



# Machine learning tools in production engineering

Michael Rom<sup>1</sup> · Matthias Brockmann<sup>2</sup> · Michael Herty<sup>1</sup> · Elisa Iacomini<sup>1</sup>

Received: 3 March 2022 / Accepted: 21 June 2022 / Published online: 11 July 2022  
© The Author(s) 2022

## Abstract

Machine learning methods have shown potential for the optimization of production processes. Due to the complex relationships often inherent in those processes, the success of such methods is often uncertain and unreliable. Therefore, understanding the (algorithmic) behavior and results of machine learning methods is crucial to improve the prediction of production processes. Here, mathematical tools may help. This paper shows how efficient algorithms for the training of neural networks and their retraining in the framework of transfer learning are expressed in a discrete as well as a time-continuous formulation. The latter can be analyzed and investigated using mathematical techniques from kinetic gas dynamics. The results obtained provide a first step towards explainable artificial intelligence. Based on the mathematical description, an adapted ensemble method for retraining of neural networks is proposed and compared with backpropagation algorithms. The process of training and retraining is a common task and therefore demonstrated for two very different production processes. The first one involves the prediction of specific cutting forces and the second one the prediction of particle properties in a plasma spraying coating process. For both use cases, the presented algorithms are applied and their performance is evaluated giving thereby an indication how mathematically inspired methods improve classical tasks in production processes.

**Keywords** Machine learning · Neural networks · Kinetic theory · Process validation · Advanced manufacturing

## 1 Introduction

In order to operate production systems at an optimum point, many different process influences need to be taken into account. These processes usually interact in such a complex manner that it is not possible to describe the behavior of production systems in terms of a mathematical model. In practice, the systems are operated at a set point which was found to be stable by means of experimental validation.

Modern production systems are equipped with many built-in sensors and data can be acquired during the manufacturing processes. However, these sensors usually cannot measure primary process variables but mainly state variables of the machinery, e.g., voltages, switching impulses or vibrations. In order to make use of these data, models of the governing physical processes are necessary. The data are used for validation of these models and need to be adapted to each reality of interest. In production systems, these realities of interest are mostly productivity, quality or cost optimization. Those optimization targets are in conflict with each other and companies usually choose their strategy according to product and the market.

Mathematical algorithms, which can handle this vast amount of data from the machines, yield a big potential for improving the optimization of production processes especially in the field of process validation. Process validation describes the process of finding an optimum for a certain machining process. Algorithms can be used for data regression, interpolation and similarity considerations. An algorithm needs always to be connected to a method which reflects the physical relationships of the distinct manufacturing processes. Methods for use in real production

---

✉ Michael Rom  
rom@igpm.rwth-aachen.de

Matthias Brockmann  
matthias.brockmann@fh-muenster.de

Michael Herty  
herty@igpm.rwth-aachen.de

Elisa Iacomini  
iacomini@igpm.rwth-aachen.de

<sup>1</sup> Institute of Geometry and Applied Mathematics (IGPM),  
RWTH Aachen University, Aachen, Germany

<sup>2</sup> Department of Mechanical Engineering, FH Münster,  
Münster, Germany

environments need to be furthermore robust in terms of data gaps, noise and spurious correlations. Advanced mathematical algorithms yield potentials for the development of such methods.

There exists a vast literature on mathematical models and numerical algorithms for machine learning, see, for instance, [1, 2] or literature reviews with respect to machine learning in production [3–5]. However, an extensive literature review is beyond the scope of this work.

The main objectives of this work are to demonstrate how neural network structures can be mathematically analyzed to gain insight in the sense of explainable artificial intelligence and how mathematical algorithms can improve common and recurrent tasks such as the training or retraining of neural networks. Both topics are of high relevance in the framework of production processes: first, understanding the black-box behavior of neural networks is desirable to beneficially complement or even replace expensive experiments or simulations. Second, changing process or material parameters require the repeated application of a machine, which in terms of a machine learning method corresponds to a transfer learning task, see, for instance, [6, 7]. Furthermore, the present paper aims to discuss the potential advanced mathematical algorithms in general and the methods presented in this work in particular have for production engineering and to highlight the benefits of such an interdisciplinary approach.

Our goal regarding transfer learning is the efficient retraining of an already trained neural network to adapt to a new data set. We compare the training and retraining of neural networks by backpropagation with a gradient-free algorithm using ensemble Kalman filtering. The latter is also discussed in, e.g., [8–13]. In these works, the ensemble Kalman filtering algorithm is used for training neural networks, i.e., with a random initialization of weights and biases. While this is efficient since the method is derivative-free and parallelizable, convergence may suffer from a problem called particle collapse and from only providing an approximation of gradient descent [9]. However, we use the ensemble Kalman filtering algorithm for the retraining of neural networks. Hence, the weights and biases of the network to be retrained are initialized by weights and biases previously determined for a similar data set. Consequently, a poor convergence behavior is usually not an issue in our case. Furthermore, this even allows us to simplify the formulation of the algorithm. To the authors' knowledge, the task of retraining of neural networks with the ensemble Kalman filtering algorithm or a variant of it has not been studied before.

The basis for the investigations and algorithms in this work is a special type of residual neural networks (ResNets) [14]. These are formulated in an amenable form such that mathematical tools and techniques are applicable.

There have been made several formulations [15–17]. For example, in [17], the connection of deep convolutional neural networks to partial differential equations (PDEs) is derived. In [15], the time-continuous version of a ResNet is studied and different temporal discretization schemes are discussed. There are also studies on application of kinetic methods to ResNets [18–21]. For example, in [20], the authors consider the limit of infinitely many neurons and gradient steps in the case of one hidden layer. They are able to prove a central limit theorem. In this work, we show results of ResNets in the continuous as well as in the discrete case for the first time.

By means of two different use cases, we investigate the performance of the presented machine learning algorithms. In particular, we demonstrate training and efficient retraining of ResNets. First, we study the prediction of specific cutting forces for different materials and compare the results with measurement data from, e.g., [22]. Second, the particle behavior in a plasma spraying coating process [23–25] is predicted based on large simulation data sets. These use cases stem from cooperations within the DFG project EXC-2023 “Internet of Production”. They are well suited for the demonstration of transfer learning algorithms in production engineering because they are typical recurrent processes underlying constant process or material parameter adjustments.

The present paper is structured as follows: In Sect. 2, we present advanced mathematical algorithms based on ResNets, in particular the ensemble Kalman filtering algorithm for the retraining of ResNets. Section 3 studies the performance of this retraining algorithm by investigating two use cases. The potential and increasing importance of advanced mathematical algorithms in production engineering is discussed in Sect. 4. Finally, Sect. 5 concludes the paper and gives an outlook on future work.

## 2 Advanced mathematical algorithms

In this section, we present discrete and time-continuous formulations of residual neural networks (ResNets) and show how a step towards explainable artificial intelligence can be made. Furthermore, we discuss the training and retraining of ResNets and describe an algorithm based on backpropagation and a gradient-free algorithm using ensemble Kalman filtering.

### 2.1 Mathematical formulation of residual neural networks (ResNets)

Artificial neural networks (ANNs) are multilayer perceptron networks. They can be understood as general mapping of (finitely many) input to output values. The universal

approximation theorem states that given enough layers  $L$  of the network, there exists an ANN for any continuous mapping [26]. We consider here a particular subclass of residual neural networks (ResNets). For notational convenience, denote by  $\mathbf{x}^{(0)} \in \mathbb{R}^m$  the vector of input data and by  $\mathbf{x}^{(L)} \in \mathbb{R}^m$  the output after application of  $L$  layers of the ANN. Also,  $\mathbf{W}^{(\ell)} \in \mathbb{R}^{m \times m}$  and  $\mathbf{b}^{(\ell)} \in \mathbb{R}^m$ ,  $\ell = 1, \dots, L$ , are weights matrices and bias vectors, respectively, and  $\sigma$  is a nonlinear activation function. For simplicity, we have assumed that the weights and biases are of the same dimension as  $\mathbf{x}$ . The prediction of a ResNet is given by

$$\mathbf{x}^{(\ell)} = \mathbf{x}^{(\ell-1)} + \sigma\left(\mathbf{W}^{(\ell)T} \mathbf{x}^{(\ell-1)} + \mathbf{b}^{(\ell)}\right), \ell = 1, \dots, L. \tag{1}$$

In order to derive an amenable formulation for mathematical tools, we introduce a pseudo-time. Denote by  $\Delta t > 0$  a time step and define for  $\ell = 0, \dots, L$

$$\mathbf{x}(\ell \Delta t) = \mathbf{x}^{(\ell)}, \mathbf{b}(\ell \Delta t) = \mathbf{b}^{(\ell)} \text{ and } \mathbf{W}(\ell \Delta t) = \mathbf{W}^{(\ell)}.$$

We also scale  $\sigma$  as  $\Delta t \sigma$  and the number of layers  $L$  as  $L/\Delta t$ . In the time-continuous limit, which corresponds to  $\Delta t \rightarrow 0$ ,  $L \rightarrow \infty$  such that  $L\Delta t$  is constant, we obtain

$$\begin{cases} \dot{\mathbf{x}}(t) = \sigma(\mathbf{W}(t) \mathbf{x}(t) + \mathbf{b}(t)), \\ \mathbf{x}(0) = \mathbf{x}^{(0)}. \end{cases} \tag{2}$$

A particular feature of ANNs is the training. Here, usually a larger number  $i = 1, \dots, N$  of pairs of input and target data  $(\mathbf{x}_i^{(0)}, \mathbf{y}_i)$  is presented to the ANN and optimal weights  $\mathbf{W}^{(\ell)}$  and  $\mathbf{b}^{(\ell)}$  for  $\ell = 1, \dots, L$  are obtained by minimizing a suitable loss function  $e(\mathbf{x}, \mathbf{y})$ . An example of a measure of the deviation is the half mean square error

$$e(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \|\mathbf{y} - \mathbf{x}\|_2^2. \tag{3}$$

In terms of the previously defined equations, the training procedure may be formulated as

$$\min_{W(\cdot), b(\cdot)} \frac{1}{N} \sum_{i=1}^N e(\mathbf{x}_i(T), \mathbf{y}_i), \tag{4}$$

where  $\mathbf{x}_i(T)$  denotes the solution to (2) for initial data  $\mathbf{x}_i(0) = \mathbf{x}_i^{(0)}$ . A similar formulation for the discrete problem (1) is possible. Also, we refer, for instance, to [8–11] for suitable numerical methods for problem (4).

The formulation (2) can be extended easily to include also stochastic fluctuations. This modeling assumption is inspired by stochastic neural networks with stochastic output layers [27–29]. It has been observed that this approach is useful for the optimization procedure since the random fluctuations allow to escape from local minima. For a given family of *diagonal* positive semi-definite diffusion

matrix  $K \in \mathbb{R}^{m \times m}$ , a factor  $\nu > 0$ , and a  $m$ -dimensional standard Wiener process  $dW$ , the corresponding dynamics reads

$$\begin{cases} d\mathbf{x}(t) = \sigma(\mathbf{W}(t) \mathbf{x}(t) + \mathbf{b}(t))dt + \nu K dW, \\ \mathbf{x}(0) = \mathbf{x}^{(0)}. \end{cases} \tag{5}$$

Mathematically, the feature of presenting a possibly large number  $N$  of input and target data to the ANN can be exploited in terms of the mean-field limit [21, 30–33]. Instead of considering input data  $\mathbf{x}_i^{(0)}$  individually, we consider a statistical description of these data. Analytically, this is justified if the input data are iid random variables. Furthermore, we assume that the initial distribution of the input data  $\mathbf{x}_i^{(0)}$ ,  $i = 1, \dots, N$ , is given by a probability distribution  $g_0(\mathbf{x}) : \mathbb{R}^m \rightarrow \mathbb{R}$ . Then, the probability density  $g(t, \mathbf{x})$  is obtained as (weak) solution to the Vlasov-Fokker-Planck-type partial differential equation (PDE)

$$\begin{cases} \partial_t g(t, \mathbf{x}) + \nabla_{\mathbf{x}} \cdot (\sigma(\mathbf{W}(t)\mathbf{x} + \mathbf{b}(t))g(t, \mathbf{x})) \\ \quad + \frac{\nu^2}{2} \nabla_{\mathbf{x}} \cdot \nabla_{\mathbf{x}} (K^2(\mathbf{x})g(t, \mathbf{x})) = 0, \\ g(0, \mathbf{x}) = g_0(\mathbf{x}), \end{cases} \tag{6}$$

see [34]. The equation is the mean-field limit of (5) for  $N \rightarrow \infty$  initial data. The corresponding loss function  $\frac{1}{N} \sum_{i=1}^N e(\mathbf{x}_i(T), \mathbf{y}_i)$  is obtained on the mean-field level as

$$\int_{\mathbb{R}^m} \int_{\mathbb{R}^m} e(\mathbf{x}, \mathbf{y}) g(T, \mathbf{x}) h_T(\mathbf{y}) d\mathbf{x} d\mathbf{y}, \tag{7}$$

where  $h_T(\cdot)$  is the probability distribution of the target data. For an example, consider the case of finitely many output data. Then, we have  $h_T(\mathbf{y}) = \mu_Y^N(\mathbf{y})$  where  $\mu_Y^N$  is the empirical measure on the output data:

$$\mu_Y^N(\mathbf{y}) = \frac{1}{N} \sum_{i=1}^N \prod_{k=1}^m \delta(y_k - y_{i,k}). \tag{8}$$

Rigorous convergence results are available [34] but beyond the scope of the manuscript. Also note that the given formulation is by far not the only possibility to define suitable limits of neural networks and we refer to [15–20] for further references where, e.g., limits in time and number of neurons are considered.

The previous reformulation shows that there is an underlying PDE type structure to be exploited in the following sections. The derivation is obtained for infinitely many layers  $L$  as well as infinitely many input data. Clearly, this situation is an abstraction of realistic applications of ResNets and solely done for the purpose of further analysis. For practical applications, the corresponding results have again to be discretized and implemented. Practical examples are presented in Sect. 3.

### 2.2 Towards explainable artificial intelligence

The classification of achievable output states is a typical question posed on ANNs. In the context of the formulation (6), a partial answer can be obtained by considering solutions  $g_\infty(\mathbf{x})$  that are the steady states. In the case  $m \equiv 1$  and  $K \in \mathbb{R}, K > 0$  for all  $x \in \mathbb{R}$ , this is particularly easy and we obtain

$$g_\infty(x; w_\infty, b_\infty) = \frac{C}{K^2(x)} \exp\left(\int_{\mathbb{R}} \frac{2\sigma(w_\infty x + b_\infty)}{v^2 K^2(x)} dx\right), \tag{9}$$

where  $C$  is chosen such that  $\int_{\mathbb{R}} g_\infty(x) dx = 1$ , see [34], and  $w_\infty$  and  $b_\infty$  are two constants that parametrize the possible steady states. The case  $m = 1$  corresponds to input data having only a single feature. The characterization through the steady states (9) now classifies the admissible output data distributions. Those states still depend on the choice of the activation function  $\sigma$  as well as the choice of the variance  $v^2$  and the diffusion  $K(x)$ . In the case  $\sigma(x) = x$  and  $K(x) = 1$ , we obtain, e.g., that  $g_\infty$  is in fact a Gaussian distribution showing that the ANN can represent any output data that is sampled from a Gaussian distribution with given mean and variance:  $(\bar{\mu}, \bar{v}^2)$ . The latter determine implicitly the choice of  $(w_\infty, b_\infty)$ . Other distributions can also be obtained and we refer to [34] for further examples.

### 2.3 Fast (re-)training of neural networks

Given a trained neural network, a typical problem are changing external conditions. Examples are given for the prediction of specific cutting forces for different materials and a plasma spraying coating process in Sect. 3. Here, it would be advantageous to use a trained neural network for a single specific geometry as starting point for prediction under changing geometries or materials without the need to retrain the complete network. Having a continuous formulation at hand, this problem mathematically is solved by differentiating the cost functional (7) with respect to target data  $h_T(\cdot)$ . This in turn requires to differentiate the continuous dynamics (6) to obtain the sensitivities  $\frac{\partial g}{\partial h_T}$ . Using Lagrange multiplier theory, an explicit expression of those sensitivities is obtained using adjoint variables. Those allow in turn to obtain suitable updates for the weights and biases of the neural network.

For a step size  $\eta > 0$ , the updates  $\Delta \mathbf{W}$  of weights and  $\Delta \mathbf{b}$  of biases are given by

$$\Delta \mathbf{W}(t) := -\eta \int_{\mathbb{R}} g(t, \mathbf{x}) \mathbf{x} \dot{\sigma}(\mathbf{W}(t)\mathbf{x} + \mathbf{b}(t)) \partial_x \lambda(t, \mathbf{x}) d\mathbf{x}, \tag{10}$$

$$\Delta \mathbf{b}(t) := -\eta \int_{\mathbb{R}} g(t, \mathbf{x}) \dot{\sigma}(\mathbf{W}(t)\mathbf{x} + \mathbf{b}(t)) \partial_x \lambda(t, \mathbf{x}) d\mathbf{x}. \tag{11}$$

In case of differentiable activation functions, we have denoted by  $\dot{\sigma}$  the derivative, otherwise suitable smoothing needs to be applied or a subdifferential calculus is required to define the derivative. The previous formula involves the adjoint state  $\lambda$  that formally fulfills

$$\partial_t \lambda(t, \mathbf{x}) + \sigma(\mathbf{W}(t)\mathbf{x} + \mathbf{b}(t)) \partial_x \lambda(t, \mathbf{x}) - \frac{v^2}{2} \nabla_x \cdot \nabla_x K^2(\mathbf{x}) \lambda(t, \mathbf{x}) = 0, \tag{12}$$

$$\lambda(T, \mathbf{x}) = \int_{\mathbb{R}} \partial_x e(\mathbf{x}, y) h_T(y) dy. \tag{13}$$

The continuous description allows also for a numerical implementation by suitable discretization of (6), (10), (11) and (12). This implies that a single step to update the weights is obtained by solving two partial differential equations. Depending on the dimension  $m$  of the feature vector  $\mathbf{x} \in \mathbb{R}^m$ , solving the partial differential equations can be computationally challenging. Furthermore, theoretically, one needs to solve the previous equations iteratively until  $\Delta \mathbf{W}(t) = \mathbf{0}$  and  $\Delta \mathbf{b}(t) = \mathbf{0}$  for all  $t$  which imposes additional computational constraints. A detailed discussion on advantages as well as the performance is given in [34]. Here, we propose two numerical methods for fast retraining.

#### 2.3.1 Discrete conditions through differentiation

The discrete forward propagation formula (1) allows for an analytical differentiation provided that  $\sigma$  is sufficiently smooth and the obtained differential can be used to update weights and biases sequentially. Here,  $w_{ij}^{(\ell)}$  is the entry  $(i, j)$  of the matrix  $\mathbf{W}^{(\ell)} = \mathbf{W}(\ell \Delta t)$  and similarly  $b_i^{(\ell)}$  is the  $i$ th entry of the vector  $\mathbf{b}^{(\ell)} = \mathbf{b}(\ell \Delta t)$ . The updates are given by

$$\Delta w_{ij}^{(\ell)} = -\eta x_i^{(\ell-1)} \delta_j^{(\ell)} \quad \text{and} \quad \Delta b_i^{(\ell)} = -\eta \delta_i^{(\ell)}, \tag{14}$$

where  $\eta > 0$  is a given learning rate,

$$\delta_j^{(\ell)} = \left( \sum_{n=\ell+1}^L \sum_{k=1}^m \delta_k^{(n)} w_{jk}^{(n)} \right) \dot{\sigma} \left( \sum_{i=1}^m \left( w_{ij}^{(\ell)} x_i^{(\ell-1)} \right) + b_j^{(\ell)} \right) \tag{15}$$

for  $\ell = 1, \dots, L - 1$  and

$$\delta_j^{(L)} = \left( x_j^{(L)} - y_j \right) \dot{\sigma} \left( \sum_{i=1}^m \left( w_{ij}^{(L)} x_i^{(L-1)} \right) + b_j^{(L)} \right). \tag{16}$$

The Eqs. (14)-(16) are derived using direct differentiation of the error according to (3) and they are presented for the case  $K \equiv 0$  for simplicity. In case of differentiable activation

functions, we have denoted by  $\dot{\sigma}$  the derivative, otherwise the subdifferential in the sense of Clarke [35]. An iterative procedure can be formulated for the previous updates for each set of new input and target data  $(\mathbf{x}_i^{(0)}, \mathbf{y}_i), i = 1, \dots, N$ . For each value of this set, the Eqs. (1) and (14)–(16) are computed iteratively to update the weights and biases until a mean error such as the root mean square (RMS) error

$$e_{\text{RMS},j} = \sqrt{\frac{1}{2N} \left( \sum_{i=1}^N (y_{ij} - x_{ij}^{(L)})^2 \right)}, \quad j \in \{1, \dots, m\} \quad (17)$$

is sufficiently small for each component, i.e., output quantity,  $j$ .

Note that the previous Eqs. (15) and (16) have to be solved backwards in  $\ell$ , starting at  $\ell = L$ , as in the continuous case where the backward equation for  $\lambda$  in Eq. (12) is employed. Hence, again, an iterative procedure is required and depending on  $L$  and  $m$  the computational costs might be significant. Further note that as in the continuous case we require at least subdifferentiability of the activation function.

For the training and retraining examples in Sect. 3, we use the fully discrete approach.

### 2.3.2 Gradient-free approach for high-dimensional problems

An alternative numerical method for high-dimensional optimization has been proposed recently for the fast solution to inverse problems [9, 11–13] using Kalman filtering [36]. The proposed and analyzed method uses the ensemble Kalman filter and has the advantages of being a purely primal method and of *not* relying on derivatives. In particular in the context of retraining, this is appealing since the activation function is in many cases non-differentiable. Furthermore, a primal method does not need to rely on solving backward equations for updating  $\lambda$  or  $\delta_j^{(\ell)}$  as shown in the previous section. We therefore briefly present the application of ensemble Kalman filtering for (re)training weights and biases of a neural network.

First, we define an initial ensemble of  $N_j$  particles, i.e., state vectors, by sampling from a normal distribution  $\mathcal{N}(\mathbf{u}_{\text{init}}, \sigma_c I_d)$ , where  $\mathbf{u}_{\text{init}}$  and  $\sigma_c I_d$  are the mean and the covariance matrix of the distribution, respectively. Here,  $\sigma_c$  is a parameter and  $I_d$  denotes the identity matrix with  $d$  the dimension of the vector  $\mathbf{u}_{\text{init}}$ . The state vector  $\mathbf{u}_{\text{init}}$  contains the initial weights  $w_{ij}^{(\ell)}$  and biases  $b_i^{(\ell)}$  of a neural network with layers  $\ell = 1, \dots, L$  and  $i, j = 1, \dots, m$  nodes. Note the dimension of each ensemble member

$$\mathbf{u}_j \in \mathbb{R}^{L(m^2+m)}, \quad j \in \{1, \dots, N_j\}$$

in the presented case. Hence, the total dimension of unknowns to be updated by the filter is  $N_j L(m^2 + m)$  which can be significantly larger than any number of unknowns within a suitable discretization of the continuous model (12).

For the initialization, random values or, if available, weights and biases of an already trained neural network can be used. The latter allows for an efficient retraining of a neural network to adapt the existing weights and biases to new input and/or target values. Then, retraining by an ensemble Kalman filter is an iterative process.

An ensemble of  $N_j$  particles is denoted by  $\mathbf{u} := \{\mathbf{u}_j\}_{j=1}^{N_j}$ . Its empirical covariance matrix is given by

$$C(\mathbf{u}) = \frac{1}{N_j} \sum_{j=1}^{N_j} (\mathbf{u}_j - \bar{\mathbf{u}}) \otimes (\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j) - \bar{\mathcal{G}}), \quad (18)$$

where  $\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j)$  denotes a function which, depending on an input  $\mathbf{x}^{(0)}$  and a particle  $\mathbf{u}_j$ , computes an output. In the following,  $\mathcal{G}$  is the forward propagation of a residual neural network given by (1) on the discrete level or by a discretization of Eq. (6) for the continuous case. Hence, for an input  $\mathbf{x}^{(0)}$ , respectively  $g_0(\mathbf{x})$ , chosen from a training set and weights and biases stored in the particle  $\mathbf{u}_j$ , the output of  $\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j)$  is  $\mathbf{x}^{(L)}$ , respectively  $\mathbf{x}^{(T)}$ . The mean values  $\bar{\mathcal{G}}$  and  $\bar{\mathbf{u}}$  are computed by

$$\bar{\mathcal{G}} = \frac{1}{N_j} \sum_{j=1}^{N_j} \mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j), \quad \bar{\mathbf{u}} = \frac{1}{N_j} \sum_{j=1}^{N_j} \mathbf{u}_j. \quad (19)$$

The update of a particle  $\mathbf{u}_j$  is determined by adding

$$\Delta \mathbf{u}_j = C(\mathbf{u}) \Gamma^* (\mathbf{y} - \mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j)), \quad (20)$$

where  $\mathbf{y}$  is the target corresponding to the input  $\mathbf{x}^{(0)}$  and  $\Gamma^* \in \mathbb{R}^{m \times m}$  denotes a covariance matrix. Note that in other works, e.g., [9, 11],  $\Gamma^* = (D(\mathbf{u}) + \Gamma^{-1})^{-1}$ , where  $D(\mathbf{u})$  is the covariance matrix

$$D(\mathbf{u}) = \frac{1}{N_j} \sum_{j=1}^{N_j} (\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j) - \bar{\mathcal{G}}) \otimes (\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j) - \bar{\mathcal{G}}) \quad (21)$$

and  $\Gamma$  is the covariance matrix of the measurement noise. In case of retraining of neural networks,  $\Gamma$  is unknown and therefore, as a simplification, we use  $\Gamma^*$  as an algorithmic parameter. With the updated particles, the next iteration of the (re)training algorithm starts with the forward propagation  $\mathcal{G}(\mathbf{x}^{(0)}, \mathbf{u}_j)$  for a new input  $\mathbf{x}^{(0)}$  randomly chosen from the training data set to compute (18). For a fast convergence of this algorithm, increasing the norm of the covariance matrix  $\Gamma^*$  with each iteration has turned out to be a successful choice provided that the norm stays bounded. This avoids the effect of the collapse of particles, see [9], which refers to the convergence of all particles to the same set of

weights and biases while the training error is still large. The algorithm is run until a mean error such as the root mean square (RMS) error (17) is below a given tolerance.

### 3 Use cases

In this section, we apply ResNets to two different use cases: the prediction of (i) specific cutting forces and (ii) a plasma spraying coating process. In addition, we compare the retraining algorithms presented above.

#### 3.1 Prediction of specific cutting forces

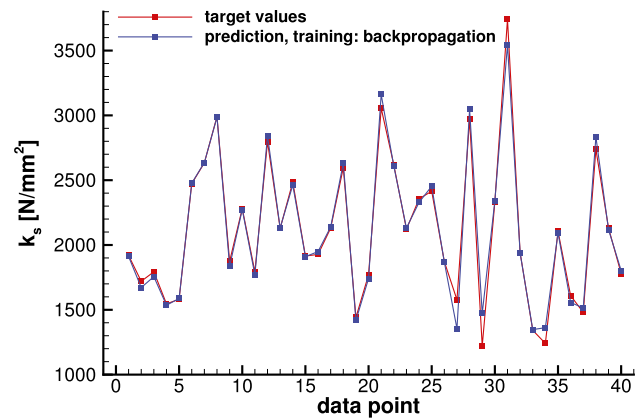
For obtaining a high workpiece quality from a cutting process, the accurate prediction of cutting forces is crucial. A wide range of parameters such as the cutting material properties, the cutting wedge geometry, the cutting speed or the chip thickness influences the cutting forces. At the Laboratory for Machine Tools and Production Engineering (WZL) at RWTH Aachen University, tables containing the specific cutting forces for a large number of materials are available. These tables were gathered in extensive investigations, for instance, by Kienzle and Victor [22]. With only a few of the mentioned parameters, a ResNet can be trained being capable of accurately predicting the specific cutting forces for a selected material. Using the ensemble Kalman filter (EnKF) retraining method introduced in Sect. 2.3.2, this ResNet can be efficiently adapted for the prediction of specific cutting forces for further materials. This will be demonstrated exemplarily in the following for two materials arbitrarily chosen from the WZL tables. For further results, we refer to [37].

##### 3.1.1 Training a ResNet for material C35G

For the training of a ResNet, we choose three input quantities, namely the chip thickness  $h$ , the rake angle  $\gamma$  and the cutting speed  $v$ . The target quantity to be predicted by the ResNet is the specific cutting force  $k_s$ . For the material C35G, 91 data points are available. For the training of the ResNet, 51 of these data points are randomly chosen such that 40 data points remain for testing the resulting network.

**Table 1** Exemplary input and target data for material C35G

Data pt.	Chip thickness $h$ [mm]	Rake angle $\gamma$ [°]	Cutting speed $v$ [m/min]	Spec. cutting force $k_s$ [N/mm <sup>2</sup> ]
1	1	15	25	1919
2	1	-6	200	1720
3	0.4	6	200	1793
4	0.8	6	100	1546
5	0.63	6	200	1581
⋮	⋮	⋮	⋮	⋮



**Fig. 1** Prediction of specific cutting forces  $k_s$  of material C35G for 40 test data points after training by backpropagation

Table 1 lists the first five of the test data points, correspondingly to Fig. 1 described below, to give an idea of the structure of the data. The training is terminated when the RMS error  $e_{\text{RMS},1}$  according to (17) is smaller than 3%. The chosen parameters are  $L = 11$ , i.e., ten hidden layers, hyperbolic tangent as activation function  $\sigma$  and  $\eta = 0.01$  for the learning rate.

After the training, the prediction for the 40 test data points is computed by forward propagation with the trained weights and biases. The result is plotted in Fig. 1 together with the target values. It is obvious that a high prediction accuracy has been achieved. This is substantiated by a small RMS error of 5.8%.

##### 3.1.2 Retraining by backpropagation or EnKF for material 9SMnPb36

For the prediction of the specific cutting forces of another material, here 9SMnPb36, the weights and biases of the trained network have to be adapted due to the changed material properties. The input values for the three input quantities  $h$ ,  $\gamma$  and  $v$  are exactly the same as for C35G, only the specific cutting forces  $k_s$  (targets) differ. Hence, the setting is similar such that starting the training of a new ResNet by initializing its weights and biases with the already determined ones

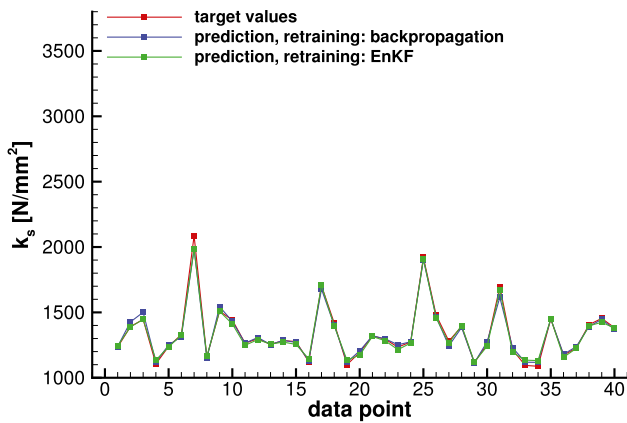


Fig. 2 Prediction of specific cutting forces  $k_s$  of material 9SMnPb36 for 40 test data points after retraining by backpropagation or EnKF

from the C35G case instead of using random values leads to a reduced training time. This procedure is called retraining. For a comparison of runtimes and prediction accuracy, the retraining is carried out by both the backpropagation algorithm of the ResNet according to (14-16) and the EnKF algorithm according to (18-20).

Since existing weights and biases are adapted, the resulting ResNet must have the same number of layers, in this case  $L = 11$ . For the retraining by backpropagation, the other two parameters are also chosen to stay the same, i.e., hyperbolic tangent as activation function  $\sigma$  and  $\eta = 0.01$  for the learning rate. The parameters for the retraining with the EnKF are set to be  $N_j = 100$  particles (ensemble size) and  $\sigma_c = 0.01$ . Since the measurement noise is unknown, we use the covariance matrix  $\Gamma^*$  and set  $\Gamma^* = n$ , where  $n$  denotes the current number of the iteration such that  $\Gamma^*$  is successively increased. Note that  $\Gamma^*$  is a scalar here because there is only one output quantity. For the retraining, again 51 of the 91 data points available for 9SMnPb36 are arbitrarily selected such that 40 data points are left for testing. The retraining is terminated when  $e_{RMS,1}$  is smaller than 3%

The results of forward propagation for the 40 test data points after retraining by backpropagation or EnKF are depicted in Fig. 2. Note that the plot uses the same scale on the y-axis for the specific cutting force  $k_s$  as Fig. 1 to demonstrate the significant difference in the target values compared with the C35G case. However, the predictions computed after retraining again show a high accuracy. The RMS errors are 5.5% and 4.8% for the network

retrained by backpropagation and the one retrained by EnKF, respectively.

The computation time for the retraining by backpropagation is 78.6 s. The convergence behavior of the EnKF algorithm is influenced by the normal distribution  $\mathcal{N}(\mathbf{u}_{init}, \sigma_c I_d)$  and the random choice of the input vector  $\mathbf{x}^{(0)}$  and the corresponding target vector  $\mathbf{y}$  in each iteration of the retraining. Hence, Table 2 contains the computation times and RMS errors of ten runs of the retraining by EnKF for 9SMnPb36. The computation time varies between 17.7 and 29.8 s which corresponds to a speed-up of 2.6 to 4.4 compared with the retraining by backpropagation. On average (23.4 s), the speed-up is 3.4. Together with the RMS errors (3.9% to 5.2%), which are smaller than the RMS error of the network retrained by backpropagation (5.5%) for all ten runs, this demonstrates the high suitability of the EnKF for retraining the weights and biases of a neural network.

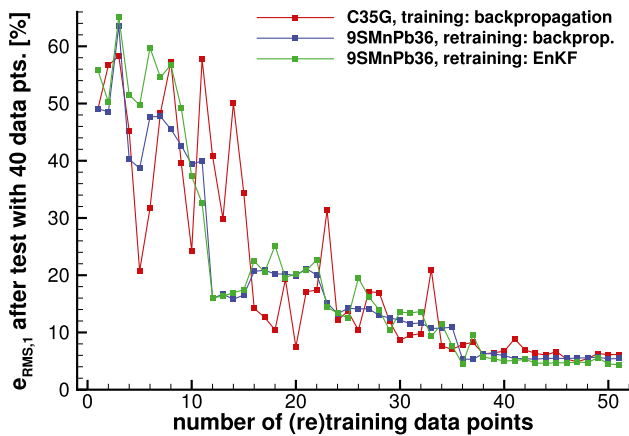
### 3.1.3 Dependence of prediction accuracy on number of training points

For training and retraining the ResNets in the previous sections, we have used  $N = 51$  data points. Now, we investigate the dependence of the accuracy of the prediction of a (re)trained net on  $N$  in the case of the cutting forces. Since gathering training data, e.g., from experiments, is often expensive,  $N$  should be kept as small as possible to reduce costs. In the following, we do the training for C35G and the retraining for 9SMnPb36 as before but for each  $N \in \{1, \dots, 51\}$ . After that, the trained or retrained ResNets are tested with the same 40 data points as before and the RMS error  $e_{RMS,1}$  for the prediction of these test data is computed. Note that each retraining is initialized with the weights and biases determined in Sect. 3.1.1.

The results for the errors are plotted in Fig. 3. The behavior for the three cases, i.e., ResNets trained for C35G and retrained for 9SMnPb36 by backpropagation or EnKF, is similar. As expected, for small values of  $N$  with approximately  $N < 15$ , the resulting ResNets lead to inaccurate predictions with errors larger than 20%. For  $N > 35$ , all errors are below 10%, which is sufficient for the application at hand. With 36 training points,  $e_{RMS,1}$  is 7.9% (C35G), 5.3% (9SMnPb36, retrained by backpropagation) and 4.4% (9SMnPb36, retrained by EnKF).

Table 2 Computation times of the EnKF retraining algorithm and resulting RMS errors for the cutting force test data for ten runs

Run	1	2	3	4	5	6	7	8	9	10	Average
time [s]	21.1	26.4	27.2	22.8	21.8	29.8	26.5	17.7	20.8	20.2	23.4
error [%]	4.8	3.9	4.8	4.8	4.8	5.2	5.2	4.6	5.2	4.8	4.8



**Fig. 3** RMS error  $e_{RMS,1}$  of the prediction of specific cutting forces  $k_s$  for 40 test data points dependent on the number of data points used for training or retraining

### 3.2 Prediction of the plasma spraying coating process

At the Surface Engineering Institute (IOT) at RWTH Aachen University, models for the simulation of plasma spraying are developed [23, 24]. Plasma spraying is a coating process, where a powder is fed into a plasma jet which melts the powder and transports the particles towards a substrate with a high velocity. Due to a large number of influencing parameters and the turbulent flow behavior, the optimization of this coating process is time-consuming and expensive. The goal is to establish a digital twin of the simulation process which significantly reduces the computation time while still sufficiently predicting the general behavior of the particles. An approach for training a ResNet similar to the one presented in the following and a comparison of ResNet results with results of a support vector machine can be found in [25]. Further ResNet results are also available in [37].

#### 3.2.1 Data preparation

At the IOT, recently 45 simulations were carried out, where for each simulation five input quantities, namely primary gas flow rate, electric current, carrier gas flow rate, powder feed rate and stand-off distance of the substrate from the particle

injector, were fixed and only one input quantity, namely the grain size of the powder particles, was varied in a fixed interval, either 15–35  $\mu\text{m}$ , 35–55  $\mu\text{m}$  or 55–75  $\mu\text{m}$ . Exemplarily, input data of four simulations are listed in Table 3. Since each of the 45 simulations contains between 1,500 and 2,000 particles, a large dataset of more than 85,000 data points is available. The output quantities of the simulations are the particle coordinates  $x$  and  $z$  on the substrate, which is a plane at a fixed position  $y$ , and the particle in-flight temperatures and velocities  $T$  and  $v$ , respectively.

For training and testing a ResNet, we use 1,000 and 10,000 data points, respectively, randomly chosen from 20 simulations. For the subsequent retraining of the determined weights and biases by backpropagation or EnKF and testing the retrained networks, the same numbers of data points are randomly chosen from another 20 simulations.

#### 3.2.2 Training a ResNet

With data for the six input quantities mentioned above, a ResNet is trained to predict the four output quantities  $x$ ,  $z$ ,  $T$  and  $v$ . The chosen parameters are  $L = 6$  (five hidden layers), learning rate  $\eta = 0.0001$  and hyperbolic tangent as activation function  $\sigma$ . Due to the rather chaotic particle behavior which is caused by the turbulence of the flow and collisions of the powder particles, an accurate prediction of the coordinates, temperatures or velocities of single particles cannot be expected. This would only be possible if these effects were accounted for by the input quantities, but information on the turbulence of each particle is obviously not available when applying the trained ResNet to a new setting. However, the network should be capable of learning the average particle behavior. The training is stopped when the average RMS error  $\bar{e}_{RMS} = (\sum_{j=1}^{n_o} e_{RMS,j})/n_o$ , cf. (17), where  $n_o = 4$  is the number of output quantities, is smaller than 22% as this value has turned out to be close to the minimum error possible in this case.

For an easier identification of physical relations, e.g., a decreasing velocity with increasing particle size, in the plots of the results, all data are sorted according to the input particle (grain) size after testing as depicted in Fig. 4. The horizontal lines mark the used grain size distributions: for each

**Table 3** Exemplary input data for plasma spraying

Sim.	Primary gas Flow [SLPM]	Electric Current [A]	Carrier gas Flow [SLPM]	Powder feed Rate [g/min]	Stand-off Dist. [mm]	Grain sizes [ $\mu\text{m}$ ]
1	40.36	461.6	6.39	28.8	126	15–35
2	40.36	532.9	5.72	15.6	153	15–35
3	41.37	473.8	4.04	12.0	169	15–35
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
45	59.87	470.3	4.04	18.0	144	55–75



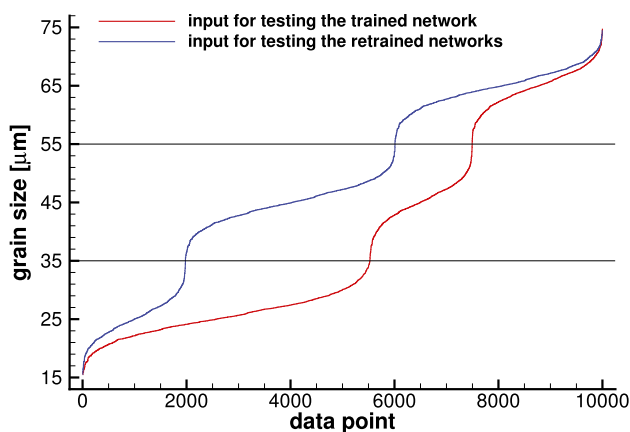


Fig. 4 Input particle (grain) size for testing the trained and retrained networks

of the 45 simulations by the IOT, the sizes of the input particles are 15–35 μm, 35–55 μm or 55–75 μm, see above and Table 3. Since the 10,000 data points were chosen randomly from 20 simulations, the numbers of data points per grain size interval differ: there are approximately 5,500, 2,000 and 2,500 particles for the grain sizes 15–35 μm, 35–55 μm and 55–75 μm, respectively, for testing the trained network. These numbers amount to approximately 2,000, 4,000 and 4,000 for testing the retrained networks.

The results of the forward propagation of the 10,000 test data points for the in-flight particle velocity  $v$  are shown in Fig. 5. The division into three parts corresponding to the input particle sizes is clearly visible. The prediction of the ResNet is less widespread than the spread of the targets. The turbulence of the flow leads to the effect that the velocities of two particles of the same size can differ significantly. The same holds for the particle temperature  $T$  and the coordinates  $x$  and  $z$  on the substrate. For the training of a ResNet,

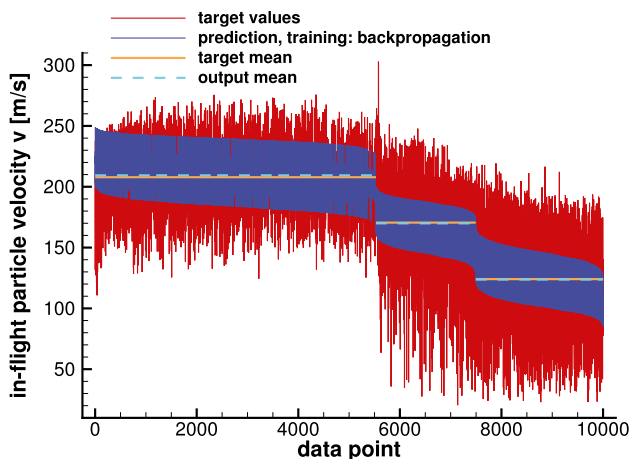


Fig. 5 Prediction of in-flight particle velocity  $v$  for 10,000 test data points after training by backpropagation

this corresponds to input values of two particles being the same while the targets differ. Hence, only the average value is learned in such a case. However, the trend of the velocity (decreasing with increasing grain size) and the three mean values according to the three grain size intervals 15–35 μm, 35–55 μm and 55–75 μm are accurately reproduced by the ResNet trained by backpropagation. In Fig. 5, target mean refers to the mean value of all target values inside the particular grain size interval and output mean is the corresponding mean value of all predictions. The deviations of the output mean values from the target mean values are 0.74% (15–35 μm), 0.35% (35–55 μm) and 0.26% (55–75 μm). With 21.9%, the average RMS error for the test data is even slightly lower than the one for the training data (22%, see above).

### 3.2.3 Retraining by backpropagation or EnKF

In contrast to the cutting force case in Sect. 3.1.2, here the input values for training and retraining differ, cf. Fig. 4. The parameters for the retraining by backpropagation are set to be the same as for the training, i.e.,  $L = 6$ ,  $\eta = 0.0001$  and hyperbolic tangent. For the retraining by EnKF, we use  $N_j = 150$  particles (ensemble size),  $\sigma_c = 0.005$  and  $\Gamma^* = 0.8nI_4$ . With both methods, a minimum average RMS error  $\bar{e}_{\text{RMS}} = 20.6\%$  can be obtained by the retraining.

The results of the forward propagation after retraining for the in-flight particle temperature  $T$  in Fig. 6 and for the particle coordinate  $x$  in Fig. 7 are of similar quality as the result after the initial training in Fig. 5 for the particle velocity  $v$ . Again, the division into three parts according to the three grain size intervals is visible, this time corresponding to the input data for retraining depicted in Fig. 4. Qualitatively, the predictions for the test data after retraining by backpropagation are similar to those after retraining by EnKF. As expected and described above, the prediction of the widespread single particle

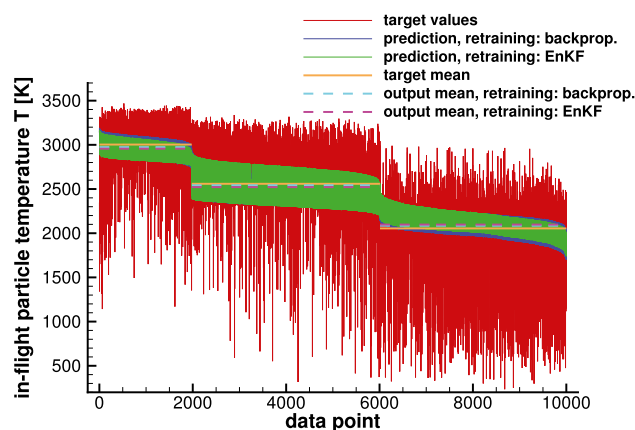
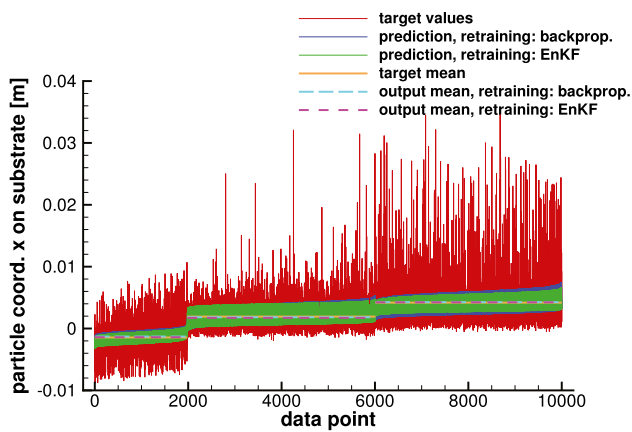


Fig. 6 Prediction of in-flight particle temperature  $T$  for 10,000 test data points after retraining by backpropagation or EnKF



**Fig. 7** Prediction of particle coordinate  $x$  on substrate for 10,000 test data points after retraining by backpropagation or EnKF

temperatures and coordinates is not possible due to the missing turbulence information. The average RMS errors regarding the test data are 21.4% for both methods. With increasing particle size, the temperature  $T$  tends to decrease and the coordinate  $x$  tends to increase. This behavior and the mean values, which are analogously computed to those in Fig. 5, are well reflected by the retrained networks. The deviations of the output mean values from the target mean values for the temperature are 0.97% and 1.55% (15–35  $\mu\text{m}$ ), 0.64% and 1.47% (35–55  $\mu\text{m}$ ) and 1.49% and 2.11% (55–75  $\mu\text{m}$ ), each for the test after retraining by backpropagation and by EnKF, respectively. Hence, the retraining by EnKF in this case leads to slightly larger deviations regarding the predicted mean values. However, a maximum deviation of 2.11% is clearly acceptable.

The computation time needed for retraining by backpropagation is 530.6 s. Table 4 lists ten runtimes for the retraining by EnKF. The speed-up lies between 2.0 and 3.8 with an average speed-up of 2.8. This again shows that using the EnKF for retraining the weights and biases of a neural network can lead to a significant reduction of computation time.

In summary, the results of the training and retraining show that a neural network is a suitable replacement for the simulation of the thermal spraying process when it comes to the prediction of the general particle behavior. While the average time needed for one simulation is about 3 h, the (re)training of a neural network only needs a few minutes, and the forward propagation with an already (re)trained network takes less than a second.

## 4 Discussion of potential for production engineering

Advanced mathematical algorithms such as shown in the previous sections can be integrated into standardized approaches for process validation, model-based process monitoring, control and quality predictions. The algorithms presented in this paper enhance common approaches of machine learning algorithms by means of development of new mathematical approaches. For an industrial usage, the application of algorithms needs to be validated with specific machining processes. Investigations showed that the performance of a specific algorithm is dependent on the use [38]. This could be related to the governing physical causalities and the underlying mathematical nature of the distinct processes. Once those relations are understood, an optimal algorithm for a specific problem can be recommended. Furthermore, data from physically similar processes could be used to retrain the models for new processes (transfer learning algorithms).

For the most manufacturing processes, advanced mathematical algorithms and machine learning can be applied to handle the vast amount of data currently unused for production. Automated approaches for feedback and learning (for instance, described in [39]) can be enhanced with algorithms such as presented in this paper. This can be used for control loops with focus on quality prediction or productivity or process validation. This initial validation of the process ensures mass production and is one of the most critical and cost-intensive processes in manufacturing.

For the manufacturing of high-integrity parts, this validation is furthermore certified from the respective authorities. If high-integrity parts fail, usually the whole system fails and hazards for the environment and human beings occur. As an example, blisks in aero engines are classified as critical parts and process validation is a highly cost-intensive process. Furthermore, this approach results in static processes which are not changed over time, i.e., new methodologies or tools can not be used for optimization. New approaches are very rare and mostly in research context (see [40]) rather than in real production environment. In these industries, however, data are collected throughout the whole manufacturing process and stored due to warranty reasons. These stored data, e.g., the cutting force data like shown in this paper, can be used for predictions of new processes with the help

**Table 4** Computation times of the EnKF retraining algorithm and resulting average RMS errors for the thermal spraying test data for ten runs

Run	1	2	3	4	5	6	7	8	9	10	Average
time [s]	183.0	138.3	261.6	158.3	206.9	157.0	171.1	254.3	202.5	155.7	<b>188.9</b>
error [%]	21.5	21.4	21.5	21.5	21.4	21.4	21.4	21.4	21.5	21.4	<b>21.4</b>

of advanced mathematical algorithms. While in the manufacturing of high-integrity parts, the demanded quality is the main objective, mass production industries such as car manufacturing are more concerned with cost and energy optimization [41]. However, the methods for process validation can be used in the same way, only with other target variables (for instance, minimizing the occurring cutting force while ensuring quality). For future applications and digitalized manufacturing environments, the importance of algorithms for data analytics will increase [42]. The algorithms presented in this paper have shown a high potential for usage in their respective field of application.

## 5 Conclusion

In this paper, we have shown how mathematical techniques may be applied to machine learning methods to investigate the algorithmic behavior of those methods and therefore help improve their performance. Residual neural networks (ResNets) as stated in this work allow for a mathematical analysis by setting up a time-continuous formulation. By exploiting their underlying structure, which is of the type of a partial differential equation (PDE), properties of a ResNet regarding their convergence behavior or possible outcome can be proven. This is possible at least for simple cases such as a single-input ResNet.

Advanced mathematical algorithms such as the ensemble Kalman filtering (EnKF) accelerate common tasks. In particular, the computation time for the retraining of neural networks can be significantly reduced if the frequently used backpropagation algorithm is replaced by the EnKF algorithm. The latter is parallelizable being a crucial advantage for its efficiency. However, even without this parallelization, we observed speed-up factors between 2.0 and 4.4 for our two use cases. For recurrent tasks, such an acceleration saves (training) time and therefore also costs. The predictions of a ResNet retrained by the EnKF algorithm are of similar accuracy as the ones of a ResNet retrained by backpropagation, even with the simplified covariance matrix  $\Gamma^*$  introduced in this work.

In production engineering, the availability of adequate data is often not a problem but their profitable use is. As more and more data become available due to an increasing number of digitalized manufacturing environments, the importance of the application of advanced mathematical algorithms in order to use the data increases. The mathematical analysis of machine learning algorithms is necessary to understand and improve algorithms. In combination with more available data, this results in an interdisciplinary task.

Up to now, properties of neural networks are only mathematically rigorously proven for simple networks. Hence, further studies are necessary starting from the presented

networks. As example, the use case of the plasma spraying process has shown that a neural network is capable of predicting the average particle behavior. However, for the accurate prediction of the behavior of each single particle, an advanced approach such as a physics-informed neural network [43] has to be applied, linking a neural network to the governing equations of the process.

**Acknowledgements** The authors thank S. R. Dokhanchi, H. Heinemann and W. Wietheger (Surface Engineering Institute (IOT), RWTH Aachen University) for providing the plasma spraying data.

**Funding** Open Access funding enabled and organized by Projekt DEAL. Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen der Exzellenzstrategie des Bundes und der Länder – EXC-2023 Internet of Production - 390621612. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production - 390621612. Funding through HIDSS-004 is acknowledged.

**Availability of data and material** Not applicable.

**Code availability** Not applicable.

## Declarations

**Ethics approval** Not applicable.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.

**Competing interests** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. *Science* 349(6245):255–260
2. Joshi AV (2019) *Machine Learning and Artificial Intelligence*. Springer
3. Kang Z, Catal C, Tekinerdogan B (2020) Machine learning applications in production lines: A systematic literature review. *Comput Ind Eng* 149:106773
4. Usuga Cadavid JP, Lamouri S, Grabot B, Pellerin R, Fortin A (2020) Machine learning applied in production planning and

- control: a state-of-the-art in the era of industry 4.0. *J Intell Manuf* 31:1531–1558
5. Weichert D, Link P, Stoll A, Rüping S, Ihlenfeldt S, Wrobel S (2019) A review of machine learning for the optimization of production processes. *Int J Adv Manuf Technol* 104:1889–1902
  6. Maschler B, Braun D, Jazdi N, Weyrich M (2021) Transfer learning as an enabler of the intelligent digital twin. *Procedia CIRP* 100:127–132
  7. Weiss K, Khoshgoftaar TM, Wang D (2016) A survey of transfer learning. *J Big Data* 3:9
  8. Haber E, Lucka F, Ruthotto L (2018) Never look back - A modified EnKF method and its application to the training of neural networks without back propagation. Preprint [arXiv:1805.08034](https://arxiv.org/abs/1805.08034)
  9. Kovachki NB, Stuart AM (2019) Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Probl* 35(9):095005
  10. Watanabe K, Tzafestas SG (1990) Learning algorithms for neural networks with the Kalman filters. *J Intell Robot Syst* 3(4):305–319
  11. Yegenoglu A, Diaz S, Krajsek K, Herty M (2020) Ensemble Kalman filter optimizing deep neural networks. In *Conference on Machine Learning, Optimization and Data Science*, volume 12514. Springer LNCS Proceedings
  12. Iglesias M, Law K, Stuart AM (2013) Ensemble Kalman methods for inverse problems. *Inverse Prob* 29(4):045001
  13. Schillings C, Stuart AM (2017) Analysis of the Ensemble Kalman Filter for Inverse Problems. *SIAM J Numer Anal* 55(3):1264–1290
  14. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778
  15. Chen TQ, Rubanova Y, Bettencourt J, Duvenaud DK (2018) Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583
  16. Lu Y, Zhong A, Li Q, Dong B (2018) Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In J. Dy and A. Krause, editors, 35th International Conference on Machine Learning, ICML 2018, 35th International Conference on Machine Learning, ICML 2018, pages 5181–5190. International Machine Learning Society (IMLS), 2018. 35th International Conference on Machine Learning, ICML 2018 ; Conference date: 10-07-2018 Through 15-07-2018
  17. Ruthotto L, Haber E (2020) Deep neural networks motivated by partial differential equations. *J Math Imaging Vis* 62:352–364
  18. Araújo D, Oliveira RI, Yukimura D (2019) A mean-field limit for certain deep neural networks. *arXiv preprint [arXiv:1906.00193](https://arxiv.org/abs/1906.00193)*
  19. Mei S, Montanari A, Nguyen P-M (2018) A mean field view of the landscape of two-layer neural networks. *Proc Natl Acad Sci* 115(33):E7665–E7671
  20. Sirignano J, Spiliopoulos K (2019) Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*
  21. Trimborn T, Gerster S, Visconti G (2020) Spectral methods to study the robustness of residual neural networks with infinite layers. *Found Data Sci* 2(3):257–278
  22. Kienzle O, Victor H (1957) Spezifische Schnittkräfte bei der Metallbearbeitung. *Werkstofftechnik und Maschinenbau* 47(H5):224–225
  23. Bobzin K, Öte M (2017) Modeling plasma-particle interaction in multi-arc plasma spraying. *J Therm Spray Technol* 26:279–291
  24. Bobzin K, Öte M, Schein J, Zimmermann S (2017) Numerical study on plasma jet and particle behavior in multi-arc plasma spraying. *J Therm Spray Technol* 26:811–830
  25. Bobzin K, Wietheger W, Heinemann H, Dokhanchi SR, Rom M, Visconti G (2021) Prediction of particle properties in plasma spraying based on machine learning. *J Therm Spray Technol* 30:1751–1764
  26. Lin H, Jegelka S (2018) Resnet with one-neuron hidden layers is a universal approximator. *NIPS'18*, page 6172–6181, Red Hook, NY, USA, 2018. Curran Associates Inc
  27. Goldberger J, Ben-Reuven E (2017) Training deep neural-networks using a noise adaptation layer. In *ICLR*
  28. Noh H, You T, Mun J, Han B (2017) Regularizing deep neural networks by noise: Its interpretation and optimization. In Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors, *Advances in Neural Information Processing Systems* 30, pages 5109–5118. Curran Associates, Inc., 2017
  29. Tran D, Dusenberry MW, Wilk MVD, Hafner D (2019) Bayesian layers: A module for neural network uncertainty. In *NeurIPS*
  30. Carrillo JA, Fornasier M, Toscani G, Vecil F (2010) Mathematical Modeling of Collective Behavior in Socio-Economic and Life Sciences, chapter Particle, kinetic, and hydrodynamic models of swarming, pages 297–336. *Modeling and Simulation in Science, Engineering and Technology*. Birkhäuser Boston
  31. Golse F (2016) On the dynamics of large particle systems in the mean field limit. In *Macroscopic and large scale phenomena: coarse graining, mean field limits and ergodicity*, pages 1–144. Springer
  32. Ha S-Y, Jin S, Kim D (2020) Convergence of a first-order consensus-based global optimization algorithm. *Math Models Methods Appl Sci* 30(12):2417–2444
  33. Jabin P-E (2014) A review of the mean field limits for Vlasov equations. *Kinetic & Related Models* 7(4):661–711
  34. Herty M, Trimborn T, Visconti G (2022) Mean-field and kinetic descriptions of neural differential equations. *Found Data Sci* 4(2):271–298
  35. Clarke FH (1990) *Optimization and Nonsmooth Analysis*. SIAM
  36. Kalman RE (1960) A new approach to linear filtering and prediction problems. *ASME J Basic Eng T*
  37. Herty M, Rom M, Visconti G (2021) Einsparung kostenintensiver Experimente und Simulationen durch Maschinelles Lernen. In D. Trauth, T. Bergs, and W. Prinz, editors, *Monetarisierung von technischen Daten - Innovationen aus Industrie und Forschung*, pages 691–709, Springer Vieweg
  38. Mayr A, Kißkalt D, Meiners M, Lutz B, Schäfer F, Seidel R, Selmaier A, Fuchs J, Metzner M, Blank A, Franke J (2019) Machine learning in production - Potentials, challenges and exemplary applications. *Procedia CIRP* 86:49–54
  39. Denkena B, Dittrich M-A, Wilmsmeier S (2019) Automated production data feedback for adaptive work planning and production control. *Procedia Manufacturing* 28:18–23
  40. Fricke K, Gierlings S, Ganser P, Venek T, Bergs T (2020) Geometry model and approach for future blisk LCA. *IOP Conf Ser Mater Sci Eng* 1024:012067
  41. Giampieri A, Ling-Chin J, Ma Z, Smallbone A, Roskilly AP (2020) A review of the current automotive manufacturing practice from an energy perspective. *Appl Energy* 261:114074
  42. de Man JC, Strandhagen JO (2017) An Industry 4.0 research agenda for sustainable business models. *Procedia CIRP* 63:721–726
  43. Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 378:686–707

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.