A machine learning approach for minimal coordinate multibody simulation

Andrea Angeli, Frank Naets and Wim Desmet

Abstract Over the years, a wide range of generalized coordinates have been proposed to describe the motion of rigid and flexible multibody systems. Depending on the type of formulation, a different equation structure is obtained for the model. Most formulations rely on a redundant number of Degrees Of Freedom (DOFs) and some associated constraints, leading to a set of Differential-Algebraic Equations (DAEs) to model the system dynamics. On the other hand, the 'Minimal Coordinate' formulation describes the dynamics through a minimal amount of DOFs and leads to a system of Ordinary Differential Equations (ODEs). For many applications, this ODE structure is an important benefit, as it enables a natural integration for state-estimation and model-based control. The backside of this approach is that it is generally not-straightforward to find a minimal number of parameters to unequivocally describe the system configurations, especially for complex mechanisms. In this work, a machine learning approach based on Auto-Encoders is proposed to find a non-linear transformation that leads to a minimal parameterization of the motion. It is shown that such non-linear transformation can be used to project into minimal coordinates while its inverse permits to perform the simulation in the reduced dimension and re-obtain the original coordinates.

1 Introduction

Over the past decades, several formulations have been proposed to describe the motion of rigid and flexible multibody systems. For rigid mechanisms, such formalisms can be divided in three main categories according to the number and typology of Degrees Of Freedom (DOFs): usage of a minimal amount of coordinates as the 'Mini-

Andrea Angeli, Frank Naets and Wim Desmet
Department Mechanical Engineering KU Leuven, Celestijnenlaan 300B 3001 Leuven (Belgium)
DMMS lab Flanders Make
e-mail: [andrea.angeli, frank.naets, wim.desmet]@kuleuven.be

mal Coordinates (MC)' [1] or 'Joint Coordinates (JC)' [2], use of angle and position coordinates as the 'Cartesian Coordinates (CC)' [3] and use of only position coordinates as the 'Natural Coordinates (NC)' [4]. The formulations have been extended to describe flexible mechanisms subject to small deformations as, for example, the 'Global Modal Parameterization (GMP)' [5, 6], the 'Floating Frame of Reference (FFR)' [7], and the 'Flexible Natural Coordinates Formulation (FNCF)' [8].

Depending on the type of formulation used, a different equation structure is obtained for the model. Currently, most of the formulations rely on a redundant number of Degrees Of Freedom (DOFs) and some associated constraints, leading to a set of Differential-Algebraic Equations (DAEs) to model the system dynamics. On the other hand, the MC formulation describes the dynamics through a minimal amount of DOFs and leads to a system of Ordinary Differential Equations (ODEs). For many applications, this ODE structure is an important benefit, as it enables a natural integration for state-estimation and model-based control. The backside of this approach is that it is generally not-straightforward to find a minimal number of parameters to unequivocally describe the system configurations, especially for complex mechanisms.

In this work, starting from a reference multibody simulation expressed in Natural Coordinates, a machine learning approach is proposed to reduce the model to Minimal Coordinates. An Auto-Encoder (AE) neural network is used to find a non-linear transformation that leads to a minimal parameterization of the rigid motion. The inverse function can then be used to perform the simulation of the model dynamics in the reduced dimension and re-obtain the full-order coordinates.

The paper is organised as follows: Section 2 describes the proposed approach and Section 3 presents an application example; Section 4 reports some concluding remarks.

2 Auto-Encoders for minimal parameterization of multibody systems

This section describes the proposed procedure: in the first part Auto-Encoders are introduced, while in the second part their application to multibody dynamics is presented.

2.1 Auto-Encoders

Recently, novel non-linear dimensionality reduction techniques have been proposed to address the limitations of classic linear methods such as Principal Component Analysis (PCA) [9]. In particular Auto-Encoders, a sub-class of Neural Networks, seem promising to achieve such aim [10]. Here, we provide a brief overview of the key concepts of these methods.

2.1.1 Neural Networks

In general, a Neural Network (NN) is the composition of several basic units as the one reported in Fig. 1.

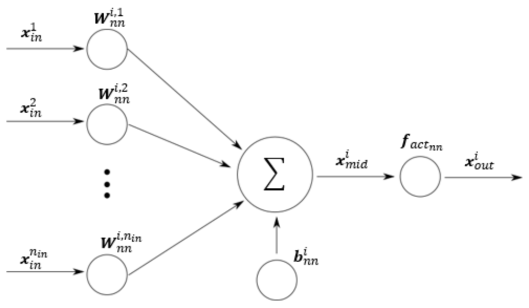

Fig. 1 A Neural Network unit. The terms of the input $\boldsymbol{x}_{in}$ are multiplied by the weights in $\boldsymbol{W}_{nn}$ and added to the bias in $\boldsymbol{b}_{nn}$ before going through the activation function $\boldsymbol{f}_{act_{nn}}$. The full layer will be composed by $i = 1, \ldots, n_{out}$ units to obtain the n_{out} terms of the output $\boldsymbol{x}_{out}$.

A vector $\boldsymbol{x}_{in} \in \mathbb{R}^{n_{in}}$ is the input to the NN layer. It is multiplied by the weight matrix $\boldsymbol{W}_{nn} \in \mathbb{R}^{n_{out} \times n_{in}}$ and a bias vector $\boldsymbol{b}_{nn} \in \mathbb{R}^{n_{out}}$ is added to obtain $\boldsymbol{x}_{mid} \in \mathbb{R}^{n_{out}}$. A so-called ‘activation function’ $\boldsymbol{f}_{act_{nn}}$ is then applied to each element i of $\boldsymbol{x}_{mid}$, to obtain the output vector $\boldsymbol{x}_{out} \in \mathbb{R}^{n_{out}}$:

$$\boldsymbol{x}_{out} = \boldsymbol{f}_{act_{nn}}(\boldsymbol{x}_{mid}) = \boldsymbol{f}_{act_{nn}}(\boldsymbol{W}_{nn}\, \boldsymbol{x}_{in} + \boldsymbol{b}_{nn}) \tag{1}$$

The choice of the activation function typically depends on the application, but the most common are: the linear function (lin), the Rectified Linear Unit (ReLU) and the sigmoid or logistic function (sig). They are, respectively, reported below:

$$\boldsymbol{x}_{out} = \boldsymbol{f}_{act_{lin}}(\boldsymbol{x}_{mid}) = \boldsymbol{x}_{mid} \tag{2}$$

$$\boldsymbol{x}_{out} = \boldsymbol{f}_{act_{ReLU}}(\boldsymbol{x}_{mid}) = \max(0,\, \boldsymbol{x}_{mid}) \tag{3}$$

$$\boldsymbol{x}_{out} = \boldsymbol{f}_{act_{sig}}(\boldsymbol{x}_{mid}) = \frac{1}{1 + e^{-\boldsymbol{x}_{mid}}} \tag{4}$$

The parameters of the NN $\mathscr{P}_{nn} = \{\boldsymbol{W}_{nn},\, \boldsymbol{b}_{nn}\}$ will be trained (or optimized) to minimize a certain cost function as, for example, the difference between $\boldsymbol{x}_{out}$ and a reference $\boldsymbol{x}_{ref}$.

If more layers are stacked so that the output of a layer is the input to following one, the NN is called ‘deep’ and it typically allows to approximate more complex functions. For a full review of ‘Deep Learning’ methods, the reader is referred to [11].

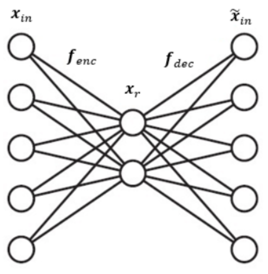

Fig. 2 The structure of a shallow undercomplete Auto-Encoder.

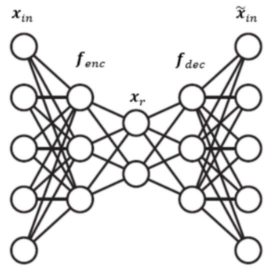

Fig. 3 The structure of a deep undercomplete Auto-Encoder

2.1.2 Auto-Encoder structure

An Auto-Encoder (AE) is a Neural Network with a particular, symmetric structure. Its 'shallow undercomplete' version is shown in Fig. 2. It is composed by an encoding function $\boldsymbol{f}_{enc}$ where 'shallow' indicates that such function is composed by a single layer of neural units while 'undercomplete' implies that $\boldsymbol{f}_{enc}$ shrinks the input $\boldsymbol{x}_{in} \in \mathbb{R}^{n_{in}}$ to $\boldsymbol{x}_r \in \mathbb{R}^{n_r}$ with $n_r < n_{in}$. Then, its specular decoding function $\boldsymbol{f}_{dec}$ aims to reconstruct the output $\tilde{\boldsymbol{x}}_{in} \in \mathbb{R}^{n_{in}}$ as close as possible to the input $\boldsymbol{x}_{in}$:

$$\boldsymbol{x}_r = \boldsymbol{f}_{act_{enc}}(\boldsymbol{W}_{enc}\, \boldsymbol{x}_{in} + \boldsymbol{b}_{enc}) \tag{5}$$

$$\tilde{\boldsymbol{x}}_{in} = \boldsymbol{f}_{act_{dec}}(\boldsymbol{W}_{dec}\, \boldsymbol{x}_r + \boldsymbol{b}_{dec}) \tag{6}$$

The network parameters $\mathscr{P}_{AE} = \{\boldsymbol{W}_{enc} \in \mathbb{R}^{n_r \times n}, \boldsymbol{b}_{enc} \in \mathbb{R}^{n_r}, \boldsymbol{W}_{dec} \in \mathbb{R}^{n \times n_r}, \boldsymbol{b}_{dec} \in \mathbb{R}^{n}\}$ are trained to minimize the difference between the output and the input:

$$\underset{\mathscr{P}_{AE}}{\operatorname{argmin}} \, ||\tilde{\boldsymbol{x}}_{in}(\mathscr{P}_{AE}) - \boldsymbol{x}_{in}||^2 \tag{7}$$

The bottleneck $n_r < n_{in}$ acts as regularization, preventing the AE from simply copying the inputs and, instead, forcing it to learn a relevant reduced parametrization.

It can be noted that if $\boldsymbol{f}_{act_{enc}}$ and $\boldsymbol{f}_{act_{dec}}$ are set as linear, the Auto-Encoder aims to reproduce PCA. In fact, the obtained AE weights will span the same subspace as the principal components; they would correspond if additional constraints are imposed on the AE such as weights tied $\boldsymbol{W}_{dec} = \boldsymbol{W}_{enc}^{\mathrm{T}}$ and orthogonal $\boldsymbol{W}_{enc}\, \boldsymbol{W}_{enc}^{\mathrm{T}} = \boldsymbol{I}^{n_r}$, where $\boldsymbol{I}^{n_r}$ is the identity matrix of dimension n_r. However, the order of the singular vectors based on singular values is, in general, not guaranteed with Auto-Encoders.

In this work we will use a deep Auto-Encoder as in Fig. 3 with non-linear activation functions and more layers to obtain a non-linear version of PCA to describe the redundant DOFs of a multibody model as a function of the identified minimal coordinates.

2.2 Minimal Coordinate multibody simulation

Here, the procedure is presented. It consists of data collection from a reference multibody simulation to train the Auto-Encoder. Then, the AE encoding function is used to obtain the Minimal Coordinate parametrization, while the AE decoding function is used to obtain the reduced-order model, perform the reduced simulation and, finally, reconstruct the full coordinates.

2.2.1 Reference simulation for AE training

In order to effectively apply the proposed method, consistent coordinates are necessary, meaning that they should consist of purely displacement coordinates rather than a combination of displacements and rotations. Thus, the starting point is a multibody model expressed in Natural Coordinates, where only position DOFs are used:

$$\boldsymbol{M}\,\ddot{\boldsymbol{x}} + \boldsymbol{f}_{spring} + \boldsymbol{f}_{damper} + \boldsymbol{\Phi}_x^{\mathrm{T}}\,\boldsymbol{\lambda} = \boldsymbol{f}_{ext} \tag{8}$$

$$\boldsymbol{\Phi}(\boldsymbol{x}) = \boldsymbol{0} \tag{9}$$

Where, given n DOFs and n_c constraints, $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of Natural Coordinates of the system and the double-dot accent indicates the second time derivative, $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is a constant mass matrix, $\boldsymbol{f}_{ext} \in \mathbb{R}^n$ is the vector of external forces, $\boldsymbol{f}_{spring}$ and $\boldsymbol{f}_{damper}$ are the forces given respectively by spring and damper elements, $\boldsymbol{\lambda} \in \mathbb{R}^{n_c}$ is the vector of Lagrange multipliers, $\boldsymbol{\Phi} \in \mathbb{R}^{n_c}$ is the constraint vector and $\boldsymbol{\Phi}_x$ is its Jacobian with respect to $\boldsymbol{x}$.

A reference simulation of such model is performed at timesteps $\mathscr{T} = \{0, \dots, t_{n_t-1}\}$ and given $\boldsymbol{x}^i = \boldsymbol{x}(\mathscr{T} = t_{i-1})$ the data are collected in the form:

$$\mathscr{X} = \{\boldsymbol{x}^1,\, \dots,\, \boldsymbol{x}^{n_t}\}, \qquad \mathscr{F}_{ext} = \{\boldsymbol{f}_{ext}^1,\, \dots, \boldsymbol{f}_{ext}^{n_t}\}$$
$$\mathscr{F}_{spring} = \{\boldsymbol{f}_{spring}^1,\, \dots, \boldsymbol{f}_{spring}^{n_t}\}, \qquad \mathscr{F}_{damper} = \{\boldsymbol{f}_{damper}^1,\, \dots, \boldsymbol{f}_{damper}^{n_t}\}$$

Each set $\mathscr{X}, \mathscr{F}_{ext}, \mathscr{F}_{spring}, \mathscr{F}_{damper} \in \mathbb{R}^{n\times n_t}$ contains n_t samples drawn from a (set of) reference simulation(s) and is stored together with the matrix $\boldsymbol{M}$.

$\mathscr{X}$ is used as input to train the Auto-Encoder parameters $\mathscr{P}_{AE}$ in order to minimize the mean squared reconstruction error:

$$\underset{\mathscr{P}_{AE}}{\operatorname{argmin}} \frac{1}{n_t} \sum_{i=1}^{n_t} \left(\tilde{\boldsymbol{x}}^i(\mathscr{P}_{AE}) - \boldsymbol{x}^i\right)^2,\ \forall \boldsymbol{x}^i \in \mathscr{X} \tag{10}$$

Sigmoid activation functions as in Eq. 4 are used in order to ensure that the AE functions have consistent derivatives as required by their evaluation in Sec. 2.2.2. The 'Root Mean Square Propagation (RMSprop)' algorithm [12], a variation of 'Stochastic Gradient Descent (SGD)', is used for the AE parameter optimization. The dimension of the AE bottleneck n_r is set equal to the known number of Minimal Coordinates of the system. The procedure returns $\boldsymbol{f}_{enc}$ that gives the MC vector $\boldsymbol{x}_r$ and $\boldsymbol{f}_{dec}$ that backprojects it to the Auto-encoder Coordinates (AC) $\tilde{\boldsymbol{x}}$:

$$\boldsymbol{x}_r = \boldsymbol{f}_{enc}(\boldsymbol{x}) \tag{11}$$

$$\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \boldsymbol{f}_{dec}(\boldsymbol{x}_r) \tag{12}$$

2.2.2 Minimal Coordinate model

The kinetic energy $\mathscr{K}$ of the system can be described as a function of the obtained Minimal Coordinates:

$$\dot{\boldsymbol{x}} \approx \frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \dot{\boldsymbol{x}}_r \tag{13}$$

$$\mathscr{K} = \frac{1}{2} \dot{\boldsymbol{x}}_r^{\mathrm{T}} \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \right)^{\mathrm{T}} \boldsymbol{M} \frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \dot{\boldsymbol{x}}_r \tag{14}$$

Leading to the following inertial forces in the case of a single minimal coordinate $\boldsymbol{x_r}$:

$$\boldsymbol{f}_{m,r} = \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \right)^{\mathrm{T}} \boldsymbol{M} \frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \ddot{\boldsymbol{x}}_r \tag{15}$$

$$\boldsymbol{f}_{g,r} = \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}} \right)^{\mathrm{T}} \boldsymbol{M} \frac{\partial^2 \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}^2} \dot{\boldsymbol{x}}_r^2 \tag{16}$$

In case of spring-damper elements in the system, their forces are modeled through additional Neural Networks. In particular, a 'NN potential function' $\boldsymbol{f}_{u,r}(\boldsymbol{x}_r)$ is built with parameters $\mathscr{P}_U$ trained to approximate the NC spring forces projected into Minimal Coordinates:

$$\underset{\mathscr{P}_U}{\operatorname{argmin}} \frac{1}{n_t} \sum_{i=1}^{n_t} \left(\frac{\partial \boldsymbol{f}^i_{u,r}(\mathscr{P}_U)}{\partial \boldsymbol{x}^i_{\boldsymbol{r}}} - \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x}^i_{\boldsymbol{r}}} \right)^{\mathrm{T}} \boldsymbol{f}^i_{spring} \right)^2 \ , \ \forall \boldsymbol{f}^i_{spring} \in \mathscr{F}_{spring} \tag{17}$$

$$\boldsymbol{x}^i_r = \boldsymbol{f}_{enc}(\boldsymbol{x}^i) \ , \ \forall \boldsymbol{x}^i \in \mathscr{X} \tag{18}$$

Similarly, the parameters $\mathscr{P}_D$ of a 'NN damper function' $\boldsymbol{f}_{d,r}(\boldsymbol{x}_r, \dot{\boldsymbol{x}}_r)$ are trained to approximate the NC damping forces projected into Minimal Coordinates:

$$\underset{\mathscr{P}_D}{\operatorname{argmin}} \frac{1}{n_t} \sum_{i=1}^{n_t} \left(\boldsymbol{f}^i_{d,r}(\mathscr{P}_D) - \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x}^i_{\boldsymbol{r}}} \right)^{\mathrm{T}} \boldsymbol{f}^i_{damper} \right)^2 , \ \forall \boldsymbol{f}^i_{damper} \in \mathscr{F}_{damper} \tag{19}$$

Hence, the procedure allows to describe the dynamics as a function of the Minimal Coordinates:

$$\boldsymbol{f}_{m,r}(\boldsymbol{x}_r,\, \ddot{\boldsymbol{x}}_r) + \boldsymbol{f}_{g,r}(\boldsymbol{x}_r,\, \dot{\boldsymbol{x}}_r) + \frac{\partial \boldsymbol{f}_{u,r}(\boldsymbol{x_r})}{\partial \boldsymbol{x_r}} + \boldsymbol{f}_{d,r}(\boldsymbol{x}_r,\, \dot{\boldsymbol{x}}_r) = \left(\frac{\partial \boldsymbol{f}_{dec}}{\partial \boldsymbol{x_r}}\right)^{\mathrm{T}} \boldsymbol{f}_{ext} \tag{20}$$

At the first timestep $i = 1$, the NC initial position $\boldsymbol{x}^{i=1}$ is reduced to $\boldsymbol{x}_r^{i=1}$ with the encoding function $\boldsymbol{f}_{enc}$ while the initial velocity and acceleration are supposed zero. The derivatives of the decoding function are calculated through automatic differentiation and used to obtain the reduced order model. The dynamics equation is integrated using a central difference scheme, obtaining $\boldsymbol{x}_r^{i=2}$ and $\tilde{\boldsymbol{x}}^{i=2} = \boldsymbol{f}_{dec}(\boldsymbol{x}_r^{i=2})$. The procedure is repeated until $i = n_t - 1$.

3 Application example

The methodology is demonstrated on the rigid model of a MacPherson suspension shown in Fig. 4. It consists of 6 bodies: a lower control arm is linked to the chassis by two spherical joints and to the steering knuckle by a spherical joint; the knuckle is linked to the tie-rod by a spherical joint and to the strut by a prismatic joint and a spring-damper element. The system is loaded through a time varying vertical force on the knuckle.

The procedure described in Section 2 is followed with the AE trained to find the mapping to a single ($n_r = 1$) Minimal Coordinate. The comparison of the dynamics between the original full coordinates and the Auto-Encoder approximation is shown in Fig. 5. These results show a relatively close match between the original NC model

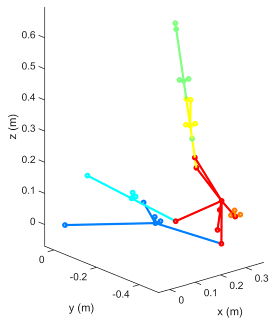

Fig. 4 The suspension model.

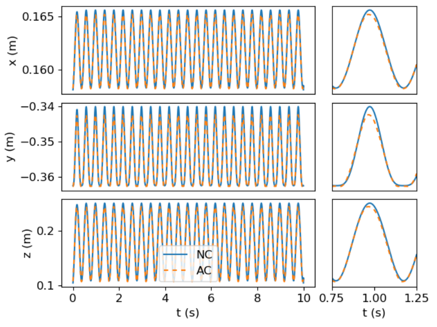

Fig. 5 On the left, comparison of the simulation for the knuckle centre of gravity in ‘Natural Coordinates (NC)’ $\boldsymbol{x}$ and ‘Auto-encoder Coordinates (AC)’ $\tilde{\boldsymbol{x}}$. On the right, close-up to show the not perfect match.

motion and the AE projected model. However, some differences exist, in particular at maximum displacement levels.

4 Conclusions

With respect to redundant Degree-Of-Freedom approaches for multibody simulations, the Minimal Coordinate formulation has some benefits such as the possibility to express the dynamics as Ordinary Differential Equations. However, it is often infeasible to set up the required analytic relations between the motion of all bodies as a function of the MC in the case of closed-loop topologies.

In this work, a machine learning approach that reduces a multibody model to Minimal Coordinates is proposed. It is based on a deep Auto-Encoder that trains a non-linear encoding function in order to retrieve the minimal parameters and an inverse decoding function that is used to describe the dynamics in the reduced space and backproject to the full coordinates. An application example is shown.

Acknowledgements The Research Fund KU Leuven, the Flanders Innovation & Entrepreneurship Agency within the ENDURANCE project and The Research Foundation - Flanders (FWO) are gratefully acknowledged for their support.

References

1. M. Hiller and A. Kecskemethy, "Dynamics of multibody systems with minimal coordinates," in *Computer-Aided Analysis of Rigid and Flexible Mechanical Systems*, pp. 61–100, Springer, 1994.
2. S. M. Issa and K. P. Arczewski, "Kinematics and dynamics of multibody system based on natural and joint coordinates using velocity transformations," *Journal of Theoretical and Applied Mechanics*, vol. 36, no. 4, pp. 905–918, 1998.
3. E. J. Haug, *Computer aided kinematics and dynamics of mechanical systems*, vol. 1. Allyn and Bacon Boston, 1989.
4. J. G. de Jalón and E. Bayo, *Kinematic and dynamic simulation of multibody systems: the real-time challenge*. Springer Science & Business Media, 2012.
5. O. Brüls, P. Duysinx, and J.-C. Golinval, "A model reduction method for the control of rigid mechanisms," *Multibody System Dynamics*, vol. 15, no. 3, pp. 213–227, 2006.
6. O. Brüls, P. Duysinx, and J.-C. Golinval, "The global modal parameterization for non-linear model-order reduction in flexible multibody dynamics," *International journal for numerical methods in engineering*, vol. 69, no. 5, pp. 948–977, 2007.
7. A. A. Shabana, *Dynamics of multibody systems*. Cambridge university press, 2013.
8. M. Vermaut, F. Naets, and W. Desmet, "A flexible natural coordinates formulation (fncf) for the efficient simulation of small-deformation multibody systems," *International Journal for Numerical Methods in Engineering*, vol. 115, no. 11, pp. 1353–1370, 2018.
9. L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.
10. D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.
11. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
12. G. Hinton, "Neural networks for machine learning. coursera,[video lectures]," 2012.