Performance evaluation of feedforward neural networks for modeling a vehicle to pole central collision

WITOLD PAWLUS
witolp09@student.uia.no

KJELL G. ROBBERSMYR
kjell.g.robbersmyr@uiu.no

HAMID REZA KARIMI
hamid.r.karimi@uia.no

University of Agder, Faculty of Engineering and Science, Department of Engineering
Serviceboks 509, N-4898 Grimstad
NORWAY

Abstract: Artificial Neural Networks (ANNs) have strong potential in modeling nonlinear systems. This paper presents application of a feedforward neural network which utilizes back-propagation learning algorithm, in the area of modeling a vehicle to pole central collision. Kinematics of a typical mid-size vehicle impacting a rigid pole is reproduced by the means of neural networks approach. Firstly, a network is trained with the appropriate data set (acceleration, velocity, and displacement) and subsequently it is tested and simulated. We also provide a comparison concerning the efficiency and performance of each ANN created in this research. It is judged which of them generates the most satisfactory output in the shortest time.

Key–Words: Feedforward neural network, Back-Propagation Algorithm, vehicle crash modeling

1 Introduction

Currently, lumped parameter modeling (LPM) and finite element method (FEM) are the most popular analytical tools in modeling the crash performance of an automobile [1]-[9]. The major advantage of a FEM model is its capability to represent geometrical and material details of the structure. The major disadvantage of FE method is its cost and the fact that it is time-consuming. To obtain good correlation of a FEM simulation with test measurements, extensive representation of the major mechanisms in the crash event is required. This increases costs and the time required for modeling and analysis. In [10] a bogie instead of a real car was modeled in a software and its behavior was compared to the real experiment’s results. In [11] the multibody occupant model was constructed and its response for the crash pulse was compared with the full-scale FE model (modeled in LS-DYNA3D), proving that such a simplification is justified. [12]-[14] illustrate how the complicated, complete mesh model of a car can be further decomposed into less complex arrangements.

Vehicle crash investigation is an area of up-to-date technologies application: they have extremely high potential for creation of vehicle collision dynamic models and their parameters establishment - e.g. in [15] and [16] values of spring stiffnesses and damping coefficients for lumped parameter models (LPM) were determined by the use of radial basis artificial neural network (RBFN) and the responses generated by such models were compared with the ones obtained via analytical solutions. Results confirmed usefulness of this method - correlation with the reference experimental car’s behavior was good.

Contributions of this paper are as follows. We present the evaluation of the neural network simulation results with the full scale experimental data obtained from the crash test elaborated in [17]. Thanks to choosing the appropriate neural network’s structure we are able to reproduce the car’s kinematics during the crash event (acceleration, velocity, and displacement). The training and simulation stages were performed for the feedforward ANN (artificial neural network) with the back-propagation learning algorithm.

2 Experimental setup description

In this study there are presented results of modeling vehicle to pole collision - schematic representation of the experiment is presented in Fig. 1. The mid-size car impacted the obstacle with the velocity of 35 km/h. Data acquisition was based on high-speed and normal-speed video recordings as well as acceleration measurements - see Fig. 2.

2.1 Crash pulse analysis

Acceleration measurements and its integrals (velocity and displacement) are illustrated in Fig. 3. They constitute complete vehicle’s kinematics during the colli-
From the crash pulse analysis we obtain the data listed in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial impact velocity $V$ [km/h]</td>
<td>35</td>
</tr>
<tr>
<td>Rebound velocity $V'$ [km/h]</td>
<td>7</td>
</tr>
<tr>
<td>Maximum dynamic crush $d_c$ [cm]</td>
<td>52</td>
</tr>
<tr>
<td>Time when it occurs $t_m$ [ms]</td>
<td>76</td>
</tr>
<tr>
<td>Permanent deformation $d_p$ [cm]</td>
<td>50</td>
</tr>
</tbody>
</table>

### 3 Feedforward neural network

According to [18]-[21] an ANN (Artificial Neural Network) is a network of single neurons joined together by synaptic connections. Fig. 4 shows an exemplary case of a three-layer feedforward neural network. It consists of a three neuron input layer, a two neuron output layer and a four neuron intermediate layer, called a hidden layer. It is a fully connected multilayer network because all neurons in a particular layer are fully connected to all neurons in the subsequent layer. Please note that each neuron consists of a weighted summer and an activation function.

The activation function which we are going to use is a sigmoid function. We do so because sigmoid function is differentiable and monotonic which allows us to use it in the Back-Propagation Algorithm in the network learning stage.

#### 3.1 Network learning - Back-Propagation Algorithm

Back-Propagation Algorithm (BPA) is a method of neural networks learning known as supervised learning. All the input possibilities are associated with the desired outputs. Weights are adjusted in such a way that the error between the actual and desired outputs reaches some given minimum value [18]. BPA is one of the most common forms of training techniques. It takes advantage of gradient-descent optimization method which is also known as delta rule when applied to feedforward networks. In another words, we may say that feedforward network with the Back-Propagation Algorithm is called a Multilayer Perceptron (MLP).

The network training (iterative process) can be characterized as follows. In the first learning step the input signals (time values) are passed through the network to determine the outputs from each neuron in each network layer and to determine the initial values of weights coefficients. Then, the actual output signals from the network (estimated acceleration) are compared with the desired output values (target values - reference acceleration). This measured real crash pulse has been randomly divided into three parts in the following way: **Training data:** 50% - the network is adjusted according to its error; **Validation data:** 25 % - it measures network generalization and
stops the training when the generalization does not improve: Testing data: 25 % - it provides an independent measure of network performance during and after the training. The difference between target values and network outputs (error signal) is subsequently propagated back to all the neurons - the weights remain unchanged in this stage. They can be modified only when the error signal for each neuron is computed (to the previous weights values we add a product of this error, the input signal to a considered neuron, the derivative from the activation function, and the coefficient referred to as a learning rate). The simulation stage works in the similar way, the only difference is that the network structure is already established and we determine the accuracy of the created network for prediction of another target values, i.e. vehicle’s velocity and displacement signals. The input signal to the network is still the time vector (which corresponds to the crash duration period). Afterwards, using a new set of inputs, information is passed through the network again (using the new weights) and errors at the output layer are computed. The process is repeated until [18]: 1. The performance index J reaches an acceptable low value; 2. A maximum iteration count (number of epochs) has been exceeded; 3. A training-time period has been exceeded.

Below we present only the major equations which govern the BPA [18]. The notation used is as follows: $s_j$ - weighted sum of signals for a single neuron, $N$ - total number of inputs, $w_{ji}$ - weight value for $j$th neuron in a particular layer and $i$th input, $x_i$ - input to a neuron, $b_j$ - bias for a neuron, $y_j$ - particular neuron’s output, $\eta$ - learning rate, $\delta_j$ - error computed for a given neuron, $J$ - performance index (cost function), $M$ - total number of neurons, $l$ - particular layer’s number, $k$ and $T$ - coefficients used to designate current and previous e.g. weights values.

Single neuron summation:

$$s_j = \sum_{i=1}^{N} w_{ji}x_i + b_j.$$  

(1)

Sigmoid activation function:

$$f(s_j) = \frac{1}{1 + e^{-s_j}}.$$  

(2)

Delta rule:

$$\Delta w_{ji}(kT) = \eta \delta_j x_i.$$  

(3)

New weight:

$$w_{ji}(kT) = w_{ji}((k - 1)T) + \Delta w_{ji}(kT).$$  

(4)

Output layer:

$$\delta_j = y_j(1 - y_j)(d_j - y_j)$$  

(5)

$$J = \frac{1}{2}\sum_{j=1}^{M}(d_j - y_j)^2.$$  

(6)

Other layers:

$$[\delta_{j}]_l = [y_j(1 - y_j)]_l\left(\sum_{j=1}^{N} w_{ji}\delta_j\right)_{l+1}.$$  

(7)

Having the equations listed above, we proceed to the networks creation and simulation.

3.2 Network structure

The aim of this work is to design such a neural network, which simulated will give us the response as similar to the reference signals (i.e. car’s kinematics during a crash - see Fig. 3) as possible. The network should have a simple structure and the number of iterations in the learning stage (number of epochs) should not be too large. Since we employ an ANN in the area of time-domain signals, it is noting (as it will be shown later) that to reproduce plots which are not complex and smooth (e.g. the displacement plot from Fig. 3) it is enough to use simpler neural network’s structure than to reproduce more complex time responses (e.g. the acceleration plot from Fig. 3). After the trial and error method it was found that the following structure is sufficient to obtain the desired ANN simulation’s outcome (displacement and velocity plots): 1 input layer - input signal consisting of 1751 samples (crush time interval lasts 175 ms and the sampling rate is equal to 10 000 Hz); 1 hidden layer - 25 neurons with sigmoid activation function; 1 output layer - 1 neuron with linear activation function; number of iterations - to make the simulation stage efficient, apart from the performance goal which is set to be zero, simultaneously we provide the maximum number of epochs which is equal to 50.

4 Simulation results

One needs to keep in mind that in the training stage there should be used the appropriate corresponding data sets (input to the network is time, target value is e.g. reference car’s crush for the displacement simulation, reference car’s velocity changes for the velocity simulation and so on). See Fig. 5, Fig. 7, and Fig. 9 to observe the simulation’s outcome for the displacement, velocity, and acceleration, respectively. In Fig. 6, Fig. 8, and Fig. 10 please find the performance plots for the corresponding simulation runs.

When it comes to the network’s creation for the vehicle’s acceleration reproduction, it turned out that
the previously mentioned network’s structure is not sufficient. Therefore in the trial and error process it was verified, that to make the network closely follow the reference acceleration plot, instead of hidden layer with 25 neurons, a layer with 200 neurons should be used. The rest of the structure remains unchanged. For the comparison purposes we present the results for ANN containing 1 hidden layer with 200 neurons and 2 hidden layers each containing 100 neurons. For the summary of the ANNs simulations refer to Table 2.

The outcome of the simulation is satisfactory. The predicted displacement is the same as the original car’s crush. The number of epochs specified at the beginning of the training stage was not even reached - after 24 iterations the network achieved the performance goal equal to zero and because of such a low MSE value the network’s output is identical with the target value (real car’s displacement). The training stage of the network for the velocity simulation was completed after the specified number of epochs. It means that if the only constraint was the performance goal, the learning procedure would still be continued. However, the MSE reached after 50 iteration was also low - for that reason the predicted network’s output still closely resembles the original car’s velocity changes. Both of the above learning operations were quick, the limit time of 1 second was not exceeded. Interesting results were obtained for the acceleration prediction. As we see, a feedforward network with one hidden layer containing a given number of neurons produces better overall results than a similar network containing the same number of neurons but distributed on two hidden layers. For both mentioned networks there occurred no overtraining - performance plots observed in Fig. 10 settle on a steady value of MSE, i.e. MSE does not increase for the increasing number of epochs [22]. The reason for the better performance of the one-layer network consisting of 200 neurons than two-layer network each layer consisting of 100 neurons is that for the second case there is more weights to be trained therefore more neurons would be needed to achieve the same results accuracy as for the first case. The simulation results of the above firstly mentioned ANN are more effective: the value of MSE is still relatively low after 50 iterations, the total computation time stays in the acceptable limit and the generated network’s output follows the measured acceleration crash pulse. Because of the fact that for two-layer network used for the acceleration prediction there exist more neural connections and the network is fully connected (each neuron in 1st layer is connected with every neuron in the 2nd layer), such long computation time was reached.
5 Conclusions

Feedforward neural networks with Back-Propagation Algorithm are suitable for reproducing car’s kinematics during a collision. Satisfactory results can be obtained even if the network’s structure is simple. However, it is of paramount importance to select the appropriate network’s arrangement. As we see from the above results, the more complicated characteristic we would like to achieve as a network’s output, the more neurons such a network should have. The matter of efficiency also plays a crucial role in neural networks applications - the faster a network is, the better its overall rating (of course under the condition of a satisfactory output). It is also noting that at some point, the increasing number of epochs (iterations in the learning stage) does not improve the results whatsoever. In such a case, the only way to obtain network’s better output is to redesign the network’s structure.

The area which may be further explored is the application of neural networks approach to represent different collision types. It is advisable to verify if a network trained for one set of data (e.g. acceleration measured during a vehicle to barrier low speed central collision) is capable of reproducing another collision type (e.g. prediction of the acceleration in a vehicle to vehicle high speed central impact). Another suggestion is to compare performance of a network trained by using another algorithm or different approximation method with the results obtained in this paper.

References:


