A Comparison of Nonlinear Kalman Filtering Applied to Feed–forward Neural Networks as Learning Algorithms

Wieslaw Pietruszkiewicz
SDART Ltd
One Central Park, Northampton Road
Manchester M40 5WW, United Kindgom
Email: wpietrus@sdart.co.uk

Abstract—In this article we present an application of Kalman filtering in Artificial Intelligence, where nonlinear Kalman filters were used as a learning algorithms for feed–forward neural networks. In the first part of this article we have examined two modern versions of nonlinear filtering algorithms i.e. Unscented Kalman Filter and Square Root Central Difference Kalman Filter. Later, we present performed experiments, where we have compared UKF and SRCDKF with an reference algorithm i.e. Gradient Descent Error Backpropagation being the most popular neural network learning algorithm. To prove filters high learning abilities in case of noisy problems, we have used a noisy financial dataset during the experiments. This dataset was selected due to unusually separable classes subspaces. The results of experiments, presented in the last part of this paper, show greater accuracy for nonlinear Kalman filters that overperformed popular Error Backpropagation learning algorithm.

Index Terms—Neural Networks, Unscented Kalman Filter, Square Root Central Difference Kalman Filter, Bankruptcy prediction

I. INTRODUCTION

Neural networks are one of the most popular Artificial Intelligence methods. They are used in regression or classification tasks. Their flexibility and ability to work as “black-box" causes that many researchers and practitioners chose them among the other algorithms. The crucial part of constructing neural networks, apart of selecting its appropriate architecture, is to use an accurate and robust learning algorithm. The most popular learning algorithm for feed–forward neural networks is Gradient Descent Error Backpropagation, which is easy to apply and is often efficient enough. However it gives unsatisfactory results when the learning material was affected by the noise. Herein we present the results of experiments, that were performed using noisy data. We compared 2 nonlinear Kalman learning algorithms i.e. Unscented Kalman Filter and Square Root Central Difference Kalman Filter with Gradient Descent Error Backpropagation. The two presented filters and other Kalman filters are very popular algorithms, having various applications in control systems or navigation. However, applications of Kalman filtering in Artificial Intelligence are found too rarely, while modern derivative–free nonlinear filters are almost “black-boxes" comparing to the older generation of filters.

II. THE UNSCENTED TRANSFORMATION AND UNSCENTED KALMAN FILTER

Real life applications of data processing often have to deal with noisy datasets. When an observed signal is a superposition of pure signal and noise, its usefulness is decreasing as well as accuracy of data processing methods. If we assume that observed object or process may be represented as a dynamic system, then its general form will be:

\[ x_{k+1} = G(x_k, u_k, \nu_k) \]  
\[ y_k = H(x_k, n_k) \]

where: \( G \) – process equation, \( H \) – observation equation, \( x_k \) - state variables, \( u_k \) - system inputs, \( y_k \) - system outputs, \( \nu_k \) - process noise and \( n_k \) - observation noise.

Kalman filters (sometimes loosely related) belong to one family, being a popular group of filtering algorithms. They have many applications in different technical problems e.g estimation of observed variables disturbed by noise or estimation of parameters of an unknown system – with is a typical machine learning task.

Retransformed system for parameter estimation transforms into:

\[ w_{k+1} = w_k + r_k \]
\[ d_k = g(x_k, w_k + c_k) \]

There are some examples of the successful Kalman filtering usage for neural networks training. In [1] researchers presented various EKF approaches to neural network learning. Another research [2] described how EKF was applied in currency prediction done by neural network. EKF was also used to train neural network that was applied in chemistry [3]. In [4] comparison of EKF training with Levenberg-Marquardt algorithm was shown. The other papers describe different types of filters e.g [5] compared EKF as well as UKF or [6] presented a comparison of EKF, UKF and Divided Difference Filter (DDF). UKF was also used to learn neural network applied to mechanics [7], while [8] described an usage of UKF in neural framework for unmanned aerial vehicles.

I. INTRODUCTION

Neural networks are one of the most popular Artificial Intelligence methods. They are used in regression or classification tasks. Their flexibility and ability to work as “black-box" causes that many researchers and practitioners chose them among the other algorithms. The crucial part of constructing neural networks, apart of selecting its appropriate architecture, is to use an accurate and robust learning algorithm. The most popular learning algorithm for feed–forward neural networks is Gradient Descent Error Backpropagation, which is easy to apply and is often efficient enough. However it gives unsatisfactory results when the learning material was affected by the noise. Herein we present the results of experiments, that were performed using noisy data. We compared 2 nonlinear Kalman learning algorithms i.e. Unscented Kalman Filter and Square Root Central Difference Kalman Filter with Gradient Descent Error Backpropagation. The two presented filters and other Kalman filters are very popular algorithms, having various applications in control systems or navigation. However, applications of Kalman filtering in Artificial Intelligence are found too rarely, while modern derivative–free nonlinear filters are almost “black-boxes" comparing to the older generation of filters.

II. THE UNSCENTED TRANSFORMATION AND UNSCENTED KALMAN FILTER

Real life applications of data processing often have to deal with noisy datasets. When an observed signal is a superposition of pure signal and noise, its usefulness is decreasing as well as accuracy of data processing methods. If we assume that observed object or process may be represented as a dynamic system, then its general form will be:

\[ x_{k+1} = G(x_k, u_k, \nu_k) \]  
\[ y_k = H(x_k, n_k) \]

where: \( G \) – process equation, \( H \) – observation equation, \( x_k \) - state variables, \( u_k \) - system inputs, \( y_k \) - system outputs, \( \nu_k \) - process noise and \( n_k \) - observation noise.

Kalman filters (sometimes loosely related) belong to one family, being a popular group of filtering algorithms. They have many applications in different technical problems e.g estimation of observed variables disturbed by noise or estimation of parameters of an unknown system – with is a typical machine learning task.

Retransformed system for parameter estimation transforms into:

\[ w_{k+1} = w_k + r_k \]
\[ d_k = g(x_k, w_k + c_k) \]

There are some examples of the successful Kalman filtering usage for neural networks training. In [1] researchers presented various EKF approaches to neural network learning. Another research [2] described how EKF was applied in currency prediction done by neural network. EKF was also used to train neural network that was applied in chemistry [3]. In [4] comparison of EKF training with Levenberg-Marquardt algorithm was shown. The other papers describe different types of filters e.g [5] compared EKF as well as UKF or [6] presented a comparison of EKF, UKF and Divided Difference Filter (DDF). UKF was also used to learn neural network applied to mechanics [7], while [8] described an usage of UKF in neural framework for unmanned aerial vehicles.

I. INTRODUCTION

Neural networks are one of the most popular Artificial Intelligence methods. They are used in regression or classification tasks. Their flexibility and ability to work as “black-box" causes that many researchers and practitioners chose them among the other algorithms. The crucial part of constructing neural networks, apart of selecting its appropriate architecture, is to use an accurate and robust learning algorithm. The most popular learning algorithm for feed–forward neural networks is Gradient Descent Error Backpropagation, which is easy to apply and is often efficient enough. However it gives unsatisfactory results when the learning material was affected by the noise. Herein we present the results of experiments, that were performed using noisy data. We compared 2 nonlinear Kalman learning algorithms i.e. Unscented Kalman Filter and Square Root Central Difference Kalman Filter with Gradient Descent Error Backpropagation. The two presented filters and other Kalman filters are very popular algorithms, having various applications in control systems or navigation. However, applications of Kalman filtering in Artificial Intelligence are found too rarely, while modern derivative–free nonlinear filters are almost “black-boxes" comparing to the older generation of filters.

II. THE UNSCENTED TRANSFORMATION AND UNSCENTED KALMAN FILTER

Real life applications of data processing often have to deal with noisy datasets. When an observed signal is a superposition of pure signal and noise, its usefulness is decreasing as well as accuracy of data processing methods. If we assume that observed object or process may be represented as a dynamic system, then its general form will be:

\[ x_{k+1} = G(x_k, u_k, \nu_k) \]  
\[ y_k = H(x_k, n_k) \]

where: \( G \) – process equation, \( H \) – observation equation, \( x_k \) - state variables, \( u_k \) - system inputs, \( y_k \) - system outputs, \( \nu_k \) - process noise and \( n_k \) - observation noise.

Kalman filters (sometimes loosely related) belong to one family, being a popular group of filtering algorithms. They have many applications in different technical problems e.g estimation of observed variables disturbed by noise or estimation of parameters of an unknown system – with is a typical machine learning task.

Retransformed system for parameter estimation transforms into:

\[ w_{k+1} = w_k + r_k \]
\[ d_k = g(x_k, w_k + c_k) \]
where: \( w_k \) – is stationary process relating to parameters (weights), \( r_k \) – is an artificial noise driving \( w \) vector and \( d_k \) – is an observation of \( w \) vector.

The estimation of parameter is impossible to be achieved by an original Kalman filter (KF - see [9]), due to its purely linear construction. This limitation does not apply to a group of nonlinear Kalman filters e.g. Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) or Central Difference Kalman Filter (CDKF). While EKF is the oldest and the most popular nonlinear algorithm, in a matter of fact it is very simple extension of KF to nonlinear problems, done by simple linearization of estimated systems in form of first order Taylor series. Therefore, EKF is not accurate enough in case where strong nonlinearity exists.

An example of more accurate nonlinear Kalman filter can be Unscented Kalman filter that incorporates Unscented Transformation (UT in abbrev. – see [10] or [11]). The main idea of UT is to calculate statistics of random variables by propagating a set of specially chosen points (sigma points) via the real model of the system and calculating their statistic parameters, comparing to approximation achieved from inaccurate system’s simplification (EKF). The detailed steps of UKF in form for parameter estimation are (more details and other UKF forms may be found in [12], [13] or [14]):

*Step I – Initialization*

\[
\hat{x}_0 = E[w] \tag{5}
\]

\[
P_{w0} = E \left[ (w - \hat{w}_0) (w - \hat{w}_0)^T \right] \tag{6}
\]

For \( k \in \{1, \ldots, \infty\} \)

*Step II – Time-update:*

\[
\hat{w}_k^- = \hat{w}_{k-1} \tag{7}
\]

\[
P_{w_k^-} = P_{w_{k-1}} + R_{r_{k-1}} \tag{8}
\]

*Step III – Calculation of sigma points for measurement-update:*

\[
\chi_{k|k-1} = \left[ \hat{w}_k \quad \hat{w}_k + \tau \sqrt{P_{w_k^-} \quad \hat{w}_k^T - \tau \sqrt{P_{w_k^-}}} \right] \tag{9}
\]

*Step IV - Measurement update equations:*

\[
T_{k|k-1} = g(x_k, \chi_{k|k-1}) \tag{10}
\]

\[
\hat{d}_k = \sum_{i=0}^{2L} u_i^{(m)} T_{i,k|k-1} \quad \text{or} \quad \hat{d}_k = g(x_k, \hat{w}_k^-) \tag{11}
\]

\[
P_{d_k} = \sum_{i=0}^{2L} u_i^{(c)} \left( T_{i,k|k-1} - \hat{d}_k \right) \left( T_{i,k|k-1} - \hat{d}_k \right)^T + R_{r_{k}} \tag{12}
\]

\[
P_{w_k,d_k} = \sum_{i=0}^{2L} u_i^{(c)} \left( \chi_{i,k|k-1} - \hat{w}_k \right) \left( T_{i,k|k-1} - \hat{d}_k \right)^T \tag{13}
\]

\[
K_k = P_{w_k,d_k} P_{\hat{d}_k}^{-1} \tag{14}
\]

\[
\hat{w}_k = \hat{w}_k^- + K_k \left( d_k - \hat{d}_k \right) \tag{15}
\]

\[
P_{w_k} = P_{w_k^-} - K_k P_{d_k} K_k^T \tag{16}
\]

Where:

\[
x^a = \left[ x^T \quad \nu^T \quad n^T \right]^T, \tag{17}
\]

\[
\chi^a = \left[ (\chi^2)^T \quad (\chi^\nu)^T \quad (\chi^a)^T \right]^T, \tag{18}
\]

\[
\tau = \sqrt{l + \lambda}. \tag{19}
\]

The used symbols denote: \( \lambda \) – scaling parameter, \( l \) – dimensionality of extended state vector, \( R^p \) – process noise covariance, \( R^m \) – measurement noise covariance.

### III. SQUARE ROOT CENTRAL DIFFERENCE KALMAN FILTER

The second nonlinear filter we have used is Square Root Central Difference Kalman Filter (SRCDKF) being a derivative-free, like all Sigma Point Kalman Filters (SPKF) and less computational variant of Central Difference Kalman Filter. The term “Square Root” denotes a more efficient method of matrix inversion comparing to CDKF. SRCDKF steps and equations, in version for parameter estimation, are [12]:

*Step I - Initialization:*

\[
\hat{w}_0 = E[w] \tag{20}
\]

\[
S_{w0} = \text{chol} \left\{ E \left[ (w_0 - \hat{w}_0) (w_0 - \hat{w}_0)^T \right] \right\} \tag{21}
\]

For \( k \in \{1, \ldots, \infty\} \)

*Step II - Time-update:*

\[
S_{w_k} = \sqrt{\lambda_{RLS}} S_{w_{k-1}}, \quad \text{or} \quad S_{w_k} = S_{w_{k-1}} + D_{r_{k-1}} \tag{22}
\]

where:

\[
D_{r_{k-1}} = - \text{diag} \left\{ S_{w_{k-1}} \right\} + \sqrt{\text{diag} \left\{ S_{w_{k-1}} \right\}^2 + \text{diag} \left\{ R_{r_{k-1}} \right\}} \tag{23}
\]

*Step III - Calculate sigma–points for measurement–update:*

\[
\chi_{k|k-1} = \left[ \hat{w}_k \quad \hat{w}_k + hS_{\hat{w}_k} \quad \hat{w}_k - hS_{\hat{w}_k} \right] \tag{24}
\]

*Step IV - Measurement–update:*

\[
T_{k|k-1} = g(w_k, \chi_{k|k-1}) \tag{25}
\]

\[
\hat{d}_k = \sum_{i=0}^{2L} u_i^{(m)} T_{i,k|k-1} \tag{26}
\]

\[
P_{w_k,d_k} = \sum_{i=0}^{2L} u_i^{(c)} \left( \chi_{i,k|k-1} - \hat{w}_k \right) \left( T_{i,k|k-1} - \hat{d}_k \right)^T \tag{27}
\]

\[
K_k = P_{w_k,d_k} P_{\hat{d}_k}^{-1} \tag{28}
\]

\[
\hat{w}_k = \hat{w}_k^- + K_k \left( d_k - \hat{d}_k \right) \tag{29}
\]

\[
P_{w_k} = P_{w_k^-} - K_k P_{d_k} K_k^T \tag{30}
\]
\[ S_{dk} = \text{qr}\left\{ \sqrt{w_{i1}^{(c1)}} \left( \Upsilon_{1:L,k|k-1} - \Upsilon_{L+1:2L,k|k-1} \right) \right\} \]

\[ \sqrt{w_{i1}^{(c2)}} \left( \Upsilon_{1:L,k|k-1} + \Upsilon_{L+1:2L,k|k-1} - 2\Upsilon_{0,k|k-1} \right) \sqrt{\Sigma_n} \right\} \]

\[ P_{wk}d_k = \sqrt{w_{i1}^{(c1)}} S_{w_k}^{-1} \left[ \Upsilon_{1:L,k|k-1} - \Upsilon_{L+1:2L,k|k-1} \right]^T \]

\[ K_k = \left( P_{wk}d_k / S_{wk}^T \right) / S_{dk} \]

\[ \hat{w}_k = \hat{w}_k^- + K_k \left( d_k - d_k^- \right) \]

\[ U = K_k S_{dk} \]

\[ S_{wk} = \text{cholupdate} \left\{ S_{w_k}, U, -1 \right\} \]

The used operators denote: chol – is Cholesky method of matrix factorization, diag – is operation of null-ing all elements of square matrix apart of its diagonal, qr – is QR matrix decomposition and cholupdate – is a Cholesky factor updating.

From systems theory perspective, the process of machine learning is a parameters estimation for identified (learned) dynamic system. For feed–forward neural networks system’s Equations 1 and 2 may be transformed into a form of [1]:

\[ w_{k+1} = w_k + \nu_k \]

\[ y_k = h_k \left( w_k, u_k \right) + n_k \]

where: \( k \) – step number, \( w \) – state variables i.e. neurons weights vector, \( y \) – output vector and \( u \) – input vector. It is assumed that both \( \nu \) and \( n \) are white noise i.e. \( E \left[ \nu_i \nu_j^T \right] = \delta_{ij} R^\nu \) and \( E \left[ n_i n_j^T \right] = \delta_{ij} R^\nu \) covariace matrices respectively, while \( E \left[ \nu_i \omega_j^T \right] = 0 \) for all \( i \) and \( j \).

The Equation 32 represents optimized neural weights being driven by noise (estimated noise is equivalent to the weights correction in error backpropagation algorithm). The network observation is represented by Equation 33

UKF and SRCDKF were selected as a learning algorithms for neural networks instead of EKF, because numerous experiments prove better performance of SPKF (a general group where UKF and SRCDKF belong to) over standard EKF e.g. [15], [12] or [14]. It is also important that SPKFs are also simpler to apply (comparing to EKF) as they do not require to compute matrix of partial derivatives (the Jacobian matrix).

We compare both nonlinear filters i.e. UKF and SRCDKF with Error Backpropagation, which for a loop-free network with an error \( E = \sum_k (t_k - o_k)^2 \), where \( k \) is a number of output units, calculates the optimal weights \( w_{ij} \) according to gradient descent rule [16]:

\[ \Delta w_{ij} = \frac{\partial E}{\partial w_{ij}} = 2 \sum (t_k - o_k) \frac{\partial o_k}{\partial w_{ij}} , \] (34)

In the classic and the most popular version of Error Backpropagation – Gradient Descent algorithm, the weights corrections are multiplied by a constant value (learning ratio). It can lead to increase of error for noise data in case of large values of learning ratio, while low values of this parameter increase the number epochs. The important capability of Kalman filters is an auto–adjustment of gain ratio according to the estimated error. As the result, weights correction are finely tuned to samples i.e. noisy samples do not cause weights oscillations like for EB algorithm.

**IV. RESULTS OF EXPERIMENTS**

To prove high capabilities of SPKFs i.e. UKF and SRCDKF, we have performed experiments over a financial dataset described at [17] (dataset is available at [18]) , that was used test classification accuracy for neural networks, which where trained by different learning algorithms – UKF, SRCDKF and EB. The used dataset contained samples from 128 bankrupt and 112 non-bankrupt companies, being described by 30 real-valued attributes and 1 class variable (failed/non-failed). All 30 input attributes were financial ratios and as it can be noticed on Figure 1 samples from 2 groups were not clearly separated (the colour of dots denotes their class assignments). Therefore, class attribute was noisy. However, the noise wasn’t artificially added to the dataset, but was a result of unclear bankruptcy procedures used by courts (giving different values of class attribute for similar companies). We expected that Error Backpropagation algorithm will be constantly oscillating due to opposite weight corrections for misrecognized samples.

As many from the input attributes were redundant, we selected 5 attributes. These the most important attributes were presented in Table I. A subset of them, containing from 2 to 5 attributes to was used to build neural networks in different sizes. The number of neurons in hidden layer was set to 5, 10 or 15 neurons and activation function was tansign. For each network variant (no. of inputs and no. of hidden neurons) we tested classification accuracy using UKF (Table II), SRCDKF (Table III) and Gradient Descent Error Backpropagation (Table IV).

The Figure 2 presets how classification error was decreasing when the numbers of epochs was increasing. For this part of experiments we have tested the progress of learning using different values of learning ratio i.e. 0.1, 0.05 and 0.02.

The Error Backpropagation epochs are equivalent for Kalman learners iterations. For Kalman learners – UKF and SRCDKF – it was necessary to examine influence of iterations number on classification error. The results of these experiments are presented on Figure 3 (for UKF filter) and Figure 4 (for SRCDKF filter). As it can be noticed on Figure 3 increasing number of iterations caused reduction of the classification error.

The similar observation was possible for SRCDKF trained networks (Figure 4) and for both variants of networks it was
TABLE I

The Most Important Attributes (Decreasing Order of Significance)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Ratio</th>
<th>Selected attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_9$</td>
<td>net profit $\div$ total assets</td>
<td></td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>net profit $\div$ receivables</td>
<td></td>
</tr>
<tr>
<td>$X_{15}$</td>
<td>net profit $\div$ (equity + long term liabilities)</td>
<td></td>
</tr>
<tr>
<td>$X_5$</td>
<td>working capital $\div$ total assets</td>
<td></td>
</tr>
<tr>
<td>$X_2$</td>
<td>cash $\div$ total assets</td>
<td></td>
</tr>
</tbody>
</table>

Rejected attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Ratio</th>
<th>Rejected attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_6$</td>
<td>working capital $\div$ sales</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Fig. 1. Samples distribution for selected 5 attributes

Fig. 2. Classification error vs number of epochs for different values of learning ratio (Gradient Descent Error Backpropagation)
TABLE II
THE ERRORS FOR CLASSIFICATION DONE BY UKF TAUGHT NETWORKS

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Unscented Kalman Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of neurons</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>29.05%</td>
</tr>
<tr>
<td>3</td>
<td>23.65%</td>
</tr>
<tr>
<td>4</td>
<td>23.65%</td>
</tr>
<tr>
<td>5</td>
<td>25.31%</td>
</tr>
</tbody>
</table>

TABLE III
THE ERRORS FOR CLASSIFICATION DONE BY SRCKF TAUGHT NETWORKS

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Sigma Points Kalman Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of neurons</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>29.88%</td>
</tr>
<tr>
<td>3</td>
<td>27.80%</td>
</tr>
<tr>
<td>4</td>
<td>26.14%</td>
</tr>
<tr>
<td>5</td>
<td>24.07%</td>
</tr>
</tbody>
</table>

TABLE IV
THE ERRORS FOR CLASSIFICATION DONE BY EB TAUGHT NETWORKS

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Backpropagation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of neurons</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>28.22%</td>
</tr>
<tr>
<td>3</td>
<td>27.39%</td>
</tr>
<tr>
<td>4</td>
<td>29.46%</td>
</tr>
<tr>
<td>5</td>
<td>29.04%</td>
</tr>
</tbody>
</table>

Fig. 3. The classification error vs number of iteration for 5, 10 and 15 neurons in the hidden layer (UKF algorithm)

Fig. 4. The classification error vs number of iteration for 5, 10 and 15 neurons in the hidden layer (SPKF algorithm)
observed that a large number of neurons in the hidden layer did not reflect their accuracy. This proves that network architecture must be tailored to the problem and the usage of oversized networks only increases numeric calculations.

Summarising, the results of classification were better for both UKF and SRCDKF than for Error Backpropagation. However, Kalman filters required adjustments of their parameters (like noise matrices or scaling parameters). In our opinion it is their disadvantage, while they are generally superior to Error Backpropagation.

V. CONCLUSIONS

The nonlinear Kalman filtering is a family of algorithms too rarely used in Artificial Intelligence. These algorithms, expanded by modern nonlinear versions of filters, may be efficiently used in different tasks. In this paper we have presented an application of nonlinear Kalman filters as learning algorithms for neural networks. The results of performed algorithms prove that Kalman learners offer high accuracy for networks taught in the presence of strong noise. Both tested algorithms i.e. UKF or SRCDKF achieved better results in presented experiments than Error Backpropagation and their accuracy was almost equal. It must be noted that Backpropagation required less adjustments of parameters and was easier to apply. However, in our opinion better accuracy of networks taught by Kalman filters compensated this. Therefore, as demand on better AI methods is increasing Kalman filters offer great potential.

In the future research we plan to examine if a set of suboptimal filtering parameters could be defined, that would make filtering applications easier. We also plan to conduct a research over the usage of nonlinear Kalman filtering for other types of neural networks.

REFERENCES