CONCEPT AND APPLICATION OF AUTOMATIC PART-RECOGNITION WITH ARTIFICIAL NEURAL NETWORKS FOR FE SIMULATIONS

Spruegel, Tobias C.; Wartzack, Sandro
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

Currently available finite element software is consequently getting more and more user-friendly, and simulation knowledge must be expanded to a growing pool of new and less-experienced users; for example designers. These users need to be assisted when performing simulation tasks. A promising way is to check the FE results for plausibility, but therefore, it is essential to know what kind of parts should be simulated and checked for plausibility. This paper presents a concept for the automatic recognition of standard parts within the FE software, using a predefined Artificial Neural Network (ANN). An existing approach is extended to deal with the issues of arbitrarily oriented parts in CAD (computer-aided design) assemblies of differing size in 3D space. Hence a Principal Component Analysis (PCA), and a spherical detector surface for FE nodes in spherical coordinates is used to gain input parameters for an ANN. In the paper, the concept and an application for certain standard components are demonstrated.

Keywords: Simulation, Design informatics, Artificial Neural Networks (ANN), part-recognition, FE simulation

Contact:
Tobias C. Spruegel
Friedrich-Alexander Universität Erlangen-Nürnberg
Mechanical Engineering
Germany
spruegel@mfk.fau.de

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 20th International Conference on Engineering Design (ICED15), Vol. nn: Title of Volume, Milan, Italy, 27.-30.07.2015

1 INTRODUCTION

The finite element method is, alongside computer-aided design (CAD), one of the most efficient methods for optimizing the quality of engineering tasks (Klein, 2010). Furthermore the currently available finite element (FE) software is getting more and more user-friendly, which can lead to the idea that proper FE simulations can be done by everyone. According to a recent survey (Boucher, 2013), companies struggle to expand simulation knowledge to a growing pool of users. In particular new users, such as designers, are doing more and more FE simulations but do not have the knowledge of experienced computation engineers. Therefore these users need to be assisted when performing simulation tasks. A promising way is to check the results of FE simulations from less experienced users for plausibility. In contrast to validation, plausible results are considered to be apparently, likely valid. For FE plausibility checks it is essential to know what kinds of parts need to be simulated, and how the results of these simulations can be compared with available data; i.e. from previous simulations of similar parts. Obviously, a plausibility check for a M6 through-screw is completely different from the check of a dowel pin with a 20 mm diameter.

This paper presents a concept for the automatic recognition of parts by analysis of the corresponding FE mesh, using a predefined Artificial Neural Network (ANN). ANNs are used because part-recognition is highly sophisticated, and several issues arise when parts need to be identified. Moreover, ANNs are a well-known approach for creating simple and fast approximations (Jin et al., 2001). The concept in this paper has to deal with the following requirements. At first, parts must be detected as quickly as possible, and with very high accuracy within the FE program only on basis of the FE-mesh. Secondly, the parts must be detected only on basis of their geometry: no further semantic information, such as the name of the part, is considered. As neutral CAD formats are usually used for geometry transfer between CAD and FE software semantic information and the modelling history of the part is unavailable within the FEA. Thirdly, parts with any rotation, translation, size and shape within a CAD assembly must be detected with the approach.

Current object recognition is done mainly using imagery, and is often for military applications. An example is the detection of radar signatures, and the recognition of specific objects in these images (Roth, 1990). The approach in this paper is very different. As the parts will be recognized within the FE program, the nodal information can be easily used to detect parts. The described process can be divided roughly into two different tasks: the training and the usage of the ANN (see Figure 1).

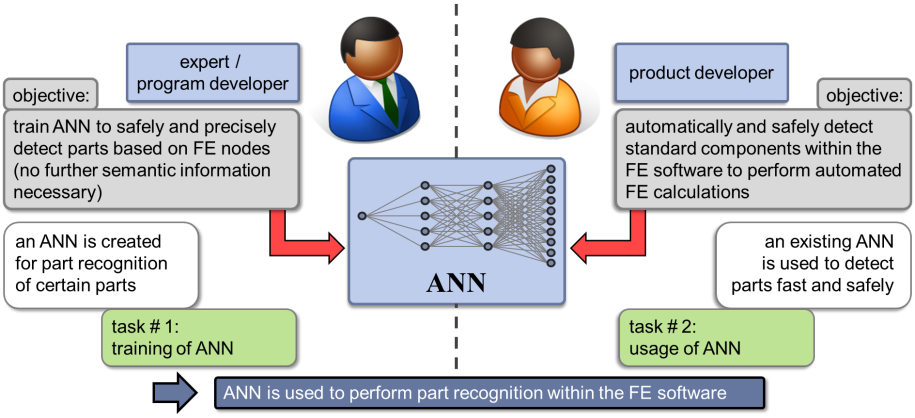

Figure 1. The tasks for part recognition with ANN

The first task is the training of the ANN, which is done by an expert that is familiar with the details of mathematical meta-models, such as ANN. The expert also decides which parts will be trained, and handles the whole process of the creation of the network. Furthermore, the expert implements the trained network in easy-to-use software tools. The product developer, on the other hand, uses the trained network within the FE software for automatically detecting parts. The developer does not necessarily need any knowledge in the field of meta-modelling and only needs to understand how to use the tools given by the expert. In section 2, the state of the art of Artificial Neural Networks (ANN) and Principal Component Analysis (PCA) is explained briefly. This is essential for the explanation of the actual concept for part recognition with an ANN in section 3. The main focus of this paper is on the first task: the training of the ANN by the expert. The preparation of the data and the training is

explained in seven steps. In section 4 the usage of the ANN (task # 2) by the product developer is mentioned briefly.

2 STATE OF THE ART

2.1 Artificial Neural Networks (ANN)

ANNs imitate the information processing capabilities of nervous systems. Dendrites, synapses, cell bodies and axons are the minimum structures that need to be adopted from the biological model by the artificial neurons. Therefore, artificial neurons consist of input channels, a cell body and an output channel. The connection between neurons is associated with weights (Rojas, 1996).
According to Masters (1993), ANNs can be applied in several fields, such as classification, auto-association, time-series prediction, function approximation, and regression.

2.1.1 Basic multilayer architecture of ANN

A multilayer network consists of a set of neurons in two or more layers. The first layer is always the input layer with a fixed number of input neurons and is used to transfer the input parameters from the input layer to the other hidden layers with their neurons. The number of input parameters is identical to the number of neurons in the first layer of the network. Likewise, the number of output parameters is the same as the number of neurons in the output layer. The amount of neurons in the hidden layer, as well as the number of these hidden layers, needs to be chosen individually. An ANN with four input and two output parameters, as well as three neurons on one hidden layer, can be seen in Figure 2.
ANNs can be seen as weighted directed graphs, where the output of a neuron is connected to the input of another neuron with corresponding weights. According to the connection pattern (architecture of the ANN), the networks can be grouped into two categories: feed-forward networks with no loops in the graph; and feedback networks, in which loops can occur. (Jain and Mao, 1996)

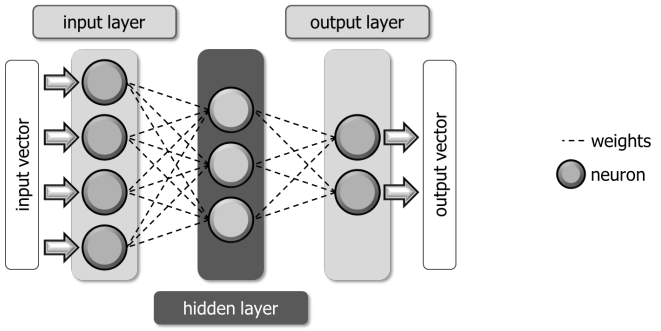

Figure 2. Artificial neural network with input layer, one hidden layer and output layer

According to Masters (1993), it is nearly impossible to specify an effective architecture, given the specifications of a problem. The architecture has to be determined iteratively. Nevertheless, several rules of thumb exist for the number of hidden neurons for a specific problem. For example, the size of the hidden layer should be somewhere between the input layer size and the output layer size (Masters, 1993). The number of hidden nodes is a very important parameter of a neural network. If the number of neurons is too small, the generated network is not sufficiently accurate and underfits the model. If the number of neurons is too large, the network tends to overfit the data, which causes poor generalization on data not used for training (Fletcher et al., 1998). According to Huang and Babri (1998), it is well-known that standard single-hidden layer feed-forward networks, with at most N hidden neurons, can learn N distinct samples with zero error. Xu and Chen (2008) present a novel approach for determining the optimal number of hidden layer neurons for feed-forward neural networks, and their application in Data-Mining.

2.1.2 Training and usage of ANN

Before an ANN can be operated, the weights must be calculated. This is called the “training” of a neural network. According to Masters (1993) and Rojas (1996), three different ways of training can be distinguished:

- Supervised learning: Each training sample consists of completely specified inputs and corresponding output parameters which are used to calculate the weights. Thereby, the desired output of the network can be compared with the calculated values of the network. Consequently, an error can be calculated and the weights are corrected according to the magnitude of the error.
- Unsupervised learning: Several input samples are used to calculate the weights, but no corresponding output parameters are provided to the network. A typical assumption is that each input arises from one of several classes, and that the output of the network is an identifier for the class to which the input sample belongs.
- Reinforcement learning: This method can be seen as a combination of both supervised and unsupervised learning. On one hand, the output of the network is not specified, but on the other hand, the network is told whether the calculated output is a good or bad approximation of the intended value.

After the training, the network can be used (i.e. for function approximation). For this purpose, the input parameters are transferred to the ANN via the input layer. Then the network calculates the corresponding output for the given input parameters (usage of the ANN). The process of training and use of an ANN can be seen in Figure 3. (Spruegel and Wartzack, 2014a)

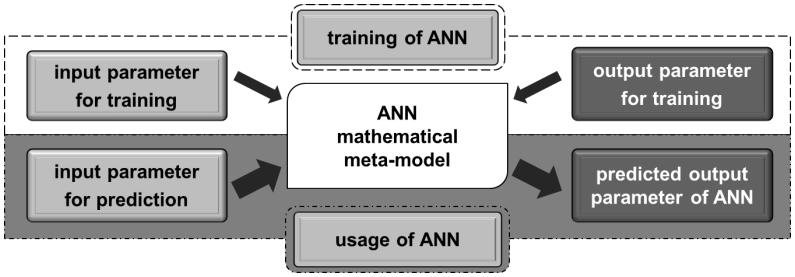

Figure 3. Training and usage of artificial neural network (ANN)

2.2 Principal Component Analysis (PCA)

Principal Component Analysis is the best known and widely used dimension-reducing technique (Jolliffe, 2011). Generally, almost any data matrix can be simplified by PCA (Wold et al., 1987). In this paper, the calculation of the principal axes is important. These axes can be calculated from arbitrary data, also from the Cartesian coordinates of nodes in 3D space, as shown in Figure 4. The principal axes of the considered parts can be aligned with the coordinate axes of the 3D space (in Figure 4 these axes are marked by dimension 1 and dimension 2). As the principal axes do not specify any direction of the parts in space, the parts can align in different positions. The difference is a rotation of 180 degrees around the center of gravity of the point cloud for each dimensional axis. These different positions are marked in Figure 4 with different colors.

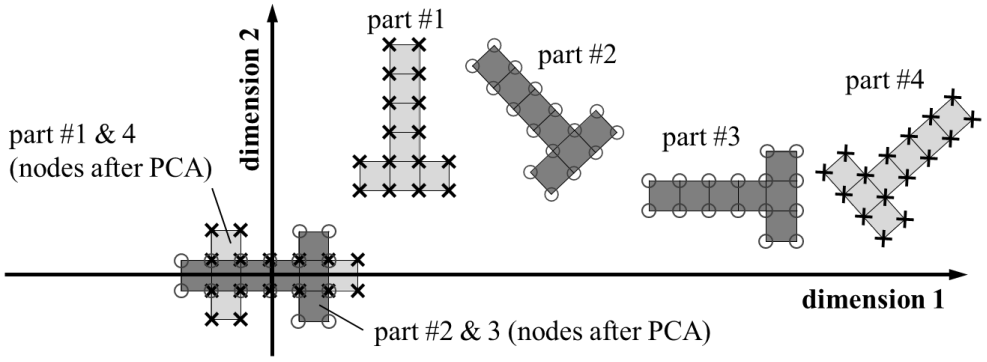

Figure 4. Shift and rotation of parts with the use of their principal axes

3 PART-RECOGNITION TASK # 1: TRAINING BY EXPERT

The concept in this paper is an extension to the already published concepts in Spruegel and Wartzack (2014a, 2014b). Shortcomings, such as the need for the consideration of many different rotation

angles, are overcome by the use of PCA. Furthermore, a coordinate system transformation from Cartesian to spherical coordinates, and the use of a spherical detector surface, eliminates issues with the detection of parts of different sizes. The whole process with seven steps can be seen in Figure 5, and each step is explained in this section. The main advantages of the presented concept is the automatic detection of parts, as well as the short time that is needed to scan a whole assembly for specific parts, such as standard components with no semantic information whatsoever. At first, data for the training of the ANN must be generated, and the individual steps have to be done once for every part that needs to be recognized with the ANN.

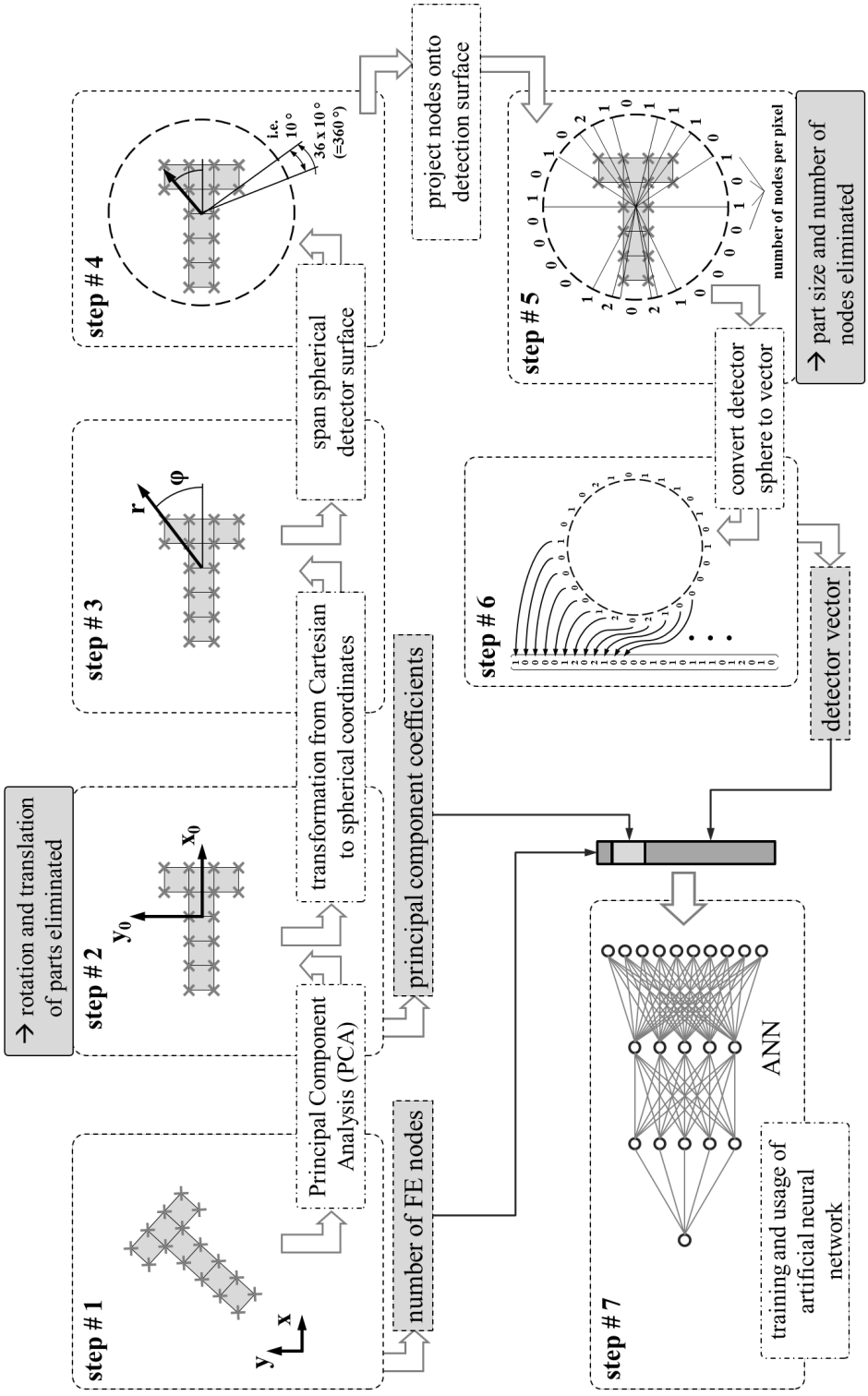

Figure 5. Seven steps for training an ANN to perform part recognition based on FE node coordinates

3.1 Part-Recognition: Step # 1

The geometry information of a CAD part is transferred to the FE software, and meshed with a fixed element type and element size. The meshing with 10-node-tetrahedral solid FE elements is well suited for meshing parts of an individual shape automatically. Meshing with an element size of 0.75 mm and a strict meshing algorithm is well suited for the part-detection of standard components, such as bolts, screws or circlips. Within the FE software, the coordinates of each node can be extracted in Cartesian coordinates, in relation to the global coordinate system.

3.2 Part-Recognition: Step # 2

In CAD assemblies parts can be oriented arbitrarily in 3D space. Therefore, a PCA is used to rotate and shift the parts into a Cartesian coordinate system at the axis origin, with $x = y = z = 0$. This is achieved by shifting the center of gravity of the coordinate points into the point of origin. Afterwards, the part is rotated, so that the parts' principal axes align with the axes of the coordinate system at the origin. As the principal axes contain no information on the direction, the rotation is not distinct. Hence, for each axis there are two different positions and the corresponding mirrored positions, and for all axes twelve possible positions need to be considered. The two mirrored positions per axis can be seen as reflected parts along a plane, spanned by the two remaining axes (shown in Figure 7 for a demonstrator part). After this step, the infinite number of arbitrary orientations in 3D space has been reduced to twelve possible orientations for any considered part.

3.3 Part-Recognition: Step # 3 and # 4

In the third step, the Cartesian coordinates are transferred to spherical coordinates. Still, the considered parts are very different in size, and with equal meshing (edge length 0.75 mm) there is a significant difference in the number of nodes for every part. Therefore, a detector surface with the same number of detector pixels is introduced in step # 4. The pixels are only defined by two angles: the inclination/polar angle; and the azimuth/azimuthal angle. A typical angle range could be 10°, which leads to 36 x 36 (= 1,296) pixels. The fixed number of pixels is essential, as the input of an ANN always needs a fixed number of input parameters. Since the pixels are only defined by the two angles, the radial distance and therefore the size of the considered parts can be eliminated from the consideration.

3.4 Part-Recognition Steps # 5 and # 6

In the fifth step, the part's nodes are projected onto the detector surface, originating from the spherical coordinate system. Then the number of projected nodes on each pixel is summed up and saved in a matrix of size 36 x 36. Afterwards, the matrix is reshaped in step # 6, to form a column vector. This vector is used to form the column vector input for the ANN, including also the number of nodes from step # 1, and the principal component coefficients from the PCA in step # 2.

3.5 Part-Recognition Step # 7

In step # 7 the ANN is trained. Thus, the previous steps must be performed for each part that will be recognized later. As the principal axes do not specify the direction, parts need to be mirrored and rotated along the three axes. At most, twelve different orientations per part have to be considered. After that, an output value is defined for the different parts. The further twelve orientations of each part have the same output value.

To sum it up, the ANN is trained with an input matrix of 1,306 rows (1,296 from detection matrix; 9 from principal coefficients; 1 from the number of nodes in FE mesh), and the number of columns depends on the number of parts multiplied by twelve (for the possible orientations after the PCA). As mentioned above, the number of hidden neurons is set approximately to the number of training samples, because this will result in a good ANN with low errors. The ANN will be trained with two hidden layers. The training algorithm for the network will be *trainscg*, which is integrated in the MATLAB® Neural Network Toolbox™. *Trainscg* is a scaled conjugate gradient backpropagation algorithm, which can handle a large number of neurons, on one hand, and requires only small hardware resources, on the other hand. After the training of the network, the performance must be evaluated. An evaluation of mathematical meta-models can be done using the network to predict output parameters for known samples. Consequently, the results of the ANN and the actual values can

be compared (Walter et al., 2013). A promising way to evaluate meta-models is the Coefficient of Prognosis (CoP), which was introduced by Most and Will (2008). Hereby y is the true output value, whereas $\hat{y}$ is the calculated output of the meta-model. The corresponding mean value is μ, σ is the standard deviation of the whole dataset, and N is the number of samples.

$$CoP = \left(\frac{E[Y \cdot \hat{Y}]}{\sigma_Y \cdot \sigma_{\hat{Y}}}\right)^2 = \left(\frac{\sum_{k=1}^{N}(y_k - \mu_y)\cdot(\hat{y}_k - \mu_{\hat{y}})}{(N-1)\cdot\sigma_y \cdot \sigma_{\hat{y}}}\right)^2 \quad (1)$$

The CoP value of a mathematical meta-model should be above 98 %, to achieve very good results (Walter and Wartzack, 2012).

4 PART RECOGNITION TASK # 2: APPLICATION BY PRODUCT DEVELOPER

As mentioned earlier, part-recognition is essential for the identification of a specific overarching simulation issue. After training, the ANN can be used for part-recognition by the product developer within the FE-software. For this purpose, the previously mentioned steps # 1 to # 6 are performed for the parts that will be recognized. In step # 7 the input vector is handled over to the ANN, and an output value is calculated. This value is compared to the specified output values of the trained parts, and the new part can either be recognized or is classified as a new part that has not been taken into consideration so far.
The concept of recognition must be integrated into the FE software, and has to be easy in use for less-experienced simulation personnel, such as design engineers. For the product developer it is not important how the concept or how the ANN can identify parts from a given FE mesh; these details only need to be dealt with by the expert.

5 APPLICATION OF THE PART-RECOGNITION CONCEPT

5.1 Training of the Neural Network

A total of 94 standard components (shown in Figure 6) are used for the primary realization, including parts from twelve different DIN and ISO standards with diverse parameter values (i.e. from ISO 4762 – M6 x 20 to ISO 4762 – M10 x 35). Each part has a distinct output value, which depends on the standard, the nominal dimension, and the size of the part. The thousands position of the output value is defined by the standard (i.e. 6,xxx for ISO 1207); the hundreds position depends on the nominal dimension (i.e. x,1xx for the size M6); and the tens position is defined by the length of the screw (i.e. x,x10 for a screw with length 20 mm). Therefore, the ISO 1207 – M6x20 screw gets the output number 6,110, whereas the ISO 1207 – M8x25 screw gets the number 6,220, and so on. The minimum difference of the distinct output value from different parts is at least ten.

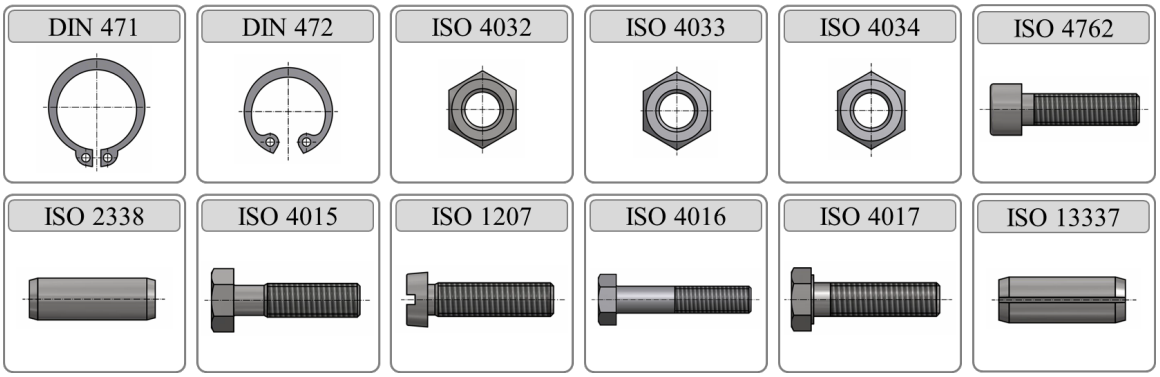

Figure 6. Different standard components for the application of the part recognition concept

At first, the parts are meshed within the FE software ANSYS®, and the node coordinates are handed over to MATLAB®. Subsequently, a PCA is performed, and each FE node is shifted into principal space. Simultaneously, the principal component coefficients are calculated, and the number of nodes is evaluated. After the transformation from Cartesian to spherical coordinates, the detectorsphere is used to evaluate the number of nodes per detector pixel. Thereafter, the input vector for the ANN is created

by transforming both the principal component matrix (size 9 x 9) and the detector matrix (size 36 x 36), along with the number of FE mesh nodes, into one single row vector (size 1 x 1,306).
The described procedure is done twelve times for each part, to take into consideration the different alignments of the principal components along the Cartesian Coordinate Axis. Therefore, the nodes in the principal space after the PCA are rotated and mirrored along the different axes. In Figure 7, the twelve different orientations can be seen. For visualization purposes, each part's center of gravity deliberately does not align with the origin of the axes.

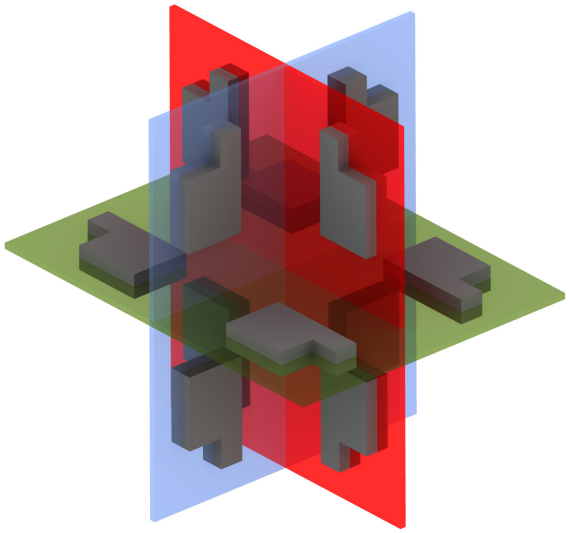

Figure 7. Visualization of possible orientations after PCA (part's center of gravity deliberately not aligned with origin)

The architecture of the ANN (shown in Figure 8) consists of 1,306 neurons in the input layer, 2 hidden layers with 550 neurons each, and an output layer with one single neuron. Within the hidden layer, tan-sigmoid transfer functions are used, and in the output layer a linear transfer function is used. Due to performance issues, resulting from the large number of weights that must be calculated, a Scaled Conjugate Gradient Algorithm is used for training. The memory requirements of the *trainscg* algorithm are relatively small, and yet it is much faster than standard gradient descent algorithms (Beale et al., 2014).

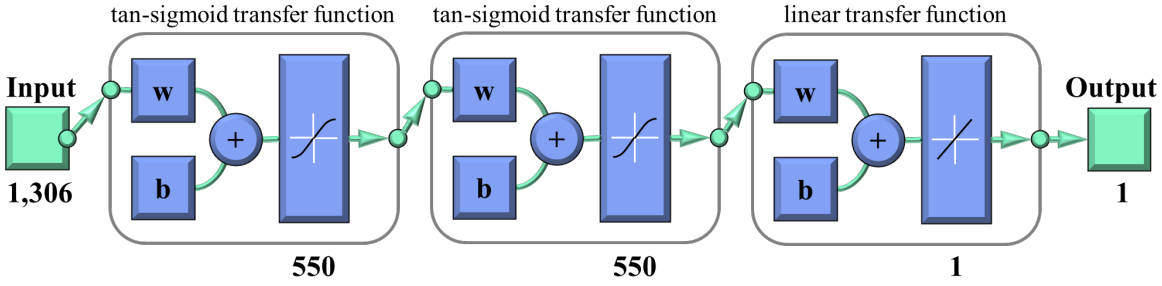

Figure 8. Architecture of the used ANN with two hidden layers with 550 neurons each

5.2 Evaluation of the trained Neural Network

After training, the ANN must be evaluated, and the corresponding output to the input data (used for the training) can be calculated. The network must be able to predict the output with extremely high precision, as the output was already known to the network, and was used for the calculation of the ANN weights during training. As expected, the Coefficient of Prognosis value is 100 %, which means that the network is predicting the correct values. In Figure 9, the difference between the expected/exact and the predicted output is plotted for all 94 parts. A maximum deviation of ~2e-8 can be observed for part number 3. This part is a circlip (DIN 471 – 20 x 1.2), and is shown as an example in the same figure, with the intended output value of 1,130, and the calculated output value from the ANN of

1,130.000000019972. It is obvious that this is a very small deviation, and that the part can be predicted with high accuracy, as the nearest intended output value would be 1,120 (calculated output 1,119.9999999993808) for DIN 471 – 15 x 1. With a simple rounding, this results in a 100% accuracy of the part-recognition approach with an ANN.

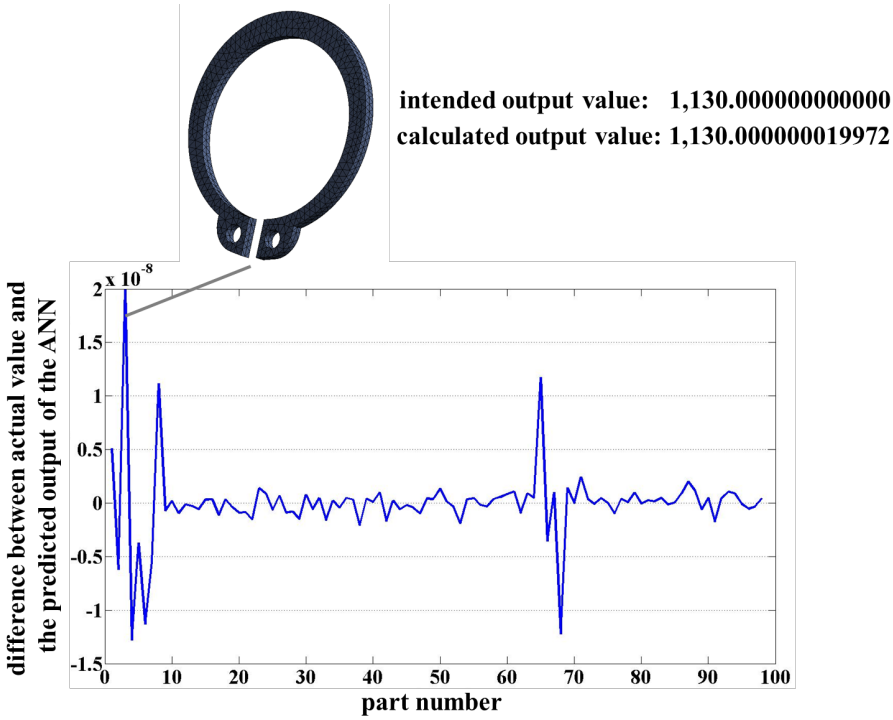

Figure 9. Deviation of ANN output for each of the 94 parts

5.3 Application of the trained Neural Network

The application of the ANN is done by the product developer. Therefore, the method described in this paper is applied on parts in a CAD assembly within the FE software. At first, all parts in the assembly are meshed with the defined tetrahedral elements of size 0.75 mm, and the node coordinates are transferred to the functions that are provided by the expert. The part-recognition itself is performed in less than one second per part, and parts that were not considered by the expert can also be identified, as the network calculates an output that obviously does not belong to any predefined part. This is very important as in an assembly may contain many different parts that are designed individually, and are only used once for a specific product.

6 LIMITATIONS

The presented approach in this paper works very well for previously trained parts, even with slight geometry variations thanks to the use of an ANN. As the geometry variations between the previously trained parts and the part − that is to be recognized − are increasing, the recognition of the ANN is getting less and less precise. This can be overcome by the initial training of the ANN with ideal geometry and parts with slight modifications. In contrast to the application of the ANN the training of the Neural Network is very hardware-consuming and an expert is needed for the training and preparation of the training data.

7 CONCLUSION

The presented approach on part-recognition in this paper fulfils the predefined requirements. Parts can be detected rapidly, and with very high accuracy within the FE software. Only an FE mesh is needed as input for the presented method in this paper for part recognition with ANN. Furthermore, unknown parts can be easily recognized as unknown, as the predicted output of the ANN does not belong to any predefined part during the generation of the ANN by an expert. The detection only is done on basis of the geometry of the part: no further semantic information, such as, for example, the name or the

designation according to a specific standard, is needed to do the recognition. Last but not least, parts with any rotation and translation in 3D space, and of any size, are covered by the presented approach, due to the use of Principal Component Analysis, and the coordinate system transformation from Cartesian to spherical coordinates. It is also very important to use a spherical detector surface, as the size of the parts must be irrelevant, and the differing number of nodes per part can be transferred to a fixed number of input neurons from an ANN.

REFERENCES

Beale, M.H., Hagan, M.T. and Demuth, H.D. (2014) Neural Network Toolbox: User's Guide, Natick:MathWorks Inc.

Boucher, M. (2013) Enhance Engineering: Reduce Time and Optimize Products with Simulation Best Practices. Research Report Aberdeen Group.

Fletcher, L., Katkovnik, V., Steffens, F.E. and Engelbrecht, A.P. (1998) Optimizing the Number of Hidden Nodes of a Feedforward Artificial Neural Network, IEEE World Congr. Comput. Intell., Int. Joint Conf. Neural Networks, Anchorage, pp. 1608-1612.

Huang, G.B. and Babri, H.A. (1998) Upper Bounds on the Number of Hidden Neurons in Feedforward Networks with Arbitrary Bounded Nonlinear Activation Functions. IEEE Transactions on Neural Networks, Vol. 9, No. 1, pp. 224-229.

Jain, A.K. and Mao, J. (1996) Artificial Neural Networks: A Tutorial. IEEE Computer, Vol. 29, No. 3, pp. 31-44.

Jin, R., Chen, W. and Simpson, T.W. (2001) Comparative studies of metamodelling techniques under multiple modelling criteria, AIAA Technical Report 2000-4801.

Jolliffe, I.T. (1986) Principal Component Analysis. New-York:Springer.

Klein, B. (2010) FEM: Grundlagen und Anwendung der Finte-Element-Methode im Maschinen- und Fahrzeugbau. Wiesbaden:Vieweg+Teubner.

Masters, T. (1993) Practical neural network recipes in C++. Boston: Academic Press.

Most, T. and Will, J. (2008) Meta-model of Optimal Prognosis – An automatic approach for variable reduction and optimal meta-model selection, Optimization and Stochastic Days 5.0, Weimar Germany, 20-21 November 2008.

Rojas, R. (1996) Neural Networks: A systematic introduction. New York: Springer.

Roth, M.W. (1990) Survey of Neural Network Technology for Automatic Target Recognition. IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 28-43.

Spruegel, T.C. and Wartzack, S. (2014a) Konzept zur automatischen Bauteilerkennung innerhalb der FE-Software Umgebung mittels Künstlichen Neuronalen Netzen, DfX-Symposium, Bamberg Germany, 1-2 October 2014, Hamburg:TuTech, pp. 47-56.

Spruegel, T.C. and Wartzack, S. (2014b) Konzept zur automatischen Bauteilerkennung von Normteilen in ANSYS Mechanical mittels Künstlichen Neuronalen Netzen, ANSYS Conference & 32. CADFEM Users' Meeting, Nuremberg Germany, 4-6 June 2014.

Walter, M. Spruegel, T. and Wartzack, S. (2013) Tolerance analysis of systems in motion taking into account interactions between deviations. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 227, No. 5, pp. 709–719.

Walter, M. and Wartzack, S. (2012) From vector-chain-based to geometrical Tolerance Analysis of Systems in Motion – The Use of Metamodels and their evaluation, 9th Annual Optimization and Stochastic Days, Weimar Germany, 29-30 November 2012.

Wold, S., Esbensen, K. and Geladi P. (1987) Principal Component Analysis, Chemometrics and Inteligent Laboratory Systems, Vol. 2, pp. 37-52.

Xu, S. and Chen, L. (2008) A Novel Approach for Determining the Optimal Number of Hidden Layer Neurons for FNN's and Its Application in Data Mining, 5th International Conference on Information Technology and Applications, Cairns Australia, 23-26 June 2008, Bathurst: Macquarie Scientific, pp. 683-686.

ACKNOWLEDGEMENTS

This work is supported by the Bavarian Research Foundation (BFS) within the scope of the Bavarian research project for "efficient product and process development by knowledge based-simulation" FORPRO²
(AZ 1071-13).