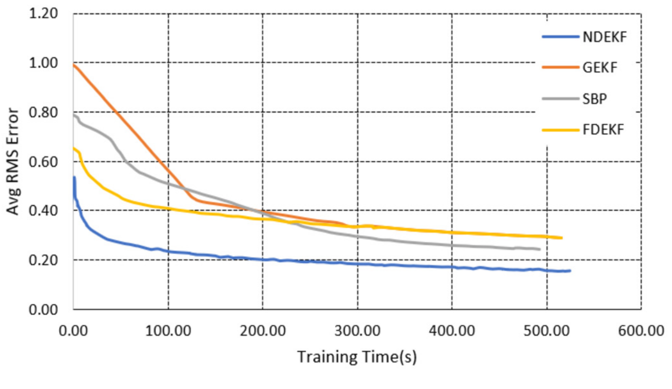

Fig. 7. Puskorius' Reported RMSE for GEKF, NDEKF, FDEKF, SBP.

of the NDEKF, achieving the lowest average error and fastest convergence time amongst the sampled algorithms.

The NDEKF and FDEKF performed excellently, with the NDEKF, on average, attaining exceptional convergence characteristics, surpassing all algorithms, including standard backpropagation. The authors state, "NDEKF's trained networks exhibit superior generalization and function approximation capabilities. Even in the limit of complete decoupling of weights (FDEKF), performance was often comparable to or better than SBP."

The unscented Kalman filter (UKF) is a sigma point Kalman filtering method proposed by Julier et al. to generalize the KF to non-linear systems and measurements [38]. The UKF utilizes a deterministic sampling methodology, namely the unscented transformation, to choose a set of sigma points around the mean. The selected sigma points are therein used to produced a new covariance and mean estimate [80]. The UKF removes the necessity for the linearization of the system model by determining the Jacobian matrices. The calculation of the Jacobian matrices is computationally expensive [81]. Literature shows that the use of unscented transforms provides a more accurate and reliable estimate of the system state covariance and mean than the EKF [82].

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 UKF yields better results than the EKF [37]. 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 use of Jacobian matrices [83]. 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. In addition to improved performance and non-complex implementation, the UKF does not increase computational complexity, rather it can operate at the same level of computational complexity as the EKF [84].

In another study, Darojah et al. compared the results achieved in the previously mentioned paper, in which the EKF was used, to an UKF implementation [72]. 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.

Zhan et al. suggest that while the EKF provides a straightforward implementation, several drawbacks can be combated with the use of the UKF [69]. 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.

3.2. H-infinity filters

H-infinity filters have received significant attention in recent years due to their ability to be employed across a wide range of domains, including signal reconstruction and neural network training. In distinction to Kalman filters, the H-infinity filter does not necessitate apriori knowledge of the noise statistics. H-infinity filtering will attempt to minimize the impact of the worst possible disturbances on the estimation error, therein achieving higher robustness against model uncertainty.

To this end, the H-infinity filter has been considered a viable alternative to training feedforward neural networks. Tang et al. present this concept in a paper published in 2007, where H-infinity filtering is utilized for the task of neural network training and pruning. The fundamental motivation behind employing H-infinity filtering was that it was more efficient and robust than the Kalman filter, which already successfully trained neural networks. Furthermore, unlike the Kalman filter, which is only optimal for normally distributed noise, the H-infinity filter circumvents making assumptions about the noise distribution. Consequently, employing H-infinity filtering to neural network training will augment robustness to deviations in network weight initialization and deterministic disturbance in observations. The results presented in the paper indicate that applying the H-infinity to feedforward neural network training and pruning achieved excellent generalization results and reduced network complexity by decreasing overall network weights.

3.3. Particle filters

The particle filter (PF), or sequential Monte Carlo strategy, is a method advanced by Pierre Del Moral for estimating the states of a dynamic system [85]. The algorithm put forth by Del Moral utilizes a set of particles, or samples, to generate a posterior distribution of a stochastic process given noisy observations. Each particle is therein designated a likelihood weight, which is a representation of the probability that the respective particle will be sampled from the generated probability density function [86,87]. The core concept is therein to recursively update the particles, thus achieving an approximation of the posterior distribution at the following time step [88,89]. The KF can be seen as an implementation of the particle filter assuming a normal distribution of particles and a time series mapping which preserves the normality of the distribution [90].

The PF is widely praised for its capability in handling nonlinearity. The PF has been shown to deliver exceptional performance when implemented in machine learning systems. Liangcheng et al. were among the first to propose the use of particle filters for parameter tuning [91–93]. The authors integrated the particle filter into the learning process of a support vector machine (SVM), achieving excellent results.

In more recent research, the implementation of PFs in neural network training has shown promising results. Nazari et al. used a particle filter to estimate and train neural network weights. Similar to KF algorithms, the authors used the weights to represent system states. They compared their results against standard backpropagation and a genetic algorithm utilizing an evapotranspiration estimation problem. The PF-based training method achieved the best overall performance and generalization ability, achieving the second lowest training RMSE score and lowest test RMSE score than any of the tested algorithms on the selected problem [73].

3.4. Smooth variable structure filters

The smooth variable structure filter (SVSF) is a state and parameter estimation algorithm introduced in 2007 by Habibi et al., which is built upon variable structure theory and sliding mode concepts [94]. The SVSF, similar to the KF, is formulated as a predictor-corrector method, and has been utilized in various studies to train feedforward neural networks, providing low mean squared error and exceptional generalization [95,96]. The fundamental concept of the SVSF is to compel produced state estimates to converge within a specific true state trajectory boundary based on sliding mode concepts.

To utilize the SVSF for network training, the network needs to be framed as an iterative estimation problem, similar to the KF implementation methodology discussed prior. The algorithm begins by randomly initializing the network weights. The Jacobian matrix is then computed by differentiating the networks transfer function with respect to the network weights. A state transformation is conducted on the weights using the Jacobian matrix, with the aim of linearly associating the output vector to the system states, or network weights. The output vector can therein be calculated as the product of the Jacobian matrix and the network weights. The error is subsequently estimated as a function of the difference between the target output and computed network output. Following the calculation of the error, the SVSF gain matrix is calculated as a function of the current and previous error measurement, specified convergence rate, and smoothing boundary layer widths. The weights are updated using the weight estimate, SVSF

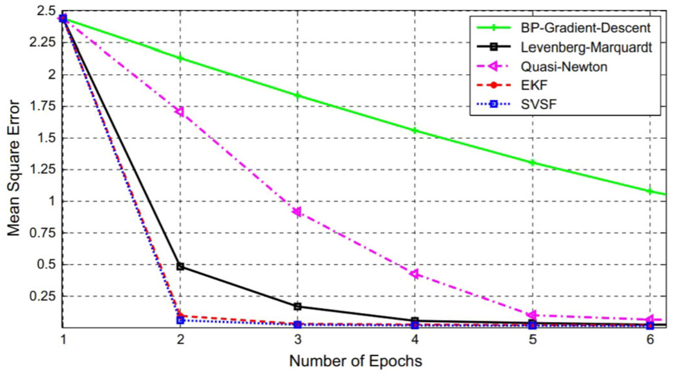

Fig. 8. SVSF, DEKF and Standard Backpropagation MSE for a Cancer Classification Problem [98].

gain matrix and current estimated measurement error. After the updating of the weights, the a posteriori output vector is estimated as the product of an identity matrix and the updated weights. The a posteriori error is achieved as the difference between the target output and a posteriori output vector. Finally, the true network weights are computed by multiplying the Jacobian by the recently updated weight matrix. This process is repeated for each training epoch, finishing only when a specified stopping criteria has been met. The process described above can be formalized as [97]:

$$\hat{w}_{k+1|k} = \hat{w}_k \tag{33}$$

$$\hat{x}_{k+1|k} = C_{k|linearized}\hat{w}_{k+1|k} \tag{34}$$

$$z_{k+1|k} = \mathbf{I}\hat{x}_{k+1|k} \tag{35}$$

$$E_{k+1|k} = z - \hat{z}_{k+1|k} \tag{36}$$

$$K_{k+1} = diag[|E_{k+1|k}| + \gamma|E_{k|k}| \circ sat(\frac{E_{k+1|k}}{\psi})]diag(E_{k+1|k})^{-1} \tag{37}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}E_{k+1|k} \tag{38}$$

$$\hat{w}_{k+1|k+1} = C_{k|linearized}\hat{x}_{k+1|k+1} \tag{39}$$

Ahmed et al. were among the first to publish the use of SVSF for neural network training. Using three variant classification tasks, the authors perform a comparative study on the performance of the SVSF, DEKF, and standard backpropagation. They report faster convergence using the SVSF, requiring less time to reach the same performance as the DEKF and standard backpropagation methodologies. Furthermore, the SVSF achieved minimum MSE and excellent generalization capability. Fig. 8 depicts the performance of the selected algorithms on a cancer classification problem, showcasing the superior performance of the SVSF, achieving the lowest MMSE faster than any of the other tested algorithms [98].

The SVSF does not require as much storage as the EKF, due to not having to compute the covariance matrix. However, the SVSF necessitates additional forward computation to estimate the a posteriori error. Ismail et al. propose that performing a set of selected modifications to the SVSF algorithm can drastically reduce the computational and storage requirements of the SVSF. The authors name the modified algorithm the reduced-SVSF (RSVSF). The RSVSF utilizes a mini-batch training strategy to capitalize on parallel computation in an effort to improve computational time. The error size is scaled down to a scalar value with a static measure like MSE, in turn significantly decreasing the size of the Jacobian matrix. A tuning parameter is employed in the RSVSF, which when set to zero eliminates the a posteriori error calculation [99].

In the RSVSF, a learning rate is introduced to regulate weight modifications, allowing for larger step sizes and faster learning. Lastly, a decay option is added to decrease the variable boundary

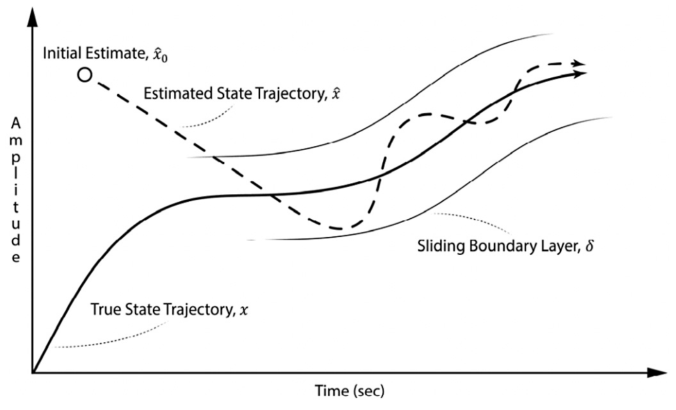
Initial Estimate, $\hat{x}_0$
Estimated State Trajectory, $\hat{x}$
Amplitude
Sliding Boundary Layer, δ
True State Trajectory, x
Time (sec)

Fig. 9. Sliding Innovation Filter Concept: Effects of the Switching Gain and Sliding Boundary Layer [100].

layers over numerous iterations to allow for smaller values which aid fine-tuning. These modifications substantially curtail computation and storage requirements for the SVSF. Ismail et al. perform a set of experiments on the RSVSF utilizing various selected datasets. They report excellent convergence characteristics and storage requirements.

3.5. Sliding innovation filters

Within the greater scope of estimation theory, a comprise between optimality and robustness has always existed, as briefly discussed prior. The Sliding Innovation Filter (SIF) (Fig. 9), an estimation strategy proposed by Gadsden et al., is sub-optimal insofar as it does not provide the optimal solution to a linear estimation problem. However, with the assistance of the switching gain, the SIF is much more robust to modeling disturbances and uncertainties.

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 $\tilde{z}_{k+1|k}$. However, the fundamental difference between the formulation of the Kalman Filter and the SIF is the calculation of gain K. The prediction stage of the SIF is formulated as:

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{40}$$

$$P_{k+1|k} = AP_{k|k}A^T + Q_k \tag{41}$$

$$\tilde{z}_{k+1|k} = z_{k+1} - C\hat{x}_{k+1|k} \tag{42}$$

Similar to the Kalman filter, after the prediction stage the update stage occurs:

$$K_{k+1} = C^+\overline{sat}(|\tilde{z}_{k+1|k}|/\delta) \tag{43}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}z_{k+1} \tag{44}$$

$$\begin{aligned} P_{k+1|k+1} =& (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T \\ &+ K_{k+1}R_{k+1}K_{k+1}^T \end{aligned} \tag{45}$$

where C^+ represents to the pseudoinverse of the measurement matrix, $\overline{sat}$ refers to the saturation value of the diagonal of that term which is bounded between -1 and +1, $\tilde{z}_{k+1|k}$ is the absolute value of the innovation and δ signifies the width of the sliding boundary layer.

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].

Alsadi et al. presented the concept of a feedforward neural network optimization methodology utilizing the SIF. The authors build on the success of the non-linear estimation algorithm, the UKF, stating that existing literature has vividly depicted the accuracy and efficiency of employing the UKF for the purpose of feedforward neural network training. In addition, due to not requiring linearization, the UKF has been viewed as suitable for systems with unknown linearities. The authors postulate that the advantages of the UKF can be further augmented with the employment of the SIF. To this end Alsadi et al. train a feedforward neural network with a single hidden layer to solve a regression based problem using the SIF and perform a comparative analysis with the UKF. They report excellent convergence characteristics for both the UKF and SIF, however the SIF was able to reach a similar mean squared error in substantially less time than the UKF, indicating the advantage in complexity and robustness of the SIF.

It is important to note that the SIF implementation discussed prior is based on a linear formulation. The non-linear formulation of the SIF, known as the Extended Sliding Innovation Filter (ESIF), can be introduced to further account for nonlinearities in the same manner the EKF was implemented.

The fundamental structure of the nonlinear SIF estimation process is akin the linear SIF procedure, with the formulation of the gain being the principal difference. Take for instance a nonlinear system function $f(\hat{x}_{k|k}, u_k)$ and a nonlinear measurement function $h(\hat{x}_{k+1|k})$. Utilizing partial derivatives, the linearized forms of the aforementioned nonlinearities can be achieved as such:

$$F_k = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k|k}, u_k} \tag{46}$$

$$H_{k+1} = \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k+1|k}} \tag{47}$$

Subsequently the prediction stage can be formulated as:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \tag{48}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \tag{49}$$

$$\tilde{z}_{k+1|k} = z_{k+1} - h(\hat{x}_{k+1|k}) \tag{50}$$

After the state prediction and error covariance matrix are produced during the respective Prediction stage, the update stage can be conducted with the following set of equations [35]:

$$K_{k+1} = H_{k+1}^+\overline{sat}(|\tilde{z}_{k+1|k}|/\delta) \tag{51}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}z_{k+1} \tag{52}$$

$$\begin{aligned} P_{k+1|k+1} =& (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T \\ &+ K_{k+1}R_{k+1}K_{k+1}^T \end{aligned} \tag{53}$$

where H_{k+1}^+ is the pseudoinverse of the linearized measurement matrix at time $k+1$.

Although no research has been conducted on the practical employment of the ESIF for neural network training, we suggest that future research should be dedicated towards experimenting with the training of simple feedforward neural networks utilizing the ESIF with a focus on analyzing the complexity. In addition, the generalization capabilities of the training procedure can be further augmented with an adaptive variation of the SIF, proposed by Lee et al., which entails a time varying sliding boundary layer:

$$\delta_{k+1} = S_{k+1}(S_{k+1} - R_{k+1})^{-1}|\tilde{z}_{k+1|k}| \tag{54}$$

The width of the time varying sliding boundary layer δ_{k+1} is therefore a function of the innovation covariance matrix S_{k+1}, the measurement noise covariance matrix R_{k+1} and the absolute magnitude of the innovation $\tilde{z}_{k+1|k}$.

Lee et al. present empirical evidence for the effectiveness of the adaptive SIF in maintaining accurate estimates in the face of system faults, therein translating to increased generalization and robustness in the presence of dataset outliers. The SIF can present massive advantages for neural network training and therefore future research should be dedicated to advancing the current state of the SIF's involvement in the field of intelligent estimation.

4. Estimation-based recurrent neural network training

In addition to the integration of estimation based training techniques for feedforward neural networks, research has supported the utilization of these methods for recurrent neural networks (RNNs). Gradient descent has been widely overlooked as an efficient training algorithm for RNNs due to pragmatic difficulties such as being slow and ineffective in converging to an optimal solution. Various researchers have therefore explored the implementation of non-gradient training methods as a potential alternative for gradient-based approaches in RNN training. Simulation results for problems in function approximation and pattern classification indicate that training algorithms based on non-gradient methods require significantly fewer presentations of training data than gradient-based algorithms. Second-order training algorithms, such as the EKF, unlike higher-order training algorithms, do not necessitate batch processing. This makes them a powerful alternative to gradient-based methods for specific RNNs applications.

Puskorius et al. utilize a DEKF approach to tackle the problem of recurrent neural network training, applying their novel implementation to variant control problems. Using three different control problems, the authors compare the performance of the DEKF against a basic gradient training approach. They report back excellent performance on behalf of the DEKF, which consistently outperformed the gradient-based approach. Indicating that estimation based training techniques have a promising future within the recurrent neural network training procedure [101]. Sum et al. state that the recurrent neural network training problem, like the feedforward neural network training procedure, can be formulated as an EKF problem [102]. The introduction of the EKF is postulated by the authors to reduce computational complexity when training and pruning a recurrent neural network.

However, Wang et al. claim that various studies which aim at successfully implementing estimation based training methods with recurrent neural network training make numerous assumptions and need to be further developed theoretically and tested practically [103]. In their paper, Wang et al. attempt to develop an effective EKF-based RNN training approach with a controllable training convergence. The proposed algorithm is tested on a nonlinear dynamical benchmark system. The algorithm was successful in ensuring training convergence and improving on convergence speed. Other authors have also discussed the inherent disadvantages of using gradient-based methods when training recurrent neural networks. They state that gradient-based approaches entail slow convergence speed and extended training sequences for achieving satisfactory performance. As noted previously, the UKF is capable of handling non-linearities to a higher degree than the EKF. The result of UKF implementation for non-linear applications is typically an augmented standard of estimates, without the necessity of numerical derivatives, therein popularizing the utilization of the UKF for non-linear systems [104].

While Wang et al. focused soley on the EKF, they add that the work can be further extended for an UKF-based training approach for better performance. Choi et al. compared the performance of an EKF and UKF algorithm for RNNs training on various time-varying channels [105]. The experimental results displayed that the UKF training procedure yielded uniformly superior performance, although also increasing computational cost. In another paper, Choi et al. conducted further testing of both the EKF and UKF algorithms for RNN training utilizing a time-series prediction problem [106]. The results support the initial findings with the UKF achieving excellent performance; however, producing an increase in computational complexity. Although there is currently no research conducted on the use of the SVSF in RNN training, theoretically the SVSF could perform better than the UKF and reduce computational cost.

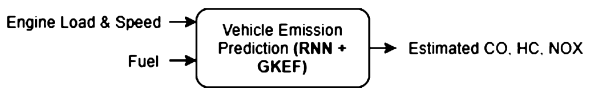

Fig. 10. Vehicle Emission Prediction Diagram.

5. Intelligent estimation in the professional domain

As discussed in the previous section, the integration of intelligent estimation has the potential to enhance the overall neural network training process. While estimation methods, as a whole, have been applied to diverse fields such as telecommunications and air traffic control, there are many professional domains which can be aided with the deployment of intelligent estimation methods. The automotive domain has been extensively explored by researchers in aim of analyzing the effects and advantages of utilizing intelligent estimation techniques.

Vehicle emission (Fig. 10) is a large issue with the state of contemporary cars and has been such for many decades. Emissions released from motor vehicles have had a devastating effect on the environment and it inhabitants. Vehicle emission control aims at reducing motor vehicle pollution with emphasis on central components, such as the internal combustion engine. Models, of which were mandated by the government, were developed in aim of estimating vehicle emission based on particular use patterns. However, actual vehicle emission is dependent on real-world driving patterns. Neural networks have been utilized in the effort to accurately estimate these vehicle emissions. Specifically, using a small quantity of engine variables, recurrent neural networks have been utilized to estimate instantaneous engine-out emission. Researchers found that recurrent neural networks trained with an intelligent estimation technique, namely GEKF, provided accurate and reliable results in estimating instantaneous emissions levels [107].

Intelligent estimation techniques have also been applied to various other issues in the automotive domain. The catalyst converter is a fundamental component of motor vehicles, which aims at converting unwanted and potentially environmentally damaging emissions into less-toxic pollutants. Control of the air to fuel ratio is central to the conversion process, with the goal of regulating the ratio about stoichiometry. A binary sensor is utilized to measure the engine-out exhaust, indicating whether it is fuel or air dominant. The catalyst converter is prone to temporal degradation, necessitating constant performance monitoring (Fig. 11). Researchers deployed neural networks trained with decoupled EKF's in aim of analyzing conversion efficiency, utilizing engine operating conditions, pre and post catalyst sensors. The results indicate exceptional model performance while using intelligent estimation techniques [108].

Engine misfire detection is the process of detecting when a fragment of the air-fuel mixture fails to ignite. Engine misfires are common, however, consistently recurring engine misfires can significantly lead to the deterioration of the catalyst converter.

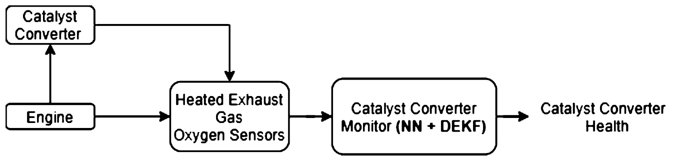
Catalyst Converter
Engine
Heated Exhaust Gas Oxygen Sensors
Catalyst Converter Monitor (NN + DEKF)
Catalyst Converter Health

Fig. 11. Catalyst Converter Monitor Diagram.

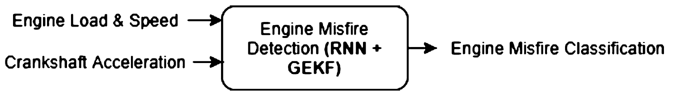
Engine Load & Speed
Crankshaft Acceleration
Engine Misfire Detection (RNN + GEKF)
Engine Misfire Classification

Fig. 12. Engine Misfire Detection Diagram.

Therefore, the immediate detection of engine misfires is crucial to maintaining automotive integrity. Researchers approached this problem by utilizing basic engine variables and crankshaft acceleration to train a recurrent neural network using the GEKF (Fig. 12). They report excellent performance with few classification errors [108].

Although the practical implementations provided above exclusively revolve around the automotive domain, it is evident that intelligent estimation contains the potential for reliable pragmatic application in numerous other fields. The question must be posed, why aren't intelligent estimation methods utilized in greater frequency across various domains? This has to do with the level of research being conducted in the field of intelligent estimation, particularly concerning improving computational complexity. The largest drawback to deploying such techniques is the increased computational cost associated with these methods. Research has already endeavoured to improve computational cost with the introduction of decoupling and other approaches. However, much more attention needs be driven toward algorithmic complexity prior to witnessing increased implementations. In addition, there is very little conversation on the impact of time delay and its consequence on stability, see [109], as a result of the increased computational complexity. With the introduction of more research aimed at improving the computational complexity inherent in these techniques, the more we can expect to see intelligent estimation methods utilized across professional domains.

6. Concluding remarks

Deep learning is a fundamental topic in the evolution of machine learning. The value of algorithmic performance has influenced the development of novel model training methods, intending to augment model accuracy and reduce computational cost. For decades, the primary approach to artificial neural network training was a first-order-based gradient computation. However, due to inherent issues, such as local convergence, non-gradient-based training techniques have been investigated as an alternative to traditional methods.

Estimation-based training approaches are a form of non-gradient-based training techniques. Initially appearing in the late 1980s, estimation-based training methods were opted for to circumvent the convergence limitations of standard backpropagation. The extended Kalman filter (EKF) was an initial implementation which achieved excellent convergence speed; however, displayed poor scalability. Subsequently, numerous researchers endeavoured to modify the EKF implementation to reduce model complexity. Utilizing local approaches to training individual neurons in the network and decoupling to group network weights, researchers achieved excellent algorithmic computational characteristics. In addition to the EKF, various other filtering strategies were utilized, including: the unscented Kalman filter (UKF), the particle filter (PF), the smooth variable structure filter (SVSF), and the sliding innovation filter (SIF).

Estimation-based training techniques have demonstrated promising results when combined with machine learning. As artificial intelligence and machine learning continues to evolve, and its useful applications grow, estimation-based training methods present themselves as an alternative approach to traditional gradient-based methods. Furthermore, coupling estimation-based techniques with new research areas, such as physics-informed machine learning, will yield new insights and further improvements.

A series of crucial questions naturally arise concerning the implementation of intelligent estimation methods. A primary question being, what types of problems can intelligent estimation excel in? The literature displays that intelligent estimation achieves optimal performance when dealing with classification-based problems, particularly when decoupling can be utilized to respectively scale applications. However, there is still a need for more research to be conducted on the application of intelligent estimation methods to variant problem types, as well as its adoption in industry. Exploring regression-based problems can further assist with diversifying current knowledge of intelligent estimation performance. Future work needs to be conducted on the pragmatic implementation of these methods in specific fields of application, such as condition monitoring or healthcare. Furthermore, advancements in the field of estimation theory will provide more accurate filtering algorithms which can be implemented within the training procedure with the goal of improving training execution.

Nomenclature

$^{-1}, ^{+}, ^{T}$	Notation denoting an inverse, a pseudo inverse and a transpose, respectively
A, B	The system and input gain matrix, respectively
C	The measurement matrix
K	The filter gain matrix
P	The state error covariance matrix
Q	The covariance of the process (system) noise
R	The covariance of the observation (measurement) noise
u_k, z_k	The system input and measurement vectors at time k
w_k, v_k	The system and measurement noise vectors at time k, respectively
δ	The width of the sliding boundary layer
$\overline{sat}$	The diagonal of the saturation value

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Y. Bar-Shalom, X.-R. Li, T. Kirubarajan, Estimation with Applications to Tracking and Navigation, John Wiley & Sons, Inc., New York, USA, 2001.
[2] P.S. Maybeck, Stochastic Models, Estimation, and Control, Academic Press, 1982.
[3] Y. Ma, Q. Yu, I. Cohen, Target tracking with incomplete detection, Comput. Vis. Image Underst. 113 (apr 2009) 580–587.
[4] T. Quach, M. Farooq, Maximum likelihood track formation with the viterbi algorithm, in: Proceedings of 1994 33rd IEEE Conference on Decision and Control, vol. 1, 1994, pp. 271–276.
[5] M. Shehab El Din, A.A. Hussein, M.F. Abdel-Hafez, Improved battery SOC estimation accuracy using a modified UKF with an adaptive cell model under real EV operating conditions, IEEE Trans. Transport. Electrific. 4 (jun 2018) 408–417.

[6] H. Sorenson, Parameter Estimation: Principles and Problems, Marcel Dekker, New York, 1980.
[7] J.H. Donnelly, R.E. Shannon, Minimum mean-squared-error estimators for simulation experiments, Commun. ACM 24 (apr 1981) 253–259.
[8] S.A. van de Geer, Least squares estimation, in: Encyclopedia of Statistics in Behavioral Science, John Wiley & Sons, Ltd, Chichester, UK, oct 2005.
[9] Y. Iiguni, H. Sakai, H. Tokumaru, A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter, IEEE Trans. Signal Process. 40 (4) (1992) 959–966.
[10] Q.V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, A.Y. Ng, On Optimization Methods for Deep Learning, tech. rep., 2011.
[11] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, MIT Press, Cambridge, 2016.
[12] H. Larochelle, Y. Bengio, J. Louradour, L.U. Ca, Exploring Strategies for Training Deep Neural Networks Pascal Lamblin, tech. rep., 2009.
[13] B. Mehlig, Artificial neural networks, in: Studies in Systems, Decision and Control, vol. 131, jan 2019, pp. 11–35.
[14] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, T. Goldstein, Training Neural Networks Without Gradients: a Scalable ADMM Approach, tech. rep., 2016.
[15] J.L. Davidson, X. Hua, D. Ashlock, Comparison of genetic algorithm, regression, and Newton's method for parameter estimation of texture models, in: Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, IEEE, 1996, pp. 201–206.
[16] R. Mahajan, G. Kaur, Neural Networks using Genetic Algorithms, Tech. Rep. 14, 2013.
[17] R. Battiti, First- and second-order methods for learning: between steepest descent and Newton's method, Neural Comput. 4 (mar 1992) 141–166.
[18] B.C. Cetin, J.W. Burdick, 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 - Conference Proceedings, vol. 1993-January, Institute of Electrical and Electronics Engineers Inc., 1993, pp. 836–842.
[19] A. Atakulreka, D. Sutivong, Avoiding Local Minima in Feedforward Neural Networks by Simultaneous Learning, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNAI, vol. 4830, Springer Verlag, 2007, pp. 100–109.
[20] M. Bianchini, M. Gori, M. Maggini, On the problem of local minima in recurrent neural networks, IEEE Trans. Neural Netw. 5 (2) (1994) 167–177.
[21] H. Kwak, B.-T. Zhang, Understanding Local Minima in Neural Net-works by Loss Surface Decomposition, tech. rep., feb 2018.
[22] J.L. Brierton, Techniques for avoiding local minima in gradient-descent-based ID algorithms, in: R. Trebits, J.L. Kurtz (Eds.), Radar Sensor Technology II, vol. 3066, SPIE, jun 1997, pp. 130–135.
[23] M. Gori, A. Tesi, On the problem of local minima in backpropagation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1) (1992) 76–86.
[24] J.T.H. Lo, Y. Gui, Y. Peng, Solving the Local-Minimum Problem in Training Deep Learning Machines, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNCS, vol. 10634, Springer Verlag, nov 2017, pp. 166–174.
[25] F. Ahmad, N.A.M. Isa, M.K. Osman, Z. Hussain, Performance comparison of gradient descent and genetic algorithm based artificial neural networks training, in: Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, ISDA'10, 2010, pp. 604–609.
[26] V. Ranganathan, S. Natarajan, A New Backpropagation Algorithm without Gradient Descent, arXiv, jan 2018,.
[27] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, W. Maass, Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets, arXiv, jan 2019.
[28] K. Watanabe, S.G. Tzafestas, Learning algorithms for neural networks with the Kalman filters, J. Intell. Robot. Syst. 3 (dec 1990) 305–319.
[29] S. Haykin, Kalman filters, in: Kalman Filtering and Neural Networks, John Wiley & Sons, Inc., New York, USA, 2001, pp. 1–21.
[30] X. Yuan, Y. Wang, C. Yang, Z. Song, W. Gui, Weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes operation optimization of the oil-refining processes view project soft sensor modeling for industrial processes view project weighted linear dynamic system for feature representation and soft sensor application in nonlinear dynamic industrial processes, IEEE Trans. Ind. Electron. 65 () (2018).
[31] E.A. Wan, R. Van Der Merwe, The Unscented Kalman Filter for Nonlinear Estimation Perceptual Wavelet Adaptive Denoising of Speech View Project the Unscented Kalman Filter for Nonlinear Estimation, 2000.
[32] R. Endo, S. Shiota, N. Tosaka, Structural damage identification of frame model based on filtering algorithm, in: Inverse Problems in Engineering Mechanics III, Elsevier, jan 2002, pp. 153–160.
[33] G. Welch, G. Bishop, et al., An Introduction to the Kalman Filter, 1995.
[34] R.J. Meinhold, N.D. Singpurwalla, Understanding the kalman filter, Am. Stat. 37 (2) (1983) 123–127.
[35] Y. Kim, H. Bang, Introduction to Kalman filter and its applications, in: Introduction and Implementations of the Kalman Filter, IntechOpen, may 2019.
[36] V. Stojanovic, N. Nedic, Joint state and parameter robust estimation of stochastic nonlinear systems, Int. J. Robust Nonlinear Control 26 (14) (2016) 3058–3074.
[37] Y. Hao, Z. Xiong, F. Sun, X. Wang, Comparison of unscented Kalman filters, in: Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, ICMA 2007, 2007, pp. 895–899.
[38] S.J. Julier, J.K. Uhlmann, A New Extension of the Kalman Filter to Nonlinear Systems, tech. rep.
[39] P. Chen, H. Ma, S. Gao, Y. Huang, Modified extended Kalman filtering for tracking with insufficient and intermittent observations, Math. Probl. Eng. 2015 (2015).
[40] N.U. Ahmed, S.M. Radaideh, Modified extended Kalman filtering, IEEE Trans. Autom. Control 39 (6) (1994) 1322–1326.
[41] W. Li, Y. Ji, H-infinity filtering for a class of nonlinear discrete-time systems based on unscented transform, Signal Process. 90 (2010) 3301–3307.
[42] B. Ristic, S. Arulampalam, N. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House Publishers, 2004.
[43] S.A. Gadsden, Smooth variable structure filtering: Theory and applications, PhD Thesis, McMaster University, 2011.
[44] S.A. Gadsden, M. Al-Shabi, The sliding innovation filter, IEEE Access (2020).
[45] J. Hartikainen, S. Särkkä, Kalman filtering and smoothing solutions to temporal Gaussian process regression models, in: 2010 IEEE International Workshop on Machine Learning for Signal Processing, Aug. 2010, pp. 379–384, ISSN: 2378-928X.
[46] L. Martino, J. Read, Joint introduction to gaussian processes and relevance vector machines with connections to kalman filtering and other kernel smoothers, CoRR, arXiv:2009.09217 [abs], 2020.
[47] R. Frigola, Y. Chen, C.E. Rasmussen, Variational gaussian process state-space models, CoRR, arXiv:1406.4905 [abs], 2014.
[48] S. Eleftheriadis, T.F.W. Nicholson, M.P. Deisenroth, J. Hensman, Identification of Gaussian Process State Space Models, 2017.
[49] A. Xie, F. Yin, B. Ai, S. Cui, Learning while Navigating: a Practical System Based on Variational Gaussian Process State-Space Model and Smartphone Sensory Data, 2019.
[50] G. Bebis, M. Georgiopoulos, Why network size is so important, IEEE Potentials 13 (4) (1994) 27–31.
[51] S. Ruder, An Overview of Gradient Descent Optimization Algorithms, sep 2016.
[52] N. Ketkar, Deep Learning with Python, Apress, 2017.
[53] C.E. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning.
[54] S. Narayan, The generalized sigmoid activation function: competitive supervised learning, Inf. Sci. 99 (6 1997) 69–82.
[55] A.F. Agarap, Deep learning using rectified linear units (relu), arXiv, 3 2018.
[56] R.D. Reed, R.J. Marks, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, MIT Press, 1999.
[57] W. Jin, Z.J. Li, L.S. Wei, H. Zhen, The Improvements of bp Neural Network Learning Algorithm, Institute of Electrical and Electronics Engineers (IEEE), 11 2002, pp. 1647–1649.
[58] A. Sherstinsky, Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network, Phys. D: Nonlinear Phenom. 404 (2018) 8.
[59] L. Medsker, D.L. Jain, B.R.L.N.Y. Washington, Recurrent Neural Networks Edited by Design and Applications, 2001.
[60] M. Bodén, A Guide to Recurrent Neural Networks and Backpropagation, 2001.
[61] G. Dorrner, Neural networks for time series processing.
[62] J.S. Zhang, X.C. Xiao, Predicting chaotic time series using recurrent neural network, Chin. Phys. Lett. 17 (2000) 88–90.
[63] P. Malhotra, L. Vig, P. Agarwal, G. Shroff, Timenet: Pre-trained deep recurrent neural network for time series classification.
[64] J. Connor, R. Martin L. A. I, transactions on neural, and undefined 1994, Recurrent neural networks and robust time series prediction,' ieeexplore.ieee.org.
[65] S. Singhal, L. Wu, Training multilayer perceptrons with the extended kalman algorithm, in: D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems 1, Morgan-Kaufmann, 1989, pp. 133–140.
[66] G.V. Puskorius, L.A. Feldkamp, Decoupled extended Kalman filter training of feedforward layered networks, in: Proceedings. IJCNN-91-Seattle: International Joint Conference on Neural Networks, IEEE, 1991, pp. 771–777.
[67] S. Shah, F. Palmieri, M. Datum, Optimal filtering algorithms for fast learning in feedforward neural networks, Neural Netw. 5 (sep 1992) 779–787.
[68] N. Mert Vural, Suleyman S. Kozat, An Efficient EKF Based Training Algorithm for LSTM-Based Online Learning.
[69] R. Zhan, J. Wan, Neural network-aided adaptive unscented Kalman filter for nonlinear state estimation, IEEE Signal Process. Lett. 13 (jul 2006) 445–448.
[70] H.-S. Tang, S. Xue, T. Sato, H∞ Filtering in Neural Network Training and Pruning with Application to System Identification, J. Comput. Civ. Eng. 21 (Jan. 2007) 47–58, American Society of Civil Engineers.
[71] R. Ahmed, M. El Sayed, S.A. Gadsden, J. Tjong, S. Habibi, Artificial neural network training utilizing the smooth variable structure filter estimation strategy, Neural Comput. Appl. 27 (apr 2016) 537–548.

[72] Z. Darojah, E.S. Ningrum, D.S. Purnomo, The training of feedforward neural network using the unscented Kalman filter for voice classification application, in: Proceedings - International Electronics Symposium on Knowledge Creation and Intelligent Computing, IES-KCIC 2017, vol. 2017-January, Institute of Electrical and Electronics Engineers Inc., dec 2017, pp. 21–25.
[73] M. Nazari, S. Shamshirband, The particle filter-based back propagation neural network for evapotranspiration estimation, ISH J. Hydr. Eng. 26 (jul 2020) 267–272.
[74] N. Alsadi, W. Hilal, O. Surucu, A. Giuliano, S.A. Gadsden, J. Yawney, M. Al-Shabi, Neural network training loss optimization utilizing the sliding innovation filter, in: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV, vol. 12113, May 2022.
[75] G.V. Puskorius, L.A. Feldkamp, Parameter-based Kalman filter training: theory and implementation, in: Kalman Filtering and Neural Networks, John Wiley & Sons, Inc., New York, USA, apr 2003, pp. 23–67.
[76] K. Kaneshige, X. Wang, M. Saewong, V. Syrmos, Development of node-decoupled extended Kalman filter (NDEKF) training method to design neural network diagnostic/prognostic reasoners, in: T. Kundu (Ed.), Health Monitoring and Smart Nondestructive Evaluation of Structural and Biological Systems III, in: SPIE, vol. 5394, International Society for Optics and Photonics, 2004, pp. 375–384.
[77] G.V. Puskorius, L.A. Feldkamp, Extensions and enhancements of decoupled extended kalman filter training, in: Proceedings of International Conference on Neural Networks (ICNN'97), vol. 3, 1997, pp. 1879–1883.
[78] M.C. Nechyba, Yangsheng Xu, Cascade neural networks with node-decoupled extended kalman filtering, in: Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. Towards New Computational Principles for Robotics and Automation, 1997, pp. 214–219.
[79] S. Murtuza, S.F. Chorian, Node decoupled extended kalman filter based learning algorithm for neural networks, in: Proceedings of 1994 9th IEEE International Symposium on Intelligent Control, 1994, pp. 364–369.
[80] A. Gadsden, S. Habibi, D. Dunne, T. Kirubarajan, Nonlinear estimation techniques applied on target tracking problems, J. Dyn. Syst. Meas. Control 134 (sep 2012).
[81] J.T. Ambadan, Y. Tang, Sigma-point kalman filter data assimilation methods for strongly nonlinear systems, J. Atmos. Sci. 66 (2) (01 Feb. 2009) 261–285.
[82] Nonlinear Filtering, ch. 7, John Wiley & Sons, Ltd, 2008, pp. 293–353.
[83] J.J. LaViola, A comparison of unscented and extended Kalman filtering for estimating quaternion motion, Proceedings of the American Control Conference 3 (2003) 2435–2440.
[84] R. Van Der Merwe, E.A. Wan, Efficient Derivative-Free Kalman Filters for Online Learning, tech. rep..
[85] P. Del Moral, Filtrage non-linéaire par systèmes de particules en interaction, C. R. Acad. Sci., Ser. 1 Math. 325 (sep 1997) 653–658.
[86] S. Godsill, Particle Filtering: The First 25 Years and Beyond, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., 5 2019, pp. 7760–7764.
[87] A.H. Sayed, P.M. Djurić, F. Hlawatsch, Distributed Kalman and Particle Filtering, 6 2018.
[88] H.R. Künsch, Particle filters, Bernoulli 19 (2013) 1391–1403.
[89] B. Liu, S. Cheng, Y. Shi, Particle Filter Optimization: A Brief Introduction, 2016.
[90] A. Pervan, T. Murphey, Algorithmic Materials: Embedding Computation Within Material Properties for Autonomy, 1 2019.
[91] C. Liangcheng, Y. Huizhong, A learning method of support vector machine based on particle filters, in: 2009 Fifth International Conference on Natural Computation, vol. 1, 2009, pp. 431–435.
[92] P. Insom, C. Cao, P. Boonsrimuang, D. Liu, et al., A Support Vector Machine-Based Particle Filter Method for Improved Flooding Classification, ieeexplore.ieee.org, 2015.
[93] Z. Wei, T. Tao, D. ZhuoShu, E. Zio, A Dynamic Particle Filter-Support Vector Regression Method for Reliability Prediction, Elsevier, 2013.
[94] S. Habibi, The smooth variable structure filter, Proc. IEEE 95 (5) (2007) 1026–1059.
[95] M. Al-Shabi, S. Habibi, Iterative smooth variable structure filter for parameter estimation, ISRN Signal Process. 2011 (2011) 725108.
[96] M. Avzayesh, M. Abdel-Hafez, M. AlShabi, S. Gadsden, The smooth variable structure filter: a comprehensive review, Digit. Signal Process. 110 (dec 2020) 102912.
[97] R. Ahmed, M. El Sayed, S. Andrew Gadsden, S. Habibi, Fault Detection of an Engine Using a Neural Network Trained by the Smooth Variable Structure Filter, 2011.
[98] R. Ahmed, M. El Sayed, S. Andrew Gadsden, J. Tjong, S. Habibi, Artificial neural network training utilizing the smooth variable structure filter estimation strategy.
[99] M. Ismail, M. Attari, S. Habibi, S. Ziada, Estimation theory and neural networks revisited: Rekf and rsvsf as optimization techniques for deep-learning, Neural Netw. 108 (2018) 509–526.
[100] A.S. Lee, S.A. Gadsden, M. Al-Shabi, An adaptive formulation of the sliding innovation filter, IEEE Signal Process. Lett. 28 (2021) 1295–1299, Conference Name: IEEE Signal Processing Letters.
[101] G.V. Puskorius, L.A. Feldkamp, Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks, IEEE Trans. Neural Netw. 5 (2) (1994) 279–297.
[102] J. Sum, J.P.F. Sum, C.S. Leung, L.W. Chan, Extended Kalman filter in recurrent neural network training and pruning Extended Kalman lter in recurrent neural network training and pruning, tech. rep., 1996.
[103] X. Wang, Y. Huang, Convergence study in extended Kalman filter-based training of recurrent neural networks, IEEE Trans. Neural Netw. 22 (apr 2011) 588–600.
[104] A. Sitz, U. Schwarz, J. Kurths, H.U. Voss, Estimation of parameters and unobserved components for nonlinear systems from noisy time series, Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Topics 66 (jul 2002) 016210.
[105] J. Choi, T.H. Yeap, M. Bouchard, Online State-Space Modeling Using Recurrent Multilayer Perceptrons with Unscented Kalman Filter, aug 2005.
[106] J. Choi, A.C.C. Lima, S. Haykin, Kalman filter-trained recurrent neural equalizers for time-varying channels, IEEE Trans. Commun. 53 (mar 2005) 472–480.
[107] G. Jesion, C. Gierczak, G. Puskorius, L. Feldkamp, J. Butler, The Application of Dynamic Neural Networks to the Estimation of Feedgas Vehicle Emissions, 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227), vol. 1, IEEE, 1998, pp. 69–73.
[108] L. Feldkamp, G. Puskorius, A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification, Proc. IEEE 86 (11) (1998) 2259–2277.
[109] Z. Xu, X. Li, V. Stojanovic, Exponential stability of nonlinear state-dependent delayed impulsive systems with applications, Nonlinear Anal. Hybrid Syst. 42 (2021) 101088.

Naseem Alsadi received his bachelor's degree in Computer Engineering from the University of Guelph (Ontario, Canada) with a dual focus on advanced mechatronics and artificial intelligence. He is currently pursuing a PhD in Mechanical Engineering with applications in Artificial Intelligence at McMaster University with funding from NSERC and an Ontario Graduate Scholarship. Naseem's research interests include machine learning techniques, blockchain strategies, and cognitive dynamic systems. He is currently a graduate research assistant in the Intelligence and Cognitive Engineering (ICE) Laboratory in the Department of Mechanical Engineering at McMaster University. Naseem is also involved in various research projects with industrial partners such as Adastra Corp. (Toronto, Ontario) where he delivers data solutions for industrial applications.

S. Andrew Gadsden earned his bachelor's in Mechanical Engineering and Management (business) and his PhD in Mechanical Engineering in the area of estimation theory and mechatronics from McMaster University (Ontario, Canada). Andrew worked concurrently as a postdoctoral fellow at McMaster University and as a Project Manager in the pharmaceutical industry for three years. He is currently an Associate Professor at McMaster University in the Department of Mechanical Engineering, and was previously an Associate Professor at the University of Guelph and an Assistant Professor at the University of Maryland. Andrew is a distinguished researcher and teacher, having earned NASA's 2021 Robert H. Goddard Award in Science (air-LUSI team), the 2019-2020 University Research Excellence Award for the College of Engineering and Physical Sciences (University of Guelph), the 2019 SPIE Rising Researcher Award (Baltimore, Maryland), the 2019 Distinguished Professor Award for Excellence in Teaching (University of Guelph Faculty Association), and a 2018 Ontario Early Researcher Award.

John Yawney earned his PhD in Applied Mathematics from the University of Waterloo (Ontario, Canada) and has extensive experience with building production AI/ML systems and performing numerical and statistical analyses. He is currently the Chief Analytics Officer at Adastra Corp. John has executive oversight of the data science practice, leads the development of the AI solutions portfolio, and holds an Adjunct Professor position in the Department of Mechanical Engineering at McMaster University. The advancement of the analytics space at Adastra Canada and global offices rely on John's expertise and his development of practical but theoretically sound solutions and AI service frameworks, for which he leverages his joint academic and industry focus. His research portfolio/focus includes numerical simulations of nonlinear systems, epidemiological modeling, applied AI applications, and smart or cognitive systems.