J Therm Spray Tech (2021) 30:1751–1764
https://doi.org/10.1007/s11666-021-01239-2

PEER REVIEWED

Prediction of Particle Properties in Plasma Spraying Based on Machine Learning

K. Bobzin[1] · W. Wietheger[1] · H. Heinemann[1] · S. R. Dokhanchi[1] · M. Rom[2] · G. Visconti[2]

Submitted: 5 February 2021 / in revised form: 2 July 2021 / Accepted: 2 July 2021 / Published online: 2 August 2021

Abstract Thermal spraying processes include complex nonlinear interdependencies among process parameters, in-flight particle properties and coating structure. Therefore, employing computer-aided methods is essential to quantify these complex relationships and subsequently enhance the process reproducibility. Typically, classic modeling approaches are pursued to understand these interactions. While these approaches are able to capture very complex systems, the increasingly sophisticated models have the drawback of requiring considerable calculation time. In this study, two different Machine Learning (ML) methods, Residual Neural Network (ResNet) and Support Vector Machine (SVM), were used to estimate the in-flight particle properties in plasma spraying in a much faster manner. To this end, data sets comprising the process parameters such as electrical current and gas flow as well as the in-flight particle velocities, temperatures and positions have been extracted from a CFD simulation of the plasma jet. Furthermore, two Design of Experiments (DOE) methods, Central Composite Design (CCD) and Latin Hypercube Sampling (LHS), have been employed to cover a set of representative process parameters for training the ML models. The results show that the developed ML models are able to estimate the trends of particle properties precisely and dramatically faster than the computation-intensive CFD simulations.

✉ S. R. Dokhanchi
dokhanchi@iot.rwth-aachen.de

[1] Surface Engineering Institute (IOT), RWTH Aachen University, Aachen, Germany

[2] Institute for Geometry and Applied Mathematics (IGPM), RWTH Aachen University, Aachen, Germany

Keywords atmospheric plasma spraying · CFD simulation · computational speed-up · design of experiments · in-flight particle properties · machine learning · metamodel

Abbreviations/Nomenclature

ANN	Artificial Neural Network
CCD	Central Composite Design
CFD	Computational Fluid Dynamics
LHS	Latin Hypercube Sampling
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
ResNet	Residual Neural Network
SVM	Support Vector Machine
b	Bias parameter, SVM
$b^{(l)}$	Bias vector, ResNet
C	Constant, SVM
e	Square error for a single training example, ResNet
e_{tr}	Mean square error for training set, ResNet
$f(x)$	True values, SVM
$g(x)$	Prediction values, SVM
$k(x_i, x_j)$	Kernel function, SVM
l	Index of layer, ResNet
L	Number of layers, ResNet
N	Number of data points/particles
n_i	Number of test data points of simulation i
n_N	Number of neurons per hidden layer, ResNet
n_O	Number of outputs, ResNet
P	Number of predictors, SVM
p_i	General prediction values, SVM or ResNet
R_{sq}	R-squared error
$\bar{t}$	Mean of general true values, SVM or ResNet
t_i	General true values, SVM or ResNet

T_p	In-flight particle temperature
$\overline{T}_p$	Mean in-flight particle temperature over all simulations ("grandmean")
$\overline{T}_{p,i}$	Mean in-flight particle temperature of simulation i
$T_{p,i,j}$	In-flight particle temperature of particle j of simulation i
v_p	In-flight particle velocity
w	Normal vector, SVM
$W^{(l)}$	Weights matrix, ResNet
$x^{(0)}$	Vector of input values, ResNet
$x^{(L)}$	Vector of prediction values, ResNet
x_p	In-flight particle x-coordinate
y	Vector of target values, ResNet
z_p	In-flight particle z-coordinate
$\alpha_i\alpha_i^*$	Lagrange multipliers, SVM
γ	Kernel scale
$\delta^{(l)}$	Local error, ResNet
ε	Upper error bound, SVM
η	Learning rate, ResNet
ξ, ξ^*	Slack variables, SVM
σ_1, σ_2	Activation functions, ResNet
$\sigma_1^{'}, \sigma_2^{'}$	Derivatives of act. functions, ResNet
$\phi(x)$	Mapping function, SVM
Ω	Input variable space, SVM
$\tilde{\Omega}$	Feature space, SVM

Introduction

The coating process in Thermal Spraying (TS) is associated with many complex physical phenomena. Due to the large number of parameters involved in this coating technology as well as the nonlinear relationships between these parameters, precise control and optimization of the TS processes is a lengthy and expensive undertaking. Not all of the influencing parameters can be controlled, because on the one hand the effect of many variables on the coating process is not quantitatively measurable, and on the other hand the technical possibilities for an adequate process monitoring are still lacking. Hence, simulation and modeling approaches like the Computational Fluid Dynamics (CFD) are often employed to capture the involved complex physical phenomena. Although CFD offers high potential for understanding the sub-processes of the TS coating technology, the tradeoff between the accuracy of the model and the computational cost has been always a challenge in CFD problems. The simulation of the particle free-jet in a multi-arc plasma spraying process, which is the focus of this study, requires elevated computational cost, while not sacrificing the accuracy of the model(Ref 1).

A promising possibility for substitution of the computationally expensive CFD simulations in plasma spraying is to create a Digital Twin of the process using Machine Learning (ML) algorithms. Digital Twin is referred to as a virtual and computerized representation of a physical system in real space including the data and information that ties the virtual and real systems together (Ref 2). This digital replication occurs mainly by integration of the artificial intelligence methods, with the aim of system optimization, monitoring as well as prognostics (Ref 3). This leads subsequently to greater efficiency, accuracy and economic benefits for the considered system (Ref 4). The majority of prior research works have used experimental data sets to create Digital Twins for the TS process variants with the objective of predicting the particle properties or controlling the process parameters (Ref 5, 6). There are only few studies in the literature that used simulation data sets for training ML models in TS (Ref 7). The motivation of using simulation results is the opportunity to cover a broad range of process parameters, while providing that much experimental data is barely possible. This results not only in enhancement of the prediction accuracy of the model, but also in speeding up computations dramatically.

The goal of the present study is to take the primary steps toward building up a fast Digital Twin for the plasma spraying process to predict the in-flight particle properties based on various input process parameters using ML methods. To this end, several sets of process parameters and particle properties are acquired from CFD simulations of the plasma jet. The data preparation is carried out using two different design of experiments (DOE) methods, namely Central Composite Design (CCD) and Latin Hypercube Sampling (LHS). Finally, the prepared data are fed into a Residual Neural Network (ResNet) and a Support Vector Machine (SVM) to predict the particle properties. The results of the different ML models and DOE methods are then compared with each other in terms of the calculated prediction accuracy. Due to the randomness of the particle behavior caused by particle collisions and the turbulence of the plasma flow, a precise prediction of the properties of each single particle cannot be expected with the ML methods at hand. However, the accurate prediction of average particle properties serves as a key performance indicator in plasma spraying and can significantly help, for example, in investigating the interrelationships between process parameters and coating properties. Hence, the objective in this work is to accurately predict the average particle behavior depending on different sets of process parameters.

Numerical Modeling

The simulation data sets of this study are obtained from a former numerical model, simulating the plasma spraying process of a three-cathode plasma generator performed at the Surface Engineering Institute. To resolve different physical phenomena and reduce the model complexity of the entire system, the plasma spraying process is divided into two sub-processes that are modeled separately: the plasma generator model and the plasma jet model. In the plasma generator model, the flow characteristics at the plasma generator outlet including the temperature and velocity profiles as well as the profiles of turbulent kinetic energy and turbulent eddy dissipation were determined. By using these determined profiles as a boundary condition at the inlet of the plasma jet model, the two sub-models are coupled. A two-equation Shear Stress Transport (SST) turbulence model was used to simulate the turbulence inside the plasma generator as well as in the plasma jet. A detailed description of the numerical modeling used in this study can be found in (Ref 8, 9). For an accurate description of the plasma-particle interaction in plasma spraying, the influences of the plasma on the particles and vice versa were considered in the plasma jet model in a two-way coupled manner (Ref 10). Furthermore, a validation of the plasma generator and the plasma jet models was conducted by comparing numerical results to experimental data (Ref 11).

Figure 1 shows the simulated particle trajectories and their temperatures inside the plasma jet exemplary for one simulation. For each simulation, a virtual clipping plane is defined to export the particle properties at specific stand-off distances. The particle properties include the in-flight particle coordinates on the clipping plane, the velocities and the temperatures. The simulation models were created in ANSYS CFX version 20.2 (ANSYS, Inc., Canonsburg, USA). For each simulation, the calculated number of particle trajectories was set to 2000. Aluminium oxide was used as the feedstock material for the simulations. Further details regarding the procedure of preparing the simulation data are described in the next section.

Data Preparation

Simulations often involve larger numbers of variables compared to physical experiments. It is necessary to find a set of input parameters, namely the design matrix, so that potentially the best-fitting predictive model can be constructed on the resulting data sets formed by the design matrix (Ref 12). Furthermore, this allows understanding the cause-and-effect relationships in the system by changing the designed input variables and observing the resulting changes in the system output (Ref 13). Therefore, two different DOE methods, CCD and LHS, were employed in this study to cover a set of representative input process parameters for the simulations. The parameter setup for the CCD and LHS methods is given in Table 1. Totally six different process parameters were considered for the DOE approach: primary gas flow (Argon), electric current,

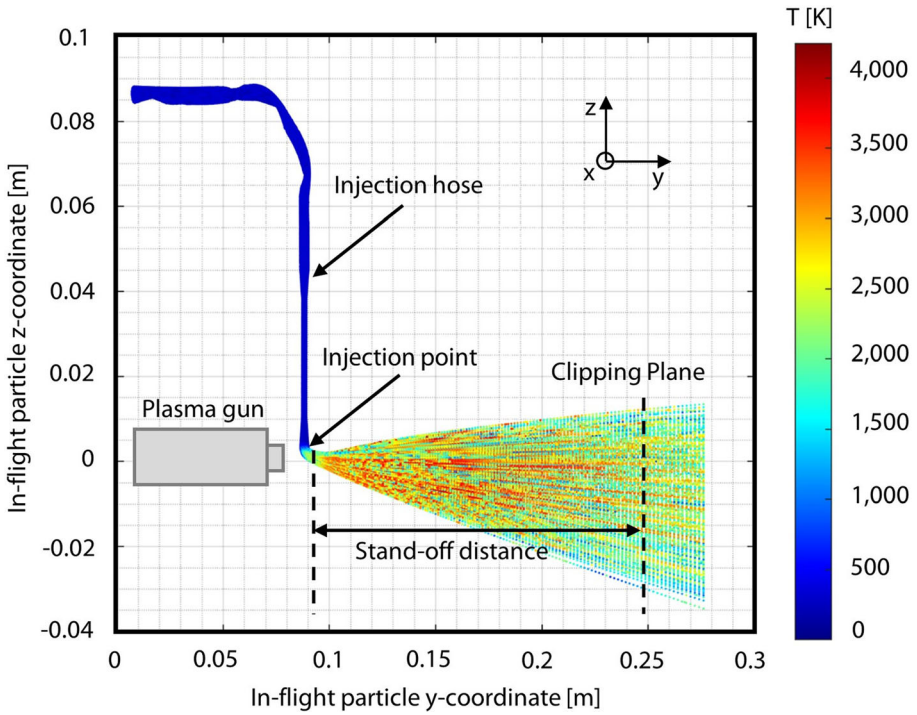

Fig. 1 Exemplary simulated particle trajectories and their temperatures in plasma jet

carrier gas flow, powder feed rate, particle size distribution at the injection point and stand-off distance. The particle sizes were divided into three different fractions to cover the broad spectrum of the possible particle size distributions in plasma spraying. The DOE methods were implemented in the MATLAB environment and were linked with the batch job scheduler of the simulation runs to create an automated data preparation pipeline. Overall, 45 simulations were carried out for the CCD data sets and another 45 simulations for the LHS data sets. In the following, both DOE methods and the structure of the data for the simulations are briefly described.

Central Composite Design (CCD)

CCD is based on a two-level full or fractional factorial design, which has additionally 2k (k is the number of independent variables) points between the axes and a set of repeated points at the centroid (N_0) (Ref 14). Figure 2 shows a geometric view of a CCD for a two-factor full factorial design. CCD is used widely in constructing second-order response surface models (Ref 15).

Random errors are inevitable in physical experiments, and the output may be different even with the same experimental settings. In the contrary, the computer experiments are deterministic and multiple trials result in identical outputs. Hence, carrying out several runs at the centroid is only meaningful in physical experiments (Ref 12). In this study, the number of computational experiments was set to 45, which corresponds to a CCD with 6-factor fractional design ($2^{k-1} + 2k + N_0$).

Table 1 Parameter setup for the DOE methods

Parameter [unit]	Interval
Primary gas flow, SLPM	40-60
Electric current, A	400-540
Carrier gas flow, SLPM	3.5-7
Powder feed rate, g/min	10-30
Particle size distribution, μm	−35 +15; −55 +35; −75 +55
Stand-off distance, mm	100-180

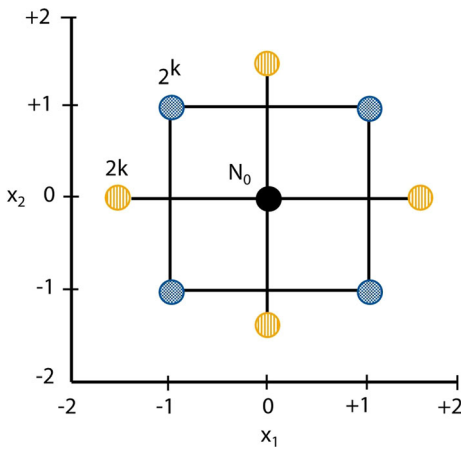

Fig. 2 Geometric view of central composite design for k = 2 factors

Latin Hypercube Sampling (LHS)

LHS is one of the most popular space-filling designs that aims at reducing the variance of sample mean (Ref 16). It is a stratified sampling technique, which divides the multi-dimensional experimental domain into N strata of equal marginal probability, where N is the number of sample points, in a way that each stratum contains only one sample point along each space dimension and then samples once from each stratum (Ref 12).

The maximin distance criteria can be imposed as an optimality criterion for construction of LHS to further decrease the variance of the sample mean. A maximin LHS maximizes the minimum distance between each pair of experimental points within the experimental domain, see Fig. 3. This optimality criterion ensures that the experimental points are spread out uniformly through the domain and therefore, no point lies too far away from a design point (Ref 17). This results in an enhancement of the prediction accuracy of the constructed model. LHS is a very suitable and powerful DOE technique for computer experimentation, which can serve various numbers of runs and input variables. In this study, the same number of runs as the CCD method was used for the LHS method to ensure the comparability of the results.

Structure of Simulation Data

As mentioned earlier, for each of the DOE methods introduced in the above "Central Composite Design (CCD)" and "Latin Hypercube Sampling (LHS)" sections, 45 simulations are performed, respectively, with different input process parameters, see Table 1. For instance, the simulation data sets gathered from the LHS method for the parameters primary gas flow, electric current, carrier gas flow, powder feed rate, particle size distribution and stand-off distance, respectively, are:

1. 40.36 SLPM, 461.6 A, 6.39 SLPM, 28.8 g/min, −35 +15 μm, 126 mm
2. 40.36 SLPM, 532.9 A, 5.72 SLPM, 15.6 g/min, −35 +15 μm, 153 mm
3. 41.37 SLPM, 473.8 A, 4.04 SLPM, 12.0 g/min, −35 +15 μm, 169 mm

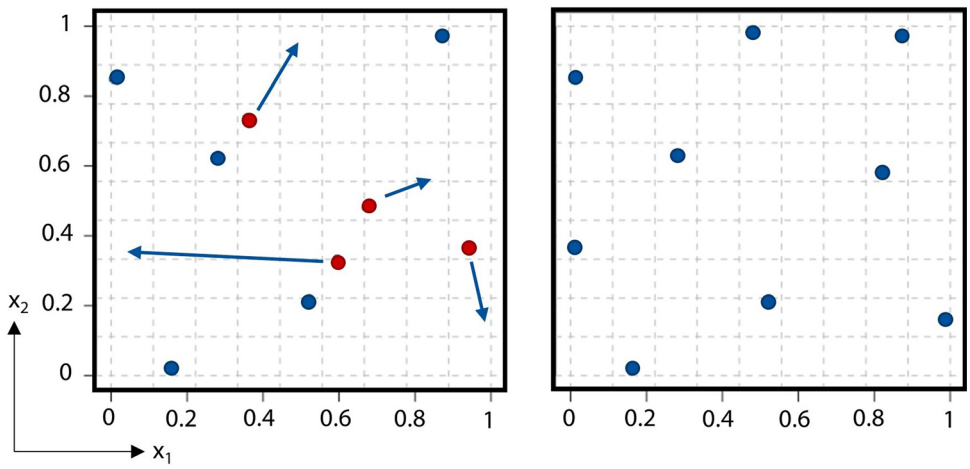

Fig. 3 Transformation of a 2D LHS (left) to a maximin LHS (right)

⋮

45. 59.87 SLPM, 470.3 A, 4.04 SLPM, 18.0 g/min, −75 +55 µm, 144 mm

The CCD simulation data are also structured into 45 simulations. As it is evident from the above structure, within each of the 45 CCD or LHS simulations, only the particle size can vary in the specified range and the five other process parameters are kept constant. The outputs of the simulations are the in-flight particle properties of the 2000 simulated particle trajectories per simulation, respectively. However, regarding the different process parameters within each simulation, not all of the 2000 simulated particle trajectories can reach the specified stand-off distance. Hence, the exact number of output data per simulation for the 45 CCD or LHS simulations is not the same and can vary between 1500 and 2000 particle trajectories. The inputs and outputs of each simulation are provided with indices to be able to assign the particles of each simulation for the ML models.

Machine Learning Algorithms

The DOE methods provide the representative simulation data sets for training the ML models that are SVM and ResNet. The inputs of the prediction models are the process parameters listed in Table 1. The outputs are the particle properties including the in-flight particle temperatures T_p [K] and velocities v_p [m/s] as well as the in-flight particle x-coordinates x_p [m] and z-coordinates z_p [m] at specific stand-off distances on the virtual substrate (clipping plane).

Due to the collisions of the particles and the turbulence of the plasma flow, even particles of nearly the same size can have different coordinates in the plasma jet and thus, vary greatly in temperature and velocity. Hence, it can be barely expected that the ML models could predict single particle properties with high accuracy, but the average particle properties should be reproducible with a sufficiently small error.

The results from the LHS and CCD methods were each partitioned into one training data set and one test data set. From each of the respective 45 simulations, 75% of the data are used as training data and the remaining 25% as test data. As described in the previous section, the number of particles per simulation may differ and thus the overall number of particles in the training and test data sets for the CCD and LHS methods is different. The training data for CCD contain 64,858 particles and the test data include 21,612 particles, while these numbers amount to 64,728 and 21,566 for the LHS, respectively. The training and test data used for the two ML models were kept identical. Even though both the SVM and ResNet are trained and tested with the whole training and test data out of the 45 simulations respectively, the allocation of the particles to each simulation is still known by use of the indices mentioned in "Structure of Simulation Data" section as data labels. This is utilized in the evaluation of the results in "Results and Discussion" section. In the following, the SVM and ResNet algorithms used in this study are described.

Support Vector Machine (SVM)

The SVM theory introduced by Vapnik (Ref 18) has faced dramatic attention in statistical learning theory and has been increasingly applied by researchers in various fields, where the TS forms no exception (Ref 19, 20). SVM is a supervised-learning algorithm that uses structural risk minimization and a symmetrical loss function, which equally penalizes high and low errors. An important property of the SVM regression is that its computational complexity does not depend on the dimension of the input

space. Furthermore, it has great generalization capability, with high prediction accuracy (Ref 21).

The goal of linear SVM regression is to find an approximated hyperplane for the true model f in the form of:

$$g(x) = \langle w, \phi(x) \rangle + b \quad \text{(Eq 1)}$$

where w is the normal vector of g, ϕ is a mapping function, which could initially be considered as identity function, and b is a bias parameter. The predicted values from g should have a bounded deviation no more than ε from the true values $f(x)$, i.e.,

$$|g(x) - f(x)| \leq \varepsilon \quad \text{(Eq 2)}$$

The distance between the hyperplane g and the farthest point away is called margin and it is proportional to $\frac{1}{||w||}$. The boundary of a maximal margin is called a support vector, see Fig. 4. In addition, g should be maximally flat, i.e. $||w||$ should be as small as possible and the margin as large as possible (Ref 22).

In practical cases, this kind of hyperplane is not guaranteed to exist. In order to cope with otherwise infeasible constraints, the slack variables ξ and ξ^* are introduced to construct a soft margin hyperplane. Consequently, the constrained optimization problem could be formulated as (Ref 23):

$$\text{Minimize}: \quad \frac{1}{2}||w||^2 + C\sum_i \left(\xi_i + \xi_i^*\right) \quad \text{(Eq 3)}$$

$$\text{Subject to}: \quad g(x) - f(x) \leq \varepsilon + \xi_i^* \quad \text{(Eq 3.1)}$$

$$f(x) - g(x) \leq \varepsilon + \xi_i \quad \text{(Eq 3.2)}$$

$$\xi_i, \xi_i^* \geq 0 \quad \forall i = 1, \ldots, |\Omega| \quad \text{(Eq 3.3)}$$

where Ω denotes the input variable space and $C > 0$ is a constant that determines the penalties for training errors. A closed form representation of the regression hyperplane g could be derived from the dual form of the optimization problem above:

$$g(x) = \sum_i \left(\alpha_i^* - \alpha_i\right)\langle x_i, x \rangle + b \quad \text{(Eq 4)}$$

where α_i, α_i^* are Lagrange multipliers (Ref 23).

The already introduced linear form of SVM regression could be transformed into a nonlinear feature space via a nonlinear mapping $\phi : \Omega \rightarrow \tilde{\Omega}$. The dot product in $\tilde{\Omega}$ could be expressed by the kernel function $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. With the implicit mapping of kernel function k, it is possible to directly compute the hyperplane g in the nonlinear feature space. With this so-called kernel trick, the final form of the approximated hyperplane could be expressed as:

$$g(x) = \sum_i \left(\alpha_i^* - \alpha_i\right)k(x_i, x) + b \quad \text{(Eq 5)}$$

where the corresponding constrained optimization problem is now formulated in the transformed feature space $\tilde{\Omega}$ instead of in the original input variable space Ω, thanks to the implicit mapping ϕ and the kernel function k (Ref 24). One advantage of SVM is that although the training involves nonlinear optimization, the corresponding objective function is convex, and therefore, any local solution represents also a global optimum (Ref 25).

In this study, the implementation of the SVM regression algorithm was carried out using the Statistics and Machine Learning Toolbox of MATLAB. In order to make the inputs and targets insensitive to the scales and magnitudes on which they are processed, a preprocessing step has been carried out to standardize the training data sets. The standardization was done based on the so-called z-score method, in which the corresponding standardized data have a mean value of zero and a standard deviation of one. Hence, the shape of the original data set is retained.

Four single-output SVM models, corresponding to the four outputs, for each of the two DOE methods (LHS and CCD) were developed. For training the regression models, Gaussian kernels based on Eq 6 were employed, where γ represents the kernel scale.

$$k(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|^2}{2\gamma^2}\right) \quad \text{(Eq 6)}$$

The training of the SVM models has been conducted with different kernel scales as given in Table 2 to choose the best prediction accuracy. The term P in Table 2 denotes the number of predictors, which is equal to $P = 6$ in this study. Furthermore, a 10-fold cross-validation was used to

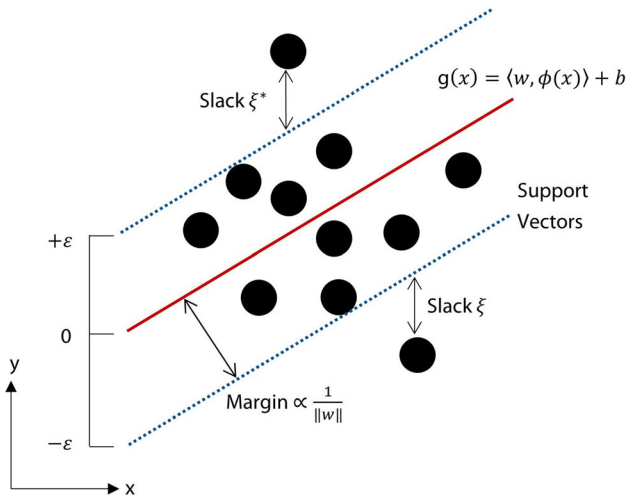

Fig. 4 Illustration of the support vectors, margins and slack variables in SVM regression

Table 2 Kernel scales of different Gaussian kernels applied for training the SVM models

Kernel type	Kernel scale
Fine Gaussian	$\gamma = \sqrt{\frac{P}{32}}$
Medium Gaussian	$\gamma = \sqrt{\frac{P}{2}}$
Coarse Gaussian	$\gamma = \sqrt{8P}$

analyze the level of generalization and prevent possible overfitting.

Residual Neural Network (ResNet)

The classical Artificial Neural Network (ANN) is a multilayer perceptron represented by a mathematical function which maps input values to output values. For an ANN with L layers and a vector $x^{(0)}$ containing input values, the output vector $x^{(L)}$ representing the prediction of the ANN is determined by:

$$x^{(L)} = \sigma_2\left(W^{(L)^T}\sigma_1\left(W^{(L-1)^T}\sigma_1\left(\cdots\sigma_1\left(W^{(1)^T}x^{(0)} + b^{(1)}\right) + \cdots\right) + b^{(L-1)}\right) + b^{(L)}\right) \quad \text{(Eq 7)}$$

where $W^{(l)}$ and $b^{(l)}$, $l = 1, \ldots, L$, are weights matrices and bias vectors, respectively, σ_1 is a nonlinear activation function, e.g., hyperbolic tangent or ReLU, and σ_2 is an activation function which may differ from σ_1 and which may be linear. For a given target vector y, the goal is to minimize the deviation of the output vector $x^{(L)}$ from y. This deviation is often measured by a loss function, where for regression problems the square error

$$e = \left\|y - x^{(L)}\right\|_2^2 \quad \text{(Eq 8)}$$

is commonly used. Note that Eq 8 states the error for a single training example, i.e., for one target vector y. For the computation of the mean square error of a training set with N entries, all errors are summed up and divided by N. To minimize the error, suitable weights matrices $W^{(l)}$ and bias vectors $b^{(l)}$, $l = 1, \ldots, L$, have to be determined. This is done by applying an iterative training process using backpropagation as described below.

In practice, the prediction $x^{(L)}$ of an ANN in Eq 7 is computed by forward propagation, which successively predicts the output vector $x^{(l)}$ of each layer $l = 1, \ldots, L$ of the network by:

$$x^{(l)} = \begin{cases} \sigma_1\left(W^{(l)^T}x^{(l-1)} + b^{(l)}\right), & l = 1, \ldots, L-1, \\ \sigma_2\left(W^{(l)^T}x^{(l-1)} + b^{(l)}\right), & l = L. \end{cases} \quad \text{(Eq 9)}$$

ResNets are a particular class of ANNs designed to improve the training of deep networks (Ref 26). The ResNet used in this work is fully connected without skip connections. The only difference in the computation of the ResNet output compared with a standard ANN is the addition of the output $x^{(l-1)}$ of the previous layer to the right-hand side of the forward propagation formula in Eq 9 for $l = 1, \ldots, L-1$. Here, a ResNet is used where the number of neurons per hidden layer is set to be equal to the number of features (six in this setting, see Table 1). This is denoted as simplified ResNet (SimResNet). Its properties have been discussed, for instance, in (Ref 27, 28). For the SimResNet, the prediction or forward propagation formula reads:

$$x^{(l)} = \begin{cases} x^{(l-1)} + \sigma_1\left(W^{(l)^T}x^{(l-1)} + b^{(l)}\right), & l = 1, \ldots, L-1, \\ \sigma_2\left(W^{(l)^T}x^{(l-1)} + b^{(l)}\right), & l = L. \end{cases} \quad \text{(Eq 10)}$$

The forward propagation of Eq 10 is the first step in one iteration of the training algorithm. Subsequently, the weights $w_{ij}^{(l)}$ and the biases $b_i^{(l)}$ are updated for the next iteration by backpropagation, i.e., by adding

$$\Delta w_{ij}^{(l)} = -\eta x_i^{(l-1)}\delta_j^{(l)} \quad \text{and} \quad \Delta b_i^{(l)} = -\eta\delta_i^{(l)} \quad \text{(Eq 11)}$$

respectively, where η is the learning rate,

$$\delta_j^{(L)} = 2\left(x_j^{(L)} - y_j\right)\cdot\sigma_2'\left(\sum_{i=1}^{n_N}\left(w_{ij}^{(L)}x_i^{(L-1)}\right) + b_j^{(L)}\right) \quad \text{(Eq 12)}$$

and

$$\delta_j^{(l)} = \left(\sum_{k=1}^{n_O}\delta_k^{(L)}w_{jk}^{(L)} + \sum_{m=l+1}^{L-1}\sum_{k=1}^{n_N}\delta_k^{(m)}w_{jk}^{(m)}\right)\cdot\sigma_1'\left(\sum_{i=1}^{n_N}\left(w_{ij}^{(l)}x_i^{(l-1)}\right) + b_j^{(l)}\right) \quad \text{(Eq 13)}$$

for $l = 1, \ldots, L-1$. Here, n_O and n_N denote the number of outputs (predictions) and the number of neurons per hidden layer, respectively. The formulas of Eq 11-13 are derived using the optimality condition of the minimization problem of the error given in Eq 8. With the updated weights matrices and bias vectors, the next training iteration starts with the forward propagation of Eq 10.

The iterative process of forward- and backpropagations described above is applied to a set of training data. For each input value of this set, Eq 10–13 are computed

iteratively to update the weights and biases until a mean error regarding the whole training set is sufficiently small.

For this study, a MATLAB code developed at the Institute for Geometry and Applied Mathematics is used for training and testing ResNets for the CCD and LHS data sets. The hyperparameters, which have to be fixed prior to the training, are $\eta = 0.01$ (learning rate), $L = 11$ (ten hidden layers), hyperbolic tangent as activation function σ_1 and the identity function as activation function σ_2. The weights and biases of the ResNet are initialized by Glorot (also called Xavier) initialization (Ref 29). Analogously to the SVM model, the input and target data are standardized for each physical quantity individually by the z-score method. In each iteration of the subsequent training, all input data are forward propagated at once (full batching). The final ResNet outputs for the test data are scaled back to their particular physical range. Two multi-output ResNets are trained: one for the CCD data and one for the LHS data. The structure of the applied ResNets is visualized in Fig. 5, which in addition illustrates the forward propagation procedure of the ResNet (Eq 10) compared to a standard ANN (Eq 9).

Results and Discussion

In this chapter, the results of the ML models are presented and discussed. For each data set produced by different experimental designs, separate prediction models are trained. Then, the target values on the virtual substrate, which are the particle temperatures, velocities and positions (x and z-coordinates) are tested by the corresponding predefined test data sets. Hence, in the following only the results of the test data and not the training data for different ML models and DOE methods are presented and discussed.

Due to the data labeling described in "Machine Learning Algorithms" section, the assignment of the particles to their particular simulation is known. Hence, for a qualitative comparison of ML and simulation results, the average particle behavior per simulation can be investigated. Exemplarily, the mean particle temperatures $\overline{T}_{p,i}$ per simulation $i \in [1, 45]$ are computed by

$$\overline{T}_{p,i} = \frac{1}{n_i} \sum_{j=1}^{n_i} T_{p,i,j} \qquad \text{(Eq 14)}$$

where n_i denotes the number of test particles of simulation i and $T_{p,i,j}$ the particle temperature of particle j of

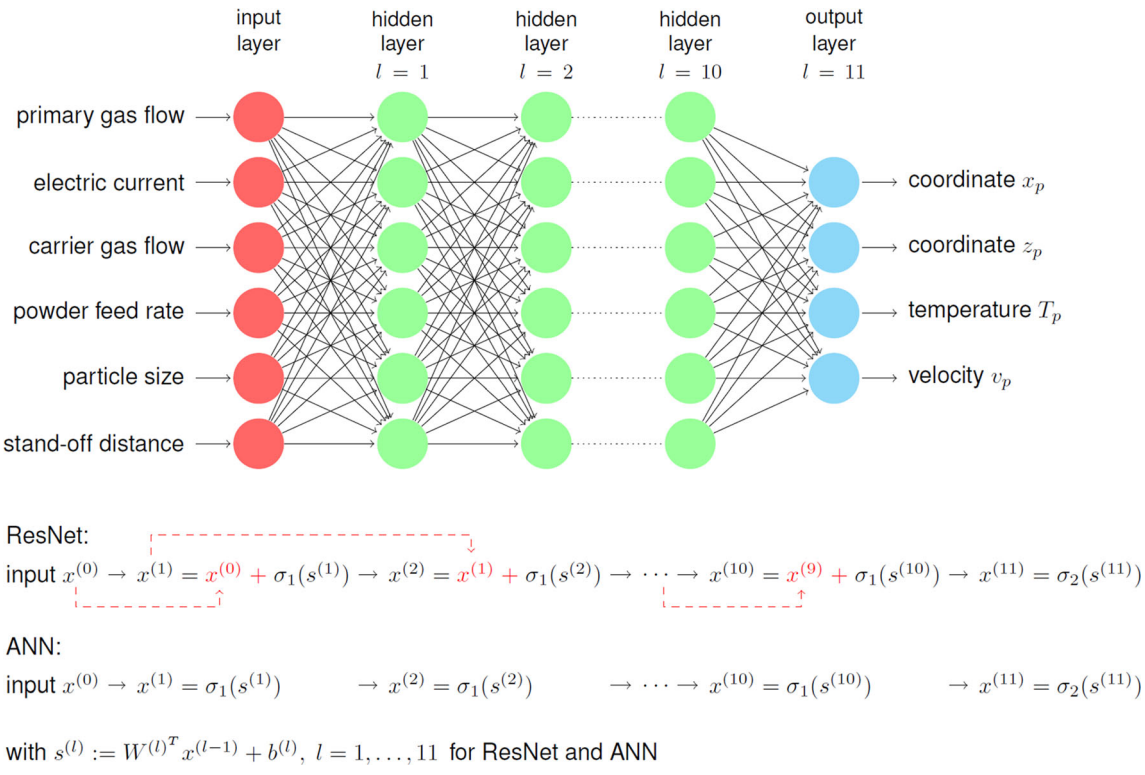

Fig. 5 Structure of the applied ResNet and its forward propagation procedure to compute the output vector $x^{(11)}$ with comparison to a standard ANN

simulation i. The mean value over all 45 simulations is then computed by

$$\overline{T}_p = \frac{1}{45}\sum_{i=1}^{45}\overline{T}_{p,i} \quad \text{(Eq 15)}$$

and denoted by "grandmean" in the following. The means and grandmeans of the particle velocities and positions are computed analogously.

For a quantitative evaluation of the ML results, two statistical measures are considered. To evaluate the prediction accuracy of the single particle properties, the mean absolute percentage error (MAPE) is calculated. Given N data points, the MAPE is defined by

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{t_i - p_i}{t_i}\right| \quad \text{(Eq 16)}$$

with true values t_i and predictions p_i. Furthermore, the R-squared value, for N data points, true values t_i with mean $\bar{t}$ and predictions p_i defined by

$$R_{sq} = 1 - \frac{\sum_{i=1}^{N}(t_i - p_i)^2}{\sum_{i=1}^{N}(t_i - \bar{t})^2} \quad \text{(Eq 17)}$$

is calculated to evaluate the prediction accuracy of the average particle properties.

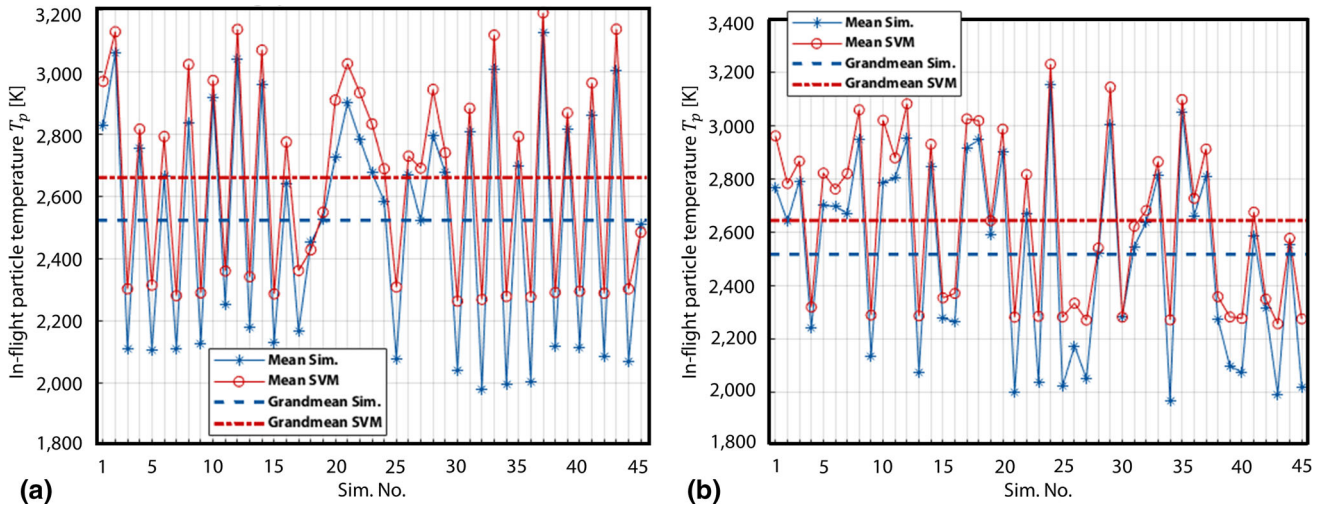

Fig. 6 Results of the mean particle temperatures per simulation for SVM model from (a) CCD and (b) LHS data sets

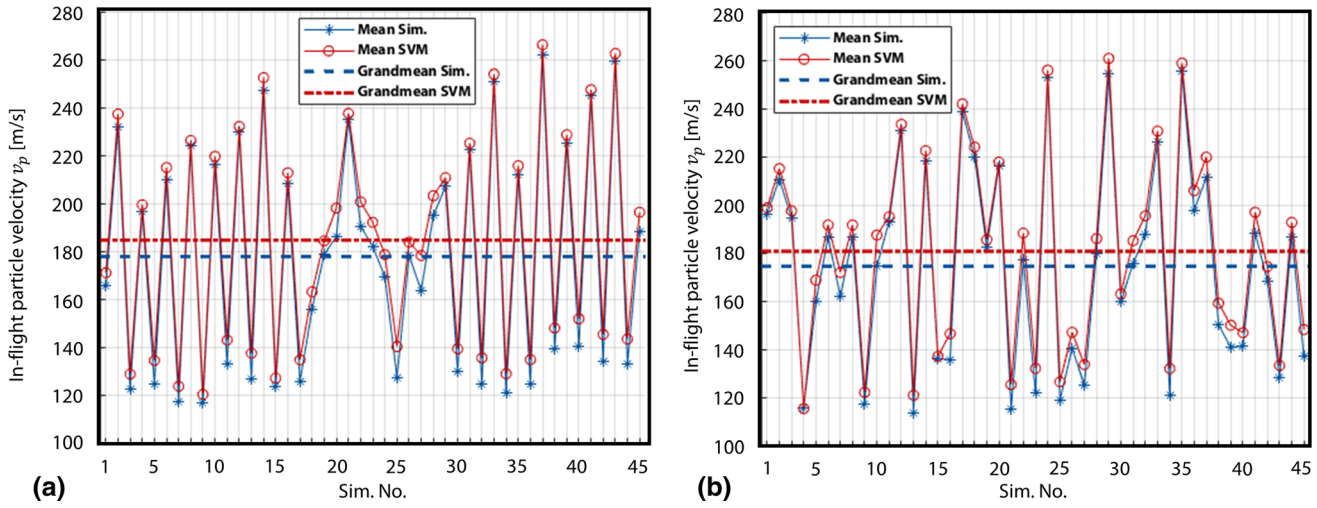

Fig. 7 Results of the mean particle velocities per simulation for SVM model from (a) CCD and (b) LHS data sets

SVM Results

Figure 6 shows the results of the mean particle temperatures $\overline{T}_{p,i}$ per simulation $i \in [1, 45]$, see Eq 14, from the (a) CCD and (b) LHS data sets. The mean values predicted by the SVM model, shown in red, are denoted with "Mean SVM", while the corresponding true values from the simulation model, displayed in blue, are labeled with "Mean Sim.". The grandmeans according to Eq 15 are also plotted in Fig. 6.

In the same way, the results of the mean predicted particle velocities from the (a) CCD and (b) LHS data sets are depicted in Fig. 7. Figures 6 and 7 demonstrate that the developed metamodels have high accuracy in predicting the mean in-flight particle temperatures and velocities from the input process parameters. Furthermore, it is observed that the developed SVM models have slightly better performance in predicting the particle properties with higher temperatures and velocities than the lower ones. In other words, in cases where the particles penetrated deeply into the plasma jet, thus resulting in higher temperatures and velocities, the models could find better relationships between the input process parameters and the particle properties. This has been observed for both CCD and LHS data sets in case of the SVM metamodels.

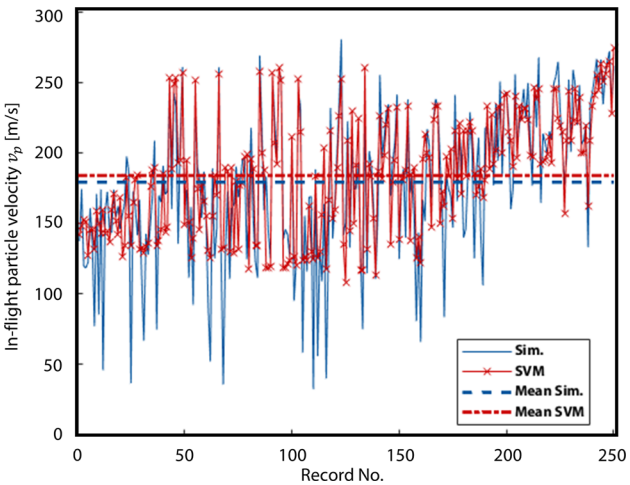

Fig. 8 Exemplary trend of the predicted particle velocities of SVM model from LHS data sets

The predicted and true values of the single particle velocities exemplarily from the LHS data sets are shown in Fig. 8. For a clear presentation, only 250 data points from the total 45 simulations are randomly selected. It is evident that the metamodel can replicate the trend of the particle velocities in the plasma jet. The prediction of the mean particle velocities and temperatures is more accurate than the prediction of the single particle properties. As mentioned earlier, this can be explained with the stochastic nature of the plasma spraying process and the turbulence of the plasma flow, which makes it difficult to predict the behavior of each single particle as it depends on many factors that have influence on each other.

The statistical values MAPE (Eq 16) and R-squared (Eq 17) for prediction of single and average particle properties by SVM model from different DOE methods are given in Table 3. While the performance of the SVM model in terms of prediction accuracy of average particle properties is the same for CCD and LHS data sets, the results of the single particle properties shown in Table 3 indicate a slight improvement in prediction accuracy in case of the LHS experimental design in comparison with the CCD method. This confirms the suitability of the LHS for computational experiments.

Figure 9 shows the distribution of the predicted particle coordinates of the SVM model from LHS data sets exemplarily for one simulation. It is clear that the predictions of the single particle coordinates are much less accurate than the particle velocities and temperatures. As previously mentioned, this is due to the fact that the behavior of single particles is to some extent random in a plasma spraying process, while the essence of ML is to learn and predict regular data. In contrast, the SVM model predicts the mean particle coordinates per simulation more accurately with R-squared values of 0.86 and 0.88 for x and z-coordinates, respectively. The accurate prediction of the mean particle coordinates can be used as a tool to find the position of the maximum particle intensity in the free-jet and consequently e.g. to adjust the injection settings or to position a particle diagnostic device (Ref 30).

The average prediction time of the SVM metamodels for the predefined test data sets was calculated to be about

Table 3 Statistical values for prediction of single and average particle properties by SVM model from different DOE methods

Statistic parameter	MAPE		R-squared	
Property	Single particle temperature	Single particle velocity	Mean particle temperature	Mean particle velocity
CCD	19.78%	22.75%	0.82	0.97
LHS	18.49%	21.11%	0.82	0.97

4.2 s, which is dramatically faster than one CFD simulation of the plasma jet with an average calculation time of 3 hours.

Neural Network Results

The results of the ResNet model for mean particle temperatures from the (a) CCD and (b) LHS data sets are illustrated in Fig. 10. Likewise, the predicted mean particle velocities per simulation from the (a) CCD and (b) LHS data sets are depicted in Fig. 11. It is evident, that the ResNet model has replicated the mean particle temperatures and velocities with high accuracy. Furthermore, it is observed that the ResNet model, in contrast to SVM model, can predict the lower range of the particle properties as good as the upper range. Hence, the model grandmeans show a better agreement with the grandmeans of the simulation than in the SVM case.

The prediction accuracy of the ResNet models in terms of MAPE and R-squared for single and average particle properties from both experimental designs is given in Table 4. In agreement with the SVM results, the ResNet model also shows higher accuracy for the LHS experimental design. Furthermore, the comparison of the model accuracies given in Tables 3 and 4 demonstrates that overall the ResNet model shows an enhancement in accuracy regarding the prediction of single and average particle properties.

Figure 12 illustrates the results of the mean particle x-coordinates per simulation from the (a) CCD and (b) LHS data sets. The ResNet model shows high accuracy in prediction of mean particle coordinates with the R-squared value of 0.99 for predicted x- and z-coordinates.

Figure 13 depicts the training error of the ResNet model for both the CCD and LHS data sets over 1,000 iterations, for each iteration computed by the mean square error

$$e_{tr} = \frac{1}{N}\sum_{i=1}^{N}\left[(t_i - p_i)^2_{T_p} + (t_i - p_i)^2_{v_p} + (t_i - p_i)^2_{x_p} + (t_i - p_i)^2_{z_p}\right], \quad \text{(Eq 18)}$$

cf. Eq 8, where t_i and p_i denote standardized true and predicted values, respectively, N is the number of particles in the training data set and the indices T_p, v_p, x_p, z_p denote for which quantity the particular squared difference is computed. It is evident that the training error out of the LHS data sets is slightly lower compared to the CCD data sets, demonstrating the suitability of LHS for computational experiments.

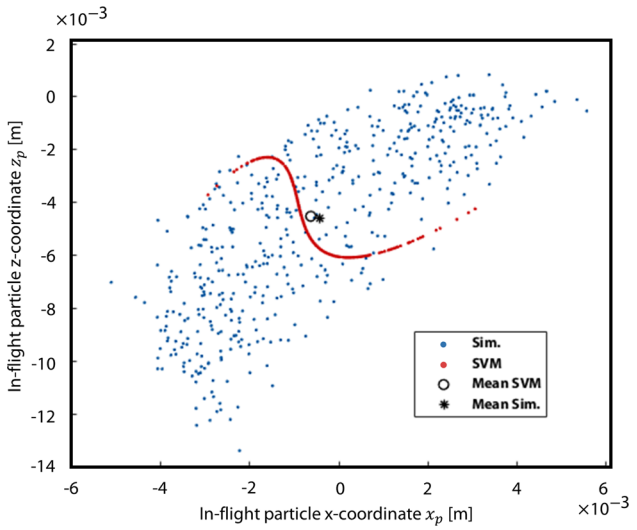

Fig. 9 Exemplary distribution of the particle coordinates of SVM model from LHS data sets for one simulation

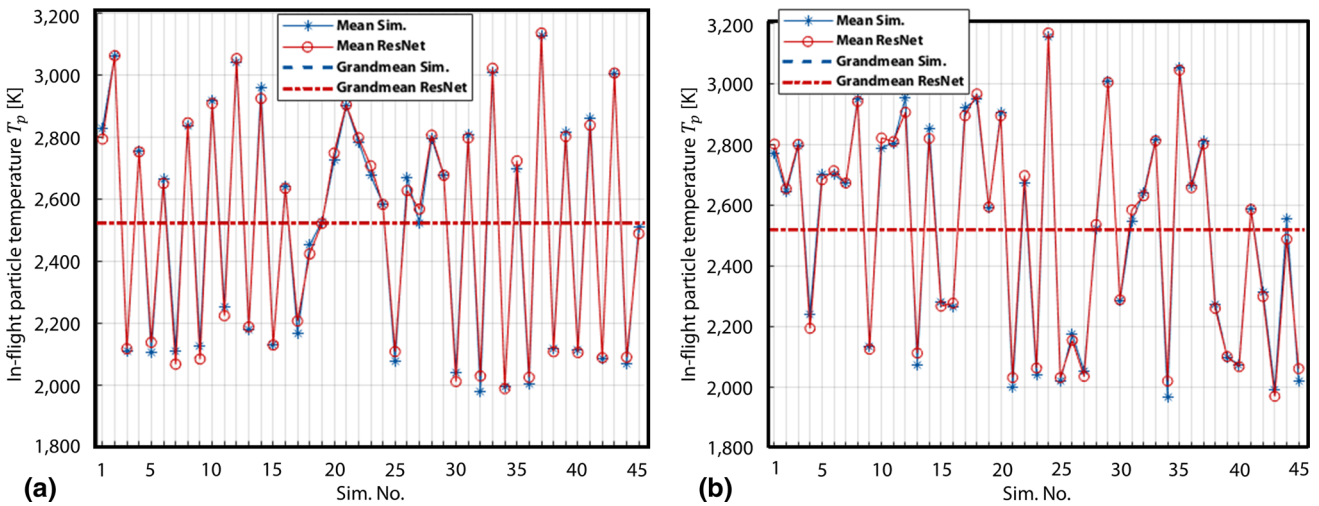

Fig. 10 Results of the mean particle temperatures per simulation for ResNet model from (a) CCD and (b) LHS data sets

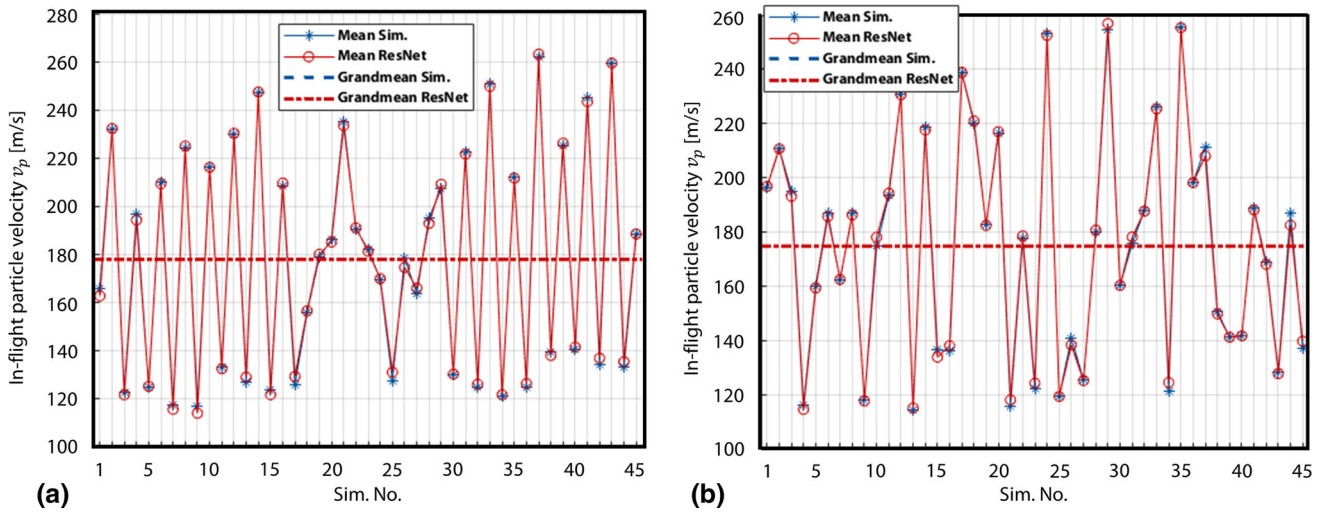

Fig. 11 Results of the mean particle velocities per simulation for ResNet model from (a) CCD and (b) LHS data sets

Table 4 Statistical values for prediction of single and average particle properties by ResNet model from different DOE methods

Statistic parameter	MAPE		R-squared	
Property	Single particle temperature	Single particle velocity	Mean particle temperature	Mean particle velocity
CCD	19.68%	21.88%	0.99	0.99
LHS	18.36%	20.45%	0.99	0.99

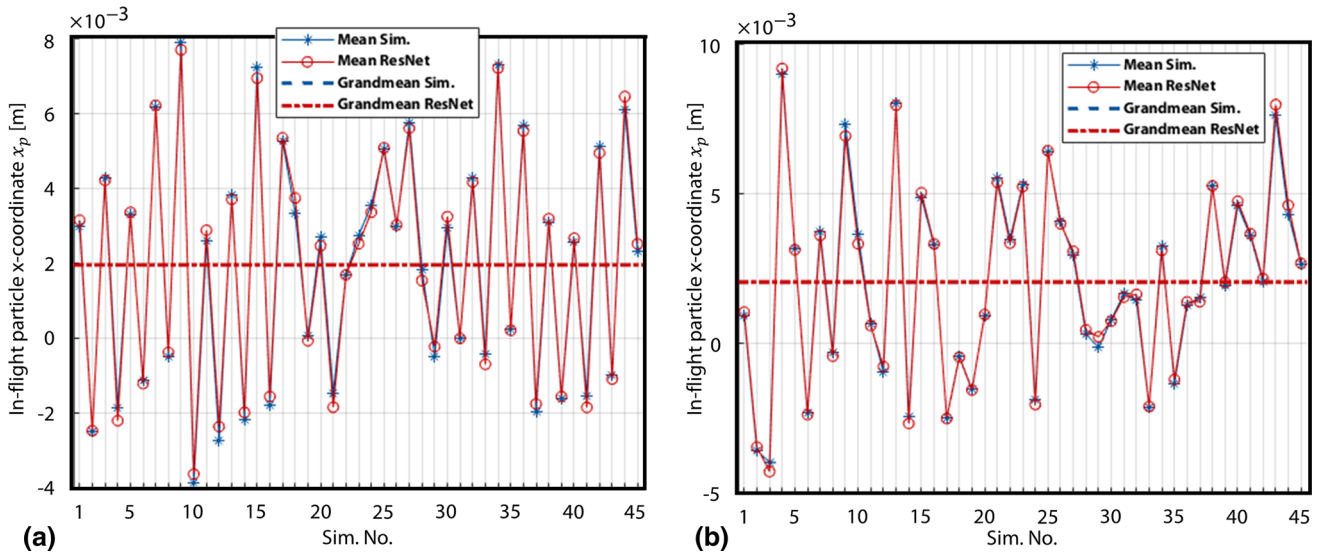

Fig. 12 Results of the mean particle x-coordinates per simulation for ResNet model from (a) CCD and (b) LHS data sets

The computation time for the ResNet prediction, i.e., the forward propagation, for the predefined test data sets is about 0.01 s, which again is a significant decrease compared with the average simulation time of 3 hours.

Conclusions and Outlook

The aim of this study was to take the primary steps towards creating a fast Digital Twin for the plasma spraying process to predict the in-flight particle properties based on input process

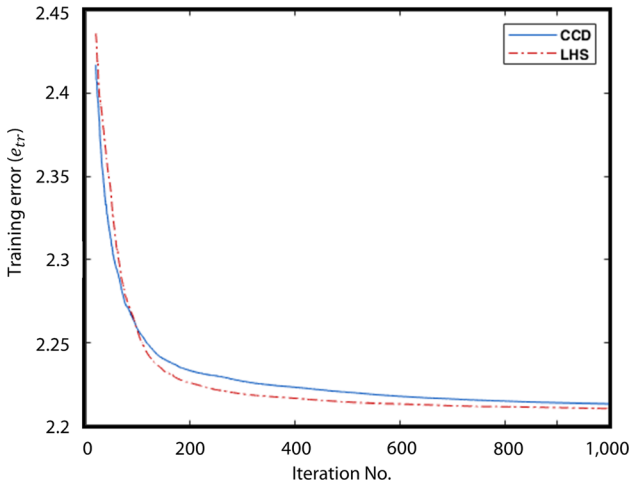

Fig. 13 Iterative error of the ResNet model during the training process

parameters. The data sets for training the ML models have been acquired from a CFD model of the plasma jet. Central Composite Design (CCD) and Latin Hypercube Sampling (LHS) have been employed to cover a set of representative process parameters with reducing the number of tests, while selecting the most valuable sample data. The developed metamodels, namely Residual Neural Network (ResNet) and Support Vector Machine (SVM), are able to replicate the average particle properties with high accuracy, while reducing the computational cost dramatically. The average computational time of one plasma jet simulation is about three hours, while the average prediction time of the metamodels for the predefined test data sets is between 0.01 and 4.2 seconds. The following conclusions can be drawn from the presented results:

- Demonstrating the suitability of the SVM and ResNet metamodels in combination of the CCD and LHS methods for prediction of particle properties in plasma spraying
- Substitution of computationally expensive CFD simulations for ML models with dramatic decrease in calculation time
- Accurate prediction of the mean particle temperatures, velocities and coordinates by SVM and ResNet based on various input process parameters
- Minor increase in prediction accuracy of single particle properties in case of using LHS method for data preparation compared to CCD
- Enhancement in accuracy regarding the prediction of single and average particle properties by ResNet compared to SVM

The results showed that the average particle properties could be predicted by the metamodels much more accurately than the behavior of single particles. This phenomenon is expected, since the plasma spraying is a stochastic process that involves many influencing factors. Thus, the behavior of single particles is much more random in comparison to average particle behavior. The results of the metamodels from the LHS data sets showed a minor enhancement in terms of the prediction accuracy, which confirmed the suitability of space-filling designs for computation experiments.

For a more accurate prediction of the behavior of single particles, the concept of physics-informed neural networks (PINNs) (Ref 31) could be applied. This incorporates the outputs of the ResNet into the system of partial differential equations (PDEs) underlying the simulations. In the spirit of discovering "hidden fluid mechanics" (Ref 32), it could be possible to significantly improve the prediction of single particles even through only a selection of the corresponding PDEs. This would finally lead to a compromise in computational cost between the fast ML predictions in this work and the time-consuming simulations.

Future studies could additionally validate the results of the metamodels by carrying out experimental in-flight particle diagnostics. Moreover, the developed models in the context of this study can provide a good starting point for creating the complementary concept of Digital Shadow for plasma spraying by combining further reduced models and experimental data analytics of the process chain.

Acknowledgments Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612. Simulations were performed with computing resources granted by RWTH Aachen University under project rwth0570.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. J.P. Trelles, C. Chazelas, A. Vardelle and J.V.R. Heberlein, Arc plasma torch modeling, *J. Therm. Spray Tech.*, 2009, **18**(5–6), p 728–752.

2. W. Kritzinger, M. Karner, G. Traar, J. Henjes and W. Sihn, Digital twin in manufacturing: a categorical literature review and classification, *IFAC-PapersOnLine*, 2018, **51**(11), p 1016–1022.
3. K. Rajratna, B. Vikas, J. Santosh, M. Roshan, Digital twin: manufacturing excellence through virtual factory replication. Glob. J. of Eng. Sci. Res. (GJESR) (2018)
4. E. Negri, L. Fumagalli and M. Macchi, A review of the roles of digital twin in CPS-based production systems, *Proc. Manuf.*, 2017, **11**, p 939–948.
5. A.-F. Kanta, G. Montavon, M. Vardelle, M.-P. Planche, C.C. Berndt and C. Coddet, Artificial neural networks vs fuzzy logic: simple tools to predict and control complex processes—application to plasma spray processes, *J. Therm. Spray. Tech.*, 2008, **17**(3), p 365–376.
6. T.A. Choudhury, C.C. Berndt and Z. Man, An extreme learning machine algorithm to predict the in-flight particle characteristics of an atmospheric plasma spray process, *Plasma Chem. Plasma Process.*, 2013, **33**(5), p 993–1023.
7. J. Zhu, X. Wang, L. Kou, L. Zheng and H. Zhang, Prediction of control parameters corresponding to in-flight particles in atmospheric plasma spray employing convolutional neural networks, *Surf. Coat. Technol.*, 2020, **394**, p 125862.
8. M. Öte, "Understanding multi-arc plasma spraying", Dissertation, RWTH Aachen; Shaker Verlag GmbH, Vol. 44, 2016.
9. K. Bobzin and M. Öte, Modeling multi-arc spraying systems, *J. Therm. Spray Tech.*, 2016, **25**(5), p 920–932.
10. K. Bobzin and M. Öte, Modeling plasma-particle interaction in multi-arc plasma spraying, *J. Therm. Spray Tech.*, 2017, **26**(3), p 279–291.
11. K. Bobzin, M. Öte, J. Schein, S. Zimmermann, K. Möhwald and C. Lummer, Modelling the plasma jet in multi-arc plasma spraying, *J. Therm. Spray. Tech.*, 2016, **25**(6), p 1111–1126.
12. K.-T. Fang, R. Li and A. Sudjianto, *Design and Modeling for Computer Experiments*, Chapman and Hall/CRC, Boca Raton, 2006.
13. D.C. Montgomery, *Design and Analysis of Experiments*, John Wiley and Sons, New Jersey, 2017.
14. J.R. Wagner, E.M. Mount and H.F. Giles, *Design of Experiments*, Elsevier, Extrusion, 2014, p 291–308
15. R.H. Myers, *Response Surface Methodology*, Wiley, New Jersey, 2016.
16. M.D. Shields, J. Zhang, The generalization of Latin hypercube sampling, *Reliab. Eng. Syst. Saf.*, 2016, **148**, p 96–108.
17. V.R. Joseph, Y. Hung, Orthogonal-maximin Latin hypercube designs, Stat. Sin. p 171–186 (2008)
18. C. Cortes and V. Vapnik, Support-vector networks, *Mach Learn*, 1995, **20**(3), p 273–297.
19. T. Gurgenc, O. Altay, M. Ulas and C. Ozel, Extreme learning machine and support vector regression wear loss predictions for magnesium alloys coated using various spray coating methods, *J. Appl. Phys.*, 2020, **127**(18), p 185103.
20. J. Xue, M. Huang, Optimization of plasma spray process via orthogonal test design method SVM, and improved PSO, *Int. J. Mater. Mech. Manuf.*, 2017, **5**, p 153–158.
21. M. Awad, R. Khanna, *Support Vector Regression, Efficient Learning Machines*, ed. by M. Awad, R. Khanna, (Apress, New York 2015), p. 67–80
22. B. Schölkopf and A.J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press, Cambrdige, 2009.
23. A.J. Smola and B. Schölkopf, A tutorial on support vector regression, *Stat. Comput.*, 2004, **14**(3), p 199–222.
24. J. Shawe-Taylor and N. Cristianini, *Kernel Methods For Pattern Analysis*, Cambridge University Press, Cambridge, 2004.
25. C.M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, Berlin, 2016.
26. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (Cvpr), 27.06.2016 - 30.06.2016 (Las Vegas, NV, USA), IEEE, p. 770–778 (2016)
27. M. Herty, T. Trimborn, G. Visconti, Kinetic theory for residual neural networks (2020)
28. C. Gebhardt, T. Trimborn, F. Weber, A. Bezold, C. Broeckmann, M. Herty, Simplified ResNet approach for data driven prediction of microstructure-fatigue relationship, *Mech. Mater.*, 2020, **151**, p 103625.
29. X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, p 249–256 (2010)
30. K. Bobzin, W. Wietheger, M.A. Knoch and S.R. Dokhanchi, Estimation of particle mass flow rate in free jet using in-flight particle diagnostics in plasma spraying, *J. Therm. Spray Tech.*, 2020, **29**(5), p 921–931.
31. M. Raissi, P. Perdikaris and G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, 2019, **378**, p 686–707.
32. M. Raissi, A. Yazdani and G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, *Sci. (N. Y.)*, 2020, **367**(6481), p 1026–1030.