Estimation and Fusion for Tracking Over Long-Haul Links Using Artificial Neural Networks

Qiang Liu, *Member, IEEE,* Katharine Brigham, *Member, IEEE,* and Nageswara S. V. Rao, *Fellow, IEEE*

***Abstract*—In a long-haul sensor network, sensors are remotely deployed over a large geographical area to perform certain tasks, such as tracking and/or monitoring of one or more dynamic targets. A remote fusion center fuses the information provided by these sensors so that a final estimate of certain target characteristics – such as the position – is expected to possess much improved quality. In this work, we pursue learning-based approaches for estimation and fusion of target states in long-haul sensor networks. In particular, we consider learning based on various implementations of artificial neural networks (ANNs). The joint effect of (i) imperfect communication condition, namely, link-level loss and delay, and (ii) computation constraints, in the form of low-quality sensor estimates, on ANN-based estimation and fusion, is investigated by means of analytical and simulation studies.**

***Index Terms*—Long-haul sensor networks, state estimate fusion, artificial neural networks, estimation bias, error regularization, root-mean-square-error (RMSE) performance, reporting deadline.**

I. Introduction

Sensor networks have been deployed in many real-world applications, including military, security, healthcare, and environmental monitoring, among others [2]. We are primarily interested in one class of such networks, namely, the long-haul sensor networks, where sensors with sensing, data processing, and communication capabilities are deployed to cover a very large geographical area, such as a continent or even the entire globe. A remote sensor measures certain parameters of interest from the dynamic target(s) on its own, and then sends the state estimates it derives from these measurements to the fusion center. The fusion center collects data from multiple such sensors and fuses the data to obtain global estimates periodically at specified time instants. A global estimate is expected to be more accurate than those provided by the individual sensors, and this benefit is often referred to as the fusion gain.

When sensor data are communicated over long-haul links, for instance, the satellite links, due to the long distances (tens of thousands of miles), the signal propagation time can be significant, with round-trip times (RTTs) of well over half a second for geostationary earth orbits (GEOs) [27]. Communication over the satellite links is also characterized by sporadic high bit-error rates (BERs) and burst losses. The losses incurred during transmission or resulting from the message drop due to occasional high BERs could further reduce the number of reliable estimates available at the fusion center. Consequently, the global estimates may not be promptly and accurately finalized by the fusion center, leading to degraded fusion performance and even failures to comply with the system requirements on the worst-case estimation error and/or maximum reporting delay, both crucial elements for near real-time performance in many applications. Besides these communication constraints, some sensors are also prone to degraded performance, notably in the form of estimation bias, as a result of poor calibration or environmental factors. This can be considered as a manifestation of computation constraints imposed on the fusion center.

Some works have attempted to address estimation and/or fusion under variable communication loss and/or delay conditions. In [6], [12], [28], [29], estimation and fusion performance using Kalman filters (KFs) under variable packet loss rates have been considered. Studies including [24], [35], [36] have addressed the so-called out-of-sequence-measurement (OOSM) issue – where an OOSM is defined as a measurement that has been generated earlier but arrives later – and their common goal is to update the current state estimate with an earlier measurement without reordering the measurements and recalculating the state estimator recursively. More recently, [22] and [26] have exploited retransmission to recover some of the lost messages over time so that the effect of information loss can be somewhat mitigated. A dynamic online selective fusion mechanism based on the projected information gain is proposed in [21] so that the final time for fusion is dynamically determined depending on if enough information has arrived at the fusion center. A staggered estimation scheduling scheme is proposed in [19] that aims to explore the temporal relationships of adjacent data within an estimation interval to improve the estimation and fusion performance.

A number of data fusion methods have been developed over the years, with a primary goal of taking the data from multiple sensors and combining them to produce a condensed set of meaningful information with the highest possible degree of

Manuscript received ... ; revised ... ; accepted ... .

This work was supported by the Mathematics of Complex, Distributed, Interconnected Systems Program, Office of Advanced Computing Research, U.S. Department of Energy, and the SensorNet Project within the Office of Naval Research, through Oak Ridge National Laboratory managed by UT-Battelle, LLC for U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Q. Liu and N. S. V. Rao are with the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37831. Email: {liuq1,raons}@ornl.gov.

K. Brigham is with the School of Engineering and Computing Sciences, Durham University, Durham DH1 3LE, UK. E-mail: katharine.brigham@durham.ac.uk.

Preliminary versions of this paper were included in *Proc. 16th Int. Conf. Inf. Fusion (FUSION)*, Istanbul, Turkey, Jul. 2013. and *Proc. 18th Int. Conf. Inf. Fusion (FUSION)*, Washington, DC, Jul. 2015.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier

accuracy and certainty [3], [30]. Whereas most conventional state fusion approaches produce fused estimates by linearly combining the available sensor data, the use of nonlinear fusers is still fairly unexplored. Essentially, information fusion is a regression problem, and many existing regression analysis techniques can be considered. Since in many applications, field tests may be performed a priori using the available sensor networks to collect test measurements, we are interested in the use of learning-based fusers that are able to learn how to fuse the data based on these measurements in a nonlinear fashion.

Artificial neural networks (ANNs) have been applied to tasks such as pattern classification, clustering/categorization, function approximation, and prediction/forecasting, among others [14], [16]. In the literature, [7] and [10] proposed using ANNs to determine the weights for linearly combining sensor state estimates. More recently, we proposed learning-based nonlinear fusion [4], [20]; the main contribution of this paper is to further investigate the ANN-based fusers, accounting for both communication and computation constraints in long-haul sensor networks and their effects on fusion performance. In particular, besides performing the core function of learning from true target trajectories and sensor data and then applying such learned patterns to testing data to be combined by the fusion center, we also consider how to effectively perform such tasks with (1) limited training data; (2) lost data in both training and testing stages; and (3) sensor bias. Of concern here is the performance of generalization capabilities from training to testing stages, using various ANN implementations, under variable communication and computation constraints. A ballistic target tracking application is used to demonstrate the performance of our learning-based approach.

The remainder of this paper is organized as follows: We first review the estimation/fusion concepts and two closed-form (i.e., non-learning-based) fusers in Section II and the fundamentals of ANNs and the learning algorithm called backpropagation in Section III. Next in Section IV, the enhanced versions of a standard backpropagation algorithm are considered that could improve the generalization capabilities of the training-based fuser to new data. We consider the communication and computation constraints in Sections V and VI respectively, highlighting the effect of missing data and bias on the training and testing stages. Simulation results of a ballistic target tracking application are presented and analyzed in Section VII before we conclude this paper in Section VIII.

II. State Estimation and Fusion

The goal of a state estimator is to extract the state information $\mathbf{x}$ from a measurement $\mathbf{z}$ corrupted by noise; this is done by sequentially running a filter that outputs the state estimate $\hat{\mathbf{x}}$ and its associated error covariance matrix $\mathbf{P}$. A sensor in a long-haul network assumes the role of such a filter. The fusion center, on the other hand, simply combines the estimates generated by sensors using a certain fusion rule.

There are some well-known closed-form fusers. For instance, in tracking applications, the track-to-track fuser (T2TF) [3] is a fuser optimal in the linear minimum mean-square error (LMMSE) sense. The fused state estimate $\hat{\mathbf{x}}_F$ and its error covariance $\mathbf{P}_F$ are defined for two sensors as

$$\mathbf{P}_F = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1}, \tag{1}$$

$$\hat{\mathbf{x}}_F = \mathbf{P}_F(\mathbf{P}_1^{-1}\hat{\mathbf{x}}_1 + \mathbf{P}_2^{-1}\hat{\mathbf{x}}_2), \tag{2}$$

where $\hat{\mathbf{x}}_i$ and $\mathbf{P}_i$ are the state estimate and error covariance from sensor i, respectively. The error cross-covariance $\mathbf{P}_{ij}$, the error cross-covariance between sensors i and j, has been omitted from the equations since it is generally unknown. This rule can be readily extended to multiple sensors as well. An important feature of this fuser is that $\mathbf{P}_F$ often promises an estimation error that is lower than the actual error; hence, the fuser is sometimes considered as an "optimistic fuser" in the literature [3].

In another fusion method – the covariance intersection (CI) algorithm, the geometric intersection of the individual covariance ellipses is considered as the error covariance of the fused estimate. The intersection is characterized by the convex combination of sensor covariances:

$$\mathbf{P}_F = (\omega_1\mathbf{P}_1^{-1} + \omega_2\mathbf{P}_2^{-1})^{-1} \tag{3}$$

$$\hat{\mathbf{x}}_F = \mathbf{P}_F\left(\omega_1\mathbf{P}_1^{-1}\hat{\mathbf{x}}_1 + \omega_2\mathbf{P}_2^{-1}\hat{\mathbf{x}}_2\right), \quad \omega_1 + \omega_2 = 1 \tag{4}$$

where $\omega_1, \omega_2 > 0$ are weights to be determined (e.g., by minimizing the determinant of $\mathbf{P}_F$). A fast CI algorithm has recently been proposed in [31] where the weights are found based on an information-theoretic criterion so that ω_1 and ω_2 can be solved for analytically as follows:

$$\omega_1 = \frac{D(p_1, p_2)}{D(p_1, p_2) + D(p_2, p_1)}, \tag{5}$$

where $D(p_A, p_B)$ is the Kullback-Leibler (KL) divergence from $p_A(\cdot)$ to $p_B(\cdot)$, and $\omega_2 = 1 - \omega_1$. When the underlying estimates are Gaussian, the KL divergence can be computed as

$$D(p_i, p_j) = \frac{1}{2}\left[\ln\frac{|\mathbf{P}_j|}{|\mathbf{P}_i|} + \mathbf{d}_X^T\mathbf{P}_j^{-1}\mathbf{d}_X + \mathrm{tr}(\mathbf{P}_i\mathbf{P}_j^{-1}) - k\right], \tag{6}$$

where $\mathbf{d}_X = \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j$, k is the dimensionality of $\hat{\mathbf{x}}_i$, and $|\cdot|$ and $\mathrm{tr}(\cdot)$ denote the determinant and trace respectively. This fast-CI fuser can also be extended to more than two sensors [31], although the equations are somewhat more involved. It is important to note that this fuser, or any CI-based fuser, is "pessimistic" in the sense that $\mathbf{P}_F$ indicates a worse-than-actual error performance. This can be useful in practice since $\mathbf{P}_F$ provides a conservative measure of the error performance.

In contrast to these closed-form linear fusers, we are primarily interested in using nonlinear functions to fuse sensor data, which may potentially yield better results than with linear fusion. Field tests with known true target states facilitate learning-based fuser design as many types of regression analysis methods exist and can be used to learn or compute the parameters of the fusing function we wish to estimate from these field tests. In this work, we look at the use of Artificial Neural Networks (ANNs) for fusing the state estimates as they are known to be able to approximate any continuous function given sufficient parameters.

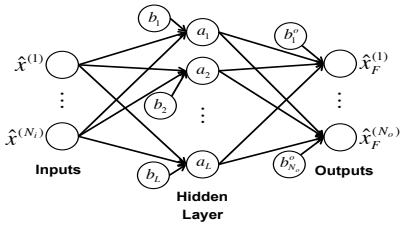


Fig. 1: An example of a three-layer feedforward neural network

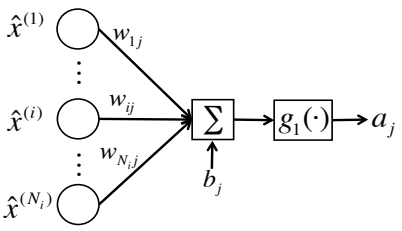


Fig. 2: Weights from N_i inputs to the hidden node j

III. Artificial Neural Networks (ANNs): An Overview

ANNs are a class of statistical models that consist of sets of adaptive numerical parameters that are tuned by a learning algorithm, and are capable of approximating non-linear functions of their inputs. There are different types of ANNs, but we focus on the simplest feedforward neural networks where connections between the units do not form a directed cycle or loop (i.e., no feedback exists in the network).

The structure of a three-layer feedforward neural network is shown in Fig. 1. This network consists of an input layer, a hidden layer, and an output layer, interconnected by weights represented by the arrows between the layers. Information moves forward in one direction, from the input nodes, through the hidden nodes, and to the output nodes. In our settings, the inputs $\hat{x}^{(1)}$, ..., $\hat{x}^{(N_i)}$ can be the state estimates from the sensors, and the outputs $\hat{x}_F^{(1)}$, ..., $\hat{x}_F^{(N_o)}$ are the global (fused) state estimates. There is also a bias unit that is connected to each node in addition to the input nodes. The output of the j^{th} hidden node, a_j, is given by

$$a_j = g_1\left(\sum_{i=1}^{N_i} w_{ij}\hat{x}^{(i)} + b_j\right), \tag{7}$$

where the parameters w_{ij} and b_j are the weights and biases respectively, $\hat{x}^{(i)}$ is the i-th input element to the ANN, and $g_1(\cdot)$ is a nondecreasing function called the activation function, typically a bounded function such as the sigmoid. A simple diagram illustrating this node function is shown in Fig. 2. If we concatenate all of the hidden node outputs a_j into a vector $\mathbf{a} = [a_1, ..., a_L]^T$ and let $\mathbf{g_1}(\cdot)$ denote the function (of appropriate dimension, e.g., L here) with a vector argument where element-wise mapping $g_1(\cdot)$ is performed, then we can write the hidden node outputs as

$$\mathbf{a} = \mathbf{g_1}(\mathbf{W}_H^T\hat{\mathbf{x}} + \mathbf{b}_H), \tag{8}$$

where $\mathbf{W}_H = [w_{ij}]_{N_i \times L}$ is the matrix of weights whose transpose is multiplied by the input vector $\hat{\mathbf{x}} = [x^{(1)}, ..., x^{(N_i)}]^T$, and $\mathbf{b}_H = [b_1, ..., b_L]^T$ is a vector of the biases for each hidden node. The fused output of our network, $\hat{\mathbf{x}}_F$, which is an N_o-dimensional vector, is then given by

$$\hat{\mathbf{x}}_F = \mathbf{g_2}(\mathbf{W}_o^T\mathbf{a} + \mathbf{b}_o), \tag{9}$$

where $\mathbf{W}_o = [w_{ij}^o]_{L \times N_o}$ is another weight matrix, $\mathbf{b}_o = [b_1^o, ...b_{N_o}^o]^T$ is the vector of biases for each output, and $g_2(\cdot)$ is another activation function that is performed element-wise in $\mathbf{g_2}(\cdot)$.

When the target outputs are available, a well-known approach to determining the neural network parameters is called backpropagation. Backpropagation is based on gradient descent; the weights and biases are initialized with random values and then iteratively updated to reduce the error according to some user-defined error function, e.g., the mean-squared error. Once the network parameters are learned from training data, new inputs can simply be fed into the neural network to obtain fused outputs. The d-dimensional parameter vector $\mathbf{w}$ that contains all of the neural network parameters is

$$\begin{aligned}\mathbf{w} = [&\mathbf{W}_H(1,1), \mathbf{W}_H(1,2), ..., \mathbf{W}_H(L,N_i),\\ &\mathbf{b}_H(1), ..., \mathbf{b}_H(L), \mathbf{W}_o(1,1), ..., \mathbf{b}_o(N_o)]^T.\end{aligned} \tag{10}$$

Note that if we have N_i network inputs, L hidden nodes, and N_o network outputs, then the dimension of $\mathbf{w}$ is $d = L(N_i+1) + N_o(L+1)$. The state estimates from each sensor are used as a network input, so $N_i = Ns$ since there are N sensors generating s-dimensional state estimates, and $N_o = s$ so that the neural network outputs a s-dimensional fused state estimate.

Assume we want to minimize some function $S(\mathbf{w})$ with respect to the vector $\mathbf{w}$. Then the increment in each step of the Gauss-Newton method, an iterative method that uses the first and second derivatives of an error function to find a point where the derivative is zero, would be

$$\Delta\mathbf{w} = -[\nabla^2 S(\mathbf{w})]^{-1}\nabla S(\mathbf{w}), \tag{11}$$

where $\nabla^2 S(\mathbf{w})$ and $\nabla S(\mathbf{w})$ are the Hessian and the gradient, respectively, of $S(\mathbf{w})$. If we let $S(\mathbf{w})$ be a sum-of-squares function over m training patterns and $f(\cdot)$ be the learned function that maps the ANN input to output, e.g.,

$$\begin{aligned}S(\mathbf{w}) &= \sum_{k=1}^{m}(y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2\\ &= \sum_{k=1}^{m}(e_k(\mathbf{w}))^2 = \mathbf{e}(\mathbf{w})^T\mathbf{e}(\mathbf{w}),\end{aligned} \tag{12}$$

where $y^{(k)}$ is the true target state in the k^{th} training pattern, $e_k(\mathbf{w}) = y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w})$ and $\mathbf{e}(\mathbf{w}) = [e_1(\mathbf{w}), ..., e_m(\mathbf{w})]^T$, then we can approximate the Hessian and the gradient with the Jacobian, $\mathbf{J}$, of the error vector $\mathbf{e}(\mathbf{w})$, ignoring the common coefficient 2, and rewrite the weight update equation as

$$\Delta\mathbf{w} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}), \tag{13}$$

where

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_1(\mathbf{w})}{\partial w_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_m(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_m(\mathbf{w})}{\partial w_d} \end{bmatrix}. \tag{14}$$

The Levenberg-Marquardt (LM) modification[1] to the Gauss-Newton method discussed above is

$$\Delta \mathbf{w} = -(\mathbf{J}^T\mathbf{J} + \eta\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}), \tag{15}$$

where $\mathbf{I}$ is the identity matrix, and $\eta > 0$ is a damping factor, which is adjusted at each iteration of the parameter update. If a step results in an increased $S(\mathbf{w})$, then η is multiplied by some factor ν, and if a step results in a decreased $S(\mathbf{w})$, then η is divided by ν. The Jacobian of $\mathbf{e}(\mathbf{w})$ can be computed using the backpropagation approach as described in [13]. As the combination of the steepest descent algorithm and the Gauss-Newton method, the LM algorithm is used in this study to train the ANNs as it can be implemented efficiently and is considered to be one of the faster training methods with relatively good convergence performance [34].

IV. Improving the Generalizability of the ANN Training

Overfitting is one of the most critical issues during neural network training, or more broadly, for any learning-based design. The error on the training set is driven to a small value, but when new testing data are input to the learned network, the error can be potentially large. In other words, the network has memorized the training examples, but it has not learned to generalize to new situations. In this section, we consider a few techniques for improving the generalizability of the LM-based ANN training.

A. Multiple ANNs

It is preferable to train several networks to ensure that a network with good generalization is found. Simple as it sounds, using multiple neural networks to train the same set of data[2] and averaging their outputs might yield superior performance by diversifying the training process. Typically each backpropagation training session starts with different initial weights and biases and these conditions may lead to very different solutions for the same problem. In addition, the network structure can be diversified as well by using a different number of hidden nodes or even hidden layers. Just a slight change in the structure would result in a different neural network with a completely new set of parameters. This approach can be especially helpful for a small and noisy dataset.

B. Bayesian Regularization

Another method for improving generalization capabilities of a learning algorithm is called regularization. This involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set, as shown in Eq. (12). An additional term is added to the original performance function, usually in the form of a cost or penalty function for complexity, such as restrictions for smoothness or bounds on the vector space norm [11]. Examples of such regularization techniques (typically applied to linear models) include $\mathcal{L}_2$, $\mathcal{L}_1$, and the more recent $\mathcal{L}_{1/2}$ [32] regularization methods. In this work, though, we investigate two regularization techniques specifically designed for ANNs.

MacKay [23] proposed a Bayesian framework to overcome the problem in interpolating noisy data, which can be directly applied to the neural network learning problem. In this framework, the weights and biases of the network are assumed to be random variables with specified distributions. The regularization parameters are related to the unknown variances associated with these distributions. Another goal is to reduce the effective number of parameters used by the model – in this case, the number of network parameters actually needed to solve a particular problem. Hence, MacKay's Bayesian regularization expands the original sum-of-squares cost function to search for the minimum error using "minimum" parameters by introducing two Bayesian hyperparameters, α and β, to balance the dual needs during the learning process. In particular, the objective function to be minimized is in the form of

$$\begin{aligned} S(\mathbf{w}) &= \beta \sum_{k=1}^{m} (y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2 + \alpha \|\mathbf{w}\|^2 \\ &= \beta \cdot SSE + \alpha \cdot SSP. \end{aligned} \tag{16}$$

It can be seen that in addition to the sum-of-squared-errors term, an additional term, the sum-of-squared parameters (SSP), is added to the objective function. The SSP is simply the square of the $\mathbb{L}_2$ norm of the d-dimensional vector $\mathbf{w}$ introduced in Eq. (10). The hyperparameters are updated as

$$\alpha = \frac{\gamma}{SSP}, \text{ and } \beta = \frac{m - \gamma}{SSE}, \tag{17}$$

where

$$\gamma = d - \alpha \operatorname{tr}((\mathbf{J}^T\mathbf{J})^{-1}) \tag{18}$$

is considered as the number of "effective" parameters (weights and biases) being used by the network, thus giving an indication on how complex the network should be. These parameter update steps[3] can be easily incorporated into each iteration of the LM update equation as in Eq. (15). Essentially, this Bayesian regularization technique can be considered as a special $\mathcal{L}_2$ regularizer with iteratively updated hyperparameters.

C. Regularization Using Error Covariance Matrices

It is noted that the sensors also provide additional information regarding the state estimates: the error covariances. It is

[1]There are other variations of the algorithm presented here. For example, there are different ways to select the initial damping factor η and increase/decrease its value. Also, the matrix consisting of only the diagonal elements of $\mathbf{J}^T\mathbf{J}$ may be used in place of the identity matrix $\mathbf{I}$.

[2]In our setting, only a small set of training data from previous field tests are available and the testing stage occurs in the form of real-time tracking, hence we do not pursue cross-validation [1] approaches where data are iteratively divided into training, validation, and testing sets to improve the generalization capabilities.

[3]Poland [25] provided another way to update α as

$$\alpha = d/(2.0 \cdot SSW + \operatorname{tr}((\mathbf{J}^T\mathbf{J})^{-1}))$$

which has been shown to provide more smooth and robust updates compared to Eq. (17).

still an open question of how to best utilize this information, if at all, when designing or using these nonlinear fusers. It is proposed here that these error covariance estimates can be used when training the neural network to improve its generalization capability. If we assume that the state estimates from the sensors are equal to the true states plus noise, that is,

$$\hat{\mathbf{x}}_i = \mathbf{x} + \mathbf{n}_i, \tag{19}$$

where i is the sensor index, and $\mathbf{n}_i$ is zero-mean white Gaussian noise with covariance $\mathbf{P}_i$, then in fusing these state estimates, the neural network will also transform and fuse the noise. We can therefore try to further reduce the variance of the output in addition to the overall error by adding the output variance to the objective function that we minimize when determining the neural network parameters. But if one were to use a nonlinear activation function such as the tangent hyperbolic sigmoid function, $f(x) = \frac{2}{1+e^{-2x}} - 1$, then the variance becomes quite difficult to determine analytically. However, from [5], we have the following upper bound on the variance of the function $g(X)$ of a random variable X, if $X \sim \mathcal{N}(\mu, \sigma^2)$:

$$\mathbb{V}\text{ar}[g(X)] \leq \sigma^2 \mathbb{E}[g'(X)]^2, \tag{20}$$

where $g'(X)$ is the derivative of $g(X)$. We can write the output $\hat{\mathbf{x}}_F$ as a function of the noise $\mathbf{n} = [\mathbf{n}_1^T, ..., \mathbf{n}_N^T]^T$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$. $\mathbf{P}$ is a block diagonal matrix where the blocks are $\mathbf{P}_1$ through $\mathbf{P}_N$:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{0}_{s\times s} & \cdots & \mathbf{0}_{s\times s} \\ \mathbf{0}_{s\times s} & \mathbf{P}_2 & \cdots & \mathbf{0}_{s\times s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{s\times s} & \cdots & \mathbf{0}_{s\times s} & \mathbf{P}_N \end{bmatrix}, \tag{21}$$

where s is the number of states in the true target state $\mathbf{x}$. For the ANN fuser, if we let the activation function for each hidden node be $g_1(x) = \frac{2}{1+e^{-2x}} - 1$ and the output activation function $g_2(x) = x$, then the ANN fuser output would be given by

$$\begin{aligned} \hat{\mathbf{x}}_F &= \mathbf{W}_o^T \mathbf{g_1}\left(\mathbf{W}_H^T \hat{\mathbf{x}}\right) + \mathbf{b}_o \\ &= \mathbf{W}_o^T \mathbf{g_1}\left(\mathbf{W}_H^T \mathbf{x} + \mathbf{b}_H + \mathbf{W}_H^T \mathbf{n}\right) + \mathbf{b}_o. \end{aligned} \tag{22}$$

Now if we let $\boldsymbol{\theta} = \mathbf{W}_H^T \mathbf{n}$, which is simply a linear transformation of the Gaussian random variable $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$, then $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_H^T \mathbf{P} \mathbf{W}_H)$. Therefore, using the vector form of the variance inequality Eq. (20) also found in [5], we have

$$\mathbb{V}\text{ar}[\hat{\mathbf{x}}_F(\boldsymbol{\theta})] \leq \mathbb{E}\left[\frac{\partial \hat{\mathbf{x}}_F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] \cdot \mathbf{W}_H^T \mathbf{P} \mathbf{W}_H \cdot \mathbb{E}\left[\frac{\partial \hat{\mathbf{x}}_F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right]^T. \tag{23}$$

However, even determining $\mathbb{E}\left[\partial \hat{\mathbf{x}}_F(\boldsymbol{\theta}) / \partial \boldsymbol{\theta}\right]$ is still quite complicated. Suppose $\boldsymbol{\theta}_0 = \mathbf{W}_H^T \mathbf{x} + \mathbf{b}_H$, then we can obtain the following relationship:

$$\mathbb{V}\text{ar}[\hat{\mathbf{x}}_F(\boldsymbol{\theta})] < 16 \mathbf{W}_o^T \exp(\boldsymbol{\Theta}) \mathbf{W}_H^T \mathbf{P} \mathbf{W}_H \exp(\boldsymbol{\Theta}) \mathbf{W}_o, \tag{24}$$

in which $\boldsymbol{\Theta} = 8 \mathbf{W}_H^T \mathbf{P} \mathbf{W}_H \odot \mathbf{I}_L + 4 \boldsymbol{\theta}_0^D$, where $\odot$ denotes the Hadamard product, $\mathbf{I}_L$ is the $L \times L$ identity matrix, $\boldsymbol{\theta}_0^D$ is the diagonal matrix whose diagonal elements are the elements of the vector $\boldsymbol{\theta}_0$, and finally, $\exp(\cdot)$ denotes the exponential of a diagonal matrix. The proof of Eq. (24) can be found in the Appendix. Still, this multiplicative relationship is not neat enough for use in the additional term for error regularization, considering the complexity of the matrix $\boldsymbol{\Theta}$. Therefore, to simplify, an extended form of the term $\mathbf{W}_H^T \mathbf{P} \mathbf{W}_H$ (that can yield a scalar) will be used to add to the error function for minimization. We can modify the LM algorithm described in the previous section to incorporate this additional term using a tradeoff parameter λ:

$$S(\mathbf{w}) = \sum_{k=1}^{m} (y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2 + \lambda \mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}, \tag{25}$$

where

$$\boldsymbol{\Sigma} = \left[\begin{array}{c|c} \mathbf{P}_R & \mathbf{0}_{NLs\times(d-NL)s} \\ \hline \mathbf{0}_{(d-NL)s\times NLs} & \mathbf{0}_{(d-NL)s\times(d-NL)s} \end{array}\right], \tag{26}$$

and $\mathbf{P}_R \in \mathbb{R}^{(NLs)\times(NLs)}$ is a block diagonal matrix with each block consisting of the matrix $\mathbf{P}$, repeated L times (once for each hidden node):

$$\mathbf{P}_R = \begin{bmatrix} \mathbf{P} & \mathbf{0}_{Ns\times Ns} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P} \end{bmatrix}. \tag{27}$$

Recall that N is the number of sensors and s is the number of states, so each block diagonal matrix $\mathbf{P}$ is of size $Ns \times Ns$. Following the LM approach, when we update $\mathbf{w}$ with $\Delta\mathbf{w}$, we get the following update to our error function $S(\mathbf{w})$:

$$\begin{aligned} S(\mathbf{w}+\Delta\mathbf{w}) &= \sum_{k=1}^{m} (y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}+\Delta\mathbf{w}))^2 \\ &\qquad + \lambda(\mathbf{w}+\Delta\mathbf{w})^T \boldsymbol{\Sigma}(\mathbf{w}+\Delta\mathbf{w}) \\ &= \sum_{k=1}^{m} (e_k(\mathbf{w}+\Delta\mathbf{w}))^2 + \lambda(\mathbf{w}+\Delta\mathbf{w})^T \boldsymbol{\Sigma}(\mathbf{w}+\Delta\mathbf{w}) \\ &= \left[\mathbf{e}(\mathbf{w}+\Delta\mathbf{w})\right]^T \mathbf{e}(\mathbf{w}+\Delta\mathbf{w}) \\ &\qquad + \lambda(\mathbf{w}+\Delta\mathbf{w})^T \boldsymbol{\Sigma}(\mathbf{w}+\Delta\mathbf{w}) \\ &\approx \left[\mathbf{e}(\mathbf{w}) + \nabla\mathbf{e}(\mathbf{w})\Delta\mathbf{w}\right]^T \left[\mathbf{e}(\mathbf{w}) + \nabla\mathbf{e}(\mathbf{w})\Delta\mathbf{w}\right] \\ &\qquad + \lambda(\mathbf{w}+\Delta\mathbf{w})^T \boldsymbol{\Sigma}(\mathbf{w}+\Delta\mathbf{w}), \end{aligned} \tag{28}$$

where the following substitutions are made: $y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}+\Delta\mathbf{w}) = e_k(\mathbf{w}+\Delta\mathbf{w})$ (line 1 to line 2), and $\mathbf{e}(\mathbf{w}+\Delta\mathbf{w}) = [e_1(\mathbf{w}+\Delta\mathbf{w}), ..., e_m(\mathbf{w}+\Delta\mathbf{w})]^T$ (line 2 to line 3). In the fourth line of Eq. (28), we approximate and substitute $\mathbf{e}(\mathbf{w}+\Delta\mathbf{w})$ with its first-order Taylor expansion about $\mathbf{w}$. Furthermore, we know that at the minimum of $S(\mathbf{w}+\Delta\mathbf{w})$, the gradient with respect to $\Delta\mathbf{w}$ is zero, so we have:

$$\begin{aligned} &\frac{dS(\mathbf{w}+\Delta\mathbf{w})}{d\Delta\mathbf{w}} \\ \approx& 2\left[\nabla\mathbf{e}(\mathbf{w})\right]^T \left[\mathbf{e}(\mathbf{w}) + \nabla\mathbf{e}(\mathbf{w})\Delta\mathbf{w}\right] + 2\lambda\boldsymbol{\Sigma}(\mathbf{w}+\Delta\mathbf{w}) \\ =& 2\mathbf{J}^T\mathbf{e}(\mathbf{w}) + 2\mathbf{J}^T\mathbf{J}\Delta\mathbf{w} + 2\lambda\boldsymbol{\Sigma}\mathbf{w} + 2\lambda\boldsymbol{\Sigma}\Delta\mathbf{w} \\ =& 0, \end{aligned} \tag{29}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{e}(\mathbf{w})$.

The parameter update equation can then be computed as $\Delta\mathbf{w} = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{\Sigma})^{-1}(\mathbf{J}^T\mathbf{e}(\mathbf{w}) + \lambda\mathbf{\Sigma}\mathbf{w})$, and with the LM modification to include the damping factor, we obtain the final parameter update equation which incorporates the error covariance estimates into the training using the LM method:

$$\Delta\mathbf{w} = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{\Sigma} + \eta\mathbf{I})^{-1}(\mathbf{J}^T\mathbf{e}(\mathbf{w}) + \lambda\mathbf{\Sigma}\mathbf{w}). \quad (30)$$

Note that when $\lambda = 0$, Eq. (30) is reduced to the standard LM update Eq. (15).

The inclusion of $\mathbf{\Sigma}$ in the regularization term of the objective function (i.e., Eq. (25)) is meant to limit the effect of large estimation errors (as represented by the large error covariance entries in individual $\mathbf{P}$) during the learning process. Indeed, when $\mathbf{\Sigma} = \mathbf{I}_{d\times d}$, then Eq. (25) is reduced to the standard $\mathcal{L}_2$ regularization. However, a close examination of Eqs. (26) and (27) reveals that only a subset of the elements in $\mathbf{w}$ are used in the regularizer, i.e., those in the weight matrix $\mathbf{W}_H$. Therefore, our covariance-based regularization method is somewhat of a greedy approach in that only those parameters directly related to the sensor inputs are incorporated into the regularization term. While empirically it eventually converges just like the other LM-based methods (as verified in our extensive simulations in Section VII), the convergence speed depends upon the error covariance information supplied by the sensors that affects the weight update step size as shown in Eq. (30). In addition, increased computational complexity (e.g., matrix multiplication and inversion) leads to overall longer time toward convergence, a trade-off for achieving improved error performance to be shown in Section VII.

V. Training and Testing with Missing Data

Having provided the training algorithm and techniques for its generalization, we now focus on the communication and computation constraints and their impact on the learning process. To begin with, the communication link loss and delay reduces the amount of data that can be effectively used for training and testing purposes, which must be accounted for by the fusion center in devising efficient learning-based fusers.

A. Data Loss during Training

As training is performed based on data gathered from historical field tests, usually the true target states are obtained via some other sources that are not subject to the same link loss and delay conditions as from the sensors, whereas some sensor estimates may have never arrived. As such, the fusion center simply has a smaller set of training data that are input to an ANN. Suppose the link-level loss rate is p_L across all training patterns, then the effect this loss has on training can be interpreted in two different ways. First, the actual number of available training patterns $\tilde{m}$ is often less than m, and thus the training objective simply uses SSE terms for the available $\tilde{m}$ patterns. In other words, in Eq. (12), based on the actual pattern availability, we have

$$S(\mathbf{w}) = \sum_{k=1}^{\tilde{m}}(e_k(\mathbf{w}))^2 + R_g(\mathbf{w}) = \tilde{\mathbf{e}}(\mathbf{w})^T\tilde{\mathbf{e}}(\mathbf{w}) + R_g(\mathbf{w}), \quad (31)$$

where $\tilde{\mathbf{e}}$ is the error vector containing $\tilde{m}$ elements and $R_g(\mathbf{w})$ is the regularization term containing the vector $\mathbf{w}$. Alternatively, the number of available patterns follows the following binomial distribution: $\tilde{m} \sim \mathcal{B}(m(1-p_L), mp_L(1-p_L))$. As a result, the original SSE term in Eq. (12) can now be expressed as

$$S(\mathbf{w}) = \sum_{k=1}^{m} i_k(e_k(\mathbf{w}))^2 + R_g(\mathbf{w}) = \mathbf{e}(\mathbf{w})^T\mathbf{\Lambda}\mathbf{e}(\mathbf{w}) + R_g(\mathbf{w}), \quad (32)$$

where i_k is the associated Bernoulli random variable, i.e., $i_k = 1$ with probability $1 - p_L$ and 0 otherwise, and the diagonal matrix $\mathbf{\Lambda} = \mathbf{i}^D$, where $\mathbf{i} = [i_1, i_2, ..., i_m]^T$. These results can be easily extended to patterns with different link loss rates.

B. Data Loss and Delay during Testing

The testing phase of the learning process, namely, to apply the learned ANN parameters to new sensor inputs, coincides with the actual online tracking process. The loss and delay inherent over the communication link, as a result, plays a somewhat different role compared to the off-line training phase.

In most tracking tasks that impose nearly real-time performance, the fusion center is required to finalize its fused state within a very tight deadline, by which time one or more sensor estimates may have not yet arrived (although they might arrive later). Thus, in contrast to the off-line training phase, the communication delay is a major limiting factor in the desired performance.

The structure of the ANN-based learning requires that the input data be complete in order to generate the desired output. Whereas it is in general difficult to interpolate missing values in the training phase because the underlying time-domain relationship between the training data is not readily available, it is possible that the fusion center uses adjacent sensor estimates that are available to interpolate the missing ones, based on the knowledge it has on the possible trajectory of the underlying target being tracked. Therefore, when some of the inputs are missing, the fusion center can use prediction and retrodiction (when applicable) techniques that utilize previous and subsequent sensor estimates, respectively, to manually fill in the missing inputs [22]. Of course, should all other methods fail (as under extremely lossy link conditions), the fusion center can still bypass the neural network altogether by using prediction on its previously fused states to generate the immediate fused value, although this is likely to compromise the accuracy performance by a large degree.

VI. Sensor Bias and Error Regularization

Another concern arising from the long-haul tracking is the sensor information quality. Notably, sensor biases could potentially degrade the fusion performance. To be answered are questions such as how the ANN-based fuser would perform with the presence of variable bias levels and whether the learning process could potentially improve the training and testing data quality. In this section, we discuss bias-related issues pertaining to the ANN training and testing.

An important fact is that a sensor i might not be aware of its biases. Sometimes the biases are an integral part of the filtering process, due to linearization and coordinate conversion; while other times it is due to poor calibration or environmental factors that result in sensor measurement biases. As such, the error covariance matrix $\mathbf{P}_i$ – which is also supplied to the fusion center – is likely to be an optimistic measure of the actual estimation error. Fortunately, thanks to the available true target states during the training phase, the fusion center can evaluate the potential estimation biases from the sensor data and "correct" the error covariance matrix during its training process.

To illustrate, consider a generic sensor estimate as $\hat{\mathbf{x}}_i$ whereas the true target state is $\mathbf{x}$. Then the error covariance matrix is defined as $\mathbf{P}_i = \mathbb{E}[(\hat{\mathbf{x}}_i - \mathbf{x})(\hat{\mathbf{x}}_i - \mathbf{x})^T]$. Now suppose $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{i,u} + \mathbf{b}_i$, where $\hat{\mathbf{x}}_{i,u}$ is a (hypothetically) unbiased estimate for sensor i and $\mathbf{b}_i$ is the bias vector, then we have

$$\begin{aligned}\mathbf{P}_i &= \mathbb{E}[(\hat{\mathbf{x}}_i - \mathbf{x})(\hat{\mathbf{x}}_i - \mathbf{x})^T] \\ &= \mathbb{E}[(\hat{\mathbf{x}}_{i,u} - \mathbf{x} + \mathbf{b}_i)(\hat{\mathbf{x}}_{i,u} - \mathbf{x} + \mathbf{b}_i)^T] \\ &= \mathbb{E}[(\hat{\mathbf{x}}_{i,u} - \mathbf{x})(\hat{\mathbf{x}}_{i,u} - \mathbf{x})^T] \\ &\qquad + \mathbb{E}[\mathbf{b}_i\mathbf{b}_i^T] + 2\mathbb{E}[(\hat{\mathbf{x}}_{i,u} - \mathbf{x})\mathbf{b}_i^T] \\ &= \mathbf{P}_{i,u} + \mathbb{E}[\mathbf{b}_i\mathbf{b}_i^T] + 2\mathbb{E}[(\hat{\mathbf{x}}_{i,u} - \mathbf{x})\mathbf{b}_i^T] \\ &\approx \mathbf{P}_{i,u} + \overline{\mathbf{b}_i\mathbf{b}_i^T}, \end{aligned} \tag{33}$$

where $\mathbf{P}_{i,u}$ is the error covariance matrix for an unbiased estimate. In the extreme case where a sensor is fully oblivious to its bias, this is the matrix it communicates to the fusion center. The last line has used two approximations: (1) the bias is largely uncorrelated with the true target state, and (2) the empirical average, as denoted by the term $\overline{\mathbf{b}_i\mathbf{b}_i^T}$, is used for the expectation of the outer (tensor) product of the bias vector and itself.

In an ideal scenario, the estimation bias at a sensor is deterministic and consistent, and the fusion center can easily obtain the bias vector and subtract it from the associated sensor estimates during the testing stage. However, with more complex forms of bias, instead of correcting the sensor estimates, the fusion center may want to use Eq. (33) for error regulation so that the updated matrices more closely match the actual error levels of the sensor estimates. Doing so might be of better benefit to the prediction (and retrodiction) when the fusion center encounters missing data during online tracking. The performance comparison using different error regularization methods will be shown in the next section.

VII. Performance Evaluation

In this section, the performance of the ANN-based fusers is evaluated for a coasting ballistic target tracking application via simulations. The tracking performance is characterized by position root-mean-square errors (RMSEs) during the testing stage. Different configurations during the learning process are considered, along with variable communication and computation constraints. The tracking performance with ANN-based fusers is also compared against that using the track-to-track fuser and fast-CI fuser introduced in Section II.

A. Simulation Models

The trajectory of a ballistic target, from launch to impact, is divided into three basic phases: boost, coast, and reentry. We model two sensors tracking a ballistic target in the exo-atmospheric coast phase whose trajectory is determined by a nonlinear state-space model. These sensors generate state estimates of the target's position and velocity, which are subsequently sent to the fusion center where they are combined to produce a final state estimate of the target. Furthermore, state-dependent errors are introduced with the use of a simple radar model in simulating the sensor measurements.

1) Target Model: The state-space model of a ballistic coast target has the form $\dot{\mathbf{x}} = [\mathbf{v}\ \ \mathbf{a}]^T$, where $\mathbf{x} = [\mathbf{p}^T\ \ \mathbf{v}^T]^T$ is the state vector consisting of the target's position $\mathbf{p} = [x\ \ y\ \ z]^T$ and velocity $\mathbf{v} = [\dot{x}\ \ \dot{y}\ \ \dot{z}]^T$ in the Earth-centered inertial (ECI) coordinate system (i.e., the coordinate system does not rotate; it is fixed relative to the "fixed stars", and its origin is at the center of the Earth [18]).

In the coast phase, gravity is considered to be the dominating force acting on a ballistic target, so the total acceleration is $\mathbf{a} = \mathbf{a}_G$, where $\mathbf{a}_G$ is the gravitational acceleration. Assuming a spherical Earth model [18], we have $\mathbf{a}_G = -(\mu/\|\mathbf{p}\|^3)\mathbf{p}$, where $\mathbf{p}$ is the target position vector from the Earth's center to the target, $\|\mathbf{p}\| = \sqrt{x^2+y^2+z^2}$ is its length, and $\mu = 3.986012 \times 10^5\ \mathrm{km}^3/\mathrm{s}^2$ is the Earth's gravitational constant. The continuous-time model of the system is given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\mu x/r^3 \\ -\mu y/r^3 \\ -\mu z/r^3 \end{bmatrix}, \tag{34}$$

where $r = \sqrt{x^2+y^2+z^2}$. An algorithm for computing the state propagation can be found in [33].

2) Sensor Measurement Model: In tracking applications, the target dynamics are usually modeled in Cartesian coordinates, while the measurements are typically available in sensor coordinates (most often spherical coordinates) [37]. We simulate the measurements following [17] for a ballistic coast target. The measurement model is given by $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{z}_w$, and the measurements of the range (r), elevation (E), and azimuth (A) of the target are computed as follows:

$$\mathbf{z} = \begin{bmatrix} r \\ E \\ A \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \tan^{-1}\left(z/\sqrt{x^2+y^2}\right) \\ \tan^{-1}(x/y) \end{bmatrix} + \mathbf{z}_w, \tag{35}$$

where $\mathbf{z}_w$ is white Gaussian noise with covariance $\mathbf{R} = diag\left([\sigma_r^2\ \ \sigma_E^2\ \ \sigma_A^2]\right)$.

A simplified radar model is used to generate state values for σ_r and σ_E so that the errors are state-dependent and correlated across sensors. We will only consider the error that is dependent on the signal-to-noise ratio (SNR) for both the range and angle measurement accuracy since they usually dominate their overall respective radar error [8]. From [8], we have the following relationship between the standard deviation

of the SNR-dependent random range and angle measurement errors and the SNR: $\sigma_r, \sigma_E, \sigma_A \propto \frac{1}{\sqrt{SNR}}$. The SNR (from the well-known radar range equation) is inversely proportional to $\tilde{r}^4$, where $\tilde{r}$ is the range from the sensor to the target. To simplify, we assume a number of the radar parameters from the radar range equation are constant (e.g., the radar pulse duration, antenna gain, etc.) so that $\sigma_r, \sigma_E, \sigma_A \propto \tilde{r}^2$. The range and elevation errors of a ballistic target/satellite tracking phased array radar, the Cobra Dane, are found in Table 1 of [9] as 15 ft and $0.05°$, respectively. These parameters are used to find reasonable values for scaling the σ_r, σ_E, and σ_A values used in these simulations to generate the state-dependent measurement noise.

3) Generating State Estimates: Since the measurement noise is additive in spherical coordinates, a bias is introduced into the state estimates in Cartesian coordinates. A recursive best linear unbiased estimator (BLUE) was developed in [37] for a linear system and shown to be theoretically optimal (in the mean-squared error sense) among all linear unbiased filters in Cartesian coordinates. This filter is used to generate the state estimates in these simulations to account for the converted measurements. As in [37], the filter is initialized with an effectively large state error covariance and a highly inaccurate state estimate.

4) Communication Loss and Delay Profiles: The message-level loss and delay characteristics are determined by the long-haul link conditions. Again, we assume that each message sent by a sensor is lost during transmission with probability p_L, whereas a probability density function $h(t)$ models the overall delay t that any message experiences to be successfully delivered to the fusion center, and the shifted exponential distribution [22] $h(t) = \exp\left((T-t)/\mu_D\right)/\mu_D$ for $t \geq T$ where T is the fixed initial delay and μ_D is the mean of the additional random delay. We consider the case where $T = 0.5$ s and $\mu_D = 0.3$ s. The fusion deadline D_F describes the time by which the fusion center must have combined the individual sensor estimates and generated a fused estimate. For example, when $p_L = 0.4$, the probability that a sensor estimate is available by the deadline is $(1 - 0.4) \times H(1.5) = 0.579$, where $H(\cdot)$ is the corresponding distribution function for $h(\cdot)$. In our studies, the performance over such a lossy link is compared with that in an ideal communication scenario, i.e., when there is no loss or delay.

5) Training and Testing Data Setup: The initial states of the training and test targets are randomly generated from a normal distribution with the mean set to the following (in km for position, and km/s for velocity):

$$\begin{aligned}\mathbf{x}(0) &= [x(0) \quad \dot{x}(0) \quad y(0) \quad \dot{y}(0) \quad z(0) \quad \dot{z}(0)]^T \\ &= [71.31,\ 0.946,\ 3794.5,\ 3.577,\ 5413.0,\ -5.676]^T,\end{aligned}$$

with a standard deviation of 500 m and 5 m/s for each position and velocity component, respectively. In our default setting, two target trajectories are available for training. Measurement data are generated according to the sensor measurement model introduced earlier, and state estimates (and the associated error covariance matrices) – also available in the training data set – are computed from these measurements for each trajectory. Data are available for each trajectory segment that lasts 30 seconds.

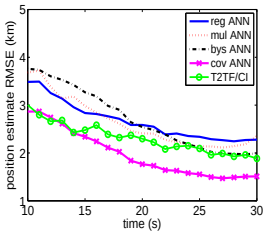
6) ANN Setup: The ANN used for training and testing has one hidden layer with a number of hidden nodes. The number of hidden nodes needed depends on the complexity of the function we are trying to estimate. Using too few hidden nodes may yield a poor approximation to the actual function. Using too many hidden nodes results in overfitting of the data so that while the neural network may precisely give the desired outputs for the training data, it may not generalize well to unseen data. Unfortunately, there is no precise method that provides the optimal number of hidden nodes needed to properly model the data. But a good rule of thumb [15] is that it lies between the number of input and output nodes. Hence, we select three hidden nodes in our default setting. The tangent hyperbolic sigmoid function is used as the activation function from the input to hidden layers.

B. Fusion Performance without Sensor Biases

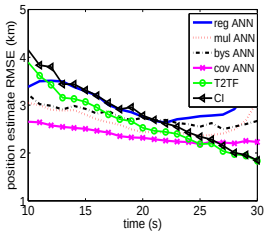
In Fig. 3(a), tracking performances using variations of the ANN-based fuser, including the regular ANN ("reg-ANN"), multiple ANN with 5 component ANNs of the same structure ("mul-ANN"), ANN with Bayesian regularization ("bys-ANN"), and ANN with error covariance regularization ("cov-ANN"), along with that of the T2TF/CI fuser[4] are plotted. From these plots, we observe that although the original ANN fuser does not yield estimates as good as those from the T2TF/CI by using only two available training trajectories, some of the enhanced variations do, especially "cov-ANN". In practice, Bayesian regularization is often applied when combined with other techniques, such as multiple ANNs; as a stand-alone technique, its performance is not always superior to that of the regular ANN as the Bayesian formulation depends on how the trade-off parameters are selected. From our simulations, there are a total of 33 parameters to be determined in the regular ANN, and after applying regularization about 3 or 4 parameters are removed, which means the performance of the two should not differ by much under normal circumstances. In general, the multiple ANN approach would outperform the original ANN-based fuser by varying levels, depending on such factors as how the parameters are initialized in each component network. The covariance error regularization method can effectively use the extra information from the covariance matrices supplied by the sensors and provide improved performance compared to all other techniques.

As the link-level loss goes up, there is an increasing chance that the fusion center cannot receive both sensor estimates for any time epoch during the testing stage. In addition, the training data are also likely to be obtained from past testing data – that experienced similar losses – with the addition of the true target states being revealed later, but not the sensor estimates. The communication delay also reduces the chance that a message is successfully received by the fusion center by a certain deadline. Therefore, with incomplete sensor data,

[4]When p_L = 0, due to the lack of process noise, these two fusers yield the same performance.

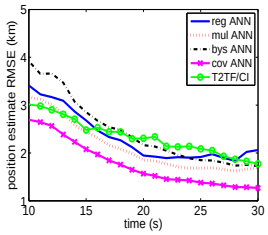


(a) $p_L = 0$, $D_F = 2$ s

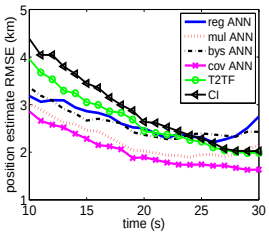


(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 3: Position RMSEs: 2 training trajectories, 3 hidden nodes

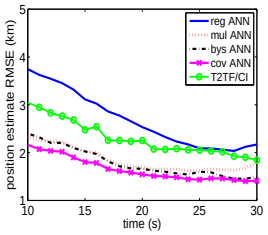


(a) $p_L = 0$, $D_F = 2$ s

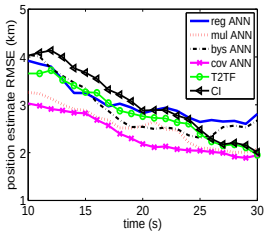


(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 4: Position RMSEs: 5 training trajectories, 3 hidden nodes

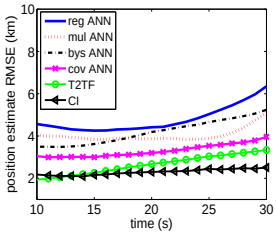


(a) $p_L = 0$, $D_F = 2$ s

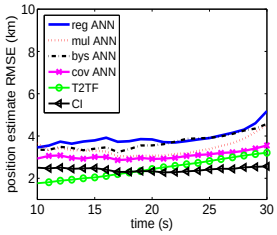


(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 5: Position RMSEs: 2 training trajectories, 6 hidden nodes



(a) $p_L = 0$, $D_F = 2$ s



(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 6: Position estimate RMSE with unlearned bias

the fusion center could use predicted estimates it derives for the individual sensors, and once these predicted values are obtained[5], the inputs to the ANNs are complete. Training data may contain such predicted estimates too. As a result, all the training algorithms can be run with a full set of input to the ANN, even when a portion of the training data are not from the sensors, but rather derived by the fusion center during earlier tracking. Fig. 3(b) shows the performance with 50% sensor data loss in both training and testing data. In the previous case, a fusion deadline D_F of 2 s can effectively reduce any effect of early cutoff; in contrast, here the deadline D_F is 1 s, resulting in even more unavailable original sensor data by the deadline. Comparing among different cases, there is an increase in the tracking error in non-learning based T2TF and CI fusers compared to their respective no-loss case. However, the ANN learning-based approaches are largely able to retain most of their respective lossless tracking performance, thereby demonstrating the major benefit of adopting these fusers in counteracting the negative effect of communication constraints.

Next, we explore how the learning-based fusion performance can be potentially improved with the use of more training data and/or an ANN of a larger size. In Fig. 4, the plots for the same settings except with five trajectories are shown. From these plots, the error curves for ANN-based fusers are variably lower compared to those in Fig. 3, reflecting the benefit of training more data. On the other hand, with the same amount of training data (two trajectories), we increase the number of hidden nodes from three to six, and the error plots are shown in Fig. 5. It can be seen here again that using more hidden nodes can, by and large, improve the error performance somewhat for both loss cases, although the training process becomes more complex with more network parameters to be determined. But regardless of the chosen data or network size, a consistent observation is the superiority of the proposed covariance-based ANN fuser over others in terms of tracking accuracy.

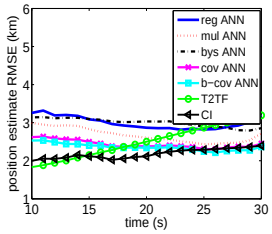
[5]To ensure fair comparison, all fusers adopt the same prediction method.

C. Fusion Performance with Sensor Biases

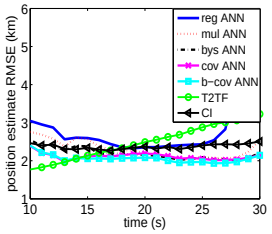
We examine the cases where the training and/or testing data contain biased estimates. First, suppose in the test data, a positive uniform bias is added to the (unbiased) measurement noise at one of the sensors, with its mean magnitude set at 0.1% of the noise level. The fusion performance is shown in Fig. 6. Even such a small bias results in the error curves now trending upward across the board. Due to the mismatch between the training and testing data in terms of bias profiles, the unlearned bias (i.e., the bias not present in the training data) reduces the generalization capability of the trained parameters, resulting in elevated estimation errors across all learning-based cases. It is interesting to note that the learning-based fusers with more than half the original sensor data missing, as in Fig. 6(b), would yield even better tracking performance than those of the same type in the lossless case thanks to the reduced effect of the mismatch between the training and testing data.

Next, suppose in one of the training trajectories data share the same bias profile as described above whereas estimates in the other remain unbiased. The fusion performance is shown in Fig. 7, where performance of the "cov ANN" under the bias correction method introduced earlier has been plotted as well ("b-cov ANN"). We can observe improved generalization performance for both lossless and lossy conditions compared to their respective performance with bias mismatch as shown in Fig. 6. With an additional bias correction step, the covariance regularization method is able to output even better estimates, although from the plots, this improvement is somewhat limited and can be reduced when a higher loss rate is encountered.

Finally, all others kept equal, the mean bias level is now increased to 0.2% of the unbiased measurement noise, and

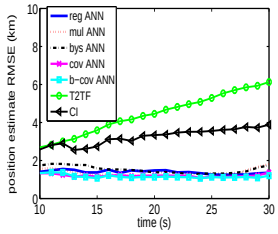


(a) $p_L = 0$, $D_F = 2$ s

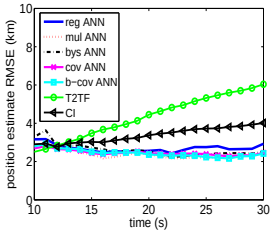


(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 7: Position estimate RMSE with bias correction



(a) $p_L = 0$, $D_F = 2$ s



(b) $p_L = 0.5$, $D_F = 1$ s

Fig. 8: Position estimate RMSE with bias correction: higher bias

the estimation errors are plotted in Fig. 8. Whereas the T2TF and the CI fuser output increasingly error-prone estimates (more so for the T2TF), the ANN-based fusers are able to retain much of their error levels as in the previous case, thereby demonstrating their superior performance in tracking with potentially more biased sensor data. Nevertheless, with such an elevated bias level and over half the original sensor data missing, the improvement in tracking accuracy over the non-learning fusers is reduced, as a result of the increased mismatch level (from extended prediction over biased estimates) between the training and testing data. Thus, in order to improve the generalizability of the training process, one needs to ensure a good match in features such as bias levels between training and testing data.

VIII. Conclusions

In this work, we have provided a quantitative study on the potential benefits of artificial neural network learning-based fusers for sensor fusion in target tracking over long-haul sensor networks. In particular, the effects of variable communication link-level loss and delay conditions as well as sensor bias profiles on the performance of a variety of ANN implementations have been investigated. The extra covariance information provided by the sensors has been shown to provide better generalization guarantees in the presence of the above communication and computation uncertainties. Extensions of this work may be studied in a setting with a much larger training dataset comprising of heterogenous trajectories and/or variable data quality levels, so that prior to training, a classification process should be carried out in order to reduce the potential mismatch between the training and testing datasets and improve the overall generalizability of the learned patterns.

Appendix
Proof of Eq. (24)

Proof: From Eq. (22), let

$$\xi(\theta) = \frac{2}{1+e^{-2(\theta+\theta_0)}} - 1, \tag{36}$$

$\boldsymbol{\theta} = \{\theta_k, k = 1, 2, ..., L\}$, and $\boldsymbol{\theta}_0 = \{\theta_{0,k}, k = 1, 2, ..., L\}$, so that the concatenated L-vector $\boldsymbol{\xi}(\boldsymbol{\theta})$ represents the output of the hidden nodes. Its derivative with respect to $\boldsymbol{\theta}$, $\partial\boldsymbol{\xi}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$, is an $L \times L$ diagonal matrix with entries $4\left(1+\exp(-2(\theta_k+\theta_{0,k}))\right)^{-2}$, $k = 1, 2, ..., L$. Next we focus on finding the expected value of $\left(1+\exp(-2(\theta_k+\theta_{0,k}))\right)^{-2}$:

$$\begin{aligned} & \mathbb{E}[\left(1+\exp(-2(\theta_k+\theta_{0,k}))\right)^{-2}] \\ < \; & \mathbb{E}[\left(\exp(-2(\theta_k+\theta_{0,k}))\right)^{-2}] \\ = \; & \mathbb{E}[\exp(4(\theta_k+\theta_{0,k}))]. \end{aligned} \tag{37}$$

It can be shown that θ_k is a Gaussian random variable with $\theta_k \sim \mathcal{N}\left(0, (\mathbf{W}_{H,[:,k]})^T\mathbf{P}\mathbf{W}_{H,[:,k]}\right)$, where $\mathbf{W}_{H,[:,k]}$ denotes the k^{th} column of matrix $\mathbf{W}_H$. Consequently, $\exp(4(\theta_k + \theta_{0,k}))$ is a log-normal random variable. Recall the mean of a log-normal random variable $Y \sim \text{Lognormal}(\mu, \sigma^2)$ is $\exp(\mu + \sigma^2/2)$. Consequently, the mean of $\exp(4(\theta_k + \theta_{0,k}))$ is found to be $\exp(4\theta_{0,k} + 8(\mathbf{W}_{H,[:,k]})^T\mathbf{P}\mathbf{W}_{H,[:,k]})$. To aggregate the result for all entries, after some matrix manipulation, we have

$$\mathbb{E}\left[\partial\left(\boldsymbol{\xi}(\boldsymbol{\theta})/4\right)/\partial\boldsymbol{\theta}\right] < \exp(8\mathbf{W}_H^T\mathbf{P}\mathbf{W}_H \odot \mathbf{I}_L + 4\boldsymbol{\theta}_0^D), \tag{38}$$

where $\odot$ denotes the Hadamard product, $\mathbf{I}_L$ is the $L \times L$ identity matrix, $\boldsymbol{\theta}_0^D$ is the diagonal matrix whose diagonal elements are the elements of the vector $\boldsymbol{\theta}_0$, and finally $\exp(\cdot)$ denotes the exponential of a diagonal matrix. Accounting for the coefficient 4 left out earlier, we let $\boldsymbol{\Theta} = 8\mathbf{W}_H^T\mathbf{P}\mathbf{W}_H \odot \mathbf{I}_L + 4\boldsymbol{\theta}_0^D$, plug everything into Eq. (23) and then obtain Eq. (24). ■

References

[1] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H. T. Lin. *Learning from Data – A Short Course*. AMLbook.com, 2012.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, Aug. 2002.

[3] Y. Bar-Shalom, P. K. Willett, and X. Tian. *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, Storrs, CT, 2011.

[4] K. Brigham, B. V. K. Vijaya Kumar, and N. S. V. Rao. Learning-based approaches to nonlinear multisensor fusion in target tracking. In *Proc. 16th International Conference on Information Fusion (FUSION)*, pages 1320–1327, Istanbul, Turkey, Jul. 2013.

[5] T. Cacoullos and V. Papathanasiou. On upper bounds for the variance of functions of random variables. *Statistics and Probability Letters*, 3(4):175–184, Jul. 1992.

[6] A. Chiuso and L. Schenato. Information fusion strategies and performance bounds in packet-drop networks. *Automatica*, 47:1304–1316, Jul. 2011.

[7] F. Chowdhury. A neural approach to data fusion. In *Proc. 14th American Control Conference (ACC)*, pages 1693–1697, Seattle, WA, Jun. 1995.

[8] G. R. Curry. Radar system performance modeling, 2005.

[9] E. Filer and J. Hartt. Cobra dane wideband pulse compression system. In *Proc. IEEE EASCON*, pages 26–29, Washington, DC, Sep. 1976.

[10] L.-W. Fong and C.-Y. Fan. Multisensor fusion algorithms for maneuvering target tracking. In *Proc. 1st IEEE International Conference on E-Learning in Industrial Electronics*, pages 80–84, Hammamet, Tunisia, Dec. 2006.

[11] F. D. Foresee and M. T. Hagan. Gauss-Newton approximation to Bayesian learning. In *Proc. 4th International Conference on Neural Networks*, pages 1930–1935, Houston, TX, Jun. 1997.
[12] V. Gupta, B. Hassibi, and R. M. Murray. On sensor fusion in the presence of packet-dropping communication channels. In *Proc. 44th IEEE Conference on Decision and Control (CDC)*, pages 3547–3552, Seville, Spain, Dec. 2005.
[13] M. T. Hagan and M.-B. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, Nov. 1994.
[14] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, Upper Saddle River, NJ, 2008.
[15] J. Heaton. *Introduction to Neural Networks with Java*. Heaton Research, Inc., Chesterfield, MO, 2008.
[16] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial neural networks: a tutorial. *Computer*, 29(3):31–44, Mar. 1996.
[17] T. H. Kerr. Streamlining measurement iteration for EKF target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 27(2):408–421, Mar. 1991.
[18] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. Part II: Motion models of ballistic and space targets. *IEEE Transactions on Aerospace and Electronic Systems*, 46(1):96–119, Jan. 2010.
[19] Q. Liu, N. S. V. Rao, and X. Wang. Staggered scheduling of sensor estimation and fusion for tracking over long-haul links. *IEEE Sensors Journal*, 16(15):6130–6141, Aug. 2016.
[20] Q. Liu, X. Wang, and N. S. V. Rao. Artificial neural networks for estimation and fusion in long-haul sensor networks. In *Proc. 18th International Conference on Information Fusion (FUSION)*, pages 460–467, Washington, DC, Jul. 2015.
[21] Q. Liu, X. Wang, and N. S. V. Rao. Fusion of state estimates over long-haul sensor networks with random loss and delay. *IEEE/ACM Transactions on Networking*, 23(2):644–656, Apr. 2015.
[22] Q. Liu, X. Wang, N. S. V. Rao, K. Brigham, and B. V. K. Vijaya Kumar. Effect of retransmission and retrodiction on estimation and fusion in long-haul sensor networks. *IEEE/ACM Transactions on Networking*, 24(1):449–461, Feb. 2016.
[23] D. J. Mackay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–492, May 1992.
[24] M. Mallick and K. Zhang. Optimal multiple-lag out-of-sequence measurement algorithm based on generalized smoothing framework. In *Proc. SPIE, Signal and Data Processing of Small Targets*, San Diego, CA, Apr. 2005.
[25] J. Poland. On the robustness of update strategies for the Bayesian hyper-parameter α. http://www-alg.ist.hokudai.ac.jp/~jan/alpha.pdf. Accessed on Jan. 11, 2017.
[26] N. S. V. Rao, K. Brigham, B. V. K. Vijaya Kumar, Q. Liu, and X. Wang. Effects of computing and communications on state fusion over long-haul sensor networks. In *Proc. 15th International Conference on Information Fusion (FUSION)*, pages 1570–1577, Singapore, Jul. 2012.
[27] D. Roddy. *Satellite Communications*. McGraw-Hill, New York, NY, 2006.
[28] E. I. Silva and M. A. Solis. An alternative look at the constant-gain Kalman filter for state estimation over erasure channels. *IEEE Transactions on Automatic Control*, 58(12):3259–3265, Dec. 2013.
[29] S. Sun, L. Xie, W. Xiao, and Y. C. Soh. Optimal linear estimation for systems with multiple packet dropouts. *Automatica*, 44:1333–1342, May 2008.
[30] P. K. Varshney. *Distributed Detection and Data Fusion*. Springer-Verlag, New York, NY, 1997.
[31] Y. Wang and X. R. Li. Distributed estimation fusion with unavailable cross-correlation. *IEEE Transactions on Aerospace and Electronic Systems*, 48(1):259–278, Jan. 2012.
[32] Z. Xu, X. Chang, F. Xu, and H. Zhang. $L_{1/2}$ regularization: A thresholding representation theory and a fast solver. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1013–1027, Jul. 2012.
[33] M. Yeddanapudi, Y. Bar-Shalom, K. R. Pattipati, and S. Deb. Ballistic missile track initiation from satellite observations. *IEEE Transactions on Aerospace and Electronic Systems*, 31(3):1054–1071, Jul. 1995.
[34] H. Yu and B. M. Wilamowski. Levenberg–Marquardt training. In *Industrial Electronics Handbook*, volume 5, chapter 12, pages 1–15. CRC Press, Boca Raton, FL, 2011.
[35] K. Zhang, X. R. Li, and Y. Zhu. Optimal update with out-of-sequence measurements. *IEEE Transactions on Signal Processing*, 53(6):1992–2004, Jun. 2005.
[36] S. Zhang and Y. Bar-Shalom. Optimal update with multiple out-of-sequence measurements with arbitrary arriving order. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):3116–3132, Oct. 2012.
[37] Z. Zhao, X. R. Li, and V. P. Jilkov. Best linear unbiased filtering with nonlinear measurements for target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 40(4):1324–1336, Oct. 2004.

Qiang Liu is currently a Research Associate in the Computer Science and Mathematics Division at Oak Ridge National Laboratory in Oak Ridge, TN. He received his Ph.D. in Electrical and Computer Engineering from Stony Brook University, Stony Brook, NY, in 2014. His research areas include networked data and information fusion, signal/target detection and estimation, statistical signal processing, wireless sensor networks, cognitive radios, cyber-physical systems, transport protocols, high performance computing, and software defined networking. Dr. Liu has published over 20 peer-reviewed papers in prestigious conference proceedings and journals, and has reviewed dozens of works by others. He has been invited to present his work at various venues and is the recipient of several best paper and travel grant awards.

Katharine Brigham received a B.E. degree in Electrical Engineering and Mathematics from Vanderbilt University, Nashville, TN, in 2003, and an M. Eng. in Systems Engineering from Cornell University, Ithaca, NY, in 2008. She later obtained her Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2015, and started work for Voci Technologies as a Senior Engineer in Pittsburgh, PA. There, she worked on several government grants involving automatic language recognition, speaker recognition, and dereverberation in speech. In 2016, Dr. Brigham joined the School of Engineering and Computing Sciences at Durham University as a Teaching Fellow in Electrical Engineering. Her research interests include machine learning/pattern recognition, signal and image processing, and speech recognition.

Nageswara S. V. Rao received his B.Tech from National Institute of Technology, Warangal, India, in Electronics and Communications Engineering in 1982, M.E. in Computer Science and Automation from Indian Institute of Science, Bangalore, India, in 1984, and Ph.D. in Computer Science from Louisiana State University, Baton Rouge, LA, in 1988. He is currently a Corporate Fellow in the Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, where he joined in 1993. He was a Technical Director of C2BMC Knowledge Center at Missile Defense Agency from 2008 to 2010. He has published more than 350 technical conference and journal papers in the areas of sensor networks, information fusion, and high-performance networking. He is a Fellow of IEEE, and received 2005 IEEE Technical Achievement Award and 2014 R&D 100 Award.