Comprehensible artificial intelligence-based models for raveling of porous asphalt

M. Miradi, A.A.A. Molenaar
Road and Railway Engineering, Delft University of Technology

Abstract

Among a large number of existing artificial intelligence (AI)-based techniques, artificial neural network (ANN) is one of the most widely used techniques, which has been applied to many road engineering problems. They are especially useful as predictive models because they do not require prior knowledge of the input data distribution and have shown high prediction performance. However, despite their high degree of accuracy, these AI models are difficult to interpret. For many problems, it is desirable to extract knowledge in a comprehensible manner so that the users can gain a better understanding of the solution. The traditional if-then rules are usually the most understandable ones. In the Netherlands, the surface damage of porous asphalt, being raveling, draws a lot of attention because porous asphalt (PA) has been applied to more than 70% of Dutch highways. A better understanding of this problem can lead to lower maintenance cost. Therefore, this paper applies two comprehensive modeling techniques to the problem of raveling. The first technique is rule extraction from function approximating neural networks (REFANN), which extract rules from ANN. The second one is regression trees. These modeling techniques were applied to the data of 72 PA sections obtained from SHRP-NL.
1. Introduction

During the last decade interest in ANN has grown significantly, owing to their ability to represent nonlinear relationships that are difficult to model by means of other computational methods. In ANN, information is encoded among the various connection weights in a distributed manner [1]. The distributivity contributes to profound learning capabilities, as the individual computing elements in the network are capable of adjusting their connections to carry out the best possible fitting. While this feature enhances learning, it makes it almost impossible to come up with a reasonable interpretation of the overall structure of the network in terms of easily understood logical constructs (like if-then statement, frames, etc.) [2]-[4]. Therefore, it is very difficult to understand how an ANN has solved a problem.

This study investigates the surface damage of Dutch porous asphalt, a top layer that is used for noise reducing purposes on more than 70% of the Dutch motorway system [5]. Although road users enjoy from many advantages of PA such as noise reduction and driving comfort, road authorities struggle with the fact that surface damage, namely raveling, appears rather short after construction leading to a short lifespan of PA wearing courses. It has been shown that the pavement life can be as short as four years. A short lifespan leads to significant maintenance costs [6], [7]. Because of the high maintenance costs an increase in the pavement life is highly desirable. To achieve this goal, better knowledge about the factors that control the PA lifespan is required. It is expected that regression trees and rule extraction from ANN will provide this knowledge in a comprehensible way.

The existing literature shows that although many researchers have actively been busy with extracting symbolic rules from trained ANN for classification problems [8]-[13], only a few methods have been developed to extract rules for regression problems [14], [15]. The rules generated from ANN are expressed as symbolic rules of the form: if condition of inputs (x) is satisfied, then output (y) is f(x). f(x) is a function of x where x is the input parameters obtained from data. This type of rules is acceptable because of their similarity to traditional logical sentences. However, rule extraction is an extremely difficult task for arbitrarily configured networks, and some approximation and simplification in this process can lead to wrong conclusions, and to meaningless rule extraction.

In this paper, the results of applying REFANN [15] are presented. REFANN is an algorithm that extracts rules for regression problems. An implementation of this algorithm, using C++ programming language, was developed. The data from the project Strategic Research Program Netherlands (SHRP-NL), which was carried out between 1990 and 2000, were used. The goal is to develop comprehensible model which connects input parameters being PA mixture/construction, traffic, and climate factors to PA raveling using REFANN. To reduce the number of rules and to simplify the rule conditions, before applying REFANN, redundant hidden units and irrelevant input parameters are removed by a pruning method called NN pruning for function approximation (N2PFA). Next to that, rules are also generated using regression trees. For a detailed description of regression trees, the reader is referred to [16].

The remainder of the paper is organized as follows: Section 2 provides a brief review of porous asphalt. Section 3 reviews the existing works of rule extraction. Section 4 describes the rule extraction algorithm in detail. Section 5 explains the data selection. In section 6, the results are analyzed and discussed. The conclusions are presented in section 7.

2. Porous asphalt (PA)

In the Netherlands, PA was introduced in the early eighties. In 1987, it was decided to start
applying PA courses on a larger scale. Three years later, the decision was made that the entire PA should be applied to the main motorway network (3,200 km). On heavily trafficked roads PA is applied in a thick layer of 50 mm [17]. Currently more than 70% of the Dutch motorways are covered with a PA top layer [5].

The dominant damage type of PA is raveling, which is loss of aggregate from the pavement surface. Due to raveling, the average service life is about 8-12 years but in narrow curves raveling is already a problem after four years. Padmos [18] reports that raveling is the cause for maintenance of the top layer in about 76% of the cases. Raveling is caused by different factors such as traffic, climate, design and construction imperfections. This research is focused on finding the relation between raveling and the mentioned factors to increase the service life of PA as long as possible.

3. Related works and motivation

Neural network rule extraction algorithms differ in expressive power, quality, translucency and complexity. Andrew et al. [19] translate the quality of neural rules as accuracy, comparing to the underlying network, comprehensibility and consistency of these rules. Towell and Shavlik [20] proposed a method called subset considering all possible subsets of incoming weights of hidden and output neurons to find all combinations greater than the threshold. The problem of their method was the exponentially growing number of possible conjunctive propositional rules (Number of subsets was $2^{\text{Number input}}$). McMillan et al. [21] and Alexander et al. [22] considered RuleNet method to find $M$ of $N$ rules and propositional rules using weight templates exploring large spaces of candidate rules. However, RuleNet works only for discrete-valued features. Thrun [23] used validity interval analysis (VIA) to extract rules by mapping the inputs directly to the outputs. Although VIA is a global method, it produces numerous rules which are too specific. Craven and Shavlik [24] proposed TREPAN which uses a decision tree instead of rules and it treats neural network as “oracle” answering queries. TREPAN works better than other inductive algorithms, but works with classifiers not approximators. Geczy and Usui [25] extracted logical functions performed by the neural network. In the method proposed by Tsukimoto [26], network function was approximated by lower order logical polynomials, which delivered poor results.

Most existing published reports have focused on extracting symbolic rules for solving classification problems. For example, the MofN algorithm [27] and GDS algorithm [28] extract MofN rules; BRAINNE [29], Bio-RE, Partial-RE, Full-RE [30], NeuroRule [31] and GLARE [32] generate disjunctive normal form (DNF) rules; and FERNN [33] extracts either MofN or DNF rules depending on which kind of rules is more appropriate.
1) Split the data into 3 subsets: training, cross-validation, and test sets.
2) Choose a network with a large number of hidden units and train this network to minimize the error function.
3) Let \( \text{trnerr} = \text{error on the training data set, valerr = error on the cross-validation data set and } \text{tmnbest} = \text{trnerr}, \text{valbest} = \text{valerr}, \text{ermx} = \max\{\text{trnbest, valbest}\} \).
4) Remove redundant hidden units:
   4.1) Let \( \text{trnerr}_m \) and \( \text{valerr}_m \) be the errors on training data set and cross validation data set with all input connections to hidden unit \( m \) set to 0, \( m = 1, 2, \ldots, H \).
   4.2) Let \( \text{trnerr}_h \) be the smallest among all \( \text{trnerr}_m \).
   4.3) Retrain network with all connections from input units to hidden unit \( h \) set to 0.
   4.4) Let \( \text{trnerr} \) and \( \text{valerr} \) be the errors of the retrained network on training and cross-validation data respectively.
   4.5) If \( \text{trnerr} \leq (1 - \alpha)\text{ermx} \) and \( \text{valerr} \leq (1 + \alpha)\text{ermx} \), then
       Remove hidden unit \( h \).
       Set \( \text{trnbest} = \min\{\text{trnerr}, \text{trnbest}\} \), \( \text{valbest} = \min\{\text{valerr}, \text{valbest}\} \) and \( \text{ermx} = \max\{\text{trnbest, valbest}\} \).
       Set \( H = H - 1 \) and go to Step 4.1.
5) Remove irrelevant inputs:
   5.1) Let \( \text{trnerr}_l \) and \( \text{valerr}_l \) be the errors on training data set and cross validation data set with all input connections from input unit to the remaining hidden units set to 0, \( l = 1, 2, \ldots, N \).
   5.2) Let \( \text{trnerr}_n \) be the smallest among all \( \text{trnerr}_l \)’s.
   5.3) Retrain the network with all connections from the input unit \( n \) to the hidden units set to 0.
   5.4) Let \( p \) and \( q \) be the errors of the retrained network on the training data and on the cross validation data, respectively.
   5.5) If \( \text{trnerr} \leq (1 - \alpha)\text{ermx} \) and \( \text{valerr} \leq (1 + \alpha)\text{ermx} \), then
       Remove input unit \( n \).
       Set \( \text{trnbest} = \min\{\text{trnerr}, \text{trnbest}\} \), \( \text{valbest} = \min\{\text{valerr}, \text{valbest}\} \) and \( \text{ermx} = \max\{\text{trnbest, valbest}\} \).
       Set \( N = N - 1 \) and go to Step 5.1.

Fig. 1. The algorithm of Neural Network pruning for function approximation (N2PFA)

Few methods have been devised to extract rules from trained ANNs for regression. ANN-DT [34] is one such algorithm which is capable of extracting rules from function approximating networks. The algorithm regards ANNs as “black boxes.” It produces decision trees based on the networks’ inputs and the corresponding outputs without analyzing the hidden units’ activation values and the connection weights of the networks.

4. Rule extraction

Neural network training and pruning Algorithm

Pruning algorithms, aim to remove redundant units from the initial network structure and try to increase the generalization ability while maintaining high approximation accuracy. In this section, the Neural Network pruning for function approximation (N2PFA) [15] algorithm is described. This training and pruning approach is suitable for networks with a single hidden layer. The available data samples are first divided into 3 subsets: the training, the cross-validation and the test sets. The samples in the cross-validation set are used to determine when the pruning should be terminated, while the samples in the test set are used to measure the accuracy of the method and to compare the results to those obtained by other methods. The steps of N2PFA are given in Fig. 1 [15]. Typically, at the start of the algorithm when there are still many hidden units in
the network, the value of trnerr will be much smaller than valerr. The value of trnerr increases as more and more units are removed. As the network is approaching its optimal structure, the value of valerr will decrease. ermax is used to determine if a unit should be removed instead of trnbest alone or valbest alone. The value of valbest can be small but the network does not predict the data in the test set well. This could be caused by the small number of samples available for cross-validation or the uneven distribution of the data in the training and cross-validation sets. ermax is assigned the larger of the two values trnbest and valbest as it is tried to remove as many redundant network units as possible. Finally, the parameter $\alpha$ is introduced to control the likelihood that a unit will be removed [15]. With a larger value of $\alpha$, units are more likely to be removed. However, the accuracy of the resulting network on the test data set may deteriorate. Extensive experiments have been conducted to find a value for this parameter that works well for our problems.

Approximating activation function

Since the hidden unit activation function $h(x)$ is the hyperbolic tangent function $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, antisymmetric activation function, it is sufficient to illustrate how the approximation is done just for the positive values of $x$. The function $h(x)$ can be approximated by a piecewise linear function as follows. Suppose that the input $x$ ranges from 0 to $x_m$. A simple and convenient approximation of $h(x)$ is to over-estimate it by a piecewise linear function $L(x)$ as shown in Fig. 2.

![Fig. 2. Tangh(x) function (solid curve) for $x \in [0,x_m]$ is approximated by linear function (dashed lines).](image)

To ensure that $L(x)$ is larger than $h(x)$ everywhere between 0 to $x_m$, the line on the left should intersect the coordinate $(0, 0)$ with a gradient of $h'(0) = 1$, and the line on the right should intersect the coordinate $(x_m, h(x_m))$ with a gradient of $h'(x_m) = 1 - h^2(x_m)$. Thus, $L(x)$ can be written as (1).

$$L(x) = \begin{cases} x & \text{if } 0 \leq x \leq x_0 \\ h'(x)(x-x_m) + h(x_m) & \text{if } x > x_0 \end{cases} \quad (1)$$

The point of intersection $x_0$ of the two lines is given by (2).

$$x_0 = \frac{h(x_m) - x_m h'(x_m)}{h^2(x_m)} \quad (2)$$

Rule generation

A single rule cannot approximate the nonlinear mapping performed by the networks well. With dividing the input space of the data into sub-regions more rules are produced and each rule can approximate the network output more accurately. However, a large number of rules do not provide useful knowledge to the road authorities. Hence, a balance must be achieved between rule accuracy and simplicity [15]. Online back-propagation was used instead of Quasi-Newton which is proposed by Setiono et al. [15]. REFANN generates rules from a neural network as follows:
1) Train and prune a network with one hidden layer and one output unit.
2) For each hidden unit $i = 1, 2, \ldots, H$:
   2.1) Determine $x_{im}$ from the training samples and compute $x_{i0}$ according to (2).
   2.2) Define positive part of a 3-piece approximating linear function $L_i(x)$ as (1).
3) Using pair of points 0 and $x_{i0}$ of function $L_i(x)$ divide the input space into $2^H$ sub-regions.
4) For each non-empty sub-region, generate a rule as follows:
   4.1) Define a linear equation that approximates the network’s output for input sample $p$ in this sub-region as the consequence of the extracted rule (3) and (4).
   
   \[
   \hat{y}_p = \sum_{i=1}^{H} v_i L_i(s_{ip})
   \]  
   \[
   s_{ip} = \sum_{j=1}^{N} w_{ij} I_{jp}
   \]  
   Where $v_i$ is the weight between a hidden unit and an output unit and $w_{ij}$ is the weight between an input unit and a hidden unit.
4.2) Generate the rule condition: ($C_1$ and $C_2$ and $\cdots$ $C_H$), where $C_i$ is either $0 \leq s_{ip} \leq x_{i0}$, or $s_{ip} > x_{i0}$.

5. **Selection**

In this study, the database provided by the SHRP-NL research program was used. SHRP-NL was started in 1990 as a downscaled version of the American project and ended ten years later at 2000. The SHRP-NL data set contains a total of 240 test sections. Every test section is 300 m long with two reserved areas of 25 m at the beginning and at the end of the test section that are used to drill cores of the pavement structure. Each section is divided into three subsections of 100 m [6]. This study used the 72 available PA subsections.

<table>
<thead>
<tr>
<th>Severity</th>
<th>Percentage of stone loss per m²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>6-10</td>
</tr>
<tr>
<td>Moderate</td>
<td>11-20</td>
</tr>
<tr>
<td>Severe</td>
<td>&gt; 20</td>
</tr>
</tbody>
</table>

**Condition data**
The condition surveys consisted of very detailed surveys by well trained inspectors who determined the percentage of light, moderate and severe damage (Table 1). In order to summarize these three values in one value that can be used for model development, an overall damage, called $Meq$, was calculated using $Meq = 0.25$ Light + Moderate + 5 Severe, where $Meq$ is the amount of equivalent moderate damage and L, M, and S are, the amount of light, moderate, and severe damage [% of total area], respectively.

**Mixture, traffic and climate data**
The mixture composition data were used as retrieved from the available cores. From each road section, 6 cores have been taken. For each core, information is available on gradation, density, bitumen content, void content, type of stone used. For the void content, the coefficient of variation was taken as an input variable because calculations showed that void content showed a significant amount of variation (greater than 10%). It was expected that the void content and the large variation thereof could explain the damage as observed. Four stone
types were used in the PA mixtures being greywacke, crushed siliceous river gravel, porphyry, greywacke/ Greyquartsite. With respect to the gradation it was concluded that the gradation could be characterized by means of the percentage of fine aggregate and percentage of coarse aggregate and D\(_{50}\) and Cu of coarse fraction where D\(_{50}\) is sieve size through which 50% of the coarse material passes, Cu is the coefficient of uniformity (D\(_{60}\)/D\(_{10}\)), percentage of fine aggregate is the percentage of material passing the 2 mm sieve, percentage of coarse aggregate is the percentage of material which remains on 2 mm sieve.

The SHRP-NL database contains information about the average daily traffic and the growth rate. In some cases, information on the growth rate was not available and then a value of 5% was adopted. The information on the growth rate was interpreted as follows: if in 1986 the number of vehicles was 100 and the growth rate between 1986 and 1993 was 6%, the number of vehicles in 1993 was 106. Based on this information, the cumulative amount of traffic until the dates of the visual condition surveys (1993) was calculated. The following equations ((5) to (7)) are used for this.

\[
GPy = (GP - 1)^\frac{1}{1993-1986}
\]

\[
NMVCD = \frac{NMV}{GPy^{1993-CD}}
\]

\[
CAT5 = GPy \times NMVCD \times 5
\]

Where GP is the percentage of traffic growth between years 1986 and 1993, GPy is the percentage of traffic growth per year, NMV is the number of motor vehicles for each 24 hours on the test section, CD is year of construction, NMVCD is the number of motor vehicles for each 24 hours on the test section in the year of construction, and CAT5 is cumulative amount of traffic 5 years after construction. GP, NMV and CD were obtained from the SHRP-NL database while GPy, NMVCD, and CAT5 were calculated by (5) to (7) [6].

The SHRP-NL database also contains information about the number of days per year at which the minimum temperature was 0°C or lower and the number of days at which the maximum temperature was 25°C or higher. This was used to calculate the cumulative number of cold days and the cumulative amount of warm days.

6. Results and discussion

As listed in Table 2, the data set has 12 input parameters and one output parameter. The goal is to predict the equivalent moderate raveling (Meq) five years after construction. The values of all 12 input parameters were scaled to the interval [-1, 1], while the output (Meq) was scaled so that it ranged in the interval [0, 1].

The value of the parameter \(\alpha\) was set to 0.02 during the first stage of the algorithm when hidden units were checked for removal, and it was set to 0 in the second stage when input units were checked. The value was larger in the first stage, as the initial networks contained more than the required number of hidden units and tried to trim the networks as much as possible.

One of the trained networks was selected to illustrate in details how the rules were extracted by REFANN. This network was initialized with 12 hidden units. Number 12 was found to be large enough to fit the training data, but not too large that it would unnecessarily slow down the network training. The number of training, cross validation and test samples were 58, seven, and seven, respectively. After N2PFA pruning, only three hidden unit (hidden units 3, 4, and 6) remained. Except for the connection from input Mixture Density, all other input connections were still present in the pruned network.
Table 2. Input and output parameters

<table>
<thead>
<tr>
<th>Input/Condition parameter</th>
<th>Output/Decision parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Mixture Density (MD)</td>
<td>1. Meq(raveling) 5 years after construction (Meq5)</td>
</tr>
<tr>
<td>2. Bitumen Percentage (BIT)</td>
<td></td>
</tr>
<tr>
<td>3. Void Content (VC)</td>
<td></td>
</tr>
<tr>
<td>4. Coefficient of Variation of Void Content (COVC)</td>
<td></td>
</tr>
<tr>
<td>5. Stone Type (ST)</td>
<td></td>
</tr>
<tr>
<td>6. Percentage of Fine (FINE)</td>
<td></td>
</tr>
<tr>
<td>7. Percentage of Coarse (COARS)</td>
<td></td>
</tr>
<tr>
<td>8. CU</td>
<td></td>
</tr>
<tr>
<td>9. D50</td>
<td></td>
</tr>
<tr>
<td>10. Warm days (WD)</td>
<td></td>
</tr>
<tr>
<td>11. Cold Days (CD)</td>
<td></td>
</tr>
<tr>
<td>12. Cumulative traffic intensity of the first 5 years after construction (CAT5)</td>
<td></td>
</tr>
</tbody>
</table>

After multiplying the coefficients of \( L_i(s_y) \) by the connection weight value from the hidden unit to the output unit (see (3)) and rescaling the input and output data back into their original values, we obtained (8) for hidden neuron unit 3.

\[
L(x) = \begin{cases} 
  x_3 & \text{if } 0 \leq x_3 \leq 0.9963 \\
  0.001x_3 - 0.9581 & \text{if } x_3 > 0.9963
\end{cases}
\]

The condition and the output generated from the nonempty input region for \( x_3 \) is:

Rule 1. If \( 0 \leq 0.94\text{Bitumen} - 0.80\text{Void_Content} + 0.49\%\text{Coarse} - 0.45\text{CU} + 2.03\text{D50} - 1.10\text{ColdDays} + 0.74\text{Cu_Traffic_5} \leq 0.9963 \)

Then \( Y_{h3} = -2.42\text{Bitumen} + 2.07\text{Void_Content} -1.27\%\text{Coarse} + 1.14\text{CU} -5.22\text{D50} -2.83\text{ColdDays} -1.89\text{Cu_Traffic_5}. \)

Similarly, the hyperbolic tangent function at hidden units 4 and 6 are approximated by (9) and (10).

\[
L(x) = \begin{cases} 
  x_4 & \text{if } 0 \leq x_4 \leq 0.9995 \\
  0.004x_4 + 0.9632 & \text{if } x_4 > 0.9995
\end{cases}
\]

\[
L(x) = \begin{cases} 
  x_6 & \text{if } 0 \leq x_6 \leq 0.9991 \\
  0.003x_6 - 0.9733 & \text{if } x_6 > 0.9991
\end{cases}
\]

The condition and the output for nonempty input regions of hidden units 4 and 6 are:

Rule 2. If \( 0 \leq -3.39\text{VC} -1.60\text{COVC} + 4.06\text{ST} -2.22\text{CU} + 0.82\text{D50} + 2.64\text{WD} -1.05\text{CAT5} \leq 0.9995 \)

Then \( Y_{h4} = 75.75\text{VC} + 35.74\text{COVC} -90.71\text{ST} + 49.59\text{CU} -18.27\text{D50} -59.07\text{WD} 23.37\text{CAT5}. \)

Rule 3. If \( 0 \leq -2.10\text{BIT} - 1.07\text{VC} -1.95\text{COVC} + 0.70\text{ST} + 0.73\%\text{Coarse} -1.96\text{CU} + 1.60\text{D50} + 1.62\text{WD} -1.50\text{CD} -1.22\text{CAT5} \leq 0.9991 \)

Then \( Y_{h6} = 56.20\text{BIT} + 28.69\text{VC} + 51.99\text{COVC} -18.76\text{ST} + 12.97\text{FINE} -19.43\text{COARS} + 52.34\text{CU} -42.80\text{D50} -43.15\text{WD} + 40.20\text{CD} + 32.47\text{CAT5}. \)

It should be noted that the coefficients of the two parallel hyperplanes that divide the input space into the two regions are equal to the weights \( W_{ij} \) from the \( j \)th input unit to the hidden unit. In the rules generated, the coefficient of each variable present in that rule is the indicative of the degree and sign of the correlation of that variable with the output. In real-life problems, however, inputs are correlated, and output(s) needs to be expressed in terms of some form of algebraic expressions of input variables [15]. The proposed technique offers a tradeoff between the number, accuracy, and interpretability of the rules.

Rules were then generated using regression trees. 10-fold cross validation for regression trees resulted in the following combination of rules:

Rule 1. If Bitumen < 3.95 then Meq5 = 10.80.
Rule 2. If Bitumen > 3.9 AND Traffic_intensity>16342 AND CU > 1.8 then Meq5 = 9.7.
Rule 3. If Bitumen > 3.9 AND Traffic_Intensity<28842 AND %Fine <18 and CU < 1.8 then Meq5 = 2.55.

Regression trees used a maximum of four input parameters to fit the data after pruning. REFANN used all parameters except for mixture density. To test the quality of the rules generated by REFANN with the rules generated by regression trees, we calculated the root means square error (RMSE) for both group of rules (see Table 3).

As can be seen in Table 3, the RMSE of the rules generated by regression trees is considerably lower than RMSE of the rules extracted by REFANN. Although REFANN fits the data with more input parameters, it does not give a better model as shown by the first three reported RMSE. The pruned regression tree with less input variables appears to result in a far better fitting of data.

<table>
<thead>
<tr>
<th>Method – Rule Number</th>
<th>Root Means Square Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFANN – Rule 1</td>
<td>38.84</td>
</tr>
<tr>
<td>REFANN – Rule 2</td>
<td>22.56</td>
</tr>
<tr>
<td>REFANN – Rule 3</td>
<td>29.98</td>
</tr>
<tr>
<td>Regression Tree – Rule 1</td>
<td>2.75</td>
</tr>
<tr>
<td>Regression Tree – Rule 2</td>
<td>6.94</td>
</tr>
<tr>
<td>Regression Tree – Rule 3</td>
<td>2.39</td>
</tr>
</tbody>
</table>

REFANN rules are both natural and simple to understand. However, their accuracy seems to be rather low for small datasets with many input parameters. Although Setiono et al. [15] have claimed that REFANN works even for low numbers of data points, our results showed otherwise. One solution to this problem could be to decrease the number of input features by another method before applying N2PFA and REFANN. However, one could argue that we have already done this using the second part of N2PFA (removing irrelevant input parameters). This leaves the question if the pruning algorithm of input parameters does an optimal selection because the only input parameter removed by N2PFA was mixture density. REFANN rule extraction is most useful when it is necessary to check whether what the network has learned is reasonable. Another conclusion which could be drawn is that when the number of data points is not enough, the trained neural network does not learn and approximate well enough.

7. **Conclusions**

Neural Network rule extraction is applied when it is necessary for a system to explain its actions, something a classic neural network normally cannot do. Unfortunately, rule extraction is an extremely difficult task for arbitrarily configured networks, and some approximation and simplification in this process can lead to wrong conclusions, and to meaningless rule extraction. In this paper, N2PFA for neural network pruning and REFANN were applied for extracting of regression rules to the limited data available (72 road sections) to analyze the raveling of PA. These rules were compared to the one achieved by regression trees. According to the calculated RMSE, the rules extracted by REFANN have a much higher error than the rules generated by regression trees. Furthermore, Regression tree fits with a maximum of four input parameters when REFANN uses all except for one. One could conclude that apart from the advantages of N2PFA and REFANN, they need a relative great number of data points to approximate well enough. Regarding dataset, despite the fact that it took ten years to gather the SHRP-NL dataset, the number of data points for raveling is not
great enough to extract meaningful rules with REFANN. Generally, it can be concluded that although predictive ANN models are very powerful, opening these black boxes in the form of rule extraction to reach comprehensibility will result in low performance and therefore they are less suitable for comprehensible modeling.

8. References