18th International Conference on Information Fusion
Washington, DC - July 6-9, 2015

Artificial Neural Networks for Estimation and Fusion in Long-Haul Sensor Networks

Qiang Liu
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794–2350
Email: qiangliu@ece.sunysb.edu

Xin Wang
Department of Electrical and Computer Engineering
Stony Brook University
Stony Brook, NY 11794–2350
Email: xwang@ece.sunysb.edu

Nageswara S. V. Rao
Oak Ridge National Laboratory
Oak Ridge, TN 37831–6355
Email: raons@ornl.gov

***Abstract*—We consider long-haul sensor networks where 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 to improve the accuracy of the final estimates of certain target characteristics. In this work, we pursue artificial neural network (ANN) learning-based approaches for estimation and fusion of target states in long-haul sensor networks. The joint effect of (1) imperfect communication condition, namely, link-level loss and delay, and (2) 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, backpropagation, error regularization, root-mean-square-error (RMSE) performance, reporting deadline.**

I. Introduction

Sensor networks can be found in many real-world applications, such as security, healthcare, and environmental monitoring, among others [1]. In one subclass of networks, namely, the long-haul sensor networks, 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, and are tasked to perform target tracking and monitoring. 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 serves to collect data from multiple such sensors and fuse these 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.

Many challenges exist in estimation and fusion applications over such long-haul networks. Due to the long distances (e.g., tens of thousands of miles for satellite links), the propagation time can be significant. In addition, communication is often characterized by sporadic high bit-error rates (BERs) and burst losses that can effectively reduce the number of reliable estimates available at the fusion center. As a result, 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, some sensors may also be prone to estimation bias, as a result of poor calibration or environmental factors.

Existing studies have attempted to address estimation and/or fusion under variable communication loss and/or delay conditions. In [4], [18], estimation and fusion performance using Kalman filters (KFs) under variable packet loss rates have been considered. Studies such as [17], [21] have addressed filtering in the context of out-of-sequence measurements (OOSMs), where all data would finally arrive despite the random delay. [15] has exploited retransmission to recover some of the lost messages over time so that the effect of information loss can be somewhat mitigated. A staggered estimation scheduling scheme is proposed in [12] that aims to explore the temporal domain relationships of adjacent data within an estimation interval to improve the estimation and fusion performance. More recently, [13] has considered an information feedback mechanism where fused estimates are fed back to a subset of sensors in order to improve their information quality, and in turn, the overall fusion performance.

A number of data fusion methods have been developed over the years, with a primary goal of taking in the data from multiple sensors and combining them to produce a condensed set of meaningful information with the highest possible degree of accuracy and certainty [2]. 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. 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 [8]. More recently, [3] has proposed learning-based nonlinear fusion including ANNs. This study of ANN-based fusers

accounts for both communication and computation constraints and aims to yield improved long-haul target tracking and monitoring performance under these constraints. In particular, besides exploring 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 perform such tasks effectively with (1) very limited training data; (2) lost data in both training and testing stages; and (3) sensor biases. Of concern is the performance of generalization capabilities from training to testing data under variable constraints.

The remainder of this paper is organized as follows: We first overview the fundamentals of ANNs and the learning algorithm called backpropagation in Sec. II. In Sec. III, the enhanced variations of a standard backpropagation solution are explored that could potentially improve the generalization capabilities of the training-based fuser to new data. The effects of missing data and estimation bias are studied in Secs. IV and V respectively. Simulation results of a ballistic target tracking application using different fusers are presented and analyzed in Sec. VI before we conclude the paper in Sec. VII.

II. Artificial Neural Networks (ANNs) and Backpropagation

We are primarily interested in using nonlinear functions to fuse sensor data that may potentially yield better results than with closed-form 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.

A. Structure of an ANN

Broadly speaking, ANNs are a class of statistical models that consist of sets of adaptive weights (i.e., 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 simple 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 such a three-layer neural network is shown in Fig. 1. This network consists of an input layer, a hidden layer, and an output layer, interconnected by weights (to be determined) which are represented by the arrows between the layers. Apparently, information moves forward in one direction, from the input nodes, through the hidden nodes, and to the output nodes.

In our settings, the inputs $\hat{x}^{(1)}$, ..., $\hat{x}^{(N_i)}$ are 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 (not shown in the figure) that is connected to each node in addition to the input nodes. The output of the j^{th} hidden node, a_j, is

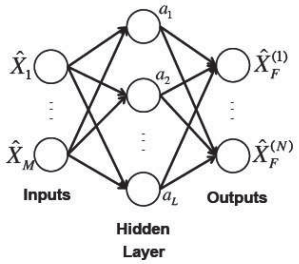


Fig. 1: An example of a three-layer feedforward neural network

given by

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

where the parameters w_{ij} and b_j are typically called the weights and biases, respectively. $\hat{x}^{(i)}$ is an input feature (e.g., a state estimate from a sensor), and $g_1(\cdot)$ is a nondecreasing function called the activation function, which is typically a bounded function such as the sigmoid function. If we concatenate all of the hidden node outputs a_j into a vector $\mathbf{a} = [a_1, ..., a_L]^T$, then we can write the hidden node outputs as

$$\mathbf{a} = g_1(\mathbf{W}_H^T\hat{\mathbf{x}} + \mathbf{b}_H), \tag{2}$$

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

$$\hat{\mathbf{x}}_F = g_2(\mathbf{W}_o^T\mathbf{a} + \mathbf{b}_o), \tag{3}$$

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.

B. Backpropagation

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 are initialized with random values and are 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

$$\begin{aligned}\mathbf{w} = [&\mathbf{W}_H(1,1), \mathbf{W}_H(1,2), ..., \mathbf{W}_H(L,N_i),\\ &\mathbf{b}_H(1), ..., \mathbf{b}_H(L), \mathbf{W}_o(1,1), ..., \mathbf{b}_o(N_o)]^T.\end{aligned} \tag{4}$$

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 this weight vector, then the Gauss-Newton method for updating $\mathbf{w}$, an iterative method that uses the first and second derivatives of a function to find a point where the derivative is zero, would be

$$\Delta\mathbf{w} = -[\nabla^2 S(\mathbf{w})]^{-1}\nabla S(\mathbf{w}), \tag{5}$$

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, e.g.,

$$\begin{aligned} S(\mathbf{w}) &= \sum_{k=1}^{m}(y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2 \\ &= \sum_{k=1}^{m}(e_k(\mathbf{w}))^2 = \mathbf{e}(\mathbf{w})^T\mathbf{e}(\mathbf{w}), \end{aligned} \tag{6}$$

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 $\mathbf{e}(\mathbf{w})$ and rewrite Eq. (5) as

$$\Delta\mathbf{w} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}), \tag{7}$$

where

$$\mathbf{J} = \begin{bmatrix} \frac{\partial e_1(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_1(\mathbf{w})}{\partial w_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_m(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial e_m(\mathbf{w})}{\partial w_d} \end{bmatrix}. \tag{8}$$

The Levenberg-Marquardt (LM) modification [7] to the Gauss-Newton method is

$$\Delta\mathbf{w} = -(\mathbf{J}^T\mathbf{J} + \eta\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}(\mathbf{w}) \tag{9}$$

where $\mathbf{I}$ is the identity matrix, and $\eta > 0$ is a damping factor, which is adjusted at each iteration of the weight update. If a step (i.e., weight update) 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 [7]. 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 performance.

III. 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 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 an 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. (6). An additional term is added to the original performance function, usually in the form of a cost or penalty function for complexity.

In the Bayesian framework proposed by MacKay [16], 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. In the meantime, the effective number of parameters actually used by the model – in this case, the number of network weights – is also sought to be reduced. Hence, MacKay's Bayesian regularization expands the original sum-of-squares cost function to search for the minimum error using minimum weights 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

$$\begin{aligned} S(\mathbf{w}) &= \frac{\beta}{2}\sum_{k=1}^{m}(y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2 + \frac{\alpha}{2}\sum_{k=1}^{d} w_i^2 \\ &= \beta \cdot SSE + \alpha \cdot SSW. \end{aligned} \tag{10}$$

It can be seen that an additional term, namely, the sum of squared weights (SSW), is added to the objective function. The SSW is simply the square of the $\mathbf{L}^2$ norm of the d-dimensional weight vector $\mathbf{w}$ introduced in Eq. (4). The hyperparameters are updated as

$$\alpha = \frac{\gamma}{2 \cdot SSW}, \text{ and } \beta = \frac{m - \gamma}{2 \cdot SSE}, \tag{11}$$

where

$$\gamma = d - \alpha \text{ tr}((\mathbf{J}^T\mathbf{J})^{-1}) \tag{12}$$

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 can be easily incorporated into each iteration of the LM weight update equation as in Eq. (9).

C. Regularization Using Error Covariance Matrices

Since the sensors also provide additional information regarding the state estimates, i.e., the error covariances, we are interested in incorporating these error covariance estimates to train the ANNs and improve the neural network's generalization capability.

In particular, a quadratic term containing the error covariance term is incorporated into the objective term, along with a tradeoff parameter λ [3]:

$$S(\mathbf{w}) = \sum_{k=1}^{m} (y^{(k)} - f(\hat{\mathbf{x}}^{(k)}, \mathbf{w}))^2 + \lambda \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} \tag{13}$$

where

$$\mathbf{\Sigma} = \left[\begin{array}{c|c} \mathbf{P}_R & \mathbf{0}_{NLs \times (d-NL)s} \\ \hline \mathbf{0}_{(d-NL)s \times NLs} & \mathbf{0}_{(d-NL)s \times (d-NL)s} \end{array} \right]. \tag{14}$$

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

$$\mathbf{P}_R = \begin{bmatrix} \mathbf{P} & \mathbf{0}_{Ns \times Ns} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P} \end{bmatrix}, \tag{15}$$

and $\mathbf{P}$ is a block diagonal matrix where the blocks are $\mathbf{P}_1$ through $\mathbf{P}_N$:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1 & \mathbf{0}_{s \times s} & \cdots & \mathbf{0}_{s \times s} \\ \mathbf{0}_{s \times s} & \mathbf{P}_2 & \cdots & \mathbf{0}_{s \times s} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{s \times s} & \cdots & \mathbf{0}_{s \times s} & \mathbf{P}_N \end{bmatrix}, \tag{16}$$

where the state estimates from the sensors are assumed to be equal to the true states plus noise, i.e., $\hat{\mathbf{x}}_i = \mathbf{x} + \mathbf{n}_i$, in which i is the sensor index, and $\mathbf{n}_i$ is zero-mean white Gaussian noise with covariance $\mathbf{P}_i$; in other words, $\mathbf{n} = [\mathbf{n}_1^T, ..., \mathbf{n}_N^T]^T$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$. 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 $Ns \times Ns$.

The weight update equation can be similarly 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 weight update equation which incorporates the error covariance estimates into the training using the LM method:

$$\Delta \mathbf{w} = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{\Sigma} + \eta\mathbf{I})^{-1}(\mathbf{J}^T\mathbf{e}(\mathbf{w}) + \lambda\mathbf{\Sigma}\mathbf{w}). \tag{17}$$

IV. Training and Testing with Missing Data

Having provided the training algorithm and techniques for generalization, we now focus on the communication and computation constraints and their impact on the learning process. First, the communication link loss and delay further 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. (6), based on the actual pattern availability, we have

$$S(\mathbf{w}) = \sum_{k=1}^{\tilde{m}} (e_k(\mathbf{w}))^2 = \tilde{\mathbf{e}}(\mathbf{w})^T \tilde{\mathbf{e}}(\mathbf{w}), \tag{18}$$

where $\tilde{\mathbf{e}}$ is the error vector containing $\tilde{m}$ elements. 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. (6) can now be expressed as

$$S(\mathbf{w}) = \sum_{k=1}^{m} i_k (e_k(\mathbf{w}))^2 = \mathbf{e}(\mathbf{w})^T \mathbf{\Lambda} \mathbf{e}(\mathbf{w}), \tag{19}$$

where i_k is the indicator function in which $i_k = 1$ 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 as 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 [14] to manually fill in the missing inputs. 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 margin.

V. Sensor Bias and Error Regularization

Another concern arising out of 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 about estimation biases 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 for its training.

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 $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_u + \mathbf{b}_i$ where $\hat{\mathbf{x}}_u$ is an unbiased estimate for $\mathbf{x}$ and $\mathbf{b}_i$ is the bias vector, then we have

$$\begin{aligned}\mathbf{P}_i &= \mathbb{E}[(\hat{\mathbf{x}}_i - \mathbf{x})(\hat{\mathbf{x}}_i - \mathbf{x})^T] \\ &= \mathbb{E}[(\hat{\mathbf{x}}_u - \mathbf{x} + \mathbf{b}_i)(\hat{\mathbf{x}}_u - \mathbf{x} + \mathbf{b}_i)^T] \\ &= \mathbb{E}[(\hat{\mathbf{x}}_u - \mathbf{x})(\hat{\mathbf{x}}_u - \mathbf{x})^T] + \mathbb{E}[\mathbf{b}_i\mathbf{b}_i^T] + 2\mathbb{E}[(\hat{\mathbf{x}}_u - \mathbf{x})\mathbf{b}_i^T] \\ &= \mathbf{P}_u + \mathbb{E}[\mathbf{b}_i\mathbf{b}_i^T] + 2\mathbb{E}[(\hat{\mathbf{x}}_u - \mathbf{x})\mathbf{b}_i^T] \\ &\approx \mathbf{P}_u + \overline{\mathbf{b}_i\mathbf{b}_i^T}, \end{aligned} \tag{20}$$

where $\mathbf{P}_u$ is the error covariance matrix for an unbiased estimate. In the extreme case when 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, the first one being that the bias is largely uncorrelated with the true target state, and the second being a running average (as denoted by the overline) 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. (20) 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 the online tracking. The performance comparison using different error regularization methods will be shown in the next section.

VI. Performance Evaluation

In this section, the performance of the ANN-based fusers is evaluated for a coasting ballistic target tracking application via simulations. Different configurations during the learning process are considered, so are 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 to be outlined below.

A. Simulation Models

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 [11]).

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 [11], 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 $\mu = 3.986012 \times 10^5$ km^3/s^2 is the Earth's gravitational constant. The continuous-time model of the system is given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -\mu x/r^3 \\ -\mu y/r^3 \\ -\mu z/r^3 \end{bmatrix} \tag{21}$$

where $r = \sqrt{x^2+y^2+z^2}$. An algorithm for computing the state propagation can be found in [20].

2) Sensor Measurement Model: We simulate the measurements following the simulation in [10] for a ballistic coast target. The measurement model is given by $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w}$, and the measurements of the range (r), elevation (E), and azimuth (A) of the target are computed as follows:

$$\mathbf{z} = \begin{bmatrix} r \\ E \\ A \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \tan^{-1}\left(z/\sqrt{x^2+y^2}\right) \\ \tan^{-1}(x/y) \end{bmatrix} + \mathbf{w}, \tag{22}$$

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

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. From [5], we have $\sigma_r, \sigma_E, \sigma_A \propto \frac{1}{\sqrt{SNR}}$, where the signal-to-noise ratio SNR is inversely proportional to $\tilde{r}^4$ ($\tilde{r}$ is the range from the sensor to the target). 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 [6] 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. Zhao et. al. [22] developed a recursive BLUE filter for a linear system that is 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.

4) Communication Loss and Delay Profiles: The message-level loss and delay characteristics are determined by the long-haul link conditions. Suppose each message sent by a sensor is lost during transmission with probability p_L, whereas a probability density function $f(t)$ models the overall delay t that any message experiences to be successfully delivered to the fusion center, and the shifted exponential distribution [15] $f(t) = \exp\left((T-t)/\mu_D\right)/\mu_D$ 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.

5) Closed-Form Fusers: For performance comparison, we consider two closed-form (i.e., non-learning-based) fusers. The track-to-track fuser (T2TF) [2] is a fuser theoretically 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:

$$\mathbf{P}_F = (\mathbf{P}_1^{-1} + \mathbf{P}_2^{-1})^{-1}, \tag{23}$$

$$\hat{\mathbf{x}}_F = \mathbf{P}_F(\mathbf{P}_1^{-1}\hat{\mathbf{x}}_1 + \mathbf{P}_2^{-1}\hat{\mathbf{x}}_2), \tag{24}$$

where $\hat{\mathbf{x}}_i$ and $\mathbf{P}_i$ are the state estimates and error covariance from sensor i, respectively, The error cross-covariance $\mathbf{P}_{ij} = \mathbf{P}_{ji}^T$, the error cross-covariance between sensors i and j, has been omitted from the equations since it is often unknown. This rule can be readily extended to multiple sensors.

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:

$$\mathbf{P}_F = (\omega_1\mathbf{P}_1^{-1} + \omega_2\mathbf{P}_2^{-1})^{-1} \tag{25}$$

$$\hat{\mathbf{x}}_F = \mathbf{P}_F\left(\omega_1\mathbf{P}_1^{-1}\hat{\mathbf{x}}_1 + \omega_2\mathbf{P}_2^{-1}\hat{\mathbf{x}}_2\right), \quad \omega_1 + \omega_2 = 1 \tag{26}$$

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 is proposed in [19] 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:

$$\omega_1 = \frac{D(p_1, p_2)}{D(p_1, p_2) + D(p_2, p_1)} \tag{27}$$

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:

$$D(p_i, p_j) = \frac{1}{2}\left[\ln\frac{|\mathbf{P}_j|}{|\mathbf{P}_i|} + \mathbf{d}_X^T\mathbf{P}_j^{-1}\mathbf{d}_X + Tr(\mathbf{P}_i\mathbf{P}_j^{-1}) - k\right] \tag{28}$$

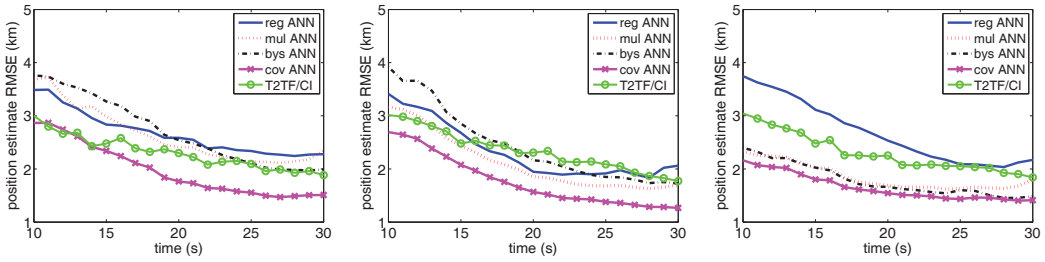
where $\mathbf{d}_X = \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j$, k is the dimensionality of $\hat{\mathbf{x}}_i$, and $|\cdot|$ denotes the determinant. This fast-CI fuser can also be extended to more than two sensors, although the equations are somewhat more involved.

6) 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):

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

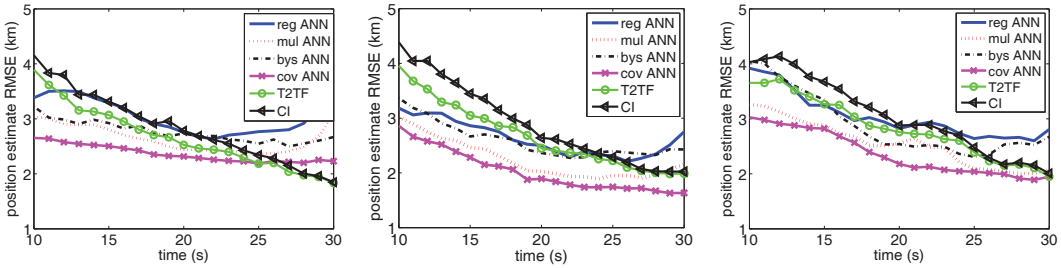
with a standard deviation of 500 m and 5 m/s for each position and velocity component, respectively. In our default setting, two target trajectories are available for training. Measurement data are generated according to the sensor measurement model introduced earlier, and state estimates (and the associated 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.

7) 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 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 [9] 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.



(a) (b) (c)

Fig. 2: Position estimate RMSE over time with ANN-based fusers as well as T2TF and fast-CI fuser with p_L = 0, D_F = 2 s: (a) two training trajectories, three hidden nodes; (b) five training trajectories, three hidden nodes; (c) two training trajectories, six hidden nodes



(a) (b) (c)

Fig. 3: Position estimate RMSE over time with ANN-based fusers as well as T2TF and fast-CI fuser with p_L = 0.5, D_F = 1 s: (a) two training trajectories, three hidden nodes; (b) five training trajectories, three hidden nodes; (c) two training trajectories, six hidden nodes

B. Fusion Performance without Sensor Biases

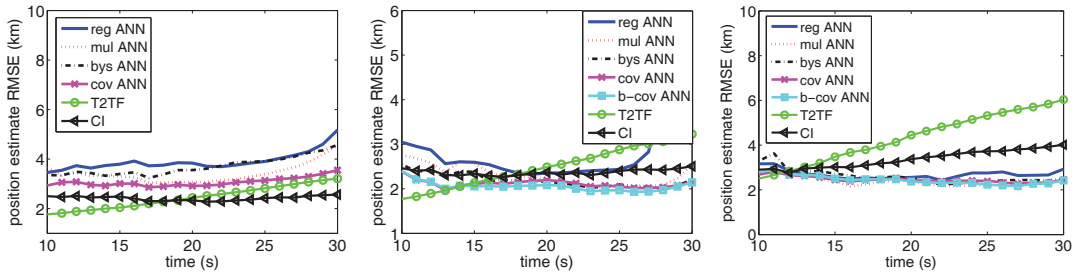
In Fig. 2(a), tracking performances using variations of the ANN-based fuser, including the regular ANN ("reg-ANN"), multiple ANN with 5 component ANNs of the same structure ("mul-ANN"), ANN with Bayesian regularization ("bys-ANN"), and ANN with error covariance regularization ("cov-ANN"), along with that of the T2TF/CI fuser are plotted. We observe that the original ANN fuser doesn't yield estimates as good as those from the T2TF/CI by using only two available training trajectories. 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 set. In general, the multiple ANN approach would outperform the original ANN-based fuser by variable levels, depending on such factors as how the weights are initialized in each component network. Most notably, 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.

In Figs. 2(b) and 2(c), the performance plots with more training data (five trajectories) and more hidden nodes (six) are shown respectively. From these plots, the error curves for ANN-based fusers are variably lower compared to those in Fig. 2(a), reflecting the major benefit of training more data and/or using a somewhat more complex neural network. But regardless of the chosen data or network size, a consistent observation is the superiority of the covariance-based ANN fuser over others in terms of tracking accuracy.

As the link-level communication loss and/or delay increases, so does the chance that the fusion center cannot receive both sensor estimates for any time epoch during the testing stage. With incomplete sensor data, the fusion center could use predicted estimates it derives for the individual sensors to interpolate the missing training and testing sensor data so that the completeness of the ANN inputs can be guaranteed. Fig. 3(a) shows the performance with 50% sensor data loss in both training and testing data. Whereas a fusion deadline of D_F = 2 s can effectively reduce any effect of early cutoff, here the deadline D_F = 1 s can result in more missing original sensor data by the deadline. Comparing different cases, there is an increase in the tracking error in non-learning based T2TF and CI fusers compared to their respective no-loss counterparts. However, the ANN learning-based approaches are largely able to retain their tracking performance under the lossless situation, thereby demonstrating the major benefit of adopting these fusers in counteracting the negative effect of communication constraints. Similar observations can be made from cases with more training data and more hidden nodes.

C. Fusion Performance with Sensor Biases

Starting with the baseline case as shown in Fig. 3(a), we examine the cases where the training and/or testing data contain biased estimates. First, suppose in the test data, a positive uniform bias, with its mean magnitude set at 0.1% of the noise level, is added to the (unbiased) measurement noise at one of the sensors. The fusion performance is shown in Fig. 4(a). Due to the mismatch between the training and testing data in bias profiles, the unlearned bias (i.e., bias absent from the training data) reduces the generalization capability of the trained parameters, resulting in elevated estimation



(a) (b) (c)

Fig. 4: Position estimate RMSE over time with ANN-based fusers as well as T2TF and fast-CI fuser with $p_L = 0.5$, $D_F = 1$ s and a bias of (a) 0.1% of noise in one testing data; (b) 0.1% of noise in training and testing data; (c) 0.2% of noise in training and testing data

errors across all learning-based cases. 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. 4(b), where performance of the "cov ANN" under the bias correction method introduced earlier is plotted as well ("b-cov ANN"). With an additional bias correction step, the covariance regularization method is able to output even better estimates, although this improvement seems somewhat limited. Finally, all others kept equal, the mean bias level is now increased to 0.2% of the unbiased measurement noise. From the plots in Fig. 4(c), whereas the T2TF and the CI fuser output increasingly error-prone estimates, the ANN-based fusers are able to retain much of their error levels from the previous case, thereby demonstrating their advantages in tracking with biased sensor data. Nevertheless, with higher link loss rates, any improvement in tracking accuracy over the non-learning fusers can be reduced, as a result of the increased mismatch level (from extended prediction over biased estimates) between the training and testing data.

VII. Conclusion

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. Extensions of this work may be studied in a setting with a much larger training dataset but with heterogenous trajectories and/or variable data quality levels, so that prior to training, a classification process needs to be carried out in order to increase the potential match between the training and testing datasets and improve the generalizability of the learned patterns.

References

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

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

[3] 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 2013)*, pages 1320–1327, Jul. 2013.

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

[5] G. R. Curry. *Radar System Performance Modeling*. Artech House Publishers, 2004.

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

[7] M. T. Hagan and M.-B. Menhaj. Training feedforward networks with the marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993, Nov. 1994.

[8] S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, 2008.

[9] J. Heaton. The number of hidden layers, 2008. http://www.heatonresearch.com/node/707.

[10] T. H. Kerr. Streamlining measurement iteration for ekf target tracking. *IEEE Transactions on Aerospace and Electronic Systems,*, 27(2):408–421, Mar. 1991.

[11] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. part ii: Motion models of ballistic and space targets. *Aerospace and Electronic Systems, IEEE Transactions on*, 46(1):96–119, Jan. 2010.

[12] Q. Liu, X. Wang, and N. S. V. Rao. Staggered scheduling of estimation and fusion in long-haul sensor networks. In *Proc. 16th International Conference on Information Fusion (FUSION 2013)*, pages 1699–1706, Istanbul, Turkey, Jul. 2013.

[13] Q. Liu, X. Wang, and N. S. V. Rao. Information feedback for estimation and fusion in long-haul sensor networks. In *Proc. Information Fusion (FUSION), 2014 17th International Conference on*, Salamanca, Spain, Jul. 2014.

[14] 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*, 2015.

[15] Q. Liu, X. Wang, N. S. V. Rao, K. Brigham, and B. V. K. Vijaya Kumar. Performance of state estimate fusion in long-haul sensor networks with message retransmission. In *Proc. Information Fusion (FUSION), 2012 15th International Conference on*, pages 719–726, Singapore, Singapore, Jul. 2012.

[16] D. J. Mackay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–492, May 1992.

[17] 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.

[18] E. I. Silva and M. A. Solis. An alternative look at the constant-gain kalman filter for state estimation over erasure channels. *Automatic Control, IEEE Transactions on*, 58(12):3259–3265, Dec. 2013.

[19] Y. Wang and X. Li. Distributed estimation fusion with unavailable cross-correlation. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(1):259–278, Jan. 2012.

[20] 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.

[21] S. Zhang and Y. Bar-Shalom. Optimal update with multiple out-of-sequence measurements with arbitrary arriving order. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(4):3116–3132, Oct. 2012.

[22] 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.