A Performance Evaluation of the Alpha-Beta (α-β) Filter Algorithm with Different Learning Models: DBN, DELM, and SVM

Junaid Khan 1 and Kyungsup Kim 1,2,*

1 Department of Environmental & IT Engineering, Chungnam National University, Daejeon 34134, Korea
2 Department of Computer Science & Engineering, Chungnam National University, Daejeon 34134, Korea
* Correspondence: sclkim@cnu.ac.kr

Abstract: In this paper, we present a new Multiple learning to prediction algorithm model that used three different combinations of machine-learning methods to improve the accuracy of the α-β filter algorithm. The parameters of α and β were tuned in dynamic conditions instead of static conditions. The proposed system was designed to use the deep belief network (DBN), the deep extreme learning machine (DELM), and the SVM as three different learning algorithms. Then these learned parameters were trained by the machine-learning algorithms tuned to the α-β filter algorithm as a prediction module, and they gave the final predicted results. The MAE and RMSE were used to evaluate the performance of the proposed α-β filter with different learning algorithms. Each algorithm recorded different best-case accuracy results; for the DBN, we achieved 3.60 and 2.61; for the DELM, we obtained the best-case result of 3.90 and 2.81; and finally, for the SVM, 4.0 and 3.21 were attained in terms of the RMSE and MAE, respectively, as compared to 5.21 and 3.95. When assessed in comparison with the typical alpha–beta filter algorithm, the proposed system provided results with better accuracy.

Keywords: alpha–beta filter; deep belief network; DELM; SVM; accuracy

1. Introduction

Tracking filters are vital in target tracking problems. By using these filters, we can tackle real estimation problems, which can also help reduce tracking errors. The α-β family of filters was developed and created during the Cold War [1]. Usually, this family of filters is utilized to track stable velocity and constant acceleration by using the Kalman filter as a continual state filter. However, the ability of these filters to accurately track high-momentum maneuvering targets is minimal, characterized by jerky motions. The alpha-beta (α-β) filter algorithm [2] is a suitable method for the observer; it is mainly used for data smoothing, control, and estimation. This algorithm is very lightweight and has the same functionality as the Kalman filter. If we compare the α-β filter’s performance with the Kalman filter [3] and the linear filter, we can see that the α-β filter performs better than these filters because it does not require a complex system model. Another exciting feature of this filter is that α-β requires less memory and computation time.

State-of-the-art progress in technology, including the deep belief network (DBN) [4], the deep extreme learning machine (ELM) [5], classification and regression trees (CARTs) [6], the support vector machine (SVM) algorithm [7], and other machine-learning advancements have improved our living standards and assist us in different ways. These methods are mainly based on our current knowledge and the data extracted from existing and past data, which enable us to make progressive decisions for the future to minimize losses and achieve the maximum benefits [2]. To enable us to benefit from these algorithms to progress as much as possible, first, we train them by using historical data; the more data an algorithm has access to, the more accurate the results are. When the training stage of
such an algorithm is finished, the algorithm is ready for use in the designed application. However, at this stage, one issue that arises is that these algorithms are designed for a particular task and setting; therefore, the performance of such algorithms degrades over time as their operational environments change. Numerous well-known algorithms are used to overcome this limitation [1], such as DELM, DBN, and SVM algorithms, and the stacked generalization [8] technique has been proposed to improve the accuracy in prediction and classification.

When this machine-learning technique [9] is attached to an algorithm, the performance may improve in different ways. We applied these three algorithms as a learning unit to the α-β filter to boost the performance of the filters. The alpha-beta filter algorithm is the most basic linear observer method, and it is utilized to evaluate problem and application control. We obtained this method from the Kalman filter. The Kalman filter [10] is the most complex method; in comparison, it is easy to use the alpha-beta filter in different applications. It also displays a better performance than other filters.

In this paper, new ML-based algorithms were implemented, and three different combinations of these algorithms were used to enhance the accuracy of the α-β filter algorithm and to tune the parameters of α and β in dynamic conditions. The proposed system uses the deep belief network, support vector machine, and deep extreme learning machine as three different learning algorithms. Then these learned parameters are trained by the machine-learning algorithms, tuned to the α-β filter algorithm as a prediction module, and provide the final predicted results. The MAE and RMSE were used to calculate the performance of the proposed α-β filter with different learning algorithms. As evaluated by comparing the proposed algorithm with the typical alpha-beta filter algorithm, it was shown that the proposed system provides results with better accuracy. Finally, we compared the results of the DELM, DBN, and SVM. We concluded that when the DBN was attached to the alpha-beta filter, and the performance was very high compared to the other algorithms. Figure 1 shows the conceptual model of the alpha-beta filter model. Each algorithm recorded different best-case accuracy results; for the DBN, we achieved 3.60 and 2.61; for the DELM, we obtained best-case results of 3.90 and 2.81; and finally, for the SVM, 4.0 and 3.21 were attained in terms of the RMSE and MAE, respectively, as compared to 5.21 and 3.95. In summary, the main contributions of our paper are as follows:

1. The development of a deep-belief-network-based alpha-beta filter;
2. The development of an SVM-based alpha-beta filter;
3. The development of a DELM-based alpha-beta filter;
4. The performance evaluation of both conventional algorithms and the proposed algorithm;
5. A shift from the static approach to a dynamic approach.

![Figure 1. The conceptual model of the proposed learning algorithm for the alpha-beta filter model.](image-url)
The paper is organized as follows: In Section 2, related work regarding similar filters is discussed in detail. Section 3 sheds light on the proposed methodology of the different learning methods for the alpha-beta filter, including the DELM, DBN, and SVM. In Sections 4 and 5, the implementation and results are discussed briefly. Finally, Section 6 concludes the paper. The abbreviations used in this paper are defined in Table 1.

Table 1. Notations with description.

<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN</td>
<td>Deep belief network</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial neural network</td>
</tr>
<tr>
<td>DELM</td>
<td>Deep extreme learning machine</td>
</tr>
<tr>
<td>α-β</td>
<td>Alpha-beta</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root mean square error</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean absolute error</td>
</tr>
<tr>
<td>KF</td>
<td>Kalman filter</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional Integrative Derivative</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer perceptron</td>
</tr>
<tr>
<td>BP</td>
<td>Backpropagation</td>
</tr>
<tr>
<td>HL</td>
<td>Hidden layer</td>
</tr>
<tr>
<td>AF</td>
<td>Activation Function</td>
</tr>
</tbody>
</table>

2. Related Work

We carried out a broad review of the literature on several renowned performance evaluation and prediction techniques. In recent years, researchers have proposed several techniques to develop and enhance filtering mathematics, and these enhanced models are used in different practical application areas.

Discrete data are commonly applied to predict the kinematics of moving objects. These data are related to air traffic control, antisubmarine warfare, missile interceptions, and similar distinguished applications. Methods have been specially designed for radar tracing to estimate velocity and positioning based on noisy data. Some of the filters from this family have been used for applications other than enabling accurate tracking and prediction, comparable to the Kalman filter [11–15]. The author of Reference [16] used the filters to make predictions and track rate fluctuations. Tracking parameters were also a topic of interest for the author of Reference [17]. The author of Reference [18] aimed to improve the efficiency of the filter family and introduced a new method to improve their performance. Their contribution also shed light on how balanced α-β filters can be developed in terms of performance.

Moreover, α-β-γ filters have been applied in computer vision for different applications. The author of Reference [19] compared the α-β-γ filter’s performance with that of the Kalman filter. They observed that the Kalman filter converged coefficients to approximately constant levels, and for this reason, the computation of the filter was shown to be ineffective. Due to this inefficiency, the α-β filter gave a good performance, and these promising results show that, compared with the Kalman filter, this filter has less computational time requirements. The author of Reference [20] also implemented similar filters to forecast target locations within an image plane. The target’s position was captured in every iteration to predict one-step-forward locations by putting the data into an α-β-γ filter. By testing and using this method, the tracking performance was shown to be effective and improved. The author of Reference [21] compared the α-β-γ filter and the Kalman filter in terms of their performance. They refined the filter parameter values to prove that the α-β-γ filter performs better than the Kalman filter. The author of Reference [22] proposed the cascaded Proportional Integrative Derivative (PID) law to control motor positions, using an α-β-γ filter. This method using α-β filters has a new design and procedure and obtains more precise results.
Recently, the author of Reference [23] proposed a genetic-based algorithm to determine the best parameter values of the $\alpha$-$\beta$-$\gamma$ filter. The noise in the filter was at acceptable levels, and they also achieved a better improvement in performance.

The author of Reference [24] proposed a perfect model to track accuracy enhancement, using the $\alpha$-$\beta$-$\gamma$-$\delta$ filter; this method is named “the third order filter” (in which four parameters are used, i.e., the $\alpha$-$\beta$-$\gamma$-$\delta$ filter). The third temporal derivative of interest value is the added value (called “jerk”). The filter used for tracking can also forecast the second-order derivative value of interest. In conclusion, by using these second- and third-order derivatives, the author claimed that the tracking filter accuracy was notably improved. In Reference [14], a feed-forward backpropagation neural network model with the function tan-sigmoid and linear function was developed to forecast the consumption of energy in smart houses. In Reference [21], another method was proposed, using a related type of predictive model and multilayer perceptron (MLP) for short-term energy usage. The Levenberg–Marquardt backpropagation algorithms and a scaled conjugate gradient was used. In Reference [22], an ANN energy prediction technique for smart homes was proposed to forecast energy consumption at different time periods (hour, day, week, and year). This strong NN and pro-energy system assists in forecasting and assembling the energy capacity. The Taguchi technique has been used to estimate the influence of data on the energy capacity [23]. The author of Reference [24] proposed another effective mixture method utilized by an auto-regressive integrated moving (ARIM) average for energy prediction. The author of Reference [25] used composite ANN and PSO algorithms for the optimization of the energy consumption of electric apparatus. This method is based on the IoT management of home energy systems (HEMs) for smart homes. The HEM algorithm presented in Reference [26] is also a helpful method for use in the optimization energy consumption and prediction.

The author of Reference [2] developed a method based on the DELM to improve the accuracy of the $\alpha$-$\beta$ filter algorithm. For instance, the author of Reference [25] used the bat algorithm and the alpha-beta filter algorithm to determine parameter preferences and optimize the consumption of energy in smart homes, and the deep ELM was also used. The author of Reference [26] proposed a method named the adaptive alpha-beta filter and attached it to the robust BPNN, and this method was used for innovative target tracking. The author of Reference [27] proposed a comparison-based method to solve the threshold problem, using the alpha-beta family of filters. When working with different devices to improve living standards, we set a threshold, i.e., positive or negative. However, this does not mean that if the value is positive, the living standards are improved, or if the value is negative, the living standards are worsened. Instead, error values were set to enable comparisons, and the alpha-beta filter family was used to achieve the best solution for this problem.

The author of Reference [28] proposed a new accuracy-improvement-based methodology for the alpha-beta filter. They developed a method based on ANN learning in prediction methods for indoor navigation systems to enhance the precision of the alpha-beta filter by reducing the error. The ANN [29] is also the most widely used algorithm.

3. Proposed Schemes

We proposed different algorithms, including DELM, DBN, and SVM, and added them to the $\alpha$-$\beta$ filter. These three machine-learning algorithms were used to evaluate improvements in accuracy and performance, and we compared their results with the conventional $\alpha$-$\beta$ filter. Usually, historical data are used to train forecasting algorithms; these training processes are performed to determine the relationships and hidden relationships between input and output values. Next, the input data are used to train a model; therefore, the purpose of input data is to predict outputs. Predictive algorithms perform well when the training-data environments are similar to the input data and application settings. However, conventional prediction algorithms do not allow for variations in trained models under varying dynamic input situations.
To deal with such problems, we proposed different learning models for application to prediction models. The learning models used DELM, DBN, and SVM. The prediction model used the α-β filter algorithm. These learning modules train data and tune in a prediction model to increase the prediction precision and the performance of the algorithm. The complete conceptual model is shown in Figure 1. In our proposed design, the learning module serves as a monitor for the retrieval of the output of the forecasting algorithm as a response, and it constantly observes the performance of the forecasting algorithm. The learning unit may also consider external constraints that may influence the α-β filter algorithm’s performance. The output of the prediction algorithm also analyzes the current external factors; the tunable parameters may need to be updated by the learning module to the prediction module or when environmental triggers are detected. The learning unit completely replaces the proficient model in the prediction algorithm to increase its performance with regard to prediction accuracy. The complete architecture of our proposed scheme is shown in Figure 2.

![Figure 2. The complete architecture of our proposed scheme.](image)

### 3.1. The Deep Belief Network (DBN)-Based Learning Model

We proposed a deep-belief-network-based (DBN) alpha-beta filter. The model was first developed by Hinton [30]. It is also a very popular prediction method which uses real-world data to predict unseen data. To boost the model performance, we combined this model with other typical time-series prototypes. The model contains different learning components with less complexity and restructures the inputs concerning the probability. The model is mainly based on the restricted Boltzmann machine (RBM) and pre-trains the network with the help of unsupervised learning for every couple of layers. The model of the RBM comprises a hidden layer, Boolean hidden units, a visible layer, and a two-fold-layer neural network.
In the learning module, we used the deep belief network. Inputs to the DBN are the temperature-sensor values and humidity-sensor values. The outputs of the DBN are the alpha-beta values in the prediction module. This model comprises two inputs, and the probability distribution can learn various sets of data. Furthermore, the model consists of a visible layer of units with symmetrical connections, and the model has only one HL of hidden components. Inside the same layer, there are no interconnections. However, it still requires a bipartite graph and the development of its neurons. The hidden-layer probability distribution and layer-wise configuration of the learning process [31] are displayed as follows:

\[
EE(v,h) = -\sum_{i=1}^{n_v} ai vi - \sum_{y=1}^{n_h} hy y - \sum_{i=1}^{n_v} \sum_{y=1}^{n_h} h y w y i v i
\]  

\[
P(v,h) = \left( \frac{e^{-E(v,h)}}{\sum_v \sum_h e^{-E(v,h)}} \right)
\]  

Meanwhile, in the visible layer, \(vi\) is the \(i\)th neuron binary state. The other representations are as follows: The visible layer, \(n_v\), is the number of neurons. The \(y\)th is the Boolean neuron inside the hidden layer of the binary state, and \(hy\) is the number of neurons, \(n_h\), in the hidden layer. The \(wy\) and \(i\) are the weight matrices between the HL and the visible layer. The \(ai\) is the bias vector for the hidden layers and the visible layer, by:

\[
P(h_i = 1|v) = \sigma \left( bj + \sum_{j=1}^{n_h} \omega j, ivi \right)
\]  

\[
P(v_i = 1|h) = \sigma \left( ai + \sum_{j=1}^{n_h} \omega j, ihj \right)
\]  

The above two equations symbolize the activation functions of the HL and the visible layer, where the sigmoid activation function [2] is referred to as \(\sigma\).

Now, we explain how the final output predicts the DBN regression model, and different models are created to train each terminal node. Normalization was completed for each feature’s train and test datasets to adjust the original data between 0 and 1. The Min–Max scaler function was used, and it was fed into the training model. The new input feature \(xi\) means the original value; the input feature minimum value is min(\(x\)), and max(\(x\)) represents the maximum value, which is the new rescaled value of \(x_i\).

The prediction performance for the test data was measured by the error metrics. Equation (5) shows how the data are normalized:

\[
\text{new } x_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}
\]  

In the prediction module, we used the alpha-beta filter. The inputs to the filter are the temperature sensor values. The sensor-reading module obtains the sensor-temperature-reading data and inputs it to the compute delta temperature module. The compute delta temperature module outputs are used as inputs to the predicted actual temperature module and updated velocity. The predicted accurate temperature and initial state module values are input to the previous updated state module. The previous updated velocity module takes the updated value module and the initial velocity state values as inputs. The predicted actual temperature module takes the computed delta temperature, the estimated temperature, and the alpha values as inputs and produces the actual temperature value. The structure model of the deep belief network is shown in Figure 3.
3.2. Deep Extreme Learning Machine (DELM)-Based Model

In the proposed method, the beneficial features of DL and the ELM are combined, and the approach is called the DELM. Figure 4 shows the model of the DELM; the DELM model uses an input layer of two neurons, and we used five hidden layers and an individual HL consisting of twelve neurons and two output layers.

![Figure 3](image-url)  
**Figure 3.** The structure model of deep belief network with alpha-beta filter.

![Figure 4](image-url)  
**Figure 4.** Structure of DELM with alpha-beta filter.

We trained the DELM on historical data to improve the algorithm by taking a different combination of training and testing samples and validating the output results with input data samples. In the training unit of the proposed model, two input parameters were taken, i.e., temperature and humidity values, by a DELM. The model output was also alpha-beta filter values, which were given to the prediction unit as input values. The DELM worked by tuning the parameters to the alpha-beta filter, addressing estimated errors in sensor readings, and intelligently updating the alpha and beta values. The alpha-beta performance was continuously monitored by examining its output results in the training unit.
The alpha values (current temperature values) and the beta values (humidity values) were input into the deep extreme learning machine to the prediction algorithm unit. The prediction algorithm unit was based on alpha and beta filter values, and these values were taken as inputs to predict the desired temperature. The alpha-beta filter does not require all the historical input values, and only the prior outcome values help the algorithm become more intelligent. The system determines the actual state information because of the prior state values; the algorithm is lightweight [11]. In the current research, we tested the dataset with temperature values from noisy temperature-sensor-reading data on the alpha-beta filter.

In comparison, noise is always dependent on temperature-sensor readings and other conditions, and the temperature is deeply dependent on the humidity level and is always affected by increases or decreases in humidity in environments. When a filter reads the temperature sensor and removes noise from incoming data, it acquires the temperature with regard to time, T, and estimates the accurate temperature. The performance is mainly measured by tuning the alpha (temperature) and beta (humidity level) parameters in the alpha-beta filter. The tuned parameters are always updated after every iteration. The structure of the DELM is shown in Figure 4.

Deep Extreme Learning Machine

The deep ELM is a renowned and exciting method which combines the extreme learning machine (ELM) and deep learning. The standard ANN algorithm needs to be trained more for extensive data, involves extra time consumption, has an inexpensive learning rate, and sometimes may lead to overfitting of the model [2]. The ELM method has been used in classification and regression tasks because of its efficiency; this technique is computationally cheap, and the learning rate is speedy. The model comprises two input layers and five hidden layers, with each HL consisting of 10 neurons and two output layers.

Initially, input feature, \( A = [a_{11}, a_{12}, \ldots, a_{1z}] \); a training sample, \([A, B] = \{a_k, b_k\} (i = 1, 2, \ldots, Z)\); and a targeted matrix, \( B = [b_{11}, b_{12}, b_{13}, \ldots, b_{1Z}]\) were taken. The matrices A and B can be described by Equations (6) and (7), respectively. The terms “\( a \) and \( b \)” denote the feature of the input and output matrix, and the weights between the input layer and the hidden layer were adjusted arbitrarily by the ELM, while the weights between the kth input layer nodes and lth hidden layer nodes are represented by \( w_{kl} \), as shown in Equation (8). Furthermore, the HL and output layer neurons’ weights are randomly fixed by ELM and are given in Equation (9), and the weight between the input and HL nodes is symbolized by \( \gamma_{kl} \):

\[
A = \begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1z} \\
a_{21} & a_{22} & \ldots & a_{2z} \\
a_{31} & a_{32} & \ldots & a_{3z} \\
\vdots & \vdots & & \vdots \\
a_{p1} & a_{p2} & \ldots & a_{pz}
\end{bmatrix}
\]  
(6)

\[
B = \begin{bmatrix}
b_{11} & b_{12} & \ldots & b_{1z} \\
b_{21} & b_{22} & \ldots & b_{2z} \\
b_{31} & b_{32} & \ldots & b_{3z} \\
\vdots & \vdots & & \vdots \\
b_{r1} & b_{r2} & \ldots & b_{rz}
\end{bmatrix}
\]  
(7)
Then, in the hidden layers, the biases are arbitrarily selected by the extreme learning machine, given by Equation (10). Furthermore, the ELM calculates the $g(x)$ function, which is the activation function used for the ELM. Equation (11) describes the resultant matrix, and the column vector resultant matrix, $T$, is depicted in Equation (12):

$$w = \begin{bmatrix} w_{11} & \ldots & w_{1p} \\ w_{21} & \ldots & w_{2p} \\ \vdots & \vdots & \vdots \\ w_{l1} & \ldots & w_{lp} \end{bmatrix}$$

$$\gamma = \begin{bmatrix} \gamma_{11} & \ldots & \gamma_{1r} \\ \gamma_{21} & \ldots & \gamma_{2r} \\ \vdots & \vdots & \vdots \\ \gamma_{p1} & \ldots & \gamma_{pr} \end{bmatrix}$$

If we compute Equations (11) and (12), the desired values are attained in Equation (13). $H$ denotes the hidden layer output, and the transpose of $V$ is written as $V'$. Meanwhile, the least squares method was used to simplify the weight matrix parameters, which are denoted by $\gamma$, as shown in Equation (14) [2]:

$$H\gamma = V'$$

$$\gamma = H^+V'$$

The regularization [32] $\gamma$ values were used to further generalize and stabilize the network. The trial-and-error method was selected, because no specific methods are used to recognize the number of hidden layers neurons, and this method also works for the selection of the number of nodes that must be chosen. The output neuron in the second hidden layer is calculated by Equation (15):

$$H_1 = V\gamma^+$$
where $\gamma^+$ is generally the inverse of matrix $y$ and the result of the HL-2 can usually be computed by (16):

$$H_2 = V\gamma^+$$ (16)

$$g(W_1H + B_1) = H_1$$ (17)

The parameters used in Equation (17) are defined as follows: $W_1$ signifies the weight matrix of the initial two HLs, while $H$ represents HL. The probable outputs of the first and second HL are denoted by $H_1$ and bias $B_1$:

$$W_{HE} = g^{-1}(H_1)H_E^+$$ (18)

where $H_E^+$ is the inverse of $H_E$, and AF is denoted by $g(x)$, as shown in Equation (18), and updates the expected output of HL2. We identified any suitable AF $g(x)$, as shown below:

$$H_2 = g(W_{HE}H_E)$$ (19)

The update to $\gamma$, which is the weight matrix between HL2 and HL3, is presented in Equation (20), and $H_3^+$ is the inverse of $H_3$; the final result of HL3 is given in Equation (21):

$$\gamma_{new} = H_2^+V$$ (20)

$$H_3 = V\gamma_{new}^+$$ (21)

where $\gamma_{new}$ is the weight matrix, and its inverse is written as $V\gamma_{new}^+$. The deep ELM classifies the matrix $W_{HE1} = [B_2, W_2]$, and the final results of the third layer are calculated by using the equations shown above ((13) and (14)):

$$H_3 = g^{-1}(H_2W_2 + B_2) = g(W_{HE1}H_{E1})$$ (22)

$$W_{HE1} = \gamma^{-1}(H_3)H_{E1}^+$$ (23)

For the rest of the layers, $g(x)$ is the activation function (AF), and $g^{-1}(x)$ is its inverse AF, and the second hidden layer is denoted by $H_2$. The weights between the hidden layer 2 and layer 3 are represented by $W_2$, where $B_2$ signifies the bias. The $H_{E1}$ inverse is characterized by $H_{E1}^+$, and Equation (24) represents the sigmoid function. Now, we compute the output of the third HL, as given in Equation (25):

$$g(x) = \frac{1}{1 + e^{-x}}$$ (24)

$$H_3 = g(W_{HE1}H_{E1})$$ (25)

Regarding the third HL and the ending layer output, the resultant weighted matrix is computed in Equation (26), and the expected output of the third HL is shown in Equation (27). The other hidden-layer calculations follow the same procedure as $H_5, H_6, H_7$, and so on:

$$\gamma_{new} = H_4^+ \left( \frac{1}{\lambda} + H_4^T H_4 \right)^{-1} V$$ (26)

$$H_4 = V\gamma_{new}^+$$ (27)

3.3. SVM-Based Learning Module

The third algorithm used is also a prevalent method, and we proposed a support vector machine (SVM) method that was primarily created for use in binary classification tasks. Initially, we trained the SVM on historical data in the training unit by taking a different combination of training and testing samples and then validating the output results with input data samples. In the training unit of the proposed model, we took two input parameters, i.e., temperature values and humidity values, via a support vector
The SVM output was also alpha-beta filter values, which were input into the prediction unit as input values. The SVM worked by tuning the parameters to the alpha-beta filter, continuously trying to address the assessed errors in sensor readings, and intelligently updating alpha and beta values. The performance of the alpha-beta filter was continuously monitored by examining its output results in the training unit. Today, many variations of the SVM have been developed to resolve more dense classification and regression problems [2] with the help of kernel tricks. SVMs always rely on the input size and the nature of the problem, and a suitable kernel function can be selected from radial basis functions (RBFs), linear functions, polynomial functions, etc. For the selection of kernels, we performed experiments by using three different kernels. The results attained through the use of a linear kernel were the best and are reported in this paper. The structure of the proposed SVM-based alpha-beta filter is shown in Figure 5.

Figure 5. The structure of the proposed SVM-based alpha-beta filter.

3.4. Alpha-Beta Filter

Filters belonging to the alpha-beta family of filters are the simplest filters used for smoothness, control, and estimation. Their architecture is similar to linear filters. The main benefit of the alpha-beta filter is that it is very easy to use because a complex benchmark is not required in order to train it for another algorithm. The model is obtained from the KF [2]. The alpha-beta filter needs very little space and computation power compared to the Kalman filter. The equations shown below are mathematical calculations which define each algorithm step. In the first step, initiation occurs, as represented in Equations (28) and (29):

\[ x_{k-1} = c_1 \]  
\[ v_{k-1} = c_2 \]  

Equation (30) was applied to update the position, and to read sensor data, Equation (31) was used:

\[ \hat{x}_k = x_{k-1} + v_{k-1} \cdot \Delta t \]  
\[ x_j = Sensor() \]  

To compute the difference, Equation (32) was used, and for the calculation and prediction of positions, Equation (33) was applied:

\[ \Delta \hat{x}_k = x_j - \hat{x}_k \]
\[ \bar{x}_k = \bar{x}_k + \alpha \cdot \Delta \bar{x}_k \]  

(33)

To calculate predicted velocity, Equation (34) was applied, and to update the position and velocity for the next iteration, we used Equations (35) and (36):

\[ \hat{v}_k = \hat{v}_{k-1} + \beta \cdot \Delta \bar{x}_k / \Delta t \]  

(34)

\[ i_{k-1} = i_k \]  

(35)

\[ v_{k-1} = v_k \]  

(36)

4. Implementation and Performance Evaluation

In this section, we discuss the implementation of the proposed method and the final results.

4.1. Implementation

Experiments regarding a deep-belief-network-based \( \alpha \)-\( \beta \) filter, a deep-extreme-learning-machine-based \( \alpha \)-\( \beta \) filter, and a support-vector-machine-based alpha-beta filter were carried out on MSI DESKTOP-SC4U005. The computer has an “11th Gen Intel(R) Core (TM) i7-11700KF CPU @ 3.60 GHz, 32 GB ram, Nvidia Quadro M1200 4 GB graphics card, and MATLAB R2022a”. The implementation and simulation configuration are shown in Table 2. For the analysis of our models and the performance assessment of the multi-learning-algorithm-based \( \alpha \)-\( \beta \) filter, we used the real weather dataset [2] of Korea and the data gathered by temperature and humidity sensors over three years, comprising hourly data with simulated noisy sensor readings, and some errors were added to the data. The total number of days for three years was \( 365 \times 3 = 1095 \), and there were 26,280 total data instances. Initially, when the filter-readings sensor obtained values via the typical method, the root mean square error, a value of 5.21, was obtained, which is very high in terms of the RMSE. By using these three algorithms, we attempted to reduce the error. The data representation is shown in Figure 6.

![Figure 6. Representation of humidity, sensor readings, temperature, and error.](image-url)
### Table 2. Implementation and simulation configuration.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming tool</td>
<td>MATLAB R2021b</td>
</tr>
<tr>
<td>Operating system</td>
<td>Window 11</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel(R) Core (TM) i7-11700KF CPU@3.60 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>32 GB</td>
</tr>
<tr>
<td>Learning algorithms</td>
<td>DBN, DELM, and SVM</td>
</tr>
<tr>
<td>Target algorithm</td>
<td>Alpha-Beta filter</td>
</tr>
<tr>
<td>Graphics</td>
<td>Nvidia Quadro M1200</td>
</tr>
</tbody>
</table>

4.2. Performance Criterion

The overall performance of the proposed approach was comprehensively assessed by using two deterministic performance metrics, i.e., RMSE and mean absolute error, and a heatmap representation of the correlation between the training and testing datasets was created as shown in Figure 7.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=0}^{N} (R_i - P_i)^2} \quad (37)
\]

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |R_i - P_i| \quad (38)
\]

\[
K = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{(x - \bar{x})^2 \sum (y - \bar{y})^2}} \quad (39)
\]

![Heatmap representation of the correlation between training and testing datasets.](image)

Figure 7. Heatmap representation of the correlation between training and testing datasets.

5. Results and Discussion

In this section, we briefly discuss the multi-algorithm-based alpha-beta filter. The proposed hybrid method was based on the DBN, SVM, and DELM. Figure 8 shows the three different cross-validation paradigms used in training and testing.
4.2. Performance Criterion

The overall performance of the proposed approach was comprehensively assessed by using two deterministic performance metrics, i.e., RMSE and mean absolute error, and a heatmap representation of the correlation between the training and testing datasets was created as shown in Figure 7.

\[
\text{RMSE} = \sqrt{\frac{1}{N} \sum (R - P)^2}
\]

\[
\text{MAE} = \frac{1}{N} \sum |R - P|
\]

\[
K = \sum (x - \bar{x})(y - \bar{y})
\]

Figure 7. Heatmap representation of the correlation between training and testing datasets.

5. Results and Discussion

In this section, we briefly discuss the multi-algorithm-based alpha-beta filter. The proposed hybrid method was based on the DBN, SVM, and DELM. Figure 8 shows the three different cross-validation paradigms used in training and testing.

Figure 8. Three different cross-validation paradigms used in training and testing.

5.1. DBN-Based Alpha-Beta Filter Results

The DBN learning model was constructed by using training and testing sets of 70/30. The input to the DBN was composed of temperature values and humidity values, and two output data were used. The DBN was composed of two steps. In the first step, we used supervised learning to construct the RBM network. The RBM settings were a batch size of 12, momentum of 0, and 300 epochs. In the second step, the RBM network learned the backpropagation algorithm of supervised learning. The backpropagation used a batch size of 12 and 300 epochs. The DBN was used for the \( \alpha-\beta \) filter algorithm to learn from, and the learned values of the DBN were passed through to the \( \alpha-\beta \) filter to produce the final output values. The mean-absolute-error and root-mean-square-error values were estimated for the typical alpha-beta filter and the proposed DBN-based \( \alpha-\beta \) filter to evaluate their performance. Regarding the typical method, we obtained an RMSE of 5.216 and an MAE of 3.951 when we applied the DBN with two inputs, two outputs, and the RBM. The three-fold cross-validation method was used to comprehensively calculate the performance. The best results were an RMSE of 3.605 and an MAE of 2.610. The results are shown in Table 3. The performance of our model varied depending on the number of nodes constituting the DBN.

5.2. DELM-Based Alpha-Beta Filter Results

We implemented a deep-extreme-learning-machine-based \( \alpha-\beta \) filter; here, the same training and test set was used: 30/70. The input to the DELM was composed of temperature sensor values and humidity sensor values, there were five hidden layers, and each hidden layer was composed of 12 neurons in the hidden layer and a sigmoid activation function (AF); the linear AF was used by the output layer. In our research, we used two output layers and named them alpha and beta in the final prediction results; these two values were input to the alpha-beta filter to compute the final output results. To evaluate the performance of the DELM-based alpha-beta filter, the root mean square error and mean absolute error were estimated regarding the typical alpha-beta filter and the newly proposed DELM-based \( \alpha-\beta \) filter. When we applied the DELM, the result was overwhelmingly positive. The three-fold cross-validation method was used to comprehensively evaluate the performance. The best results were an RMSE of 3.901 and an MAE of 2.811. The DELM-based results are shown in Figure 9.
5.1. DBN-Based Alpha-Beta Filter Results

The DBN learning model was constructed by using training and testing sets of 70/30. The input to the DBN was composed of temperature values and humidity values, and two output data were used. The DBN was composed of two steps. In the first step, we used supervised learning to construct the RBM network. The RBM settings were a batch size of 12, momentum of 0, and 300 epochs. In the second step, the RBM network learned the backpropagation algorithm of supervised learning. The backpropagation used a batch size of 12 and 300 epochs. The DBN was used for the $\alpha$-$\beta$ filter algorithm to learn from, and the learned values of the DBN were passed through to the $\alpha$-$\beta$ filter to produce the final output values. The mean-absolute-error and root-mean-square-error values were estimated for the typical alpha-beta filter and the proposed DBN-based $\alpha$-$\beta$ filter to evaluate their performance. Regarding the typical method, we obtained an RMSE of 5.216 and an MAE of 3.951 when we applied the DBN with two inputs, two outputs, and the RBM. The three-fold cross-validation method was used to comprehensively calculate the performance. The best results were an RMSE of 3.605 and an MAE of 2.610. The results are shown in Table 3. The performance of our model varied depending on the number of nodes constituting the DBN.

5.2. DELM-Based Alpha-Beta Filter Results

We implemented a deep-extreme-learning-machine-based $\alpha$-$\beta$ filter; here, the same training and test set was used: 30/70. The input to the DELM was composed of temperature sensor values and humidity sensor values, there were five hidden layers, and each hidden layer was composed of 12 neurons in the hidden layer and a sigmoid activation function (AF); the linear AF was used by the output layer. In our research, we used two output layers and named them alpha and beta in the final prediction results; these two values were input to the alpha-beta filter to compute the final output results. To evaluate the performance of the DELM-based alpha-beta filter, the root mean square error and mean absolute error were estimated regarding the typical alpha-beta filter and the newly proposed DELM-based $\alpha$-$\beta$ filter. When we applied the DELM, the result was overwhelmingly positive. The three-fold cross-validation method was used to comprehensively evaluate the performance. The best results were an RMSE of 3.901 and an MAE of 2.811. The DELM-based results are shown in Figure 9.

5.3. SVM-Based Alpha-Beta Filter Results

Figure 10 shows the sample data used for the SVM-based alpha-beta filter algorithm. The SVM took two inputs, i.e., temperature and humidity, with ten support vectors, and these support vectors were used to map functions and apply the kernel transformation. The final values were input into the $\alpha$-$\beta$ filter to produce the output results. However, this method is complex and did not produce a good output performance. Sample data used for SVM-based alpha-beta filter (200 data instances) is shown in Figure 11. For the performance assessment of the SVM-based $\alpha$-$\beta$ filter, the mean absolute error and root mean square error of the typical alpha-beta filter and the newly proposed SVM-based $\alpha$-$\beta$ filter were estimated. When we applied the SVM, the result was overwhelmingly positive. The three-fold cross-validation method was used to comprehensively calculate the performance. The best results were an RMSE of 4.015 and an MAE of 3.218. The SVM-based alpha-beta filter results are shown in Figure 10.

![Figure 9](image-url) Figure 9. The deep-extreme-learning-machine-based alpha-beta filter results.

![Figure 10](image-url) Figure 10. SVM-based alpha-beta filter output.
Figure 11. Sample data used for SVM-based alpha-beta filter (200 data instances).

For the performance assessment of the SVM-based $\alpha$-$\beta$ filter, the mean absolute error and root mean square error of the typical alpha-beta filter and the newly proposed SVM-based $\alpha$-$\beta$ filter were estimated. When we applied the SVM, the result was overwhelmingly positive. The three-fold cross-validation method was used to comprehensively calculate the performance. The best results were an RMSE of 4.015 and an MAE of 3.218. The SVM-based alpha-beta filter results are shown in Figure 10.

Table 3 shows comparisons of the different algorithms’ performances; we reached the conclusion that the DBN algorithm prevails compared to the DELM and SVM when applied to the alpha-beta filter. Figures 12 and 13 show the comparison of the typical alpha-beta filter and the proposed multi-method in terms of RMSE and MAE.

Table 3. The comparison of typical alpha-beta filter and proposed multi-method in terms of RMSE and MAE.

<table>
<thead>
<tr>
<th>Performance Metrics</th>
<th>$\alpha$-$\beta$ Filter without Learning Algorithm</th>
<th>$\alpha$-$\beta$ Filter with DBN</th>
<th>$\alpha$-$\beta$ Filter with DELM</th>
<th>$\alpha$-$\beta$ Filter with SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSE 1</td>
<td>5.216</td>
<td>3.605</td>
<td>3.901</td>
<td>4.015</td>
</tr>
<tr>
<td>MAE 1</td>
<td>3.951</td>
<td>2.610</td>
<td>2.811</td>
<td>3.218</td>
</tr>
</tbody>
</table>

1 Root mean square error (RMSE), Mean absolute error (MAE).
5.216
3.951
3.605
2.61
3.901
2.811
4.015
3.218

Figure 12. The comparison of typical alpha-beta filter and proposed multi-method in terms of RMSE and MAE.

Figure 13. The error minimization of typical alpha-beta filter and proposed multi-method.

6. Conclusions

It is always difficult to enhance the prediction accuracy of an algorithm. In this paper, we proposed a new learning model for prediction algorithms, including DBN-, DELM-, and SVM-based $\alpha$-$\beta$ filter algorithms, used to enhance the accuracy and tune the parameters of $\alpha$ and $\beta$ in dynamic conditions. The design of the proposed system uses the deep belief network, the support vector machine, and the deep extreme learning machine as learning algorithms for the alpha-beta filter algorithm to increase the prediction accuracy of the filter. The performance of the proposed $\alpha$-$\beta$ filter with the application of different learning algorithms was evaluated by using MAE and RMSE values. We compared the results of these four algorithms used in the alpha-beta filter algorithm, and the results with the best accuracy were from ANN, with an RMSE of 3.605 and an MAE of 2.610, as compared with the typical alpha-beta filter algorithm, which achieved an RMSE of 5.216 and an MAE of 3.951 in terms of RMSE and MAE.
Author Contributions: Data curation, J.K.; formal analysis, J.K.; funding acquisition, K.K.; investigation, J.K.; methodology, J.K.; software, J.K.; supervision, K.K.; validation, K.K.; review and editing, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(No.2020-0-01441, Artificial Intelligence Convergence Research Center(Chungnam National University)).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: This work was supported by research fund of Chungnam National University. This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(No.2020-0-01441, Artificial Intelligence Convergence Research Center(Chungnam National University)).

Conflicts of Interest: The authors declare no conflict of interest.

References

3. Ullah, I.; Fayaz, M.; Kim, D. Improving accuracy of the Kalman filter algorithm in dynamic conditions using ANN-based learning module. *Symmetry* 2019, 11, 94. [CrossRef]
4. Phyo, P.P.; Jeenanunta, C. Daily Load Forecasting Based on a Combination of Classification and Regression Tree and Deep Belief Network. *IEEE Access* 2021, 9, 152226–152242. [CrossRef]
5. Fayaz, M.; Kim, D. A prediction methodology of energy consumption based on deep extreme learning machine and comparative analysis in residential buildings. *Electronics* 2018, 7, 222. [CrossRef]


27. Sighencea, B.I.; Stanciu, R.I.; Sorândaru, C.; Căleanu, C.D. The Alpha-Beta Family of Filters to Solve the Threshold Problem: A Comparison. Mathematics 2022, 10, 880. [CrossRef]


