

A Neural Network Meta-Model and its Application for Manufacturing

David Lechevalier, Ronay Ak, Y Tina Lee, Steven Hudak, Sebti Foufou

▶ **To cite this version:**

David Lechevalier, Ronay Ak, Y Tina Lee, Steven Hudak, Sebti Foufou. A Neural Network Meta-Model and its Application for Manufacturing. 2015 IEEE International Conference on Big Data, Oct 2015, Santa Clara, United States. hal-01411031

HAL Id: hal-01411031
https://hal.science/hal-01411031

Submitted on 6 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Neural Network Meta-Model and its Application for Manufacturing

David Lechevalier
Le2i,
Université de Bourgogne
BP 47870, 21078 Dijon, France
david.lechevalier@etu.u-bourgogne.fr

Steven Hudak
University of Maryland,
Baltimore County, 21250, MD, US
hudak1@umbc.edu

Ronay Ak, Y.Tina Lee
Systems Integration Division, Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA
ronay.ak@nist.gov
yung-tsun.lee@nist.gov

Sebti Foufou
CSE Department, College of Engineering,
Qatar University, Qatar
sfoufou@qu.edu.qa

***Abstract*—Manufacturing generates a vast amount of data both from operations and simulation. Extracting appropriate information from this data can provide insights to increase a manufacturer's competitive advantage through improved sustainability, productivity, and flexibility of their operations. Manufacturers, as well as other industries, have successfully applied a promising statistical learning technique, called neural networks (NNs), to extract meaningful information from large data sets, so called big data. However, the application of NN to manufacturing problems remains limited because it involves the specialized skills of a data scientist.**

This paper introduces an approach to automate the application of analytical models to manufacturing problems. We present an NN meta-model (MM), which defines a set of concepts, rules, and constraints to represent NNs. An NN model can be automatically generated and manipulated based on the specifications of the NN MM. In addition, we present an algorithm to generate a predictive model from an NN and available data. The predictive model is represented in either Predictive Model Markup Language (PMML) or Portable Format for Analytics (PFA). Then we illustrate the approach in the context of a specific manufacturing system. Finally, we identify future steps planned towards later implementation of the proposed approach.

Keywords—neural network; meta-model; data analytics; PMML; manufacturing;

I. INTRODUCTION

The manufacturing industry generates a large amount of data [1]. At each manufacturing level (i.e., from the shop floor to the enterprise level), data is produced and recorded to monitor the operations that occur. One way to extract useful information from "big data" is through the application of data analytics. Applying data analytics on manufacturing data is a promising way to improve the efficiency of the manufacturing system as well as reducing the cost of production at every manufacturing level [2]. In particular, neural networks (NN), a statistical technique, has been widely used in data analytics. Examples of the NN applications are available at the process level [3], [4], at the machine level [5], [6], at the factory level [7], [8] and at the supply chain level [9], [10]. These applications demonstrate how manufacturers can make their systems smarter using neural networks. However, developing an NN model requires data science knowledge that manufacturers often do not have. As a contribution to automating the application of neural networks in manufacturing, we propose an NN meta-model (NNMM) that encapsulates data scientist knowledge about neural networks. We introduce an algorithm to automatically manipulate a neural network model (NNM), built using the meta-model and illustrate how manufacturers can leverage the meta-model to apply neural networks to their manufacturing systems.

The paper is organized as follows. Section II provides the use case scenario and required background about both neural networks and meta-models. Section III presents the NNMM and an algorithm to manipulate an NNM created using this meta-model. Section IV introduces a manufacturing use case to illustrate the capabilities of using the meta-model in the manufacturing area. We conclude this paper by presenting future work that supports automation of the neural networks application on a manufacturing system. This paper will be of interest to manufacturers that look to apply neural networks on their manufacturing systems, and schema developers who look to represent data science knowledge.

II. CONTEXT AND BACKGROUND

In this section, we introduce the use case scenario and we provide background about both neural networks and meta-models.

A. Use case scenario

This section describes a case study that focuses on predicting the energy consumed by a milling machine tool. The goal of this case study is to predict estimated energy consumption corresponding to the machine's operational parameters. The following three input variables are used to estimate the required energy to manufacture a certain workpiece:

Feed rate: the velocity at which the tool is fed

Spindle speed: rotational speed of the tool

Depth of cut: the actual depth of material that the tool is removing

A milling machine tool obviously involves more parameters. For simplicity, we decided to only keep these three parameters which have the biggest impact on the energy consumption in the milling process.

B. Regression Analysis and Neural Network (NN)

We present the characteristics of regression analysis and neural networks. Regression analysis is a statistical process used to investigate the relationships between variables [11]. Regression techniques can be roughly divided into "linear" and "non-linear" regressions. A dataset is defined as a $\mathcal{D} = \{(x_{i1}, x_{i2}, \dots, x_{ip}, y_i) | i = 1, \dots, n\}$, where p and n indicate the number of input variables and the number of samples in our dataset, respectively. y_i is the corresponding output target scalar for each input vector $\boldsymbol{x}_i$ [12].

NNs are ooriginally inspired by the function of the neurons in the brain. They are typically used for classification and forecasting. They are frequently used as an alternative to standard nonlinear regression and cluster analysis techniques. NNs are composed of computing units (called neurons or nodes) operating in parallel. These units are arranged in different layers and interconnected by weighed edges (called synapses). A layer is the term used for a vertical row of neurons. Each of these computing units performs a few simple operations and communicates the results to its neighboring units. From a mathematical viewpoint, NNs consist of a set of nonlinear (e.g., sigmoidal) basis functions with free parameters; i.e., $\boldsymbol{w}$ or weights, that are adjusted. The objective of the adjustment is to minimize the error associated with regression in an iterative process called training using a dataset. The basis functions are called activation functions.

There are many types of NNs. Feed-forward neural networks (FNNs) and recurrent neural networks (RNNs) are the two main types. A RNN has neurons that transport a signal back through the network (at least one feedback connection in recurrent network) whereas FNNs feed outputs from individual neurons forward to one or more neurons or layers in the network [13], [14]. Multilayer perceptron (MLP) neural networks and Radial basis function (RBF) neural networks are two of the most common types of FNNs used as empirical nonlinear regression models. RBF networks use a radial basis function, i.e., a Gaussian kernel, as the activation function. RBFs networks have similar universal approximation capabilities as MLP networks. For the theory and application of the RBF networks, we refer the readers to [13], [15].

A feed-forward neural network that represents our use case is illustrated in **Fig. 1**. The neural network is composed of an input layer, a hidden layer and an output layer. The input layer is the layer containing the input neurons. In our example, the input neurons represent the three input parameters: feed rate, spindle speed and depth of cut. The hidden layer contains the hidden neurons. The hidden layers are used to represent the relationships between the input layer and the output layer. The hidden layers allow a representation of the relationships among the variables when these relationships cannot be captured in a regression model. At both the input and hidden layers, there is a bias neuron. The bias neuron, which allows shifting the activation function to the left or right for improving the learning process, is a neuron with a constant output. This neuron is treated as a regular neuron in the associated layer. Finally, the output layer contains the output neuron that represents the predicted energy in our example. The free parameters $\boldsymbol{w}$ called weights are assigned to the edges that connect the different

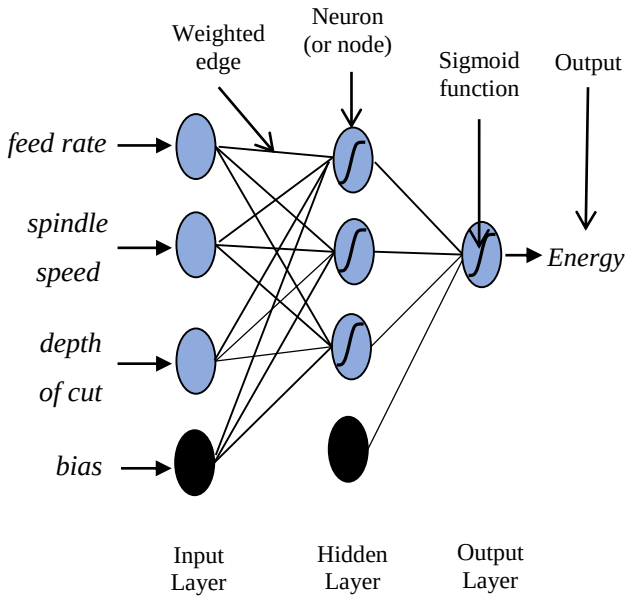


Fig. 1. Example of a MLP NN for estimating a quantity of interest

layers. During the training, these weights are adjusted to minimize the error between the output value of the NN and the real output value for a given data sample. In essence the weights represent the sensitivity of the output to the input variable. Equation (1) introduces how to compute the output value using the weights.

Each layer receives input signals generated by the previous layer, produces output signals through an activation function (e.g. a sigmoid function) and distributes them to the subsequent layer through the neurons. The network output is given by the following expression:

$$f(\boldsymbol{X}; \boldsymbol{w}) = \hat{y}(\boldsymbol{X}) = \psi\left(w_{0l} b + \sum_{j=1}^{h} w_{jl} \phi\left(w_{0j} b + \sum_{k=1}^{p} w_{ij} \boldsymbol{x}_k\right)\right), \quad (1)$$

where $\boldsymbol{X}$ is an input vector with p entries, $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p)$, p is the number of input signals (variables), $\boldsymbol{x}_k$ is the k^{th} input signal, $\boldsymbol{w}$ is the weight vector, h indicates the number of hidden neurons, w_{ij} is the synaptic weight from ith neuron to jth neuron, $\phi()$ and $\psi()$ are the activation (transfer) functions from input layer to hidden, and hidden layer to output layer, respectively, and b stands for the bias factor.

An estimate $\widehat{\boldsymbol{w}}$ of $\boldsymbol{w}$ can be obtained by a training procedure aimed at minimizing the quadratic error function, E, on a training set:

$$E(w) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 , \quad (2)$$

where $\hat{y}_i = f(\boldsymbol{X}_i; \hat{w})$ represents the output provided by the NN in correspondence of the input $\boldsymbol{X}_i$, and n is the number of the training samples.

C. *Motivation and Requirements for a Neural Network Meta-model in Manufacturing*

This section presents the motivation and requirements to develop an NNMM for manufacturing. The survey presented in the section I shows that neural networks are widely used at different manufacturing applications. For each application, a neural network needs to be built based on the manufacturing knowledge involved. This manual process requires data science knowledge, in addition to manufacturing knowledge. A manufacturer that does not have sufficient data science knowledge will need external help to apply data analytics on a manufacturing system. In [16], authors describe a framework to automatically apply data analytics on a manufacturing system. The suggested framework uses manufacturing and data science knowledge to establish relationships between these two worlds. One central component of the framework is a meta-model repository where knowledge of manufacturing and data science are described.

A meta-model is a set of concepts, rules and constraints used to describe a specific domain [17]. Thus, a meta-model provides domain-specific abstractions representing all of the relevant domain-specific core components. Such abstractions offer capabilities to maintain consistency across an entire domain, as well as capabilities to understand models of that domain in a common way. Models, as instances of a given meta-model, instantiate objects of the meta-model to represent a system inside the specific domain. If a model conforms to a domain-specific meta-model, it can be used to generate other models by simply manipulating the model objects in accordance with the meta-model specifications and meaning. Moreover, libraries instantiating meta-model components can be provided to facilitate the representation of the system inside a model.

An NNMM can contribute to the described framework by providing the capabilities to represent and communicate neural networks. An MM can provide a common way to represent NNs regardless of the available data or the specific software or programming languages used to generate the neural network. The NNMM can also facilitate the exchange and the manipulation of models that are built based on the NNMM. Using an NNMM, an algorithm can automatically create an NNM to represent an NN. Another algorithm can manipulate the generated model according to the meta-model specification and, finally, achieve a specific data analytics task using an NN as represented in the NN model. One example of task is the training of a neural network.

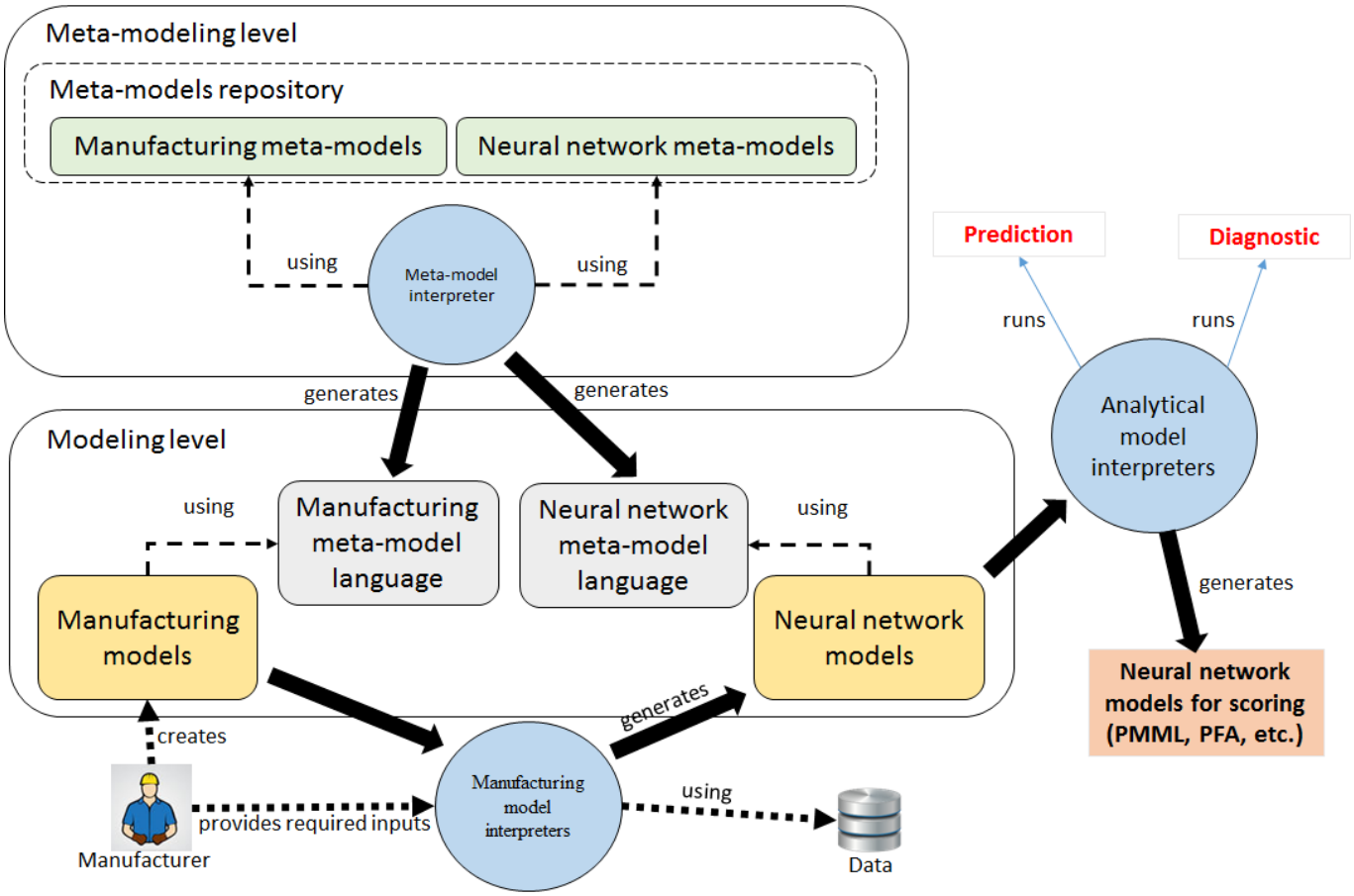


Fig. 2. Workflow to automatically apply data analytics on a manufacturing system

Manufacturers can leverage these capabilities to generate neural network models representing manufacturing systems and then apply data analytics for their systems. **Fig. 2** illustrates a workflow that does this, assuming that meta-models representing manufacturing knowledge and analytical knowledge are available. A manufacturer can represent a manufacturing system in a model using available concepts defined in a manufacturing meta-model. Once the manufacturing model is built, the manufacturer defines the target of the analysis and provides collected data to an algorithm. Defining the target of the analysis consists in defining the studied metric and the type of analytics the manufacturer wants to run (i.e., classification or regression). From the manufacturing model, the target of the analysis and the data, the algorithm automatically generates a neural network model using the concepts defined in the NNMM. In our use example, a manufacturer could have a milling machine model as its manufacturing model. The manufacturer would treat energy as the studied metric and regression as the type of analytics. The algorithm would then generate the NNM representing an NN as described in **Fig. 1**. Section V presents initial capabilities that this algorithm should provide.

In this workflow, it is important to understand that there are several types of manufacturing models. One model could be a simulation model while another model could be a production scheduling model. In both cases, these models can be very complex. Running data analytics directly from these models requires 1) developing an algorithm to apply neural networks specifically for each type of model and 2) modifying one's algorithm every time these models are modified. Providing an NNMM enables generation of an NNM from the manufacturing model. Once an NNM is generated, the manufacturer executes data analytics from the NNM. If a manufacturing model is modified, it does not imply that the NNM needs to be modify as well and the manufacturer does not need to repeat the step of generating the NNM. A manufacturer can keep the existing NNM model and run data analytics from this NNM until the new manufacturing model requires the NNM to be modified. Thus, manufacturers save time and money by reusing the NNM. In addition, an important modification of the manufacturing model

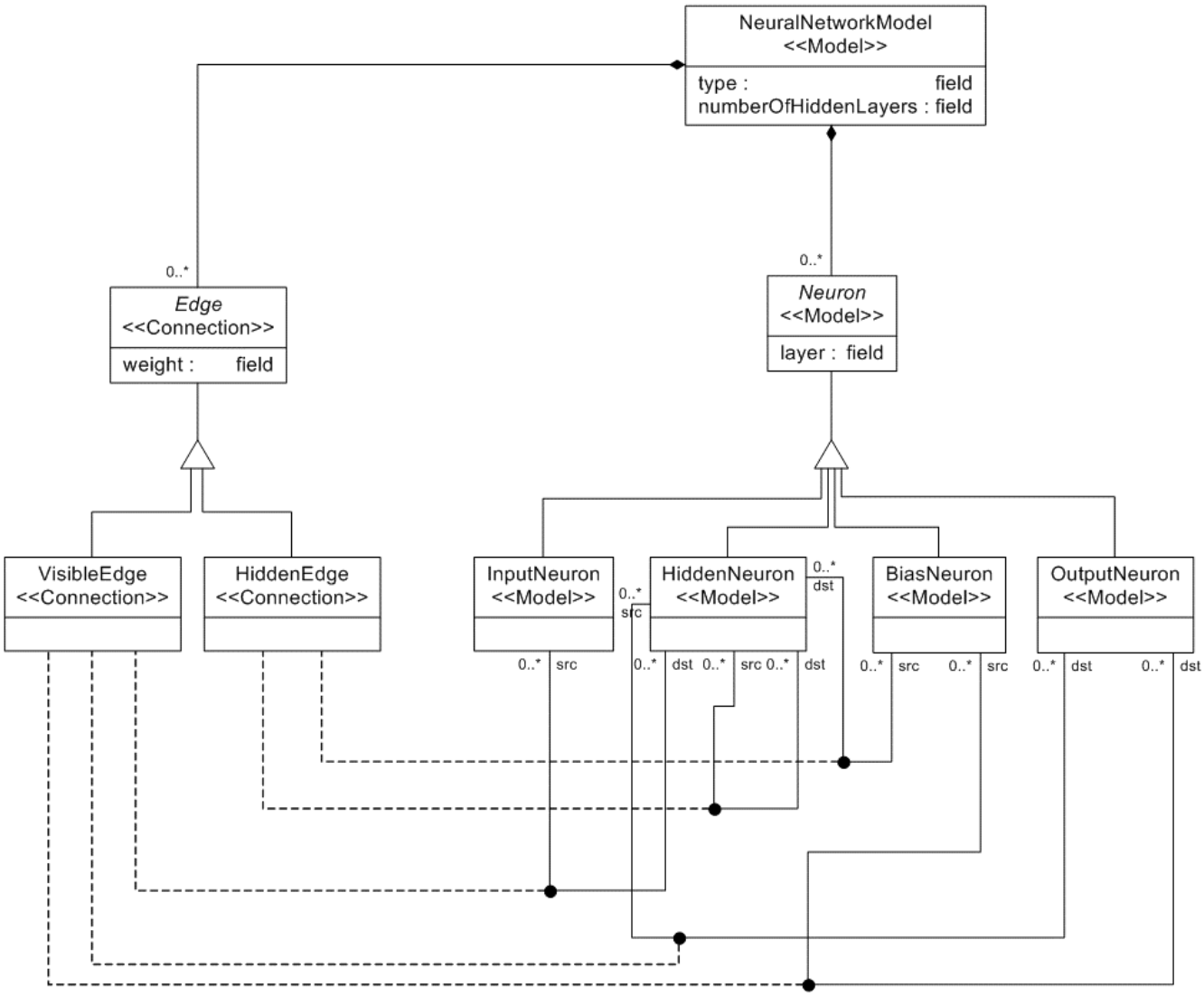


Fig. 3. Neural Network Meta-Model

only requires a modification of the algorithm to transform this model into a neural network model. The algorithms to manipulate this neural network model do not need modification.

The NNM created in the modeling environment represents only the structure of the neural network. Using this NNM, the data given as inputs by the manufacturer, and available tools or libraries implementing machine learning techniques, another algorithm can execute data analytic tasks. The choice of the executed task depends on the type of analysis that the manufacturer defines at the beginning of the workflow. The algorithm can also generate analytical models in standard format to apply scoring using these models and existing software that understands this format. Scoring is the process of using a model to make predictions about behavior.

III. Description of the Neural Network Meta-model

As described in Section II.B, a variety of neural networks are available. Creating an NNMM must facilitate representing any neural network. Moreover, using an NNMM, models have to be easy to understand and manipulate. To achieve these requirements, the meta-model needs to provide capabilities to represent any specificity of any neural network. The meta-model needs to provide explicit and clear concepts to make the model generation, understanding and manipulation simple. This paper will present our NNMM to represent neural networks and meet these requirements. It also describes the architecture of the second algorithm mentioned in the workflow.

A. *Description of the Neural Network Meta-model*

In this section, an NNMM developed in the Generic Modeling Environment (GME) [18] is proposed. GME is a tool for creating custom, domain-specific modeling environments. The domain-specific modeling environment is used to create high-level descriptive models of objects in a given domain. GME is also used to build models of real world objects in that domain. The NNMM represents different types of neural networks, including FNN, through various abstractions. In this work, we exclude the representation of RNN. An RNN 1) allows an edge to have the same neuron as its source and its destination and 2) requires higher-level abstractions than the abstractions presented in this NNMM. Including RNN will be a future step to allow the representation of all kinds of neural networks.

The meta-model needs to be easy to use, thus we have chosen a simplified representation of each concept. **Fig. 3** shows the NNMM. A *NeuralNetworkModel* concept is composed of *Neuron* and *Edge* concepts. *Neuron* is an abstract concept and is extended by four concepts: *InputNeuron*, *HiddenNeuron*, *BiasNeuron* and *OutputNeuron* concepts. *Edge* is an abstract concept that is extended by the *VisibleEdge* and *HiddenEdge* concepts. A *VisibleEdge* is used to represent an edge between an input neuron and a hidden neuron, between a hidden neuron and an output neuron and between a bias neuron and an output neuron. Edges between two hidden neurons and between a bias neuron and a hidden neuron are represented using a *HiddenEdge*.

This NNMM is built to facilitate its adoption by allowing the extension of the meta-model to meet new requirements. For instance, in the proposed NNMM, a layer is represented by a neuron attribute . However, one can create a concept called *Layer* as a new concept as part of a *NeuralNetworkModel.*

B. *Algorithm Generating a Predictive Model from an NNM and Data*

To illustrate the capabilities of an NNMM, we implement an algorithm that automatically generates a predictive model using a neural network model and data. Four tasks must be executed to complete the workflow from an NNM and data to a predictive model in a specific format e.g., Predictive Model Markup Language (PMML) [19] or Portable Format for Analytics (PFA) [20]. Tasks, as shown in **Fig. 4**, are 1) extract the available information specified in the NNM 2) build a data structure representation of the NNM using the extracted information, 3) train this data structure with the given data and updating the NNM, and 4) convert the trained data structure into a standardized or neutral format.

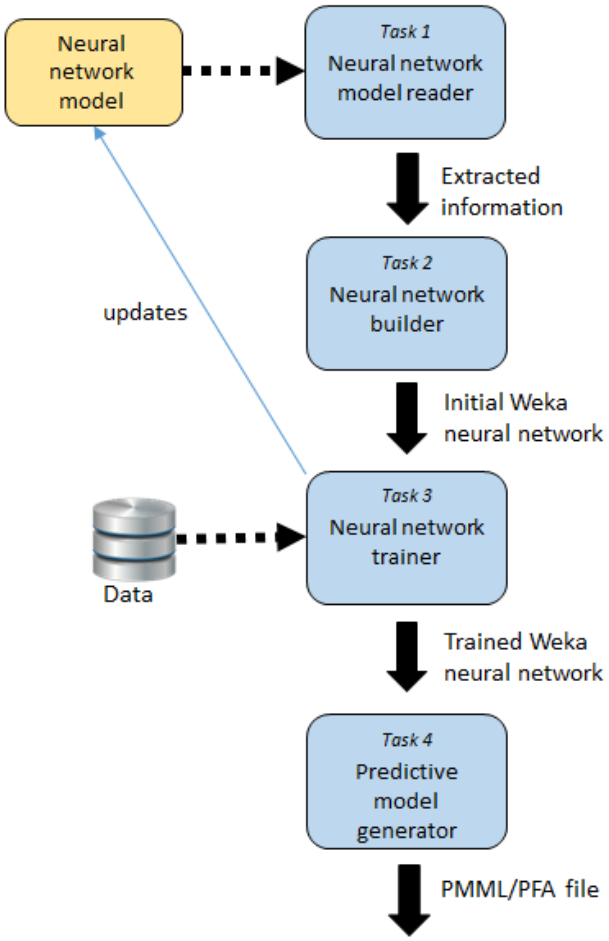


Fig. 4. Process flow to generate a predictive model from an NNM and data

Fig. 4 represents the process flow of the algorithm as an example of an analytical model interpreter presented in the workflow in **Fig. 2**. The NNM is implemented using GME. GME allows users to develop an algorithm that can directly manipulate the implemented model. First, the algorithm executes the first task. The relevant information extracted from the NNM is the type of neural network, the number of hidden layers, the number of hidden neurons per layer, the number of

inputs neurons, the number of outputs neurons, and the names the NNM uses to refer to each neuron and connection within the network. This information is provided as the input of the second task. The second and third tasks are accomplished using Weka [21], a Java machine-learning library. Using the information given as inputs, the algorithm can instantiate a Weka data structure representing the neural network specified in the model. Using data provided at the beginning of the workflow, the algorithm trains the neural network using functions implemented in the Weka library. The algorithm provides capabilities to customize the training by modifying several parameters of the training: learning rate, momentum, number of iterations, size of validation and random seed that is the number used to initialize the pseudorandom number generator. Default training parameter values are suggested as well. Finally, the NNM is updated to include the neural network edge weights that have been computed during the training. The final task is accomplished using Java PMML (JPMML) application program interface (API) [22], a Java library for producing and scoring PMML documents, and using Titus [23], a Python library for producing and scoring PFA documents.

Using these libraries and the Weka data structure trained during the previous task, a PMML or PFA document can be generated for scoring. The conversion from the Weka data structure, which represents the trained neural network to a PMML or PFA document involved considerable programming because Weka does not contain a method for exporting a neural network into either of these formats. This programming requires understanding of the Weka library and the PMML and PFA specifications. The algorithm is developed to provide an automatic way to build, train and manipulate a neural network from a neural network model.

IV. USE CASE

In this section, we will present the NNM and an implementation of the algorithm generating a predictive model for the use case we described in Section II.A.

A. *Milling Process Analysis: Process Description and Neural Network Representation*

The data set used in this case study was generated for [24]. To generate the data, Park et al. describe an intelligent machine monitoring framework that consists of two agents: i) a data management and extraction agent, and ii) a data-driven machine-learning and knowledge-extraction agent. The former agent consists of an MTConnect agent [25] and a data post-processor. The MTConnect agent retrieves raw sensor data from a machine tool and systematically converts and organizes the data into semantically meaningful input features and response outputs.

The raw data includes the timestamp, power demand, feed rate, spindle speed, and numerical control (NC) code block (an NC code block corresponds to a specific cutting operation). The raw data is processed to obtain derived data, such as the average feed rate, average spindle speed, and cumulative energy consumption for each NC code block. The data set also includes simulated data, such as the depth of cut and cutting strategy (i.e., climb versus conventional milling). This data is determined by comparing two sequential tool positions reported by the MTConnect agent relative to the actual dimensions and location of the workpiece. The data set was generated from a part machined with different cutting operations such as face milling, contouring, slotting, pocketing, and drilling. Refer to [24], [26], [27] for detailed information about the experimental design and data processing techniques used to generate the data set.

With three input variables and one output variable, the NN structure therefore consists of three input neurons and one output neuron. For simplicity, in this example, we have used only one hidden layer and three hidden neurons.

B. *Using the Neural Network Model and Data to Generate PMML file*

Using the NNMM, an NNM that represents our case study is created. **Fig. 5** shows the NNM created in GME to represent our use case.

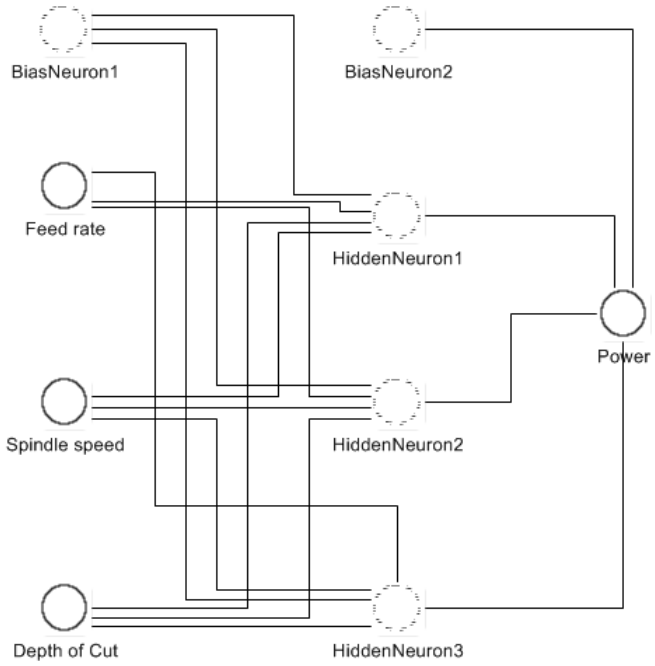


Fig. 5. NNM to study the energy consumed by a milling machine tool

The input layer contains three input neurons representing the three parameters given as inputs: feed rate, spindle speed and depth of cut. The input layer includes a bias neuron called *BiasNeuron1*. The hidden layer including the bias neuron called *BiasNeuron2* is represented. Finally the output neuron represents the power prediction.

The algorithm described in the previous section provides an interface to provide the data and customize the parameters of the training if needed. **Fig. 6** presents the graphical user interface (GUI) that pops up when a user calls the algorithm.

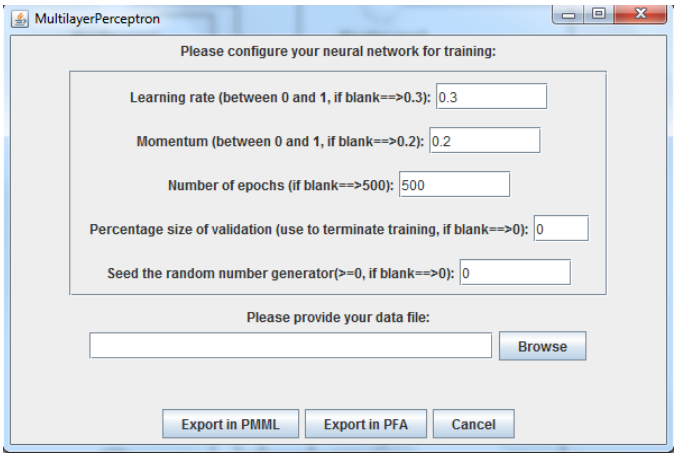


Fig. 6. Interface to provide the needed inputs to the algorithm

As mentioned in the previous section, the user does not have to change the parameters of the neural network training and can use the suggested values for the training. Data has to be provided to the algorithm to enable the training. The user can decide the format (with two options, PMML and PFA) used to export the predictive model that the algorithm generates. By clicking on the appropriate button from the GUI, the user launches the first task described in Section III.B. Following the process described in Section III.B, the algorithm trains a neural network as specified by the NNM. The algorithm updates the weight of every edge inside the NNM in GME as shown in **Fig. 7**. Finally, it creates a predictive model file in the requested format.

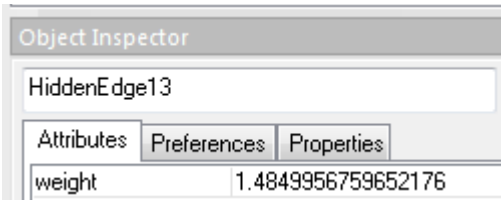


Fig. 7. Example of edge weight instantiation

The predictive model file generated from the algorithm allows the predictive model to be used by other software that understands the format of the file. Doing so users of that software can score new data against the predictive model. A manufacturer can create this predictive model and use it to establish diagnostics on the manufacturing system. The predictive model can also be used in different manufacturing facilities that use the same milling machine tool. Each factory can establish diagnostics by using their own data against this model. This flow can facilitate the application of data analytics in the manufacturing industry.

V. Conclusion and Future Work

In order to improve systems that are already quite efficient, manufacturers can take advantage of data analytics [2]. A neural network is one such technique that allows them to run regression analysis or classification to establish diagnostic or prognostic. To facilitate the representation and manipulation of neural networks, we propose a neural network meta-model. We introduce the specification of this meta-model and an algorithm describing the manipulation of the neural network to generate predictive models. The algorithm illustrates how a neural network model, created using the meta-model, is used as input with data to generate a predictive model in a standard or neutral format. Using the algorithm, a manufacturer can apply neural networks on a manufacturing system without much data science knowledge. The described use case shows an application of the algorithm to one particular system, a milling process.

We also describe a full workflow to apply neural networks on a manufacturing system from a manufacturing model. Our current work contributes to this workflow through the generation of NNMs using the NNMM using an algorithm to manipulate this NNM. To allow manufacturers to automatically generate neural network models, an algorithm should be able to transform a manufacturing model (that can be a simulation model, a mathematical model, etc.) into a neural network model using the neural network meta-model as described in the referenced framework.

Future work includes (1) providing a manufacturing meta-model to represent manufacturing systems at different levels and (2) designing and implementing a step by step process to transform a manufacturing model into a neural network model to develop a manufacturing model interpreter as described in **Fig. 2**. The first task requires defining concepts at different manufacturing levels. Classifications are already available at the process level such as in [28]. By taking advantage of the available classifications, one can define a manufacturing meta-model to represent processes, machines, factories, etc. The second task requires defining the steps to transform a manufacturing model into a neural network model. This second task requires one to identify the type and the target of the analysis, and the manufacturing knowledge represented in the manufacturing model. The algorithm designed for the second task will need to define the inputs and outputs of the neural network depending on the scenario. Extracting information using a neural network model will be different at the process level than it is at the factory assembly line level because of the complexity. The number of hidden layers and hidden neurons will change from one level to the next. Research has been done to define the most appropriate structure (i.e., the number of hidden layers and hidden neurons) of an NN and can be implemented in the algorithm developed in the second task [29].Achieving these two tasks seems promising to obtain a fully automated workflow to apply neural networks for manufacturing systems. Finally, the workflow can be generalized by including other meta-models that represent other techniques for applying data analytics. Developing these meta-models and the required algorithms to ensure the transformation between models will provide a full framework for predictive analytics in manufacturing as described in [16].

Acknowledgement

This work was supported by National Institute of Standards and Technology's Foreign Guest Researcher Program and the Ecole Supérieure d'Electricité (SUPELEC), sponsor of Ronay Ak, and the NIST Student Undergraduate Research Fellowship program.

We thank our colleagues Anantha Narayanan and Sudarsan Rachuri from NIST who provided insight and expertise that greatly assisted the research. We also thank our colleagues KC

Morris, Albert Jones and Sharon Kemmerer for their valuable comments that helped to improve the quality of the paper.

REFERENCES

[1] M. Young., and D. Pollard. "What businesses can learn from big data and high performance analytics in the manufacturing industry" Big Data Insight Group, 2012.

[2] J. Manyika, et al. "Big data: The next frontier for innovation, competition and productivity." *Technical report*, McKinsey Global Institute, 2011.

[3] P.G. Benardos and G. Cl Vosniakos. "Prediction of surface roughness in CNC face milling using neural networks and Taguchi's design of experiments." Robotics and Computer-Integrated Manufacturing 18, no. 5: 343-354, 2002.

[4] WC. Chen, PH. Tai, MW. Wang, WJ. Deng, and CT. Chen. "A neural network-based approach for dynamic quality prediction in a plastic injection molding process." Expert systems with Applications 35, no. 3 (2008): 843-849, 2008.

[5] D. Dornfeld and M. F. DeVries. "Neural network sensor fusion for tool condition monitoring." CIRP Annals-Manufacturing Technology 39.1 : 101-105. 1990.

[6] N. Ghosh, Y. B. Ravi, A. Patra, S. Mukhopadhyay, S. Paul, A. R. Mohanty, and A. B. Chattopadhyay. "Estimation of tool wear during CNC milling using neural network-based sensor fusion." Mechanical Systems and Signal Processing 21, no. 1: 466-479, 2007.

[7] Y. Lin, J. Shie, and C. Tsai. "Using an artificial neural network prediction model to optimize work-in-process inventory level for wafer fabrication." Expert Systems with Applications 36, no. 2: 3421-3427, 2009.

[8] KJ Wang, J. C. Chen, and Y-S. Lin. "A hybrid knowledge discovery model using decision tree and neural network for selecting dispatching rules of a semiconductor final testing factory." Production planning & control 16, no. 7: 665-680, 2005.

[9] M. Chiu, and Grier Lin. "Collaborative supply chain planning using the artificial neural network approach." Journal of Manufacturing Technology Management 15, no. 8: 787-796, 2004.

[10] R. Carbonneau, K. Laframboise, and R. Vahidov. "Application of machine learning techniques for supply chain demand forecasting." European Journal of Operational Research 184, no. 3: 1140-1154, 2008.

[11] M. A. Golberg and H. A. Cho, *Introduction to Regression Analysis*. WIT Press, 2004.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2013.

[13] K.-L. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning*. Springer Science & Business Media, 2013.

[14] R. Rojas, *Neural Networks: A Systematic Introduction*. Springer Science & Business Media, 2013.

[15] R. J. Howlett and L. C. Jain, Radial Basis Function Networks 2: New Advances in Design. Physica, 2013

[16] D. Lechevalier, A. Narayanan, and S. Rachuri. "Towards a domain-specific framework for predictive analytics in manufacturing." IEEE International Conference on Big Data, 2014.

[17] A.Ledeczi, A. Bakay, M.Maroti, P.Volgyesi, G.Nordstrom, J.Sprinkle, and G. Karsai. "Composing domain-specific design environments." Computer 34.11: 44-51, 2001

[18] A.Ledeczi, M.Maroti, A. Bakay, G. Karsai, J.Garrett, C. Thomason, G.Nordstrom, J.Sprinkle, and P.Volgyesi. "The generic modeling environment." In Workshop on Intelligent Signal Processing, Budapest, Hungary, vol. 17. 2001.

[19] A. Guazzelli, Alex, M. Zeller, W. Lin, and G. Williams. "PMML: An open standard for sharing models." The R Journal 1, no. 1: 60-65, 2009.

[20] J. Pivarski, "PFA specification v0.7", 2015. [Online]. Available: http://scoringengine.org/. [Accessed: June 30th, 2015].

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. "The WEKA data mining software: an update." ACM SIGKDD explorations newsletter 11, no. 1: 10-18, 2009.

[22] JPMML, 2015. [Online]. Available: http://openscoring.io/. [Accessed: June 30th, 2015].

[23] Titus, 2015. [Online]. Available: http://opendatagroup.com/. [Accessed: June 30th, 2015].

[24] J. Park, K. H. Law, R. Bhinge, A. Srinivasan, D. Dornfeld, M. Helu, and S. Rachuri, "A generalized data-driven energy prediction model with uncertainty for a milling machine tool using Gaussian Process," ASME International Manufacturing Science and Engineering Conference, 2015.

[25] A. Vijayaraghavan, W. Sobel, A. Fox, D. Dornfeld, and P. Warndorf, "Improving Machine Tool Interoperability Using Standardized Interface Protocols: MT Connect," International Symposium on Flexible Automation, 2008.

[26] R. Bhinge, N. Biswas, D. Dornfeld, J. Park, K. H. Law, M. Helu, and S. Rachuri, "An intelligent machine monitoring system for energy prediction using a Gaussian Process regression," IEEE International Conference on Big Data, 2014.

[27] M. Helu, S. Robinson, R. Bhinge, T. Bänziger, and D. Dornfeld, "Development of a Machine Tool Platform to Support Data Mining and Statistical Modeling for Machining Processes," Machine Tool Technologies Research Foundation Annual Meeting, 2014.

[28] R. Todd, A. DK. Allen, and L. Alting. "Manufacturing processes reference guide". Industrial Press Inc., 1994.

[29] SK. Gnana, and S. N. Deepa. "Review on methods to fix number of hidden neurons in neural networks." Mathematical Problems in Engineering, 2013