Using Data from Physical Experiments to Train Machine Learning Material Models

Daniel Sommer[1], Pauline Böhringer[2], Markus Stoll[3], Peter Middendorf[1]

[1]University of Stuttgart, Institute of Aircraft Design, Germany
[2]Mercedes-Benz AG, Research and Development, Sindelfingen, Germany
[3]Renumics GmbH, Karlsruhe, Germany

Abstract

Structural analysis of mechanical components, such as predicting the deformation behavior of sheet metal or assessing the crash safety of a vehicle, typically relies on finite element analysis (FEA). One critical aspect influencing the quality of these simulations are the material models that describe the relationship between strains and stresses. However, the development and selection of the most appropriate models is a significant challenge that involves costly and time-consuming testing and calibration procedures.

Recently, data-driven material models based on machine learning (ML) methods, such as artificial neural networks (ANNs), have shown the potential to substitute classical analytical models. They promise fast computation, a high level of flexibility and thus the adaptability to new materials [11, 5]. However, these ML material models are usually calibrated to a certain material with training data obtained from simulations and still require the classic analytical models. The stress outputs to given strains are needed in this supervised machine learning task because the stress field cannot be measured from materials testing. The present learning approach merely produces an alternative ML-based representation of the existing classical material models.

To circumvent the need for a classic material model, we present a methodology to train artificial neural networks using exclusively data available from physical experiments [3]. Instead of using a classic material model to predict stresses, we use only physical equations based on the global reaction force and the measurable displacement field on the surface of the test specimen to train the ANN. To verify comprehensive coverage of the required strain input space, we employ data-centric methods. This strategy contributes to improving the robustness and accuracy of our machine learning models, which will lead to a more efficient approach to ML material model development.

1 Introduction

Material models, correlating strain with stress, significantly impact calculation quality and realism of finite element simulations and are essential in structural analysis, encompassing small components to vehicle safety evaluation during development. Balancing accuracy and efficiency adds complexity to selecting suitable models, especially for novel materials requiring new constitutive models. Both cases demand engineers' expertise, as does the subsequent parameter identification for a specific material. This involves material-specific adjustments, achievable only through an extensive, expensive, and often manual testing and calibration process. The goal is to align physical test measurements of deformation and global force-displacement responses with the material models' computations inside Finite Element Analysis (FEA).

Constitutive models within FEA offer a certain level of accuracy, though even advanced ones considering damage and failure are constrained by their analytical nature. Simulation precision depends on proper constitutive formulation, quality of parameter identification, and finally the engineers' expertise. Conversely, computational efficiency is crucial for iterative processes in classical material models like plasticity or damage formulations. Each incremental step of explicit nonlinear FEA is computed with these routines.

In the conventional method for material characterization, material parameters are derived from experimental tests. The goal is to acquire parameters for established material models that accurately replicate material behavior. One method available for this is the finite element model updating

(FEMU), typically relying on mechanical tests captured with full field measurement of the strain fields using digital image correlation (DIC). The strategy involves developing a finite element model of the characterization test itself. Material model parameters are iteratively adjusted within an optimization framework until aiming to determine material properties with maximum accuracy, which is found when the computed strain distribution aligns with the measured one, mirroring the convergence of test force [15, 25, 7]. An alternative technique, known as the virtual fields method (VFM), eliminates the need for FEA simulations to deduce the parameters [9]. Instead, it relies on a limited set of uniaxial tensile tests and heterogeneous strain fields acquired through DIC. The core of the VFM is the minimization of the difference between externally measured virtual work and internally computed virtual work during the optimization process for material model parameters. For this, the material model must be available as a callable subroutine.

The outlined methods all operate on the premise of utilizing a pre-existing material model, which requires the subsequent determination of its parameters. As a result, these approaches are not particularly well-suited for the modelling of new materials unless an appropriate material model specifically tailored for those materials is first developed.

To address these limitations, researchers are increasingly exploring the replacement of traditional analytically derived constitutive models with machine learning material models (MLMM). These models offer the advantage of being trainable without an intricate understanding of existing material theories, as well as the capability to perform efficient calculations without the need for iterative equation solving, along with self-monitoring capabilities.

An early instance of applying MLMM was the successful utilization of artificial neural networks (ANN) within Finite Element Analysis (FEA) to describe concrete by Ghaboussi et al. [11] in 1991. While various publications focus on constitutive modelling for different material types, comprehensive overviews such as the review on deep learning in computational mechanics by Oishi & Yagawa [18] and compilation of manifold attempts and approaches for composite materials by Liu et al. [17] offer in-depth read-ups.

In recent research, Bonatti & Mohr [5] introduced a potent neural network with customized cells designed specifically for constitutive materials modelling. This innovation addresses challenges such as time- or strain-step-dependence and self-consistency. Most of the existing approaches predominantly learn the stress-strain relationship either from FEA simulations or directly from conventional models. They thus have to be categorized as surrogate data-driven models and are not yet detached from traditional material laws, as they still rely on the latter for the strain-to-stress mapping problem. These data-driven models can still provide benefits like faster computation during simulations and simplified parameter identification, requiring less in-depth understanding for practical application compared to the underlying model. However, their full potential can only be fully realized when they can be trained solely on experimental data.

Machine Learning Models in general normally require a labelled set of inputs and outputs to train the model using the backpropagation strategy [20]. When the output is unknown, it is typically impossible to train a model. When decoupling MLMM from classical models in the case at hand, only the input strain field is known, as it is measurable trough DIC from experiments. But the stress field output is not determinable from experiments.

Notable works in data-driven modelling from experimental data include the contributions by Kirchdoerfer and Ortiz [16, 24], who created a material database by capturing physical sample snapshots. Through a minimization problem, they enforced equilibrium constraints, yielding the strain-stress relation for the material. While this model-free approach depends on accurate material databases for different deformation scenarios, its stress prediction quality is linked to the size of the database. Flaschel et al. [10] proposed another method for automated discovery of isotropic hyperelastic constitutive laws, using sparse regression to find analytical expressions from a vast set of candidate models, combined and optimized to fit experimental displacement and force values. Ghaboussi et al. [12] introduced auto-progressive training of constitutive models based on neural networks, while Huang et al. [14] highlighted the significance of training on experimental data, employing neural networks pre-trained on analytical models and closed formulations for plasticity. Detaching MLMM from training data rooted in classical material models remains a major challenge, as emphasized by Liu et al. [17].

In this study, we introduce an extended approach for calibrating neural network-based constitutive models, which establish a relationship between strains and stresses, utilizing only data available in an experiment. The input data includes the strain field obtainable through digital image correlation and the global force measurements. By avoiding the need for Finite Element Analysis during calibration, we formulate loss functions based on physical constraints and global force. These loss functions are subsequently employed to optimize the neural network model's weights and biases. The process

begins with a brief overview of the proposed method, followed by the presentation of the loss formulation and the optimization technique. Finally, we validate the overall method and showcase the MLMM's performance within finite element analysis simulations. This extends our previous publication [3], where this process was shown for linear elastic materials using Feed Forward Neural Networks, to non-linear plastic modelling of materials using Recurrent Neural Networks (RNN).

2 Method

2.1 Small Introduction to Neural Networks and MLMM

Neural networks are computational models inspired by the human brain, composed of interconnected nodes, or neurons, that process and transmit information. Recurrent Neural Networks (RNN) are a type of neural network designed to work with sequences of data, such as time series or natural language. Figure 1 shows a generic neural network with input nodes, a definable number of hidden layers and the output nodes. Shown on the right is one neuron within this network, where the history h depicts the recurrent nature of the neuron allowing information to flow not just from input to output, but also within the network itself within a feedback-loop. This enables them to capture temporal dependencies and context, making them well-suited for tasks like speech recognition or any temporal series. A typical neuron takes the inputs i_n with individual weights w_n and passes them through an activation function θ together with the history h and a bias value b. The weights and biases are the trainable parameters of the network. For the family of recurrent networks, many neurons with more complex internal relaying of inputs, biases, and histories exist, e.g. LSTM or GRU [6].

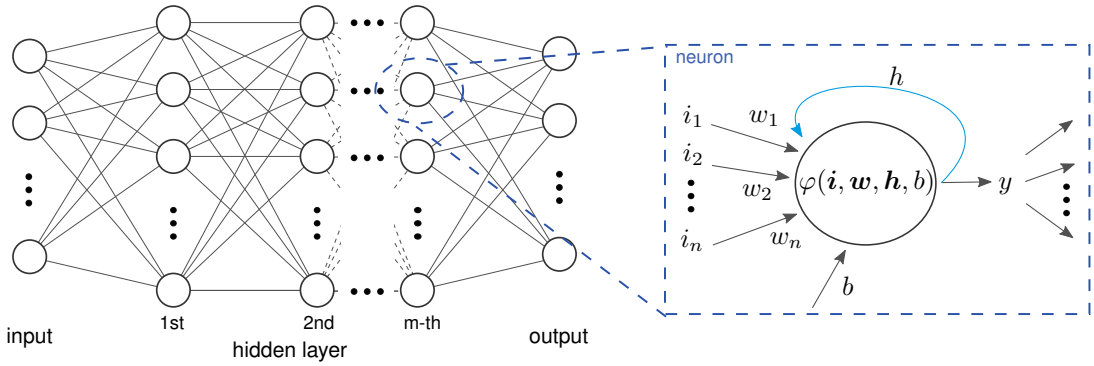

Figure 1: Generic neural network architecture, zoom on a single recurrent neuron. (extended from [3])

Within the scope of materials modelling, the neural network shall predict the stresses for the current explicit time step $\boldsymbol{\sigma}_t$ based on the typical input available to material models, such as the strains $\boldsymbol{\varepsilon}_t$. One could choose to add the last stresses $\boldsymbol{\sigma}_{t-1}$ to the input as well, as some analytical models depend on this, but is not necessary when using a recurrent network architecture as they will carry the history forward internally. Our neural network will thus assume the relationship $f : \varepsilon_{ij} \rightarrow \sigma_{ij}$.

2.2 Outline of the Method of Training on Experimental Data

To be able to train such machine learning material models solely on data discoverable through experiments, a new method is required to calibrate the neural network to represent a new material. As stated before, classical supervised machine-learning techniques, such as backpropagation, are only available when the output of the MLMM can be compared directly to a true target output. For the targeted output of the MLMM, the stresses $\boldsymbol{\sigma}$, the measurement of the stress field in an experiment is not possible.

The raw experimental results available from coupon testing are limited to the time-progression of the global reaction force F trough load-cells and the displacement field $\boldsymbol{u}$ on the surface of the specimen by using digital image correlation (DIC). Using a stereo-camera setup for the DIC, $\boldsymbol{u}$ can be measured in three-dimensions with x, y and z components. From this, the surface strains ε_{xx}, ε_{yy} and ε_{xy} can be calculated. Through thickness components can only be claimed through, e.g. a dual stereo DIC setup measuring from both sides of the specimen. Even then, the progression

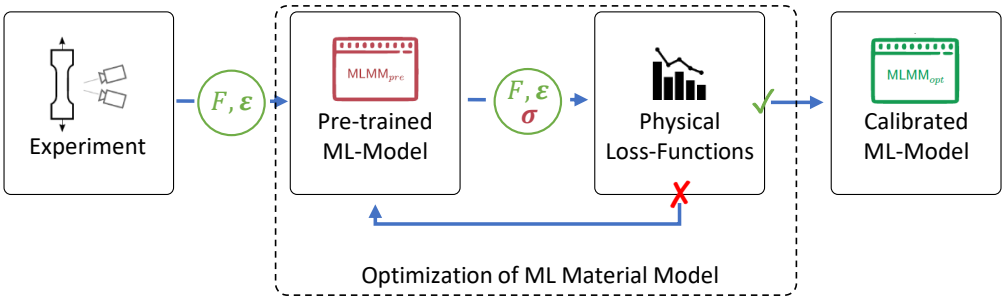

Figure 2: Outline of the Method of Training MLMM on Experimental Data

inside the specimen will be unclear. We limit ourselves to using the measurable 2D-strain state for our models and methods. Figure 2 shows the experiment with outputs F and ε on the left.

The measured strain-field ε is fed directly into a pre-trained machine learning material model denoted MLMM$_{pre}$. It is pre-trained using a classical analytical model, such as `*MAT_24`, for one single material, which is similar to the new, experimental material. This is a staring point for a reasonable assumption on the new model. The MLMM$_{pre}$ will predict the stresses $\boldsymbol{\sigma}$, which will not yet fit the new material, but cannot be compared to any ground truth, since none can be obtained from the experiment.

These values, F and ε from the experiment and $\boldsymbol{\sigma}$ as predicted by the pre-trained MLMM, can now be used to check the admissibility through physical loss-functions. These loss functions will verify global and local equilibria to measure the correct calibration of the MLMM. As long as the prediction of stresses is incorrect, the loss functions will capture the errors. They will be used to update the parameters of the MLMM using our own newly developed optimization strategy. For the first iteration, the starting parameters will be those of the pretrained model.

The stresses will be predicted with the updated parameters of the MLMM using the unchanged measured force and strain field. From hereon, the optimization process is iterated until the equilibrium conditions are met and the loss-functions are minimized. The resulting optimized MLMM$_{opt}$ is then the calibrated machine learning material model representing the material in the experiment.

2.3 Loss-Functions

This section presents the loss functions used to determine whether the MLMM is calibrated onto the new material. It presents a summarized two-dimensional version of the introduction of the loss functions in our peer-reviewed paper [3].

To be considered a valid representation of the material to be modelled, the final MLMM$_{opt}$ within FEA must deliver the same force F and result in the same displacement field u if the specimen is subjected to the same overall displacement as imposed in the laboratory test. To eliminate the degradation of speed in the calibration of the MLMM$_{opt}$, FE simulations that implement the current state of the MLMM are avoided and the following balance and conservation principles are applied to formulate loss functions onto the discrete measurement:

$$\int_V \boldsymbol{\sigma} : d\boldsymbol{\varepsilon}\, dV = \int_S \boldsymbol{t} \cdot d\boldsymbol{u}\, dS \tag{1}$$

$$\operatorname{div} \boldsymbol{\sigma} = \begin{pmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} \\ \frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} \end{pmatrix} = 0 \tag{2}$$

$$\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n} \tag{3}$$

Equation (1) is formulated on the clamped specimen's boundary and considers the equilibrium of the internal work and the work done by the external force, where $\boldsymbol{\sigma}$ and ε is the stress and strain tensor respectively, V the volume of the body under consideration, $\boldsymbol{t}$ and $\boldsymbol{u}$ define the traction and displacement field $\boldsymbol{u}$ at the surface boundary S. Equation (2) defines the second set of losses, ensuring the force equilibrium inside the specimen. And the last set of losses, defined by Equation (3), makes use of especially the zero traction $\boldsymbol{t}$ condition at the free boundary and its geometric normal $\boldsymbol{n}$ of the specimen.

These three relations of the continuum are used to construct three types of loss functions on the discrete measured experimental data trough DIC and the force-transducer. In the following, (m) denotes a measurement stage, that is, strain fields through time of the experiments. (p) are the measured points of the strain-field. (p, m) would then describe a value at point p of stage m. Capital M and P are the total amount of measurement stages and points, respectively.

2.3.1 Loss: Force / Global Energy Balance

To cast the global balance of energy from equation (1) in a loss-function, it is discretized to yield:

$$L_{force} = \frac{1}{M}\sum_{m=1}^{M}\left(\underbrace{\left(\frac{1}{du^{(m)}}\sum_{p=1}^{P} d\boldsymbol{\varepsilon}^{(m,p)^T}\boldsymbol{\sigma}^{(m,p)}V^{(m,p)}\right)}_{F^{(m)}_{pred.}} - F^{(m)}\right) \stackrel{!}{=} 0. \tag{4}$$

Stresses $\boldsymbol{\sigma}^{(p,m)}$ are predicted by the (pre-calibrated) MLMM based on $\boldsymbol{\varepsilon}^{(p,m)}$. $du^{(m)}$ and $F^{(m)}$ are the effective global displacement increment at the force-boundary and measured force at acquisition stage m, respectively. The measured force $F^{(m)}$ is subtracted from the predicted force $F^{(m)}_{pred}$, the latter calculated based on equation (1), to give an error between experiment and MLMM-prediction on the force.

2.3.2 Loss: Divergence

The MLMM might fulfil the energy-balance and match the correct force, but it may do this by having sources and sinks in the energy-field. Thus, we formulate the second loss according to equation (2), based on the first Cauchy-Euler law of motion or momentum balance, neglecting volumetric accelerations and dynamic effects. We thus require the divergence to vanish (i.e. to be free of force sources and sinks) at any point. To be able to evaluate the divergence and thus the different gradients of the stress-components, the calculation is carried out on a structured evenly-spaced point mesh, where the measurement points p are interpolated onto the overlay-grid points $\hat{p}$ using Clough-Tocher interpolation [1]. This produces a C1-smooth, piecewise cubic interpolating surface which minimizes the gradients. This mapping strategy allows the calculation of the divergence field on a structured mesh without the necessity of introducing shape functions. The resulting losses for checking the absence of divergence are the two following, reflecting the components of equation (2).

$$L_{div,x} = \frac{1}{M\hat{P}}\sum_{m=1}^{M}\sum_{\hat{p}=1}^{\hat{P}}\left|\frac{\partial\sigma_{xx}}{\partial x}^{(\hat{p},m)} + \frac{\partial\sigma_{xy}}{\partial y}^{(\hat{p},m)}\right| \stackrel{!}{=} 0 \tag{5}$$

$$L_{div,y} = \frac{1}{M\hat{P}}\sum_{m=1}^{M}\sum_{\hat{p}=1}^{\hat{P}}\left|\frac{\partial\sigma_{yx}}{\partial x}^{(\hat{p},m)} + \frac{\partial\sigma_{yy}}{\partial y}^{(\hat{p},m)}\right| \stackrel{!}{=} 0, \tag{6}$$

where $\hat{P}$ is the total number of grid points on the structured overlay grid and the gradients $\partial\sigma_{ij}/\partial x_k$ are evaluated as the second order accurate central differences in the respective directions, i.e.,

$$\frac{\partial\sigma_{ij}}{\partial x_k}^{(\hat{p})} = \frac{\sigma_{ij}^{(\hat{p}_{i+1})} - \sigma_{ij}^{(\hat{p}_{i-1})}}{2d_k}. \tag{7}$$

2.3.3 Loss: Force-Free Boundaries

The last demand on the prediction of the MLMM is formulated by the traction-free boundaries where no force is applied, derived from equation (3):

$$L_{bnd,free} = \frac{1}{MP}\sum_{m=1}^{M}\sum_{p=1}^{P_{FR}}\sqrt{{t_x^{(p)}}^2 + {t_y^{(p)}}^2}, \tag{8}$$

with $\boldsymbol{t} = \boldsymbol{\sigma}\cdot\boldsymbol{n}$, where $\boldsymbol{\sigma}$ is known from the prediction of the MLMM and $\boldsymbol{n}$ is the unit normal vector on the boundary. P_{FR} is the number of measurement points composing the traction-free boundary.

2.4 Optimization Strategy

The various presented equations, which are intended to apply to the predicted stresses $\boldsymbol{\sigma}(\boldsymbol{x})$ of a trained MLMM with given trainable parameters $\boldsymbol{x} \in \mathbb{R}^n$, can be viewed as the loss functions $L_{force}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{bnd}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{div,x}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{div,y}(\boldsymbol{\sigma}(\boldsymbol{x}))$ for training through a deviation from equality as described above. Since we are considering several objective functions, the training based on experimental data can be presented as a multi-objective optimization problem of the following form

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = (L_{force}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{bnd}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{div,x}(\boldsymbol{\sigma}(\boldsymbol{x})), L_{div,y}(\boldsymbol{\sigma}(\boldsymbol{x}))) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{R}^n, \end{aligned} \tag{9}$$

with n equal to the number of trainable parameters (i.e. weights and bias values) of the MLMM and a feasible solution $\boldsymbol{x}$ corresponding to a possible allocation of the weights and bias values of the network. While in the single-objective case it is clearly defined which solution in an optimization problem is better or worse (i.e. larger or smaller objective function value depending on whether the objective function is to be minimized or maximized), in multi-objective optimization Pareto optimal solutions are considered. A feasible solution $\hat{\boldsymbol{x}} \in X$ to a multi-objective optimization problem of the form

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} \in X, \end{aligned} \tag{10}$$

with objective function $f : X \subset \mathbb{R}^n \to \mathbb{R}^k$ is called Pareto optimal if there does not exist another feasible solution $\boldsymbol{x} \in X \setminus \{\hat{\boldsymbol{x}}\}$ with $f_i(\boldsymbol{x}) \leq f_i(\hat{\boldsymbol{x}})$ for $i = 1, ..., k$. We say in this context that $\boldsymbol{x}$ dominates $\hat{\boldsymbol{x}}$. In summary, we look for solutions that cannot improve in one component of the objective function without worsening another.

Multi-objective optimization problems can be transformed into a single-objective optimization problem using different scalarization methods. A popular method is the weighted sum scalarization

$$\begin{aligned} \min \quad & L_{total}(\boldsymbol{x}) \coloneqq w_{force} L_{force}(\boldsymbol{\sigma}(\boldsymbol{x})) + w_{div,x} L_{div,x}(\boldsymbol{\sigma}(\boldsymbol{x})) + w_{div,y} L_{div,y}(\boldsymbol{\sigma}(\boldsymbol{x})) + w_{bnd} L_{bnd}(\boldsymbol{\sigma}(\boldsymbol{x})) \\ \text{s.t.} \quad & \boldsymbol{x} \in \mathbb{R}^n, \end{aligned} \tag{11}$$

with predefined weights $w_{force}, w_{bnd}, w_{div,x}$ and $w_{div,y}$. A major difficulty lies in finding an optimal set of weights, considering that the solution obtained strongly depends on it. We consider in the following both optimization variants for the evaluation of the method and show that a good model can be obtained with a suitable weighting as well as a multi-objective optimization strategy without specifying a suitable weighting beforehand. For both optimization variants, evolutionary algorithms are considered. Evolutionary algorithms (see e.g. Rechenberg [19] Schwefel [21] Hansen et al. [13]) are iterative randomized optimization algorithms that are based on the idea of biological evolution principles. They use in each iteration a set of possible solutions (population) which are to be improved by generating stochastic variations of the considered solutions (mutation) and selecting the best candidate solutions out of the population (selection). For the single-objective optimization, the *pseudogradient-ES* algorithm already applied for the training of a linear elastic MLMM on experimental data and described in detail in Böhringer et al. [3] is used. For the multi-objective optimization, we use the *NSGA-III* Algorithm proposed by Deb & Jain [8] and the framework provided by Blank & Deb [2].

2.5 Pre-Training of MLMM and Implementation into LS-DYNA

To create the pre-trained model needed for the start of the optimization, recurrent neural networks are chosen to capture the path-dependent behavior of materials exhibiting plastic behavior. Preliminary studies [22] using state-of-the art RNN Cells, e.g. Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs), showed that they can capture the training strain paths with low error, but fail to run in explicit simulations due to small time stepping, as they are dependent on the sampling of the strain paths. Linearized Minimal State Cells (LMSC) [4] are used, which do overcome this effect by modifying the GRU to be self-consistent. Meaning, a zero-increment in strain will not affect stress-output and the outcome of a strain-increment will stay the same, no matter the steps it will be applied in. Additionally, it embeds a concept of minimal states, minimizing the history-states to be passed onto the next time step.

A two-dimensional mapping is chosen reading $f : (\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy})^T \rightarrow (\sigma_{xx}, \sigma_{yy}, \sigma_{xy})^T$, to reflect the two-dimensional nature of the DIC measurement of the strain field, as discussed before. Normalization of in- or output is not undertaken, and the neural network is set up with a depth of 4 layers, a width of 35 and the LMSC consisting of 6 units, resulting in 8710 trainable parameters in the network.

To generate the stress-strain pairs for pretraining, we employ an elasto-plastic model with J2-plasticity (similar to `*MAT_24`) implemented in Python. This facilitates fast generation of stress outputs and provides unbiased material model output rather than generating the training data from FE-Simulations with added noise from e.g. hourglassing, dynamic effects etc. For pretraining, we set the material model's parameters to represent a generic DP600 cold-rolled steel. Strain paths are created using two different strategies: a spline-based generator and a segment-based generator. Spline paths are created using a definable number of control points and a limit on minimum and maximum total strain. Segmented paths are defined by the length of the strain path, the number of segments as well as a deviation angle between the segments. With each generator, 1665 individual paths are generated three times with samplings of 300, 600 or 1000 points. Every strain path is rotated $20\times$ in 2D-space, resulting in 199800 paths in the dataset.

As the pretrained model does not have to be very refined, as it shall only provide a starting point, the pretrained model did not undergo extensive hyperparameter optimization and is only trained for 23 epochs with a batch size of 64 and a learning-rate of 0.03 with a 0.5 power-law decay. A 10% split is taken aside for validation. After 23 epochs, a mean squared error (MSE) of $3.49 \cdot 10^{-5}$ on the training set and $3.56 \cdot 10^{-5}$ on the validation set is reached. Loss-progression and worst prognosis for a path in the validation set is shown in figure 3. Noticeably, in some sections the MLMM is off from the `*MAT_24` groundtruth, but recovers after and follows the correct procession. This model shall suffice as a starting point, even if training for more epochs will lower the MSE further.

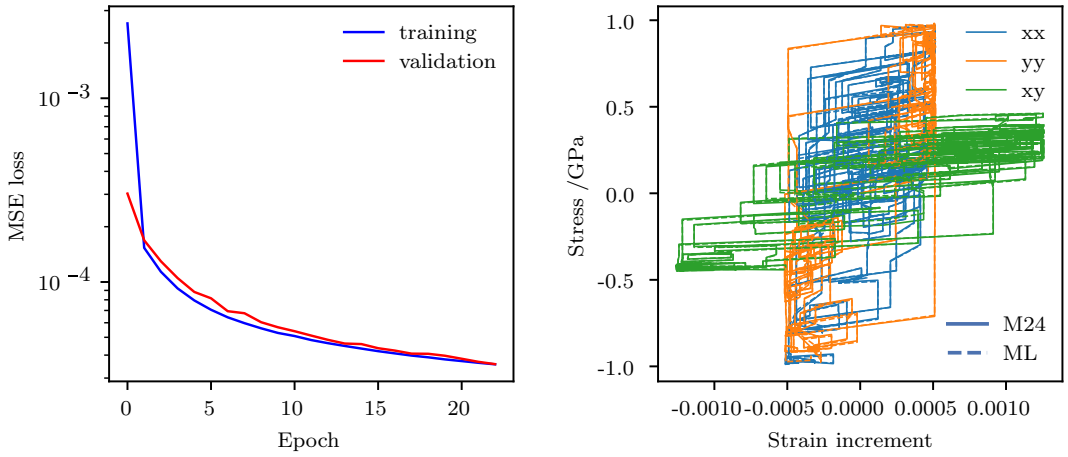

Figure 3: Result of MLMM pretraining for DP600: Left: mean squared error development during training over 25 epochs; Right: Prognosis of ML-Model on a validation path.

To evaluate the pre-trained model and MLMM in general, we build upon the interprocess communication user material for artificial intelligence as introduced in Sprave et al. [23]. For this, we created an advanced Python listener which takes over the handling of the recurrent network for stress prognosis. This script will load the neural network, handle communication, feed the correct inputs to the ML-Model and output to LS-DYNA dynamically, depending on network architecture. Additionally, for RNN, every integration-point (IP) will have a distinct history in FEA, meaning the state of the network at each IP must be loaded and stored before and after prediction. This implementation will work for solids and shells. For the latter, it also adds hybridisation capabilities, which are important for our case using only in-plane components in the MLMM. Out-of-plane shear stresses σ_{xz} and σ_{yz} for shells are calculated similarly as in `*MAT_24` with the `_2D` option. An estimate on the thickness strain ε_{zz} can also be added to the MLMM's prognosis, if not already included in it. This hybridization necessitates the definition of the shear modulus, which is determinable easily from testing. Figure 4 shows the pretrained MLMM for DP600 in an explicit LS-DYNA simulation on a generic hat-profile in three point bending mode and highlights the importance of this hybridization method, as without it, no through-thickness shear components would be present, and the results will not be correct.

With the pretrained DP600 model created and validated, a starting point for the calibration with optimization onto a new material is ready.

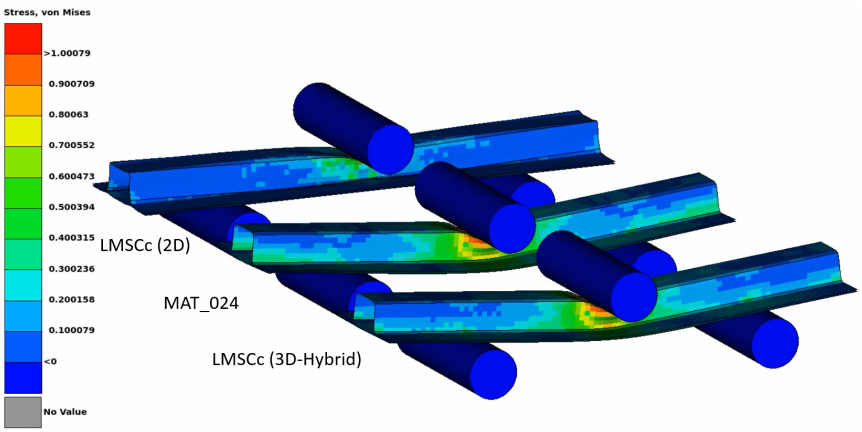

Figure 4: Validation of the MLMM pretrained for DP600 on a generic hat-profile in three point bending. Middle: `*MAT_24` reference; Top: LMSC without hybridization (2D components only); Bottom: LMSC with hybridization active for a hybrid 3D plane stress state.

2.6 Analysis of Training and Target Data

The creation of 199,800 stress-strain paths for pre-training, which was accomplished using an elasto-plastic model with J2-plasticity as detailed in section 2.5. It involves random strategies that provide a wide variety of scenarios. This ensures comprehensive coverage of the input space, which is only possible during the pre-training phase of our material model, where both the strain and stress values are utilized for training.

For the training on experimental data, as discussed in the sections on a tensile test patch (3.1) and a small tensile test (3.2), we will solely utilize strains and forces that can also be measured in experiments.

We employ histograms to analyze the strain components $\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}$, providing a granular view of the distribution of various strain values in our dataset. This is shown in figure 5. Observably, every range of the simulation data is covered by training data, indicating a well-represented training dataset.

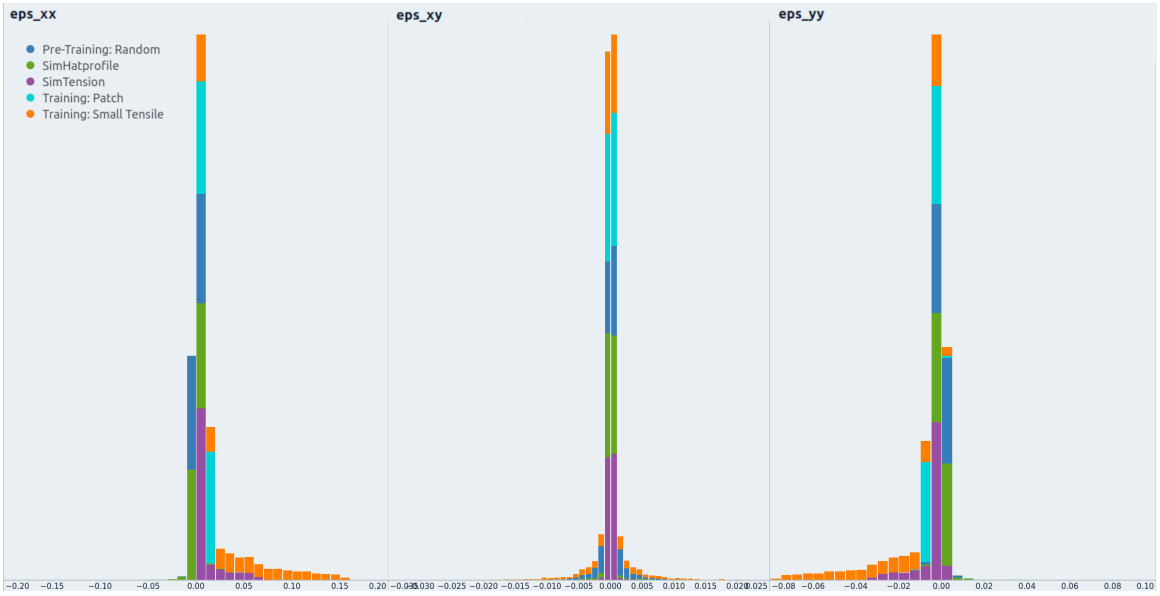

Figure 5: This histogram portrays the distribution of values for each strain component, $\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}$. To read the histogram, one should observe both the height of each bar, which indicates the frequency of data points in that range, and the width of the bars, which indicates the range of values contained within that bin.

Additionally, we compare the generated data with data derived from well-established simulations, specifically a tension simulation (fig. 10) and a hat profile simulation (fig. 4). These simulations act as benchmarks to verify the sufficiency of our input space coverage. We apply the Uniform Manifold Approximation and Projection (UMAP) technique to the strain components $\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}$ and stress components $\sigma_{xx}, \sigma_{yy}, \sigma_{xy}$ randomly sampled from these paths. UMAP is a dimensionality reduction technique that operates by constructing a high-dimensional graph representation of the data, then optimally embedding this graph into a lower-dimensional space, preserving as much of the significant structure of the data as possible. This process involves finding the nearest neighbors of each point, constructing a weighted graph, and then embedding this graph into a lower-dimensional space by minimizing the cross-entropy between the two representations. Through this, we derive a visual and low-dimensional representation, shown in figure 6. A majority of the established simulation data has proximate training data. All analyses were conducted using the open-source tool, Renumics Spotlight (github.com/renumics/spotlight).

Figure 6: The UMAP plot shows the similarity of randomly sampled strain and stress components. Larger dots represent (pre-)training data, while smaller dots indicate data from established simulations (hatprofile and tension test).

3 Validation

We present a validation of the introduced method based on a virtual tensile test while only using the global force F and the strain field, which is also available in the corresponding experiment. For the simulation, we use the explicit FEA solver LS-DYNA to model first a rectangular tensile specimen (patch) under tensile load, as obtained by omitting the outer edge containing the restraint in a tensile test specimen (see figure 7 top) and later extend the sample at the edge to capture the full range of a DIC measurement (see figure 7 bottom), thereby providing a more diverse dataset for training.

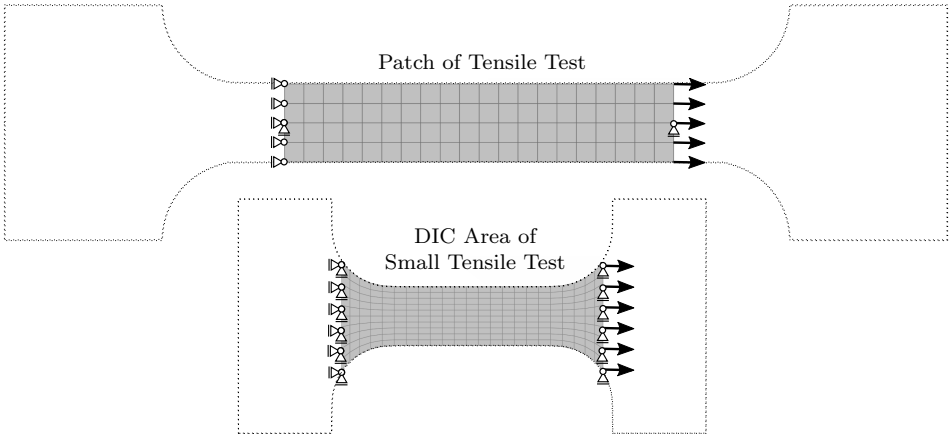

Figure 7: Virtual specimen used for optimization. Top: Patch modelled as middle section of large tensile test; Bottom: Small tensile test with area viewable by DIC system. Specimen not to scale.

3.1 Optimization on Patch of Tensile Test

The LMSC-Model, which was pretrained for a DP600 steel as described in section 2.5 and which we refer to as $MLMM_{DP600}$ is adapted for a DP800 steel using the loss functions introduced in section 2.3. While the architecture of the MLMM stays the same, the trainable parameters of the network (i.e the weights and bias values) are adjusted such that the loss function is minimized. We

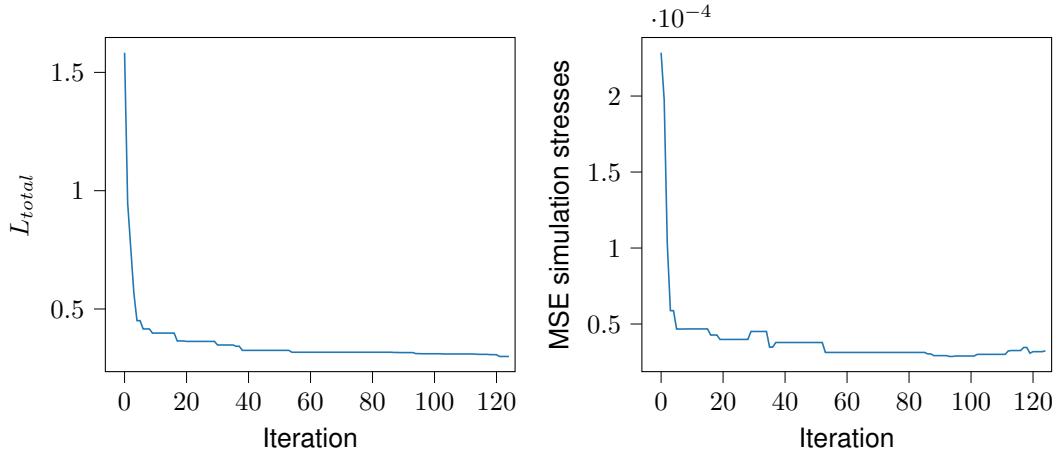

Figure 8: Left: Progress of the loss value L_{total} during single-objective optimization, Right: Development of the normalized MSE over iterations for random paths and paths seen in the simulation.

start with an optimization using the weighted sum scalarization in equation (11) with a weighting of $(w_{force}, w_{bnd}, w_{div,x}, w_{div,y}) = (10, 1, 1, 1)$ and the optimization algorithm *pseudogradient-ES*. Figure 8 shows the progress of the optimization loss L_{total} of the best found model in each iteration, where in every iteration 20 solutions (i.e. models) are being considered. After 125 iterations, a MLMM which we refer to as $MLMM_{opt,single}$ is found with a loss value of 0.2993. To compare the training strategy to the conventional training of ML-models, we look at a MLMM of the same architecture trained directly with strain-stress data from the classical formulation of the material model (as described

in 2.5) for the DP800 and call this model $MLMM_{DP800}$. This directly trained model $MLMM_{DP800}$ has given the same weights a total loss value of 0.3053. To measure how well the optimized model can predict the stresses corresponding to the strains seen in the virtual experiment, we compute the mean squared error

$$\text{MSE}(\sigma, \hat{\sigma}) = \frac{1}{6 \cdot M \cdot P} \sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{i \in \{xx,yx,yy\}} (\sigma_i^{(p,m)} - \hat{\sigma}_i^{(p,m)})^2 \tag{12}$$

between the correct stresses $\sigma_i^{(p,m)}$ and the stresses $\hat{\sigma}_i^{(p,m)}$ predicted by the optimized MLMM. This is done to validate how well the alternative proposed loss function serves as a substitute to the commonly used mean squared error loss function based on the correct stresses. During optimization, the MSE error decreases from 0.000229 to $2.901 \cdot 10^{-5}$ which is even slightly better than the MSE of $3.04 \cdot 10^{-5}$ of the directly trained MLMM $MLMM_{DP800}$. That shows that the model learns to predict the stresses seen in the virtual tension test.

For the multi-objective optimization, we consider the same virtual experiment starting with the same pretrained model $MLMM_{DP600}$ and run the *NSGA-III* algorithm for 300 iterations while considering 10 solutions in each iteration. In the last population, 8 final solutions which were not eliminated due to other found dominating solutions remain and are evaluated with the MSE of the simulation stresses given in the virtual experiment (like in the single-objective optimization scenario). Figure 9 shows the different loss function values considered in the multi-objective optimization of the final eight solutions (blue points), the pretrained model $MLMM_{DP600}$ (green point), the directly trained model $MLMM_{DP800}$ (orange point) and the loss values given the stresses in the simulation provided by the classic material model (red point). The best found model (i.e. the model with the smallest MSE) which we refer to as $MLMM_{opt,multi}$ (black point in figure 9) has a MSE of $4.15 \cdot 10^{-5}$ which is similar to the directly pretrained model $MLMM_{DP800}$ and the model $MLMM_{opt,single}$ found in the single-objective optimization. Table 1 gives an overview of the different loss function values, as

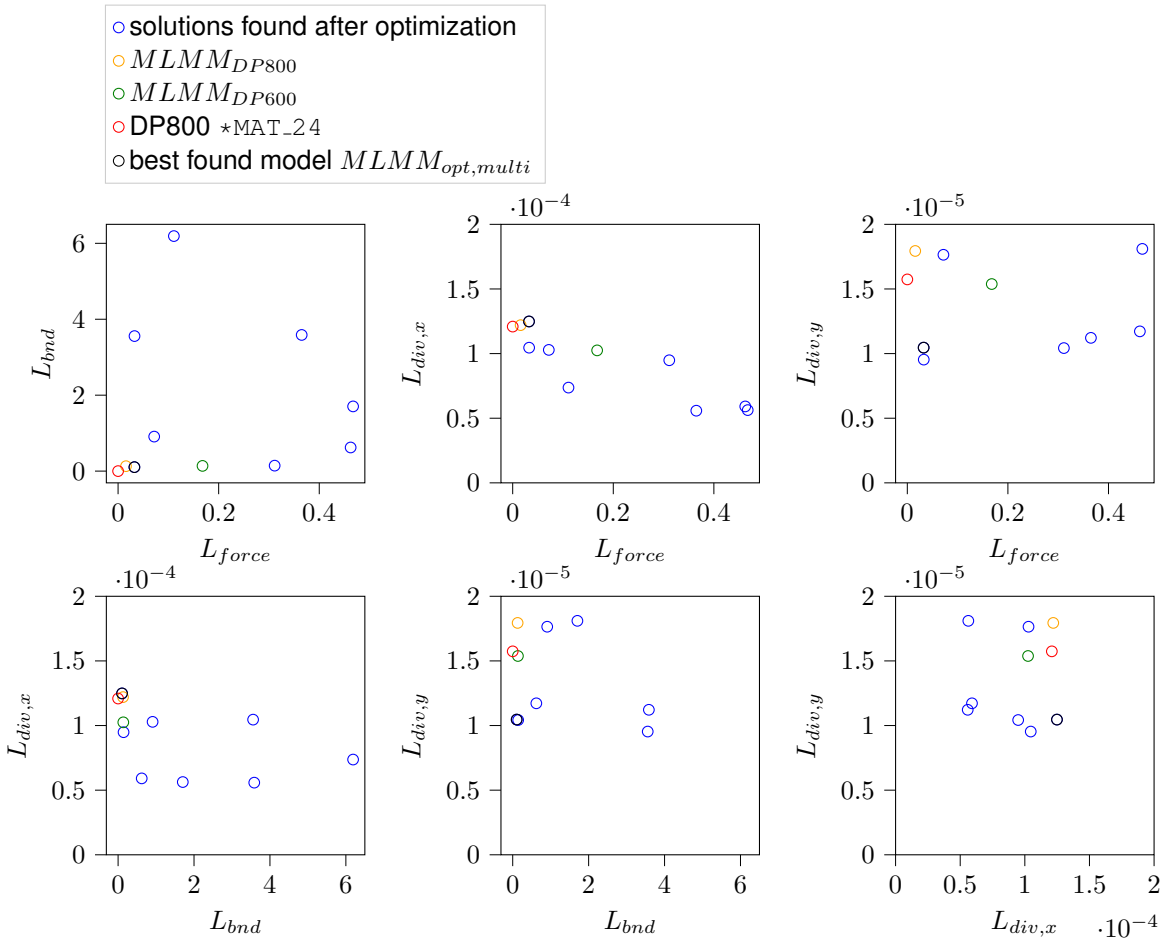

Figure 9: Results of the multi-objective optimization process

well as the MSE of the simulation stresses of the different trained models and the pretrained model $MLMM_{DP600}$.

	L_{force}	L_{bnd}	$L_{div,x}$	$L_{div,y}$	MSE simulation stresses
$MLMM_{DP800}$	0.0157	0.1295	0.000121	$1.79 \cdot 10^{-5}$	$3.04 \cdot 10^{-5}$
$MLMM_{DP600}$	0.1679	0.1372	0.000102	$1.54 \cdot 10^{-5}$	0.00125
$MLMM_{opt,single}$	0.0296	0.0454	0.000122	$1.59 \cdot 10^{-5}$	$2.90 \cdot 10^{-5}$
$MLMM_{opt,multi}$	0.0324	0.1043	0.000124	$1.05 \cdot 10^{-5}$	$4.15 \cdot 10^{-5}$

Table 1: Overview of the different components of the optimization loss and the MSE of the simulation stresses for the different models: the pretrained model $MLMM_{DP600}$ at the beginning of the optimization, the directly trained reference model $MLMM_{DP800}$ and the optimized models $MLMM_{opt,single}$ and $MLMM_{opt,multi}$ obtained through the single and multi-objective optimization strategy.

We finally validate both trained models $MLMM_{opt,single}$ and $MLMM_{opt,multi}$ in a tension test simulation as well as the hatprofile simulation like it was done with the pretrained model trained on random strain-stress paths. The total force in the tension simulation as shown in figure 10 is fitted

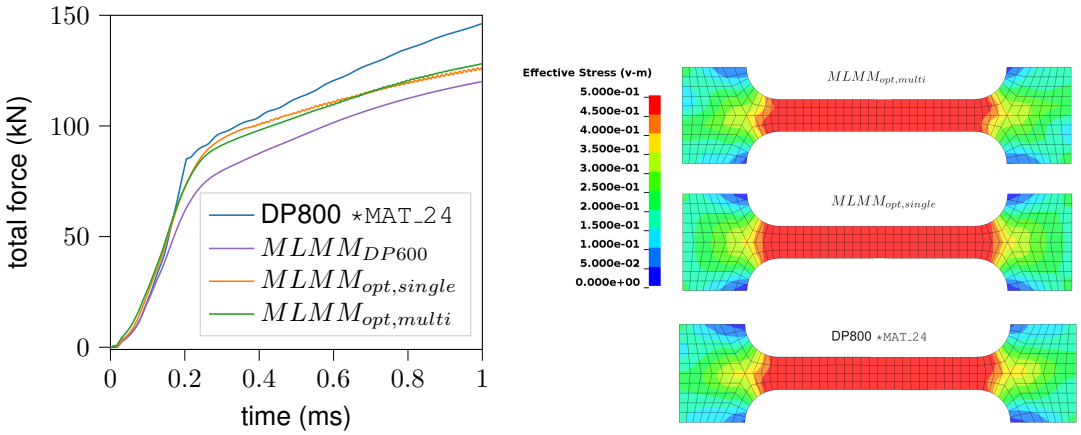

Figure 10: Validation of the optimized models in a tension simulation

better by the optimized model than the starting model $MLMM_{DP600}$ but shows a growing deviation at higher strains which can be explained by the lower strain sizes in the patch simulation.

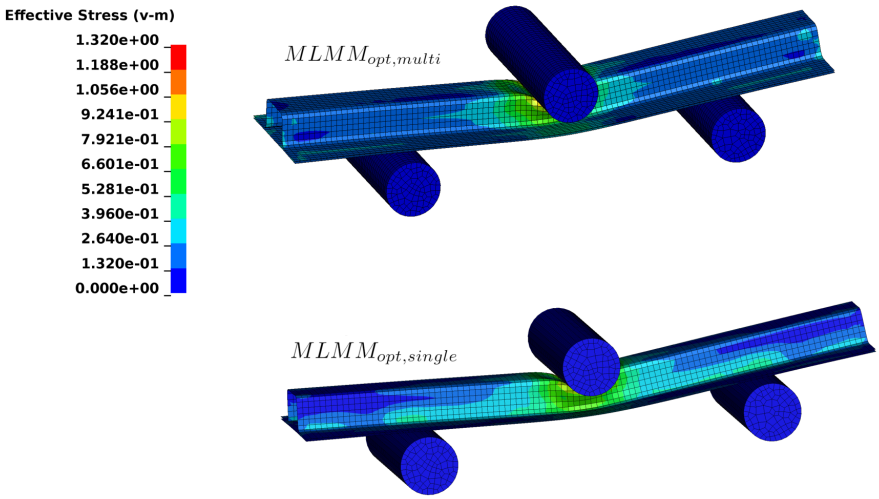

Figure 11: Validation of the optimized models in a hatprofile simulation

3.2 Optimization on Small Tensile Test

As a second validation of the re-training of the MLMM onto another material, the small tensile test (STT) is considered for in the optimization. The test is re-modelled in FEA to represent the DIC-viewable area of the specimen, as shown in figure 7, bottom. Using the single-objective optimization using *pseudogradient-ES* with a scalar loss with weights $(10, 1, 1, 1) = (w_{force}, w_{bnd}, w_{div,x}, w_{div,y})$, the starting point is again the pre-trained MLMM on DP600. Results for the optimization after 200 iterations are shown in table 2. All losses and the MSE on simulation stresses reaches values as in the previous runs on the patch. Strikingly, the divergence losses are much higher when using the STT compared to the patch (0.1241 vs. 0.000122). As observed before, the patch was homogeneous in every element, and thus gradient free on the strains, and effectively has no divergence in the strain field. This will result in a divergence-free stress field. This can also be seen when analysing figure 9 for the patch, the divergence fluctuates with the weights, but stays in the low 10^{-5}. In the case of the STT, the strain field is heterogeneous, gradients and thus divergences can exist. This is one of the reasons, why the resulting MLMM validated within FEA on the large tensile test in figure 12 correlates better on the stress field and does not have pronounced asymmetries. The better fit of the force at higher strains (time ≥ 0.4 ms) for the STT (fig.12) compared to the patch (fig.10) can be attributed to higher strains present in the STT. This was already anticipated in the data-analysis in section 2.6 with 5.

	L_{force}	L_{bnd}	$L_{div,x}$	$L_{div,y}$	MSE simulation stresses
$MLMM_{opt,single,STT}$	0.0056	0.1591	0.1241	0.1179	$4.06 \cdot 10^{-5}$

Table 2: Results of single-objective optimization of the pretrained DP600 MLMM onto the small tensile test from DP800.

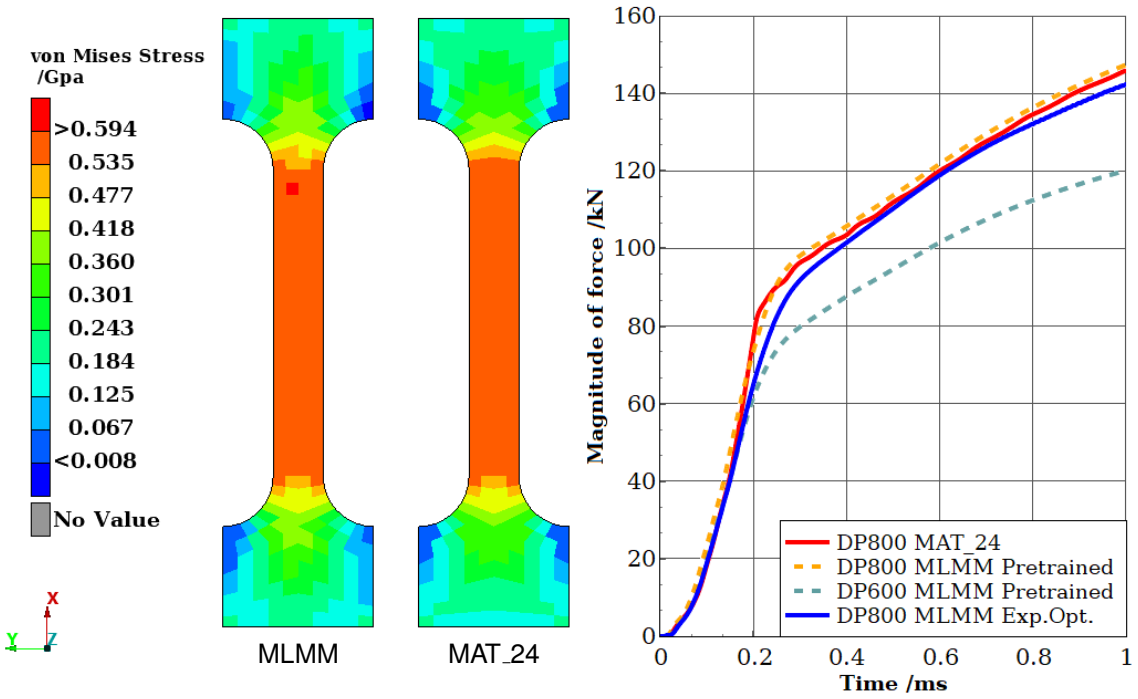

Figure 12: Validation of MLMM trained on small tensile test with divergence on large A80 tensile test. Stress field (left) at 0.53 ms.

4 Summary & Outlook

We proposed a method to train machine learning material models for a new material based solely on data acquirable in experiments and validated the method for a recurrent ML-model modelling plastic behavior with two variations of a virtual tension test. For this purpose, loss functions based on physically motivated equations were introduced. We showed that training the model with the proposed alternative loss functions can be done by taking a weighted sum of the different losses, as well as optimizing the loss functions simultaneously by a multi-objective optimization approach without the need to predefine suitable weights. The different trained models were then used and validated in different simulations.

Since the performance of the optimized models depends heavily on the training data (as can be seen in the evaluation), further variants of characterization experiments or a combination of various experiments can be tested for training in the future to generate a diverse and representative dataset for training. Though the method is designed to train MLMM with real experiments, the method still has to be validated using data with possible measurement noise obtained in physical tests.

5 Funding

This work was supported by the Federal Ministry for Economic Affairs and Climate Action of Germany for project AIMM (Artificial Intelligence for Materials Modelling; Identification 19I20024).

References

1. Alfeld, P. A trivariate clough—tocher scheme for tetrahedral data. *Computer Aided Geometric Design* **1,** 169–181. ISSN: 0167-8396. doi:https://doi.org/10.1016/0167-8396(84)90029-3. https://www.sciencedirect.com/science/article/pii/0167839684900293 (1984).
2. Blank, J. & Deb, K. pymoo: Multi-Objective Optimization in Python. *IEEE Access* **8,** 89497–89509 (2020).
3. Böhringer, P. et al. A strategy to train machine learning material models for finite element simulations on data acquirable from physical experiments. *Computer Methods in Applied Mechanics and Engineering* **406,** 115894. ISSN: 0045-7825. doi:https://doi.org/10.1016/j.cma.2023.115894. https://www.sciencedirect.com/science/article/pii/S0045782523000178 (2023).
4. Bonatti, C. & Mohr, D. On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids. *Journal of the Mechanics and Physics of Solids* **158,** 104697. ISSN: 0022-5096. doi:https://doi.org/10.1016/j.jmps.2021.104697. https://www.sciencedirect.com/science/article/pii/S0022509621003161 (2022).
5. Bonatti, C. & Mohr, D. One for all: Universal material model based on minimal state-space neural networks. *Science Advances* **7,** eabf3658. doi:10.1126/sciadv.abf3658 (2021).
6. Cho, K. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* **abs/1406.1078.** arXiv: 1406.1078. http://arxiv.org/abs/1406.1078 (2014).
7. Cooreman, S., Lecompte, D., Sol, H., Vantomme, J. & Debruyne, D. Identification of Mechanical Material Behavior Through Inverse Modeling and DIC. *Experimental Mechanics* **48,** 421–433. doi:10.1007/s11340-007-9094-0 (08/2008).
8. Deb, K. & Jain, H. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* **18,** 577–601. doi:10.1109/TEVC.2013.2281535 (2014).
9. Fabrice Pierron, M. G. *The Virtual Fields Method* doi:10.1007/978-1-4614-1824-5 (2012).
10. Flaschel, M., Kumar, S. & De Lorenzis, L. Unsupervised discovery of interpretable hyperelastic constitutive laws. *Computer Methods in Applied Mechanics and Engineering* **381,** 113852. ISSN: 0045-7825. doi:https://doi.org/10.1016/j.cma.2021.113852 (2021).

11. Ghaboussi, J., Garrett, J. H. & Wu, X. Knowledge-Based Modeling of Material Behavior with Neural Networks. *Journal of Engineering Mechanics* **117,** 132–153. doi:10.1061/(ASCE)0733-9399(1991)117:1(132) (1991).
12. Ghaboussi, J., Pecknold, D. A., Zhang, M. & Haj-Ali, R. M. Autoprogressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering* **42,** 105–126. doi:10.1002/(SICI)1097-0207(19980515)42:1<105::AID-NME356>3.0.CO;2-V (1998).
13. Hansen, N., Arnold, D. & Auger, A. *Evolution Strategies* doi:10.1007/978-3-662-43505-2_44 (01/2015).
14. Huang, D., Fuhg, J. N., Weißenfels, C. & Wriggers, P. A machine learning based plasticity model using proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering* **365,** 113008. ISSN: 0045-7825. doi:10.1016/j.cma.2020.113008 (2020).
15. Kajberg, J. & Lindkvist, G. Characterisation of materials subjected to large strains by inverse modelling based on in-plane displacement fields. *International Journal of Solids and Structures* **41,** 3439–3459. doi:10.1016/j.ijsolstr.2004.02.021 (06/2004).
16. Kirchdoerfer, T. & Ortiz, M. Data-driven computing in dynamics. *International Journal for Numerical Methods in Engineering* **113,** 1697–1710. doi:10.1002/nme.5716 (2018).
17. Liu, X., Tian, S., Tao, F. & Yu, W. A review of artificial neural networks in the constitutive modeling of composite materials. *Composites Part B: Engineering* **224,** 109152. ISSN: 1359-8368. doi:10.1016/j.compositesb.2021.109152 (2021).
18. Oishi, A. & Yagawa, G. Computational mechanics enhanced by deep learning. *Computer Methods in Applied Mechanics and Engineering* **327.** Advances in Computational Mechanics and Scientific Computation—the Cutting Edge, 327–351. ISSN: 0045-7825. doi:10.1016/j.cma.2017.08.040 (2017).
19. Rechenberg, I. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution* in (Frommann-Holzboog Verlag, 1973).
20. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning internal representations by error propagation (1986).
21. Schwefel, H.-P. *Evolution and Optimum Seeking* (01/1995).
22. Sommer, D., Troff, B. & Middendorf, P. in *Current Perspectives and New Directions in Mechanics, Modelling and Design of Structural Systems* 273–278 (CRC Press, 09/2022). doi:10.1201/9781003348443-43. https://doi.org/10.1201/9781003348443-43.
23. Sprave, J., Erhart, T. & Haufe, A. *An Interprocess Communication based Integration of AI User Materials into LS-DYNA* in *14th European LS-Dyna Konferenz, Baden-Baden, Germany* (10/15/2023).
24. Stainier, L., Leygue, A. & Ortiz, M. Model-free data-driven methods in mechanics: material data identification and solvers. *Computational Mechanics* **64,** 381–393. doi:10.1007/s00466-019-01731-1. https://doi.org/10.1007/s00466-019-01731-1 (06/2019).
25. Stander, N. et al. *DIC-based Full-Field Calibration using LS-OPT: An Update* in *15th European LS-DYNA Conference 2018, Detroit, USA* (2018).