Article

Development and Implementation of an ANN Based Flow Law for Numerical Simulations of Thermo-Mechanical Processes at High Temperatures in FEM Software

Olivier Pantalé

Laboratoire Génie de Production, Institut National Polytechnique/Ecole Nationale d'Ingénieurs de Tarbes, Université de Toulouse, 47 Av d'Azereix, F-65016 Tarbes, France; olivier.pantale@enit.fr; Tel.: +33-562442933

Abstract: Numerical methods based on finite element (FE) have proven their efficiency for many years in the thermomechanical simulation of forming processes. Nevertheless, the application of these methods to new materials requires the identification and implementation of constitutive and flow laws within FE codes, which sometimes pose problems, particularly because of the strongly non-linear character of the behavior of these materials. Computational techniques based on machine learning and artificial neural networks are becoming more and more important in the development of these models and help the FE codes to integrate more complex behavior. In this paper, we present the development, implementation and use of an artificial neural network (ANN) based flow law for a GrC15 alloy under high temperature thermomechanical solicitations. The flow law modeling by ANN shows a significant superiority in terms of model prediction quality compared to classical approaches based on widely used Johnson–Cook or Arrhenius models. Once the ANN parameters have been identified on the base of experiments, the implementation of this flow law in a finite element code shows promising results in terms of solution quality and respect of the material behavior.

Keywords: ANN flow law; constitutive behavior; radial return algorithm; numerical implementation; VUHARD; GrC15; Abaqus Explicit

check for updates

Citation: Pantalé, O. Development and Implementation of an ANN Based Flow Law for Numerical Simulations of Thermo-Mechanical Processes at High Temperatures in FEM Software. *Algorithms* **2023**, *16*, 56. https://doi.org/10.3390/a16010056

Academic Editors: Xiang Zhang and Xiaoxiao Li

Received: 16 December 2022
Revised: 10 January 2023
Accepted: 12 January 2023
Published: 13 January 2023

1. Introduction

Numerical methods for simulating the behavior of structures subjected to high thermomechanical loads, as in the case of the high-temperature forming of metallic materials, are generally based on the use of commercial finite element (FE) codes, such as Abaqus, or laboratory codes, such as DynELA [1]. These FE codes are based on two types of equations: conservation equations and constitutive equations. If the first equations are well established on the basis of physics and mechanics, it is not the same for the second type of equations: the constitutive equations. Thus, in a general way, the conservation equations concern the fundamental principles of physics, such as the mass conservation law, the momentum law (fundamental equation) and the energy law (declined as the first and second principles of thermodynamics). By themselves, these laws are not sufficient to describe the behavior of a material or a structure subjected to thermomechanical solicitations because the nature of the material's behavior translated through the behavior laws is not included in the system previously proposed. Therefore, for each type of material, it is necessary to define behavior laws whose formulation is based on observation in order to describe the behavior of this material under external forces. The quality and the accuracy of the results of any numerical simulation depend on the choice of these behavior laws and on the ability of the user to identify the coefficients of these behavior laws for a given material by performing experiments under conditions close to those encountered during the real stress of the structure in service that one wishes to design [2]. Depending on the nature of the solicitations, these tests are based on quasi-static or dynamic tensile or compression tests, tests on thermomechanical simulators such as Gleeble [3] or impact tests using gas launchers or Hopkinson bars [4].

In the thermomechanical simulation of forming processes, these behavior laws define the dependence [5] of the flow stress of the material σ^y as a function of the three input variables, which are the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T of the material, so that the general form of the flow law can be written with the following expression:

$$\sigma^y = f(\varepsilon^p, \dot{\varepsilon}, T) \tag{1}$$

These laws, due to the nature of materials and the phenomena involved [6,7] (work hardening, movement of dislocations, structural hardening, phase transformations, etc.) are highly non-linear, and their validity is restricted to a certain range of strains ε, strain rates $\dot{\varepsilon}$ and temperatures T. From the observations made, we can define two main classes of behavior laws: the flow laws based on physics and the empirical flow laws. From the mechanics of continuous media and experimental tests and depending on the materials used, several flow models have been developed in the past, including the Johnson–Cook flow law [8,9], the Zerilli–Armstrong flow law [10] and their respective derived forms [11–19], the Hansel–Spittle [20,21] or the Arrhenius [22–24] flow laws, to name only a few of the most widely used in the metal-forming processes at high temperature. As an example, and because it is widely used in numerical simulation of metal forming processes, the equation that describes the Johnson–Cook flow law [8] is given as follows:

$$\sigma^y = \left(A + B\varepsilon^{p^n}\right)\left[1 + C\ln\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)\right]\left[1 - \left(\frac{T - T_0}{T_m - T_0}\right)^m\right], \tag{2}$$

where A is the initial elastic limit of the material, B is the strain hardening coefficient, n is the strain hardening exponent, and C and m are the material constants that describe the strain rate hardening coefficient and the thermal softening coefficient, respectively. The Johnson–Cook model is the most widely used because it is simple to identify and use and has few parameters to determine [25,26].

Once the choice has been made concerning the type of flow law to be used for a material, it is then necessary, from a set of experimental tests carried out in the laboratory under conditions close to those of the structure in service, to identify the parameters of these flow laws by machine learning methods based on approaches of minimization of the calculated experiment. Therefore, the use of the Johnson–Cook flow law defined by Equation (2) requires the identification of 5 material parameters.

The main problem that researchers are confronted with after the phase of realization of the experimental tests concerns the choice of the flow law to use according to the observations made on these test results. This choice of flow law is also restricted by the FE code used and the availability of such flow laws. Thus, a user of the Abaqus FE code will turn more particularly to a Johnson–Cook [8] flow law, where it is natively implemented in this software. The choice of another form of flow law, Zerilli–Armstrong, or Arrhenius, for example, obliges the user to program himself the computation of the flow stress σ^y of the material through a VUMAT subroutine in FORTRAN 77 as proposed by Gao et al. [27], Ming et al. [28] for a Johnson–Cook flow law, or Liang et al. [24] for an Arrhenius type flow law with the following expression:

$$\sigma^y = \frac{1}{\alpha(\varepsilon)}\ln\left\{\left(\frac{Z(\varepsilon)}{A(\varepsilon)}\right)^{1/n(\varepsilon)} + \sqrt{1 + \left(\frac{Z(\varepsilon)}{A(\varepsilon)}\right)^{2/n(\varepsilon)}}\right\} \tag{3}$$

with

$$Z(\varepsilon) = \dot{\varepsilon}\exp\left(\frac{Q(\varepsilon)}{RT}\right), \tag{4}$$

where Z is the Zenner–Hollomon parameter [29], $Q(\varepsilon)$ is the apparent activation energy (J mol^{-1}), R is the universal gas constant (8.314 J mol^{-1}K^{-1}). $Q(\varepsilon)$, $A(\varepsilon)$, $\alpha(\varepsilon)$ and $n(\varepsilon)$ are expressed as a function of the strain ε through polynomial functions of degree m (varying from 1 to 9), which leads to the identification of up to 36 material parameters.

Implementing the flow law as a VUMAT FORTRAN subroutine requires the computation of the derivatives $\partial\sigma^y/\partial\varepsilon^p$, $\partial\sigma^y/\partial\dot{\varepsilon}$ and $\partial\sigma^y/\partial T$ of the flow stress σ^y, which can quickly become relatively complex as the complexity of the flow law increases, i.e., the relative complexity of the Arrhenius flow law defined by Equations (3) and (4), regarding the relative simplicity of the Johnson–Cook model defined by Equation (2), one can refer to the work proposed by Liang et al. [24] for details concerning this implementation using the safe version of the Newton–Raphson method proposed by Ming et al. [28]. The choice of the flow law to use for a problem is therefore doubly guided by the behavior of the material on the one hand, but a more important aspect is the list of flow laws implemented natively in the FE code we plan to use for the numerical simulation. At this time, there is not yet a flow law generic enough to cover a wide range of material behavior that is simple to implement and use.

As we have seen in the previous paragraph, the choice of the flow law to use is guided mainly by the list of flow laws available in the finite element code used, and very often, this choice is made at the expense of the quality of the model. For example, Zhou et al. [14], proposed the identification of the flow law of a GCr15 alloy for a continuous casting bloom with heavy reduction application as introduced by Ji et al. [30], who performed compression tests on this material. In their study, Ji et al. [30] performed compression tests on GCr15 cylinders in a temperature range of 750 °C to 1300 °C in 50 °C steps, strain rates of 0.001 s^{-1}, 0.01 s^{-1} and 0.1 s^{-1} and strains up-to 0.7. The results of these compression tests, plotted in Figure 1, show a decrease in flow stress σ^y with respect to an increase in the temperature T and a increase of σ^y with respect to an increase in the strain rate $\dot{\varepsilon}$, as in most metallic materials.

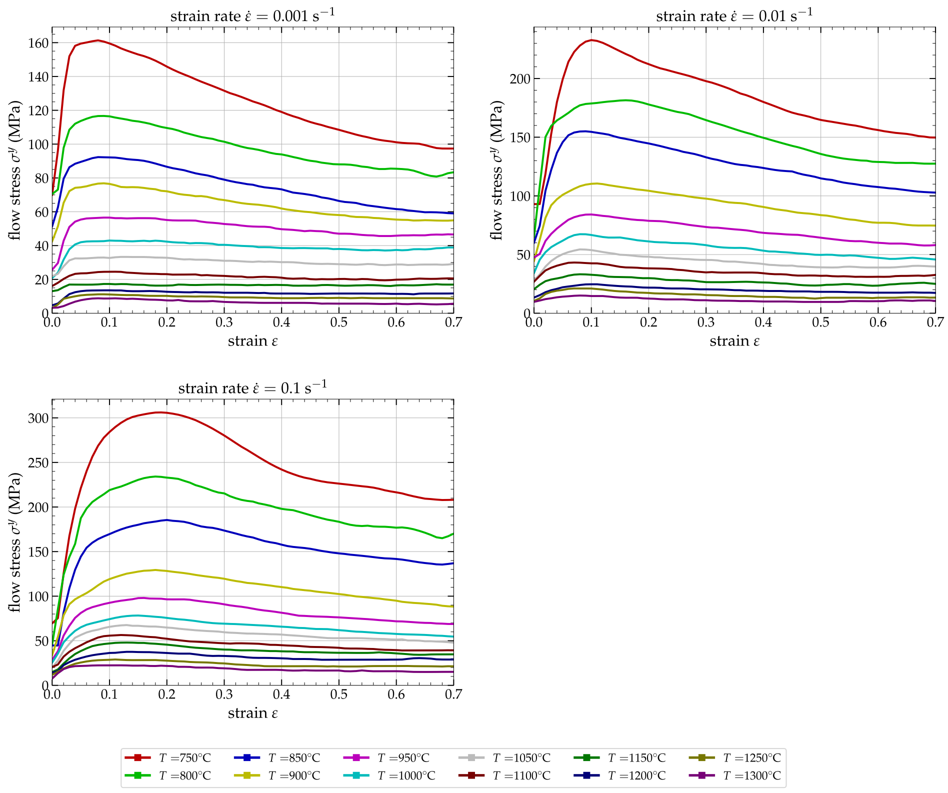


Figure 1. Original data extracted from the publication of Ji et al. [30].

The evolution of the flow stress as a function of the plastic deformation shows the presence of a dynamic recrystallization (DRX) phenomenon within the material. This phenomenon is an additional non-linearity of this type of material compared to other materials, mainly because of high temperatures and low strain rates, which should be considered when describing the material behavior. As stated in the publication of Zhou et al. [14], depending on the flow model used—Johnson–Cook, modified Zerilli–Armstrong, Arrhenius or new modified Johnson–Cook—the fidelity of considering the real behavior varies widely with the complexity of the flow model, which includes between 5 parameters for the Johnson–Cook model and 16 parameters for the Arrhenius model. Thus, and for the input data provided by Ji et al. [30], the two most common models, Johnson–Cook and Zerilli–Armstrong do not correctly describe the material behavior. Only the modified Johnson–Cook and Arrhenius models can describe correctly the behavior of the material during the compression process. Unfortunately, and this is not part of their study, if these last two models are satisfactory from a theoretical point of view, from a practical point of view for the user of a FE code such as Abaqus, it will be necessary to carry out a numerical implementation in a FORTRAN 77 VUMAT subroutine of the modified Johnson–Cook flow law or the Arrhenius law as carried out by a few authors [24,27,28] to use these laws for numerical simulation. This requires a certain expertise in the development and implementation of flow laws, which is not available to all users of the Abaqus FE code.

From this observation, and from the necessity to select a flow law for a type of material, then to identify the parameters of this flow law according to experimental tests, and finally to implement this flow law as a user subroutine in FORTRAN in the Abaqus FE code, we recently proposed in Pantalé et al. [31] an alternative approach based on the ability of artificial neural networks (ANNs) to behave as universal approximators as reported by Minsky et al. [32] and Hornik et al. [33]. In fact, artificial neural networks can solve problems that are difficult to conceptualize using traditional computational methods. Unlike a classical approach based on a regression method, an ANN does not need to know the mathematical form of the model it seeks to reproduce; hence, we do not need anymore to postulate the mathematical form of the constitutive equation to use it in a FE simulation using this kind of approach. Using a neural network instead of an analytical constitutive law can lead to a bias related to the validity of the answers according to the domain of use and the learning domain. Thus, if ANNs are efficient for the interpolation of results inside the learning domain, their behavior outside of it is not controlled. Therefore, if the input values are far from those provided during the training, the outputs can be far from the physical reality of the process. It is, of course, the same for analytical laws when modeling non-linear behavior, but if the choice of the model is made properly, based on physical considerations, they will provide results closer to reality than the ANN model. Therefore, care should be taken when using ANN-based flow laws, and the validity of the model input data should always be verified.

Implementing ANNs for plasticity in thermomechanics has been studied, and a review of the literature can be found, for example, in the work of Gorji et al. [34] concerning the use of recurrent neural networks, in that of Jamli et al. [35] concerning their application in finite element analysis of metal forming processes, or in that of Jiao et al. [36] concerning the applicability to meta-materials and their characterization. A distinction must be made between ANN-based flow models (the focus of this study) and ANN-based constitutive models. Both approaches have been studied by many researchers during the last thirty years. Ghaboussi [37] proposed an ANN-based constitutive model for concrete under monotonic biaxial loading and cyclic uniaxial loading. They extended their work by introducing adaptive and auto-progressive networks in [38,39], where the architecture of the network evolves during the learning phase to better learn the complex stress–strain behavior of the materials using a global load-deflection response, where the evaluation of the flow stress of the material computed by the ANN is combined with a radial return algorithm. Lin et al. [40] proposed an ANN to predict the flow stress of 42CrMo4 steel in hot compression tests on a Gleeble thermomechanical device and showed a very good correlation between the experimental results and the model predictions. Ashtiani et al. [41] compared the predictive capabilities of an ANN versus an analytical model for Johnson–Cook, Arrhenius, and strain-compensated Arrhenius laws and concluded that the neural network had better efficiency and accuracy in predicting the hot behavior of the Al–Cu–Mg–Pb alloy.

The underlying idea proposed in our approach is to implement a flow law described by a trained ANN as a FORTRAN 77 subroutine in the Abaqus FE code. This ANN was previously trained from the data extracted from mechanical tests of the material and can directly define the value of the flow stress σ^y as a function of the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T. After a training phase based on the use of the Python library TensorFlow [42,43], the weights and biases of the trained neural network are transcoded into a subroutine in FORTRAN 77, which is compiled and linked with the libraries of the Abaqus FE code to include the behavior of the material by allowing the computation of the flow stress σ^y as a function of ε^p, $\dot{\varepsilon}$ and T, and of its three derivatives $\partial\sigma^y/\partial\varepsilon^p$, $\partial\sigma^y/\partial\dot{\varepsilon}$ and $\partial\sigma^y/\partial T$.

The structure of this paper is as follows: Section 2 addresses the presentation of a neural-network-based flow law and its training from the data proposed by Ji et al. [30] and reported in Figure 1. The comparison of several neural network architectures regarding accuracy and implementation complexity will be presented and compared. In Section 3, we will present the transposition of this neural network into a FORTRAN 77 subroutine for the Abaqus FE code. Validation is based on the numerical simulation Abaqus Explicit FE code of a compression test in the same configuration as the one proposed by Ji et al. [30] using four different ANN flow laws. In this Section, we will present the problems of overfitting the neural network and its visible consequences on the results concerning numerical simulations. Finally, a conclusion and perspective section will conclude this paper.

2. Training of the ANN Flow Law

In this section, we briefly recall, as an introduction, some basic principles of artificial neural networks that apply to this work. The global architecture chosen to model the behavior of a material is based on a multi-layer feed-forward ANN, which, as proposed by Hornik et al. [33], can be used as a universal approximator. The architecture retained for this study concerns a neural network with two hidden layers containing a variable number of neurons on these two layers, 3 input nodes corresponding to the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T, respectively, and a single output node for the flow stress σ^y of the material. Figure 2 shows a graphical representation of the global architecture of this neural network.

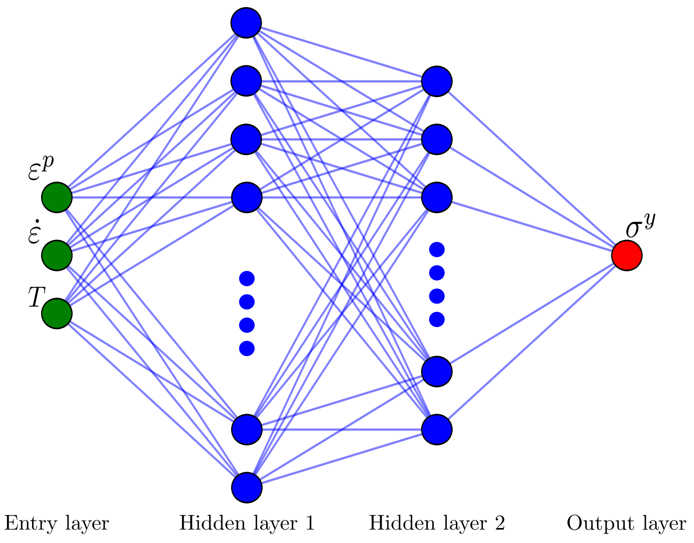


Figure 2. Global structure of the ANN flow law with two hidden layers, 3 input neurons (ε^p, $\dot{\varepsilon}$, T) and one output neuron σ^y.

The choice of the number of neurons in the two hidden layers is free, but must be reasonable. Indeed, the more neurons the network contains, the more it will reproduce faithfully the training data, but the less it will generalize to new data (the classical problem of the over-learning of neural networks). Moreover, the more neurons it contains, the more complex its mathematical structure will be, and the more computation time it will require for propagating the data inside of it within the routine included in the FE code. It is therefore necessary to respect a balance between the capacity of the network to minimize errors during the learning phase, its complexity and the computing CPU time once it is transcribed into the FE code.

2.1. Neural Network Governing Equations

According to Figure 2, the proposed neural network has 3 inputs (referred as the input vector $\overrightarrow{x}$) corresponding to the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T, respectively. These inputs are first normalized within the range $[0,1]$ to avoid an ill-conditioning of the system as presented by many other authors in the literature [40,44] since these three variables represent different physical data with very different amplitudes (0.7 for the plastic strain, 100 s^{-1} for the strain rate and 550 °C for the temperature in the case of the training data reported in Figure 1). Therefore, the three components of the input vector $\overrightarrow{x}$ are coming from the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T using the following expressions:

$$\overrightarrow{x} = \begin{cases} x_1 = \frac{\varepsilon^p - [\varepsilon^p]_{min}}{[\varepsilon^p]_{max} - [\varepsilon^p]_{min}} \\ x_2 = \frac{\ln(\dot{\varepsilon}/\dot{\varepsilon}_0) - [\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)]_{min}}{[\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)]_{max} - [\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)]_{min}} \\ x_3 = \frac{T - [T]_{min}}{[T]_{max} - [T]_{min}} \end{cases}, \tag{5}$$

where $[\,]_{min}$ and $[\,]_{max}$ are the boundaries of the range of the corresponding field. Concerning the strain rate $\dot{\varepsilon}$, and considering that its amplitude in a real case can reach 10^5 s^{-1}, as proposed in Pantalé et al. [31], we chose initially to substitute $\ln(\dot{\varepsilon}/\dot{\varepsilon}_0)$, with $\dot{\varepsilon}_0$ equal to the lowest strain rate test, for the value of $\dot{\varepsilon}$. After normalization, these three input variables are introduced into the neural network and are propagated within it by the feed-forward propagation mechanism.

Conforming to the structure of the ANN reported in Figure 2 any hidden layer k, containing n neurons, takes a weighted sum of the outputs $\overrightarrow{\hat{y}}^{(k-1)}$ of the immediately previous layer $(k-1)$, containing m neurons, given by the following equation:

$$y_i^{(k)} = \sum_{j=1}^{m} w_{ij}^{(k)} \hat{y}_j^{(k-1)} + b_i^{(k)}, \tag{6}$$

where $y_i^{(k)}$ is the entry of the ith neuron of layer k, $\hat{y}_j^{(k-1)}$ is the output of the jth neuron of layer $(k-1)$, $w_{ij}^{(k)}$ is the associated weight parameter between the ith neuron of layer k and the jth neuron of layer $(k-1)$ and $b_i^{(k)}$ is the associated bias of the ith neuron of layer k. Those weights w_{ij} and bias b_i, for each layer, are the training parameters of the ANN that we have to adjust during the training process. For the proposed model, we selected the sigmoid activation function so that each neuron in the hidden layer k provides an output value $\hat{y}$ from the input value y of the same neuron defined by Equation (6) according to the following equation:

$$\hat{y} = \frac{1}{1 + e^{-y}} \tag{7}$$

According to Equations (6) and (7), the output of each of the two hidden layers ($\overrightarrow{y}_1$ for the first hidden layer and $\overrightarrow{y}_2$ for the second hidden layer) are given by the following two equations:

$$\overrightarrow{y}_1 = \left[1 + \exp\left(-\mathrm{w}_1 \cdot \overrightarrow{x} - \overrightarrow{b}_1\right)\right]^{-1} \tag{8}$$

$$\overrightarrow{y}_2 = \left[1 + \exp\left(-\mathrm{w}_2 \cdot \overrightarrow{y}_1 - \overrightarrow{b}_2\right)\right]^{-1} \tag{9}$$

Then we compute the output s of the ANN from the output vector of the second hidden layer $\overrightarrow{y}_2$ using the following equation:

$$s = \overrightarrow{w}^T \cdot \overrightarrow{y}_2 + b \tag{10}$$

Finally, since no activation function is used for the output neuron of the ANN as is usually done in regression ANN, the flow stress σ^y can be obtained from the output s using the following equation:

$$\sigma^y = ([\sigma]_{max} - [\sigma]_{min})s + [\sigma]_{min} \tag{11}$$

2.2. Computation of the Derivatives of the Neural Network

As introduced in Section 1, implementing a flow law in a FE code requires both the computation of the flow stress σ^y as a function of the input data, performed using the previous Equations (5)–(11), but also the evaluation of the three derivatives of σ^y with respect to the input data to use a Newton–Raphson algorithm within the stress integration scheme, as proposed by many authors [24,28,45,46] based on the radial return algorithm in the Abaqus FE code. It is, therefore, necessary to perform a numerical evaluation of these three derivatives based on the ANN to obtain these quantities. It seems obvious that it is not possible to train a neural network to evaluate these values of derivatives insofar as the training data are not physically collectible data during the experimental tests. It is, therefore, necessary to predict these derivatives from the neural network architecture itself. One straightforward, but not recommended, solution to this problem is to compute numerically the derivative of σ^y with respect to ε^p, $\dot{\varepsilon}$ and T using the following relation:

$$\frac{\partial \sigma(x)}{\partial x} = \frac{\sigma(x + \delta x) - \sigma(x)}{\delta x}, \tag{12}$$

where δx is a small increase ($\delta x = 10^{-6}$ for example) applied to one of the 3 variables ε^p, $\dot{\varepsilon}$ and T. As reported in [31], we need to compute a result from the ANN 4 times to compute the flow stress and approximate the three derivatives, which is quite time consuming. The solution for this study consists, insofar as the architecture of the neural network is known through Equations (6)–(10), in analytically deriving the output s of the network with respect to the input $\overrightarrow{x}$, then integrating the data normalization operations defined by Equations (5) and (11). Given Equations (5)–(11), we can then establish in the case of a neural network containing two hidden layers and a sigmoid activation function on the two hidden layers that the derivative of σ^y with respect to the input data ε^p, $\dot{\varepsilon}$ and T is given by the following procedure.

- First, we compute the internal terms of the ANN to compute the derivative of the ANN with respect to the input vector $\overrightarrow{x}$:

$$\begin{cases} \overrightarrow{z}_1 = \exp\left(-\mathrm{w}_1 \cdot \overrightarrow{x} - \overrightarrow{b}_1\right) \\ \overrightarrow{z}_2 = \exp\left(\mathrm{w}_2 \cdot \frac{1}{1+\overrightarrow{z}_1} + \overrightarrow{b}_2\right) \\ \overrightarrow{z}_3 = \overrightarrow{w} \circ \frac{\overrightarrow{z}_2}{(1+\overrightarrow{z}_2)^2} \\ \overrightarrow{z}_4 = \frac{\overrightarrow{z}_1}{(1+\overrightarrow{z}_1)^2} \end{cases}, \tag{13}$$

where $\circ$ is the element-wise product, known as the Hadamard product, which is a binary operation that takes two matrices A and B of the same dimensions and produces another matrix C of the same dimension as the operands, where each element $C_i = A_i\, B_i$.

- Then, from the two terms $\overrightarrow{z}_3$ and $\overrightarrow{z}_4$, we can therefore compute the three derivatives of the output s with respect to the input vector $\overrightarrow{x}$ with the following equation, where $\overrightarrow{s}'$ is a vector of 3 components containing the 3 derivatives $\partial s/\partial \varepsilon^p$, $\partial s/\partial \dot{\varepsilon}$ and $\partial s/\partial T$:

$$\overrightarrow{s}' = \mathsf{w}_1^T \cdot \left[\left(\mathsf{w}_2^T \cdot \overrightarrow{z}_3\right) \circ \overrightarrow{z}_4\right] \tag{14}$$

- Finally, from Equation (14) and conforming to the normalization of the inputs introduced earlier, one can obtain the 3 derivatives of the yield stress σ^y with respect to the three inputs ε^p, $\dot{\varepsilon}$ and T using the following final equation:

$$\begin{cases} \partial\sigma/\partial\varepsilon^p = s_1' \frac{[\sigma]_{max} - [\sigma]_{min}}{[\varepsilon^p]_{max} - [\varepsilon^p]_{min}} \\ \partial\sigma/\partial\dot{\varepsilon} = \frac{s_2'}{\dot{\varepsilon}} \frac{[\sigma]_{max} - [\sigma]_{min}}{[\dot{\varepsilon}]_{max} - [\dot{\varepsilon}]_{min}} \\ \partial\sigma/\partial T = s_3' \frac{[\sigma]_{max} - [\sigma]_{min}}{[T]_{max} - [T]_{min}} \end{cases} \tag{15}$$

Equations (13)–(15) define the derivatives of the yield stress σ^y with respect to ε^p, $\dot{\varepsilon}$ and T, as computed by the ANN, and, as shown in [31], these derivatives can be used for the numerical implementation of the ANN constitutive law in a FE code.

2.3. Training of the Neural Networks

In neural network learning, it is necessary to define the objective function to be minimized and the evaluation of the model error. In this study, the error evaluation is based on the mean square error ($\mathrm{E_{MS}}$) and the root mean square error ($\mathrm{E_{RMS}}$) given by the following equation:

$$\mathrm{E_{RMS}}(\mathrm{MPa}) = \sqrt{\mathrm{E_{MS}}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\Box_i^e - \Box_i^y\right)^2}, \tag{16}$$

where N is the total number of numerical training data used, $\Box_i^y$ is the ith value predicted by the neural network, and $\Box_i^e$ is the corresponding experimental value coming from the experimental tests. The accuracy and predictive ability of the models is assessed by the mean absolute relative error ($\mathrm{E_{MAR}}$) defined by Equation (17):

$$\mathrm{E_{MAR}}(\%) = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\Box_i^y - \Box_i^e}{\Box_i^e}\right| \times 100 \tag{17}$$

The numerical implementation of the learning phase of the neural network was done in Python language, using the TensorFlow library [42,43]. The minimization procedure of the objective function is based on the use of the adaptive moment estimation (ADAM) solver proposed by Kingma et al. [47].

The training data used in this section are taken from the publication of Ji et al. [30]. Thus, data for which compression tests were performed for the 3 strain rates $\dot{\varepsilon} = 0.001\ \mathrm{s}^{-1}$, $\dot{\varepsilon} = 0.01\ \mathrm{s}^{-1}$ and $\dot{\varepsilon} = 0.1\ \mathrm{s}^{-1}$, and the 12 temperature values between 750 °C and 1300 °C in 50 °C steps are used. For each pair of data ($\dot{\varepsilon}$, T), we have a record of 71 values of flow stress σ^e corresponding to values of deformation between 0 and 0.7, regularly spaced of 0.01. We use a database of 2556 quadruplets of values (ε^p, $\dot{\varepsilon}$, T, σ^e) for the training of the ANNs.

The set of these data is used as training data for the neural network. Several neural network architectures have been studied in this work; they differ from each other by the number of neurons present in the two hidden layers. Among them, we selected 4 different architectures named 3-7-4-1, 3-9-4-1, 3-9-7-1 and 3-15-7-1 for which the name 3-*n*-*m*-1 translates an ANN with 2 hidden layers, having n neurons on the first layer and m neurons on the second layer.

All models have been trained for the same number of iterations (50,000 iterations), and around 50 min of training on a Dell XPS-13 7390 laptop running Ubuntu 22.04 LTS 64 bits with 16 GB of RAM and an Intel 4-core i7-10510U processor allow obtaining the converged parameters of the ANN models. Figure 3 shows the evolution of the training error defined by the $\log_{10}$ of the mean square error ($\log_{10}[\mathrm{E_{MS}}]$) during the training phase.

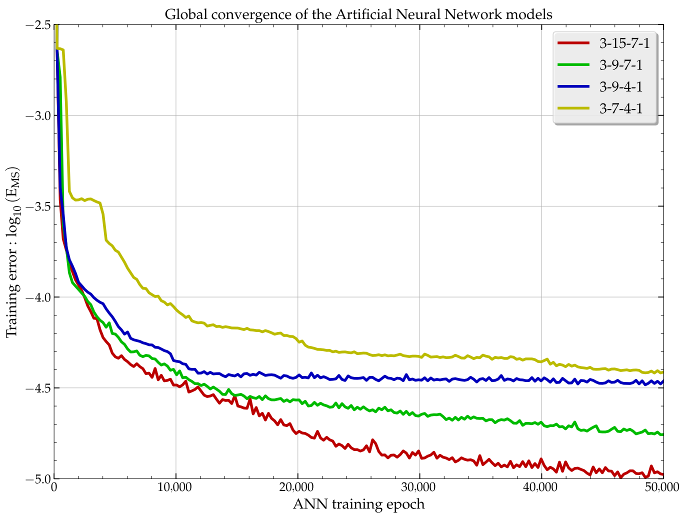


Figure 3. Convergence of the ANN models during the training phase.

As we can see on this figure, after 50,000 iterations, we can consider that we have reached a stationary state of the model learning and that it is useless to continue the learning phase. As expected, the more neurons the model contains, the more it can follow the non-linear evolution of the material's behavior and therefore the more the mean square error ($\mathrm{E_{MS}}$) during the learning phase decreases. Table 1 shows the main results of the training of these neural networks.

Table 1. Results concerning the training of the four ANN flow laws.

ANN	n_v	t **(min)**	$\mathrm{E_{MS}}$ $\times 10^{-5}$	$\mathrm{E_{MAR}}$ **(%)**	$\mathrm{E_{RMS}}$ **(MPa)**
3-7-4-1	65	48	3.91	1.88	3.05
3-9-4-1	81	48	3.29	1.70	2.75
3-9-7-1	114	49	1.83	1.25	2.44
3-15-7-1	180	50	1.01	0.97	2.30

It can be noted that the number of internal variables n_v of the networks varies from 65 to 180 for the most complex and the most powerful one, but that this complexity has no real influence on the learning time t which oscillates around a value of 50 min, regardless of the architecture chosen. Concerning the internal accuracy $\mathrm{E_{MS}}$ of the network, it varies in proportions in accordance with the graphical representation of Figure 3. From the plot in Figure 3 and the results reported in Table 1, the user would normally be tempted to select the most complex model, namely 3-15-7-1, as it gives the smallest deviation between predicted and experimental values of flow stress σ^y. This will be analyzed in the next section concerning the numerical simulation of the compression of a cylinder on the Abaqus Explicit software using the ANN flow law.

Since, as reported in Figure 3, the 3-7-4-1 model seems to have converged after the training phase, we are going to compare it with the 'accurate' model, the 3-15-7-1 model. From a more physical point of view, Figures 4 and 5 show the correlation between the data predicted by the neural network and the experimental data for the 3-7-4-1 and 3-15-7-1 networks, respectively.

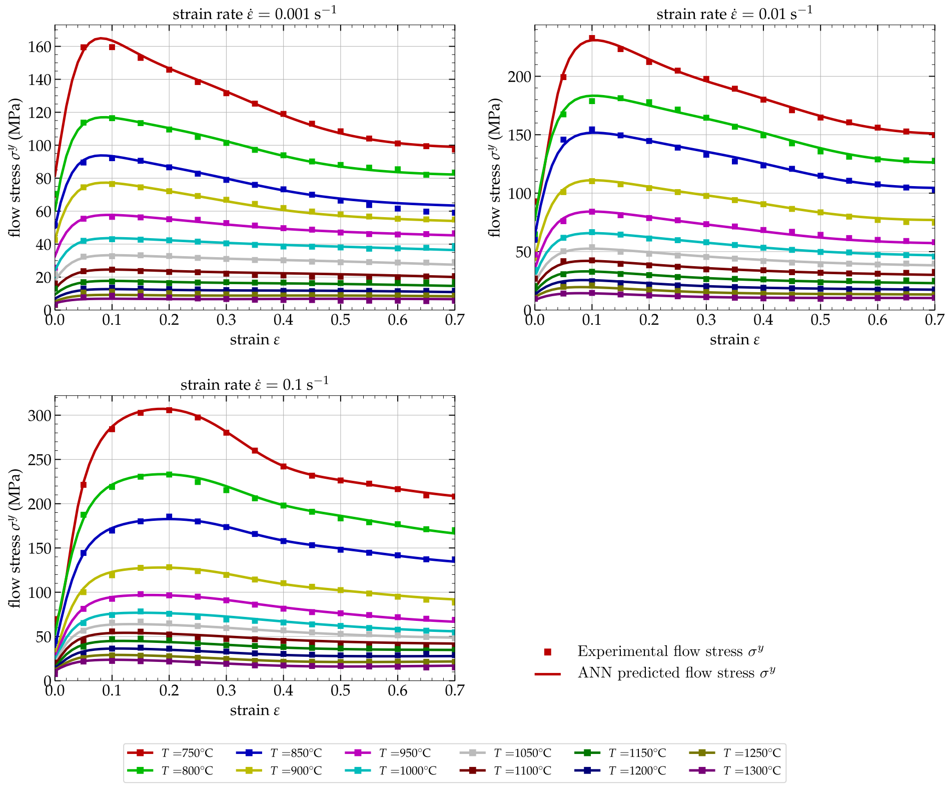


Figure 4. Comparison of the flow stress σ^y predicted by the 3-7-4-1 ANN (continuous line) and the experimental data for the GCr15 (square markers).

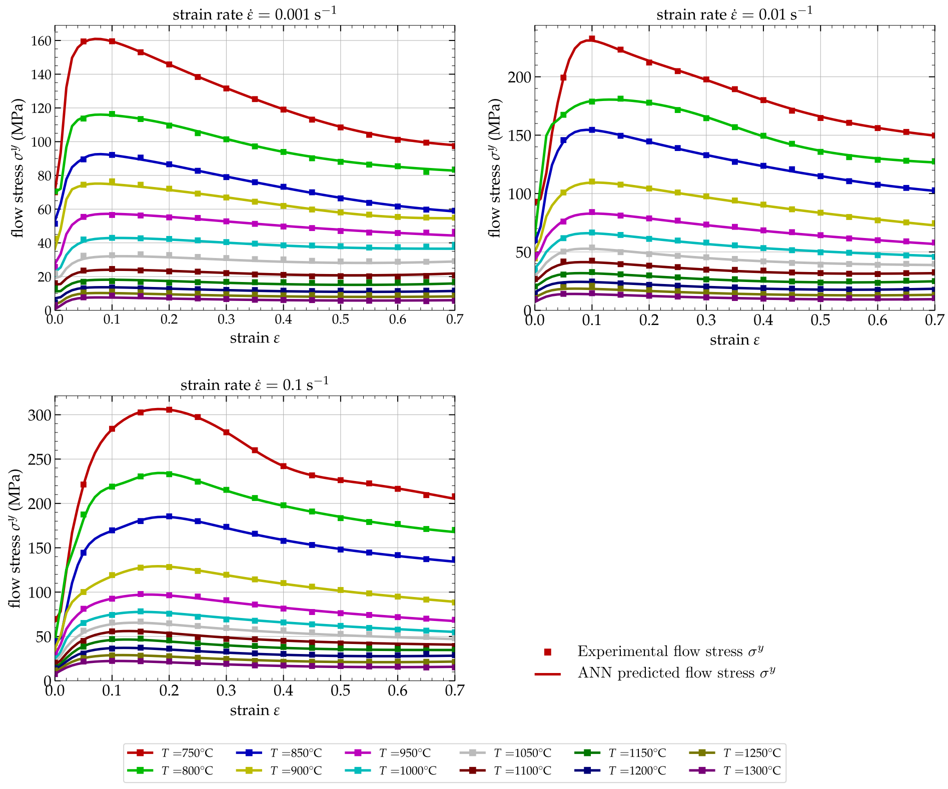


Figure 5. Comparison of the flow stress σ^y predicted by the 3-15-7-1 ANN (continuous line) and the experimental data for the GCr15 (square markers).

Analysis of Figures 4 and 5 shows a very good correlation between the ANN results and the experimental results, which is reflected by the very low values of the E_{MAR} and E_{RMS} coefficients reported in Table 1. As reported by Phaniraj [48], the correlation coefficient (R) is generally not a good measure in our case of study because it only shows the correlation of the model with respect to the data and not its accuracy, which is a determining factor in the qualification of a model. Therefore, this type of coefficient is not used in this work for comparing the different models.

Concerning the performance of the ANN flow laws, the correlation results for both reported models, are much better than the ones obtained by Zhou et al. [14] during his work on the same material with four different analytical flow laws, especially since he had to split the data into two groups according to the temperature value (one on the range $T = 750$–850 °C and one on the range $T = 850$–1300 °C) and to identify two sets of parameters for each flow law to reduce the error of his identified analytical models. This of course raises the question of the usability of those analytical laws where the temperature of the material changes from one group to the other during a thermomechanical transformation.

In our approach and by using an ANN flow law, the identified law is not only valid over the whole temperature range, but it displays a E_{MAR} value 5 times lower than the best flow law proposed by Zhou et al. [14]: the Arrhenius law with an $E_{MAR} = 3.74\%$ over the range $T = 750$–850 °C and $E_{MAR} = 5.76\%$ over the range $T = 850$–1300 °C, while the $E_{MAR} = 0.97\%$ for the 3-15-7-1 and $E_{MAR} = 1.88\%$ for the 3-7-4-1 ANN flow laws proposed here.

The disadvantage of developing a flow law model based on neural networks is the number of internal variables in the network (180 in the case of the 3-15-7-1 network), which makes it difficult to translate the network into printable results. Using a Johnson–Cook-type flow law, for example, allows the reader to quickly get an idea of the law, where the analytical formulation of the law is known to the users, and the behavior of a material is based on the knowledge of only 5 internal parameters to be identified, which makes it easy to publish in a table. Concerning an Arrhenius law, this task becomes a little more complex, as one can have from 24 to 36 coefficients. However, in our case, the publication of the 65 coefficients of the 3-7-4-1 model or the 180 coefficients of the 3-15-7-1 model makes this task delicate. As an illustration, we provide in the Appendix A all the coefficients of the 3-7-4-1 model identified during this study.

Once the identification phase is complete, it is now necessary to transpose this ANN model into a subroutine in FORTRAN or C++ that can be used by a FE code, such as Abaqus (for the FORTRAN 77 version) or DynELA (for the C++ version not presented in this paper). This is the main topic of the next section.

3. ANN Flow Law Implementation in FE Software

Once the neural network is trained as presented in Section 2.3, it can be used in a finite element code for the numerical simulation of a structure subjected to thermomechanical loading. This requires the extraction of the internal variables of the neural network and their transfer as a subroutine in FORTRAN 77 based on equations proposed in Sections 2.1 and 2.2.

3.1. Implementation of the ANN Flow Law

If we refer to the general flowchart of a finite element code as shown in Figure 6, integrating the flow law described by the ANN concerns the computation of the stress tensor σ_1 at the end of an increment, in the yellow rectangle on the figure.

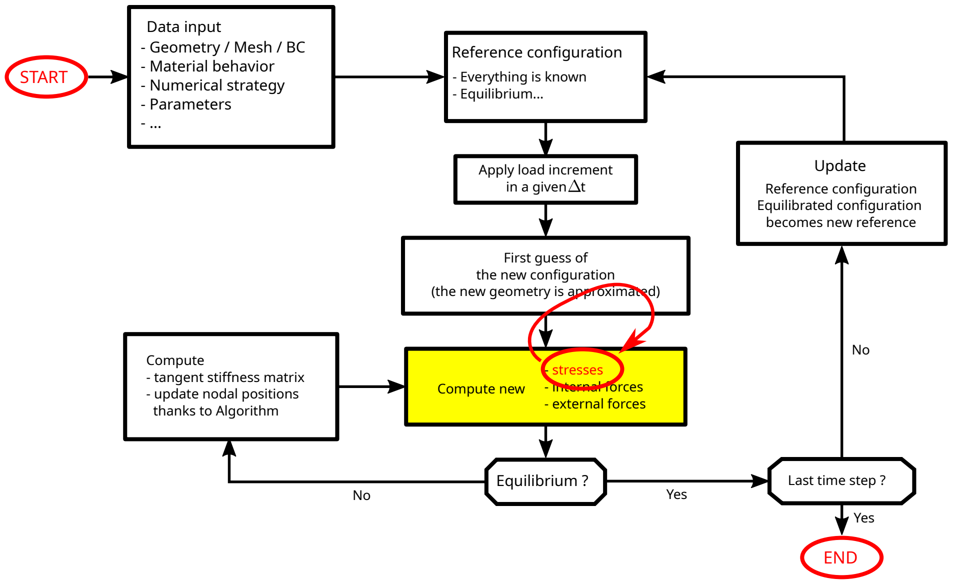


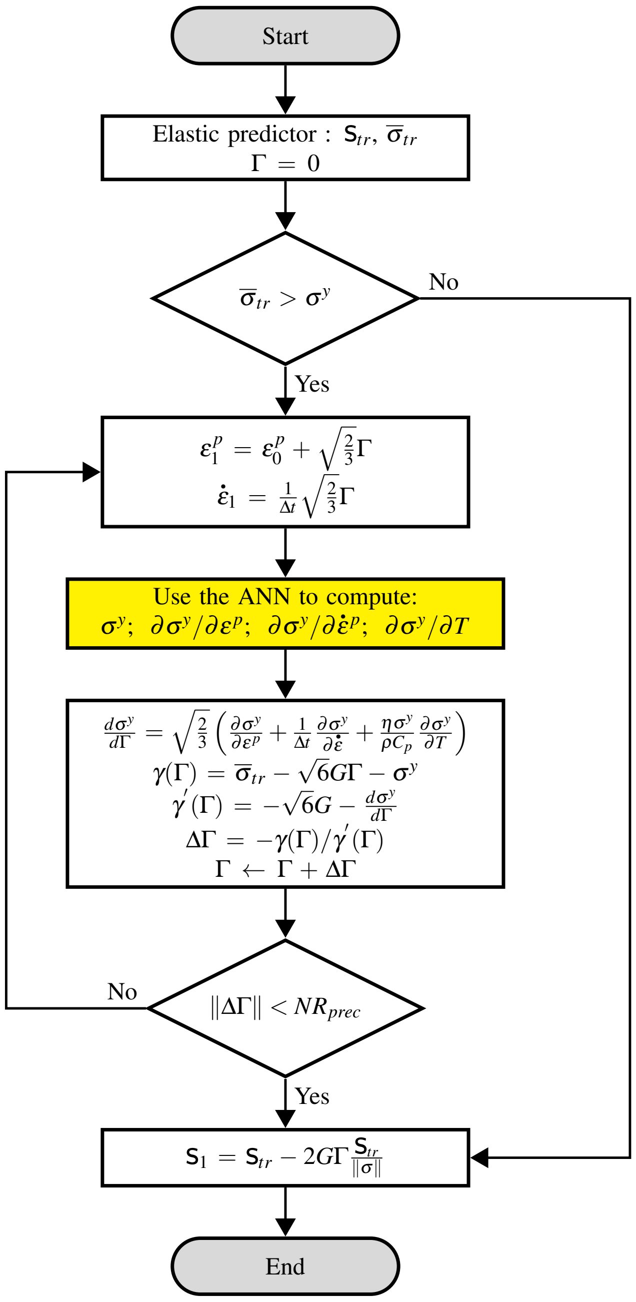
Figure 6. General flowchart of a FE code, focus on the stresses computation using an iterative solving procedure.

Within the framework of a FE formulation in large deformations such as the one used in thermomechanical modeling of processes, this computation of the stress tensor σ_1 is to be carried out on all the integration points of each element of the studied structure. Since the numerical model can include thousands of elements, themselves comprising between 1 and 8 integration points depending on the elements used, this stress computation must be as fast as possible in order not to increase the CPU time too much, but precise enough so that the results are under the physics of the process. This is even more important if we want to integrate this flow law in an explicit FE code, such as Abaqus/Explicit, for which one second of physical simulation corresponds to several million iterations of these stress computations. Thus, the complexity of the ANN, i.e., the number of computational steps that must be performed to compute the flow stress as a function of the input variables, is a major parameter in the choice of the neural network. As an example, for the model presented in Section 2.3, 180 internal variables, 15 neurons on the first hidden layer and 7 neurons on the second hidden layer were listed. Given the equations described in Sections 2.1 and 2.2, it will be necessary to compute 22 exponentials, to make matrix–vector products of size 15×3 and 15×7 plus many other numerical operations to compute the flow stress σ^y and the 3 derivatives of it with respect to ε^p, $\dot{\varepsilon}$ and T.

Implementing the ANN flow law identified above is realized here in a VUHARD subroutine, similarly as proposed by van Rensburg et al. [49], used by the Abaqus Explicit FE code in order to allow a user to program the computation of the flow stress σ^y and its 3 derivatives as a function of the model input data. This subroutine is used when calculating the stress tensor σ_1 at the end of an increment from the stress tensor at the beginning of the increment σ_0, the deformations, the material parameters and the history of the deformation at each finite element integration points, according to the stress integration algorithm based on the radial return method as described in Simo et al. [46] for the general aspects, Ming et al. [28] for Abaqus Explicit FE code, or Pantalé et al. [1] for the DynELA FE code. Thus, without going into too much detail about the stress integration scheme used in finite element codes (the curious reader can refer to [1,24,28,45] for details about this method), Figure 7 shows the location of the VUHARD subroutine used to compute the flow stress σ^y and its derivative $\partial\sigma^y/\partial\Gamma$ used in the writing of the two quantities $\gamma(\Gamma)$ and $\gamma'(\Gamma)$ used in the Newton–Raphson solving procedure from the following relation:

$$\frac{d\sigma^y}{d\Gamma} = \sqrt{\frac{2}{3}}\left(\frac{\partial\sigma^y}{\partial\varepsilon^p} + \frac{1}{\Delta t}\frac{\partial\sigma^y}{\partial\dot{\varepsilon}^p} + \frac{\eta\sigma^y}{\rho C_p}\frac{\partial\sigma^y}{\partial T}\right), \tag{18}$$

where Γ is the consistency parameter used in the radial return algorithm as defined by Simo et al. [46], Δt is the time increment, η is the Taylor–Quinney coefficient defining the amount of plastic work converted into heat energy, C_p is the specific heat coefficient, ρ is the density of the material and $\partial\sigma^y/\partial\varepsilon^p$, $\partial\sigma^y/\partial\dot{\varepsilon}$ and $\partial\sigma^y/\partial T$ are the three derivatives defined in Equation (15).



Start
Elastic predictor : S_{tr}, $\overline{\sigma}_{tr}$
$\Gamma = 0$
$\overline{\sigma}_{tr} > \sigma^y$
No
Yes
$\varepsilon_1^p = \varepsilon_0^p + \sqrt{\frac{2}{3}}\Gamma$
$\dot{\varepsilon}_1 = \frac{1}{\Delta t}\sqrt{\frac{2}{3}}\Gamma$
Use the ANN to compute:
σ^y; $\partial\sigma^y/\partial\varepsilon^p$; $\partial\sigma^y/\partial\dot{\varepsilon}^p$; $\partial\sigma^y/\partial T$
$\frac{d\sigma^y}{d\Gamma} = \sqrt{\frac{2}{3}}\left(\frac{\partial\sigma^y}{\partial\varepsilon^p} + \frac{1}{\Delta t}\frac{\partial\sigma^y}{\partial\dot{\varepsilon}} + \frac{\eta\sigma^y}{\rho C_p}\frac{\partial\sigma^y}{\partial T}\right)$
$\gamma(\Gamma) = \overline{\sigma}_{tr} - \sqrt{6}G\Gamma - \sigma^y$
$\gamma'(\Gamma) = -\sqrt{6}G - \frac{d\sigma^y}{d\Gamma}$
$\Delta\Gamma = -\gamma(\Gamma)/\gamma'(\Gamma)$
$\Gamma \leftarrow \Gamma + \Delta\Gamma$
No
$\|\Delta\Gamma\| < NR_{prec}$
Yes
$\mathsf{S}_1 = \mathsf{S}_{tr} - 2G\Gamma\frac{\mathsf{S}_{tr}}{\|\sigma\|}$
End

Figure 7. General flowchart of the radial return algorithm to compute the final stress tensor σ_1.

3.2. Numerical Simulations and Benchmarks Tests

To validate the proposed approach and to compare the different neural network architectures proposed in Section 2.3, we propose here to simulate on the Abaqus Explicit FE code the high-temperature compression of a cylinder on a Gleeble-type thermomechanical device. We consider a cylinder in compression with an initial diameter $d_0 = 8$ mm and an initial height $h_0 = 12$ mm made of GCr15 material, for which the final height after compression is $h = 6$ mm, which is a reduction of 50% of its total height. The compression of the sample is done in 10 s so that the strain rate $\dot{\varepsilon}$ is in the corresponding range of the characterization of the material behavior defined by Ji et al. [30]. Concerning the flow laws, the 4 models presented in Section 2.3 will be used and compared between them. Unfortunately, it is not possible here to compare these results with the experimental results, even if the initial shape of the specimen is the same, as these are not available in the references of the work of Zhou et al. [14] or Ji et al. [30].

The mesh of the sample is made with 850 axis-symmetric quadrilateral finite elements with 4 nodes and reduced integration (named CAX4R in the Abaqus software) with 50 elements in the vertical direction and 17 elements in the radial direction, respectively. The cylinder is between two rigid surfaces, and the Coulomb friction law with a friction coefficient at the contact surfaces was set to $\mu = 0.15$. The simulation time being fixed at 10 s in order to reduce the simulation time, considering that an explicit integration scheme is used, a global mass scaling is used for all simulations. The VUHARD subroutine is compiled using the GNU gfortran 11.3.0 and linked to the main Abaqus Explicit executable. All benchmarks tests were solved using Abaqus Explicit 2022 on a Dell XPS 13 laptop running Ubuntu 20.04 64 bits with 16 GiB of RAM and one 4 core i7-10510U Intel Processor. All computations were performed using the double precision option of Abaqus, with parallel threads execution on two cores.

Figure 8 shows the plastic strain field ε^p contourplot within the structure at the end of the simulation for both the 3-7-4-1 (left side) and the 3-15-7-1 (right side) flow laws, while Figure 9 shows the temperature field T contourplot for the same models.

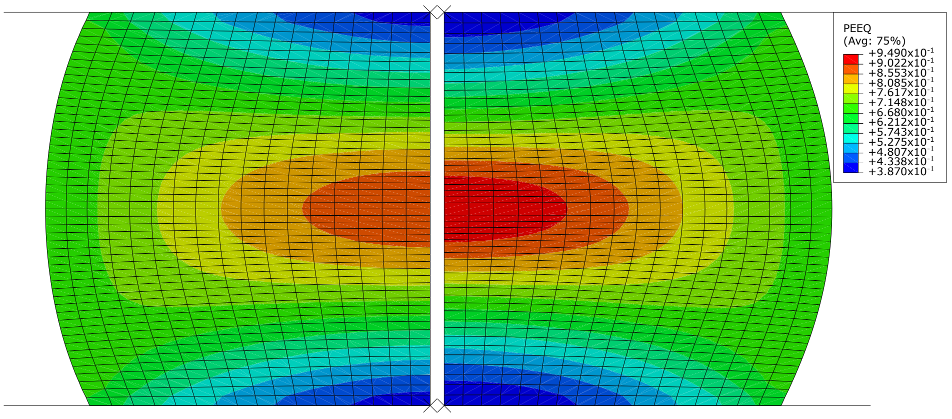


Figure 8. Equivalent plastic strain ε^p contourplot for the compression of a cylinder using the 3-7-4-1 (**left side**) and the 3-15-7-1 (**right side**) ANN flow laws.

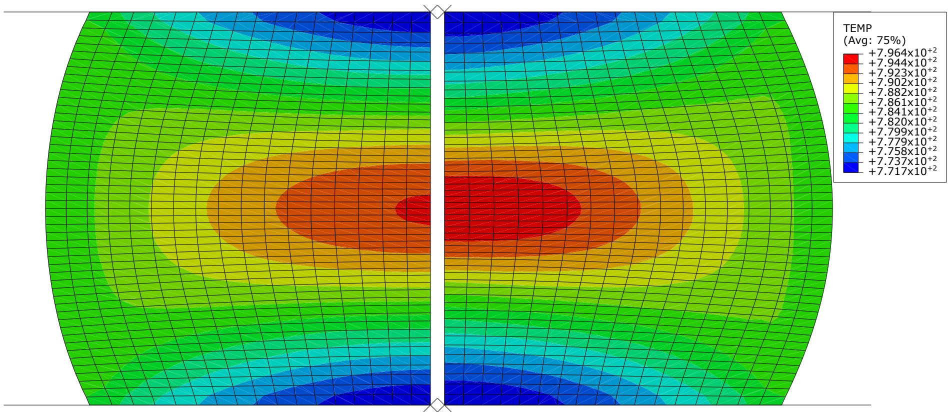


Figure 9. Temperature T contourplot for the compression of a cylinder using the 3-7-4-1 (**left side**) and the 3-15-7-1 (**right side**) ANN flow laws.

Both sides of the figures look more or less the same with some visible differences from the left to the right concerning the shape of the isovalues zones and the maximum value, but in fact, the two models with the lowest and the highest number of neurons give coherent results concerning for the plastic strain ε^p and temperature T contourplots. The maximum plastic strains are concentrated in the center of the specimen with a maximum value of $\varepsilon^p = 0.89$ for the 3-7-4-1 model and $\varepsilon^p = 0.95$ for the 3-15-7-1 model, which is slightly beyond the limit set by the training data, which varies from 0 to 0.7. As shown by Pantalé et al. [31], the flow laws defined by neural networks are able to correctly extrapolate the flow stresses σ^y when the plastic deformations are higher than at least 150% of the maximum plastic strain used during training. Concerning the temperature T, the maximum value is around $T = 795$ °C and $T = 799$ °C, which is very close and in accordance to the experiments used for the learning phase.

Figure 10 shows the evolution of the maximum radius r of the cylinder (measured at the middle of the sample height) as a function of the vertical displacement of the top of the cylinder.

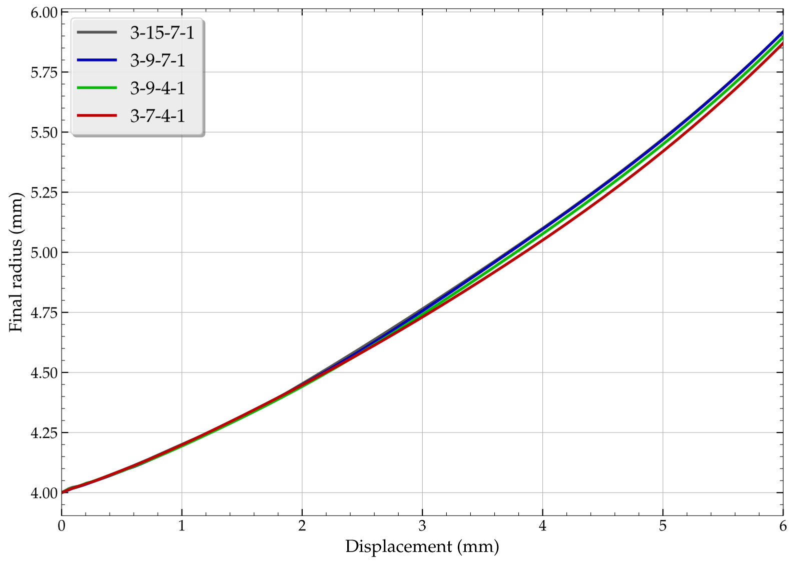


Figure 10. Evolution of the external radius r of the specimen during the compression process using the four ANN flow laws (only the 3-15-7-1 model differs).

This figure shows a slight difference of the four models during the numerical simulation. In the rest of this section, the four models will be referred to as M_{1234}. The final value therefore differs from $r = 5.870$ mm for M_1 to $r = 5.917$ mm for M_{34}.

Table 2 gathers the results allowing the comparison of the four identified flow laws.

Table 2. Compression of a cylinder using the four ANN flow laws, results for the center element of the structure.

Model	**ANN**	N_{inc}	t **(s)**	r **(mm)**	ε^p	T **(°C)**	σ **(MPa)**
M_1	3-7-4-1	1,367,147	886	5.870	0.891	794.74	161.95
M_2	3-9-4-1	1,405,471	941	5.895	0.927	798.29	178.78
M_3	3-9-7-1	1,408,680	1023	5.917	0.965	798.76	164.25
M_4	3-15-7-1	1,418,586	1263	5.917	0.950	796.56	165.80

It appears from the study of this table that the modification of the number of neurons in the hidden layers has an influence on the computation time t, which increases with the complexity of the network structure as expected and varies within the range from 886 s to 1263 s approximately since this information is hard to capture from a commercial software that does not contain accurate CPU time reports as the Abaqus software. It is obvious from this table that all models do not give exactly the same results. The computing time increases from M_1 to M_4, proof that the increase in complexity of the ANN has an influence on the global computation time t.

The results, both from the point of view of the dimensional characteristics of the sample (maximum radius r), or the internal fields, such as the temperatures T and the plastic strains ε^p and equivalent stresses σ in the center of the sample, are more or less the same for all M_{1234} flow models.

To have a better analysis of the difference between models M_{1234}, Figure 11 shows the evolution of the equivalent von Mises stress σ for the element in the center of the cylinder during the compression.

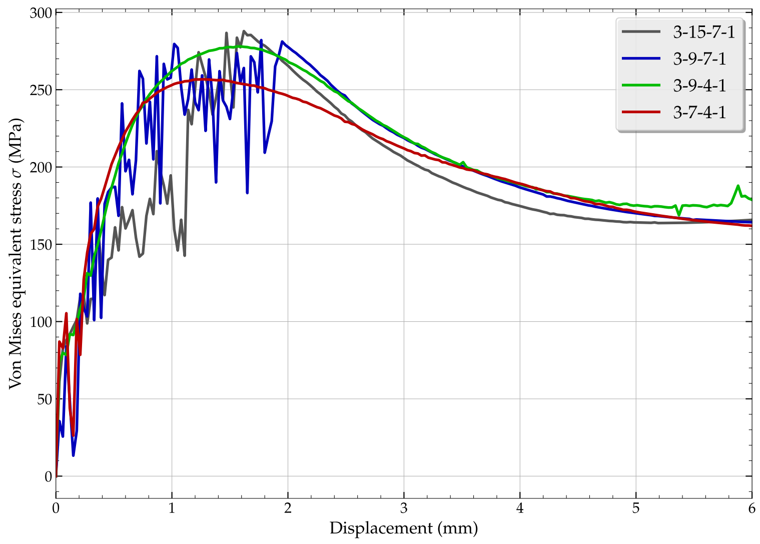


Figure 11. Evolution of the equivalent von Mises stress σ of the specimen during the compression process using the four ANN flow laws.

As shown in this figure, the M_{1234} models give equivalent results with more or less the same value of the equivalent stress σ at the end of the cylinder compression as shown in Table 2. The appearance of the curve for the M_3 and M_4 models is very oscillating in the first 1/3 of the graph. This is probably related to the fact that the M_{34} models are over-fitted and cannot serve as a universal approximator for the flow stress.

To verify this assumption, Figure 12 shows a plot of the predicted flow stress σ^y using the 3-15-7-1 ANN model as a function of the plastic strain ε^p and the plastic strain rate $\dot{\varepsilon}$ for a fixed temperature $T = 750$ °C.

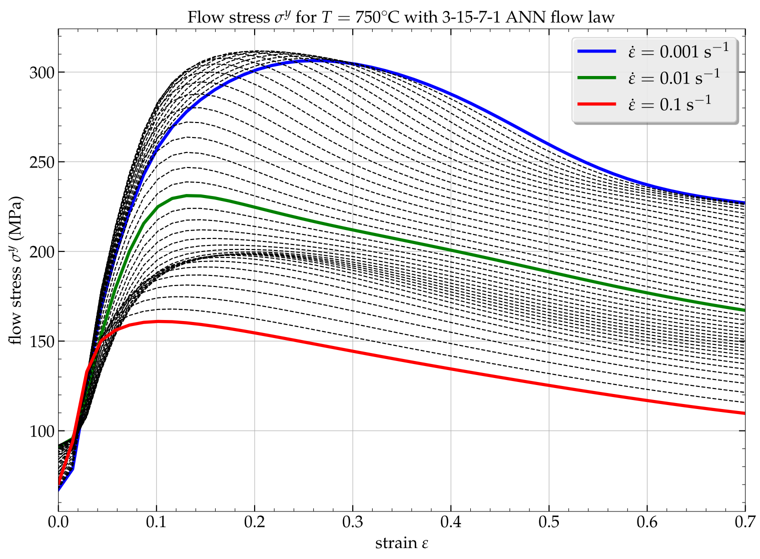


Figure 12. Predicted flow stress σ^y as a function of $(\varepsilon^p, \dot{\varepsilon})$ for a fixed temperature $T = 750$ °C using the 3-15-7-1 model.

This figure shows that when $\dot{\varepsilon}$ varies between 0.001 s^{-1} and 0.01 s^{-1}, and for a value of $\varepsilon^p < 0.3$, the representative curves of σ^y 'cross' demonstrate the poor interpolation of the flow stresses σ^y when the strain rate varies (there is a zone in black dashed lines above the blue line). Thus, for a strain rate between 0.001 s^{-1} and 0.01 s^{-1}, and for a plastic strain less than 0.3, the flow stress σ^y is greater than the value of σ^y calculated for the same plastic strain and $\dot{\varepsilon} = 0.001$ s^{-1}, which is physically not admissible for the behavior of GCr15 material. This causes the oscillations visible in Figure 11. For comparison, Figure 13 shows a similar study to Figure 12 for the 3-7-4-1 model.

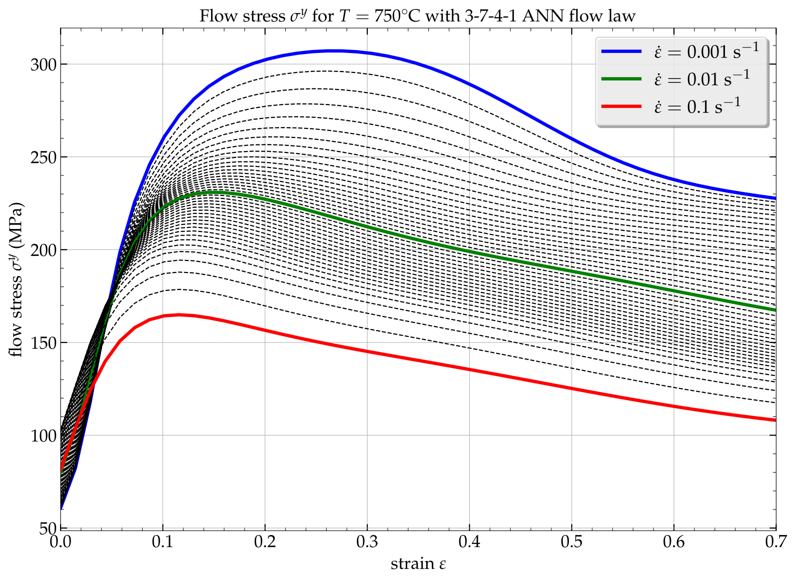


Figure 13. Predicted flow stress σ^y as a function of $(\varepsilon^p, \dot{\varepsilon})$ for a fixed temperature $T = 750$ °C using the 3-7-4-1 model.

This time, there is no visible area where the representative curves 'cross', except perhaps for low values of the strain, where it is hard to distinguish the curves.

This problem occurs for the 3-15-7-1 ANN because we do not have enough training points regarding the strain rate $\dot{\varepsilon}$ (only 3 strain rates are used) regarding the number of internal variable of the ANN, and this leads to erroneous calculations of the flow stress σ^y when the strain rate values differ from those used during training (0.001 s^{-1}, 0.01 s^{-1} and 0.1 s^{-1}). Even though the M_4 model has the lowest values of E_{MAR} and E_{RMS} during the training phase, it is not usable for numerical simulation of the GCr15 flow law in FE simulations. The same type of conclusions can be drawn for the M_3 model.

We can conclude here that only the two first models M_{12} can be used for numerical simulation of the compression of a cylinder, but we must avoid the models M_{34}, as it seems to be over-fitted, and the behavior of the GCr15 alloy is badly represented, even if the curves reported in Figure 5 show that the prediction of model M_4 is excellent.

4. Conclusions and Future Work

In this paper, a flow law based on an artificial neural network capable of predicting the flow stress of a material as a function of input data, such as the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T, for a metallic material subjected to high thermomechanical loading is presented.

From the equations that govern the writing of the neural network, the expressions of the derivatives of the flow stress of the model as a function of the plastic strain ε^p, the strain rate $\dot{\varepsilon}$ and the temperature T were established. These expressions allow the transfer of the neural network behavior into a VUHARD subroutine written in FORTRAN 77 language to allow the use of this network for the computation of the material flow stresses within the Abaqus FE code.

In a first step, data from the works of Ji et al. [30] and Zhou et al. [14], allowed to train several architectures of the proposed model and to compare the results of these ANN models regarding the fidelity of reproduction of the experimental behavior. The comparison of the results obtained allowed to validate the approach and to show the superiority of the ANN model compared to the analytical models based on Johnson–Cook or Arrhenius flow laws, both in terms of the fidelity of the model and quality of the results. In a second step, after transferring the training data to the VUHARD subroutines for the Abaqus FE code, we showed the consistency and quality of the numerical results obtained during the numerical simulation of the compression of a GCr15 alloy cylinder. In the same section, we also discussed the problems of over-fitting the ANN when the number of neurons is too large compared to the training data ranges. It is therefore important to always adjust the structure and size of the neural network to the experimental data that we wish to approximate to avoid this over-fitting phenomenon.

This work thus allowed to highlight the significant contributions of flow laws based on neural networks in numerical simulation by finite elements on a commercial FE code, such as the Abaqus software. The quality of the results obtained allows us to go further in the use of the simulation results and in particular to consider that the results of these finite element simulations can predict the phase transformations and the dynamic recrystallization within the material during the thermomechanical transformation at high temperature.

Funding: This research received no external funding.

Data Availability Statement: Source files of the numerical simulations are available from the author.

Conflicts of Interest: The author declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
DRX	Dynamic Recrystallization
CPU	Central Processing Unit
FE	Finite Element
VUMAT	User subroutine to compute the stress tensor for Abaqus/Explicit
VUHARD	User subroutine to compute the flow stress for Abaqus/Explicit

Appendix A. ANN Flow Law Coefficients

In order to complete this paper, we report here after the computing process and the 65 coefficients of the artificial neural network ANN-3-7-4-1 model used in Section 2.3. The weight matrix for the first hidden layer w_1 is a 7×3 matrix:

$$\mathsf{w}_1 = \begin{bmatrix} -0.4234 & 0.6361 & -3.3756 \\ 0.9035 & -2.0141 & -85.3653 \\ 6.3799 & -1.9357 & -0.3671 \\ -26.7227 & 0.9218 & -2.5756 \\ -0.7105 & 0.7599 & 11.5957 \\ -0.4727 & -18.4137 & 11.6583 \\ 3.4733 & 6.0173 & -2.2098 \end{bmatrix}$$

The biases of the first hidden layer $\overrightarrow{b_1}$ is a 7-component vector:

$$\overrightarrow{b}_1 = \begin{bmatrix} 1.1200 \\ 0.8351 \\ -2.9032 \\ -1.1547 \\ -4.7023 \\ -3.9429 \\ -7.2849 \end{bmatrix}$$

The weight matrix for the second hidden layer w_2 is a 4×7 matrix:

$$\mathsf{w}_2^T = \begin{bmatrix} 1.0488 & -2.2694 & 4.7134 & -8.6149 \\ 1.2225 & 1.7295 & 0.6877 & 8.2364 \\ -0.9681 & -16.4026 & -1.9820 & -0.2577 \\ -2.2942 & -7.3726 & -14.0394 & 16.5637 \\ -11.3020 & -2.9066 & -1.9601 & 0.3557 \\ -35.7421 & -3.0537 & -1.8083 & 0.1603 \\ 0.9444 & 3.5816 & 0.4988 & -0.6497 \end{bmatrix}$$

The biases of the second hidden layer $\overrightarrow{b_2}$ are a 4-component vector:

$$\overrightarrow{b}_2 = \begin{bmatrix} -1.0302 \\ -1.6695 \\ -1.6705 \\ 1.7122 \end{bmatrix}$$

The weight vector for the output layer $\overrightarrow{w}$ is a 4-component vector:

$$\overrightarrow{w} = \begin{bmatrix} 0.6684 \\ 3.0013 \\ 0.1998 \\ -0.1879 \end{bmatrix}$$

The bias of the output layer b is a scalar:

$$b = 0.1631$$

The boundaries of the range of the corresponding field are as follows:

- $\varepsilon^p \in [0.0, 0.7]$
- $\dot{\varepsilon} \in \left[0.001\ \text{s}^{-1}, 0.1\ \text{s}^{-1}\right]$
- $T \in [750\ ^\circ\text{C}, 1300\ ^\circ\text{C}]$
- $\sigma \in [3.052\ \text{MPa}, 306.096\ \text{MPa}]$.

The reference strain rate is $\dot{\varepsilon_0} = 0.001\ \text{s}^{-1}$.

References

1. Pantalé, O.; Caperaa, S.; Rakotomalala, R. Development of an object-oriented finite element program: Application to metal-forming and impact simulations. *J. Comput. Appl. Math.* **2004**, *168*, 341–351. [CrossRef]
2. Dey, S.; Borvik, T.; Hopperstad, O.S.; Langseth, M. On the influence of constitutive relation in projectile impact of steel plates. *Int. J. Impact Eng.* **2007**, *34*, 464–486. [CrossRef]
3. Lin, Y.C.; Chen, M.S.; Zhang, J. Modeling of flow stress of 42CrMo steel under hot compression. *Mater. Sci. Eng. A* **2009**, *499*, 88–92. [CrossRef]
4. Kolsky, H. An Investigation of the Mechanical Properties of Materials at very High Rates of Loading. *Proc. Phys. Society. Sect. B* **1949**, *62*, 676–700. [CrossRef]
5. Lee, W.S.; Liu, C.Y. The effects of temperature and strain rate on the dynamic flow behaviour of different steels. *Mater. Sci. Eng. A* **2006**, *426*, 101–113. [CrossRef]

6. Lennon, A.M.; Ramesh, K.T. The influence of crystal structure on the dynamic behavior of materials at high temperatures. *Int. J. Plast.* **2004**, *20*, 269–290. [CrossRef]
7. Zhang, J.; Chen, B.; Baoxiang, Z. Effect of initial microstructure on the hot compression deformation behavior of a 2219 aluminum alloy. *Mater. Des.* **2012**, *34*, 15–21. [CrossRef]
8. Johnson, G.R.; Cook, W.H. A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates, and High Temperatures. In Proceedings of the 7th International Symposium on Ballistics, The Hague, The Netherlands, 19–21 April 1983; pp. 541–547.
9. Johnson, G.R.; Holmquist, T.J. Evaluation of cylinder-impact test data for constitutive model constants. *J. Appl. Phys.* **1988**, *64*, 3901–3910. [CrossRef]
10. Zerilli, F.J.; Armstrong, R.W. Dislocation-mechanics-based constitutive relations for material dynamics calculations. *J. Appl. Phys.* **1987**, *61*, 1816–1825. [CrossRef]
11. Lin, Y.; Chen, X.M. A critical review of experimental results and constitutive descriptions for metals and alloys in hot working. *Mater. Des.* **2011**, *32*, 1733–1759. [CrossRef]
12. Li, H.Y.; Wang, X.F.; Duan, J.Y.; Liu, J.J. A modified Johnson Cook model for elevated temperature flow behavior of T24 steel. *Mater. Sci. Eng. A* **2013**, *577*, 138–146. [CrossRef]
13. Zhang, D.N.; Shangguan, Q.Q.; Xie, C.J.; Liu, F. A modified Johnson–Cook model of dynamic tensile behaviors for 7075-T6 aluminum alloy. *J. Alloy. Compd.* **2015**, *619*, 186–194. [CrossRef]
14. Zhou, Q.; Ji, C.; Zhu, M.y. Research on Several Constitutive Models to Predict the Flow Behaviour of GCr15 Continuous Casting Bloom with Heavy Reduction. *Mater. Res. Express* **2020**, *6*, 1265f2. [CrossRef]
15. Jia, Z.; Guan, B.; Zang, Y.; Wang, Y.; Lei, M. Modified Johnson-Cook model of aluminum alloy 6016-T6 sheets at low dynamic strain rates. *Mater. Sci. Eng. A* **2021**, *820*, 141565. [CrossRef]
16. Rule, W.K.; Jones, S.E. A revised form for the Johnson-Cook Strengh Model. *Int. J. Impact Eng.* **1998**, *21*, 609–624. [CrossRef]
17. Lin, Y.; Chen, X.M.; Liu, G. A modified Johnson–Cook model for tensile behaviors of typical high-strength alloy steel. *Mater. Sci. Eng. A* **2010**, *527*, 6980–6986. [CrossRef]
18. Lennon, A.M.; Ramesh, K.T. On the performance of modified Zerilli-Armstrong constitutive model in simulating the metal-cutting process. *J. Manuf. Process.* **2017**, *28*, 253–265. [CrossRef]
19. Cheng, C.; Mahnken, R. A modified Zerilli–Armstrong model as the asymmetric visco-plastic part of a multi-mechanism model for cutting simulations. *Arch. Appl. Mech.* **2021**, *91*, 3869–3888. [CrossRef]
20. Hensel, A.; Spittel, T. *Kraft- und Arbeitsbedarf Bildsamer Formgebungsverfahren*; Deutscher Verlag für Grundstoffindustrie: Leipzig, Duchland, 1978.
21. Chadha, K.; Shahriari, D.; Jahazi, M. An Approach to Develop Hansel–Spittel Constitutive Equation during Ingot Breakdown Operation of Low Alloy Steels. In *Frontiers in Materials Processing, Applications, Research and Technology*; Springer: Hyderabad, India, 2018; pp. 239–246. [CrossRef]
22. Jonas, J.; Sellars, C.; Tegart, W.M. Strength and structure under hot-working conditions. *Metall. Rev.* **1969**, *14*, 1–24. [CrossRef]
23. He, A.; Xie, G.; Zhang, H.; Wang, X. A comparative study on Johnson–Cook, modified Johnson–Cook and Arrhenius–type constitutive models to predict the high temperature flow stress in 20CrMo alloy steel. *Mater. Des. (1980–2015)* **2013**, *52*, 677–685. [CrossRef]
24. Liang, P.; Kong, N.; Zhang, J.; Li, H. A Modified Arrhenius-Type Constitutive Model and its Implementation by Means of the Safe Version of Newton–Raphson Method. *Steel Res. Int.* **2022**, *94*, 2200443. [CrossRef]
25. Nemat-Nasser, S.; Guo, W.G. Thermomechanical response of DH-36 structural steel over a wide range of strain rates and temperatures. *Mech. Mater.* **2003**, *35*, 1023–1047. [CrossRef]
26. Khan, A.S.; Sung Suh, Y.; Kazmi, R. Quasi-static and dynamic loading responses and constitutive modeling of titanium alloys. *Int. J. Plast.* **2004**, *20*, 2233–2248. [CrossRef]
27. Gao, C.Y. FE Realization of a Thermo-Visco-Plastic Constitutive Model Using VUMAT in ABAQUS/Explicit Program. In *Computational Mechanics*; Springer: Berlin/Heidelberg, Germany, 2007; p. 301.
28. Ming, L.; Pantalé, O. An Efficient and Robust VUMAT Implementation of Elastoplastic Constitutive Laws in Abaqus/Explicit Finite Element Code. *Mech. Ind.* **2018**, *19*, 308. [CrossRef]
29. Zener, C.; Hollomon, J.H. Effect of Strain Rate Upon Plastic Flow of Steel. *J. Appl. Phys.* **1944**, *15*, 22–32. [CrossRef]
30. Ji, C.; Wang, Z.; Wu, C.; Zhu, M. Constitutive Modeling of the Flow Stress of GCr15 Continuous Casting Bloom in the Heavy Reduction Process. *Metall. Mater. Trans. B* **2018**, *49*, 767–782. [CrossRef]
31. Pantalé, O.; Tize Mha, P.; Tongne, A. Efficient implementation of non-linear flow law using neural network into the Abaqus Explicit FEM code. *Finite Elem. Anal. Des.* **2022**, *198*, 103647. [CrossRef]
32. Minsky, M.L.; Papert, S. *Perceptrons: An Introduction to Computational Geometry*; MIT Press: Cambridge, UK, 1969.
33. Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Netw.* **1989**, *2*, 359–366. [CrossRef]
34. Gorji, M.B.; Mozaffar, M.; Heidenreich, J.N.; Cao, J.; Mohr, D. On the Potential of Recurrent Neural Networks for Modeling Path Dependent Plasticity. *J. Mech. Phys. Solids* **2020**, *143*, 103972. [CrossRef]
35. Jamli, M.; Farid, N. The Sustainability of Neural Network Applications within Finite Element Analysis in Sheet Metal Forming: A Review. *Measurement* **2019**, *138*, 446–460. [CrossRef]

36. Jiao, P.; Alavi, A.H. Artificial Intelligence-Enabled Smart Mechanical Metamaterials: Advent and Future Trends. *Int. Mater. Rev.* **2020**, *66*, 365–393. [CrossRef]
37. Ghaboussi, J.; Garrett, J.H.; Wu, X. Knowledge-Based Modeling of Material Behavior with Neural Networks. *J. Eng. Mech.* **1991**, *117*, 132–153. [CrossRef]
38. Ghaboussi, J.; Pecknold, D.A.; Zhang, M.; Haj-Ali, R.M. Autoprogressive Training of Neural Network Constitutive Models. *Int. J. Numer. Methods Eng.* **1998**, *42*, 105–126. [CrossRef]
39. Ghaboussi, J.; Sidarta, D. New Nested Adaptive Neural Networks (NANN) for Constitutive Modeling. *Comput. Geotech.* **1998**, *22*, 29–52. [CrossRef]
40. Lin, Y.; Zhang, J.; Zhong, J. Application of Neural Networks to Predict the Elevated Temperature Flow Behavior of a Low Alloy Steel. *Comput. Mater. Sci.* **2008**, *43*, 752–758. [CrossRef]
41. Ashtiani, H.R.; Shahsavari, P. A Comparative Study on the Phenomenological and Artificial Neural Network Models to Predict Hot Deformation Behavior of AlCuMgPb Alloy. *J. Alloy. Compd.* **2016**, *687*, 263–273. [CrossRef]
42. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. TensorFlow: A System for Large-Scale Machine Learning. In Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, Savannah, GA, USA, 2–4 November 2016; USENIX Association: Berkeley, CA, USA, 2016; pp. 265–283.
43. Mattmann, C. *Machine Learning with Tensorflow*; O'REILLY MEDIA: Shelter Island, NY, USA, 2020.
44. Lu, Z.; Pan, Q.; Liu, X.; Qin, Y.; He, Y.; Cao, S. Artificial Neural Network Prediction to the Hot Compressive Deformation Behavior of Al–Cu–Mg–Ag Heat-Resistant Aluminum Alloy. *Mech. Res. Commun.* **2011**, *38*, 192–197. [CrossRef]
45. Ponthot, J.P. Unified Stress Update Algorithms for the Numerical Simulation of Large Deformation Elasto-Plastic and Elasto-Viscoplastic Processes. *Int. J. Plast.* **2002**, *18*, 91–126. [CrossRef]
46. Simo, J.C.; Hughes, T.J.R. *Computational Inelasticity*; Interdisciplinary Applied Mathematics, Springer: New York, NY, USA, 1998.
47. Kingma, D.P.; Lei, J. Adam: A method for stochastic optimization. *arXiv Preprint* **2014**, arXiv:1412.6980
48. Phaniraj, M.P.; Lahiri, A.K. The applicability of neural network model to predict flow stress for carbon steels. *J. Mater. Process. Technol.* **2003**, *141*, 219–227. [CrossRef]
49. Jansen van Rensburg, G.; Kok, S. Tutorial on State Variable Based Plasticity: An Abaqus UHARD Subroutine. In Proceedings of the Eighth South African Conference on Computational and Applied Mechanics—SACAM2012, Johannesburg, South Africa, 3–5 September 2012.