Entwicklung von Machine Learning basierten Materialmodellen für Finite-Elemente-Simulationen

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Pauline Böhringer

Dortmund
2024

Tag der mündlichen Prüfung: 13.03.2025
Dekan: Prof. Dr. Jens Teubner
Gutachter/Gutachterinnen: Prof. Dr. Günter Rudolph, Prof. Dr.-Ing. Petra Wiederkehr

Zusammenfassung

Für die Strukturanalysen mechanischer Komponenten sind Finite-Elemente-Simulationen von großer Wichtigkeit. Sie erlauben eine umfassende und genaue Analyse, wie sich Bauteile unter verschiedenen Bedingungen verhalten. Diese Art von Simulation wird in einer Vielzahl von Bereichen angewendet, einschließlich des Umformens von Materialien bei der Herstellung von Komponenten, wie dies beispielsweise in der Automobilindustrie zum Einsatz kommt. Ebenso sind Simulationen für Crashtests essenziell, um die Sicherheit von Fahrzeugen zu bewerten. Die Güte und Präzision dieser Simulationen hängen maßgeblich von den verwendeten Materialmodellen ab, die darauf abzielen, das Verhalten von Werkstoffen unter diversen Belastungen möglichst exakt zu beschreiben. Die Erstellung dieser Modelle sowie die Bestimmung eines optimalen Modells für einen spezifischen Werkstoff sind jedoch komplex und anspruchsvoll. Sie erfordern aufwendige Test- und Kalibrierungsprozesse sowie umfangreiche Fachkenntnisse.

In dieser Arbeit wird untersucht, inwiefern sich klassische Materialmodelle durch datenbasierte Materialmodelle mit Methoden des maschinellen Lernens (ML) ersetzen lassen. Zunächst werden auf Basis des klassischen Materialmodells verschiedene Modellierungsmöglichkeiten im Bereich des maschinellen Lernens untersucht und mögliche ML-Modelle und Methoden für das Training identifiziert. Auf Basis dessen werden anhand von Trainingsdaten, welche mittels des klassischen Materialmodells erstellt wurden, unterschiedliche Modelle trainiert (unter anderem mit domänenspezifischen Anpassungen in der Struktur der Modelle). Hierbei soll in einem ersten Schritt untersucht und evaluiert werden, inwiefern klassische Materialmodelle durch ML-Modelle nachmodelliert werden können und welche Methode sich am besten als Ersatzmodell eignet. Die Auswertung der entwickelten ML-Materialmodelle wird insbesondere unter Berücksichtigung der Anwendung in verschiedenen Validierungssimulationen durchgeführt.

Der große Vorteil von datengetriebenen Materialmodellen gegenüber den bestehenden klassischen Modellen zeigt sich jedoch erst, wenn die Erstellung der ML-Modelle ohne ein bereits existierendes passendes klassisches Materialmodell möglich ist. Für die

Trainingsdaten stehen in diesem Fall lediglich Charakterisierungsversuche, welche mit dem Werkstoff durchgeführt werden zur Verfügung. Die große Herausforderung beim Training der ML-Modelle basierend auf Versuchsdaten ist die Abwesenheit der korrekten Ausgabegröße (lokale Spannungen), anhand derer die ML-Modelle während des Trainings lernen und angepasst werden.
Daher wird im zweiten Teil dieser Arbeit eine Methode vorgestellt, um ML-Materialmodelle für einen neuartigen Werkstoff anzupassen, welcher anstatt der Abweichung zwischen den vorhergesagten Ausgabewerten des ML-Modells und den korrekten Ausgabewerten physikalisch motivierte Gleichungen benutzt, welche für die vom ML-Materialmodell vorhergesagten Ausgabewerte gelten sollen, um die trainierbaren Parameter des ML-Modells zu optimieren. Der Fokus der Arbeit liegt hierbei auf der Anpassung bestehender sowie der Betrachtung und Untersuchung diverser alternativer Trainingsalgorithmen für das Training der ML-Materialmodelle.

Vorwort

Die vorliegende Arbeit entstand im Zuge einer Industriepromotion bei der Daimler AG beziehungsweise Mercedes-Benz AG im Rahmen des vom Bundesministerium für Wirtschaft und Klimaschutz geförderten Verbundprojektes AIMM - Artificial Intelligence for Material Models (Förderkennzeichen 19I20024).
Ich danke allen, die mich während meiner Promotionszeit unterstützt haben, insbesondere Dr. Said Jamei, Dr. Joachim Sprave und meinem Doktorvater Prof. Dr. Günter Rudolph.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation . . . 1
1.2 Stand der Wissenschaft . . . 2
1.3 Einordnung der Arbeit und Zielsetzung . . . 5
1.4 Aufbau der Arbeit . . . 6

2 Theoretische Grundlagen **9**

2.1 Kontinuumsmechanik, Festkörpermechanik und Materialmodelle . . . 9
2.1.1 Beschreibung der Deformation eines Körpers . . . 10
2.1.2 Spannungen . . . 11
2.1.3 Dehnungen . . . 14
2.1.4 Gleichgewichtsbedingungen . . . 15
2.2 Finite-Elemente-Simulationen . . . 15
2.2.1 Grundkonzept der FEM am Beispiel eines Stabs . . . 17
2.3 Maschinelles Lernen . . . 23
2.3.1 Lineare Regression und nichtlineare Erweiterungen . . . 24
2.3.2 Baumbasierte ML-Verfahren . . . 25
2.3.3 Künstliche neuronale Netze . . . 27

3 Training von ML-Modellen basierend auf klassischen Materialmodellen **35**

3.1 Linear elastisches Materialmodell . . . 35
3.2 Materialmodell mit Plastizität . . . 40
3.2.1 Klassische Berechnung . . . 40
3.2.2 Modellierungsmöglichkeiten im ML-Bereich . . . 44
3.2.3 Generieren von Trainings- und Testdaten . . . 47
3.2.4 Training von ML-Materialmodellen ohne Geschichtsvariablen . . . 55
3.2.5 Training von ML-Materialmodellen mit Geschichtsvariablen . . . 60
3.2.6 Auswertung, Validierung und Vergleich der Modelle . . . 72
3.2.7 Anwendung und Validierung der ML-Modelle in FE-Simulationen . . . 77

4 Training von ML-Materialmodellen basierend auf Versuchsdaten **95**
4.1 Verfügbare Trainingsdaten und grundlegende Methodik 95
4.2 Bestimmen der Fehlerfunktion 98
4.2.1 Allgemeiner Fall 98
4.2.2 Fehlerfunktion für die 2D-Formulierung von Schalenelementen . . . 102
4.3 Optimierungsstrategien für das Training mit Versuchsdaten 103
4.3.1 Training der ML-Modelle als mehrkriterielles Optimierungsproblem vs. Skalarisierung 103
4.3.2 Training der ML-Materialmodelle durch evolutionäre Algorithmen ohne Gradientenberechnung 105
4.3.3 Training mit angepasstem Backpropagation-Verfahren 108
4.4 Validierung der Methode anhand des linear elastischen Materialmodells . . 111
4.4.1 Training anhand des Patch Ausschnitts eines Zugversuches 113
4.4.2 Training anhand eines erweiterten Ausschnitts eines Zugversuches . 123
4.4.3 Verbesserung der Verallgemeinerungsfähigkeit durch Datenaugmentierung mithilfe Transformationen 131
4.5 Anwendung der Methode für ein Materialmodell mit Plastizität 135
4.5.1 Training anhand des Patch Ausschnitts eines Zugversuches 137
4.5.2 Training basierend auf neu entwickelter Zugversuchsprobe 140
4.5.3 Verbesserung der Verallgemeinerungsfähigkeit durch die Kombination mehrerer Versuchsproben 146

5 Zusammenfassung und Ausblick **157**
5.1 Zusammenfassung 157
5.2 Ausblick 160

Abbildungsverzeichnis

2.1 Referenz- und Momentankonfiguration . . . 10
2.2 Spannungskomponenten an einem Punkt eines Körpers . . . 12
2.3 Diskretisierung eines Kontinuums in Finite Elemente . . . 16
2.4 Volumen- und Schalenelemente in FE-Simulationen . . . 17
2.5 Skizze des betrachteten Stabs . . . 17
2.6 Diskretisierung des Stabs in die drei Stab-Elemente e_1, e_2, e_3 . . . 18
2.7 Ansatzfunktionen eines Stabelements . . . 19
2.8 Übersicht der Teilbereiche des maschinellen Lernens . . . 23
2.9 Beispiel einer Partitionierung des Eingaberaumes bei Entscheidungsbäumen 26
2.10 Beispiele für Aktivierungsfunktionen . . . 28
2.11 Aufbau eines künstlichen neuronalen Netzes . . . 29
2.12 Skizze eines rekurrenten Neurons . . . 32

3.1 Fehlerverläufe Training ML-Modelle linear elastisches Materialmodell . . . 37
3.2 Kragarm Simulation linear elastisches Materialmodell . . . 39
3.3 Fließkurve für den Stahl DP800 . . . 41
3.4 Einfluss der Änderung in Verläufen des klassischen Materialmodells auf spätere Zeitschritte - Beispielpfad . . . 46
3.5 Einfluss der Änderung in Verläufen des klassischen Materialmodells auf spätere Zeitschritte - Abweichungen je Zeitschritt und Komponente . . . 47
3.6 Trainingsdaten - Häufigkeitsverteilung Dehnungsinkremente stückweise lineare Dehnpfade . . . 49
3.7 Trainingsdaten - Beispielverläufe stückweise lineare Dehnpfade . . . 50
3.8 Trainingsdaten - Beispielverläufe Spline-Dehnpfade . . . 51
3.9 Trainingsdaten - Häufigkeitsverteilung Dehnungsinkremente Spline-Dehnpfade 51
3.10 Validierungsdatensatz Hutprofil - Aufbau Simulation . . . 52
3.11 Validierungsdatensatz Hutprofil - Beispielverläufe . . . 53
3.12 Vergleich Trainings- und Validierungsdatensätze - Häufigkeitsverteilung Dehnungsinkremente und plastische Dehnung . . . 54
3.13 Abdeckung der Datensätze bezüglich Triaxialität und plastischer Dehnung 54

3.14 Hyperparameterstudie LSTM und GRU (Netzarchitektur) 57
3.15 Hyperparameterstudie LSTM und GRU (Trainingsprozess, Skalierung und Aktivierungsfunktion) 57
3.16 Hyperparameterstudie LSTM und GRU (Anzahl trainierbarer Parameter) 58
3.17 Hyperparameterstudie LMSC-Modelle 59
3.18 Vergleich quantile und uniforme Knotensetzung bei Spline Transformationen 61
3.19 Spline Transformationen für die jeweiligen Eingabewerte 62
3.20 Auswertung Spline-Regressionsmodelle 63
3.21 Auswertung Entscheidungsbäume - MSE-Werte 64
3.22 Auswertung Entscheidungsbäume - R^2-Werte 64
3.23 Verlauf MSE-Werte von Random Forests bei Erhöhung der Anzahl an Schätzern 65
3.24 Hyperparameterstudie FFNN (Netzarchitektur) 66
3.25 Hyperparameterstudie FFNN (Trainingsprozess) 67
3.26 Hyperparameterstudie FFNN (Anzahl trainierbarer Parameter) 68
3.27 Schematische Darstellung FF-LMSC 69
3.28 Untersuchung verschiedener Aktivierungsfunktionen der FF-LMSC-Modelle für die Ausgabe der plastischen Dehnung 71
3.29 Untersuchung verschiedener Aktivierungsfunktionen der FF-LMSC-Modelle für die Spannungsausgabe 71
3.30 Vergleich Fehlerverlauf $\text{MSE}_{t,\sigma}$ für FFNN und FF-LMSC mit und ohne Fehlerfortpflanzung 75
3.31 Vergleich Fehlerverlauf $\text{MSE}_{t,\sigma}$ mit Fehlerfortpflanzung für den Datensatz D^0 76
3.32 Vergleich Fehlerverläufe $\text{MSE}_{t,\sigma}$, MSE_{t,σ_i}, $i = 1, 2, 4$ mit Fehlerfortpflanzung für den Datensatz D^{hat} 77
3.33 Aufbau Zugversuch Simulation 79
3.34 Auswertung Zugversuch - D3plot 80
3.35 Auswertung Zugversuch - Verläufe $\text{MSE}_{t,\sigma_{v,M}}$ und $\text{MSE}_{t,pl}$ 81
3.36 Auswertung Zugversuch - MSE-Verläufe Spannungskomponenten 81
3.37 Aufbau Hutprofil Simulation 82
3.38 Auswertung Hutprofil - D3plot (neuronale Netze) 83
3.39 Auswertung Hutprofil - Verläufe $\text{MSE}_{t,\sigma_{v,M}}$ und $\text{MSE}_{t,pl}$ 83
3.40 Auswertung Hutprofil - MSE-Verläufe Spannungskomponenten 84
3.41 Auswertung Hutprofil - Verschiebung Impaktor 85

3.42 Auswertung Hutprofil - D3plot (Spline Regressionsmodell und Random Forest) . . . 85
3.43 Aufbau der Crashbox Simulation . . . 86
3.44 Auswertung Crashbox - D3plot . . . 87
3.45 Auswertung Crashbox - Verläufe $MSE_{t,\sigma_{v,M}}$ und $MSE_{t,pl}$ . . . 88
3.46 Auswertung Crashbox - MSE-Verläufe Spannungskomponenten . . . 89
3.47 Auswertung Crashbox - Verschiebung Impaktor . . . 89
3.48 Bestandteile Kreuznapf Simulation . . . 90
3.49 Gesamter Aufbau Kreuznapf Simulation . . . 91
3.50 Kreuznapf Umformsimulation für den Zeitschritt t=46662 (Stempel und Platine) . . . 91
3.51 Auswertung Kreuznapf - D3plot . . . 92
3.52 Auswertung Kreuznapf - Verläufe $MSE_{t,\sigma_{v,M}}$ und $MSE_{t,pl}$ . . . 92
3.53 Auswertung Kreuznapf - MSE-Verläufe Spannungskomponenten . . . 93

4.1 Übersicht Trainingsmethode für ML-Materialmodelle mit Versuchsdaten . . . 97
4.2 Anordnung von Mess- und Gitterpunkten auf einer Versuchsprobe . . . 100
4.3 Notation Nachbarpunkte auf Überlagerungsgitter . . . 100
4.4 Training von ML-Materialmodellen mittels evolutionärer Algorithmen . . . 107
4.5 Notationen Ausgabeneuron neuronales Netz . . . 109
4.6 Versuchsproben für das Training mit Versuchsdaten . . . 112
4.7 Aufbau Simulation Patch-Ausschnitt Zugversuch . . . 114
4.8 Training linear elastisch V_p - Verläufe $L_{total}(w)$ und $MSE(V_p, ML)$ für $ML^{linel,36}_{180-0,2}$ . . . 114
4.9 Training linear elastisch V_p - Vergleich Optimierungsstrategien . . . 115
4.10 Training linear elastisch V_p - Verläufe von Mittelwert und Varianz der verschiedenen CMA-Optimierungsläufe . . . 116
4.11 Training linear elastisch V_p - $MSE(V_p, ML)$ Verlauf bei Optimierung einzelner Fehlerfunktionen . . . 118
4.12 Training linear elastisch V_p - Ergebnisse mehrkriterielle Optimierung . . . 120
4.13 Training linear elastisch V_p - Verläufe $L_{total}(w)$ und $MSE(V_p, ML)$ für $ML^{linel,384}_{180-0,2}$ . . . 122
4.14 Training linear elastisch V_p - Abweichung Testdatensatz . . . 123
4.15 Aufbau Simulation erweiterter Ausschnitt Zugversuch . . . 123
4.16 Häufigkeitsverteilung Triaxialität für V_p und V_{mfz} . . . 124

4.17 Training linear elastisch V_{mfz} - Verläufe $L_{total}(w)$ und MSE(V_{mfz}, ML) für $ML^{linel,36}_{180-0,2}$ . . . 125
4.18 Training linear elastisch V_{mfz} - Vergleich Optimierungsstrategien . . . 126
4.19 Training linear elastisch V_{mfz} - Verläufe von Mittelwert und Varianz der verschiedenen CMA-Optimierungsläufe . . . 127
4.20 Training linear elastisch V_{mfz} - Anwendung Gradient einzelner Fehlerfunktionen . . . 128
4.21 Training linear elastisch V_{mfz} - MSE(V_{mfz}, ML) Verlauf bei Optimierung einzelner Fehlerfunktionen . . . 128
4.22 Training linear elastisch V_{mfz} - Abweichung Testdatensatz . . . 131
4.23 Fehlerverlauf Training mit zusätzlich augmentierten Daten . . . 134
4.24 Fließkurven für DP600 und DP800 . . . 136
4.25 Training elastoplastisch V_p - Verläufe $L_{total}(w)$ und MSE(V_p, ML) . . . 137
4.26 Training elastoplastisch V_p - Ergebnisse mehrkriterielle Optimierung . . . 139
4.27 Training elastoplastisch V_p - Abweichung Testdatensatz . . . 140
4.28 Abdeckung Triaxialität und der plastischen Dehnung für V_p und V_{lz} . . . 141
4.29 Neuartige Versuchsprobe (Lochzugprobe) . . . 141
4.30 Training elastoplastisch V_{lz} - Verläufe $L_{total}(w)$ und MSE(V_{lz}, ML) . . . 142
4.31 Training elastoplastisch V_{lz} - Ergebnisse mehrkriterielle Optimierung . . . 144
4.32 Training elastoplastisch V_{lz} - Abweichung Testdatensatz . . . 146
4.33 Zusätzlich betrachtete Gissmo-Zugproben . . . 147
4.34 Unterschiedliche Belastungen der Lochzugprobe . . . 148
4.35 Abdeckung Triaxialität und der plastische Dehnung für V_{lz} und V_{kombi} . . . 149
4.36 Training elastoplastisch V_{kombi} - Verläufe $L_{total}(w)$ und MSE(V_{kombi}, ML) 149
4.37 Training elastoplastisch V_{kombi} - Abweichungen für einzelne Versuche . . . 150
4.38 Training elastoplastisch V_{kombi} - Abweichung Testdatensatz . . . 152
4.39 Training elastoplastisch V_{kombi} - Vergleich D3plot Zugversuch . . . 154

Tabellenverzeichnis

3.1 Auswertung ML-Modelle für linear elastisches Materialmodell 38
3.2 Vergleich Trainings- und Validierungsdatensätze - Mittelwert, Varianz und Schiefe 52
3.3 Übersicht Hyperparameter LSTM und GRU 56
3.4 Übersicht Hyperparameter LMSC-Modelle 59
3.5 Auswertung Random Forests 65
3.6 Übersicht Hyperparameter FFNN 66
3.7 Übersicht FFNN-Modelle mit kleinster MSE-Abweichung 67
3.8 Vergleich ML-Materialmodelle - MSE Testdatensätze ohne Fehlerfortpflanzung 73
3.9 Vergleich ML-Materialmodelle - MSE Testdatensätze mit Fehlerfortpflanzung 74
3.10 Vergleich ML-Materialmodelle - MSE_σ Testdatensätze mit Fehlerfortpflanzung 76
3.11 Auswertung Zugversuch - Vergleich MSE_σ, $\text{MSE}_{\sigma_{v,M}}$, MSE_{disp}, R^2_{pl}, MAE^{rel}_{disp} 79
3.12 Auswertung Hutprofil - Vergleich MSE_σ, $\text{MSE}_{\sigma_{v,M}}$, MSE_{disp}, R^2_{pl}, MAE^{rel}_{disp} 84
3.13 Auswertung Crashbox - Vergleich MSE_σ, $\text{MSE}_{\sigma_{v,M}}$, MSE_{disp}, R^2_{pl}, MAE^{rel}_{disp} 88
3.14 Auswertung Kreuznapf - Vergleich MSE_σ, $\text{MSE}_{\sigma_{v,M}}$, MSE_{disp},R^2_{pl} , MAE^{rel}_{disp} 93

4.1 Beispiel mögliche Gegenläufigkeit der Fehlerfunktionen 105
4.2 Training linear elastisch V_p - Mittelwerte und Varianz der unterschiedlichen CMA-Optimierungsläufe 117
4.3 Training linear elastisch V_p - Vergleich verschiedener Gewichtungen 119
4.4 Training linear elastisch V_p - Ergebnisse mehrkriterielle Optimierung . . . 119
4.5 Training linear elastisch V_p - Übersicht Fehlerwerte 121
4.6 Training linear elastisch V_{mfz} - Mittelwerte und Varianz der unterschiedlichen CMA-Optimierungsläufe 127
4.7 Training linear elastisch V_{mfz} - Vergleich verschiedener Gewichtungen . . 129
4.8 Training linear elastisch V_{mfz} - Ergebnisse mehrkriterielle Optimierung . . 129
4.9 Training linear elastisch V_{mfz} - Übersicht Fehlerwerte 130
4.10 Training elastoplastisch V_p - Übersicht Fehlerwerte 138

4.11 Training elastoplastisch V_{lz} - Vergleich verschiedener Gewichtungen 143
4.12 Training elastoplastisch V_{lz} - Ergebnisse mehrkriterielle Optimierung . . . 145
4.13 Training elastoplastisch V_{lz} - Übersicht Fehlerwerte 145
4.14 Training elastoplastisch V_{kombi} - Übersicht Fehlerwerte 151
4.15 Training elastoplastisch V_{kombi} - Übersicht Abweichungen Simulationsergebnisse 153

1 Einleitung

1.1 Motivation

In der modernen Ingenieurwissenschaft stellen Finite-Elemente-Simulationen ein unverzichtbares Werkzeug dar, um komplexe Strukturen mechanischer Komponenten unter verschiedensten Bedingungen zu analysieren. Diese numerische Methode ermöglicht es die Reaktion von Festkörpern auf äußere Belastungen vorherzusagen, was für die Entwicklung neuer Produkte und die Optimierung bestehender Konstruktionen von großer Bedeutung ist. Unternehmen streben dabei an, durch das frühzeitige Identifizieren von Schwachstellen in der Entwicklungsphase sowohl die Dauer der Entwicklung zu verkürzen als auch Kosten zu reduzieren und gleichzeitig die Qualität zu erhöhen. Die Simulationen kommen dabei in unterschiedlichsten Anwendungsfeldern zum Einsatz wie beispielsweise beim Umformen von Werkstoffen als wichtiger Prozess in der Fertigung von Bauteilen (unter anderem in der Automobilindustrie) oder auch bei Crash-Simulationen (Simulation von Unfallszenarien) bei der Entwicklung von Fahrzeugen für die Sicherheitsbewertung. Die Simulationen stellen hierbei eine deutlich kostengünstigere und effizientere Alternative beziehungsweise Ergänzung zu den real durchgeführten experimentellen Versuchen dar.

Die Genauigkeit dieser Simulationen hängt jedoch maßgeblich von der Qualität der zugrunde liegenden Materialmodelle, welche das makroskopische Verhalten eines spezifischen Werkstoffes in der Simulation beschreiben, ab. Traditionelle Materialmodelle werden basierend auf physikalischen Gesetzmäßigkeiten sowie Prinzipien der Kontinuumsmechanik hergeleitet. Die Entwicklung neuer Materialmodelle ist allerdings eine sehr anspruchsvolle und langwierige Aufgabe. Auch die Auswahl sowie Anpassung eines geeigneten Materialmodells durch die Wahl von passenden Materialparametern für den Einsatz eines neuartigen Werkstoffs in Simulationen erfordert kostspielige und langwierige Test- und Kalibrierungsverfahren sowie ein hohes Maß an Expertise. Die Basis für den Prozess der Identifikation von Materialmodell und Materialparameter bilden hier Versuche (Charakterisierungsversuche), welche die Materialeigenschaften des betrachteten Werkstoffs kenntlich machen.

Um diese aufwändigen Prozesse in der Materialmodellierung zu unterstützen und zu beschleunigen bieten Methoden der künstlichen Intelligenz (KI) und insbesondere der Teilbereich des maschinellen Lernens (ML) neue Möglichkeiten.
ML-Modelle sind in der Lage aus Daten zu lernen und nicht offensichtliche Muster und Zusammenhänge zu erkennen. Diese Erkenntnisse können beispielsweise genutzt werden, um Vorhersagen zu treffen, Entscheidungen zu unterstützen oder automatisierte Aufgaben in einer Vielzahl von Anwendungsbereichen auszuführen. In den letzten Jahren hat maschinelles Lernen aufgrund vieler neuer Entwicklungen erheblich an Relevanz einschließlich in den Einsatzgebieten der digitalen Produktentwicklung, Simulation und Materialmodellierung gewonnen (siehe [55, 46]). Im Bereich der Materialmodellierung verspricht hierbei ein Ersatz der klassischen Modelle durch datenbasierte Materialmodelle mit Methoden des maschinellen Lernens eine Möglichkeit die langwierige Erstellung der klassischen Materialmodelle zu überwinden und den Prozess der Wahl und Kalibrierung der klassischen Modelle von dem benötigten Expertenwissen abzukoppeln sowie der Möglichkeit Prozesse in der Materialmodellierung zu automatisieren und damit die benötigte Zeit sowie Kosten für die Modellentwicklung zu reduzieren. Die Motivation dieser Arbeit liegt demzufolge in der Erforschung und Entwicklung von ML-basierten Materialmodellen für Finite-Elemente-Simulationen, um Grenzen der klassischen Materialmodellierung zu überwinden und die Potenziale, Eignung aber auch die Herausforderungen des maschinellen Lernens in der Materialmodellierung zu identifizieren und zu untersuchen.

1.2 Stand der Wissenschaft

Maschinelles Lernen wurde in den letzten Jahren zunehmend in verschiedenen Anwendungsbereichen der Kontinuumsmechanik und Materialwissenschaften eingesetzt, wie eine Übersicht aus [8] zeigt. Wissenschaftler erwägen immer häufiger klassische, analytisch definierte Materialmodelle durch Ansätze, die auf datenbasierten ML-Methoden basieren, zu ersetzen beziehungsweise zu ergänzen und zu unterstützen. Diese datenbasierten Modelle bieten das Potential auch ohne umfassendes Verständnis vorhandener Materialtheorien trainiert zu werden und können eine effiziente Berechnung, die das iterative Lösen komplexer Gleichungen umgeht, ermöglichen. Des Weiteren bestehen durch das Nutzen von ML-Methoden Möglichkeiten schneller und einfacher an geeignete Materialmodelle zu gelangen oder auch in manchen Anwendungsfällen präzisere Modelle zu erhalten. Um diese Ziele zu erreichen kann maschinelles Lernen im Bereich der Materialmodellierung auf unterschiedliche Art und Weise angewendet werden.

Eine Möglichkeit ergibt sich in der Vorhersage und Identifikation von Materialparametern für die klassischen Materialmodelle, welche weiterhin bestehen bleiben und in der Simulation verwendet werden. So wird beispielsweise in [42] die Fließkurve eines Aluminium-Werkstoffs in Abhängigkeit der Dehnung, Dehnrate und Belastungsrichtung mittels eines neuronalen Netzes beschrieben. Li et al. substituierten in [48] Teile des Verfestigungsgesetzes durch Methoden des maschinellen Lernens, um die Interaktion von Dehnrate und Temperatur hinsichtlich des Verfestigungsverhaltens von Dualphasenstählen darzustellen. In [2] werden mittels eines neuronalen Netzes Schädigungsparameter identifiziert, um eine beschleunigte Materialcharakterisierung zu erreichen. Ein weiteres Beispiel sind die Arbeiten von Asteris et al., welche in [5] einen Ansatz präsentieren, um Druckfestigkeit und Elastizitätsmodul von Sandbetonmaterialien mittels eines neuronalen Netzes vorherzusagen.

Ein weiteres Anwendungsfeld ergibt sich in Ansätzen, bei denen ein Teilbereich des klassischen Materialmodells durch ML-Methoden ersetzt wird. So wird in [10] das iterative Korrekturverfahren des J2-basierten von Mises Elastoplastizitätsmodells durch ein nichtlineares Regressionsmodell ersetzt, um die Berechnung der Materialantwort zu beschleunigen. Ein sehr ähnlicher Ansatz wird in [40] mittels eines künstlichen neuronalen Netzes realisiert. In [11] wird der Dehnungstensor in den volumetrischen Anteil sowie den deviatorischen Anteil zerlegt und für den deviatorischen Anteil Spannungsvorhersagen mittels eines neu eingeführten rekurrenten neuronalen Netzes aufbauend auf einem Ansatz aus [12] vorhersagt. Weitere Arbeiten finden sich in [73], in denen ein neuronales Netz trainiert wird, welches anstelle der Spannungsausgaben den Cholesky Faktor der Tangentensteifigkeitsmatrix vorhersagt, auf dessen Basis die Spannungen berechnet werden, wobei sichergestellt wird, dass die Tangentensteifigkeitsmatrix symmetrisch positiv semidefinit ist. Weitere Ansätze aus [69] und [26] bestehen in der Modularisierung der klassischen Modelle wie beispielsweise im Fall von elastoplastischen Materialverhalten der Elastizität sowie der Fließ- und Verfestigungsgesetze, wobei für die einzelnen Module des Materialmodells jeweils ML-Methoden oder die klassische Berechnung benutzt werden können.

Der höchste Grad der ML-Integration im Bereich der Materialmodellierung ergibt sich durch das Ersetzten des gesamten klassischen Materialmodells durch ein ML-Modell, welches vollkommen losgelöst von der analytischen Formulierung in Simulationen eingesetzt werden kann. Eine der ersten Anwendungen in diesem Bereich sind die Arbeiten

von Ghaboussi et al. 1991, welche in [28] die Beschreibung des Verhaltens von Beton mittels eines künstlichen neuronalen Netzes thematisieren. Weitere Arbeiten im Bereich der Geomechanik folgen unter anderem mit [29] für die Modellierung von Sand in Triaxialversuchen, dem Materialverhalten von Laminaten [60] oder für die Anwendung für Stein [54]. Bis auf sehr wenige Ausnahmen im Bereich der Materialmodellierung, wie die Anwendung von Support Vector Machines in [74] oder die Nutzung eines polynomiellen Regressionsmodells in [41] für die Beschreibung von Geomaterialien beschränken sich die Ansätze hierbei auf künstliche neuronale Netze.
Angesichts des Ziels ein allgemeingültiges ML-Materialmodell unter Berücksichtigung von plastischem Materialverhalten zu erhalten, welches für beliebige, nichtlineare Deformationen und längere Belastungskurven eine gute Prognose liefern kann, ist es entscheidend, dass während der Berechnung der Materialantwort nicht nur die aktuellen Informationen der betrachteten Größen dem ML-Modell zur Verfügung stehen, sondern auch die Belastungsvorgeschichte bekannt ist. Aus diesem Grund wurden bereits rekurrente neuronale Netze (RNN) betrachtet, welche zeitliche Abhängigkeiten berücksichtigen können, da sie Rückkopplungsschleifen enthalten, die Informationen aus vorherigen Zeitschritten speichern und in die Berechnungen der aktuellen Zeitschritte einbeziehen. Die ersten Arbeiten finden sich bereits in [25], [30] und [57], um das Verhalten von Sand, Erde und Ton mittels einfacher rekurrenter Netze zu modellieren. In jüngster Zeit wurden auch weitere RNN-Architekturen wie LSTMs und GRUs verwendet. Dies ist beispielsweise in [15] der Fall, um viskoelastisches Materialverhalten zu erlernen oder in den Arbeiten von Mozaffar et al. in [56], welche Verbundwerkstoffe anhand von Daten aus Simulationen von Volumenelementen unter verschiedenen, nichtlinearen Belastungspfaden modellieren. Weiterhin werden in [16] LSTM-Modelle betrachtet, um Verbundwerkstoffe zu modellieren.

Bei den verschiedenen ML-Lösungsansätzen zeichnet sich zudem der Trend ab Domänenwissen in die ML-Modelle einzubauen, um deren Mangel an Interpretierbarkeit zu adressieren, die Menge an notwendigen Trainingsdaten zu reduzieren oder um physikalisch konsistente Lösungen zu erhalten. Im Bereich der konstitutiven Gleichungen beschäftigten sich hierbei diverse Arbeiten mit Untersuchungen, inwiefern die geltenden Prinzipien, welche in der klassischen Materialmodellierung zur Erstellung der Modelle eingesetzt werden durch ML-Modelle erfüllbar sind. Zu diesen Prinzipien gehören Objektivität [47] und die Konvexität der Dehnungsenergiefunktion [73].

1.3 Einordnung der Arbeit und Zielsetzung

Die in Abschnitt 1.2 genannten Arbeiten konzentrieren sich auf spezifische, festgelegte Anwendungen und größtenteils eingeschränkter Validierungen mittels festgelegter Pfade ohne eine Anwendung innerhalb größerer, komplexer Bauteil-Simulationen beziehungsweise ersetzen lediglich einen Aspekt des klassischen Materialmodells. Das Erstellen eines universellen, robusten ML-Materialmodells für eine große Bandbreite an diversen Verlaufspfaden, wie es für die Anwendung in FE-Simulationen nötig ist, bleibt weiterhin eine große Herausforderung. Die Einsatzmöglichkeit eines ML-basierten Materialmodells in größeren und komplexeren Anwendungsfeldern in Form von Bauteilsimulationen mit vielen Zeitschritten und Elementen ist nach wie vor zu zeigen und zu evaluieren. In dieser Arbeit sollen daher für ein elastisches sowie elastoplastisches Materialverhalten ML-Materialmodelle trainiert werden, welche in Hinblick auf den Einsatz in FE-Simulationen entwickelt und evaluiert sowie für unterschiedliche Anwendungen insbesondere im Crash- und Umformbereich in LS-Dyna Simulationen integriert und validiert werden sollen. Elastoplastisches Materialverhalten wird hierbei als Anwendungsfall ausgewählt, da auf diese Weise große Bereiche technisch relevanter Werkstoffe charakterisiert werden können [71]. Konkret werden hierfür ML-Ersatzmodelle für die beiden in der Simulationssoftware LS-Dyna implementierten Materialmodelle `*MAT_ELASTIC` oder `*MAT_001` (linear elastisches Modell) und `*MAT_PIECEWISE_LINEAR_PLASTICITY` oder `*MAT_024` (elastoplastisches Modell) trainiert (siehe [49]). Während sich die meisten Arbeiten im Bereich der ML-basierten Materialmodellierung auf künstliche neuronale Netze beschränken, wird in dieser Arbeit ein breiteres Feld an ML-Methoden inklusive baumbasierter ML-Verfahren und Regressionsmethoden betrachtet. Hierfür soll das klassische Materialmodell in Hinblick auf unterschiedliche Modellierungsmöglichkeiten im ML-Bereich und im Bereich des statistischen Lernens betrachtet und verschiedene ML-Modelle getestet, evaluiert und miteinander verglichen werden sowie Vor- und Nachteile der unterschiedlichen Ansätze identifiziert werden, um die am besten geeignete Methode zu ermitteln.

Des weiteren dienen in bisherigen Arbeiten ML-Materialmodelle lediglich als Substitution eines bereits entwickelten und angepassten klassischen Materialmodells und können nur auf dieser Basis trainiert werden. Um einen neuartigen Werkstoff innerhalb Simulationen modellieren zu können, muss daher nach wie vor ein klassisches Materialmodell beziehungsweise passende Materialparameter auf Basis von Charakterisierungsversuchen bestimmt werden. Der große Vorteil von ML-basierten Materialmodellen ergibt sich allerdings erst, wenn das Training der ML-Materialmodelle unabhängig der klassischen Materialmodeller-

stellung und -kalibrierung möglich ist. Daher sollen in dieser Arbeit in einem weiteren Schritt Methoden entwickelt und untersucht werden, um ein ML-Materialmodell für einen neuartigen Werkstoff auf Basis von Daten aus Charakterisierungsversuchen trainieren zu können, ohne ein bereits bestehendes, klassisches Materialmodell zu benötigen.

Die Ziele dieser Arbeit können in den folgenden beiden Forschungsfragen zusammengefasst werden:

- Wie können klassische Materialmodelle durch datengetriebene Modelle mit Methoden des maschinellen Lernens für den Einsatz innerhalb Finite-Elemente-Simulationen ersetzt werden und welche Methode ist am besten dafür geeignet?
- Wie können ML-Materialmodelle ohne ein zugrunde liegendes, klassisches Materialmodell auf Basis von Daten aus Charakterisierungsversuchen trainiert werden und welche Trainingsalgorithmen eignen sich hierfür?

1.4 Aufbau der Arbeit

Zunächst werden in Kapitel 2 die theoretischen Grundlagen, die in dieser Arbeit von Bedeutung sind, erläutert. In Abschnitt 2.1 werden relevante Konzepte aus den Bereichen der Kontinuums- und Festkörpermechanik und der Zusammenhang sowie die Aufgabe der Materialmodelle innerhalb dieses Gebiets erklärt, gefolgt von der grundlegenden Idee und Funktionsweise von Finite-Elemente-Simulationen in 2.2. Des Weiteren werden in Abschnitt 2.3 die verschiedenen, in dieser Arbeit genutzten Methoden und Modelle des maschinellen Lernens eingeführt.

In Kapitel 3 wird untersucht, inwiefern sich die klassischen Materialmodelle durch Modelle und Methoden des maschinellen Lernens ersetzen lassen. Die Basis für das Training bieten hierbei Trainingsdaten, welche anhand der bestehenden, klassischen Materialmodelle erzeugt wurden. Die ML-Materialmodelle werden zunächst in Abschnitt 3.1 für ein linear elastisches Modell trainiert. Anschließend werden in 3.2 verschiedene Methoden vorgestellt, um ein komplexeres, nichtlineares Materialmodell mit Plastizität durch ML-Modelle zu ersetzen. Hierfür wird zunächst in Abschnitt 3.2.1 das klassische Materialmodell kurz vorgestellt und basierend darauf in 3.2.2 untersucht, welche Modellierungsmöglichkeiten im ML-Bereich grundsätzlich in Frage kommen. Die verschiedenen Varianten werden in den Abschnitten 3.2.4 und 3.2.5 anhand konkreter ML-Modelle auf Basis von Trainingsdaten, deren Generierung im vorherigen Abschnitt 3.2.3 erläutert wird, umgesetzt.

Die unterschiedlichen, trainierten ML-Materialmodelle werden anschließend in Abschnitt 3.2.6 vorerst anhand verschiedener Testdatensätze außerhalb von Simulationen evaluiert, miteinander verglichen und die performantesten ML-Materialmodelle abschließend in 3.2.7 durch den Einsatz in unterschiedlichen FE-Simulationen validiert.

Als weiterführenden Schritt zu den in 3 entwickelten Modellen wird in Kapitel 4 eine Methode vorgestellt, welche es ermöglicht die ML-Materialmodelle für einen neuartigen Werkstoff anzupassen, ohne ein klassisches Materialmodell, auf dessen Basis die Trainingsdaten erzeugt werden, entwickeln zu müssen. Hierfür wird zunächst in Abschnitt 4.1 ein Überblick über die Problematik vermittelt sowie die Unterschiede zur bisherigen Trainingsdatengenerierung erklärt und erläutert, welche Daten bei Versuchen, die üblicherweise für die Erstellung der klassischen Modelle genutzt werden, verfügbar sind. Des Weiteren wird eine Übersicht der grundlegenden Methodik und Vorgehensweise für das Training von ML-Materialmodellen basierend auf diesen Versuchsdaten präsentiert. Anschließend werden in Abschnitt 4.2 die Fehlerfunktionen aus [9], auf Basis derer die ML-Materialmodelle trainiert werden, näher erläutert und deren genaue Berechnung beschrieben. Abschnitt 4.3 beschäftigt sich anschließend mit unterschiedlichen Optimierungsmethoden und Ansätzen, welche für das Training von künstlichen neuronalen Netzen eingesetzt beziehungsweise auf den betrachteten Anwendungsfall angepasst werden können. Die vorgestellte Methode wird anschließend in 4.4 zunächst für das Training eines linear elastischen Materialmodells basierend auf unterschiedlichen simulierten Versuchen und mit verschiedenen Trainingsalgorithmen validiert. Abschließend wird in Abschnitt 4.5 die Validierung für das Materialmodell mit Plastizität erweitert, unter Berücksichtigung von zusätzlichen, komplexeren simulierten Versuchen sowie durch die Nutzung von Kombinationen mehrerer Versuche, um einen größeren und vielfältigeren Trainingsdatensatz zur Verfügung zu haben.

Kapitel 5 gibt einen zusammenfassenden Überblick der Ergebnisse dieser Arbeit und eröffnet einen Ausblick zu weiteren Forschungsfeldern.

Bereits publizierte Ergebnisse

Die folgenden beiden Publikationen entstanden während der Erstellung dieser Arbeit:

[9] Pauline Böhringer, Daniel Sommer, Thomas Haase, Martin Barteczko, Joachim Sprave, Markus Stoll, Celalettin Karadogan, David Koch, Peter Middendorf, Mathias Liewald. *A strategy to train machine learning material models for finite element simulations on data acquirable from physical experiments*, In: Computer Methods in Applied Mechanics and Engineering 406 (2023)

[66] Daniel Sommer, Pauline Böhringer, Markus Stoll, Peter Middendorf. *Using Data from Physical Experiments to Train Machine Learning Material Models*, 14th European LS-DYNA Conference 2023 Baden-Baden

Inhalte der Veröffentlichungen finden sich in dieser Arbeit wieder. Beide Veröffentlichungen thematisieren das Training von ML-Materialmodellen anhand von Versuchsdaten, wie es in Kapitel 4 dieser Arbeit vorgestellt wird. Der Beitrag der Autorin der vorliegenden Dissertation an der in [9] und [66] veröffentlichten Methode liegt im Vortraining der ML-Materialmodelle (wie in Kapitel 3 dieser Arbeit thematisiert) sowie im Einbinden der Fehlerfunktionen in das Training von ML-Modellen sowie der Auswertung und Untersuchung der unterschiedlichen Trainingsmöglichkeiten.

2 Theoretische Grundlagen

Die in dieser Arbeit entwickelten Methoden beruhen auf Grundlagen der Kontinuumsmechanik, der Materialmodelle im Bereich der Finite-Elemente-Simulationen sowie des maschinellen Lernens. Im Folgenden werden die für diese Arbeit relevanten Grundlagen aus den oben genannten Gebieten erläutert und im Laufe der Arbeit verwendete Notationen eingeführt. Die beschriebenen Konzepte im Bereich der Mechanik und Finite-Elemente-Methode beruhen auf Auszügen aus [34], [70] und [52]. Detaillierte Ausführungen und tiefergehende Erläuterungen sind darin zu finden. Für weitere Erläuterungen und Details im Bereich des maschinellen Lernens wird auf [32] und [27] verwiesen.

2.1 Kontinuumsmechanik, Festkörpermechanik und Materialmodelle

Die *Festkörpermechanik* beschäftigt sich mit der Beschreibung von Veränderungen eines festen Körpers bezüglich seiner Form und Lage aufgrund von äußeren Belastungen. In der *Kontinuumsmechanik* wird hierbei die Annahme getroffen, dass der Körper nicht aus diskreten Punkten im euklidischen Raum $\mathbb{R}^3$ besteht, sondern als kontinuierliche Teilmenge modelliert wird:

Definition 2.1.1. Ein *Körper* oder *Festkörper* K ist definiert als eine 3-dimensionale zusammenhängende Mannigfaltigkeit im euklidischen Raum $\mathbb{R}^3$. Die Menge aller Randpunkte von K wird im Folgenden mit R bezeichnet.

Eine Veränderung eines solchen Festkörpers kann mathematisch durch ein System von Differentialgleichungen beschrieben werden. Materialmodelle, welche makroskopisch das Verhalten eines bestimmten Materials beschreiben und formal eine Beziehung zwischen Dehnungen und Spannungen abhängig des Werkstoffes angeben, sind hierbei Teil des Differentialgleichungssystems. Realitätsnahe Problemstellungen sind in der Regel schwer analytisch zu lösen und werden deshalb approximativ durch numerische Verfahren bestimmt. Das im Rahmen dieser Arbeit genutzte numerische Verfahren ist die Finite-Elemente-Methode, die in Abschnitt 2.2 vorgestellt wird.

2.1.1 Beschreibung der Deformation eines Körpers

Um die Veränderung der Form und Lage eines Körpers (*Deformation*) zu verschiedenen Zeitpunkten darzustellen, wird die Lage jedes Punktes des Körpers durch einen Vektor $P \in \mathbb{R}^3$ zu einem bestimmten Zeitschritt $t \in \mathbb{R}_{\geq 0}$ beschrieben.

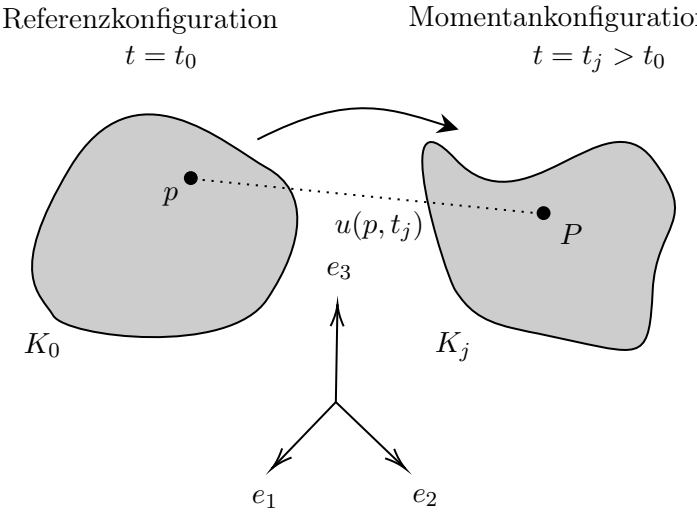


Abbildung 2.1: Ausgangszustand eines Körpers $K_0 \subset \mathbb{R}^3$ zu einem Anfangszeitpunkt t_0 (Referenzkonfiguration) und der Zustand des Körpers zu einem aktuell betrachteten Zeitpunkt $t > t_0$ (Momentankonfiguration), wobei mit e_1, e_2, e_3 die Einheitsvektoren des $\mathbb{R}^3$ bezeichnet werden

Die ursprüngliche Anordnung des Körpers K zu einem Anfangszeitpunkt t_0 wird als *Referenzkonfiguration* bezeichnet und die Anordnung zu einem aktuellen Zeitpunkt $t_j > t_0$ als *Momentankonfiguration* (siehe Abbildung 2.1). Der Körper zum Anfangszeitpunkt t_0 wird mit K_0 bezeichnet. In der Lagrange'schen Beschreibung, welche üblicherweise im Bereich der Festkörpermechanik genutzt wird, werden die relevanten Größen Verschiebungen, Spannungen und Dehnungen, welche im Folgenden beschrieben werden als stetig differenzierbare Funktionen (siehe auch [34])

$$f : K_0 \times \mathbb{R}_{\geq t_0} \to \mathbb{R}^3 \quad (p, t) \mapsto f(p, t) \tag{2.1}$$

der Zeit und der Anfangsposition eines Punktes $p \in \mathbb{R}^3$ in der Referenzkonfiguration betrachtet. Um die Veränderungen eines Körpers bezüglich seiner Form und Lage zu beschreiben, werden Verschiebungen der einzelnen Punkte eines Körpers herangezogen.

Die Verschiebung kann dafür als Funktion

$$u : K_0 \times \mathbb{R}_{\geq t_0} \to \mathbb{R}^3 \quad (p,t) \mapsto u(p,t) =: (u_x(p,t), u_y(p,t), u_z(p,t)) \tag{2.2}$$

dargestellt werden, welche die Differenz $u(p,t) = p - P$ zwischen der Lage eines Punktes $p \in K_0$ im Körper bezogen auf die Referenzkonfiguration und dem entsprechenden Punkt P zu einem Zeitpunkt $t \geq t_0$ abbildet (siehe Abbildung 2.1).
Diese Verschiebungen zu berechnen ist das Ziel in einer Finiten-Element-Simulation und wird als approximative Lösung eines Differentialgleichungssystems für das Volumen mit vorgegebenen Anfangs- und Randbedingungen (für die Menge aller Randpunkte R des Körpers K) bestimmt. Das Differentialgleichungssystem besteht grundlegend aus den folgenden Beziehungen und Bedingungen:

- Das Materialmodell (*Konstitutivgleichung*) beschreibt die Beziehung zwischen den Dehnungen und den Spannungen (siehe Abschnitt 2.1.3 und 2.1.2) abhängig von einem bestimmten Material, aus dem der Körper K besteht.

- Die Gleichgewichtsbedingungen (*Bilanzgleichungen*) beschreiben das Gleichgewicht der von außen wirkenden Kräfte und der resultierenden inneren Spannungen im Körper (siehe Abschnitt 2.1.4).

- Die Anfangs- und Randbedingungen geben für jeden Punkt am Rand R entweder die Verschiebung oder eine Oberflächenlast vor. Die erste Art der Randbedingung (*Verschiebungsrandbedingung*) wird auch als *kinematische, Dirichlet´sche* oder *wesentliche Randbedingung* bezeichnet und die zweite als *kinetische, Neumann'sche* oder *natürliche Randbedingung*.

2.1.2 Spannungen

Durch Spannungen werden punktuell Kräfte, die auf einen Körper einwirken, im Verhältnis zur Fläche, auf welche die Kraft wirkt, beschrieben. Um die Spannungen an einem Punkt eines Körpers zu beschreiben, kann eine Schnittfläche A des Körpers durch diesen Punkt betrachtet werden, wobei verteilt über diese Schnittfläche unterschiedliche Kräfte wirken. Um diesen Belastungszustand, der auf der Fläche wirkt, in einem einzelnen Punkt der Fläche definieren zu können, wird ein infinitesimal kleiner Teil der Schnittfläche ΔA mit der dort wirkenden mittleren Kraft ΔF betrachtet. Der *Spannungsvektor* k kann damit durch den Grenzwert

$$k := \lim_{\Delta A \to 0} \frac{\Delta F}{\Delta A} \tag{2.3}$$

definiert werden. Da dieser Spannungsvektor von der Wahl der Schnittfläche durch diesen Punkt abhängt, werden als Schnittebenen die Koordinatenebenen gewählt (siehe Abbildung 2.2).

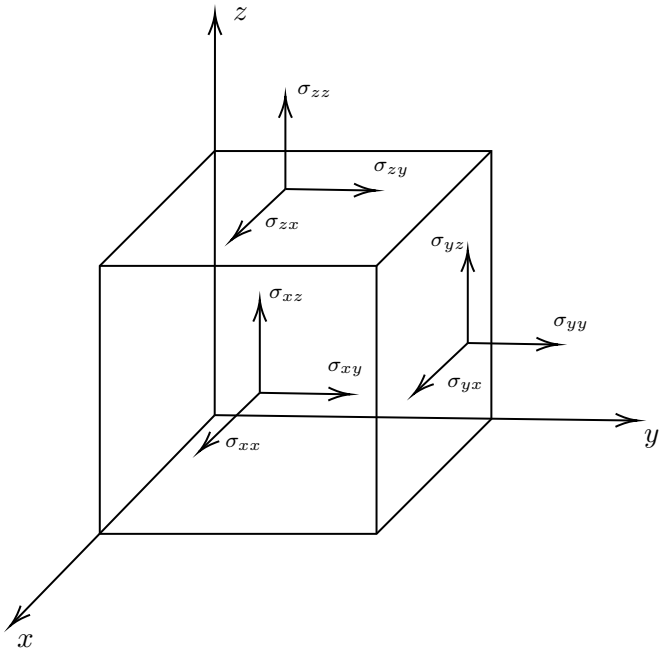


Abbildung 2.2: Spannungskomponenten bei einem infinitesimal kleinen Element des Körpers

Die dadurch erhaltenen Basisspannungsvektoren können in die jeweiligen drei Koordinatenrichtungen zerlegt und die resultierenden Spannungskomponenten in einem Tensor σ dargestellt werden:

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$

Der Spannungszustand wird in jedem Punkt eines Körpers und zu jedem Zeitpunkt durch die Einträge in diesem Spannungstensor beschrieben. Die verschiedenen Komponenten zeigen, wie ein Element in den unterschiedlichen Richtungen belastet wird. Der erste Index gibt dabei die Oberfläche und der zweite Index die Richtung der wirkenden Kraft an, wie in Abbildung 2.2 dargestellt. Die Werte auf der Hauptdiagonalen werden als *Normalspannungen*, die Werte auf der Nebendiagonalen als *Schubspannungen* und die Eigenwerte des Spannungstensors als *Hauptspannungen* bezeichnet.

Es kann gezeigt werden, dass am Rand eines Körpers mit einer wirkenden Oberflächenkraft k das Cauchy'sche Fundamentaltheorem (siehe [3])

$$k = \sigma^T n \tag{2.4}$$

erfüllt ist, wobei $n \in \mathbb{R}^3$ den Normalenvektor am jeweiligen Randpunkt bezeichnet. Da der Spannungstensor symmetrisch ist [3], wird anstelle des Tensors auch die folgende, im Rahmen dieser Arbeit verwendete Vektor-Notation (*Voigt-Notation*)

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{zx} \end{pmatrix} =: \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{pmatrix}$$

genutzt. Diese Voigt-Notation (insbesondere die Reihenfolge der Spannungskomponenten in dieser Vektorschreibweise) wird auch in der Simulationssoftware LS-Dyna genutzt, weshalb in dieser Arbeit durchweg die Art der Indizierung der Komponenten gewählt wurde.

Die Spannung wird in der Einheit Pascal 1 Pa $= 1\frac{\mathrm{N}}{\mathrm{m}^2}$, welche die Kraft in Newton (N) pro Fläche in Quadratmeter (m^2) angibt, gemessen. Soweit nicht anderweitig aufgeführt, werden die Spannungswerte in dieser Arbeit durchweg in Gigapascal (GPa) angegeben.

Um einen Spannungszustand σ durch einen skalaren Wert zu beschreiben, kann die von-Mises-Spannung betrachtet werden:

Definition 2.1.2. Für einen Spannungstensor

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$

ist die *von-Mises-Spannung* $\sigma_{v,M} \in \mathbb{R}$ gegeben durch

$$\sigma_{v,M} = \sqrt{\frac{1}{2}\left(\left(\sigma_{xx} - \sigma_{yy}\right)^2 + \left(\sigma_{yy} - \sigma_{zz}\right)^2 + \left(\sigma_{zz} - \sigma_{xx}\right)^2 + 6 \cdot \left(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2\right)\right)}. \tag{2.5}$$

2.1.3 Dehnungen

Durch die Spannungen wird ein spezifischer Verschiebungs- und Verzerrungszustand erzeugt, der für einen Punkt im Körper durch den im Folgenden vorgestellten Dehnungstensor beschrieben wird. Die Deformation eines Körpers kann auf zwei unterschiedliche Arten erfolgen: der Körper kann sich als Gesamtes ohne eine relative Änderung der Lage einzelner Punkte des Körpers bewegen, was als *Starrkörperbewegung* bezeichnet wird oder durch eine Veränderung der relativen Abstände der Körperpunkte untereinander. Letzteres wird als *Verzerrung* oder *Verformung* bezeichnet. Der Dehnungstensor dient als Maß für die Verzerrung eines Körpers. Die einzelnen Komponenten des Dehnungstensors können durch die partiellen Ableitungen der Verschiebung bezüglich der räumlichen Komponenten (siehe 2.2) definiert werden.

Die *Normaldehnungen* sind gegeben durch

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x}, \qquad \varepsilon_{yy} = \frac{\partial u_y}{\partial y}, \qquad \varepsilon_{zz} = \frac{\partial u_z}{\partial z}$$

und die *Schubdehnungen* durch

$$\varepsilon_{xy} = \varepsilon_{yx} = \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right), \quad \varepsilon_{yz} = \varepsilon_{zy} = \frac{1}{2}\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right), \quad \varepsilon_{xz} = \varepsilon_{zx} = \frac{1}{2}\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right).$$

Die jeweiligen Dehnungskomponenten können (wie auch die Spannungen) in einem ebenfalls symmetrischen Dehnungstensor (*linearisierter Verzerrungstensor*)

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{bmatrix}$$

dargestellt werden, wobei auch hier die Vektor-Notation

$$\begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{xy} \\ \varepsilon_{yz} \\ \varepsilon_{zx} \end{pmatrix} =: \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \end{pmatrix}$$

genutzt wird.

2.1.4 Gleichgewichtsbedingungen

Ein mechanisches System ist im Gleichgewicht, wenn äußere, auf den Körper wirkende Einflüsse (Kräfte) mit den dadurch im Körper entstehenden Spannungen im Gleichgewicht stehen. Dieser Zusammenhang wird durch die Bilanzgleichungen beschrieben und kann an einem beliebig kleinen Ausschnitt des Körpers hergeleitet werden (siehe [23]), wodurch sich die lokale Form (auch starke Form genannt) der *Impulsbilanz*

$$div(\sigma) + \rho b = \rho \ddot{u} \tag{2.6}$$

ergibt. Die Dichte des Körpers wird in (2.6) mit $\rho \in \mathbb{R}$ bezeichnet, $\ddot{u}$ ist die zweifache partielle Ableitung nach der Zeit t

$$\ddot{u} = \left(\frac{\partial^2 u_x(p,t)}{\partial t^2}, \frac{\partial^2 u_y(p,t)}{\partial t^2}, \frac{\partial^2 u_z(p,t)}{\partial t^2} \right)^T \tag{2.7}$$

und $div(\sigma)$ die Divergenz des Spannungstensors σ, definiert durch

$$div(\sigma) := \begin{pmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} \\ \frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} \\ \frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} \end{pmatrix}, \tag{2.8}$$

wobei die Spannung, wie in (2.1) beschrieben, als stetig differenzierbare Funktion der Zeit und eines räumlichen Punktes des Körpers betrachtet wird. Die partiellen Ableitungen $\frac{\partial \sigma_{ij}}{\partial x}, \frac{\partial \sigma_{ij}}{\partial y}, \frac{\partial \sigma_{ij}}{\partial z}$ für $i, j \in \{x, y, z\}$ bezeichnen die jeweiligen Ableitungen bezüglich der räumlichen Komponenten. Auf der linken Seite der Gleichung werden die Oberflächenkräfte ($div(\sigma)$) mit den vorgegebenen Volumenkräften $b = (b_x, b_y, b_z)^T$, die auf den gesamten Körper wirken (wie beispielsweise die Erdbeschleunigung) aufaddiert.
Dieses zeitabhängige, partielle Differentialgleichungssystem soll für ein Problem im Bereich der Strukturmechanik mit gegebenen Anfangs- und Randbedingungen gelöst werden, wobei (2.6) in jedem Punkt des Körpers erfüllt sein muss.

2.2 Finite-Elemente-Simulationen

Das Ziel einer Simulation ist die Beschreibung und Modellierung physikalischer Vorgänge unter bestimmten Annahmen und Vereinfachungen. Dies geschieht durch das Aufstellen und Lösen von partiellen Differentialgleichungen. In der Praxis sind diese

Differentialgleichungen oftmals nicht analytisch zu lösen, sondern werden durch numerische Näherungsverfahren bestimmt. Ein solches approximatives Verfahren, welches sich insbesondere zum Lösen von Problemstellungen in der Festkörpermechanik eignet, ist die Finite-Elemente-Methode (FEM), deren Idee auf der räumlichen und zeitlichen Diskretisierung des Problems basiert. Die Aufgabe der FEM ist in diesem Fall die Verformung einer Struktur (wie in 2.1.1 beschrieben) zu bestimmen, wobei die zugrunde liegende Bilanzgleichung in 2.1.4 erläutert wurde. Zur Bestimmung der approximativen Lösung wird bei der FEM nicht für das gesamte betrachtete Gebiet eine Lösung gesucht, sondern dieses in endlich viele Teilbereiche partitioniert und lokal näherungsweise Lösungen bestimmt.

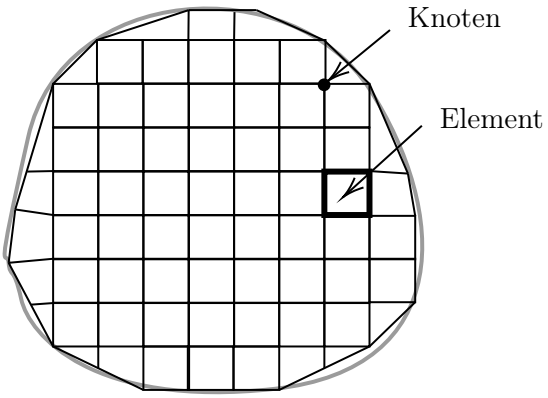


Abbildung 2.3: Diskretisierung eines Kontinuums in Finite Elemente

Bei strukturmechanischen Problemen wird hierfür der betrachtete Festkörper in endlich viele disjunkte Teilmengen, die finiten Elemente, aufgeteilt wie in Abbildung 2.3 dargestellt und die Verformung innerhalb der Elemente mittels vereinfachter sogenannter *Ansatzfunktionen* (Polynome eines bestimmten Grades) beschrieben. Diese Ansatzfunktionen sollen innerhalb eines jeweiligen Elements das Verhalten der gesuchten Funktion näherungsweise beschreiben und werden im restlichen Bereich des Körpers auf null gesetzt. Die Ansatzfunktionen sind bis auf einige variable Parameter festgelegt und müssen so gewählt werden, dass an den Übergängen zweier Elemente (*Knoten*) die gleichen Werte angenommen werden. Aus diesen sogenannten *Kompatibilitätsbedingungen* an den Knoten wird ein gesamtes, lineares Gleichungssystem aufgestellt und gelöst [52].
Die Diskretisierung des Körpers kann mit unterschiedlichen Element-Formulierungen (je nach Geometrie des Körpers) erfolgen. In Abbildung 2.4 sind Beispiele unterschiedlicher Schalen- und Volumenelemente, welche für die Diskretisierung von zwei- beziehungsweise dreidimensionalen Körpergeometrien verwendet werden können, zu sehen.

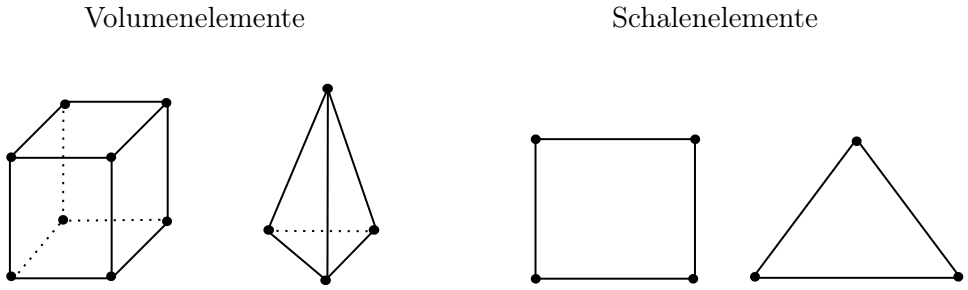


Abbildung 2.4: Volumen- und Schalenelemente in FE-Simulationen

Die Genauigkeit der durch die FEM erhaltenen Lösung ist von vielen Faktoren abhängig, wie beispielsweise der Wahl der Ansatzfunktionen, der Elementgröße oder der Wahl der Elementformulierungen.

2.2.1 Grundkonzept der FEM am Beispiel eines Stabs

Zur Veranschaulichung der grundlegenden Vorgehensweise der Finite-Elemente-Methode wird ein einfaches Beispiel angelehnt an Inhalten aus [44, 70] betrachtet. Als Beispiel dient ein Stab, wie in Abbildung 2.5 dargestellt, mit einer Ausgangslänge $L = 3$ m und einem konstanten Querschnitt $A = 1$ m^2. Das linke Ende des Stabs ist fest eingespannt und auf der rechten Seite des Stabs wird eine vorgegebene Kraft $f = 1000$ N angelegt. Als Materialmodell wird eine einfache linear elastische Beziehung der Form $\sigma = E \cdot \varepsilon$ mit $E = 210$ GPa angenommen.

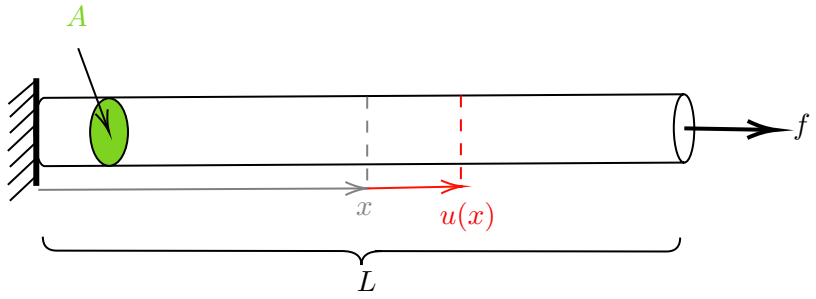


Abbildung 2.5: Skizze des betrachteten Stabs, welcher links am Rand festgehalten wird während rechts eine Kraft f aufgebracht wird

Wegen der ausschließlichen Betrachtung einer einzigen freien räumlichen Richtung (x-Richtung) werden für die Verschiebung sowie Dehnungs- und Spannungskomponenten nur die x-Komponenten betrachtet und daher der Einfachheit halber die entsprechenden Indizes weggelassen. Im Falle der Verschiebung beispielsweise wird vereinfacht u anstelle von u_x geschrieben.

1. Diskretisierung des Stabs

Der Stab wird, wie es in Abbildung 2.6 dargestellt ist, in die drei (eindimensionalen) Stabelemente e_1, e_2, e_3 mit jeweils einer Länge von $l = 1$ m unterteilt, woraus sich insgesamt vier Knoten n_0, n_1, n_2, n_3 ergeben. Die Knotenpunkte werden dementsprechend bei $x = 0, 1, 2, 3$ platziert. Da der Stab an jeder Stelle die gleiche Querschnittsfläche A hat und somit in jedem Punkt der Fläche der gleiche Spannungszustand herrscht, muss die Dicke des Stabs bei der hier betrachteten approximativen Modellierung geometrisch nicht abgebildet werden (siehe auch [70]) und es kann eine Diskretisierung in sogenannte eindimensionale Strukturelemente, welche lediglich Zug- oder Druckkräfte in eine Richtung aufnehmen können, vorgenommen werden.

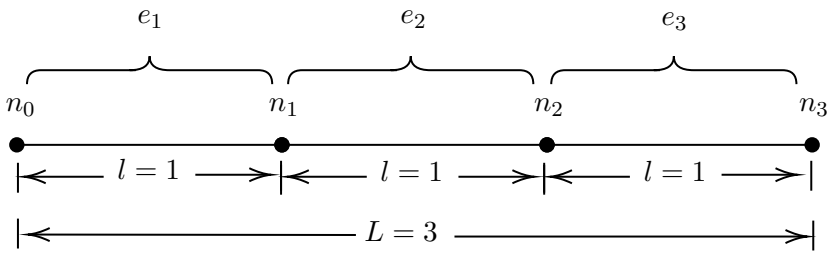


Abbildung 2.6: Diskretisierung des Stabs in die drei Stab-Elemente e_1, e_2, e_3

2. Wahl der Ansatzfunktionen

Für ein Element $e \in \{e_1, e_2, e_3\}$ der Länge l mit den Knotenpunkten i und j wird eine Ansatzfunktion gewählt, welche die Verschiebung $u(x)$ innerhalb des Elements beschreibt. Für das hier betrachtete Beispiel wird eine lineare Ansatzfunktion $u(x) = ax + b$ gewählt. Die Verschiebung $u(x)$ innerhalb des Elements kann durch eine lineare Kombination

$$u(x) = N_i(x)u_i + N_j(x)u_j \tag{2.9}$$

der Formfunktionen $N_i(x)$ und $N_j(x)$ sowie der Knotenverschiebungen u_i und u_j beschrieben werden. Die Formfunktionen $N_i(x)$ und $N_j(x)$ interpolieren dabei die Verschiebungen u_i und u_j an den Knotenpunkten des Elements. Nach den vorgegeben Interpolationsbedingungen müssen die Formfunktionen an den Knotenpunkten des Elements den Wert 1 für den entsprechenden Knoten und 0 für die jeweils anderen Knoten haben. Daraus ergeben sich, wenn ohne Beschränkung der Allgemeinheit der linke Randknoten i auf den

Ursprung des Koordinatensystems gelegt wird, die folgenden Bedingungen

$$N_i(0) = 1 \tag{2.10}$$
$$N_j(0) = 0 \tag{2.11}$$

am ersten (linken) Knoten des Elements und

$$N_j(l) = 1 \tag{2.12}$$
$$N_i(l) = 0 \tag{2.13}$$

am zweiten (rechten) Knoten des Elements (siehe auch Abbildung 2.7). Zusammen mit der Linearität ergeben sich die Formfunktionen

$$N_i(x) = \frac{l - x}{l} \tag{2.14}$$
$$N_j(x) = \frac{x}{l}, \tag{2.15}$$

welche für $l = 1$ durch

$$N_i(x) = 1 - x \text{ und } N_j = x \tag{2.16}$$

gegeben sind.

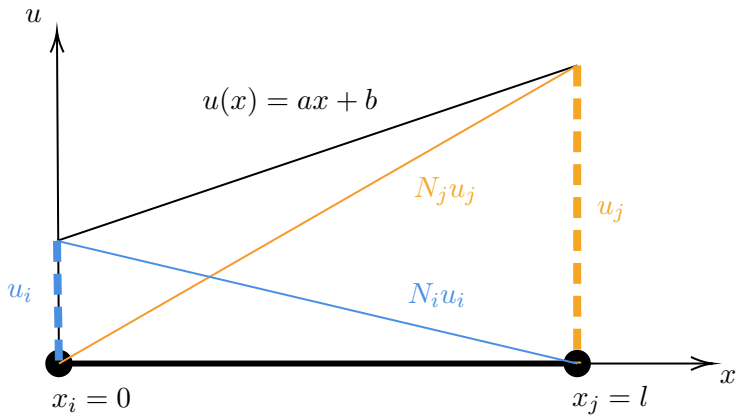


Abbildung 2.7: Ansatzfunktionen eines Stabelements

3. Bestimmen der Elementsteifigkeitsmatrizen

Für die Berechnung der Dehnungen und darauf basierend der Spannungen wird die Ableitung der durch die Ansatzfunktionen approximierten Verschiebung in (2.9) nach x

benötigt:

$$\varepsilon(x) = \frac{\partial u}{\partial x} = \frac{\partial N_i}{\partial x} u_i + \frac{\partial N_j}{\partial x} u_j = -\frac{1}{l} u_i + \frac{1}{l} u_i. \tag{2.17}$$

Basierend darauf können mittels des Materialmodells die Spannungen durch

$$\sigma(x) = E\varepsilon(x) = E\left(-\frac{1}{l} u_i + \frac{1}{l} u_i\right) \tag{2.18}$$

bestimmt werden. Die inneren Kräfte im Stabelement werden durch die Spannung σ und die Querschnittsfläche A bestimmt. Unter Berücksichtigung der Definition der Spannung als wirkende Kraft im Verhältnis zur Fläche (siehe Abschnitt 2.1.2) ergibt sich die innere Kraft im Stab durch

$$f = \sigma(x) \cdot A = E \cdot A\left(-\frac{1}{l} u_i + \frac{1}{l} u_i\right). \tag{2.19}$$

Betrachtet man das Kräftegleichgewicht an den beiden Knoten i und j eines Elements (mit den beiden dort wirkenden Kräften f_i und f_j), so ergeben sich die beiden Gleichungen

$$f_i = -f = -E \cdot A\left(-\frac{1}{l} u_i + \frac{1}{l} u_i\right) = \frac{EA}{l}(u_i - u_j) \tag{2.20}$$

$$f_j = f = E \cdot A\left(-\frac{1}{l} u_i + \frac{1}{l} u_i\right) = \frac{EA}{l}(u_j - u_i). \tag{2.21}$$

Die Gleichgewichtsbedingungen $f_i = -f$ und $f_j = f$ resultieren aus der Tatsache, dass sich die inneren Kräfte im Stabelement an den Knotenpunkten ausgleichen müssen. Die innere Kraft wird durch die Spannung und die Querschnittsfläche des Stabs bestimmt und zeigt in entgegengesetzte Richtungen an den beiden Knotenpunkten. Dies stellt sicher, dass das System im Gleichgewicht und die Summe der Kräfte an jedem Knoten null ist.

Die beiden Gleichungen in (2.20) und (2.21) für das Kräftegleichgewicht an den Knoten können äquivalent in der Matrixschreibweise

$$\begin{pmatrix} f_i \\ f_j \end{pmatrix} = \underbrace{\frac{EA}{l}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}}_{=:k_e} \begin{pmatrix} u_i \\ u_j \end{pmatrix} \tag{2.22}$$

geschrieben werden. Da die Länge jedes der einzelnen Stabelemente in dem hier betrachteten Beispiel $l = 1$ beträgt, bekommen wir zusammen mit $E = 210\ \text{GPa} = 210 \cdot 10^9 \frac{\text{N}}{\text{m}^2}$

und $A = 1\ \mathrm{m}^2$ die sogenannte *Elementsteifigkeitsmatrix*

$$k_e = \frac{EA}{l}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = \frac{210 \cdot 10^9 \left[\frac{\mathrm{N}}{\mathrm{m}^2}\right] \cdot 1[\mathrm{m}^2]}{1[\mathrm{m}]}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} = 210 \cdot 10^9 \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \left[\frac{N}{m}\right] \tag{2.23}$$

für ein einzelnes Stabelement.

Die Elementsteifigkeitsmatrix k_e beschreibt im Allgemeinen die Beziehung zwischen den Knotenkräften und den Knotenverschiebungen innerhalb eines einzelnen Finite-Elemente-Elements. Sie beinhaltet die Materialeigenschaften, die Geometrie des Elements und die Formfunktionen, welche die Verteilung der Verschiebungen innerhalb des Elements interpolieren. Für Flächen- und Volumenelemente ist die Berechnung der Steifigkeitsmatrix komplexer, da die Spannungen und Dehnungen innerhalb des Elements variieren können. In diesen Fällen müssen die Beiträge der Formfunktionen und ihrer Ableitungen über das gesamte Volumen (beziehungsweise die Fläche) des Elements integriert werden, um die Steifigkeitsmatrix zu bestimmen. Üblicherweise passiert dies approximativ mittels numerischer Integrationsmethoden wie der Gauß-Quadratur.

4. Zusammenfügen der Elementmatrizen zu einer globalen Steifigkeitsmatrix

Während im vorherigen Abschnitt lediglich ein unabhängiges, einzelnes Element betrachtet wird, müssen in einem folgenden Schritt die einzelnen Elemente wieder zu einem Gesamtsystem zusammengefügt werden, um eine Lösung für die gesamte Geometrie zu erhalten. Dies passiert durch das Aufstellen der sogenannten *globalen Steifigkeitsmatrix*. Die globale Steifigkeitsmatrix K wird durch das Zusammensetzen der einzelnen Elementsteifigkeitsmatrizen k_e gebildet. Die Größe der globalen Steifigkeitsmatrix hängt von der Anzahl an Freiheitsgraden ab. In diesem Beispiel sind es die Knotenverschiebungen von 4 Knoten in lediglich der x-Richtung, womit K durch eine 4×4 Matrix gegeben ist.

Durch das Aufstellen der globalen Steifigkeitsmatrix K für das im nächsten Schritt aufzustellende Gesamtgleichungssystems (siehe (2.25)) soll das Kräftegleichgewicht an den Knotenpunkten erfüllt und die sogenannte *Verschiebungskompatibilität* gewährleistet werden. Diese Bedingung stellt sicher, dass für zwei oder mehr Elemente, welche an einem Knotenpunkt zusammentreffen, die berechneten Verschiebungen an diesem Knotenpunkt stetig und für alle beteiligten Elemente die lokalen Verschiebungen der benachbarten Elemente an diesen Knotenpunkten gleich sind. Andernfalls können an den Knotenpunkten Löcher oder Überlappungen im kontinuierlichen Festkörper entstehen. An den Stellen, an welchen zwei Elemente durch einen Knoten verbunden sind, werden hierfür die Einträge der jeweiligen Elementsteifigkeitsmatrizen für die globale Steifigkeitsmatrix addiert. Sei

k_{ij}^e der Matrixeintrag der i-ten Zeile und j-ten Spalte der Elementsteifigkeitsmatrix von Element e, dann ergibt sich die globale Steifigkeitsmatrix für die drei in diesem Beispiel betrachteten Stabelemente e_1, e_2 und e_3 durch

$$K = \begin{pmatrix} k_{11}^{e_1} & k_{12}^{e_1} & 0 & 0 \\ k_{21}^{e_1} & k_{22}^{e_1} + k_{11}^{e_2} & k_{12}^{e_2} & 0 \\ 0 & k_{21}^{e_2} & k_{22}^{e_2} + k_{11}^{e_3} & k_{12}^{e_3} \\ 0 & 0 & k_{21}^{e_3} & k_{22}^{e_3} \end{pmatrix} = 210 \cdot 10^9 \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \tag{2.24}$$

Für eine ausführliche Herleitung und die Beschreibung eines allgemeinen Verfahrens zur Aufstellung der globalen Steifigkeitsmatrix aus den einzelnen Elementsteifigkeitsmatrizen wird auf [70] verwiesen.

Schritt 5: Aufstellen und Lösen des Gleichungssystems mit den gegebenen Randbedingungen

Basierend auf der globalen Steifigkeitsmatrix aus dem vorherigen Abschnitt wird nun das zu lösende Gesamtgleichungssystem

$$Ku = f \tag{2.25}$$

aufgestellt, wobei durch $u = (u_0, u_1, u_2, u_3)^T$ die Verschiebungen und $f = (f_0, f_1, f_2, f_3)^T$ die äußeren Kräfte an den Knoten n_0, n_1, n_2, n_3 bezeichnet werden. Für eine genaue Herleitung dieses allgemein gültigen Gleichungssystems aus der zu lösenden Differentialgleichung wird auch hier auf [70] verwiesen. Wird die am rechten Randknoten aufgebrachte Kraft $f_3 = 1000$ N sowie die Verschiebungsrandbedingung am linken Randknoten $u_0 = 0$ eingesetzt, welche sich aus dem Festhalten des Stabs auf der linken Seite ergibt, so erhält man das folgende Gleichungssystem:

$$210 \cdot 10^9 \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_0 \\ 0 \\ 0 \\ 1000 \end{pmatrix}. \tag{2.26}$$

Durch das Lösen dieses Gleichungssystems bekommt man die Knotenverschiebungen $u_1 = 4.7617 \cdot 10^{-9}$ m, $u_2 = 9.52381 \cdot 10^{-9}$ m, $u_3 = 1.42857 \cdot 10^{-8}$ m sowie die am Randknoten n_0 wirkende Kraft $f_0 = -1000$ N.

2.3 Maschinelles Lernen

Der Bereich des maschinellen Lernens (ML) als Teilgebiet der Künstlichen Intelligenz umfasst das Lernen und Lösen von Aufgabenstellungen anhand von Daten (siehe auch [27]). Die Methoden des maschinellen Lernens kommen inzwischen bei vielfältigen Aufgaben wie beispielsweise Bild-, Sprach- oder Texterkennung zum Einsatz. Grundlegend können drei Teilgebiete des maschinellen Lernens voneinander abgegrenzt werden: Überwachtes Lernen, unüberwachtes und verstärkendes Lernen. Bei Ersterem lernen die ML-Modelle anhand von Datensätzen, welche eine korrekte Lösung der fest vorgegebenen, zu erlernenden Aufgabe enthalten, während diese beim unüberwachten Lernen nicht vorhanden sind. Beim verstärkenden Lernen wird situationsabhängig durch Rückmeldung anhand von Belohnung beziehungsweise Bestrafung ein bestimmtes Verhalten erlernt.

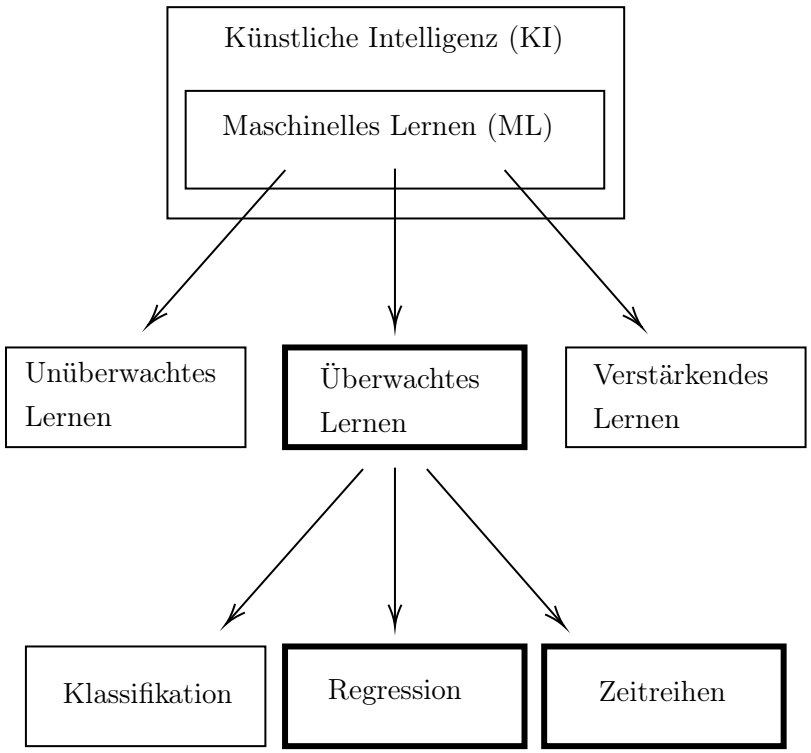


Abbildung 2.8: Übersicht der Teilbereiche des maschinellen Lernens

Bei Aufgabenstellungen des überwachten Lernens gibt es eine Menge an Eingabe, welche einen Einfluss auf zugehörige Ausgabegrößen haben. Das Ziel ist hierbei den Zusammenhang dieser Ein- und Ausgaben zu beschreiben und die Eingabegrößen zur Vorhersage der Ausgabegrößen zu benutzen. Abhängig der betrachteten Daten kann der Bereich des

überwachten Lernens wiederum in die Teilbereiche der Klassifikation, Regression und Zeitreihenanalyse eingeteilt werden, je nachdem, ob es sich um diskrete Werte, reellwertige oder sequentielle Daten handelt. Abbildung 2.8 gibt einen Gesamtüberblick über die unterschiedlichen ML-Teilbereiche.

Die Aufgabenstellungen dieser Arbeit befinden sich im Bereich der Regression und Zeitreihenanalyse, wie in Kapitel 3 näher erläutert wird. Der Fokus der beschriebenen Modelle und Methoden im Folgenden liegt daher auf diesem Gebiet. Die Erläuterungen in diesem Abschnitt basieren auf Auszügen aus [27] und [32]. Weitergehende Erklärungen sind dort zu finden.

2.3.1 Lineare Regression und nichtlineare Erweiterungen

Das Ziel bei Regressionsproblemen ist es gegeben einer Menge an n Datenpaaren $((x^1, y^1), ..., (y^n, x^n))$ mit jeweils p Eingaben $x^j = (x_1^j, ..., x_p^j)^T$ und k Ausgaben $y^j = (y_1^j, ..., y_k^j)^T$ den Zusammenhang der *abhängigen Variable* y^j von der *unabhängigen Variable* x^j zu beschreiben. Im Bereich der linearen Regression soll dieser Zusammenhang durch eine lineare Funktion $H : \mathbb{R}^p \to \mathbb{R}^k, x \mapsto Hx$ approximiert werden, welche in der Regel die quadratische Abweichung

$$(Y - XH)^T(Y - XH) \tag{2.27}$$

mit

$$X \coloneqq \begin{pmatrix} x_1^1 & \cdots & x_p^1 \\ \vdots & & \vdots \\ x_1^n & \cdots & x_p^n \end{pmatrix}, \; Y \coloneqq \begin{pmatrix} y_1^1 & \cdots & y_k^1 \\ \vdots & & \vdots \\ y_1^n & \cdots & y_k^n \end{pmatrix} \tag{2.28}$$

minimiert. Wird davon ausgegangen, dass die Spalten der Matrix X linear unabhängig sind (anderenfalls wäre eine Spalte eine Linearkombination einer anderen und somit redundant), so existiert $(X^T X)^{-1}$ und es kann gezeigt werden, dass die eindeutige Lösung für dieses Minimierungsproblem durch

$$\hat{H} \coloneqq (X^T X)^{-1} X^T Y \tag{2.29}$$

gegeben ist (siehe [32]).

Da in vielen Fällen die Beziehung zwischen den Eingaben X und Ausgaben Y nicht oder nur unzureichend durch eine lineare Funktion approximiert werden kann, können die Eingaben $x^j, j = 1, ..., n$ anhand von nichtlinearen Funktionen $f_1, ..., f_l$ mit $f_l : \mathbb{R}^p \to \mathbb{R}$

transformiert und diese transformierten Eingaben

$$V := \begin{pmatrix} f_1(x_1^1) & \cdots & f_l(x_1^1) & \cdots & f_1(x_p^1) & \cdots & f_l(x_p^1) \\ \vdots & & & & & & \vdots \\ f_1(x_1^n) & \cdots & f_l(x_1^n) & \cdots & f_1(x_p^n) & \cdots & f_l(x_p^n) \end{pmatrix} \tag{2.30}$$

durch ein lineares Modell

$$\hat{Y} = VP \tag{2.31}$$

mit $P : \mathbb{R}^{l \cdot p} \to \mathbb{R}^k$ approximiert werden. Eine mögliche Wahl für diese Transformationen sind beispielsweise Polynome oder Splines.

2.3.2 Baumbasierte ML-Verfahren

Im Folgenden wird die Funktionsweise von Entscheidungsbäumen und Random Forests in Hinblick auf die Anwendung bei Regressionsproblemen beschrieben, auch wenn baumbasierte Verfahren häufig ebenfalls für Klassifikationsaufgaben verwendet werden.

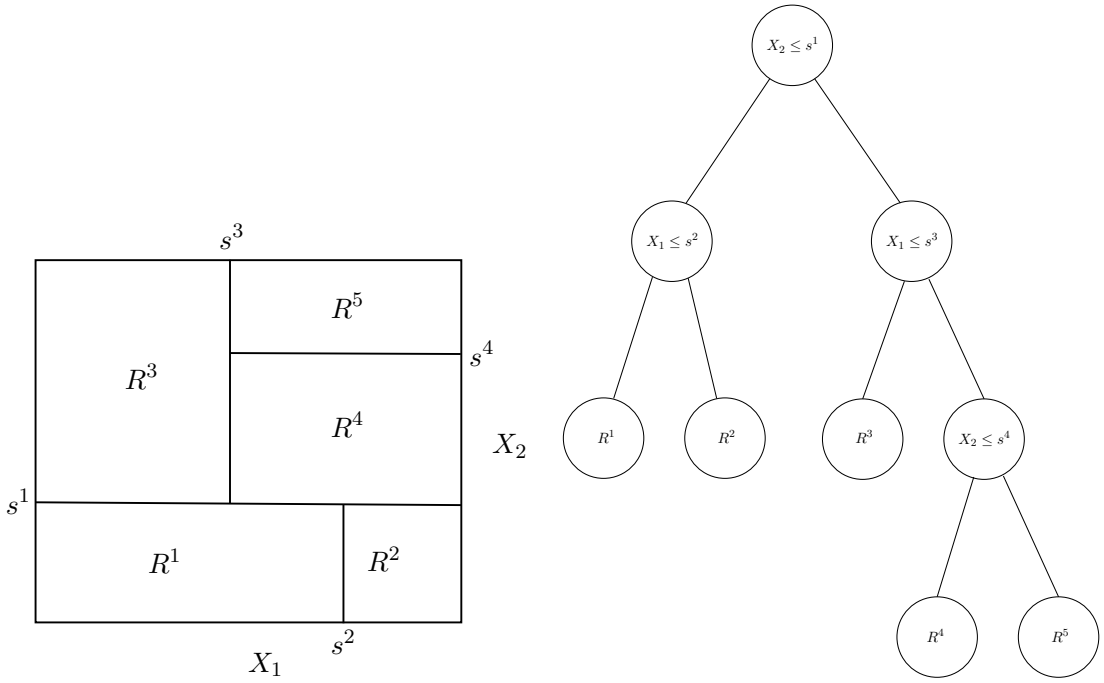
Entscheidungsbäume

Entscheidungsbäume (siehe [13]) gehören im Gegensatz zu linearen Regressionsmodellen oder neuronalen Netzen zu den sogenannten parameterfreien Modellen, die sich dadurch auszeichnen, dass die Anzahl der Parameter, welche auf Basis des Trainingsdatensatzes angepasst werden, nicht bereits vor dem Training festgelegt ist. Bei Entscheidungsbäumen wird der Eingaberaum $R \subset \mathbb{R}^p$ durch rekursive, binäre Aufteilungen in Teil-Regionen $R^1, ..., R^m$ partitioniert. Diese Partitionierung kann graphentheoretisch durch einen Baum dargestellt werden, wobei jedem Blatt oder äquivalent jeder Teilregion R^r, $r = 1, ..., m$ ein Ausgabewert beziehungsweise eine Menge an Ausgaben $c^r \in \mathbb{R}^k$ (mit k Ausgabewerten) für die Vorhersage des Modells zugeordnet ist (siehe Abbildung 2.9).
Die Antwort des Modells bei einer Eingabe x ergibt sich anschließend durch die Zuordnung eines Datenpunktes zu dem jeweiligen Teilraum:

$$f(x) = \sum_{r=1}^{m} c^r \cdot I(x \in R_r) \tag{2.32}$$

mit der Indikatorfunktion

$$I(x \in R_r) = \begin{cases} 1 & \text{falls } x \in R_r \\ 0 & \text{sonst.} \end{cases} \tag{2.33}$$



(a) Partitionierung des Eingaberaumes (b) Darstellung als Baum

Abbildung 2.9: Beispiel einer rekursiven, binären Partitionierung des Eingaberaumes in 5 Teilregionen für zwei Eingabewerte ($p = 2$) mit den Split-Punkten $s^1, ..., s^4$

Um solch eine Partition zu finden, kann der CART Trainingsalgorithmus (siehe auch [32]) genutzt werden. Dabei wird nach dem Greedy-Prinzip ausgehend von der Wurzel des Baumes an jedem Knoten nach jener binären Aufteilung (*Split*) gesucht, welche die Fehlerfunktion gegeben der aktuellen Situation (Daten und bisherige Aufteilung) minimiert. Eine Aufteilung ist dabei durch den *Split-Punkt* s, an dem der binäre Split erfolgt und der *Split-Variable* $j \in \{1, ..., p\}$ sowie der Antwort-Konstanten c^r, die angeben welche Vorhersage für die Datenpunkte an diesem Knoten ausgegeben wird, bestimmt. Die Split-Variable j und der Split-Punkt s können daher als optimale Lösung des Minimierungsproblems

$$\min_{s,j} \left[\min_{c^1} \sum_{x^i \in R^1(j,s)} (y^i - c^1)^2 + \min_{c^2} \sum_{x^i \in R^2(j,s)} (y^i - c^2)^2 \right] \tag{2.34}$$

mit $R^1(j,s) = \{x : x_j \leq s\}$ und $R^2(j,s) = \{x : x_j > s\}$ bestimmt werden. Für einen beliebig festen Split-Punkt s sowie eine Split-Variable j wird hierbei der mittlere

quadratische Fehler für

$$\tilde{c}_1 = \frac{1}{|R^1(j,s)|} \sum_{x^i \in R^1(j,s)} y^i \text{ und } \tilde{c}_2 = \frac{1}{|R^1(j,s)|} \sum_{x^i \in R^2(j,s)} y^i \tag{2.35}$$

minimal (siehe [32]).
Da Entscheidungsbäume sich den Trainingsdaten exakt anpassen können, was trivialerweise der Fall ist, wenn jedes Teilgebiet der Partition genau einen Trainingsdatenpunkt enthält, müssen bei Entscheidungsbäumen Regularisierungsmethoden genutzt werden, um die Anzahl der Freiheitsgrade einzuschränken und eine Überanpassung zu vermeiden. Zu diesen Regularisierungsmethoden gehören beispielsweise das Beschränken der Tiefe der Bäume oder eine festgelegte Mindestanzahl an Datenpunkten, die einem Blatt beim Training zugeordnet werden.

Random Forests

Random Forests gehören zu den sogenannten Ensemble-Methoden, bei denen eine Gruppe von Modellen trainiert wird und sich die Vorhersage aus der Kombination dieser Menge an Modellen ergibt. Random Forests bestehen aus mehreren Entscheidungsbäumen, welche jeweils anhand von zufällig gewählten Teilmengen des Datensatzes trainiert werden. Die Ausgabe des Modells wird anschließend durch die Antwort der einzelnen, unterschiedlichen Entscheidungsbäume beispielsweise durch den Mittelwert bestimmt.

2.3.3 Künstliche neuronale Netze

Künstliche neuronale Netze (NN) sind ML-Modelle, welche von den Netzwerken der Neuronen in biologischen Gehirnen inspiriert sind. Sie sind vielseitig einsetzbar und können auch bei Regressionsproblemen oder zur Verarbeitung von sequentiellen Daten verwendet werden. Künstliche neuronale Netze bestehen aus Schichten von Neuronen, die miteinander verbunden sind. Den Verbindungen zwischen den Neuronen der einzelnen Schichten werden Gewichte zugeordnet, welche die trainierbaren Parameter des ML-Modells darstellen und während des Trainingsprozesses angepasst werden. Beginnend bei einer Eingabeschicht mit einem Neuron für jede Eingabegröße des Problems werden die Werte bei einem *Feed Forward neuronalen Netz* (FFNN) Schicht für Schicht durch die Verbindungen zu nachfolgenden Neuronen bis zur Ausgabeschicht mit je einem Neuron für jeden Ausgabewert geleitet. Hierbei ergibt sich der Ausgabewert z eines Neurons durch die gewichtete Summe $\sum_{j=1}^{l} h_j w_j + b$ der Ausgaben $h_1, ..., h_l$ der verbundenen Neuronen der vorherigen Schicht mit Aufaddieren eines zusätzlichen Bias-Terms b und anschließender

Anwendung einer sogenannten *Aktivierungsfunktion* $g : \mathbb{R} \to \mathbb{R}$ (siehe Abbildung 2.11):

$$z = g(\sum_{j=1}^{l} h_j w_j + b). \tag{2.36}$$

In dieser Arbeit verwendete Aktivierungsfunktionen sind beispielsweise die tanh (tangens hyperbolicus), sigmoid (logistische sigmoid) und die ReLU (Rectified Linear Unit) Aktivierungsfunktion (siehe Abbildung 2.10):

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{2.37}$$

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{2.38}$$

$$\text{ReLU}(x) = \begin{cases} 0 & \text{falls } x \leq 0 \\ x & \text{falls } x > 0. \end{cases} \tag{2.39}$$

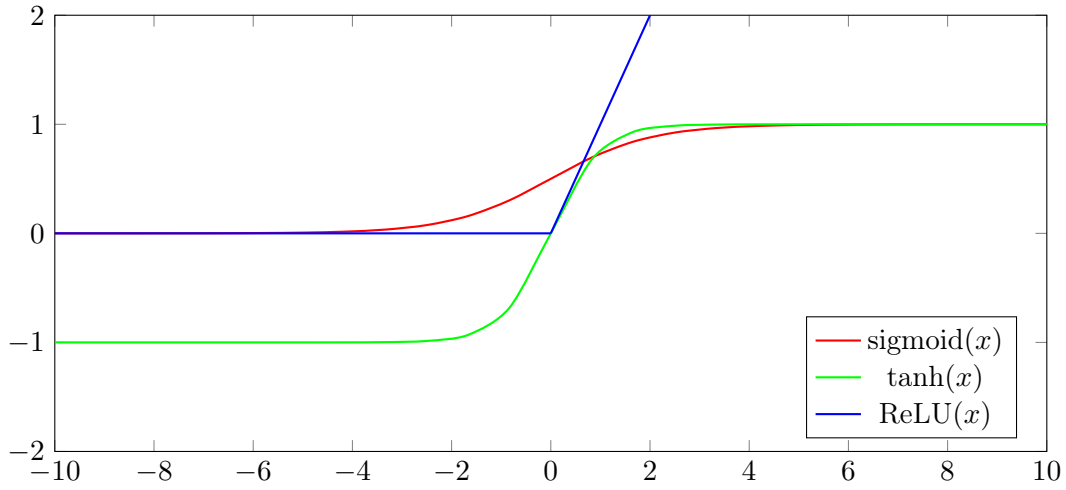


Abbildung 2.10: Beispiele für Aktivierungsfunktionen

Die trainierbaren Parameter $\theta \in \mathbb{R}^m$ des Netzes (Gewichte und Bias-Werte) werden im Laufe des Trainings angepasst, sodass sich die vorhergesagten Ausgaben $\hat{y}_i^j$ des Modells den korrekten Ausgaben y_i^j für $j = 1, ..., n$, $i = 1, ..., k$ des Trainingsdatensatzes mit n Datenpaaren und k Ausgabegrößen annähern.

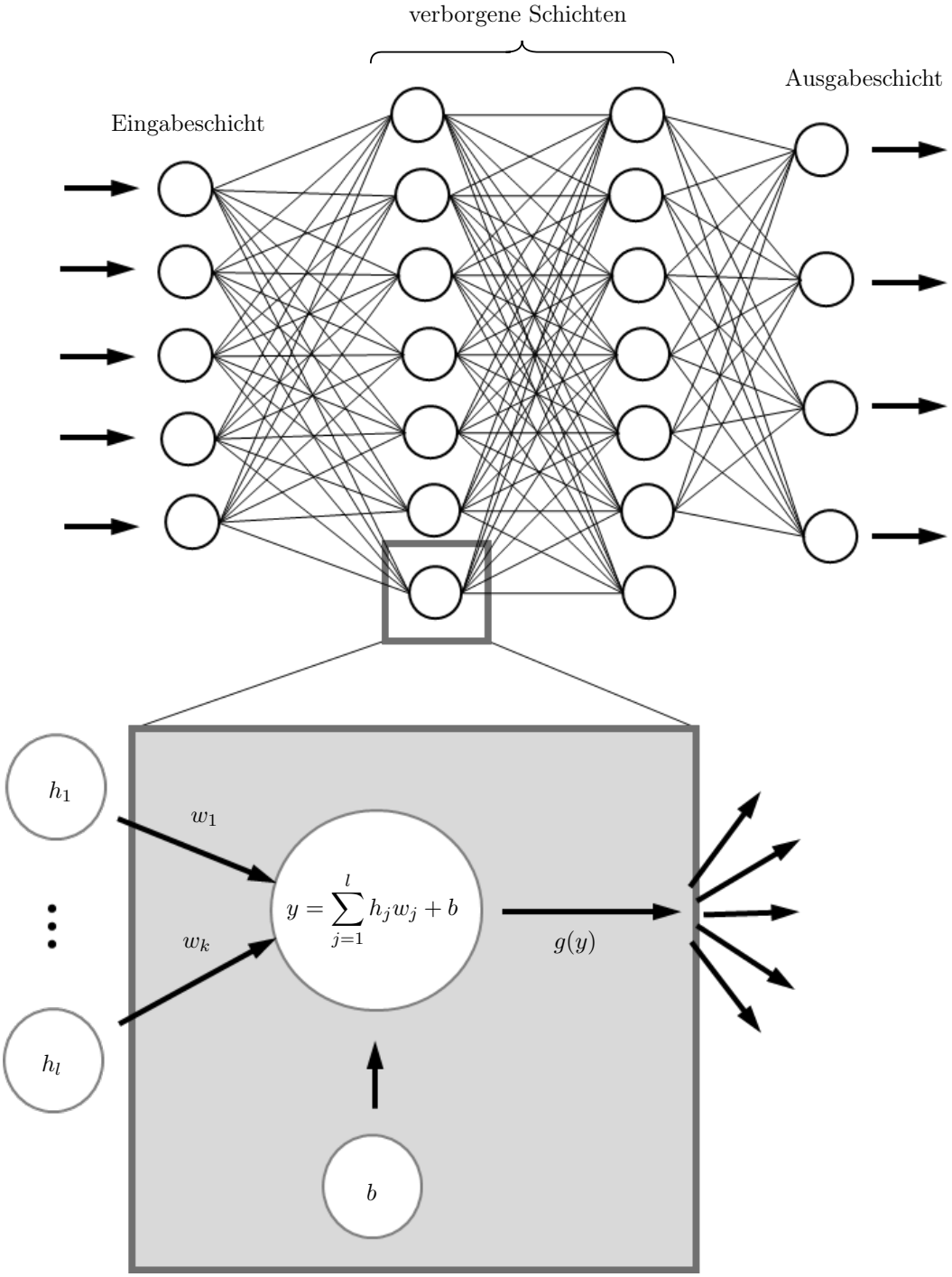


Abbildung 2.11: Aufbau eines künstlichen neuronalen Netzes

Die Abweichung zwischen der Vorhersage des Modells und dem korrekten Wert wird mittels einer Fehlerfunktion $F(\theta) \coloneqq F(\hat{y}(\theta), y)$ gemessen. Häufig wird hierbei die mittlere quadratische Abweichung (MSE) oder die mittlere absolute Abweichung (MAE) betrachtet, welche sich wie folgt berechnen:

$$\text{MSE}(\hat{y}, y) = \frac{1}{n \cdot k} \sum_{j=1}^{n} \sum_{i=1}^{k} (y_i^j - \hat{y}_i^j)^2 \tag{2.40}$$

$$\text{MAE}(\hat{y}, y) = \frac{1}{n \cdot k} \sum_{j=1}^{n} \sum_{i=1}^{k} |y_i^j - \hat{y}_i^j|. \tag{2.41}$$

Dieser Fehler $F(\theta)$ wird im Laufe des Trainings durch die schrittweise Anpassung der Modellparameter θ mithilfe eines Gradientenabstiegsverfahrens minimiert. Hierfür wird mittels *Backpropagation* der Gradient $\nabla F(\theta) = (\frac{\partial}{\partial \theta_1} F(\theta), ..., \frac{\partial}{\partial \theta_m} F(\theta))^T$, welcher sich aus den partiellen Ableitungen $\frac{\partial}{\partial \theta_j} F(\theta)$ der einzelnen Modellparameter bezüglich der Fehlerfunktion ergibt, bestimmt. Zuerst werden in einem Vorwärtsdurchlauf Schicht für Schicht die Ausgabewerte der einzelnen Neuronen berechnet und anschließend in einem Rückwärtsdurchlauf schrittweise die partiellen Ableitungen mithilfe der Kettenregel bestimmt. Im einfachsten Fall kann mit dem auf diese Weise berechneten Gradienten die Anpassung der Gewichte durch

$$\theta = \theta - \eta \cdot \nabla F(\theta) \tag{2.42}$$

mit einer *Lernrate* η erfolgen. Zur Beschleunigung des Trainings kann die Anwendung des Gradienten in (2.42) angepasst werden, beispielsweise durch die Berücksichtigung der Gradienten von vorherigen Schritten, wie es beim *Adam* Optimierer (Adaptive Moment Estimation) aus [43] der Fall ist. Adam nutzt für die Anwendung des Gradienten in einer Iteration t den Durchschnitt der Gradienten sowie den Durchschnitt der quadrierten Gradienten aus vorherigen Iterationen jeweils unter exponentiellem Zerfall:

$$\theta^{t+1} = \theta^t + u^t \tag{2.43}$$

mit

$$u_i^t = \frac{\eta \cdot \hat{m}_i^t}{\sqrt{\hat{s}_i^t + \epsilon}} \quad i = 1, ..., m \tag{2.44}$$

$$\hat{m}^t = (1 - (\beta_1)^t) \cdot m^t \tag{2.45}$$

$$\hat{s}^t = (1 - (\beta_2)^t) \cdot s^t \tag{2.46}$$

$$m^t = \beta_1 \cdot m^{t-1} - (1 - \beta_1) \cdot \theta^t \tag{2.47}$$

$$s^t = \beta_2 \cdot s^{t-1} + (1 - \beta_2) \cdot (\theta^t)^2. \tag{2.48}$$

Wird bei jedem Schritt im Gradientenverfahren der gesamte Datensatz betrachtet, so wird von einem *Batch-Gradientenverfahren* gesprochen. Im Falle eines einzelnen zufällig gewählten Datenpunktes ist von einem *stochastischen Gradientenverfahren* die Rede. Üblicherweise wird jedoch das *Mini-Batch-Gradientenverfahren* genutzt, bei dem die Gradienten auf Teilmengen einer vorgegebener Größe (*Batch-Größe*) bestimmt werden. Innerhalb einer Iteration des Trainings, genannt *Epoche*, wird der gesamte Datensatz in diese Batch-Teilmengen partitioniert und wiederholt der Gradient bezüglich dieser Teilmenge bestimmt.

Rekurrente neuronale Netze

Im Bereich der Materialmodelle werden unter anderem zeitliche Verläufe von Dehnungen und Spannungen betrachtet und daher ML-Modelle benötigt, welche sequentielle Daten berücksichtigen können. Für die Verarbeitung sequentieller Daten können *rekurrente neuronale Netze* (RNN) [62] genutzt werden, welche im Unterschied zu Feed Forward neuronalen Netzen auch rückwärts gerichtete Verbindungen beinhalten. Betrachtet man in einem rekurrenten neuronalen Netz ein einfaches Neuron, auch *(Gedächtnis-)Zelle* genannt, so bekommt diese Zelle zu jedem Zeitpunkt in der Sequenz eine Eingabe x^t und zusätzlich die eigene Ausgabe y^{t-1} aus dem vorhergegangenen Zeitschritt. Die Ausgabe y^t für den aktuellen Schritt ergibt sich dann aus einer gewichteten Summe der beiden Eingaben (Eingabe aktueller Zeitschritt und Ausgabe vorheriger Zeitschritt) mit den jeweiligen Gewichten w_x, w_y und anschließender Anwendung einer Aktivierungsfunktion g:

$$y^t = g(w_x x^t + w_y y^{t-1} + b). \tag{2.49}$$

Insbesondere hat ein rekurrentes neuronales Netz zwei Arten von Gewichten: die Gewichte für die Ausgabe des vorherigen Zeitschritts und die Gewichte für die Eingabe des aktuellen Zeitschrittes. Die aktuelle Ausgabe y^t eines Neurons, welche als Wert für den nächsten

Zeitschritt gespeichert wird, kann auch als Zustand h^t des Neurons bezeichnet werden. Für eine einfache RNN-Zelle, wie sie hier vorgestellt wird (siehe auch Abbildung 2.12), stimmt der Zustand mit der Ausgabe überein, was jedoch bei anderen Arten von Gedächtniszellen nicht zwingend der Fall sein muss.

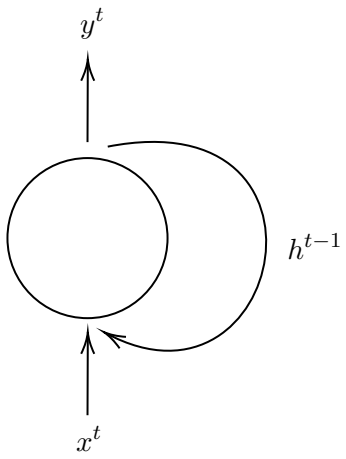


Abbildung 2.12: Skizze eines rekurrenten Neurons

Für die Betrachtung von längeren Sequenzen haben sich insbesondere LSTM und GRU Zellen etabliert, welche sich im Vergleich zu den einfachen RNN-Modellen in der Berechnung des Zustandes und der Ausgabe eines Neurons unterscheiden.

LSTM-Zellen *Long-Short-Term-Memory* (LSTM) Zellen wurden erstmals in [35] eingeführt und haben im Gegensatz zu einfachen RNN-Zellen je Neuron zwei Zustandsvariablen h^t und c^t, welche als Kurz- beziehungsweise Langzeitgedächtnis interpretiert werden können. Die Aktualisierung dieser beiden Zustände sowie die Berechnung der Ausgabe y^t basierend auf einer Eingabe x^t und der Zustände h^{t-1}, c^{t-1} des vorherigen Zeitschrittes ergibt sich wie folgt:

$$y^t = h^t = \underbrace{\text{sigmoid}(W_{xo}x^t + W_{yo}h^{t-1} + b_o)}_{=:o^t} \cdot \tanh(c^t) \tag{2.50}$$

$$c^t = \underbrace{\text{sigmoid}(W_{xf}x^t + W_{yf}h^{t-1} + b_f)}_{=:f^t} \cdot c^{t-1} \tag{2.51}$$

$$+ \underbrace{\text{sigmoid}(W_{xi}x^t + W_{yi}h^{t-1} + b_i)}_{=:i^t} \cdot \underbrace{\tanh(W_{xg}x^t + W_{yg}h^{t-1} + b_g)}_{=:g^t}. \tag{2.52}$$

Die Eingabe x^t und der Zustand h^{t-1} werden zunächst an vier einzelne Schichten als Eingabe weitergegeben, wobei sich die Ausgaben i^t, f^t, o^t, g^t dieser Schichten als gewichtete Summe wie in (2.49) mit Anwendung der entsprechenden Aktivierungsfunktion ergibt. Die Ausgaben $f^t, i^t, o^t \in [0, 1]$ dienen dazu zu kontrollieren, wie viel Informationen des Zustands c^t (reguliert durch f^t) sowie der Ausgabe g^t (reguliert durch i^t) mit in den aktuellen Zustand c^t einfließen und welcher Anteil dieses Zustands (reguliert durch o^t) an die Gesamtausgabe y^t weitergegeben wird. Diese drei Bereiche der LSTM-Zelle werden daher auch als *Gates* (*Forget-Gate*, *Input-Gate* und *Output-Gate*) und die Ausgaben f^t, i^t, o^t als *Gate-Controller* bezeichnet.

GRU-Zellen *Gated-Recurrent-Unit* (GRU) Zellen [17] haben nur je eine Zustandsvariable h^t, welche wie bei einfachen RNN-Zellen vollständig als Ausgabe y^t übernommen wird. Die Eingabe x^t und der Zustand h^{t-1} werden an drei Schichten mit den Ausgaben r^t, z^t und g^t weitergegeben, wobei z^t hier sowohl den Einfluss des vorherigen Zustands h^{t-1} als auch den Einfluss von g^t auf die Gesamtausgabe y^t kontrolliert. Als zusätzlicher Gate-Controller reguliert r^t die Weitergabe des vorherigen Zustands an die finale Schicht mit der Ausgabe g^t (siehe (2.53)). Die Aktualisierung des Zustands h^t basierend auf der aktuellen Eingabe x^t und dem vorherigen Zustandes h^{t-1} erfolgt zusammengefasst durch:

$$y^t = h^t = z^t \cdot h^{t-1} + (1 - z^t) \cdot \underbrace{\tanh(W_{xg}x^t + W_{yg}r^t \cdot h^{t-1} + b_g)}_{=:g^t} \tag{2.53}$$

mit

$$z^t := \text{sigmoid}(W_{xz}x^t + W_{yz}h^{t-1} + b_z) \tag{2.54}$$

$$r^t := \text{sigmoid}(W_{xr}x^t + W_{yr}h^{t-1} + b_r). \tag{2.55}$$

LMSC-Modelle *Linearized Minimal State Cell* (LMSC) Modelle wurden in [11] als Alternative zu konventionellen rekurrenten Zellen für die Anwendung in der Materialmodellierung vorgestellt mit dem Ziel die Anzahl der Zustände eines rekurrenten neuronalen Netzes variabel und unabhängig der Anzahl an Neuronen wählen zu können. Da klassische Materialmodelle mit einer geringen Anzahl an zwischengespeicherten Zustands-Werten auskommen, kann so die Gesamtanzahl an trainierbaren Parametern des neuronalen Netzes deutlich reduziert werden.
Die Eingabeschicht des Modells bekommt zu einem gegebenen Zeitschritt die Eingabe x^t sowie den Zustandsvektor $h^{t-1} \in \mathbb{R}^s$ von einer vordefinierten und frei wählbaren Größe

$s \in \mathbb{N}_{\geq 1}$. Die beiden Eingaben $I^t := (x^t, h^{t-1})^T$ werden zunächst durch m quadratische Schichten geschleust:

$$q_j^t = \tanh(W_j^{q1} I^t + b_j^{q1}) \cdot \tanh(W_j^{q2} I^t + b_j^{q2}) \quad j = 1, ..., m. \tag{2.56}$$

Basierend auf der Ausgabe q_m^t werden daraufhin die Zustandsvariablen wie folgt aktualisiert:

$$h^t = \exp(-\|x^t\|_2 \cdot \alpha^t) \cdot (h^{t-1} - \beta^t) + \beta^t \tag{2.57}$$

mit

$$\alpha^t = \exp(W_\alpha q_m^t + b_\alpha) \tag{2.58}$$

$$\beta^t = \tanh(W_\beta q_m^t + b_\beta). \tag{2.59}$$

Die Ausgabe des Modells wird dann basierend auf den aktualisierten Zustandsvariablen linear bestimmt:

$$y^t = W_o h^t. \tag{2.60}$$

Bemerkung 2.3.1. Die Berechnung der Zustandsvariable in (2.57) bewirkt, dass für $x^t = 0$ wegen $h^t = \exp(-0 \cdot \alpha^t) \cdot (h^{t-1} - \beta^t) + \beta^t = 1 \cdot h^{t-1}$ der Zustand für den nächsten Zeitschritt nicht verändert wird. Insbesondere gilt

$$h^{t-1} = 0 \text{ und } x^t = 0 \Rightarrow h^t = 0 \Rightarrow y^t = 0, \tag{2.61}$$

da die Ausgabe in (2.60) linear und ohne Bias-Wert berechnet wird.

3 Training von ML-Modellen basierend auf klassischen Materialmodellen

Zunächst wird untersucht, inwieweit bestehende klassische Materialmodelle durch Methoden des maschinellen Lernens beschrieben werden können und welche Modelle dafür geeignet sind. In dieser Arbeit wird im Folgenden zunächst in Kapitel 3.1 ein einfaches linear elastisches Materialmodell betrachtet. Anschließend werden in Kapitel 3.2 für ein elastoplastisches Materialmodell unterschiedliche Modellierungsmöglichkeiten sowie verschiedene ML-Modelle und Methoden untersucht, um ein datenbasiertes Ersatzmodell für das klassischen Materialmodell zu erhalten. Die unterschiedlichen Modelle werden zunächst in Abschnitt 3.2.6 anhand verschiedener Datensätze evaluiert und die performantesten Modelle in 3.2.7 mittels mehrerer Simulationen validiert. Die Dehnungskomponenten werden in den folgenden Kapiteln, wie es auch in den entsprechenden in LS-Dyna implementierten klassischen Materialmodellen der Fall ist, durchgehend in inkrementeller Form (anstelle von absoluten, aufsummierten Dehnungswerten) genutzt.

3.1 Linear elastisches Materialmodell

Als *Elastizität* wird eine reversible Verformung bezeichnet (siehe [3]). Dies bedeutet, dass bei einer Entlastung der Ursprungszustand eines Körpers wieder angenommen wird, wie es beispielsweise bei einer Feder der Fall ist. Bei Elastizität kann das Materialverhalten als bijektive Abbildung zwischen Dehnungen und Spannungen beschrieben werden. Für den Fall eines linear elastischen Materialmodells (auch als *Hooke'sches Materialgesetz* bezeichnet) ist dies für einen beliebigen Zeitschritt die lineare Abbildung $f : \mathbb{R}^6 \rightarrow$

$\mathbb{R}^6, \varepsilon^t \mapsto \sigma^t = C_\nu^E \varepsilon^t$ mit

$$\underbrace{\begin{pmatrix} \sigma_{xx}^t \\ \sigma_{yy}^t \\ \sigma_{zz}^t \\ \sigma_{xy}^t \\ \sigma_{yz}^t \\ \sigma_{zx}^t \end{pmatrix}}_{=\sigma^t} = \underbrace{\frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{pmatrix}}_{=:C_\nu^E} \underbrace{\begin{pmatrix} \varepsilon_{xx}^t \\ \varepsilon_{yy}^t \\ \varepsilon_{zz}^t \\ \varepsilon_{xy}^t \\ \varepsilon_{yz}^t \\ \varepsilon_{zx}^t \end{pmatrix}}_{=\varepsilon^t}, \tag{3.1}$$

abhängig von den beiden materialspezifischen Parametern E (*Elastizitätsmodul*) und ν (*Querkontraktionszahl*). Das klassische Materialmodell kann daher durch einfache (lineare) Regressionsmodelle modelliert werden. Insbesondere können, da die Spannungsantwort lediglich von den Dehnungen des aktuellen Zeitschrittes abhängen, ML-Modelle ohne zeitliche Abhängigkeit (wie beispielsweise Feed Forward neuronale Netze) betrachtet werden.

Da es sich beim linear elastischen Materialmodell lediglich um eine lineare Abbildung handelt, kann dieses Materialmodell direkt durch ein lineares Regressionsmodell ersetzt werden (siehe auch Bemerkung 3.1.1). In Hinblick darauf, dass für die Erstellung eines ML-Materialmodells für einen neuen Werkstoff mittels des Trainings anhand von Versuchsdaten (wie es in Kapitel 4 untersucht wird) keine zugrunde liegende klassische Berechnung (insbesondere der Annahme der Linearität) im Vorhinein bekannt ist, werden im Folgenden Feed Forward neuronale Netze als universell und flexibler einsetzbare ML-Methode betrachtet. Als Anwendungsfall werden zwei neuronale Netze mit unterschiedlichen Netzstrukturen für ein E-Modul von 210 und einer Querkontraktionszahl von $0,3$ trainiert. Für die Erzeugung der Trainingsdaten werden normalverteilt Dehnungen als Eingabe generiert und anschließend mit dem zugrunde liegenden Materialmodell gegeben durch (3.1) berechnet. Der so entstandene Trainingsdatensatz wird als $D_{train}^{\mathcal{N}(0,0.001)}$ bezeichnet, wobei sich die hier gewählte Verteilung, die Varianz und der Mittelwert an in Simulationen auftretenden Werten orientieren. Die erste betrachtete Netzstruktur besteht aus einer Eingabeschicht, welche ohne dazwischenliegende Schichten und Bias-Werte direkt mit der Ausgabeschicht verbunden ist. Dieses Modell mit lediglich 36 trainierbaren Parametern ist somit äquivalent zu einem linearen Regressionsmodell mit dem Unterschied der Art und Weise, wie die Parameter des Modells anhand der gegebenen Trainingsdaten angepasst werden (siehe Kapitel 2.3). Das trainierte neuronale Netz wird im Folgenden mit $ML_{210-0,3}^{linel,36}$

bezeichnet. Für dieses Modell wird daher nach wie vor eine lineare Beziehung zwischen Dehnungen und Spannungen angenommen. Das zweite betrachtete neuronale Netz mit insgesamt 384 trainierbaren Parametern, welches als $ML_{210-0,3}^{linel,384}$ bezeichnet wird, hat eine zusätzliche dazwischenliegende Schicht mit 32 Neuronen ohne Bias-Werte und einer ReLU Aktivierungsfunktion. Dieses ML-Modell repräsentiert ein für die zu erlernende lineare Funktion überkomplexes Modell mit redundanten Parametern und nimmt die Linearität des zugrunde liegenden Materialmodells nicht bereits im Vorfeld an. Beide Modelle werden auf Basis eines Trainingssatzes von 100.000 Dehnungs-Spannungspaaren trainiert. Für die Auswertung werden zwei Testdatensätze bestehend aus je 10.000 Dehnungs-Spannungspaaren betrachtet. Der erste Testdatensatz $D_{test}^{\mathcal{N}(0,0.001)}$ basiert auf der gleichen Wahrscheinlichkeitsverteilung wie der Trainingsdatensatz. Für einen weiteren, ebenfalls normalverteilten Testdatensatz $D_{test}^{\mathcal{N}(3,0.1)}$ wird ein anderer Mittelwert und eine andere Standardabweichung gewählt. Abbildung 3.1 zeigt den Fehlerverlauf des Trainings- und eines Testdatensatzes während des Trainings der beiden ML-Modelle. Das kleinere Modell $ML_{210-0,3}^{linel,36}$ wird hierbei mit einer Batchgröße von 1000, einer Lernrate gleich 1 und dem Adam Optimierer 20 Epochen lang trainiert. Bei dem größeren Modell mit 384 trainierbaren Parametern wird die Lernrate auf $0,1$ verringert und die Trainingszeit auf 35 Epochen erhöht.

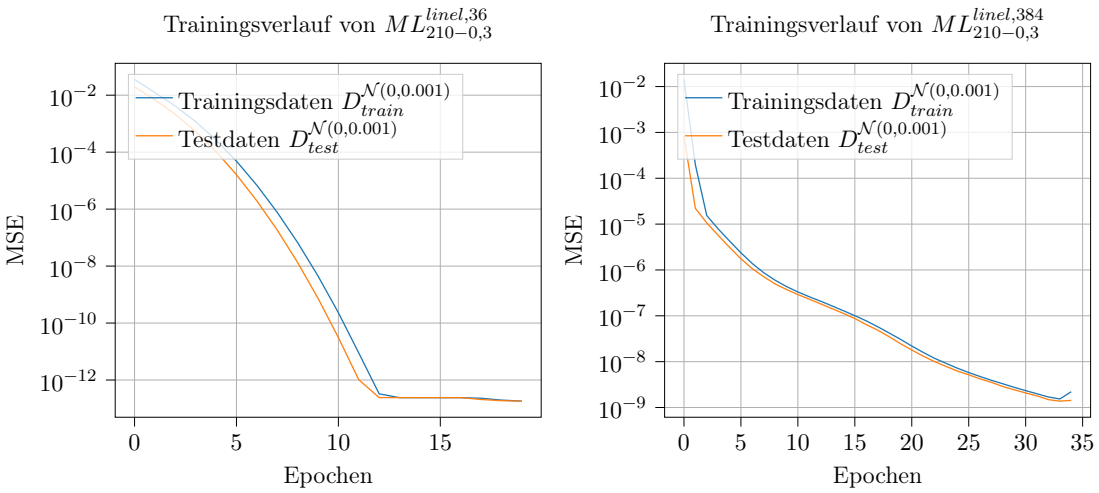


Abbildung 3.1: Verlauf des MSE-Fehlers über die Epochen im Training für den Trainingsdatensatz $D_{train}^{\mathcal{N}(0,0.001)}$ und den Testdatensatz $D_{test}^{\mathcal{N}(0,0.001)}$

Tabelle 3.1 zeigt den MSE-Fehler und den R^2-Wert der beiden ML-Modelle bezüglich des Trainings- und der beiden Testdatensätze, wobei sich die höheren MSE-Werte des zweiten Testdatensatzes $D_{test}^{\mathcal{N}(3,0.1)}$ durch die insgesamt größeren Werte aufgrund der geänderten

Verteilung ergeben. Die R^2 Werte bleiben bei allen Datensätzen hoch.

	$ML^{linel,36}_{210-0,3}$	$ML^{linel,384}_{210-0,3}$
MSE $D^{\mathcal{N}(0,0.001)}_{train}$	$1.801 \cdot 10^{-13}$	$1.463 \cdot 10^{-9}$
MSE $D^{\mathcal{N}(0,0.001)}_{test}$	$1.811 \cdot 10^{-13}$	$1.430 \cdot 10^{-9}$
MSE $D^{\mathcal{N}(3,0.1)}_{test}$	$1.591 \cdot 10^{-6}$	$5.148 \cdot 10^{-3}$
R^2 $D^{\mathcal{N}(0,0.001)}_{train}$	0.999	0.999
R^2 $D^{\mathcal{N}(0,0.001)}_{test}$	0.999	0.999
R^2 $D^{\mathcal{N}(3,0.1)}_{test}$	0.999	0.999

Tabelle 3.1: Auswertung der ML-Modelle $ML^{linel,36}_{210-0,3}$ und $ML^{linel,384}_{210-0,3}$ mittels der MSE und R^2-Werte für Test- und Trainingsdaten

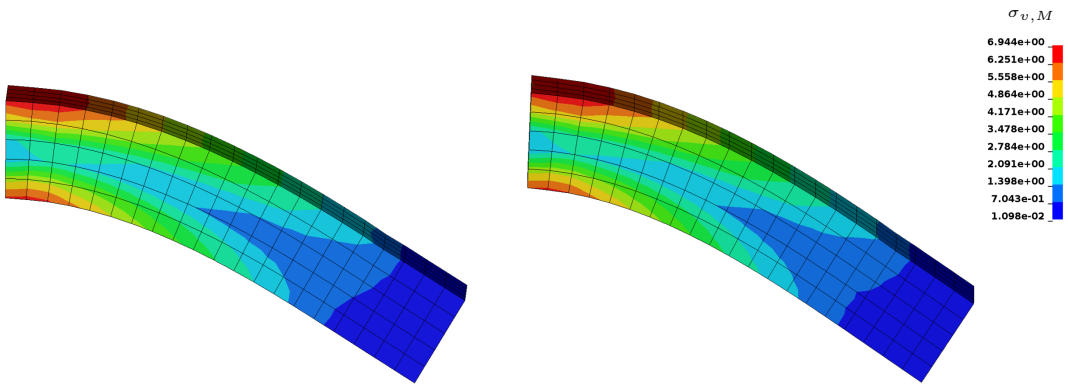
Da das neuronale Netz bestehend aus 36 trainierbaren Parametern äquivalent und damit direkt vergleichbar zu der erlernenden, linearen Funktion ist, kann direkt an den Parametern des ML-Modells (wie in (3.2) und (3.3) gezeigt) geprüft werden, wie gut das klassische Materialmodell approximiert wird. Die trainierbaren Parameter des ML-Modells $ML^{linel,36}_{210-0,3}$ werden für diesen Vergleich in einer Matrix $C_{ML^{36}_{210-0,3}} \in \mathbb{R}^{6\times6}$, gerundet auf die dritte Nachkommastelle angeordnet.

$$C_{ML^{36}_{210-0,3}} = \begin{pmatrix} 282.692 & 121.154 & 121.154 & 0 & 0 & 0 \\ 121.154 & 282.692 & 121.154 & 0 & 0 & 0 \\ 121.154 & 121.154 & 282.692 & 0 & 0 & 0 \\ 0 & 0 & 0 & 80.769 & 0 & 0 \\ 0 & 0 & 0 & 0 & 80.769 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80.769 \end{pmatrix} \tag{3.2}$$

$$C^{210}_{0,3} = \begin{pmatrix} 282.692 & 121.154 & 121.154 & 0 & 0 & 0 \\ 121.154 & 282.692 & 121.154 & 0 & 0 & 0 \\ 121.154 & 121.154 & 282.692 & 0 & 0 & 0 \\ 0 & 0 & 0 & 80.769 & 0 & 0 \\ 0 & 0 & 0 & 0 & 80.769 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80.769 \end{pmatrix} \tag{3.3}$$

Bemerkung 3.1.1. Da die lineare zu approximierende Funktion bekannt und kein Rauschen vorhanden ist (das heißt die Datenpaare liegen exakt auf der Funktion), entspricht hier ein lineares Regressionsmodell einem linearen Gleichungssystem mit $n \cdot k$ Gleichungen und $k \cdot p$ Variablen (entsprechend der zu bestimmenden Parametermatrix des Regressionsmodells), wobei durch n die Anzahl an Datenpaaren, k die Anzahl an Ausgaben und p die Anzahl an Eingaben bezeichnet wird. Da es sich um sechs Ein- und Ausgaben handelt, kann ein exaktes lineares Regressionsmodell mit nur sieben unterschiedlichen Datenpaaren bestimmt werden, wobei die Wahl dieser Datenpaare (insbesondere die Wahl der Verteilung) hierfür keine Rolle spielt.

Abbildung 3.2 zeigt das trainierte ML-Modell $ML^{linel,384}_{180-0,2}$ in einer verschiebungsgesteuerten Kragarm-Simulation. Zu sehen sind die von-Mises-Spannungen $\sigma_{v,M}$ am Ende der Simulation, wobei sowohl in der Geometrie (Verformung des Körpers) als auch bezüglich der Werte der von-Mises-Spannungen kaum Unterschiede zwischen der Simulation mit dem ML-Materialmodell und der Simulation mit dem klassischen Materialmodell zu erkennen sind. Um das ML-Materialmodell in der Simulation anwenden zu können, wurde die Python-Fortran Schnittstelle aus [67] genutzt.



Kragarm Simulation mit $ML^{linel,384}_{210-0,3}$

Kragarm Simulation mit klassischem Materialmodell

Abbildung 3.2: Vergleich der von-Mises-Spannungen $\sigma_{v,M}$ einer Kragarm-Simulation mit dem ML-Materialmodell $ML^{linel,384}_{180-0,2}$ (links) und dem klassischen linear elastischen Materialmodell (rechts)

3.2 Materialmodell mit Plastizität

In diesem Abschnitt wird ein dehnratenunabhängiges, elastoplastisches Materialverhalten betrachtet. Bei einem elastoplastischen Materialmodell liegt je nach Belastung ein elastisches oder ein plastisches Verhalten des Materials vor. Im Unterschied zu einem elastischen Materialverhalten kehrt ein Festkörper bei plastischen Verformungen nicht mehr in seinen Ursprungszustand zurück (siehe [3]). In diesem Fall hängt ein Spannungszustand nicht nur von den aktuellen Dehnungen, sondern von dem bisherigen zeitlichen Dehnungsverlauf ab. Die ML-Materialmodelle werden für das Schalenmodell des Materialmodells `*MAT_PIECEWISE_LINEAR_PLASTICITY` (`*MAT_024`) des Solvers LS-DYNA (siehe [49]) entwickelt, welches auf der von Mises'schen Elastoplastizitätstheorie (siehe [50]) basiert. Abhängig von verschiedenen Materialparametern kann dieses Materialmodell für die Modellierung vieler unterschiedlicher Werkstoffe mit elastoplastischem Verhalten, wie beispielsweise Aluminium, Stahl oder Kunststoffe genutzt werden. In diesem Kapitel werden die Materialparameter für einen Dualphasenstahl (DP-Stahl) DP800 mit einer Mindestzugfestigkeit von 800 Megapascal betrachtet. Bei Dualphasenstählen handelt es sich um eine spezielle Art von hochfestem Stahl, welcher sich aus zwei unterschiedlichen Phasen, bestehend aus Ferrit und Martensit, zusammensetzt. Diese Mikrostruktur verleiht dem Stahl eine Kombination aus hoher Festigkeit (durch das Martensit) sowie Dehnbarkeit (durch das Ferrit), was ihn für Anwendungen, welche eine hohe mechanische Belastbarkeit fordern, geeignet macht (siehe auch [68]).

3.2.1 Klassische Berechnung

Bei einer klassischen Berechnung der Spannungsausgabe des elastoplastischen Materialmodells werden zunächst die Spannungskomponenten ähnlich wie bei dem bereits betrachteten linear elastischen Materialmodell bestimmt. Um festzustellen, ob sich ein Spannungszustand im elastischen oder bereits im plastischen Bereich befindet, wird eine sogenannte *Fließbedingung* herangezogen. Dafür wird auf Basis der verschiedenen Spannungskomponenten ein skalarer Wert, die sogenannte *Vergleichsspannung* σ_v bestimmt, welche mit der Fließspannung σ^y verglichen wird. Auf Basis des Vergleichs von Fließspannung und Vergleichsspannung wird daraufhin, wie in (3.4) angegeben, bestimmt, ob sich ein Zustand im elastischen oder plastischen Bereich befindet.

$$\sigma_v - \sigma^y \begin{cases} < 0 & \text{elastischer Bereich} \\ \geq 0 & \text{plastischer Bereich} \end{cases} \tag{3.4}$$

Das hier betrachtete Materialmodell basiert auf dem von Mises'schen Fließkriterium, bei dem für die Vergleichsspannungen die von Mises Spannung wie in Definition 2.5 herangezogen wird. Befindet man sich im plastischen Bereich wird die Fließspannung durch die sogenannte *effektive plastische Dehnung* $\varepsilon_{eff,pl}$ (im Folgenden auch kurz als *plastische Dehnung* bezeichnet) angepasst, sodass die Fließspannung der Vergleichsspannung entspricht beziehungsweise sich dieser annähert. Die effektive plastische Dehnung kann dabei als Geschichtsvariable (ähnlich den Zustands- oder Gedächtnisvariablen bei rekurrenten neuronalen Netzen) betrachtet werden, welche den bisherigen Verlauf der Dehnungen zusammenfasst.
Die klassische Formulierung von `*MAT_24` ist abhängig von den materialspezifischen Parametern $E, \nu \in \mathbb{R}$ (Elastizitätsmodul und Querkontraktionszahl) sowie einer Fließkurve $C : \mathbb{R}_{\geq 0} \to \mathbb{R}$. Diese Parameter sind unabhängig davon, welcher räumliche Punkt im Körper betrachtet wird und bleiben konstant im zeitlichen Verlauf. Die Fließkurve für den in dieser Arbeit betrachteten Stahl DP800 ist in Abbildung 3.3 dargestellt. Das Elastizitätsmodul E beträgt 210 und die Querkontraktionszahl $0,3$. Basierend auf diesen Materialparametern wird das *Schubmodul* G sowie das *Kompressionsmodul* K, gegeben durch

$$G = \frac{E}{2 \cdot (1 + \nu)} \tag{3.5}$$

$$K = \frac{E}{3 \cdot (1 - 2 \cdot \nu)}, \tag{3.6}$$

einmalig und unabhängig der Eingaben (Dehnungen) bestimmt.

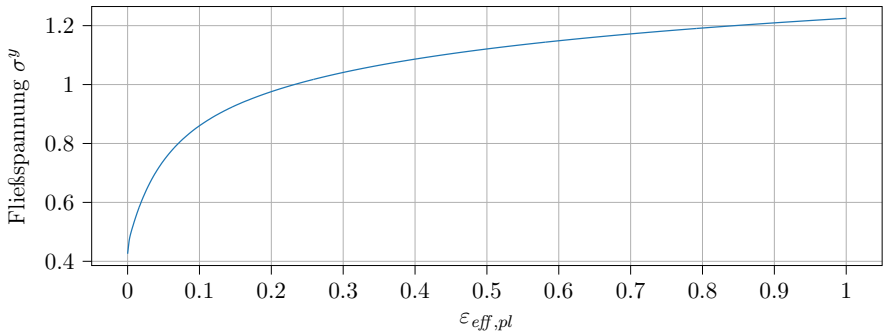


Abbildung 3.3: Fließkurve für den Stahl DP800

Die klassische Berechnung erfolgt für jeweils einen räumlichen Punkt mit Eingabe der gegebenen Dehnungsinkremente $\varepsilon_1^t, ..., \varepsilon_6^t$ des aktuell betrachteten Zeitschrittes t sowie der absoluten Spannungen $\sigma_1^{t-1}, ..., \sigma_6^{t-1}$ des vorherigen Zeitschrittes $t-1$ und der effektiven plastischen Dehnung $\varepsilon_{eff,pl}^{t-1}$ des vorhergegangenen Zeitschrittes $t-1$ auf Basis der von

Mises'schen Elastoplastizitätstheorie. Als Ausgabe werden die in der Simulation benötigten absoluten Spannungen $\sigma_1^t, ..., \sigma_6^t$ an diesem Punkt und ein aktualisierter Wert für die plastische Dehnung $\varepsilon_{eff,pl}^t$ für den betrachteten Zeitschritt t bestimmt. Die Komponenten $\varepsilon_3^t = \varepsilon_{zz}^t$ und $\sigma_3^t = \sigma_{zz}^t$ spielen bei der Schalenformulierung keine Rolle und werden grundsätzlich gleich null gesetzt. Obwohl es sich um eine Schalenformulierung handelt, werden bei der 2D-Formulierung des `*MAT_24` auch die Komponenten $\varepsilon_5^t = \varepsilon_{yz}^t, \varepsilon_6^t = \varepsilon_{zx}^t$ mitberücksichtigt, allerdings lediglich durch eine linear elastische Berechnung

$$\sigma_i^t = \sigma_i^{t-1} + E \cdot \varepsilon_i^t \quad \text{für } i = 5, 6 \tag{3.7}$$

unabhängig der restlichen Komponenten und gehen nicht in die Fließbedingung mit ein. Die übrigen Spannungskomponenten $\sigma_1^t = \sigma_{xx}^t, \sigma_2^t = \sigma_{yy}^t, \sigma_4^t = \sigma_{xy}^t$ und die aktualisierte effektive plastische Dehnung $\varepsilon_{eff,pl}^t$ werden gegeben den Dehnungen $\varepsilon_1^t = \varepsilon_{xx}^t, \varepsilon_2^t = \varepsilon_{yy}^t, \varepsilon_4^t = \varepsilon_{xy}^t$, den vorherigen Spannungen $\sigma_1^{t-1}, \sigma_2^{t-1}, \sigma_4^{t-1}$ sowie der vorherigen effektiven plastischen Dehnung $\varepsilon_{eff,pl}^{t-1}$, wie in den folgenden Schritten genauer beschrieben, für jeweils einen räumlichen Punkt und einen Zeitpunkt bestimmt:

1.) Es werden zunächst vorläufige Spannungen $\sigma_1^{trial}, \sigma_2^{trial}, \sigma_4^{trial}$ (auch *Versuchsspannungen* genannt) und eine vorläufige effektive plastische Dehnung $\varepsilon_{eff,pl}^{trial}$ bestimmt:

$$\begin{aligned}
\varepsilon_{eff,pl}^{trial} &= \varepsilon_{eff,pl}^{t-1} \\
\sigma_i^{trial} &= \sigma_i^{t-1} + dp + 2 \cdot G \cdot (\varepsilon_i^t - \varepsilon^{vol}) && \text{für } i = 1, 2 \\
\sigma_i^{trial} &= \sigma_i^{t-1} + G \cdot \varepsilon_i^t && \text{für } i = 4
\end{aligned}$$

mit

$$\varepsilon^{vol} = \frac{\varepsilon_1^t + \varepsilon_2^t + \Delta\varepsilon_{zz}^{trial}}{3} \tag{3.8}$$

$$dp = 3 \cdot K \cdot \varepsilon^{vol} \tag{3.9}$$

$$\Delta\varepsilon_{zz}^{trial} = \frac{-\nu}{1-\nu} \cdot (\varepsilon_1^t + \varepsilon_2^t). \tag{3.10}$$

2.) Die aktuelle Fließspannung $\sigma^y = C(\varepsilon_{eff,pl}^{trial})$ wird durch die gegebene materialspezifische Fließkurve $C : \mathbb{R}_{\geq 0} \to \mathbb{R}, \varepsilon_{eff,pl} \mapsto \sigma^y$ abhängig der aktuellen plastischen Dehnung $\varepsilon_{eff,pl}^{trial}$ bestimmt.

3.) Es wird überprüft, ob die Fließbedingung

$$\sqrt{3 \cdot J_2} \leq \sigma^y \tag{3.11}$$

mit

$$J_2 = (\sigma_4^{trial})^2 - q_1 \cdot q_2 - q_2 \cdot q_3 - q_1 \cdot q_3 \tag{3.12}$$

und

$$p = \frac{1}{3} \cdot (\sigma_1^{trial} + \sigma_2^{trial} + dp + 2 \cdot G \cdot (\Delta\varepsilon_{zz}^{trial} - \varepsilon^{vol})) \tag{3.13}$$

$$q_i = \begin{cases} \sigma_i^{trial} - p & \text{für } i = 1,2 \\ dp + 2 \cdot G \cdot (\Delta\varepsilon_{zz}^{trial} - \varepsilon^{vol}) - p & \text{für } i = 3 \end{cases} \tag{3.14}$$

erfüllt ist. Falls die Fließbedingung in (3.11) nicht erfüllt ist (für $\sqrt{3 \cdot J_2} - \sigma^y > 0$), wird die plastische Dehnung mittels des in [64] beschriebenen *return mapping* Algorithmus wie folgt angepasst:

Zunächst wird $D(0) = 0, \varepsilon_{eff,pl}^{trial}(0) = \varepsilon_{eff,pl}^{trial}$ und $\sigma^y(0) = \sigma^y$ initialisiert. Iterativ wird dann der Wert $D(j)$ in einer Iteration $j = 1, ..., j^\star$ durch

$$D(j) = D(j-1) - \frac{\sqrt{3 \cdot J_2} - \sigma^y(j-1) - 3 \cdot G \cdot D(j-1)}{-3 \cdot G - H(j-1) + 10^{-20}}$$

aktualisiert, wobei $H(j)$ die Ableitung der Fließkurve an der Stelle $\varepsilon_{eff,pl}^{trial}(j)$ bezeichnet. Die plastische Dehnung wird anschließend durch

$$\varepsilon_{eff,pl}^{trial}(j) = \varepsilon_{eff,pl}^{t-1} + D(j)$$

angepasst. Mit dem aktualisiertem Wert $\varepsilon_{eff,pl}^{trial}(j)$ wird mittels der Fließkurve erneut die zugehörige Fließspannung $\sigma^y(j)$ sowie $H(j)$ bestimmt bis in einer Iteration $j^\star$ die Bedingung

$$\frac{|\sqrt{3 \cdot J_2} - \sigma^y(j^\star) - 3 \cdot G \cdot D(j^\star)|}{\sqrt{3 \cdot J_2}} < 10^{-6} \tag{3.15}$$

erfüllt ist.

Abschließend werden mittels des erhaltenen $D(j^\star)$ die Versuchsspannungen durch

$$\sigma_i^{trial} = \sigma_i^{trial} - \frac{3 \cdot G \cdot D(j^\star)}{\sqrt{3 \cdot J_2}} \cdot q_i \qquad \text{für } i = 1,2$$

$$\sigma_i^{trial} = \sigma_i^{trial} - \frac{3 \cdot G \cdot D(j^\star)}{\sqrt{3 \cdot J_2}} \cdot \sigma_i^{trial} \qquad \text{für } i = 4$$

angepasst und die Spannungskomponenten $\sigma_i^t = \sigma_i^{trial}$ für $i = 1,2,4$ sowie die

plastische Dehnung $\varepsilon^t_{eff,pl} = \varepsilon^{trial}_{eff,pl}$ ausgegeben.

3.2.2 Modellierungsmöglichkeiten des klassischen Materialmodells im Bereich des maschinellen Lernens

Auf Basis der klassischen Berechnung, wie in Abschnitt 3.2.1 beschrieben, kann das Materialmodell für einen Zeitschritt t als Abbildung

$$f : \mathbb{R}^{13} \to \mathbb{R}^7, (\sigma_1^{t-1}, ..., \sigma_6^{t-1}, \varepsilon_1^t, ..., \varepsilon_6^t, \varepsilon_{eff,pl}^{t-1}) \mapsto (\sigma_1^t, ..., \sigma_6^t, \varepsilon_{eff,pl}^t)$$

betrachtet werden, wobei die Abbildung f selbst keine zeitabhängige Funktion ist, jedoch in jedem Zeitschritt t einer Simulation aufgerufen wird. Innerhalb der Simulation wird hierbei lediglich die Vorhersage der Spannungen $\sigma_1, ..., \sigma_6$ benötigt. Die effektive plastische Vergleichsdehnung $\varepsilon_{eff,pl}$ dient als zusammenfassender Gedächtniszustand des bisherigen Verlaufs und wird in jedem Zeitschritt zwischengespeichert. Durch die Existenz dieser Modell-internen Geschichtsvariable $\varepsilon_{eff,pl}$, welche jedoch innerhalb der Simulation nicht zwingend als Ausgabe erforderlich ist, ergeben sich die folgenden zwei Möglichkeiten zum Modellieren des klassischen Materialmodells:

(1) Training von ML-Modellen ohne Hinzunahme der internen Geschichtsvariable:
Das Materialmodell kann als multivariate Zeitreihe mit $((X_t, Y_t) : t \in \{1, ..., T\})$ mit $X_t = (\varepsilon_1^t, ..., \varepsilon_6^t)$ und $Y_t = (\sigma_1^t, ..., \sigma_6^t)$ modelliert werden.

(2) Training von ML-Modellen mit Hinzunahme der internen Geschichtsvariable:
Das Materialmodell kann als Regressionsproblem mit der unabhängigen Variable $X = (\sigma_1^{t-1}, ..., \sigma_6^{t-1}, \varepsilon_1^t, ..., \varepsilon_6^t, \varepsilon_{eff,pl}^{t-1})$ und der abhängigen Variable $Y = (\sigma_1^t, ..., \sigma_6^t, \varepsilon_{eff,pl}^t)$ für einen beliebig festen Zeitschritt $t \in \{1, ..., T\}$ modelliert werden.

In den folgenden Abschnitten 3.2.4 und 3.2.5 werden unterschiedliche ML-Materialmodelle bezüglich dieser beiden Modellierungsvarianten (1) und (2) für den DP800 Stahl trainiert. Weiterhin kann unterschieden werden zwischen einem (A) holistischem und einem (B) hybriden Modellierungsansatz mit oder ohne Nachberechnung eines Teils der Spannungskomponenten:

(A) Das klassische Materialmodell wird vollständig wie in Abschnitt 3.2.1 durch ein ML-Modell beschrieben. In diesem Fall sollen alle Spannungskomponenten vom ML-Modell vorhergesagt werden.

(B) Es werden nur die Komponenten eines ebenen Spannungszustandes durch das ML-Modell beschrieben und die Komponenten σ_5, σ_6 werden durch

$$\sigma_i^t = \sigma_i^{t-1} + E \cdot \varepsilon_i^t \quad \text{für } i = 5, 6 \tag{3.16}$$

elastisch nachberechnet. Die Eingaben des Modells reduzieren sich so zu $\varepsilon_1^t, \varepsilon_2^t, \varepsilon_4^t$ und die Ausgaben sind gegeben durch $\sigma_1^t, \sigma_2^t, \sigma_4^t$. Für Training mit Geschichtsvariablen kommen entsprechend die Eingaben $\sigma_1^{t-1}, \sigma_2^{t-1}, \sigma_4^{t-1}, \varepsilon_{\mathit{eff},pl}^{t-1}$ und die Ausgabe $\varepsilon_{\mathit{eff},pl}^t$ hinzu.

Ein Vorteil der hybriden Betrachtung ist eine Reduktion der Eingaben und Ausgaben für das ML-Modell sowie eine Vermeidung der Ungenauigkeit der Vorhersagen der Spannungskomponenten σ_5 und σ_6 vom ML-Materialmodell. Der Hauptgrund für eine Betrachtung der hybriden Trainingsvariante (B) ergibt sich allerdings dadurch, dass beim Training aus Versuchsdaten, wie es in Kapitel 4 thematisiert wird, die Eingaben ε_5 und ε_6, welche deutlich aufwändiger als die restlichen Dehnungskomponenten im Versuch abzumessen sind, nicht bestimmt werden müssen. Der Nachteil der hybriden Betrachtung liegt darin, dass das ML-Materialmodell nicht vollständig unabhängig von klassischen Materialparametern betrachtet werden kann, da das Elastizitätsmodul E bekannt sein muss. Dieses kann jedoch sehr leicht anhand eines einfachen Zugversuches bestimmt werden. Mit dem Ziel ML-Materialmodelle auf Basis von Versuchsdaten zu trainieren, wird deshalb in dieser Arbeit der hybride Ansatz (B) betrachtet.

Lernen mit Zeitfenstern

Im Allgemeinen kann beim Training von ML-Modellen für Zeitreihen in einigen Fällen mit Zeitfenstern gearbeitet werden, mit der Annahme, dass für eine Vorhersage des Modells die Information über einen Ausschnitt der letzten Zeitschritte genügt. In solch einem Fall werden Auszüge aus einer längeren Zeitreihe betrachtet und dem Modell zum Lernen zur Verfügung gestellt. Im Folgenden wird gezeigt, dass diese Vorgehensweise im Falle eines elastoplastischen Materialverhaltens ohne Hinzunahme der plastischen Dehnung $\varepsilon_{\mathit{eff},pl}$ selbst für längere Zeitfenster nicht in Frage kommt, da ein Spannungszustand σ^t zu einem Zeitpunkt t vom gesamten bisherigen Verlauf $(\varepsilon^s)_{s \leq t}$ der Dehnungen abhängen kann, wie das folgende Beispiel zeigt.

Betrachtet wird hierfür ein Pfad $(\hat{X}_t)_{t=0,\dots,T}$ bestehend aus den verschiedenen Eingangskomponenten (Dehnungen). Für eine festgelegte Komponente eines Eingabewertes

$i \in \{1, 2, 4\}$ wird für jedes $s \leq T$ ein Pfad $(X_t^s)_{t=0,\dots,T}$ gegeben durch

$$X_{t,j}^s = \begin{cases} \hat{X}_{t,j} + \delta & \text{falls } j = i \text{ und } s = t \\ \hat{X}_{t,j} & \text{sonst} \end{cases} \tag{3.17}$$

mit einen festgelegten Wert $\delta \in \mathbb{R}$ konstruiert. Die Ausgabewerte $Y_{t,j}^s$ der erzeugten Dehnpfade (bestehend aus den Spannungskomponenten) werden für jeden Zeitschritt iterativ auf Basis des klassischen Materialmodells bestimmt und die Abweichung zur Ausgabe $\hat{Y}_T^s$ des ursprünglichen Pfades im letzten Zeitschritt T verglichen. In Abbildung 3.5 ist die Abweichung $|Y_{T,j}^s - \hat{Y}_{T,j}|$ für einen Dehnpfad aus einer Hutprofil Simulation (siehe Abbildung 3.4) mit $T = 1000$, $\delta = 0,1$ und $i = 1$ dargestellt.

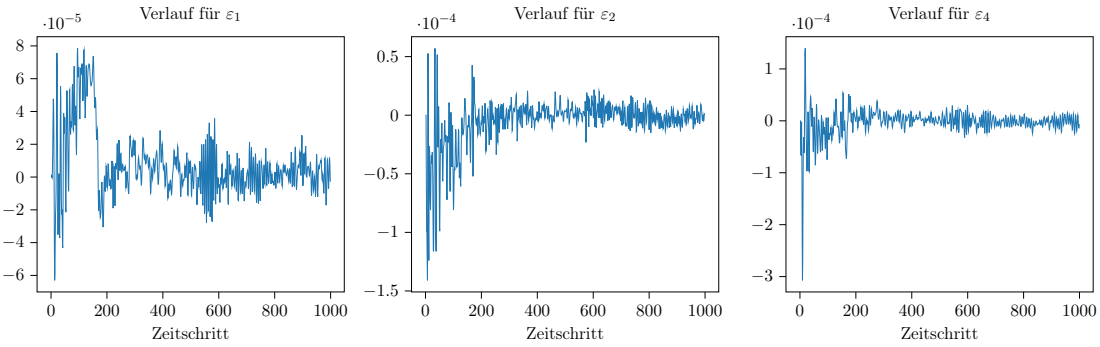


Abbildung 3.4: Dehnungsverläufe für die Komponenten $\varepsilon_1, \varepsilon_2, \varepsilon_4$ des gewählten Beispielpfades aus einer Hutprofil Simulation

Dass für die Spannungskomponenten σ_1, σ_2 und σ_4 selbst in den vorderen Zeitschritten deutliche Abweichungen zu sehen sind, zeigt, dass eine Änderung des Wertes einer einzelnen Eingangskomponente in der Zeitreihe sehr viel später noch deutliche Auswirkungen auf verschiedene Ausgabekomponenten des Modells hat. Ein Trend im Verlauf der Abweichungen ist nicht erkennbar, sodass ein höherer Einfluss bezüglich früheren oder späteren Änderungen im Pfad nicht grundsätzlich gefolgert werden kann. Die Abweichungen der Komponenten σ_i für $i = 5, 6$ sind konstant null über die Zeitschritte, da in diesem Beispiel die Eingangskomponente ε_1 variiert wird und die Ausgaben σ_5 und σ_6 nur abhängig der Dehnungskomponenten ε_5 beziehungsweise ε_6, nicht jedoch der jeweils anderen Komponenten sind (siehe (3.7)).

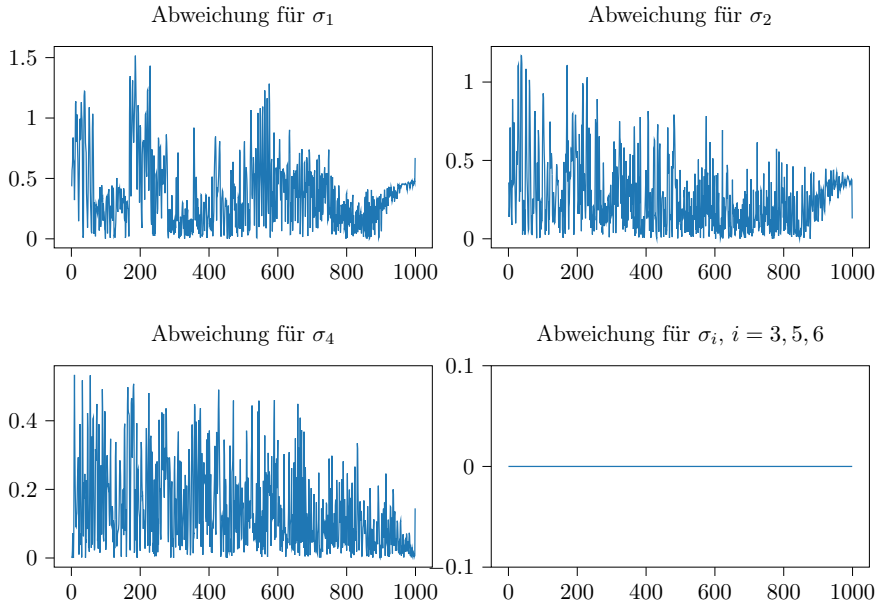


Abbildung 3.5: Abweichung $|Y^s_{T,j} - \hat{Y}_{T,j}|$ für einen Dehnpfad aus einer Hutprofil Simulation mit $T = 1000$ und $\delta = 0,1$ bei einer Veränderung der Eingangskomponente ε_i für $i = 1$ für die jeweiligen Ausgaben σ_j mit $j = 1, ..., 6$

3.2.3 Generieren von Trainings- und Testdaten

Da in diesem Kapitel ML-Ersatzmodelle für ein bereits bestehendes klassisches Materialmodell entwickelt werden, kann das klassische Materialmodell wie auch im Falle des linear elastischen Materialmodells in 3.1 für die Erzeugung von Daten genutzt werden. Hierfür werden zufällig zeitliche Verläufe der verschiedenen Dehnungskomponenten, welche im Folgenden auch als Dehnpfade bezeichnet werden, generiert, anschließend in äquidistante Zeitschritte diskretisiert und mithilfe der klassischen Berechnung (ohne eine Durchführung von Simulationen) die zugehörigen Spannungsverläufe bestimmt. Die Methoden zur Generierung der Dehnpfade wurden von Projektpartnern im Rahmen des Verbundprojektes AIMM - Artificial Intelligence for Material Models (siehe [58]) entwickelt und werden hier grundlegend zusammengefasst. Detailliertere Beschreibungen hierzu finden sich in [37] und [58]. Mit dem Ziel ein allgemeingültiges, nicht anwendungsspezifisches ML-Materialmodell zu erhalten, werden die Trainingsdaten generisch erzeugt anstatt aus anwendungsspezifischen Simulationen extrahiert, um besser balancierte Daten zu bekommen und zu erlernende Phänomene wie beispielsweise den Wechsel zwischen Belastungen und Entlastungen zu berücksichtigen.

Zufällig generierte Dehnpfade

Es werden zwei unterschiedliche Varianten betrachtet, um die Verläufe der einzelnen Dehnungskomponenten $(\varepsilon^1, ..., \varepsilon^{T^\star})$ mit $\varepsilon^i = (\varepsilon_1^i, \varepsilon_2^i, \varepsilon_4^i)$ für $i = 1, ..., T^\star$ zu generieren, welche sich aus einer Diskretisierung in $T^\star$ Zeitschritte ergeben. Die Spannungskomponenten $\sigma_1^i, \sigma_2^i, \sigma_4^i$ werden mit diesen Dehnungen als Eingabe, wie in Abschnitt 3.2.1 erläutert, bestimmt, wobei $\varepsilon_{eff,pl}^0 = \sigma_1^0 = \sigma_2^0 = \sigma_4^0 = 0$ gesetzt werden. Mit den im Folgenden beschriebenen beiden Methoden wird ein Trainingsdatensatz erzeugt, welcher mit $D^{rand} = D^{lin} \cup D^{sp}$ bezeichnet wird und sich aus den beiden einzelnen Datensätzen D^{lin} und D^{sp} bestehend aus stückweise linearen Dehnpfaden sowie aus Dehnpfaden generiert durch kubische Splines ergibt.

Stückweise lineare Dehnpfade Die stückweise linearen Pfade werden durch die Bestimmung von zufällig gewählten Punkten $p \in \mathbb{R}^3$ im Hauptdehnungsraum generiert. Die Generierung von Punkten im Hauptdehnungsraum bedeutet, dass zunächst nur die Eigenwerte der Dehnungstensoren, welche auch als *Hauptdehnungen* bezeichnet werden, erzeugt und erst in einem darauffolgenden Schritt die Diagonalmatrizen bestehend aus den Hauptdehnungen durch orthogonale Matrizen rotiert werden, um die vollständigen Dehnungstensoren für das Training der ML-Modelle zu generieren.
Für die Bestimmung der Verläufe der Eigenwerte werden zufällig k Punkte $p^0, ..., p^{k-1} \in \mathbb{R}^3$ bestimmt, welche anschließend linear und sequentiell (p^i und p^{i+1} für $i = 0, ..., k-2$) verbunden werden, wobei $p^0 = (0, 0, 0)^T$ gesetzt wird. Für die Bestimmung eines Punktes $p^i = (p_1^i, p_2^i, p_3^i)^T$ werden die Punkte jeweils in Zylinderkoordinatendarstellung

$$p_1^i = r^i \cdot sin(\phi^i) \tag{3.18}$$

$$p_2^i = r^i \cdot cos(\phi^i) \tag{3.19}$$

$$p_3^i = z^i \tag{3.20}$$

betrachtet und die Werte $\phi^i \sim \mathcal{U}(0, 2\pi), z^i \sim \mathcal{U}(a, b)$ und $r^i = l + \mu$ mit $\mu \sim \mathcal{U}(-m, m)$ zufällig gleichverteilt mit vorgegebenen Werten für a, b, l, m generiert. Diese Darstellung durch Zylinderkoordinaten im Hauptdehnungsraum ermöglicht eine Trennung der Form- und Volumenänderung (siehe auch [58]) und daher die Kontrolle über die Volumenänderung. Für volumenkonstante Deformationen kann $z^i = 0$ für $i = 0, ..., k-1$ gesetzt werden. Die stückweise linearen Pfade gegeben durch die Punkte $p^0, ..., p^{k-1} \in \mathbb{R}^3$ werden anhand einer vorgegebenen Anzahl $T^\star \in \mathbb{N}$ an äquidistanten Zeitschritten diskretisiert und die sich daraus ergebenden Eigenwerte $d^t = (d_1^t, d_2^t, d_3^t)^T \in \mathbb{R}^3$ für $t = 0, ..., T^\star$ der Dehnungstensoren, welche in einem darauffolgenden Schritt generiert werden, bestimmt.

Um die einzelnen Dehnungskomponenten $\varepsilon_1, ..., \varepsilon_6$ für das Training zu erzeugen, werden die Diagonalmatrizen $D^t = diag(d_1^t, d_2^t, d_3^t)$ gegeben durch die Eigenwerte für die einzelnen diskretisierten Zeitschritte $t = 0, ..., T^\star$ für jeden zeitlichen Verlauf mittels einer zufällig generierten, orthogonalen Rotationsmatrix $Q \in \mathbb{R}^{3\times3}$ durch

$$\begin{pmatrix} \varepsilon_1^t & \varepsilon_4^t & \varepsilon_6^t \\ \varepsilon_4^t & \varepsilon_2^t & \varepsilon_5^t \\ \varepsilon_6^t & \varepsilon_5^t & \varepsilon_3^t \end{pmatrix} = Q \begin{pmatrix} d_1^t & 0 & 0 \\ 0 & d_2^t & 0 \\ 0 & 0 & d_3^t \end{pmatrix} Q^T \text{ für } t = 0, ..., T^\star \tag{3.21}$$

transformiert. Da für das Training der ML-Materialmodelle ein ebener Dehnungs- und Spannungszustand betrachtet wird, für welchen die Dehnungs- beziehungsweise Spannungskomponenten $\varepsilon_3, \varepsilon_5, \varepsilon_6$ beziehungsweise $\sigma_3, \sigma_5, \sigma_6$ (wie in den Abschnitten 3.2.1 und 3.2.2 näher erläutert) nicht berücksichtigt werden, können Transformationsmatrizen der Form

$$Q = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.22}$$

mit einem zufällig gewählten Winkel $\theta \in [0, 2\pi]$ betrachtet werden. Auf diese Weise wird der mit D^{lin} bezeichnete Datensatz erzeugt, welcher jeweils 3300 Pfade mit einer zeitlichen Diskretisierung in $300, 600$ und 1000 Zeitschritten enthält, wobei $l = 1$ und $m = 0.1$ gewählt werden. Drei Beispielpfade mit den zugehörigen Spannungsverläufen sind in Abbildung 3.7 zu sehen. Abbildung 3.6 zeigt die relative Häufigkeitsverteilung der verschiedenen Dehnungskomponenten für den gesamten Datensatz D^{lin}.

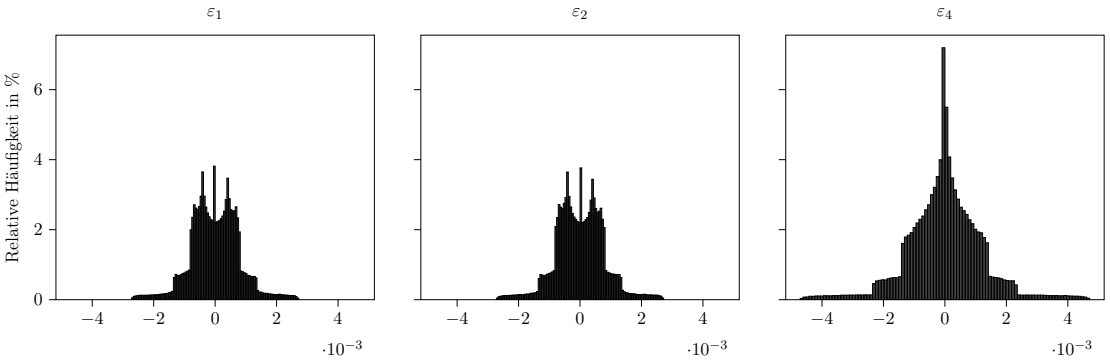


Abbildung 3.6: Häufigkeitsverteilung der jeweiligen Komponenten der Dehnungsinkremente für den Datensatz D^{lin}

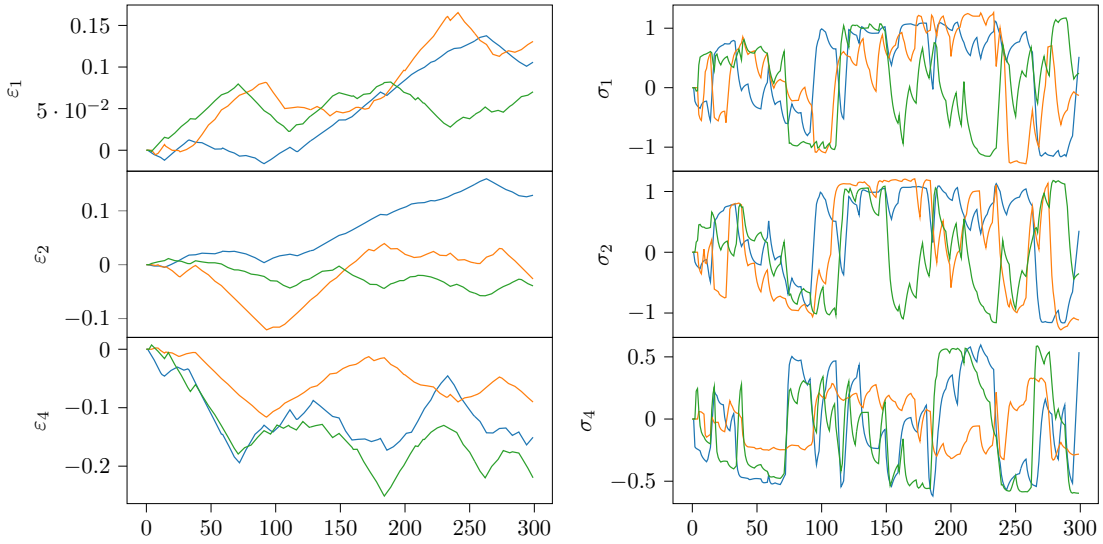


Abbildung 3.7: Beispielverläufe für stückweise lineare Dehnpfade mit 300 Zeitschritten und die zugehörigen Verläufe der Spannungskomponenten

Dehnpfade bestehend aus kubischen Splines Die zeitlichen Verläufe der einzelnen Dehnungskomponenten werden unabhängig voneinander (im $\mathbb{R}^1$) durch kubische Splines erzeugt. Zunächst wird ein Intervall $[x_{start}, x_{ende}]$ sowie die Anzahl $T^\star$ an Zeitschritten, in die der Pfad aufgeteilt wird, festgelegt. Es wird außerdem eine Anzahl $k \leq T^\star$ an Stützpunkten für die Konstruktion der Splines bestimmt.

Das Intervall $[x_{start}, x_{ende}]$ wird in $T^\star$ Punkte ($x_{start} = x_0, ..., x_{T^\star} = x_{ende}$) diskretisiert mit $x_{i+1} - x_i = \frac{x_{T^\star} - x_0}{T^\star - 1}$ für $i = 0, ..., T^\star - 1$. Aus diesen $T^\star$ Punkten werden diskret gleichverteilt (ohne Zurücklegen) $k - 2$ Stützstellen ausgewählt. Für diese k unterschiedlichen Stützstellen (der $k - 2$ gewählten inklusive x_{start} und x_{ende}) werden gleichverteilt aus einem vorgegebenen Intervall $[\varepsilon_{min}, \varepsilon_{max}]$ zugehörige Funktionswerte für die Stützstellen bestimmt. Mit diesen Stützstellen werden die (eindeutig festgelegten) normierten kubischen B-Splines bestimmt und anschließend für jeden Zeitpunkt die Werte für die Dehnung berechnet. Für die in dieser Arbeit genutzten Trainingsdaten wird das Intervall für die Funktionswerte der Stützstellen durch $[\varepsilon_{min}, \varepsilon_{max}] = [-0.3, 0.3]$ festgelegt. Des Weiteren wird ein Intervall von $[x_{start}, x_{ende}] = [0, 0.5]$ und $k = 6$ Stützstellen als geeignete Flexibilität der Pfade im Vergleich zu in Anwendungen vorkommenden Verläufen betrachtet. Mit diesen Parametern wird der Datensatz D^{sp}, bestehend aus jeweils 3300 Pfaden mit einer Diskretisierung in 300, 600 und 1000 Zeitschritte, erzeugt. Drei Beispielpfade mit den zugehörigen Spannungsverläufen sind in Abbildung 3.8 zu sehen. Abbildung 3.9 zeigt die relative Häufigkeitsverteilung der verschiedenen Dehnungskomponenten des gesamten

Datensatzes D^{sp}.

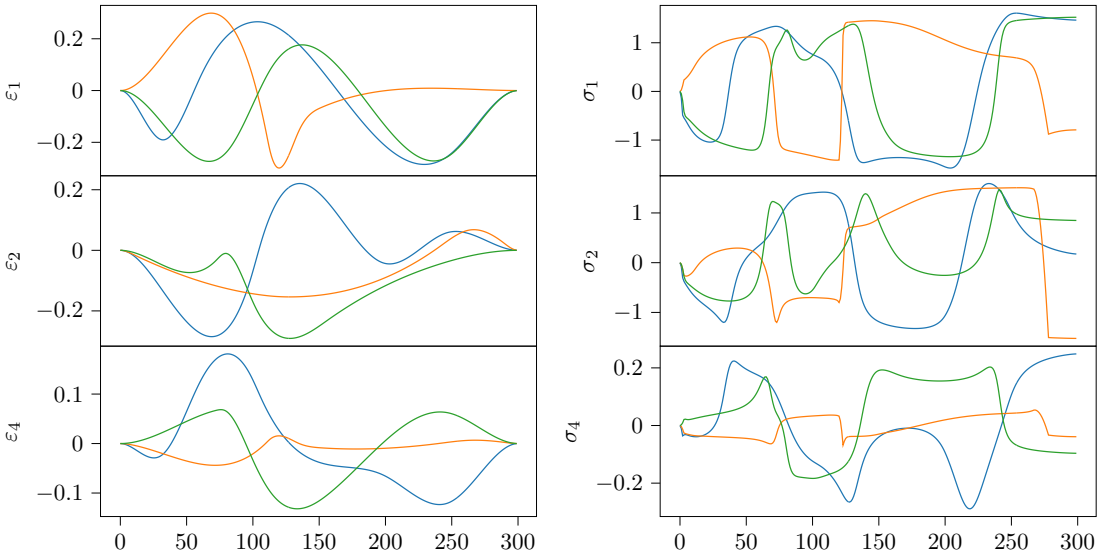


Abbildung 3.8: Beispielverläufe für Dehnpfade bestehend aus Splines mit 300 Zeitschritten und die zugehörigen Verläufe der Spannungskomponenten

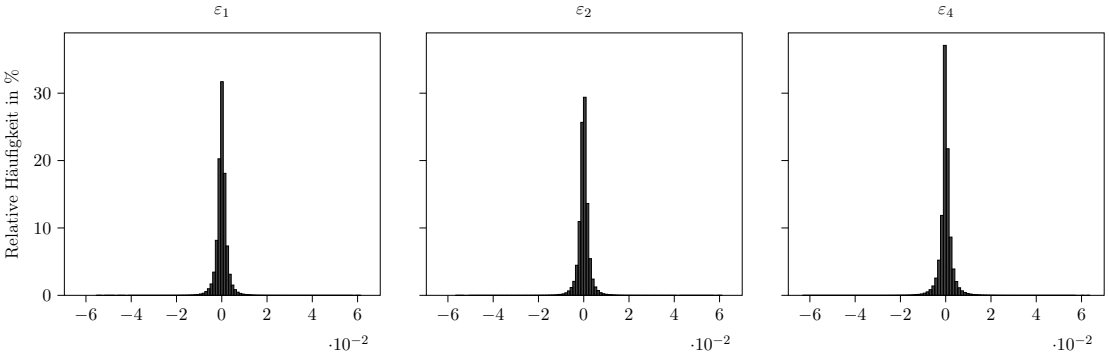


Abbildung 3.9: Häufigkeitsverteilung der jeweiligen Komponenten der Dehnungsinkremente für den Datensatz D^{sp}

Analyse und Vergleich der Daten mit einem Testdatensatz aus einer Hutprofil Simulation

Für die Validierung und den Vergleich der verschiedenen ML-Materialmodelle in Abschnitt 3.2.6 wird ein Datensatz D^{hat} aus einer Hutprofil Simulation (siehe Abbildung 3.10) betrachtet. In der Hutprofil Simulation trifft ein Impaktor (gelb), welcher als Starrkörper modelliert wird mit einer initialen translatorischen Geschwindigkeit von $-20\frac{m}{s}$ in z-Richtung auf das Hutprofil (grau), welches mit dem klassischen Materialmodell gerechnet

wird. Es werden für 640 Integrationspunkte, verteilt auf dem Profil, für jeden der 26100 Zeitschritte die Dehnungs- und Spannungskomponenten sowie die plastische Dehnung extrahiert.

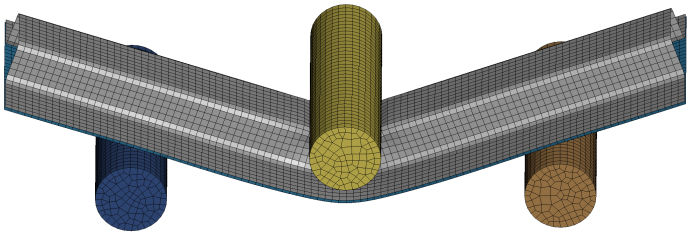


Abbildung 3.10: Hutprofil Simulation für den Validierungsdatensatz D^{hat}

	Spline-Pfade D^{sp}	Stückweise lineare Pfade D^{lin}
Mittelwert $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(-4.8 \cdot 10^{-22}, -1.4 \cdot 10^{-21}, 5.7 \cdot 10^{-22}, 2.6 \cdot 10^{-22})$	$(-2.9 \cdot 10^{-6}, 1.3 \cdot 10^{-6}, -9.9 \cdot 10^{-7}, -8.7 \cdot 10^{-7})$
Varianz $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(6.2 \cdot 10^{-6}, 6.2 \cdot 10^{-6}, 8.2 \cdot 10^{-6}, 6.9 \cdot 10^{-6})$	$(6.3 \cdot 10^{-7}, 6.3 \cdot 10^{-7}, 1.5 \cdot 10^{-6}, 9.2 \cdot 10^{-7})$
Schiefe $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(5.7 \cdot 10^{-2}, -2.7 \cdot 10^{-2}, -3.2 \cdot 10^{-2}, -5.2 \cdot 10^{-3})$	$(2.6 \cdot 10^{-2}, 5.1 \cdot 10^{-2}, 6 \cdot 10^{-3}, 1.9 \cdot 10^{-2})$

	Gesamtdatensatz aus zufällig generierten Pfaden D^{rand}	Datensatz aus Hutprofil Simulation D^{hat}
Mittelwert $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(-1.5 \cdot 10^{-6}, 6.5 \cdot 10^{-7}, -5.0 \cdot 10^{-7}, -4.4 \cdot 10^{-7})$	$(-3.4 \cdot 10^{-8}, -2.4 \cdot 10^{-8}, -5.9 \cdot 10^{-8}, -3.9 \cdot 10^{-8})$
Varianz $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(3.4 \cdot 10^{-6}, 3.4 \cdot 10^{-6}, 4.8 \cdot 10^{-6}, 3.9 \cdot 10^{-6})$	$(9.1 \cdot 10^{-9}, 1.0 \cdot 10^{-8}, 2.7 \cdot 10^{-9}, 7.3 \cdot 10^{-9})$
Schiefe $(\varepsilon_1, \varepsilon_2, \varepsilon_4, \text{gesamt})$	$(7.3 \cdot 10^{-2}, -3.2 \cdot 10^{-2}, -3.4 \cdot 10^{-2}, -4.5 \cdot 10^{-3})$	$(-6.4 \cdot 10^{-3}, 4.2 \cdot 10^{-3}, 1.4 \cdot 10^{-2}, 6.6 \cdot 10^{-4})$

Tabelle 3.2: Vergleich der unterschiedlichen Datensätze bezüglich Mittelwert, Varianz und Schiefe

Abbildung 3.11 zeigt drei Beispielpfade für die ersten 300 Zeitschritte mit den zugehörigen Spannungsverläufen. Betrachtet man die Häufigkeitsverteilung der verschiedenen Dehnungskomponenten für die drei Datensätze D^{lin}, D^{sp} und D^{hat} (siehe Abbildung 3.12), so sind alle Datensätze symmetrisch nahe null verteilt, was auch an der empirischen Schiefe und den Mittelwerten in Tabelle 3.2 erkennbar wird. Im Vergleich zu den Trainingsdatensätzen weist der Datensatz D^{hat} eine höhere Häufung von Werten nahe null in jeder der Eingangskomponenten und eine geringere Varianz sowie kleinere Werte in der plastischen Dehnung (insbesondere ein höherer Anteil an Nullwerten) auf.

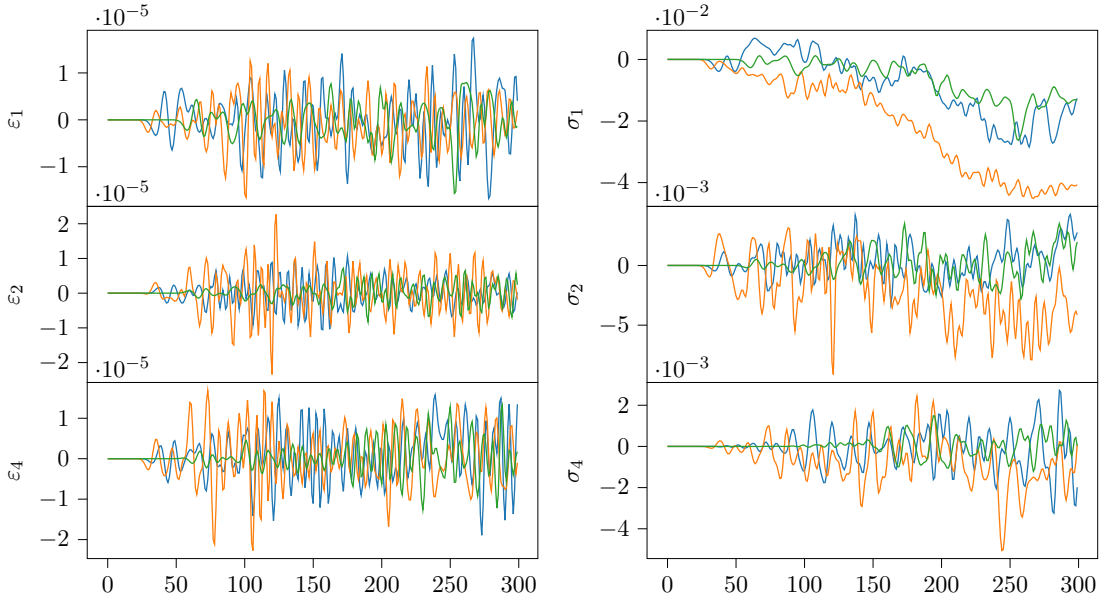


Abbildung 3.11: Beispielverläufe für Dehnpfade aus der Hutprofil Simulation für die ersten 300 Zeitschritte mit den zugehörigen Verläufen der Spannungskomponenten

Um zusätzlich eine Abdeckung der Spannungszustände in den Trainingsdatensätzen zu berücksichtigen, kann die *Triaxialität*, welche den Beanspruchungszustand beschreibt [51], betrachtet werden. Die Triaxialität gibt das Verhältnis des volumentrischen Anteils des Spannungstensors (Volumenänderung) zur Orientierung des Spannungszustandes an und berechnet sich durch den Mittelwert der Diagonaleinträge des Spannungstensors im Verhältnis zu der von Mises Spannung

$$Tri(\sigma) = \frac{1}{3 \cdot \sigma_{v,M}}(\sigma_{xx} + \sigma_{yy} + \sigma_{zz}) \in [-0.6, 0.6]. \tag{3.23}$$

Ist der Wert für die Triaxialität negativ, befindet sich der Spannungszustand im Druckbereich und bei einem positiven Wert im Zugbereich. Die Abbildung 3.13 zeigt für die unterschiedlichen Datensätze die Triaxialität der einzelnen Datenpunkte (Spannungszustände unabhängig der Zeitschritte) bezüglich den unterschiedlichen Größen der plastischen Dehnung für jeden hundertsten Datenpunkt. Eine möglichst große Abdeckung dieser beiden Größen wird auch für die Entwicklung von Charakterisierungsversuchen herangezogen (siehe [38]), unter anderem um Datensätze für das Training von ML-Materialmodellen anhand von Versuchsdaten, zu generieren, wie es in Kapitel 4 beschrieben wird. Erkennbar ist bei allen Datensätzen eine gute Abdeckung des gesamten Belastungsbereichs (sowohl im Zug- als auch im Druckbereich). Die Trainingsdatensätze weisen (im Vergleich zur

Hutprofil Simulation) auch bei größeren plastischen Dehnungen eine gute Abdeckung auf.

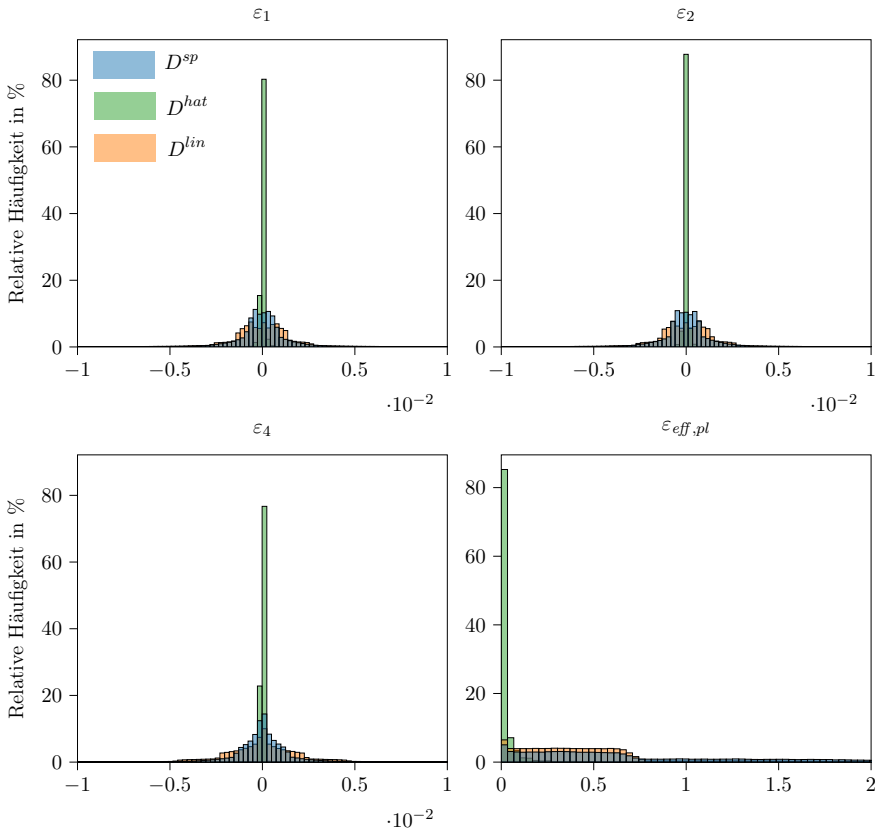


Abbildung 3.12: Häufigkeitsverteilung der jeweiligen Komponenten der Dehnungsinkremente für die verschiedenen Datensätze

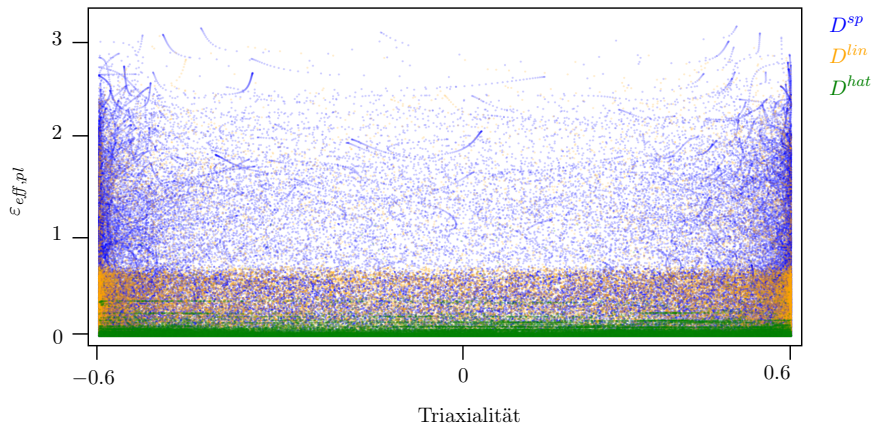


Abbildung 3.13: Abdeckung der Spannungszustände gemessen an der Triaxialität und der plastischen Dehnung bezüglich der verschiedenen Datensätze

3.2.4 Training von ML-Materialmodellen ohne Geschichtsvariablen

Zunächst werden unterschiedliche ML-Modelle ohne die Hinzunahme der plastischen Dehnung $\varepsilon_{eff,pl}$ trainiert. Hierfür wird auf den gesamten zeitlichen Verläufen der Dehn- und Spannungspfade trainiert. In jedem Zeitschritt bekommt das ML-Modell hierbei die Eingabewerte $\varepsilon_1^t, \varepsilon_2^t, \varepsilon_4^t$ in inkrementeller Form mit dem Ziel die Ausgaben $\sigma_1^t, \sigma_2^t, \sigma_4^t$ vorherzusagen. Für das Training der Modelle wird der Datensatz D^{rand}, wie in Abschnitt 3.2.3 beschrieben, genutzt, wobei dieser für die Evaluierung der Modelle in eine Test- und Trainingsteilmenge D_{train}^{rand} und D_{test}^{rand} aufgeteilt wird. Hierfür werden für den Testdatensatz 20% der Datenpunkte zufällig aus dem Gesamtdatensatz ausgewählt. Da in Simulationen die Anzahl an Zeitschritten und die Anzahl an Elementen sehr groß werden kann, werden ausschließlich ML-Modelle betrachtet, welche die Daten sequentiell bearbeiten und ohne das Training mit Zeitfenstern auskommen (siehe Erläuterungen in Abschnitt 3.2.2), wie es beispielsweise bei Transformer-Modellen oder Convolutional Neural Networks mit der Anwendung für Zeitreihen der Fall ist. In der Anwendung müssten für diese Art von ML-Modellen die unterschiedlichen Dehnungskomponenten für den bisherigen Verlauf für jeden Integrationspunkt zwischengespeichert werden, was beispielsweise in Crashsimulationen für ein Gesamtfahrzeug hunderttausende von Zeitschritten für mehre Millionen von Elementen wären [45].

Die Modelle in diesem Abschnitt wurden mithilfe der Python-Bibliotheken Keras, Tensorflow [18, 1] implementiert und trainiert. Um die rekurrenten neuronalen Netze mit Zeitreihen unterschiedlicher Längen trainieren zu können, werden die kürzeren Pfade durch Padding ergänzt. Dabei werden die Zeitreihen mittels einer hinreichend groß gewählten Konstanten (hier 10^{10}) verlängert und im Training eine zusätzliche Masking-Schicht eingesetzt, welche bewirkt, dass diese zusätzlichen Bereiche der Zeitreihe im Training nicht berücksichtigt werden. Für die Auswertung der Modelle werden die mittleren quadratischen Abweichungen (MSE) bezüglich unterschiedlicher Ausgabewerte betrachtet und wie folgt bezeichnet:

$$\text{MSE} = \frac{1}{T} \cdot \frac{1}{N} \cdot \frac{1}{3} \sum_{t=1}^{T} \sum_{n=1}^{N} \sum_{i \in \{1,2,4\}} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \tag{3.24}$$

$$\text{MSE}_{\sigma_i} \coloneqq \frac{1}{T} \cdot \frac{1}{N} \sum_{t=1}^{T} \sum_{n=1}^{N} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \text{ für } i = 1, 2, 4 \tag{3.25}$$

$$\text{MSE}_t \coloneqq \frac{1}{N} \cdot \frac{1}{3} \sum_{n=1}^{N} \sum_{i \in \{1,2,4\}} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \text{ für } t = 1, ..., T, \tag{3.26}$$

wobei mit $\hat{\sigma}_i^{n,t}$ die Vorhersage des ML-Modells für die jeweilige Spannungskomponente σ_i eines Zeitschrittes t und eines Datenpfades n bezeichnet wird.

Training rekurrenter neuronaler Netze

Für das Training unterschiedlicher rekurrenter Standardnetze wird eine Hyperparameterstudie auf Basis des Trainingsdatensatzes D_{train}^{rand} durchgeführt. Die unterschiedlichen Hyperparameter werden simultan variiert und zufällig (*Random Search*) verschiedene Konfigurationen aus dem gesamten Suchraum ausgewählt. Hierbei werden Modelle, bestehend sowohl aus LSTM als auch GRU-Zellen (siehe Abschnitt 2.3.3), betrachtet. Die Größe der neuronalen Netze bezüglich der Anzahl der trainierbaren Parameter wird durch die unterschiedliche Anzahl an Schichten und Neuronen je Schicht variiert. Außerdem wird getestet, ob eine zusätzliche Dense-Schicht (vollständig verbundene Schicht) vor der finalen Ausgabe zu einer verbesserten Prognosefähigkeit der ML-Modelle führt. Weiterhin werden Parameter variiert, welche den Trainingsprozess der neuronalen Netze betreffen, wie beispielsweise die Lernrate, die gewählte Fehlerfunktion und die Skalierung der Trainingsdaten. Die Anwendung des Gradienten erfolgt für jede Konstellation mit dem Adam-Optimierer [43]. Die Modelle werden jeweils für 300 Epochen mit einer Batchgröße von 1500 trainiert. Tabelle 3.3 zeigt eine Übersicht der untersuchten Hyperparameter und deren betrachteten Wertebereiche.

Hyperparameter	**Wertebereich**
Schichtart	LSTM, GRU
Anzahl an rekurrenten Schichten	1, 2, 3, 4
Anzahl an Dense-Schichten	1, 2
Anzahl an Neuronen je Schicht	15, 30, 50, 75, 100, 150, 200
Aktivierungsfunktion Dense-Schicht	linear, tanh, LeakyReLU, sigmoid
Fehlerfunktion im Training	quadratisch (MSE), absolut (MAE)
Skalierung	StandardScale, MaxAbsScale, MinMaxScale

Tabelle 3.3: Untersuchte Hyperparameter für das Training von LSTM und GRU-Modellen

In den Abbildungen 3.14 und 3.15 sind Streudiagramme, welche den mittleren quadratischen Fehler der trainierten Modelle getrennt nach den unterschiedlich gewählten Hyperparametern angeben, zu sehen. Um die trainierten ML-Modelle trotz verschiedener Skalierungsmethoden vergleichbar zu machen, werden die Fehlermetriken grundsätzlich bezüglich der unskalierten Daten angegeben. Jeder Punkt im Streudiagramm entspricht einem trainierten Modell mit der entsprechenden Prognosefähigkeit gemessen an dem

mittleren quadratischen Fehler des Testdatensatzes MSE D_{test}^{rand}.

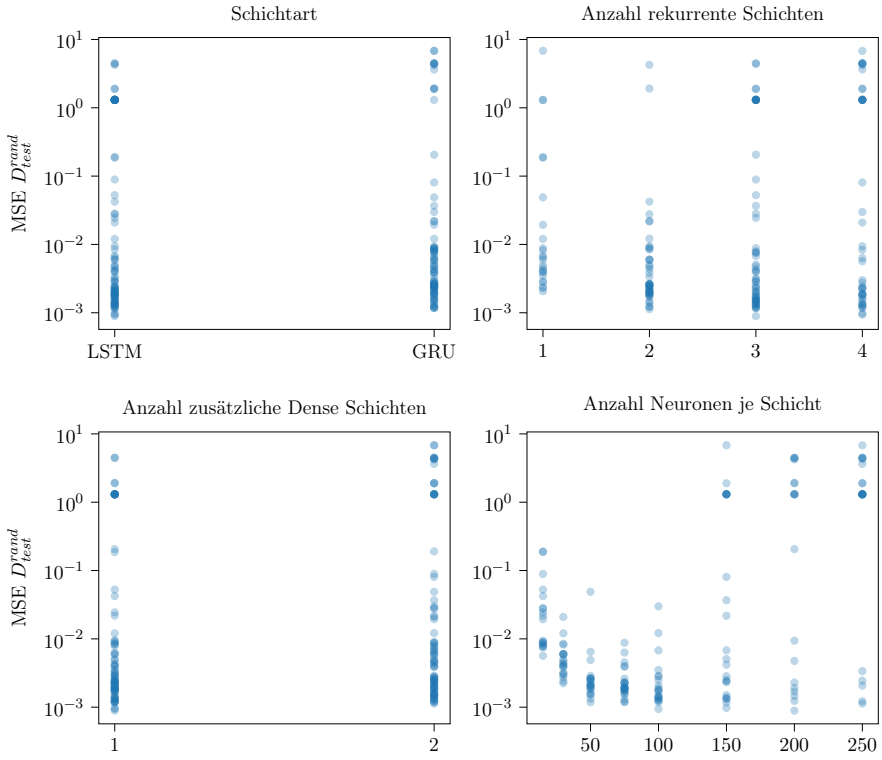


Abbildung 3.14: MSE D_{test}^{rand} der einzelnen Konstellationen an Hyperparametern, welche die Modellstruktur des neuronalen Netzes betreffen. Einen positiven Einfluss auf die MSE D_{test}^{rand}-Werte haben in erster Linie die Hyperparameter, welche die Anzahl an trainierbaren Parameter der neuronalen Netze erhöhen (Anzahl rekurrenter Schichten und Neuronen je Schicht).

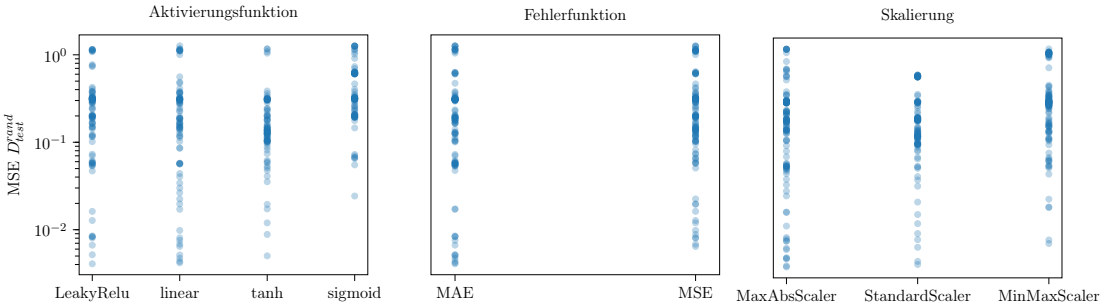


Abbildung 3.15: MSE D_{test}^{rand} der einzelnen Konstellationen an Hyperparametern (Aktivierungsfunktion, Fehlerfunktion für den Trainingsprozess und Skalierung)

In Abbildung 3.14 ist zu sehen, dass beide Arten von rekurrenten Schichten (GRU und

LSTM) in der Studie ähnlich gute Ergebnisse liefern. Modelle, welche mit der absoluten Abweichung (MAE) trainiert wurden, können einen leicht kleineren Fehler bezüglich der Testdaten aufweisen. Weiterhin ist zu erkennen, dass weder eine zusätzliche Schicht vor der Ausgabe der Spannungen noch eine andere Aktivierungsfunktion als die lineare Aktivierungsfunktion dieser Schicht zu einer besseren Prognosefähigkeit der Modelle führt. Die Modelle, bei denen die Daten für das Training mit dem StandardScaler und MaxAbsScaler transformiert werden, liefern etwas bessere Prognosen als bei einer Skalierung mit dem MinMaxScaler. Die Abbildung 3.16 zeigt außerdem, dass eine ausreichend große Anzahl an trainierbaren Parametern (im vierstelligen Bereich) der Modelle benötigt wird, um eine gute Prognosefähigkeit (MSE $D_{test}^{rand} \leq 10^{-3}$) zu erreichen.

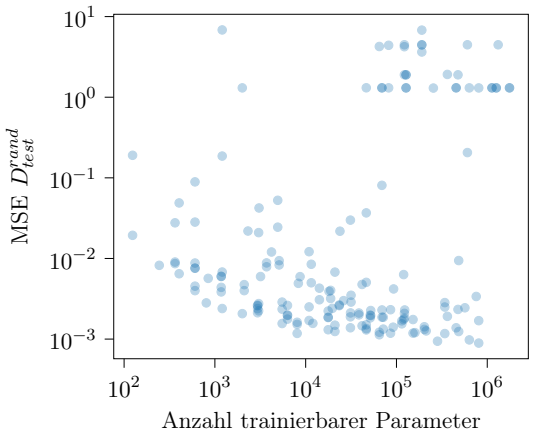


Abbildung 3.16: MSE D_{test}^{rand} der verschiedenen LSTM und GRU-Modelle nach der Gesamtanzahl an trainierbaren Parametern des neuronalen Netzes. Jeder Punkt im Diagramm entspricht einer getesteten Konfiguration an Hyperparametern.

Linearized Minimal State Cell (LMSC) Modelle

Als Alternative zu den herkömmlichen rekurrenten neuronalen Netzen werden die Linearized Minimal State Cell (LMSC) Modelle aus [11] (siehe Abschnitt 2.3.3) betrachtet. Auch für diese Modelle werden verschiedene Hyperparameter variiert (siehe Tabelle 3.4), wobei bei der Skalierung auf Methoden mit einer Translation der Daten (wie beispielsweise bei einer Standardisierung mit einem Mittelwert ungleich null) verzichtet wird, da in diesem Fall die Nullzustände nicht mehr als solche in das Modell eingehen (siehe Bemerkung 2.3.1). Die Eigenschaft einer exakten Vorhersage der LMSC-Modelle von Spannungen gleich null bei einer Eingabe von Dehnungskomponenten gleich null wäre nicht mehr gegeben. Diese Eigenschaft spielt jedoch, wie es in Abschnitt 3.2.7 später näher gezeigt

wird, eine sehr wichtige Rolle bei der Anwendung der ML-Materialmodelle innerhalb der FE-Simulationen. In Abbildung 3.17 ist zu sehen, dass die LMSC-Modelle vergleichsweise sehr robust bezüglich unterschiedlicher Hyperparameter sind, da eine Variation des Netzaufbaus kaum zu einer Änderung des Fehlers MSE D_{test}^{rand} führt. Insbesondere genügen tendenziell weniger trainierbare Parameter für eine gute Anpassung der Modelle als bei den LSTM und GRU-Modellen, was auf die kleinere Anzahl an (Gedächtnis-)Zuständen zurückgeführt werden kann.

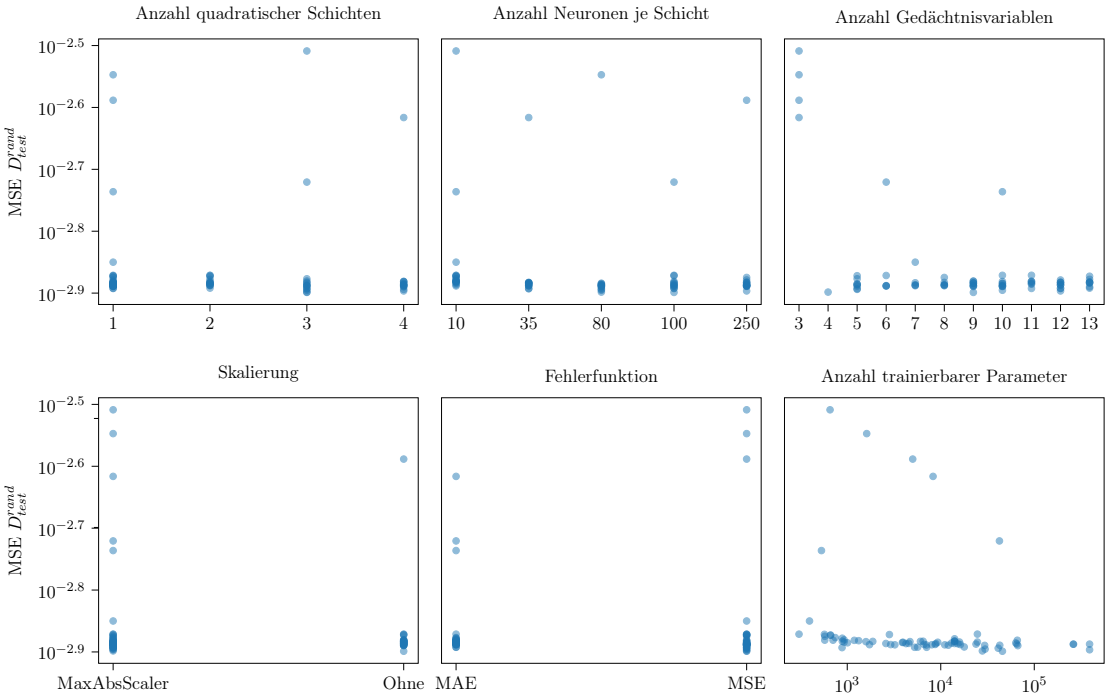


Abbildung 3.17: MSE D_{test}^{rand} der einzelnen Konstellationen an Hyperparametern bei den LMSC-Modellen

Hyperparameter	**Wertebereich**
Anzahl an quadratischen Schichten	1, 2, 3, 4
Anzahl an Neuronen je Schicht	10, 35, 80, 100, 250
Anzahl an Zustands-/Gedächtnisvariablen	$n \in \mathbb{N}$ mit $3 \leq n \leq 13$
Fehlerfunktion im Training	quadratisch (MSE), absolut (MAE)
Skalierung	Ohne, MaxAbsScale

Tabelle 3.4: Untersuchte Hyperparameter für das Training von LMSC-Modellen

Das klassische Materialmodell gibt insgesamt vier Werte (σ_i^{t-1}, $i = 1, 2, 4$ und $\varepsilon_{eff,pl}^{t-1}$) des vorhergegangenen Zeitschrittes als Eingabe in den aktuellen Zeitschritt. Es werden daher beim klassischen Materialmodell vier zwischengespeicherte Zustandswerte benötigt. Die Abbildung 3.17 zeigt, dass auch bei den ML-Modellen eine deutlich bessere Prognosefähigkeit (geringere Werte des MSE D_{test}^{rand}) für mindestens vier gespeicherte Zustände zu sehen ist.

3.2.5 Training von ML-Materialmodellen mit Geschichtsvariablen

Im folgenden Abschnitt werden ML-Modelle für den DP800 Stahl trainiert, welche die internen Geschichtsvariablen des klassischen Materialmodells direkt mitberücksichtigen. In diesem Fall werden den ML-Modellen im Training nicht die zeitlichen Verläufe der Dehnungen und Spannungen zur Verfügung gestellt, sondern unabhängig für je einen beliebig festen Zeitschritt t ein Regressionsmodell betrachtet, welches als Eingabe die Dehnungskomponenten $\varepsilon_1^t, \varepsilon_2^t, \varepsilon_4^t$ in inkrementeller Form, die zugehörigen Spannungskomponenten $\sigma_1^{t-1}, \sigma_2^{t-1}, \sigma_4^{t-1}$ in absoluter Form sowie die plastische Dehnung $\varepsilon_{eff,pl}^{t-1}$ des vorherigen Zeitschrittes und als Ausgabe die absoluten Spannungen $\sigma_1^t, \sigma_2^t, \sigma_4^t$ berücksichtigt. Für das Training dieser Modelle wird, wie auch für das Training der rekurrenten Modelle, der Trainingsdatensatz D_{train}^{rand} und der Testdatensatz D_{test}^{rand} betrachtet. Die Modelle werden mithilfe der Python-Bibliotheken Keras, Tensorflow und Scikit-Learn [18, 1, 59] implementiert und trainiert. Für die Auswertung werden die mittleren quadratischen Abweichungen (MSE) der Ausgabewerte betrachtet und die folgenden Notationen genutzt:

$$\text{MSE} = \frac{1}{T} \cdot \frac{1}{N} \cdot \frac{1}{4} \sum_{t=1}^{T} \sum_{n=1}^{N} \left((\hat{\varepsilon}_{eff,pl}^{n,t} - \varepsilon_{eff,pl}^{n,t})^2 + \sum_{i \in \{1,2,4\}} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \right) \tag{3.27}$$

$$\text{MSE}_\sigma := \frac{1}{T} \cdot \frac{1}{N} \cdot \frac{1}{3} \sum_{t=1}^{T} \sum_{n=1}^{N} \sum_{i \in \{1,2,4\}} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \tag{3.28}$$

$$\text{MSE}_{\varepsilon_{eff,pl}} := \frac{1}{T} \cdot \frac{1}{N} \sum_{t=1}^{T} \sum_{n=1}^{N} (\hat{\varepsilon}_{eff,pl}^{n,t} - \varepsilon_{eff,pl}^{n,t})^2 \tag{3.29}$$

$$\text{MSE}_{\sigma_i} := \frac{1}{T} \cdot \frac{1}{N} \sum_{t=1}^{T} \sum_{n=1}^{N} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \text{ für } i = 1, 2, 4 \tag{3.30}$$

$$\text{MSE}_t := \frac{1}{N} \cdot \frac{1}{4} \sum_{n=1}^{N} \left((\hat{\varepsilon}_{eff,pl}^{n,t} - \varepsilon_{eff,pl}^{n,t})^2 + \sum_{i \in \{1,2,4\}} (\hat{\sigma}_i^{n,t} - \sigma_i^{n,t})^2 \right) \text{ für } t = 1, ..., T, \tag{3.31}$$

wobei mit $\hat{\sigma}_i^{n,t}$ beziehungsweise $\hat{\varepsilon}_{eff,pl}^{n,t}$ die Vorhersage des ML-Modells für die jeweilige Spannungskomponente beziehungsweise für die plastischen Dehnung eines Zeitschrittes t und eines Datenpfades n bezeichnet wird. Analog werden die mittleren absoluten Abweichungen (MAE) und R^2-Werte betrachtet.

Modellierung durch Regressionsmethoden

Für die Modellierung des klassischen Materialmodells werden im Folgenden unterschiedliche klassische Regressionsmethoden, wie in Abschnitt 2.3.1 eingeführt, untersucht. Zunächst wird als Ausgangspunkt ein einfaches lineares Regressionsmodell anhand des Datensatzes D_{train}^{rand} angepasst. Das Modell erreicht im Training bereits einen R^2-Wert von 0.99541 und einen MSE-Wert von $2.019 \cdot 10^{-3}$ und für den Testdatensatz einen R^2-Wert von 0.99538 und einen MSE-Wert von $2.031 \cdot 10^{-3}$.

Als Erweiterung zur linearen Regression werden Transformationen der Eingaben durch B-Splines betrachtet. Hierfür wird der Wertebereich für die jeweils einzelnen Eingaben $j = 1, ..., 7$ des Modells in Abschnitte eingeteilt, welche durch die vorgegebene Anzahl an Knoten N^{nod} festgelegt wird. Dies geschieht hier entweder durch eine uniforme Verteilung der Knoten $t_1^j, ..., t_{N^{nod}}^j$ an den Quantilen (*quantile Knotensetzung*) oder durch eine lediglich uniforme Verteilung der Knoten in äquidistante Abschnitte (*uniforme Knotensetzung*).

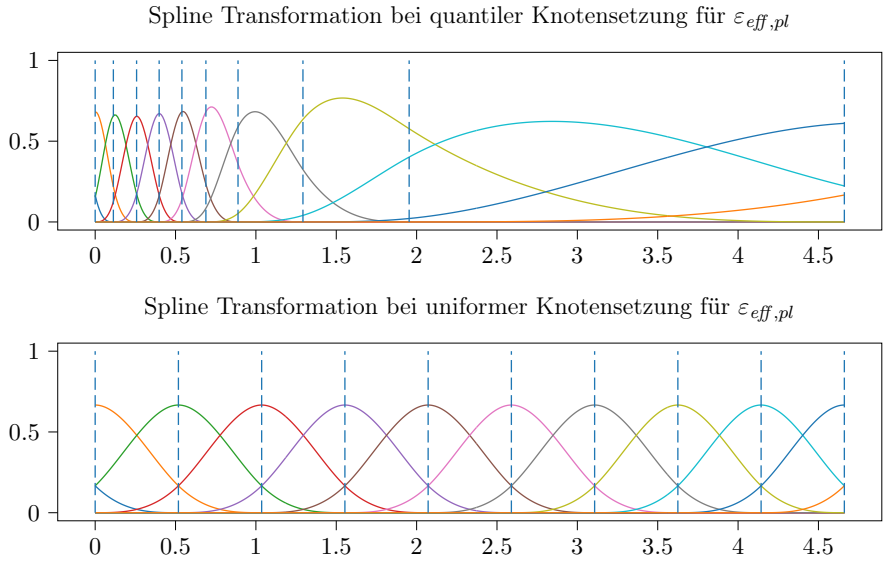


Abbildung 3.18: Vergleich der quantilen und uniformen Knotensetzung für $\varepsilon_{eff,pl}$ bei 10 Knoten und kubischen Splines

Weiterhin wird der Grad der Polynome *deg* festgelegt. Abhängig von diesen Knotenpunkten können daraufhin die Koeffizienten der $N^{nod} + deg - 1$ abschnittsweisen Polynome (normierten B-Spline Basisfunktionen) $B_1^j, ..., B_{N^{nod}+deg-1}^j$ für jeden Abschnitt und jeden Eingabewert des Modells eindeutig bestimmt werden. Jeder Eingabepunkt $x = (x_1, ..., x_7)^T := (\varepsilon_1^t, \varepsilon_2^t, \varepsilon_4^t, \sigma_1^{t-1}, \sigma_2^{t-1}, \sigma_4^{t-1}, \varepsilon_{eff,pl}^{t-1})^T \in \mathbb{R}^7$ wird anhand dieser B-Splines in $(N^{nod} + deg - 1) \cdot 7$ Eingabewerte $(B_i^j(x_j))_{i=1,...,N^{nod}+deg-1, j=1,...,7}$ transformiert und mit diesen transformierten Eingaben ein lineares Regressionsmodell angepasst.

Abbildung 3.19 zeigt die Spline Transformationen für $N^{nod} = 5$ und $deg = 3$ bei einer quantilen Knotensetzung für die jeweiligen Eingabekomponenten für die Spannungen und Dehnungsinkremente und Abbildung 3.18 die Transformationen für $deg = 3$ bei einer quantilen und einer uniformen Knotensetzung mit 10 Knoten für den Eingabewert $\varepsilon_{eff,pl}$.

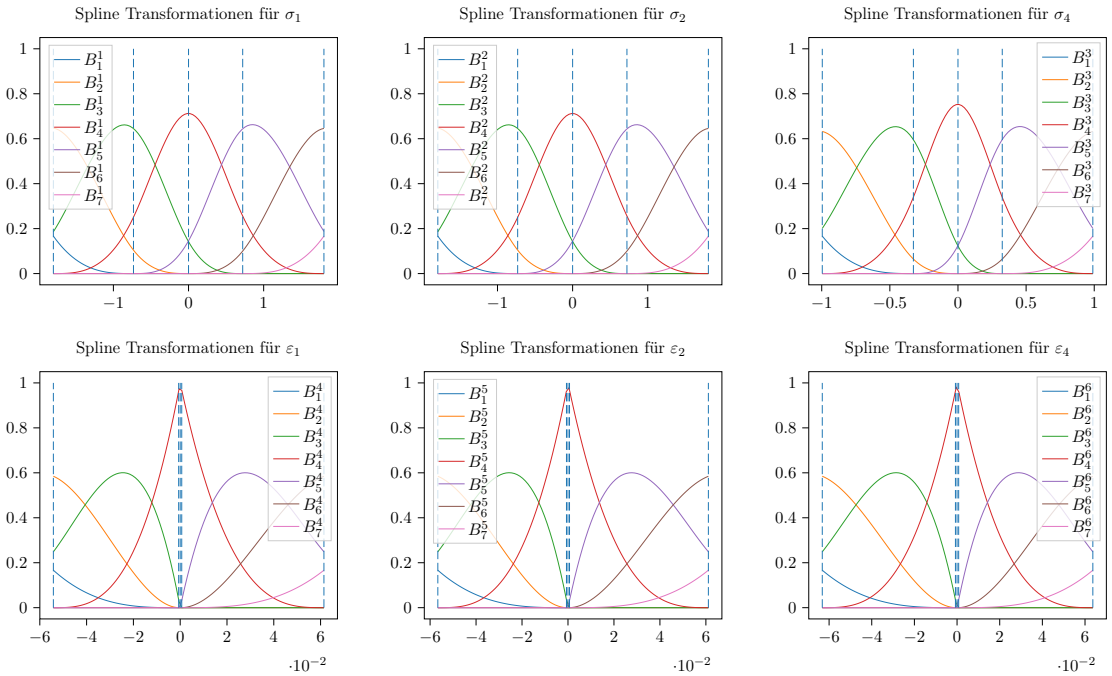


Abbildung 3.19: Die verschiedenen Spline Transformationen für die jeweiligen Eingabewerte $\sigma_1, \sigma_2, \sigma_4, \varepsilon_1, \varepsilon_2, \varepsilon_4$ bei 5 quantil-gesetzten Knoten und kubischen Splines ($deg = 3$). Die Ähnlichkeit der Splines für die einzelnen Komponenten ergibt sich durch die ähnliche Häufigkeitsverteilung der verschiedenen Komponenten (siehe Abschnitt 3.2.3).

In Abbildung 3.20 sind die MSE-Fehlerwerte beziehungsweise die R^2-Werte der angepassten Modelle mit unterschiedlichen Werten für N^{nod} und *deg* bezüglich der Trainings- und Testdaten zu sehen. Den kleinsten MSE-Fehlerwert zeigt das Spline-Regressionsmodell mit

$N^{nod} = 50$ und $deg \in \{2, 3\}$, wobei bereits ab $N^{nod} = 10$ und für $deg \geq 1$ nicht wesentlich größere Fehlerabweichungen zu sehen sind. Deutlich schlechter ist die Vorhersage lediglich (wie auch zu erwarten) für $deg = 0$, sprich für Funktionen, welche nur konstant sind.
Im Vergleich zu dem eingangs genannten einfachen linearen Regressionsmodell mit einer Abweichung von MSE $D_{test}^{rand} = 2.02 \cdot 10^{-3}$, bieten damit die Spline-Regressionsmodelle mit bestenfalls einer Abweichung von MSE $D_{test}^{rand} = 1.33 \cdot 10^{-3}$ eine nur leicht bessere Modellierungsmethode für das klassische elastoplastische Materialmodell.

	0	1	2	3
5	$4.25 \cdot 10^{-2}$	$1.36 \cdot 10^{-3}$	$1.36 \cdot 10^{-3}$	$1.84 \cdot 10^{-3}$
10	$1.25 \cdot 10^{-2}$	$1.34 \cdot 10^{-3}$	$1.34 \cdot 10^{-3}$	$1.34 \cdot 10^{-3}$
50	$2.12 \cdot 10^{-3}$	$1.32 \cdot 10^{-3}$	$1.32 \cdot 10^{-3}$	$1.32 \cdot 10^{-3}$

(a) MSE für D_{train}^{rand} bei quantiler Knotensetzung

	0	1	2	3
5	$4.26 \cdot 10^{-2}$	$1.37 \cdot 10^{-3}$	$1.36 \cdot 10^{-3}$	$1.85 \cdot 10^{-3}$
10	$1.25 \cdot 10^{-2}$	$1.34 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$1.34 \cdot 10^{-3}$
50	$2.13 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$	$1.33 \cdot 10^{-3}$

(b) MSE für D_{test}^{rand} bei quantiler Knotensetzung

	0	1	2	3
5	$4.8 \cdot 10^{-2}$	$1.98 \cdot 10^{-3}$	$1.95 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$
10	$1.35 \cdot 10^{-2}$	$1.94 \cdot 10^{-3}$	$1.96 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$
50	$2.27 \cdot 10^{-3}$	$1.57 \cdot 10^{-3}$	$1.55 \cdot 10^{-3}$	$1.52 \cdot 10^{-3}$

(c) MSE für D_{train}^{rand} bei uniformer Knotensetzung

	0	1	2	3
5	$4.8 \cdot 10^{-2}$	$1.99 \cdot 10^{-3}$	$1.96 \cdot 10^{-3}$	$1.99 \cdot 10^{-3}$
10	$1.35 \cdot 10^{-2}$	$1.95 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$
50	$2.28 \cdot 10^{-3}$	$1.58 \cdot 10^{-3}$	$1.56 \cdot 10^{-3}$	$1.56 \cdot 10^{-3}$

(d) MSE für D_{test}^{rand} bei uniformer Knotensetzung

	0	1	2	3
5	0.918731	0.997026	0.997069	0.995885
10	0.976194	0.997123	0.997119	0.997121
50	0.995627	0.997145	0.997146	0.997145

(e) R^2-Werte für D_{train}^{rand} bei quantiler Knotensetzung

	0	1	2	3
5	0.918729	0.997006	0.99705	0.995872
10	0.976131	0.997104	0.997099	0.997103
50	0.995596	0.997125	0.997127	0.997126

(f) R^2-Werte für D_{test}^{rand} bei quantiler Knotensetzung

	0	1	2	3
5	0.906766	0.995482	0.995647	0.995518
10	0.973053	0.995612	0.995537	0.995732
50	0.994895	0.996625	0.996531	0.996716

(g) R^2-Werte für D_{train}^{rand} bei uniformer Knotensetzung

	0	1	2	3
5	0.906747	0.995452	0.995624	0.995491
10	0.97304	0.995585	0.995509	0.995706
50	0.99487	0.996604	0.996511	0.996639

(h) R^2-Werte für D_{test}^{rand} bei uniformer Knotensetzung

Abbildung 3.20: MSE und R^2-Werte für die Regressionsmodelle mit Transformation durch B-Splines mit einer variierenden Anzahl an Knotenpunkten $N^{nod} \in \{5, 10, 50\}$ (Zeilen der Tabelle) und einem unterschiedlichen Grad der Polynome $deg \in \{0, 1, 2, 3\}$ (Spalten der Tabelle)

Training von Entscheidungsbäumen und Random Forests

Ein Entscheidungsbaum, welcher anhand des Trainingsdatensatzes D_{train}^{rand} ohne Regularisierungsmethoden angepasst wird, weist einen mittleren quadratischen Fehler (MSE) von $1.043 \cdot 10^{-16}$ für den Trainingsdatensatz D_{train}^{rand} und einen MSE-Wert von nur $1.468 \cdot 10^{-3}$ für den Testdatensatz D_{test}^{rand} auf. Man sieht hierbei eine deutliche Diskrepanz zwischen der Genauigkeit der Vorhersage bei den Trainingsdaten und den ungesehenen Testdaten. Aufgrund dessen werden verschiedene Regularisierungsmethoden angewendet, um

dieser Überanpassung der Entscheidungsbäume entgegenzuwirken. Es werden hierfür unterschiedliche Entscheidungsbäume trainiert, bei denen jeweils die Tiefe des Baumes max_{depth} beschränkt und die minimale Anzahl $min_{samp,split}$ an Datenpunkten, die ein Knoten aufweisen muss, damit er während des Trainings aufgeteilt wird, erhöht. Da ein Entscheidungsbaum ohne Regulierungsmethoden über eine Tiefe von 46 verfügt, wird zur Regularisierung die Tiefe auf maximal 32 beschränkt.

	2	4	16	64	256	1024
keine Beschränkung	$1.04 \cdot 10^{-16}$	$1.15 \cdot 10^{-4}$	$6.19 \cdot 10^{-4}$	$1.53 \cdot 10^{-3}$	$3.06 \cdot 10^{-3}$	$5.78 \cdot 10^{-3}$
8	$2.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$
16	$1.65 \cdot 10^{-3}$	$1.66 \cdot 10^{-3}$	$1.73 \cdot 10^{-3}$	$2.04 \cdot 10^{-3}$	$3.13 \cdot 10^{-3}$	$5.79 \cdot 10^{-3}$
24	$2.3 \cdot 10^{-5}$	$1.29 \cdot 10^{-4}$	$6.21 \cdot 10^{-4}$	$1.53 \cdot 10^{-3}$	$3.06 \cdot 10^{-3}$	$5.78 \cdot 10^{-3}$
32	$1.75 \cdot 10^{-8}$	$1.15 \cdot 10^{-4}$	$6.19 \cdot 10^{-4}$	$1.53 \cdot 10^{-3}$	$3.06 \cdot 10^{-3}$	$5.78 \cdot 10^{-3}$

(a) MSE für D_{train}^{rand}

	2	4	16	64	256	1024
keine Beschränkung	$1.47 \cdot 10^{-3}$	$1.44 \cdot 10^{-3}$	$1.53 \cdot 10^{-3}$	$2.06 \cdot 10^{-3}$	$3.32 \cdot 10^{-3}$	$5.91 \cdot 10^{-3}$
8	$2.11 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$	$2.11 \cdot 10^{-2}$
16	$2.29 \cdot 10^{-3}$	$2.29 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$	$2.49 \cdot 10^{-3}$	$3.39 \cdot 10^{-3}$	$5.91 \cdot 10^{-3}$
24	$1.47 \cdot 10^{-3}$	$1.44 \cdot 10^{-3}$	$1.53 \cdot 10^{-3}$	$2.06 \cdot 10^{-3}$	$3.32 \cdot 10^{-3}$	$5.91 \cdot 10^{-3}$
32	$1.47 \cdot 10^{-3}$	$1.44 \cdot 10^{-3}$	$1.53 \cdot 10^{-3}$	$2.06 \cdot 10^{-3}$	$3.32 \cdot 10^{-3}$	$5.91 \cdot 10^{-3}$

(b) MSE für D_{test}^{rand}

Abbildung 3.21: MSE für die Entscheidungsbäume beim Training mit D^{rand} bei Variation von $max_{depth} \in \{\text{None}, 8, 16, 24, 32, 64\}$ (Zeilen der Tabelle) und $min_{samp,split} \in \{4, 16, 64, 256, 1024, 4096\}$ (Spalten der Tabelle)

	2	4	16	64	256	1024
keine Beschränkung	1	0.9997	0.9986	0.9965	0.9929	0.9865
8	0.9492	0.9492	0.9492	0.9492	0.9492	0.9492
16	0.9962	0.9962	0.9961	0.9953	0.9927	0.9865
24	0.9999	0.9997	0.9986	0.9965	0.9929	0.9865
32	1	0.9997	0.9986	0.9965	0.9929	0.9865

(a) R^2 für D_{train}^{rand}

	2	4	16	64	256	1024
keine Beschränkung	0.9967	0.9968	0.9966	0.9953	0.9923	0.9862
8	0.949	0.949	0.949	0.949	0.949	0.949
16	0.9948	0.9948	0.9947	0.9943	0.9921	0.9862
24	0.9967	0.9968	0.9965	0.9953	0.9923	0.9862
32	0.9967	0.9968	0.9966	0.9953	0.9923	0.9862

(b) R^2 für D_{test}^{rand}

Abbildung 3.22: R^2-Werte für die Entscheidungsbäume bei einer Variation von $max_{depth} \in \{\text{None}, 8, 16, 24, 32, 64\}$ (Zeilen der Tabelle) und $min_{samp,split} \in \{4, 16, 64, 256, 1024, 4096\}$ (Spalten der Tabelle)

Die Abbildungen 3.21 und 3.22 zeigen die MSE beziehungsweise die R^2-Werte für die

jeweiligen Entscheidungsbäume mit den unterschiedlich festgelegten Regularisierungsparametern max_{depth} und $min_{samp,split}$ anhand des Test- und Trainingsdatensatzes. Bezüglich der Vorhersage des Modells bei den Testdaten liegt der niedrigste MSE und der größte R^2-Wert bei $max_{depth} = 24$ und $min_{samp,split} = 4$.
Mit diesen Regularisierungen ($max_{depth} = 24$ und $min_{samp,split} = 4$), welche die beste Verallgemeinerungsfähigkeit aufweisen, werden Random Forest mit einer unterschiedlichen Anzahl an Schätzern (Entscheidungsbäumen) trainiert. Wie aus Tabelle 3.5 und Abbildung 3.23 ersichtlich wird, kann durch den Einsatz mehrerer Schätzer der MSE Fehlerwert weiter reduziert werden.

	1 Schätzer	10 Schätzer	50 Schätzer	150 Schätzer	250 Schätzer
R^2-Wert D^{rand}_{train}	0.99971	0.99958	0.99970	0.99972	0.99972
R^2-Wert D^{rand}_{test}	0.99685	0.99841	0.99862	0.99866	0.99866
MSE D^{rand}_{train}	$1.293 \cdot 10^{-3}$	$1.855 \cdot 10^{-4}$	$1.350 \cdot 10^{-4}$	$1.269 \cdot 10^{-4}$	$1.247 \cdot 10^{-4}$
MSE D^{rand}_{test}	$1.442 \cdot 10^{-3}$	$7.237 \cdot 10^{-4}$	$6.331 \cdot 10^{-4}$	$6.166 \cdot 10^{-4}$	$6.126 \cdot 10^{-4}$

Tabelle 3.5: Vorhersagegenauigkeit der Random Forests mit unterschiedlich vielen Schätzern

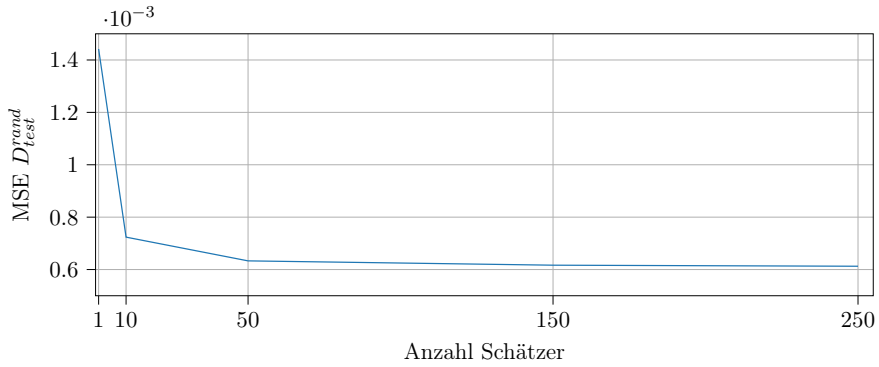


Abbildung 3.23: Fehlerreduktion des MSE D^{rand}_{test} für Random Forests bei unterschiedlich vielen Schätzern

Training von Feed Forward neuronalen Netzen

Für das Training von Feed Forward neuronalen Netzen (FFNN) werden unterschiedliche, in Tabelle 3.6 aufgelistete Hyperparameter untersucht. Die Modelle bestehen jeweils aus einer Eingabeschicht mit sieben Neuronen für jeden Eingabewert und einer Ausgabeschicht mit vier Neuronen. Die Anzahl der zwischenliegenden Schichten, die Anzahl der Neuronen

dieser Schichten und die Aktivierungsfunktionen werden in einer Hyperparameterstudie simultan variiert. Die Modelle werden für jeweils 100 Epochen mit einer Batch-Größe von 1000 trainiert. Die Abbildungen 3.24 und 3.25 zeigen die Ergebnisse (MSE D_{test}^{rand} Werte) der getesteten neuronalen Netze für die jeweiligen, einzelnen Hyperparameter.

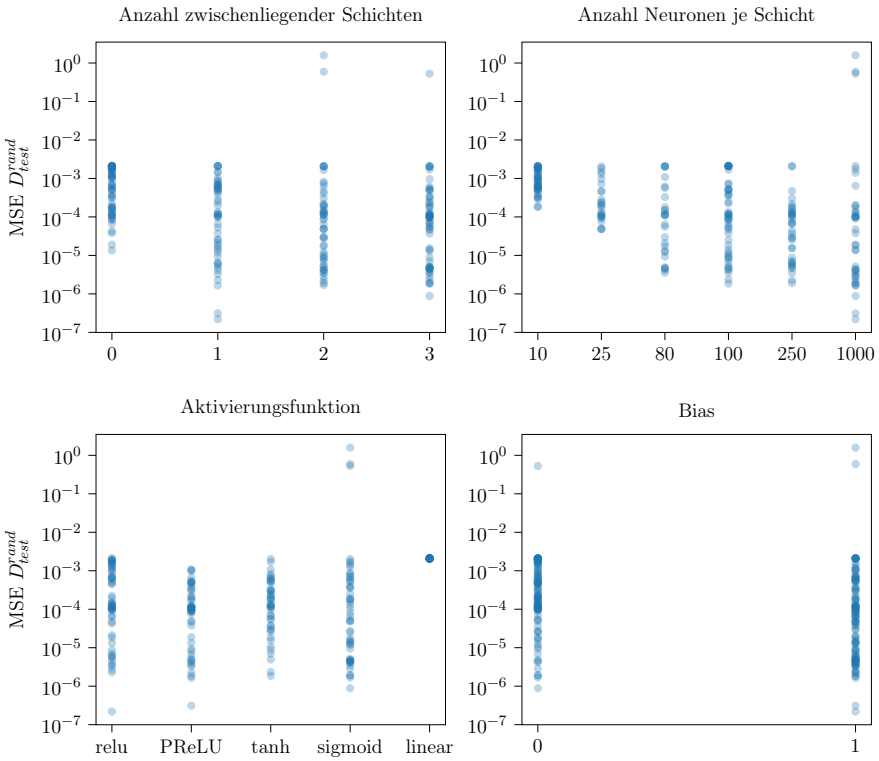


Abbildung 3.24: MSE D_{test}^{rand} der einzelnen Konstellationen an Hyperparametern, welche den Modellaufbau des neuronalen Netzes betreffen

Hyperparameter	**Wertebereich**
Anzahl an zwischenliegenden Schichten	0, 1, 2, 3
Anzahl an Neuronen je Schicht	10, 25, 80, 100, 250, 1000
Aktivierungsfunktion	linear, ReLU, tanh, PReLU, sigmoid
Bias	Mit, Ohne
Lernrate	0.01, 0.001, 0.005
Fehlerfunktion im Training	quadratisch (MSE), absolut (MAE)
Skalierung	StandardScale, MaxAbsScale, ohne

Tabelle 3.6: Untersuchte Hyperparameter für das Training von FFNN

Werden (wie hier im Training und in der Auswertung) die Vorhersagen für die einzelnen Zeitschritte unabhängig betrachtet, so ist zu erkennen, dass die Abweichung der Prognose der FFNN-Modelle im Vergleich zu den rekurrenten neuronalen Netzen aus dem vorherigen Abschnitt und den bisher betrachteten ML-Methoden für den gleichen Test- und Trainingsdatensatz deutlich kleiner ist. Dass dies nicht direkt auf eine entsprechend gute Genauigkeit der Modelle bei einer Betrachtung eines zeitlichen Verlaufs (wie es bei der Anwendung in Simulationen der Fall ist) schließen lässt, wird in Abschnitt 3.2.6 weiter thematisiert. Auch bei den FFNN-Modellen werden Netze mit mindestens so viel trainierbaren Parametern wie bei den LSTM und GRU-Modellen für eine gute Anpassung benötigt (siehe Abbildung 3.26).

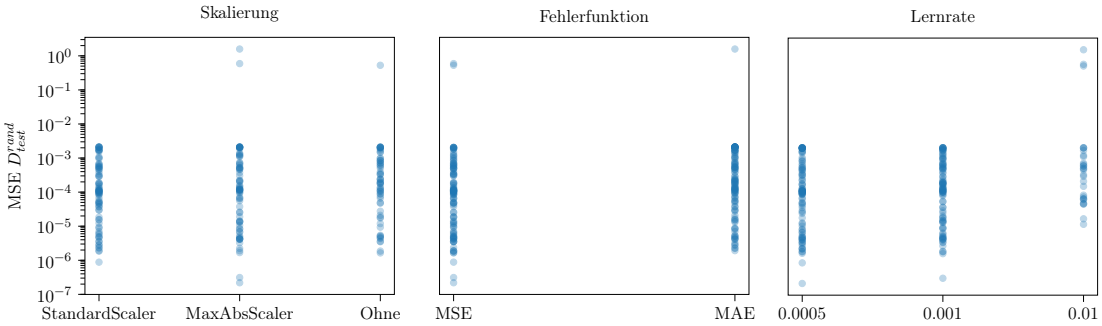


Abbildung 3.25: MSE D_{test}^{rand} der einzelnen Konstellationen an Hyperparametern, welche den Trainingsprozess beeinflussen

Rang (nach MSE D_{test}^{rand})	Anzahl Schichten	Anzahl Neuronen	Aktivierungs-funktion	Bias	Skalierung	Lernrate	Anzahl trainierbarer Parameter	MSE D_{test}^{rand}
1	1	1000	ReLU	1	MaxAbsScale	0.0005	1.013.004	$2.191 \cdot 10^{-7}$
2	1	1000	PReLU	1	MaxAbsScale	0.001	1.015.004	$3.119 \cdot 10^{-7}$
3	3	1000	sigmoid	0	StandardScale	0.0005	3.011.000	$8.819 \cdot 10^{-7}$
4	1	1000	PReLU	1	ohne	0.001	1.015.004	$1.644 \cdot 10^{-6}$
5	2	1000	sigmoid	0	MaxAbsScale	0.0005	2.011.000	$1.682 \cdot 10^{-6}$
6	3	100	tanh	1	StandardScale	0.0005	31.504	$1.852 \cdot 10^{-6}$

Tabelle 3.7: Hyperparameter der sechs besten FFNN-Modelle gemessen am kleinsten MSE-Fehlerwert für den Datensatz D_{test}^{rand}

Die Tabelle 3.7 zeigt diejenigen Konfigurationen aus der Hyperparameterstudie, welche den geringsten MSE D_{test}^{rand} aufweisen. Die beiden neuronalen Netze in Zeile 1 und 6, welche als $ML^{FFNN,3-100}$ und $ML^{FFNN,1-1000}$ bezeichnet werden, werden in den folgenden Abschnitten erweitert und für eine nähere Auswertung herangezogen.

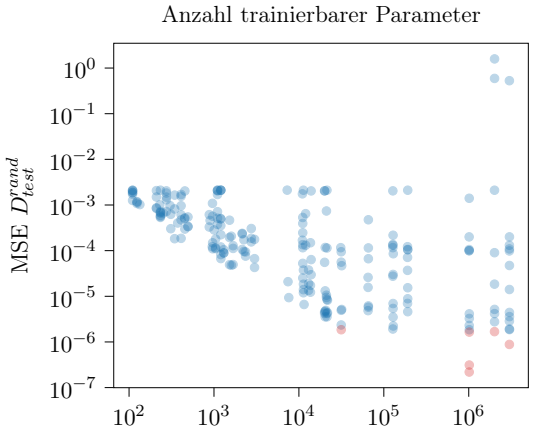


Abbildung 3.26: MSE D_{test}^{rand} der verschiedenen ML-Modelle nach der Gesamtanzahl an trainierbaren Parametern des neuronalen Netzes. Bei den rot markierten ML-Modellen handelt es sich um die sechs in Tabelle 3.7 aufgeführten Modelle.

Feed Forward Adaption der LMSC-Modelle (FF-LMSC)

Im Folgenden wird eine neu entwickelte Variante der LMSC-Modelle aus [11] (siehe Abschnitt 2.3.3) vorgestellt, welche die Vorteile dieser Modelle für den Anwendungsfall des Trainings eines ML-Materialmodells mit internen Geschichtsvariablen nutzt. Das neu entwickelte Modell, welches auch als FF-LMSC-Modell bezeichnet wird, besteht aus einer Eingabeschicht mit einem darauffolgenden, beliebig festzulegenden Feed-Forward-Block, bestehend aus einer festgelegten Anzahl verbundener Schichten an Neuronen. Die Ausgaben $q \in \mathbb{R}^k$ dieses Feed-Forward-Blocks gehen zusammen mit der Norm $\|\varepsilon^t\|_2$ der Eingabe für die Dehnungskomponenten $\varepsilon^t = (\varepsilon_1^t, \varepsilon_2^t, \varepsilon_4^t)^T$ in zwei separate finale Schichten, einer Ausgabeschicht für die Geschichtsvariable $\varepsilon_{eff,pl}$ und einer Ausgabeschicht für die Spannungen (siehe Abbildung 3.27), ein. Die Ausgabe $\varepsilon_{eff,pl}^t$ berechnet sich auf Basis von q durch

$$\varepsilon_{eff,pl}^t = \exp(-\|\varepsilon^t\|_2 \cdot \alpha_{\varepsilon_{eff,pl}}) \cdot (\varepsilon_{eff,pl}^{t-1} - \beta_{\varepsilon_{eff,pl}}) + \beta_{\varepsilon_{eff,pl}} \tag{3.32}$$

mit

$$\alpha_{\varepsilon_{eff,pl}} = g_1^{\varepsilon_{eff,pl}}(W_1^{\varepsilon_{eff,pl}} \cdot q + b_1^{\varepsilon_{eff,pl}}) \tag{3.33}$$

$$\beta_{\varepsilon_{eff,pl}} = g_2^{\varepsilon_{eff,pl}}(W_2^{\varepsilon_{eff,pl}} \cdot q + b_2^{\varepsilon_{eff,pl}}) \tag{3.34}$$

und die Ausgaben $\sigma^t \in \mathbb{R}^3$ ergeben sich analog durch

$$\sigma^t = \exp(-\|\varepsilon^t\|_2 \cdot \alpha_\sigma) \cdot (\sigma^{t-1} - \beta_\sigma) + \beta_\sigma \tag{3.35}$$

mit

$$\alpha_\sigma = g_1^\sigma(W_1^\sigma \cdot q + b_1^\sigma) \tag{3.36}$$

$$\beta_\sigma = g_2^\sigma(W_2^\sigma \cdot q + b_2^\sigma), \tag{3.37}$$

wobei mit $g_1^{\varepsilon_{eff,pl}}, g_2^{\varepsilon_{eff,pl}}, g_1^\sigma, g_2^\sigma$ die jeweiligen Aktivierungsfunktionen, mit $W_1^{\varepsilon_{eff,pl}}, W_2^{\varepsilon_{eff,pl}}$, W_1^σ, W_2^σ die trainierbaren Gewichte (Gewichtsmatrizen) und mit $b_1^{\varepsilon_{eff,pl}}, b_2^{\varepsilon_{eff,pl}}, b_1^\sigma, b_2^\sigma$ die Bias-Werte der Ausgabeschichten bezeichnet werden.

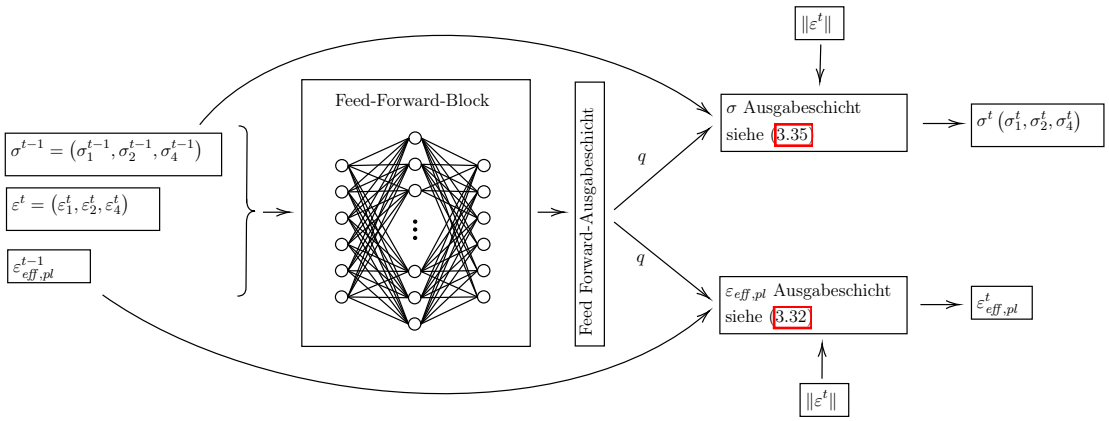


Abbildung 3.27: Schematische Darstellung der Feed Forward Variante der LMSC-Modelle

Der Vorteil der LMSC-Modelle für Nulldehnungen in der Eingabe keine Änderung bei der Ausgabe der Spannungswerte gegenüber dem letzten Zeitschritt zu erzwingen und insbesondere Nulldehnungen auf Nullspannungen abbilden zu können, wird auch hier durch (3.35) wegen

$$\varepsilon^t = (0,0,0)^T \Rightarrow \|\varepsilon^t\|_2 = 0 \Rightarrow \sigma^t = \exp(0) \cdot (\sigma^{t-1} - \beta_\sigma) + \beta_\sigma = \sigma^{t-1} \tag{3.38}$$

sichergestellt. Da die gleiche Bedingung auch auf die plastische Dehnung $\varepsilon_{eff,pl}$ übertragen werden kann, wird dies in der Ausgabeschicht für $\varepsilon_{eff,pl}$ analog durch (3.32) berücksichtigt. Zu beachten ist auch hier in Hinblick auf das Training der Modelle, dass durch eine Skalierung Nullwerte in den Eingaben nicht verändert werden dürfen, wie es beispielsweise im Allgemeinen bei einer Standardisierung (StandardScale) bei einem Mittelwert ungleich

null der Fall ist. Werden für $g_1^{\varepsilon_{eff,pl}}$ und $g_2^{\varepsilon_{eff,pl}}$ Aktivierungsfunktionen mit einem nicht-negativen Zielbereich gewählt, so kann zusätzlich wegen $\varepsilon_{eff,pl}^0 = 0$ und

$$\alpha_{\varepsilon_{eff,pl}} \geq 0 \Rightarrow M := \exp(-\|\varepsilon\|_2 \cdot \alpha_{\varepsilon_{eff,pl}}) \in [0,1] \tag{3.39}$$

$$\Rightarrow \varepsilon_{eff,pl}^t = \underbrace{M \cdot \varepsilon_{eff,pl}^{t-1}}_{\geq 0 \text{ für } \varepsilon_{eff,pl}^{t-1} \geq 0} + \underbrace{\beta_{\varepsilon_{eff,pl}}(1-M)}_{\geq 0 \text{ für } \beta_{\varepsilon_{eff,pl}} \geq 0} \geq 0 \tag{3.40}$$

induktiv gezeigt und somit sichergestellt werden, dass das neuronale Netz für die plastische Dehnung nur nicht-negative Werte ausgibt, wie es auch im klassischen Materialmodell der Fall ist.

Als Ausgangspunkt für den Aufbau des Feed-Forward-Blocks dienen die Ergebnisse der Hyperparameterstudie der FFNN-Modelle im vorherigen Abschnitt. Daher werden für diesen Teil des FF-LMSC-Modells die beiden FFNN-Modelle $ML^{FFNN,3-100}$ und $ML^{FFNN,1-1000}$ mit 3 beziehungsweise einer zwischenliegenden Schicht mit je 100 beziehungsweise 1000 Neuronen mit Bias Werten und tanh beziehungsweise ReLU Aktivierungsfunktionen betrachtet. Aufbauend auf dieser Modellstruktur werden die Aktivierungsfunktionen $g_1^{\varepsilon_{eff,pl}}, g_2^{\varepsilon_{eff,pl}}$ und g_1^σ, g_2^σ in einer weiteren Hyperparameterstudie variiert. Betrachtet werden für g_1^σ, g_2^σ eine lineare, ReLU, exponential, tanh, PReLU und sigmoid Aktivierungsfunktion und für $g_1^{\varepsilon_{eff,pl}}, g_2^{\varepsilon_{eff,pl}}$ die ReLU, exponential und sigmoid Aktivierungsfunktion, welche in ausschließlich nicht-negativen Ausgabewerten resultieren. Die Aktivierungsfunktionen der beiden Ausgabeschichten werden unabhängig voneinander separat variiert und jede mögliche Kombination getestet, wobei die jeweiligen Aktivierungsfunktionen der anderen Ausgabeschicht entsprechend den in [11] genutzten Aktivierungsfunktionen für α und β auf tanh und exponential gesetzt sind. Die Abbildungen 3.28 und 3.29 zeigen die Fehlerwerte $\text{MSE}_{\varepsilon_{eff,pl}}$ beziehungsweise MSE_σ der trainierten FF-LMSC Modelle basierend auf den beiden FFNN-Modellen $ML^{FFNN,3-100}$ und $ML^{FFNN,1-1000}$ mit den unterschiedlichen Kombinationen für $g_1^{\varepsilon_{eff,pl}}, g_2^{\varepsilon_{eff,pl}}$ beziehungsweise g_1^σ, g_2^σ bezüglich der jeweiligen Ausgabekomponenten $\varepsilon_{eff,pl}$ beziehungsweise $\sigma_1, \sigma_2, \sigma_4$. Die Kombinationen sind auf der x-Achse geordnet dargestellt, beginnend bei dem gemittelten, kleinsten Fehlerwert bezüglich beider Modelle und den jeweiligen, relevanten Ausgabewerten. Je weiter links auf der x-Achse die Funktionen angeordnet sind, desto besser funktioniert die Kombination der beiden Aktivierungsfunktionen der Ausgabeschicht für die beiden ML-Modelle gemittelt. Hierbei wird auch ersichtlich, dass manche der Kombinationen bezüglich der Spannungsausgabe lediglich für einen Feed-Forward-Block gut funktionieren und für den anderen Block einen deutlich höheren MSE_σ D_{test}^{rand} haben (beispielsweise bei $g_1^\sigma = \text{exponential}$, $g_2^\sigma = \tanh$ oder $g_1^\sigma = \text{ReLU}$, $g_2^\sigma = \text{linear}$).

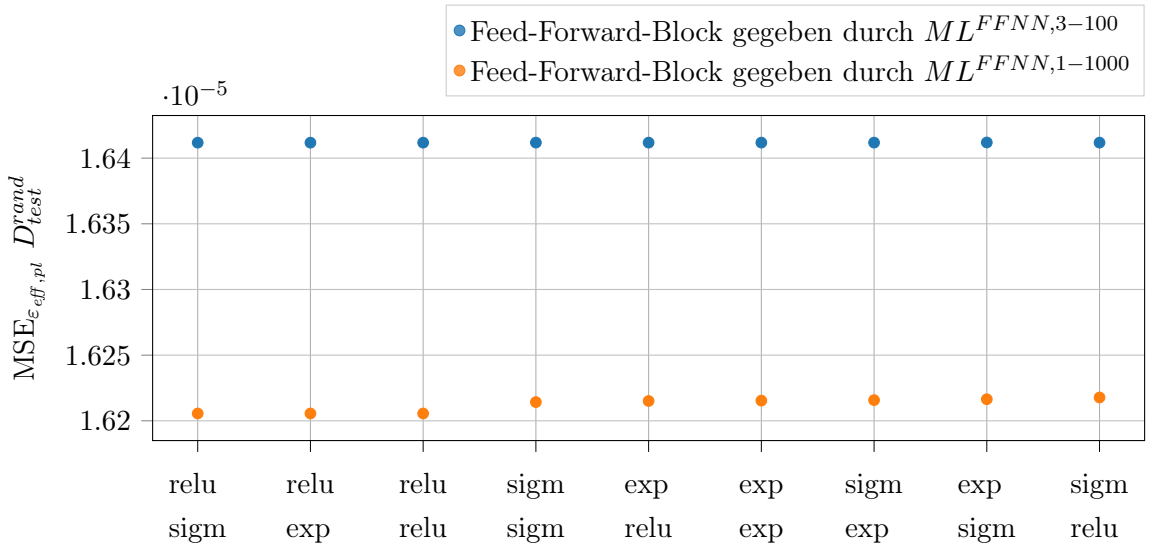


Abbildung 3.28: $\text{MSE}_{\varepsilon_{eff,pl}}\ D_{test}^{rand}$ der verschiedenen Kombinationen für die Aktivierungsfunktionen sigmoid (sigm), exponential (exp) und ReLU für $g_1^{\varepsilon_{eff,pl}}$ (obere Zeile) und $g_2^{\varepsilon_{eff,pl}}$ (untere Zeile)

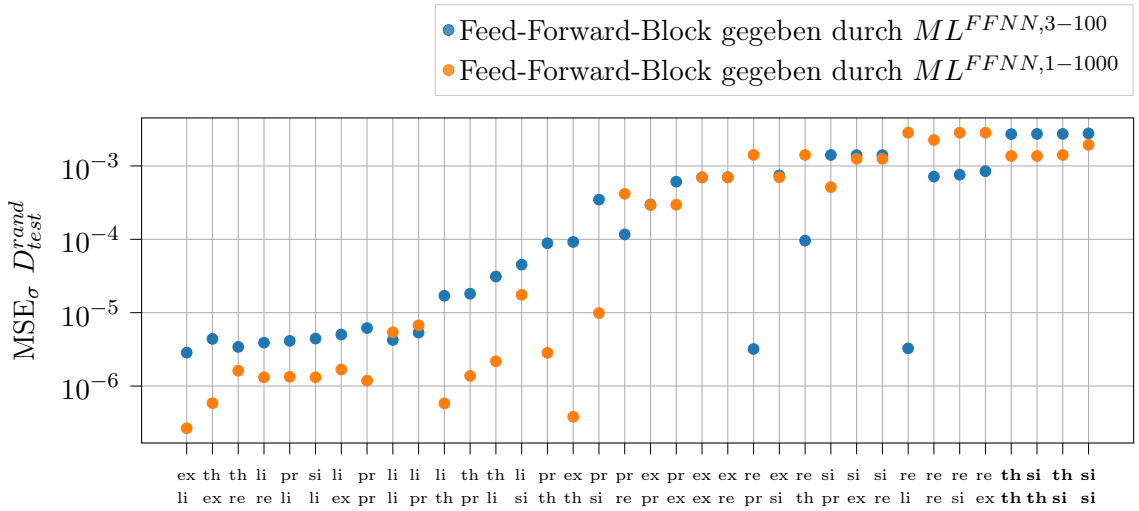


Abbildung 3.29: $\text{MSE}_{\sigma}\ D_{test}^{rand}$ der verschiedenen Kombinationen für die Aktivierungsfunktionen linear (li), ReLU (re), exponential (ex), tanh (th), PReLU (pr) und sigmoid (si) für g_1^{σ} (obere Zeile) und g_2^{σ} (untere Zeile)

Während für die $\varepsilon_{eff,pl}$-Ausgabeschicht keine großen Unterschiede bezüglich des $\text{MSE}_{\varepsilon_{eff,pl}}$

D_{test}^{rand} für die Wahl der beiden Aktivierungsfunktionen zu erkennen sind (siehe Abbildung 3.28) und alle betrachteten Kombinationen ähnlich gut für beide Modelle funktionieren, fallen für die σ-Ausgabeschicht (siehe Abbildung 3.29) vor allem Kombinationen mit zwei beschränkten Aktivierungsfunktionen (tanh und sigmoid) in der Vorhersagegenauigkeit ab (fett markiert in Abbildung 3.29). Im weiteren Verlauf werden zwei trainierte FF-LMSC-Modelle mit $g_1^{\varepsilon_{eff},pl} = \text{ReLU}, g_2^{\varepsilon_{eff},pl} = \text{sigmoid}$ und $g_1^{\sigma} = \text{exponential}, g_2^{\sigma} = \text{linear}$ betrachtet, welche abhängig des Feed-Forward-Blocks mit $ML^{FF-LMSC,3-100}$ und $ML^{FF-LMSC,1-1000}$ bezeichnet werden. In den ursprünglich in [11] vorgestellten LMSC-Modellen werden quadratische Schichten genutzt. Um zu überprüfen, ob auch mit der hier entwickelten Variante die Anpassung und Vorhersage der Modelle dadurch verbessert werden können, werden zwei zusätzliche FF-LMSC-Modelle trainiert, bei denen der Feed-Forward-Block durch quadratische Schichten analog zu (2.56) ersetzt wird. Der Ausgabevektor q einer solchen Schicht ergibt sich gegeben den Eingaben I durch

$$q = g(W^1 I + b^1) \cdot g(W^2 I + b^2) \tag{3.41}$$

mit den trainierbaren Gewichten W^1, W^2, Bias-Werten b^1, b^2 und einer Aktivierungsfunktion g. Die beiden resultierenden Modelle werden mit $ML^{FF-LMSC,3-100,qu}$ und $ML^{FF-LMSC,1-1000,qu}$ bezeichnet.

3.2.6 Auswertung, Validierung und Vergleich der Modelle

In diesem Abschnitt wird eine Auswahl der untersuchten ML-Modelle und Methoden insbesondere in Hinblick auf eine spätere Anwendung in Simulationen anhand zusätzlicher Datensätze ausgewertet und die Modelle miteinander verglichen. Bei den ML-Materialmodellen mit Hinzunahme der Geschichtsvariablen (*Feed-Forward-Modelle*) aus Abschnitt 3.2.5 ist hierbei zu beachten, dass bei einer Anwendung der Modelle in Simulationen die Ausgaben zu einem Zeitschritt t als Eingabe für den nächsten Zeitschritt $t+1$ genutzt werden. Ungenauigkeiten in der Vorhersage werden dadurch bei der Anwendung der ML-Materialmodelle zum nächsten Zeitschritt weitergetragen und der Fehler kann dadurch verstärkt werden, wodurch es zu einer Fortpflanzung des Fehlers bezüglich der Vorhersage der Modelle über die Zeitschritte kommt.
Betrachtet werden die folgenden ML-Materialmodelle aus den vorherigen Abschnitten: das einfache lineare Regressionsmodell ML^{LinReg}, das Spline-Regressionsmodell ML^{SplReg} mit einer Transformation anhand kubischer Splines mit 5 quantil-gesetzten Knoten, der Random Forest $ML^{RandFor}$ bestehend aus 250 Schätzern mit einer maximalen Baumtiefe von $max_{depth} = 24$ und $min_{samp,split} = 4$ je Schätzer, die beiden Feed-Forward neuronalen Net-

ze $ML^{FFNN,3-100}$ und $ML^{FFNN,1-1000}$ aus Abschnitt 3.2.5 und die darauf aufbauenden LMSC-Erweiterungen $ML^{FF-LMSC,3-100}$, $ML^{FF-LMSC,1-1000}$, $ML^{FF-LMSC,3-100,qu}$ sowie $ML^{FF-LMSC,1-1000,qu}$. Weiterhin wird im Bereich der rekurrenten Modelle ein LSTM-Modell ML^{LSTM}, bestehend aus drei Schichten mit je 200 Neuronen (insgesamt 805.604 trainierbare Parameter) und ein LMSC-Modell ML^{LMSC}, bestehend aus 4 quadratischen Schichten mit je 35 Neuronen und 6 Zustandsvariablen (insgesamt 8710 trainierbare Parameter) betrachtet. Für die Auswertung wird in diesem Abschnitt zusätzlich der Testdatensatz D^{hat} aus Abschnitt 3.2.3 hinzugenommen, wobei die Vorhersage der zeitlichen Verläufe zunächst außerhalb der Simulation betrachtet wird. Dies bedeutet, dass die Eingabepfade (zeitlichen Verläufe der Dehnungskomponenten) festgelegt sind und nicht von der Spannungsausgabe der ML-Modelle beeinflusst wird. Um zu evaluieren, wie gut die ML-Materialmodelle den Nullzustand, wie er insbesondere zu Beginn einer Simulation in vielen Elementen vorkommt, prognostizieren können, wird ein zusätzlicher Testdatensatz, welcher mit D^0 bezeichnet wird, betrachtet. Dieser Datensatz besteht lediglich aus einem einzelnen Dehnpfad, welcher für 100 Zeitschritte ausschließlich Dehnungsinkremente und eine plastische Dehnung gleich null als Eingabe bekommt, welche wiederum zu Spannungsausgaben gleich null führen.

	MSE D^{rand}_{test}	MSE D^{hat}	MSE D^0
ML^{LinReg}	$2.031 \cdot 10^{-3}$	$2.305 \cdot 10^{-4}$	$7.252 \cdot 10^{-7}$
ML^{SplReg}	$1.977 \cdot 10^{-3}$	$2.318 \cdot 10^{-4}$	$4.997 \cdot 10^{-7}$
$ML^{RandFor}$	$7.269 \cdot 10^{-4}$	$7.877 \cdot 10^{-4}$	$2.488 \cdot 10^{-11}$
$ML^{FFNN,3-100}$	$5.445 \cdot 10^{-7}$	$1.248 \cdot 10^{-5}$	$4.471 \cdot 10^{-8}$
$ML^{FFNN,1-1000}$	$2.986 \cdot 10^{-7}$	$6.439 \cdot 10^{-6}$	$1.556 \cdot 10^{-8}$
$ML^{FF-LMSC,3-100}$	$1.571 \cdot 10^{-7}$	$2.842 \cdot 10^{-7}$	0
$ML^{FF-LMSC,1-1000}$	$1.402 \cdot 10^{-7}$	$2.639 \cdot 10^{-7}$	0
$ML^{FF-LMSC,3-100,qu}$	$1.405 \cdot 10^{-7}$	$2.671 \cdot 10^{-7}$	0
$ML^{FF-LMSC,1-1000,qu}$	$4.382 \cdot 10^{-7}$	$1.002 \cdot 10^{-6}$	0

Tabelle 3.8: Übersicht des MSE Fehlers **ohne Berücksichtigung der Fehlerfortpflanzung** aller Feed-Forward-Modelle für alle Ausgabe-Komponenten (mit $\varepsilon_{eff,pl}$) bezüglich der verschiedenen Testdatensätze, wobei das markierte Modell $ML^{FFNN,1-1000}$ die geringsten Abweichungen aufweist.

	MSE D_{test}^{rand}	MSE D^{hat}	MSE D^0
ML^{LinReg}	$3.088 \cdot 10^{-1}$	$t^{\star(748)}$	$2.613 \cdot 10^{-3}$
ML^{SplReg}	$t^{\star(30)}$	$3.694 \cdot 10^{-2}$	$3.679 \cdot 10^{-4}$
$ML^{RandFor}$	$1.242 \cdot 10^{-1}$	$9.908 \cdot 10^{-2}$	$2.573 \cdot 10^{-11}$
$ML^{FFNN,3-100}$	$2.436 \cdot 10^{-3}$	$3.736 \cdot 10^{-1}$	$8.165 \cdot 10^{-5}$
$ML^{FFNN,1-1000}$	$1.211 \cdot 10^{-3}$	$8.946 \cdot 10^{-2}$	$7.452 \cdot 10^{-5}$
$ML^{FF-LMSC,3-100}$	$4.645 \cdot 10^{-5}$	$2.138 \cdot 10^{-2}$	0
$ML^{FF-LMSC,1-1000}$	$2.728 \cdot 10^{-5}$	$1.902 \cdot 10^{-2}$	0
$ML^{FF-LMSC,3-100,qu}$	$2.852 \cdot 10^{-5}$	$1.943 \cdot 10^{-2}$	0
$ML^{FF-LMSC,1-1000,qu}$	$2.483 \cdot 10^{-4}$	$5.713 \cdot 10^{-2}$	0

Tabelle 3.9: Übersicht des MSE Fehlers **mit Berücksichtigung der Fehlerfortpflanzung** aller Feed-Forward-Modelle für alle Ausgabe-Komponenten (mit $\varepsilon_{eff,pl}$) bezüglich der verschiedenen Testdatensätze, wobei das markierte Modell $ML^{FFNN,1-1000}$ die geringsten Abweichungen aufweist. Der Wert $t^{\star(k)}$ steht hierbei für eine MSE-Abweichung, welche nach k Zeitschritten einen Wert von 1 überschreitet.

Tabelle 3.8 zeigt die Performanz (gemessen am MSE) der ML-Materialmodelle, welche mit den Geschichtsvariablen trainiert wurden zunächst ohne Berücksichtigung der Fehlerfortpflanzung (wie es auch im Training der Fall ist) für die drei unterschiedlichen Testdatensätze D_{test}^{rand}, D^{hat} und D^0. Tabelle 3.9 zeigt zum Vergleich die entsprechenden Fehlerwerte mit Berücksichtigung der Fehlerfortpflanzung. Hierfür werden Schritt für Schritt die Eingaben $\varepsilon_{eff,pl}^{t-1}$ und $\sigma_i^{t-1}, i = 1, 2, 4$ in einem Zeitschritt t durch die vom ML-Modell vorhergesagten Ausgabewerte $\hat{\varepsilon}_{eff,pl}^{t-1}$ sowie $\hat{\sigma}_i^{t-1}, i = 1, 2, 4$ ersetzt und die Prognose der Modelle basiert auf diesen Eingabewerten, wie es auch in der Simulation der Fall wäre. Bei allen Modellen ist eine deutliche Diskrepanz zwischen der Vorhersagegenauigkeit mit und ohne Berücksichtigung der Fehlerfortpflanzung zu erkennen, was zeigt wie wichtig die Berücksichtigung dieser Fehlerfortpflanzung bei der Auswertung der Feed-Forward-Modelle ist. Die Nullzustände werden abgesehen von den FF-LMSC-Modellen auch von dem Random Forest sehr genau getroffen. Dies lässt sich dadurch erklären, dass die Teilmenge an Nullzuständen im Trainingsdatensatz innerhalb des Trainings leicht separiert und dieser Partition die null als Ausgabewert zugeordnet wird. Zu sehen ist außerdem, dass beim Training mit Geschichtsvariablen die FF-LMSC-Modelle für

jeden Datensatz insbesondere mit Berücksichtigung der Fehlerfortpflanzung die geringsten Fehlerwerte aufweisen. Abbildung 3.30 zeigt den zeitlichen Verlauf der MSE-Abweichung im direkten Vergleich für das neuronale Netz $ML^{FFNN,3-100}$ einmal ohne den LMSC-Zusatz und das auf die gleiche Weise aufgebaute neuronale Netz $ML^{FF-LMSC,3-100}$, allerdings mit den zusätzlichen in Abschnitt 3.2.5 konstruierten LMSC-Ausgabeschichten. Für beide Modelle ist die Problematik der Fehlerfortpflanzung und die dadurch im Laufe der Zeitschritte wachsende und so entstehende Ungenauigkeit der ML-Modelle in der Anwendung deutlich zu erkennen im direkten Vergleich zu der Vorhersageprognose, welche diese Fehlerfortpflanzung nicht berücksichtigt. Erkennbar ist jedoch auch, dass im Falle des FF-LMSC-Modells $ML^{FF-LMSC,3-100}$ der Fehler für alle Testdatensätze deutlich langsamer ansteigt als bei dem einfachen Feed-Forward neuronalen Netz $ML^{FFNN,3-100}$.

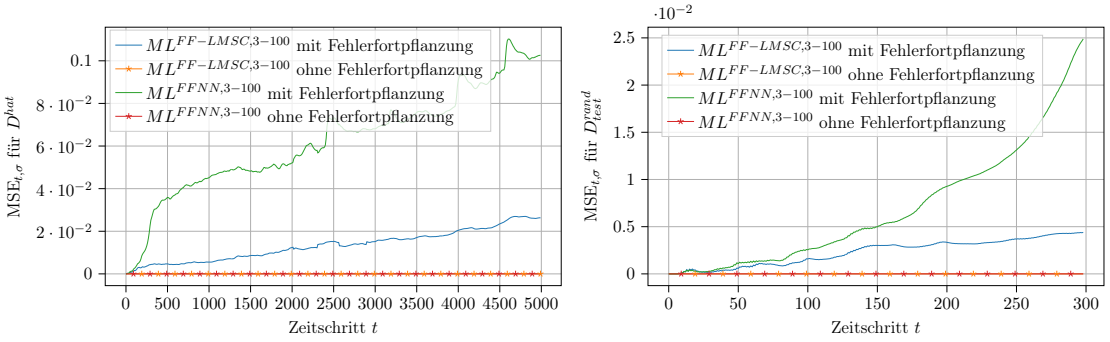


Abbildung 3.30: Vergleich MSE-Verlauf für $ML^{FFNN,3-100}$ und $ML^{FF-LMSC,3-100}$ mit und ohne Fehlerfortpflanzung für die Testdatensätze D^{rand}_{test} und D^{hat}

Abbildung 3.31 zeigt den Verlauf der Abweichung bei einem anhaltenden Nullzustand, wie er beim Validierungsdatensatz D^0 betrachtet wird mit Berücksichtigung einer Fehlerfortpflanzung über die 100 Zeitschritte für eine Auswahl der Modelle (inklusive der beiden rekurrenten Modelle). Während das rekurrente LSTM-Modell nur gegen Ende kleinere Abweichungen aufweist, ist auch hier für das Feed-Forward Netz $ML^{FFNN,1-1000}$ ein nach einigen Zeitschritten stark anwachsender Fehler zu sehen. In Tabelle 3.10 ist eine Gesamtübersicht der mittleren quadratischen Abweichungen aller Spannungskomponenten MSE_σ für alle in diesem Abschnitt betrachteten ML-Materialmodelle mit Berücksichtigung der Fehlerfortpflanzung für alle drei Evaluierungsdatensätze zu sehen. In Abbildung 3.32 sind die Fehlerverläufe $\text{MSE}_{t,\sigma}$ verschiedener ML-Modelle bezüglich der Spannungen insgesamt und der einzelnen Komponenten $\sigma_i, i = 1, 2, 4$ für den Datensatz D^{hat} zu sehen. Auch hier wird die Problematik der Fehlerfortpflanzung für einige ML-Modelle deutlich, wobei sich die Abweichung für das FF-LMSC-Modell $ML^{FF-LMSC,3-100,qu}$ und das LSTM-Modell ML^{LSTM} durchgehend auf einem niedrigeren Niveau bewegten.

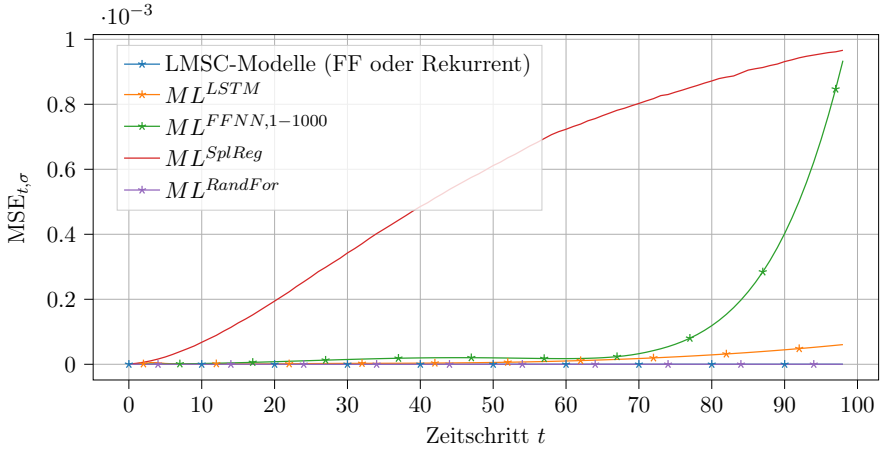


Abbildung 3.31: $\text{MSE}_{t,\sigma}$ mit Berücksichtigung der Fehlerfortpflanzung für den Datensatz D^0

	MSE_σ D^{rand}_{test}	MSE_σ D^{hat}	MSE_σ D^0
ML^{LinReg}	$3.097 \cdot 10^{-1}$	$t^{\star(748)}$	$1.821 \cdot 10^{-6}$
ML^{SplReg}	$t^{\star(30)}$	$9.881 \cdot 10^{-2}$	$4.905 \cdot 10^{-4}$
$ML^{RandFor}$	$4.801 \cdot 10^{-2}$	$1.290 \cdot 10^{-1}$	$3.318 \cdot 10^{-11}$
$ML^{FFNN,1-1000}$	$6.242 \cdot 10^{-3}$	$1.173 \cdot 10^{-1}$	$9.669 \cdot 10^{-5}$
$ML^{FFNN,3-100}$	$6.263 \cdot 10^{-3}$	$1.102 \cdot 10^{-1}$	$1.089 \cdot 10^{-4}$
$ML^{FF-LMSC,3-100}$	$6.183 \cdot 10^{-5}$	$2.778 \cdot 10^{-2}$	0
$ML^{FF-LMSC,1-1000}$	$3.628 \cdot 10^{-5}$	$2.464 \cdot 10^{-2}$	0
$ML^{FF-LMSC,3-100,qu}$	$3.792 \cdot 10^{-5}$	$2.519 \cdot 10^{-2}$	0
$ML^{FF-LMSC,1-1000,qu}$	$3.331 \cdot 10^{-4}$	$7.546 \cdot 10^{-2}$	0
ML^{LSTM}	$5.409 \cdot 10^{-4}$	$4.878 \cdot 10^{-2}$	$1.396 \cdot 10^{-5}$
ML^{LMSC}	$6.485 \cdot 10^{-5}$	$9.379 \cdot 10^{-2}$	0

Tabelle 3.10: Übersicht des MSE_σ Fehlers mit Berücksichtigung der Fehlerfortpflanzung aller Modelle bei Betrachtung der Spannungen als Ausgabe (ohne $\varepsilon_{eff,pl}$), wobei das markierte Modell $ML^{FFNN,1-1000}$ die geringsten Abweichungen aufweist. Der Wert $t^{\star(k)}$ steht hierbei für eine MSE-Abweichung, welche nach k Zeitschritten einen Wert von 1 überschreitet.

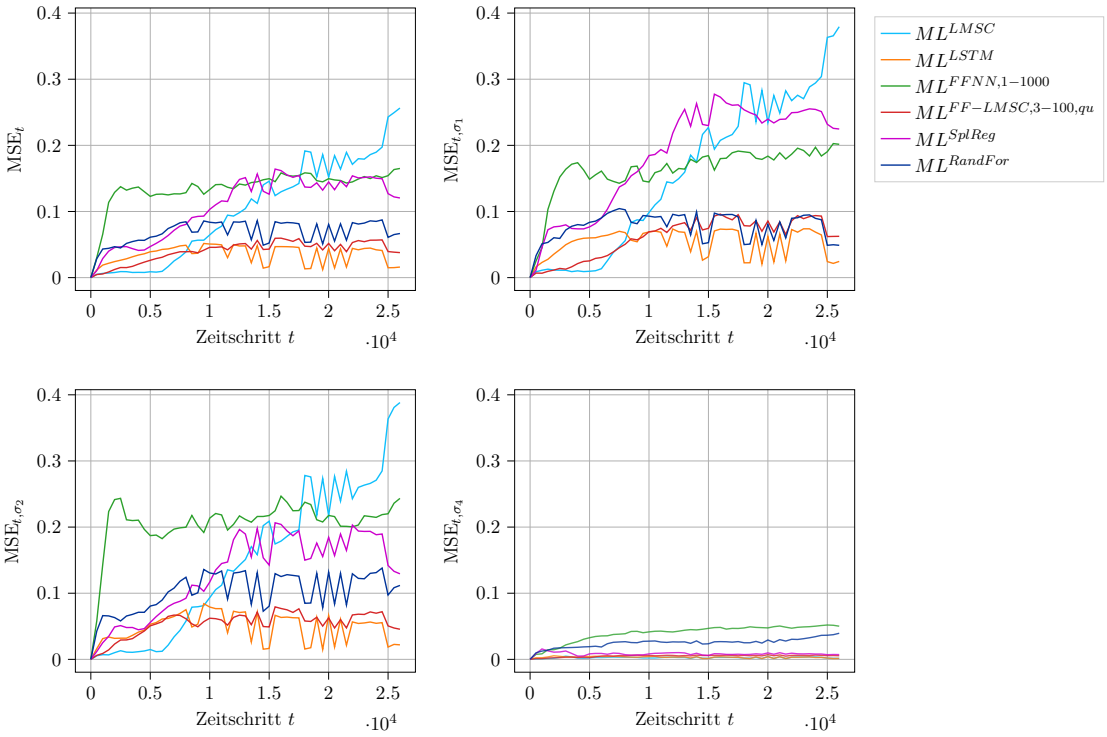


Abbildung 3.32: Verlauf des $\text{MSE}_{t,\sigma}$ und MSE_{t,σ_i}, $i = 1, 2, 4$ mit Berücksichtigung der Fehlerfortpflanzung für den Datensatz D^{hat}

3.2.7 Anwendung und Validierung der ML-Modelle in FE-Simulationen

Die in der bisherigen Auswertung besten Modelle werden in der Anwendung innerhalb verschiedener, im Rahmen des AIMM-Projektes [58] zur Verfügung gestellter Simulationen getestet und validiert. Betrachtet wird ein Zugversuch, eine Hutprofil Simulation, ein Anwendungsfall im Bereich Crash und eine Kreuznapf Umformsimulation. Um die Python-basierten ML-Materialmodelle innerhalb der LS-Dyna Simulationen anwenden zu können, wird auch hier die Interprozesskommunikation aus [67] genutzt. Für die rekurrenten Modelle werden die Zustände des neuronalen Netzes für jeden Integrationspunkt in jedem Zeitschritt nach der Vorhersage lokal zwischengespeichert und das Netz vor der Vorhersage im nächsten Zeitschritt mit den jeweiligen Zuständen initialisiert. Betrachtet werden für die Auswertung und den Vergleich der unterschiedlichen Modelle die zeitlichen Verläufe der quadratischen Abweichung zwischen den Spannungsausgaben der ML-Modelle im Laufe der Simulation und den Spannungen aus der Simulation, welche mit dem klassischen Materialmodell `*MAT_024` durchgeführt wird, gemittelt durch die Anzahl an Integrationspunkten. Werden alle Spannungen, welche von der Ausgabe der ML-Modelle

abhängen (σ_1, σ_2 und σ_4) betrachtet, so wird die Abweichung eines Zeitschrittes t mit MSE_t bezeichnet. Weiterhin wird die Abweichung bezüglich der einzelnen Komponenten, welche mit $\mathrm{MSE}_{t,\sigma_i}$ für $i = 1, 2, 4$ bezeichnet wird, betrachtet sowie eine Abweichung der von-Mises-Spannungen $\mathrm{MSE}_{t,\sigma_{v,M}}$ und die gemittelte quadratische Abweichung aller Knotenpositionen $\mathrm{MSE}_{t,pl}$ des Simulationsmodells. Die zugehörigen, über die Zeit gemittelten Fehlerwerte werden mit MSE_σ, MSE_{σ_i}, $\mathrm{MSE}_{\sigma_{v,M}}$ und MSE_{pl} bezeichnet. Für eine bessere Einordnung des gemittelten Gesamtfehlers der Knotenverschiebungen MAE_{disp} (siehe (3.44)) wird die absolute mittlere Knotenverschiebung $\bar{u}_{*\mathtt{MAT\_024}}$ einer jeweiligen Simulation, durchgeführt mit dem klassischen Materialmodell, bestimmt durch

$$\bar{u}_{*\mathtt{MAT\_024}} \coloneqq \frac{1}{3 \cdot T \cdot K} \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{i=x,y,z} \underbrace{|p_{t,k,i}^{*\mathtt{MAT\_024}} - p_{t-1,k,i}^{*\mathtt{MAT\_024}}|}_{=:u_{t,k,i}^{*\mathtt{MAT\_024}}} \tag{3.42}$$

betrachtet, wobei $p_{t,k,i}^{*\mathtt{MAT\_024}}$ die Position eines Knotens k in $i = x, y, z$-Richtung zum Zeitschritt t bezeichnet. Die relative Abweichung im Verhältnis zur mittleren Verschiebung eines Knotens in einer entsprechenden Simulation wird wie folgt bestimmt

$$\mathrm{MAE}_{disp}^{rel} \coloneqq \frac{\mathrm{MAE}_{disp}}{\bar{u}_{*\mathtt{MAT\_024}}} \tag{3.43}$$

mit

$$\mathrm{MAE}_{disp} \coloneqq \frac{1}{3 \cdot T \cdot K} \sum_{t=1}^{T} \sum_{k=1}^{K} \sum_{i=x,y,z} |\hat{u}_{t,k,i} - u_{t,k,i}^{*\mathtt{MAT\_024}}|, \tag{3.44}$$

wobei mit $\hat{u}_{t,k,i}^{*\mathtt{MAT\_024}}$ die Verschiebung eines Knotens k in $i = x, y, z$-Richtung zum Zeitschritt t der Simulation mit dem entsprechenden ML-Materialmodell bezeichnet wird. Dieser Wert dient zur Evaluierung der Performanz der ML-Materialmodelle innerhalb der Simulation, um die Modelle nicht nur untereinander sondern auch bezüglich der verschiedenen Anwendungsfälle (Simulationen) mit unterschiedlichen Wertebereichen und Größenordnungen für $p_{t,k,i}^{*\mathtt{MAT\_024}}$ besser vergleichbar zu machen.

Zugversuch

Als erste Validierungssimulation wird ein Zugversuch betrachtet. Der Zugversuch ist ein Standardversuch bei der Werkstoffcharakterisierung und dient dazu bestimmte materialspezifische Kenngrößen wie beispielsweise das Elastizitätsmodul zu ermitteln. Bei dem Versuch wird ein Probekörper des Werkstoffes an den Endkanten eingespannt und auseinandergezogen. Die hier verwendete Probengeometrie ist in Abbildung 3.33 zu sehen.

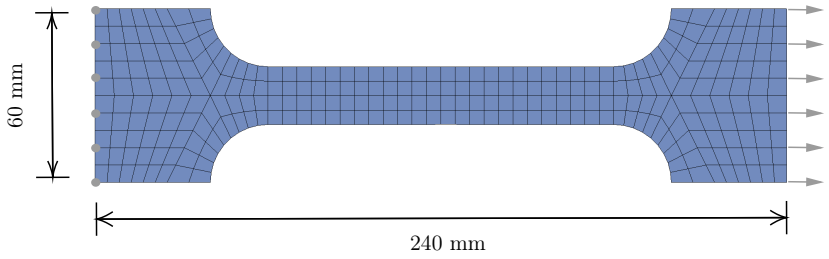


Abbildung 3.33: Aufbau Zugversuch Simulation

Das FE-Modell besteht aus insgesamt 288 Schalenelementen, welche allesamt durch ein ML-Materialmodell ersetzt werden sollen. Als Randbedingung werden die zwölf Knoten am linken Rand der Probe fixiert und eine Verschiebung der Knoten am rechten Rand entlang der Zugachse vorgegeben. Die Verschiebung wird linear mit einem Maximum von 9,6 mm über einen Zeitraum von einer Millisekunde, diskretisiert in 1988 Zeitschritte, aufgebracht. Abbildung 3.34 zeigt den D3plot nach allen 1988 Zeitschritten für die Simulationen, welche mit verschiedenen ML-Modellen (die vier Modelle mit den kleinsten Spannungsabweichungen in der Zugversuch Simulation, siehe auch Tabelle 3.11) durchgeführt wurden sowie der Simulation mit dem klassischen Materialmodell `*MAT_024` zum Vergleich.

	MSE_{σ}	$\text{MSE}_{\sigma_{v,M}}$	MSE_{pl}	MAE^{rel}_{disp}
ML^{SplReg}	$2.614 \cdot 10^{-2}$	$7.302 \cdot 10^{-2}$	$2.076 \cdot 10^{-1}$	$2.591 \cdot 10^{-1}$
$ML^{RandFor}$	$3.661 \cdot 10^{-2}$	$3.305 \cdot 10^{-2}$	$6.592 \cdot 10^{-2}$	$5.683 \cdot 10^{-1}$
$ML^{FFNN,1-1000}$	$1.097 \cdot 10^{-2}$	$1.535 \cdot 10^{-2}$	$3.526 \cdot 10^{-2}$	$1.150 \cdot 10^{-1}$
$ML^{FF-LMSC,3-100,qu}$	$2.548 \cdot 10^{-3}$	$7.384 \cdot 10^{-3}$	$1.846 \cdot 10^{-2}$	$7.907 \cdot 10^{-2}$
ML^{LSTM}	$1.554 \cdot 10^{-3}$	$2.886 \cdot 10^{-3}$	$3.913 \cdot 10^{-3}$	$9.834 \cdot 10^{-2}$
ML^{LMSC}	$2.605 \cdot 10^{-5}$	$3.843 \cdot 10^{-5}$	$2.459 \cdot 10^{-5}$	$1.020 \cdot 10^{-2}$

Tabelle 3.11: Vergleich der Abweichungen der Spannungskomponenten MSE_{σ}, der von Mises Spannung $\text{MSE}_{\sigma_{v,M}}$ und der Knotenpositionen MSE_{pl} sowie Knotenverschiebungen MAE^{rel}_{disp}

Während für das rekurrente ML^{LMSC} Modell kaum Abweichungen zu erkennen sind, treten beim LSTM Modell sehr kleine lokale Abweichungen an einzelnen Stellen auf. Das Simulationsergebnis für das quadratische FF-LMSC-Modell $ML^{FF-LMSC,3-100,qu}$ zeigt global eine ähnliche Spannungsverteilung (Bereiche mit den höheren Spannungswerten

stimmen überein, sind homogen und werden gut getroffen), aber die Spannungswerte werden insgesamt und vor allem während des späteren Simulationsverlaufs im Bereich der größeren Spannungen vom ML-Modell unterschätzt.

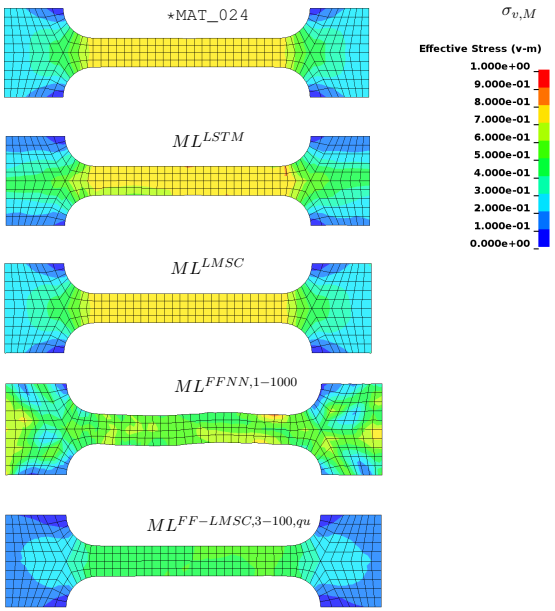


Abbildung 3.34: Vergleich des D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ der verschiedenen ML-Modelle und des klassischen Materialmodells *MAT_024 für den letzten Zeitschritt. Für die rekurrenten ML-Modelle ML^{LSTM} und ML^{LMSC} sind kaum Abweichungen zu erkennen. Für die Feed Forward Modelle $ML^{FFNN,1-1000}$ und $ML^{FF-LMSC,3-100,qu}$ sind die Abweichungen deutlich größer, was sowohl an der Spannungsverteilung auf der Fläche als auch an den Randpunkten auf der rechten Seite, welche eine höhere Verschiebung aufweisen und weiter gezogen werden, ersichtlich wird.

Deutlich größere Abweichungen sind für das Modell $ML^{FFNN,1-1000}$ zu sehen. Es tauchen hierbei nicht nur lokal an einzelnen Elementen Abweichungen in der Spannungsprognose auf, sondern auch Abweichungen in der Verteilung der Probe. Insbesondere sind die Spannungen im mittleren Teil der Probe nicht homogen. Zudem sind im Vergleich zu den anderen aufgezeigten Modellen auch Diskrepanzen in der Geometrie zu erkennen (Probe knickt seitlich weg). Diese deutlichere Abweichung (vor allem in späteren Zeitschritten) ist auch in der direkten Spannungsantwort (Abbildung 3.36) zu sehen. Abbildung 3.35 zeigt die Verläufe der Abweichungen der von-Mises-Spannungen $\sigma_{v,M}$ und der Knotenpositionen der Zugversuch Simulationen mit den verschiedenen ML-Materialmodellen.

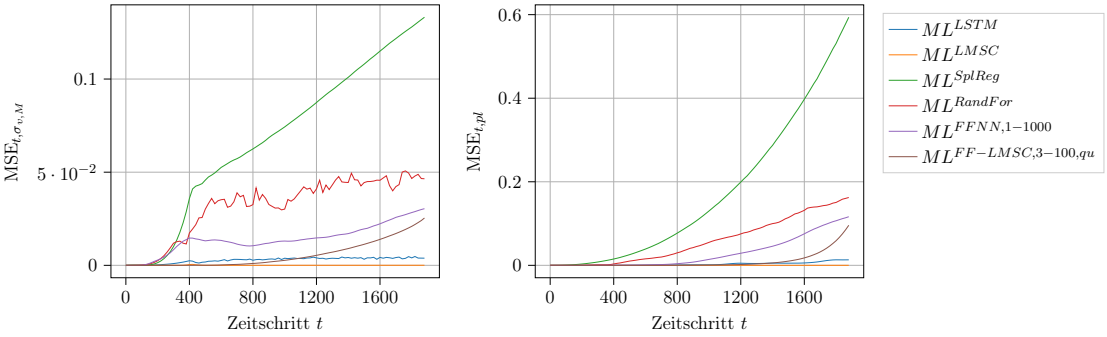


Abbildung 3.35: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der von-Mises-Spannungen $\mathrm{MSE}_{t,\sigma_{v,M}}$ und den Knotenpositionen $\mathrm{MSE}_{t,pl}$ für die Zugversuch Simulation

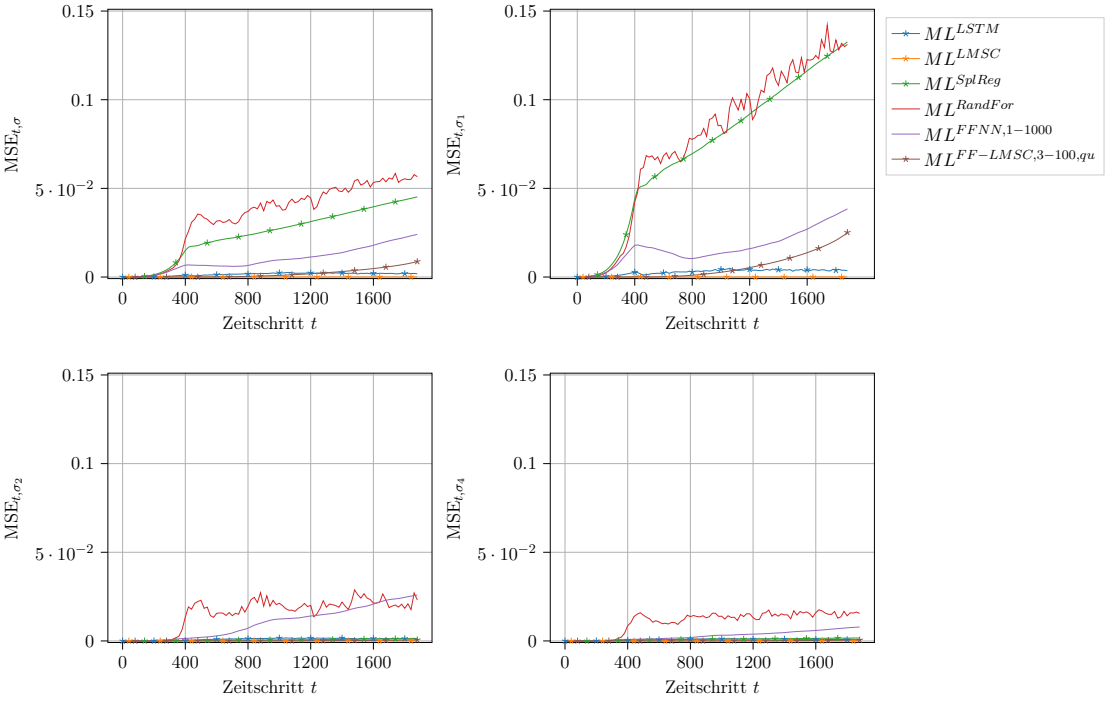
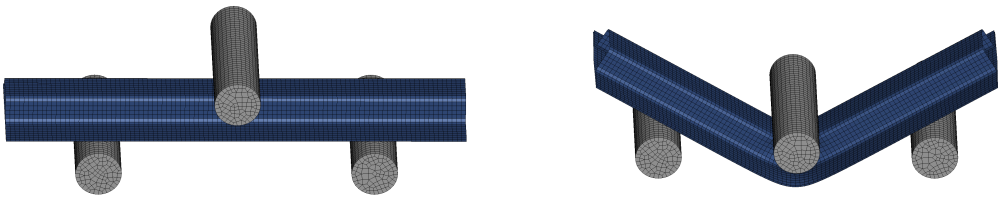


Abbildung 3.36: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der einzelnen Spannungskomponenten für die Zugversuch Simulation

Hutprofil

Als weitere Validierungssimulation dient die bereits in Abschnitt 3.2.3 vorgestellte Hutprofil Simulation, aus welcher der Validierungsdatensatz D^{hat} extrahiert wurde. Der Impaktor trifft auch hier mit einer initialen Geschwindigkeit von $-20\frac{m}{s}$ in z-Richtung auf das Profil, bestehend aus 6400 Schalenelementen, welche durch das jeweilige ML-Materialmodell ersetzt werden (siehe Abbildung 3.37). Die Simulation läuft für 5 Millisekunden, diskretisiert in insgesamt 8064 Zeitschritte.



(a) Simulation Hutprofil bei $t = 0$ (b) Simulation Hutprofil bei $t = 8064$

Abbildung 3.37: Aufbau Hutprofil Simulation

Abbildung 3.38 zeigt den D3plot nach 1280 und 8064 Zeitschritten für die Simulationen, welche mit den Modellen ML^{LSTM}, ML^{LMSC} und $ML^{FF-LMSC,3-100,qu}$ durchgeführt wurden sowie der Simulation mit dem klassischen Materialmodell `*MAT_024` zum Vergleich (wobei jeweils der Impaktor zur besseren Sichtbarkeit ausgeblendet wird). Alle drei ML-Modelle zeigen kaum erkennbaren Diskrepanzen im Laufe der Simulation bezüglich der Geometrie, allerdings sind kleinere Abweichungen in der Spannungsantwort im späteren zeitlichen Verlauf, vorrangig für die Modelle ML^{LMSC} und $ML^{FF-LMSC,3-100,qu}$, zu sehen, was auch an den Fehlerverläufen MSE_σ und MSE_{σ_i} in Abbildung 3.40 erkennbar wird. Trotz einer höheren MSE_σ-Abweichung schneidet das Modell ML^{LMSC} bezüglich der Verschiebungen und Positionen der Knoten innerhalb der Simulation etwas besser ab (siehe Tabelle 3.12, Abbildungen 3.41 und 3.39), was darauf hindeutet, dass nicht nur die Gesamtabweichung (bezüglich der Spannungen) der Modelle für eine gute Performanz innerhalb der Simulation ausschlaggebend ist, sondern auch an welchen Punkten, in welcher Kombination (Abweichung in lokaler Umgebung oder Abweichung der weiteren Komponenten) und in welchem Ausmaß die Abweichungen auftreten.

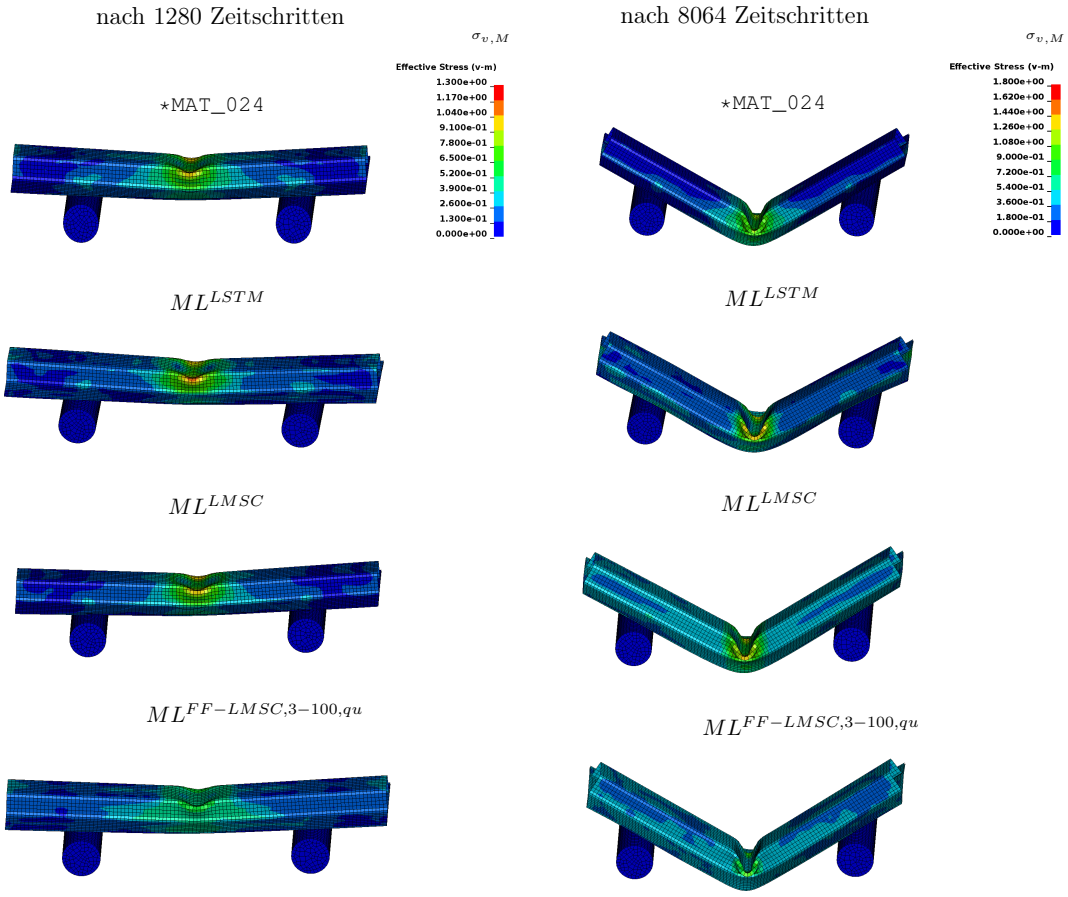


Abbildung 3.38: Vergleich D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ der verschiedenen ML-Modelle und des klassischen Materialmodells

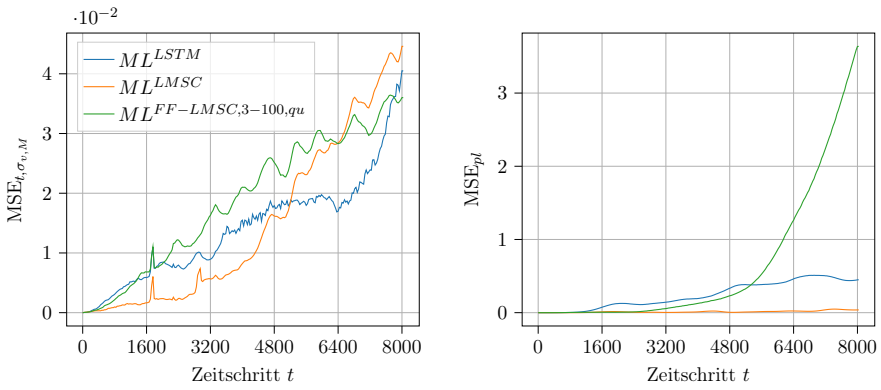


Abbildung 3.39: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der von-Mises-Spannungen $\text{MSE}_{t,\sigma_{v,M}}$ und den Knotenpositionen $\text{MSE}_{t,pl}$ für die Hutprofil Simulation

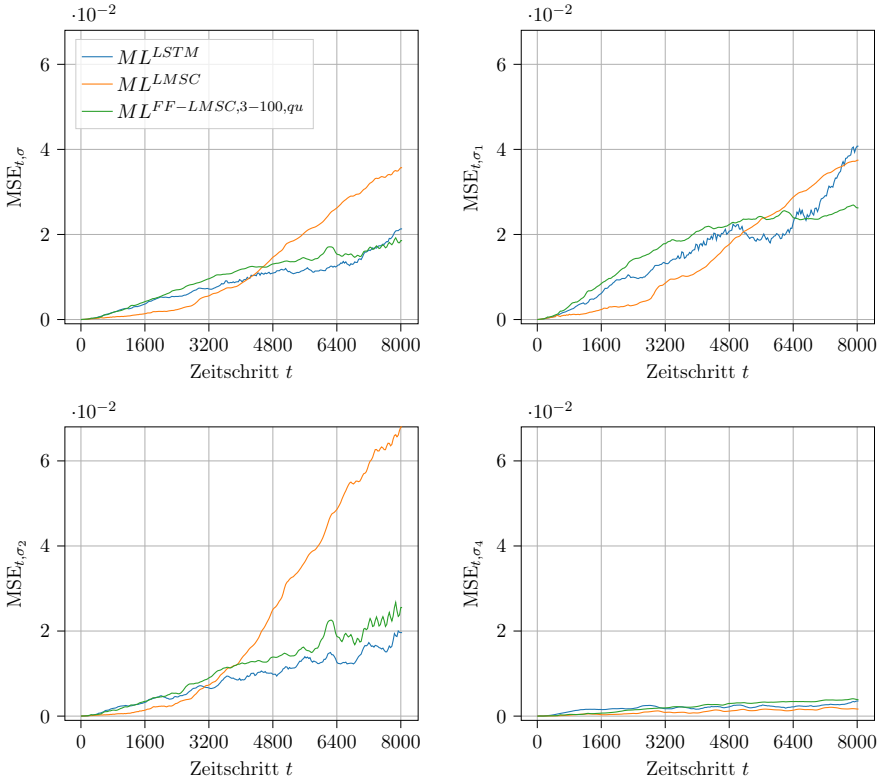


Abbildung 3.40: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der einzelnen Spannungskomponenten für die Hutprofil Simulation

	MSE_{σ}	$\text{MSE}_{\sigma_{v,M}}$	MSE_{pl}	R^2_{pl}	MAE^{rel}_{disp}
$ML^{FF-LMSC,3-100,qu}$	$1.024 \cdot 10^{-2}$	$1.878 \cdot 10^{-2}$	$6.332 \cdot 10^{-1}$	0.9999871	$1.102 \cdot 10^{-1}$
ML^{LSTM}	$8.925 \cdot 10^{-3}$	$1.414 \cdot 10^{-2}$	$2.441 \cdot 10^{-1}$	0.9999882	$8.842 \cdot 10^{-2}$
ML^{LMSC}	$1.297 \cdot 10^{-2}$	$1.481 \cdot 10^{-2}$	$1.374 \cdot 10^{-2}$	0.9999993	$6.094 \cdot 10^{-2}$

Tabelle 3.12: Vergleich der Abweichungen der Spannungskomponenten MSE_{σ}, der von-Mises-Spannung $\text{MSE}_{\sigma_{v,M}}$ und der Knotenpositionen MSE_{pl} sowie Knotenverschiebungen MAE^{rel}_{disp}

Alle drei ML-Modelle $ML^{FF-LMSC,3-100,qu}$, ML^{LSTM}, ML^{LMSC} (welche auch schon im Zugversuch die besten Ergebnisse zeigen) liefern ebenfalls solide Simulationsergebnisse für diesen Anwendungsfall. Dies zeigt auch die Verschiebungskurve des Impaktors in Abbildung 3.41, welche kaum von der entsprechenden Verschiebungskurve der Simulation mit dem klassischen Materialmodell abweicht. Weitere getestete ML-Modelle brechen in der Hutprofil Simulation entweder frühzeitig ab ($ML^{RandFor}$ und $ML^{FFNN,1-1000}$) oder

haben eine mehr als zehnfach höhere relative Abweichung MAE_{disp}^{rel} (ML^{SplReg}), was auch deutlich im D3plot zu sehen ist (siehe Abbildung 3.42).

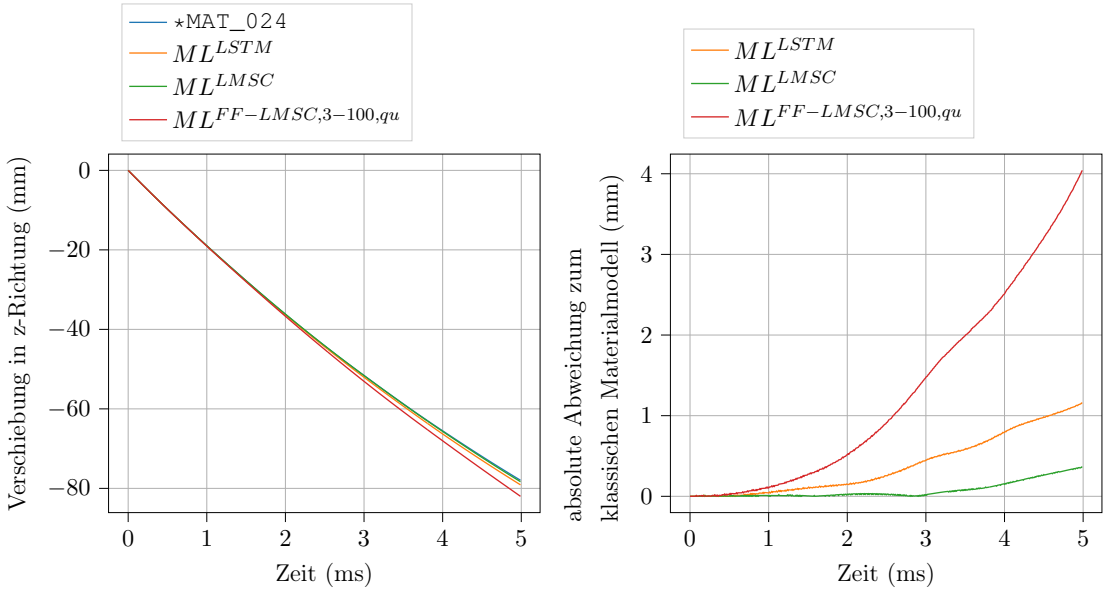
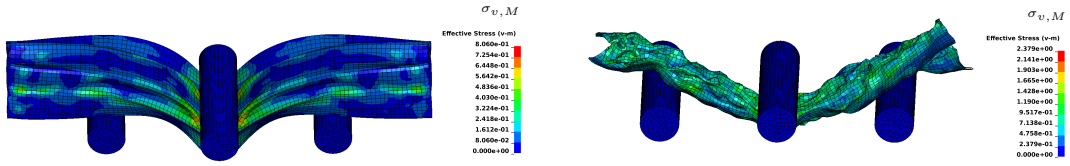


Abbildung 3.41: Links: Verschiebung des Impaktors in z-Richtung im Laufe der Hutprofil Simulation, Rechts: Absolute Abweichung der Verschiebung des Impaktors in z-Richtung im Vergleich zur Hutprofil Simulation mit dem klassischen Materialmodell *MAT_024



ML^{SplReg} zum Zeitschritt $t = 8064$ $ML^{RandFor}$ zum Zeitschritt $t = 7168$

Abbildung 3.42: D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ für den Random Forest und das Spline Regressionsmodell

Crashbox

Die Crashbox dient als Anwendungsfall im Bereich Crash. Die Simulation besteht abgesehen von der Crashbox selbst aus dem Impaktor (in Abbildung 3.43 auf der linken Seite der Crashbox zu sehen), welcher mit einer initialen Translationsgeschwindigkeit von $2.72\frac{m}{s}$ in x-Richtung auf die Crashbox trifft. Die Crashbox wird dabei durch ein Supportbauteil auf der anderen Seite (rechts im Bild) festgehalten. Der Aufbau der Simulation zu Beginn und das Ergebnis nach 13 ms im letzten Zeitschritt t=27270 mit dem klassischen Materialmodell sind in Abbildung 3.43 zu sehen.

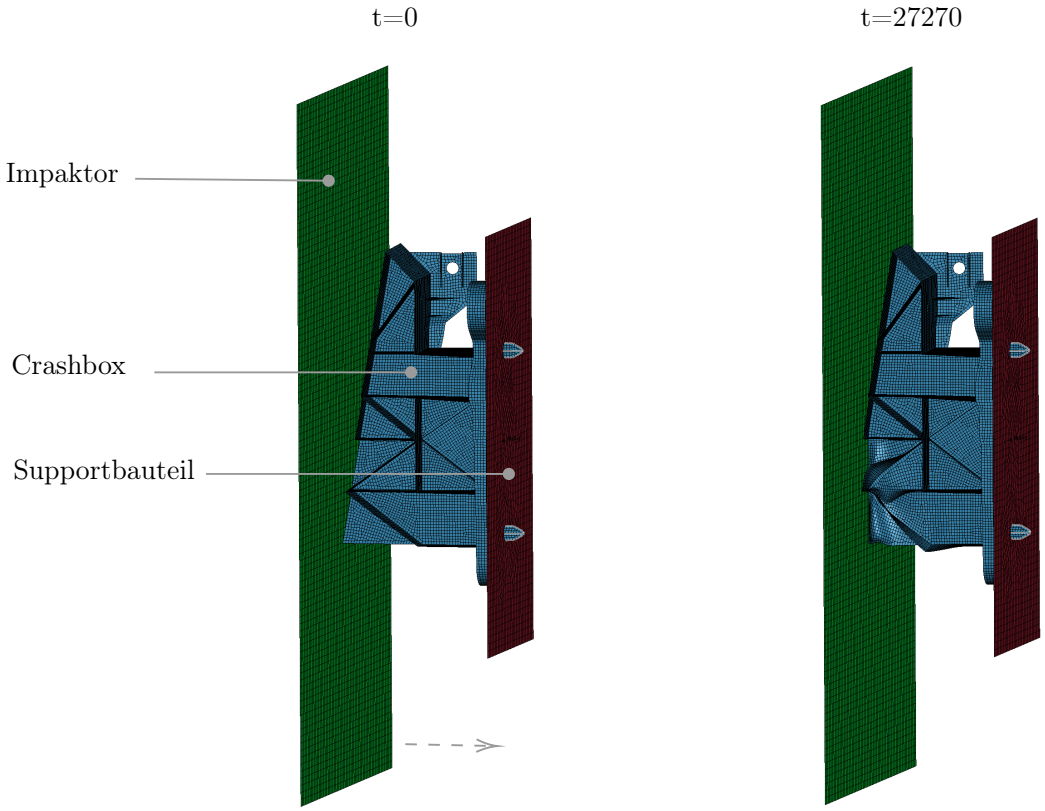


Abbildung 3.43: Links: Aufbau der Crashbox Simulation für t=0, Rechts: Das Simulationsergebnis im letzten Zeitschritt t=27270 mit dem nach rechts verschobenen Impaktor (grün) und der deformierten Crashbox (blau), welche auf der rechten Seite von dem Supportbauteil (rot) fixiert wird

Abbildung 3.44 zeigt den D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ nach den finalen 27270 Zeitschritten für die Simulationen, welche mit den Modellen ML^{LSTM} und ML^{LMSC} durchgeführt wurden, wobei die Simulationen bei beiden ML-Modellen

vollständig durchlaufen. Weiterhin zeigt Abbildung 3.44 das Simulationsergebnis des klassischen Materialmodells ⋆MAT_024 zum Vergleich. Die Simulationsergebnisse des Modells ML^{LMSC} sind bezogen auf die Geometrie und Verformung sehr ähnlich zum Simulationsergebnis mit dem klassischen Materialmodell, wobei an einigen Punkten die von-Mises-Spannungen überschätzt werden. Größere Abweichungen (insbesondere an einigen Punkten auch leicht an der Geometrie zu sehen) sind für das Modell ML^{LSTM} zu erkennen.

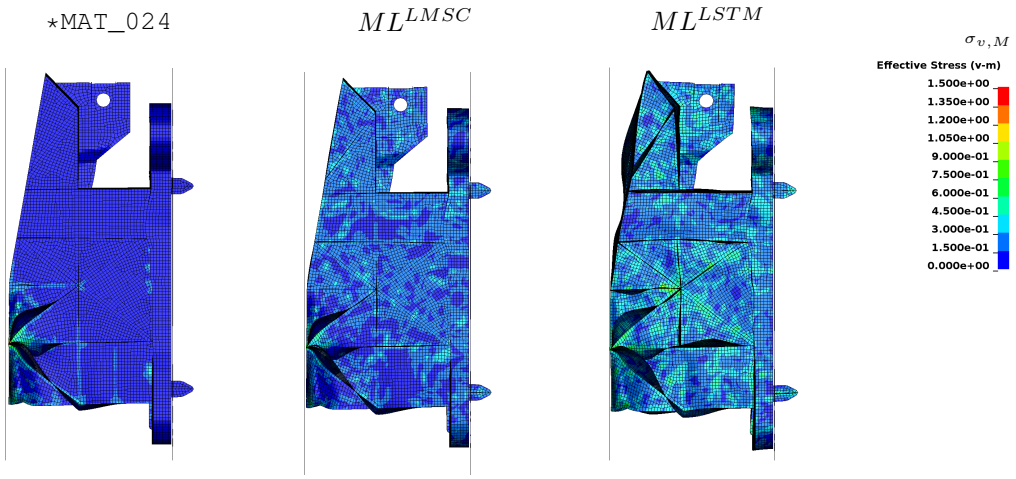


Abbildung 3.44: Vergleich D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ der ML-Modelle ML^{LMSC} und ML^{LSTM} sowie des klassischen Materialmodells ⋆MAT_024 für t=27270

Ebenfalls deutlich wird dies in Abbildung 3.45, welche den Verlauf der Abweichung der von-Mises-Spannung und der Knotenpositionen über die Zeitschritte zeigt. Während für das ML-Materialmodell ML^{LSTM} eine deutlich wachsende Abweichung bezüglich der Position der Knotenpunkte MSE_{pl} im Laufe der Simulation zu sehen ist, steigt diese Abweichung für ML^{LMSC} im Laufe der Zeit nur leicht an. Im Vergleich zu den bisher betrachteten Anwendungsfällen (Zugversuch und Hutprofil) ist die relative Abweichung der Knotenverschiebungen (siehe Tabelle 3.13) für beide Modelle deutlich höher. Begründet werden kann dies durch die deutlich längere Simulationszeit mit wesentlich mehr Zeitschritten, wodurch die Abweichung zur Simulation mit dem klassischen Materialmodell nach und nach ansteigt.

	MSE_{σ}	$\text{MSE}_{\sigma_{v,M}}$	MSE_{pl}	R^2_{pl}	MAE^{rel}_{disp}
ML^{LSTM}	$1.508 \cdot 10^{-2}$	$4.871 \cdot 10^{-2}$	0.346	0.9999128	0.533
ML^{LMSC}	$1.737 \cdot 10^{-2}$	$3.648 \cdot 10^{-2}$	0.014	0.9999971	0.149

Tabelle 3.13: Vergleich der Abweichungen der Spannungskomponenten MSE_{σ}, der von-Mises-Spannung $\text{MSE}_{\sigma_{v,M}}$ und der Knotenpositionen MSE_{pl} sowie Knotenverschiebungen MAE^{rel}_{disp} und die Werte für R^2_{pl} für die Crashbox Simulation

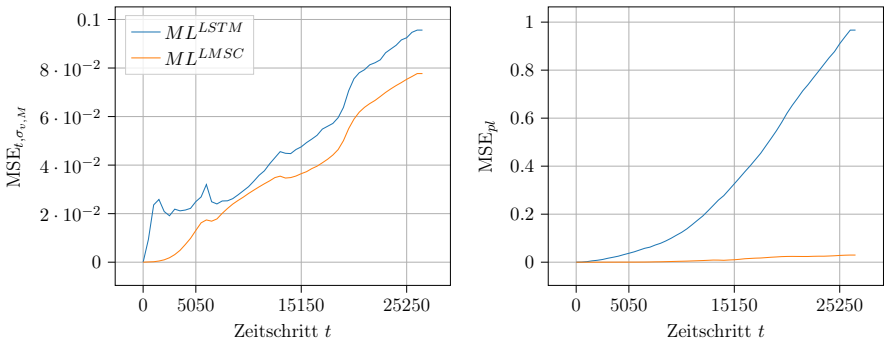


Abbildung 3.45: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der von-Mises-Spannungen $\text{MSE}_{t,\sigma_{v,M}}$ und den Knotenpositionen $\text{MSE}_{t,pl}$ für die Crashbox Simulation

Die Abweichung der Spannungskomponenten σ_1 und σ_2 bewegt sich im Laufe der Simulation für beide Modelle ML^{LSTM} und ML^{LMSC} in einem ähnlichen Wertebereich mit steigender Tendenz. Die Abweichung der Schubspannung σ_4 ist für ML^{LSTM} permanent höher. Im Mittel ist die Spannungsabweichung allerdings, wie der Tabelle 3.13 zu entnehmen ist, für das ML^{LSTM} Modell höher. Abbildung 3.47 zeigt die Verschiebung des Impaktors in x-Richtung für die beiden Simulationen mit den ML-Modellen und dem klassischen Materialmodell im direkten Vergleich. Erst nach 8 Millisekunden ist ein erkennbarer, jedoch kleiner Unterschied darin zu erkennen, dass bei den Simulationen mit den ML-Modellen der Impaktor etwas stärker eindrückt (Maximum der Kurve liegt für ML^{LSTM} bei 15,54 mm, für ML^{LMSC} bei 15,22 mm und für `*MAT_024` bei 15,18 mm) und anschließend weniger deutlich wieder zurückprallt.

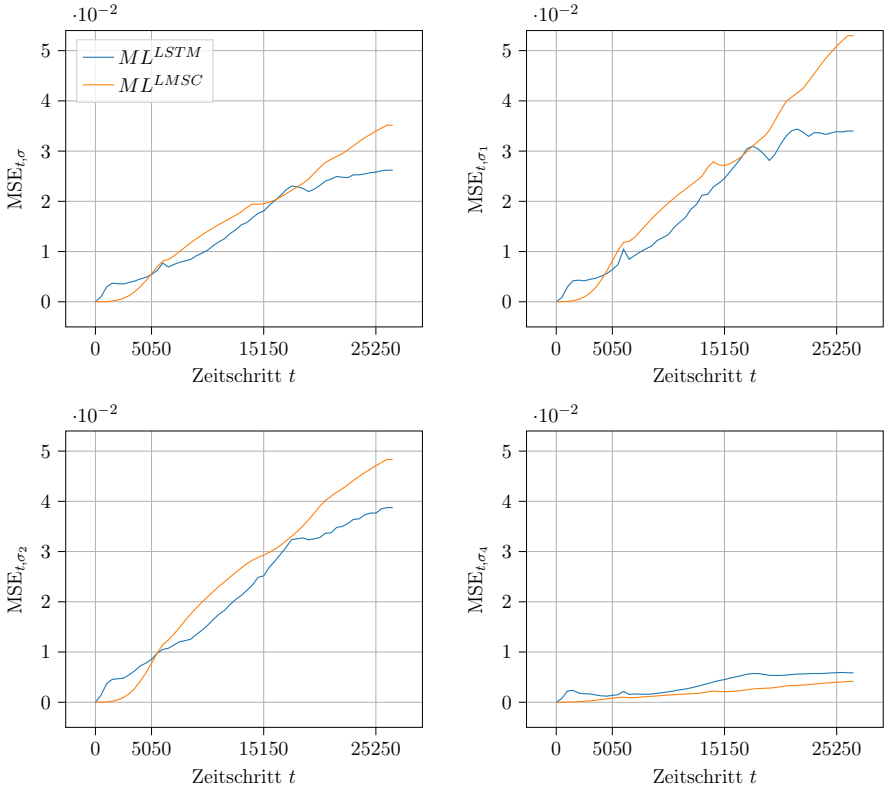


Abbildung 3.46: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der einzelnen Spannungskomponenten für die Crashbox Simulation

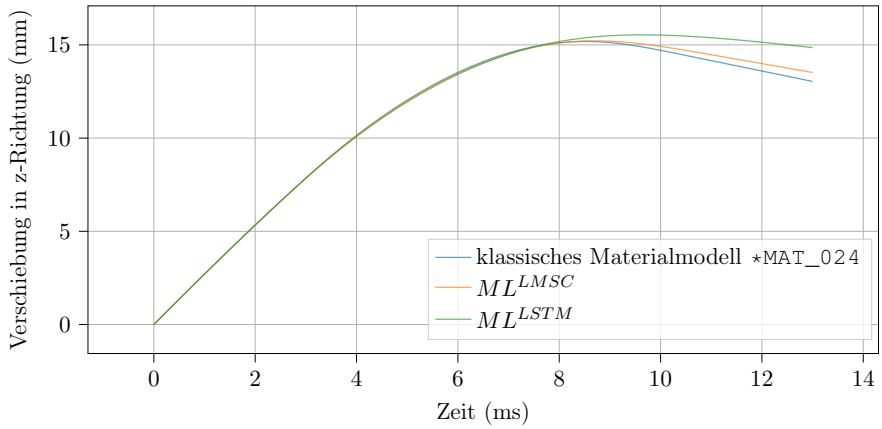


Abbildung 3.47: Verschiebung des Impaktors in x-Richtung im Laufe der Crashbox Simulation

Kreuznapf

Die Kreuznapf Simulation dient als Anwendungsbeispiel aus dem Bereich der Umformtechnik. Als Umformen wird eine Art der Fertigung von Bauteilen durch die plastische Änderung der Form eines Werkstoffes unter Beibehaltung dessen Masse bezeichnet (siehe [4]). Bei der hier betrachteten Kreuznapf Umformsimulation wird, wie in den Abbildungen 3.48 und 3.49 zu sehen, der sogenannte *Stempel* (grau) auf die *Platine* (blau), welche mit dem ML-Materialmodell gerechnet wird und aus 4570 Schalenelementen besteht, im Laufe von 25 ms, diskretisiert in 111112 Zeitschritte, aufgebracht. Die Platine ist dabei am Rand zwischen dem sogenannten *Niederhalter* (grün) und der *Matrize* (gelb) eingespannt. Der Stempel trifft mit einer vorgegebenen Verschiebung von 40 mm in z-Richtung (Translation) auf die eingespannte Platine und verformt diese. Abbildung 3.50 zeigt die durch den Stempel im Laufe der Simulation verformte Platine nach 46662 Zeitschritten.

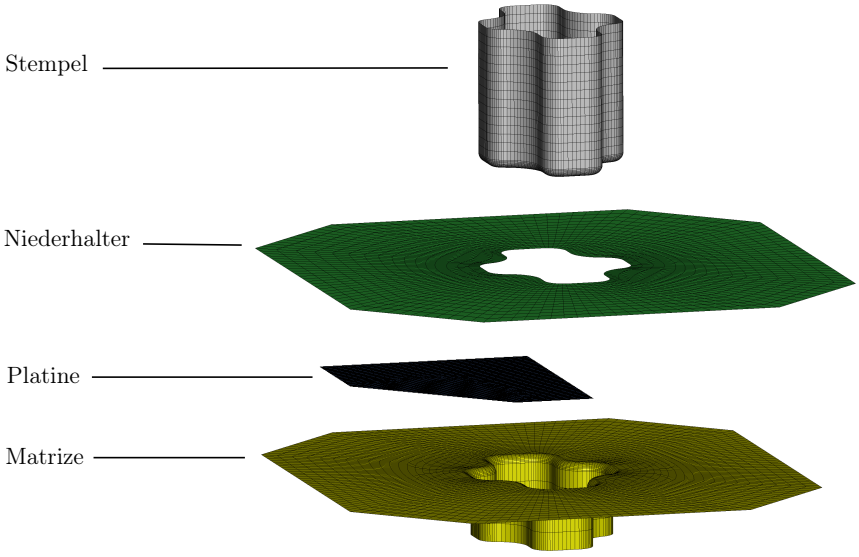


Abbildung 3.48: Die verschiedenen Bestandteile der Kreuznapf Umformsimulation: Der Stempel verformt die Platine, welche durch den Niederhalter fixiert wird und die Matrize gibt die endgültige Form vor

In Abbildung 3.51 ist der D3plot der eingedrückten Platine bezüglich der von-Mises-Spannung $\sigma_{v,M}$ für die Simulationen mit den beiden ML-Materialmodellen ML^{LSTM} und ML^{LMSC} sowie der Simulation mit dem klassischen Materialmodell zu sehen. Für das Modell ML^{LMSC} sind kaum Unterschiede zu der Simulation mit dem klassischen Materialmodell in der Geometrie zu erkennen. Die von-Mises-Spannungen sind in einem ähnlichen Wertebereich und auch Stellen mit höheren Werten stimmen mit den Ergebnissen

des klassischen Modells überein. Im Unterschied dazu sind beim Modell ML^{LSTM} in beiden Aspekten deutliche Abweichungen zum klassischen Modell erkennbar. Die von-Mises-Spannung $\sigma_{v,M}$ wird von diesem Modell an vielen Stellen überschätzt und ein Knittern in der Geometrie ist im Bereich der Einspannung zu sehen. Die größeren Abweichungen des Modells bezüglich der Geometrie und den Spannungswerten werden auch in Abbildung 3.52 und in Tabelle 3.14 deutlich.

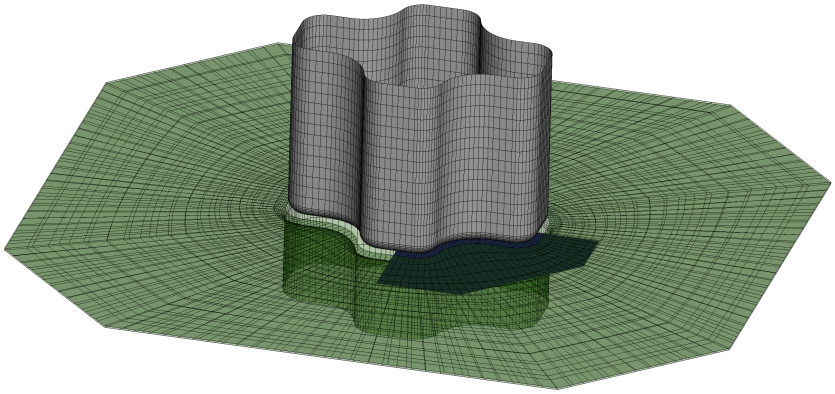
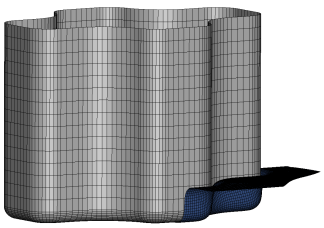
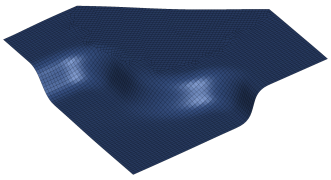


Abbildung 3.49: Gesamter Aufbau der Kreuznapf Umformsimulation für t=0: Platine (blau), welche zwischen der Matrize und dem Niederhalter (grün) eingespannt ist und der Stempel (grau), welcher im Laufe der Simulation auf die Platine trifft und diese plastisch verformt



(a) Seitliche Ansicht der Platine (blau) mit Stempel (grau) bei t= 46662



(b) Ansicht der leicht eingedrückten Platine ohne Stempel von schräg oben bei t=46662

Abbildung 3.50: Kreuznapf Umformsimulation für den Zeitschritt t=46662 (Stempel und Platine)

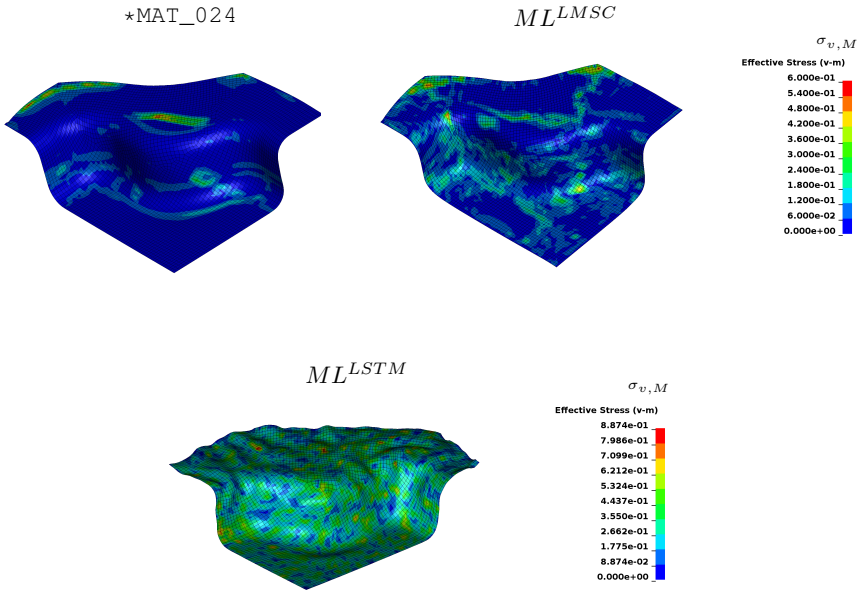


Abbildung 3.51: Vergleich D3plot bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ der ML-Modelle ML^{LMSC} und ML^{LSTM} sowie des klassischen Materialmodells *MAT_024 für t=111112

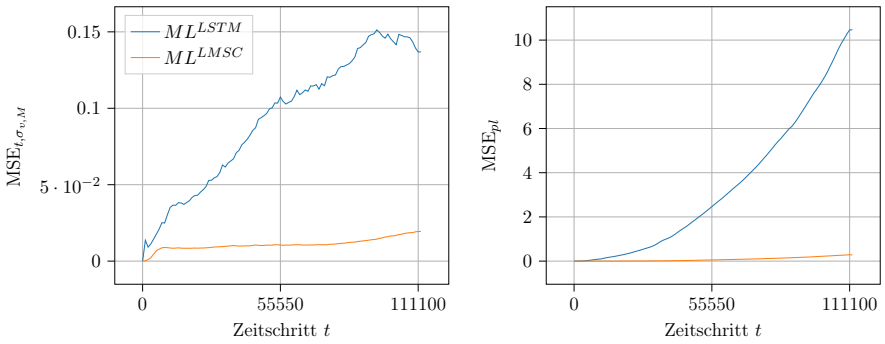


Abbildung 3.52: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der von-Mises-Spannungen $\text{MSE}_{t,\sigma_{v,M}}$ und den Knotenpositionen $\text{MSE}_{t,pl}$ für die Kreuznapf Simulation

Für diesen Anwendungsfall ist die ansteigende Abweichung in jederlei Hinsicht (Spannungskomponenten, von-Mises-Spannung und Knotenpositionen) für das Modell ML^{LSTM} deutlich höher als für das Modell ML^{LMSC}. Die Abweichungen steigen für das Modell ML^{LMSC} deutlich langsamer. Die relative Abweichung MAE^{rel}_{disp} ist für ML^{LMSC} bei der Kreuznapf Simulation trotz einer deutlich höheren Anzahl an Zeitschritten sogar niedriger als bei der Crashbox oder der Hutprofil Simulation.

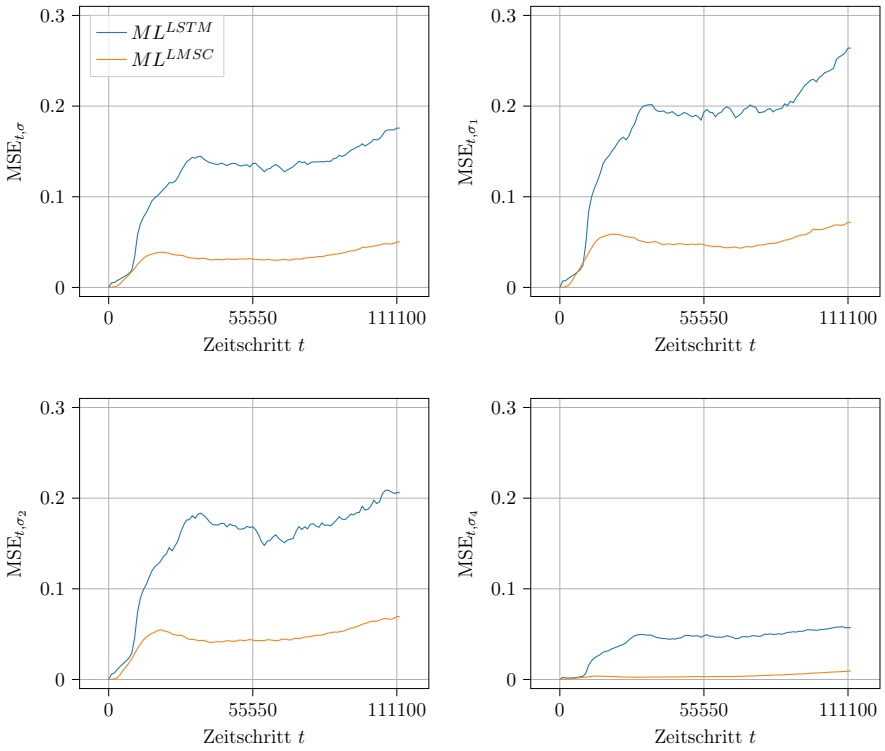


Abbildung 3.53: Zeitliche Verläufe der mittleren quadratischen Abweichung bezüglich der einzelnen Spannungskomponenten für die Kreuznapf Simulation

	MSE_{σ}	$\mathrm{MSE}_{\sigma_{v,M}}$	MSE_{pl}	R^2_{pl}	$\mathrm{MAE}^{rel}_{disp}$
ML^{LSTM}	0.124	$9.293 \cdot 10^{-2}$	3.398	0.9995258	0.157
ML^{LMSC}	0.033	$1.099 \cdot 10^{-2}$	0.083	0.9999883	0.031

Tabelle 3.14: Vergleich der Abweichungen der Spannungskomponenten MSE_{σ}, der von-Mises-Spannung $\mathrm{MSE}_{\sigma_{v,M}}$ und der Knotenpositionen MSE_{pl} sowie Knotenverschiebungen $\mathrm{MAE}^{rel}_{disp}$ und die Werte für R^2_{pl} für die Kreuznapf Simulation

Fazit

Zusammengefasst lässt sich sagen, dass ML-Materialmodelle in unterschiedlichen Anwendungen (Crash und Umformen) das klassische Materialmodell ersetzen können und (je nach ML-Modell) nur kleine Abweichungen auftauchen. Der Fehler kann sich jedoch mit der Zeit akkumulieren. In Simulationen mit sehr vielen Zeitschritten kann dies zu einer zunehmend großen Abweichung der Ergebnisse oder sogar zum Abbruch der Simulation führen. Es ist eine sehr hohe Vorhersagegenauigkeit an die Modelle gefordert,

um eine stabile Simulation (kein Abbruch der Simulation) und ähnliche Ergebnisse im Vergleich zum klassischen Materialmodell zu gewährleisten. Dies kann insbesondere, wie die durchgeführten Hyperparameterstudien zeigen, bei konventionellen neuronalen Netzen durch eine ausreichend große Netzstruktur (bezüglich der Anzahl an Schichten und Neuronen je Schicht) und damit hinreichend vielen trainierbaren Parametern erreicht werden. Das Einbringen der exakten Vorhersage von Nullspannungen in die Modellstruktur der ML-Materialmodelle, wie es bei den LMSC-Modellen der Fall ist, stellt eine wichtige Voraussetzung dar, um die Performanz der Modelle innerhalb der Simulationen deutlich zu verbessern. Diese Bedingung an die ML-Modelle kann auch für die Feed Forward neuronalen Netze im Rahmen der FF-LMSC-Modelle umgesetzt werden. Im Allgemeinen bieten jedoch rekurrente ML-Modelle (neuronale Netze) eine in der Anwendung deutlich bessere Möglichkeit zur Materialmodellierung als Modelle, welche auf einzelnen Zeitschritten unabhängig mit Berücksichtigung der internen Geschichtsvariablen trainiert werden. Erklären lässt sich dies dadurch, dass die rekurrenten ML-Modelle auf gesamten zeitlichen Dehnungs-Spannungsverläufen trainiert werden und somit die Fehlerfortpflanzung innerhalb des Trainings (mittels Backpropagation through time, siehe [72]) schon im Training berücksichtigt wird. Da das LMSC-Modell, welches sowohl auf gesamten zeitlichen Verläufen trainiert wird als auch die Voraussetzung der exakten Nullvorhersage erfüllt, in den meisten Evaluierungen und insbesondere innerhalb der diversen Validierungssimulationen die besten Ergebnisse liefern konnte, wird im anschließenden Kapitel 4 für das Materialmodell mit Plastizität dieses ML-Materialmodell betrachtet.

4 Training von ML-Materialmodellen basierend auf Versuchsdaten

Um die Daten für das Training von ML-Materialmodellen zu erzeugen, wurden bisher die klassischen Materialmodelle für ein gegebenes Material benötigt. Für einen neuartigen Werkstoff muss solch ein Materialmodell aufwendig auf Basis von Charakterisierungsversuchen (mechanische Prüfverfahren, deren Ergebnisse die Eigenschaften eines Materials erkennbar machen) erstellt beziehungsweise angepasst (kalibriert) werden. Die auf Basis der klassischen Modelle trainierten ML-Materialmodelle dienen daher lediglich als Ersatzmodelle für bestehende klassische Materialmodelle. Um die Notwendigkeit der Erstellung dieser Modelle zu umgehen, wird in diesem Kapitel eine neuartige Methode, welche bereits in [9] veröffentlicht wurde, vorgestellt, um ein ML-Materialmodell für einen neuartigen Werkstoff direkt ausschließlich basierend auf Daten aus diesen Charakterisierungsversuchen zu trainieren.
Im Unterschied zu den Trainingsdaten, welche mithilfe der klassischen Materialmodelle erzeugt werden können, sind bei den Versuchen die lokalen, punktuell wirkenden Spannungen nicht verfügbar. Das ML-Materialmodell kann daher innerhalb des Trainings nicht direkt über den Vergleich zwischen den vorhergesagten und korrekten Spannungen evaluiert werden. Der hier vorgestellte Lösungsansatz besteht darin stattdessen physikalisch motivierte Gleichgewichtsbedingungen, welche in Abschnitt 4.2 näher erläutert werden, für die Evaluierung der ML-Materialmodelle innerhalb des Trainings zu benutzen. Um den Trainingsprozess zu beschleunigen, sollen dabei Methoden des *Transfer-Learning* (siehe [27]) zum Einsatz kommen und ML-Materialmodelle, welche bereits für ein Material mit ähnlichen Eigenschaften anhand eines vorhandenen klassischen Materialmodells wie in Kapitel 3 trainiert wurden, für den neuen Werkstoff angepasst werden.

4.1 Verfügbare Trainingsdaten und grundlegende Methodik

Um für einen neuartigen Werkstoff im Rahmen der klassischen Materialmodellierung passende Materialkennwerte zu ermitteln, werden unterschiedliche Versuche wie beispielsweise Zugversuche oder Scherversuche (siehe [63]) mit diesem Werkstoff durchgeführt.

Bei diesen Charakterisierungsversuchen wird eine Kraft von einer Prüfmaschine auf eine vorgegebene Probengeometrie des entsprechenden Werkstoffes aufgebracht. Dabei werden zusätzlich zu der aufgebrachten Kraft die lokalen Verschiebungen und Dehnungen zu diskretisierten Zeitschritten während des Versuches gemessen. Letztere können durch optische Messmethoden, wie der etablierten *digitalen Bildkorrelation* (*DIC - Digital Image Correlation*) [53], bestimmt werden. Beim DIC-Verfahren wird die Bewegung eines auf der Probe aufgebrachten stochastischen Musters durch mehrere Kameras im Laufe des Versuches erfasst. Es stehen die dadurch erhaltenen Dehnungskomponenten für die unterschiedlichen Punkte auf der Probe zu den unterschiedlichen Zeitpunkten, welche als Eingabe des ML-Materialmodells dienen sowie die globale aufgebrachte Kraft zu den einzelnen Zeitschritten für das Training der ML-Materialmodelle zur Verfügung. Nicht messbar in diesen Versuchen ist jedoch, wie die globale Kraft auf der Probengeometrie lokal verteilt ist. Die punktuell wirkenden Spannungen für die einzelnen räumlichen Punkte und Zeitpunkte, welche die korrekten Ausgaben des Materialmodells darstellen, sind daher für den Lernprozess der ML-Modelle nicht verfügbar. Üblicherweise jedoch lernen ML-Modelle im Bereich des überwachten maschinellen Lernens auf Basis des Vergleichs zwischen den vorhergesagten und den korrekten Zielausgaben, im Fall der ML-Materialmodelle den Spannungen, welche als Ausgabe eines klassischen, analytischen Materialmodells gegeben ist.

Die vorgeschlagene Methodik für das Training von ML-Materialmodellen auf Basis von Versuchsdaten nutzt statt dem direkten Vergleich von korrekten und vorhergesagten Ausgaben ausschließlich Formulierungen von physikalisch motivierten Gleichungen, welche für die vom ML-Materialmodell vorhergesagten Spannungen so gut wie möglich erfüllt werden sollen, ohne die korrekten Spannungswerte zu benutzen. Abbildung 4.1 zeigt eine Übersicht der entwickelten Methode, um ML-Materialmodelle basierend auf den Dehnungen und der globalen Kraft aus einem Versuch zu trainieren, ohne Spannungen aus einem zugrunde liegenden klassischen Materialmodell zu benötigen. Als Start für das Training dient ein ML-Materialmodell, welches bereits anhand eines weiteren Werkstoffes mit ähnlichen Eigenschaften trainiert wurde. Die Art des ML-Modells (beispielsweise Netzarchitektur eines neuronalen Netzes) sowie die Initialisierung der trainierbaren Parameter des Modells wird dabei vom gewählten Startmodell festgelegt. Dieses vortrainierte ML-Materialmodell wird dabei auf Basis eines vorhandenen Materialmodells mittels Dehnungs-Spannungs-Daten wie in Kapitel 3 trainiert. Die Nutzung eines vortrainierten ML-Materialmodells dient hierbei einerseits dazu eine möglichst geeignete Modellarchitektur für das ML-Materialmodell des neuartigen Werkstoffes festzulegen und soll des Weiteren ein schnelleres Training ermöglichen. Für den neuartigen Werkstoff, für welchen das Startmodell angepasst werden

soll, werden Charakterisierungsversuche durchgeführt. Die Ergebnisse aus den Versuchen, welche aus der Gesamtreaktionskraft sowie dem Verschiebungsfeld auf der Oberfläche und den daraus resultierenden Dehnungen zu den jeweiligen Zeitpunkten bestehen, dienen als Datenbasis für das Training.

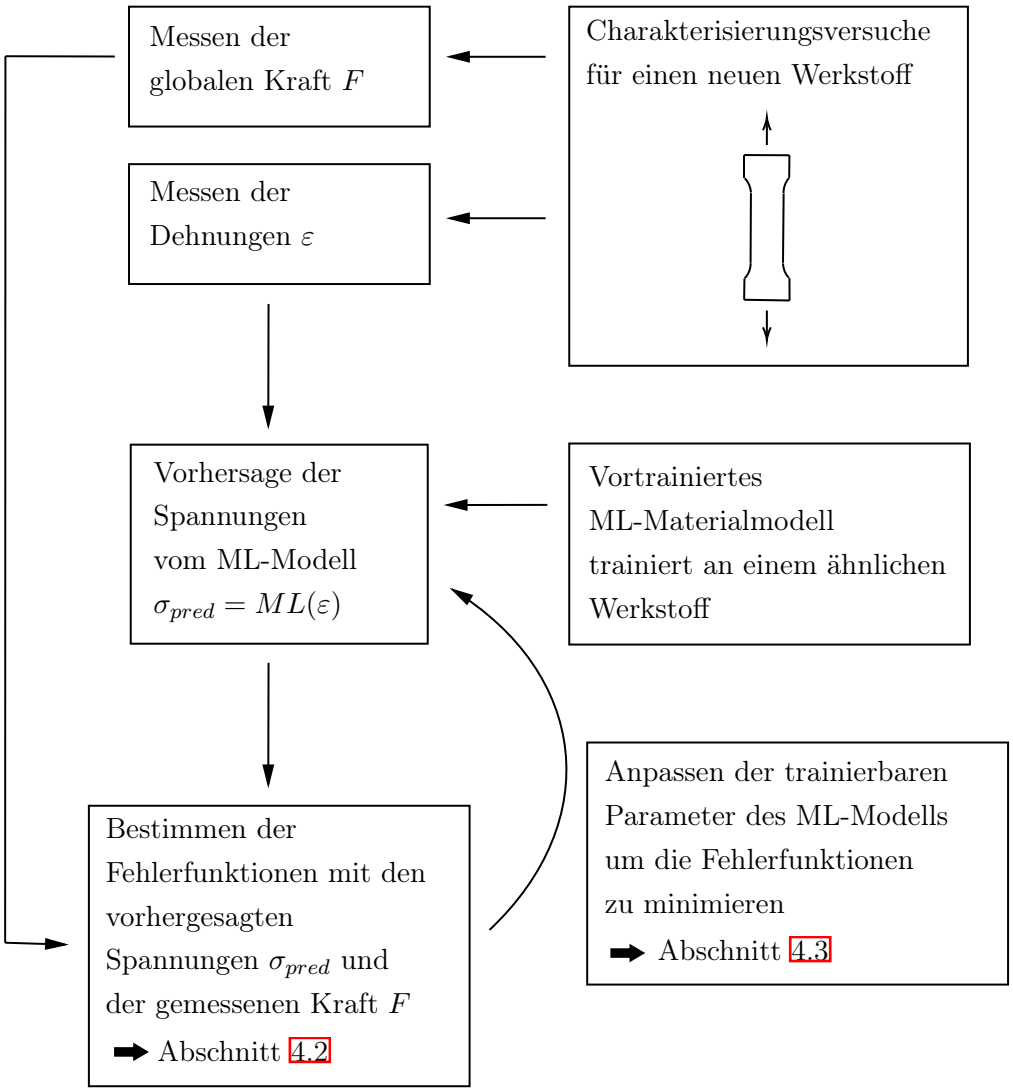


Abbildung 4.1: Übersicht der Trainingsmethode für ML-Materialmodelle basierend auf Versuchsdaten

Die Dehnungen an den gemessen Punkten der Probe zu den verschiedenen Zeitpunkten stellen dabei nach wie vor die Eingabegrößen für den Datensatz bei dem Training aus Versuchsdaten dar. Für gegebene Parameter des ML-Modells können hiermit (zunächst falsche) Spannungen bestimmt werden. Diese vom ML-Modell vorhergesagten Spannungen

können nicht direkt mit den korrekten verglichen werden, wie es üblicherweise beim Training gemacht werden würde. Stattdessen sollen mit diesen vorhergesagten Spannungen zusammen mit der im Versuch gemessenen Kraft festgelegt (in einer FE-Simulation geltenden) Gleichungen erfüllt werden. Durch die Abweichung zu diesen physikalisch motivierten Gleichungen werden alternative Fehlerfunktionen hergeleitet, auf Basis derer die trainierbaren Parameter des ML-Materialmodells angepasst werden. Die alternativen Fehlerfunktionen für das Training mit Versuchsdaten werden in Abschnitt 4.2 vorgestellt. In Abschnitt 4.3 werden mögliche Optimierungsstrategien dargestellt, um diese Fehlerfunktionen zu minimieren. Anschließend wird die Methode in Abschnitt 4.4 und 4.5 anhand unterschiedlicher ML-Modelle, Werkstoffe, Optimierungsstrategien und simulierter Versuche validiert.

4.2 Bestimmen der Fehlerfunktion

Zur Verfügung stehen für das Training der ML-Materialmodelle, wie in Abschnitt 4.1 erläutert, die im Versuch gemessene globale Kraft F sowie die durch das DIC-Verfahren gemessenen Verschiebungen u und die daraus resultierenden Dehnungen an den verschiedenen Punkten des Probekörpers zu den jeweiligen Zeitpunkten. Das trainierte ML-Materialmodell soll mit den vorhergesagten Spannungen eine möglichst gleiche Gesamtkraft F liefern und zu einem annähernd ähnlichem Verschiebungsfeld u führen, wenn der Probekörper der gleichen Gesamtverschiebung ausgesetzt wird, wie sie im Versuch gemessen wird.

4.2.1 Allgemeiner Fall

Für die vom ML-Materialmodell vorhergesagten Spannungen, ausgehend von den im Versuch gemessenen Dehnungen, sollen die folgenden drei Gleichungen

$$\int_V (\sigma : \varepsilon)\, dV = \int_A (k \cdot du)\, dA \tag{4.1}$$

$$div(\sigma) = \begin{pmatrix} \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + \frac{\partial \sigma_{xz}}{\partial z} \\ \frac{\partial \sigma_{yx}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \sigma_{yz}}{\partial z} \\ \frac{\partial \sigma_{zx}}{\partial x} + \frac{\partial \sigma_{zy}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.2}$$

$$k = \sigma \cdot n \tag{4.3}$$

erfüllt sein. Gleichung (4.1) beschreibt das global geltende Gleichgewicht der im Versuch gemessenen und der durch die vorhergesagten Spannungen bestimmte gesamt wirkende Kraft. Mit σ und ε wird der Spannungstensor beziehungsweise der Tensor der inkrementellen Dehnungen (als Funktion der Zeit und eines räumlichen Punktes $p \in V$) bezeichnet, wobei $\sigma : \varepsilon = \sum_{i=1}^{3} \sum_{j=1}^{3} \varepsilon_{ij} \cdot \sigma_{ij}$ die doppelte Kontraktion der beiden Tensoren (siehe auch [36]) beschreibt. Weiterhin kennzeichnet V das Volumen, A den Flächenrand des betrachteten Körpers sowie k die Zugkraft (Traktion) und u das Verschiebungsfeld (ebenfalls abhängig der Zeit und eines räumlichen Punktes). Gleichung (4.2) stellt sicher, dass das Kräftegleichgewicht lokal an jedem Punkt innerhalb der Probe eingehalten wird (siehe Impulsbilanz in (2.6)). Gleichung (4.3) bezeichnet die Spannungsbedingungen am Rand des Körpers, wobei mit $n \in \mathbb{R}^3$ der Normalenvektor bezeichnet wird. Während die Bedingung (4.1) für den gesamten Körper erfüllt werden soll, beschreibt (4.2) eine Bedingung für jeden einzelnen Punkt des Körpers. Die letzte Bedingung (4.3) wird insbesondere auch für die Teilmenge der Punkte am Rand der Probe gefordert, an dem keine Kräfte wirken (für k =0).

Durch die räumliche Diskretisierung des Verschiebungsfeldes, wie sie im Versuch vorliegt, wird das Volumenintegral in (4.1) durch eine Summe über die endliche Menge aller Punkte $P \subset V$, an denen die Verschiebung gemessen wird mit den entsprechenden Werten ε^p, σ^p und V^p für $p \in P$ approximiert. Außerdem ist durch eine feste Einspannung der Probe die Verschiebung du über den gesamten Querschnitt konstant. Die Richtung der gemessenen Kraft F entspricht zudem der Richtung von du (es werden keine Querkräfte gemessen). Daher vereinfacht sich das Flächenintegral über den Querschnitt A in (4.1) zu $F \cdot du$, was zu der folgenden, diskretisierten Umformulierung von (4.1) führt:

$$\sum_{p=1}^{P} \left(\sigma^p : \varepsilon^p\right) V^p = \sum_{p=1}^{P} \left(\sum_{i=1}^{6} \sigma_i^p \cdot \varepsilon_i^p \right) V^p = F \cdot du. \tag{4.4}$$

Da die Gleichung (4.4) für die jeweiligen, gemessenen Zeitpunkte $t \in T$ erfüllt sein sollte, wird der Gesamtfehler L_{force} für das Kräftegleichgewicht basierend auf (4.1) durch

$$L_{force}(\sigma) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{F^t}\, d\left(\frac{\sum_{p=1}^{P} \left(\sum_{i=1}^{6} \sigma_i^{p,t} \cdot \varepsilon_i^{p,t} \right) V^{p,t}}{du^t}, F^t \right) \tag{4.5}$$

bestimmt, mit einer festzulegenden Metrik $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$. Mit $\varepsilon^{p,t}$ wird das gemessene Dehnungsinkrement am Punkt p zum Zeitpunkt t mit dem zugehörigen Volumen $V^{p,t}$ bezeichnet. Die Spannungen $\sigma^{p,t}$ werden vom ML-Materialmodell basierend auf $\varepsilon^{p,t}$

vorhergesagt. Weiterhin bezeichnet du^t das globale Verschiebungsinkrement am Kraftrand und F^t die global gemessene Kraft zum Zeitpunkt t.
Der zweite Teil (4.2) der Fehlerfunktion im Inneren des Probekörpers ergibt sich durch die Impulsbilanz (siehe Kapitel 2.1.4)

$$div(\sigma) + \cancel{\rho b} = \cancel{\rho \ddot{u}}, \tag{4.6}$$

wobei Dichte ρ, Beschleunigung $\ddot{u}$ und Volumenkräfte b nicht berücksichtigt werden, da diese Terme additiv sowie unabhängig der Spannungsprognose und daher unabhängig des Materialmodells sind. Für die approximative Berechnung der Divergenz wird die Probe mit einem Überlagerungsgitter mit gleichmäßig verteilten Punkten versehen. Die Messpunkte p werden auf die Punkte des Überlagerungsgitters $\hat{p}$ für jeden erfassten Zeitpunkt t interpoliert, wie in Abbildung 4.2 gezeigt.

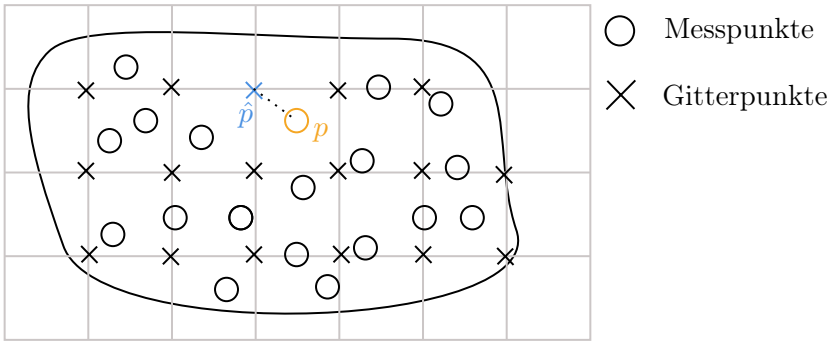


Abbildung 4.2: Beispiel einer Anordnung von Mess- und Gitterpunkten auf einer Versuchsprobe

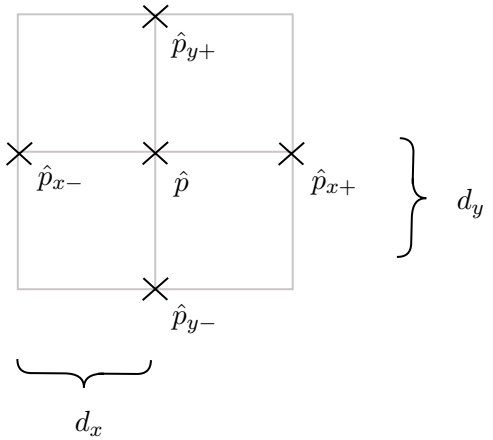


Abbildung 4.3: Notation der Nachbarpunkte eines Punktes $\hat{p}$ auf dem Überlagerungsgitter

Ausgehend von einem zweidimensionalen Divergenzfeld bekommt man die zwei Fehlerfunktionen $L_{div,x}$ und $L_{div,y}$ für die entsprechenden Divergenz-Richtungen

$$L_{div,x}(\sigma) = \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} d\left(0, \frac{\partial \sigma_{xx}^{\hat{p},t}}{\partial x} + \frac{\partial \sigma_{xy}^{\hat{p},t}}{\partial y}\right) \tag{4.7}$$

$$L_{div,y}(\sigma) = \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} d\left(0, \frac{\partial \sigma_{yx}^{\hat{p},t}}{\partial x} + \frac{\partial \sigma_{yy}^{\hat{p},t}}{\partial y}\right), \tag{4.8}$$

wobei $\hat{P}$ die Anzahl an Punkten auf dem Gitter bezeichnet. Die Gradienten werden durch die zentrale-Differenzen-Methode in den jeweiligen Richtungen approximiert

$$\frac{\partial \sigma_{ij}^{\hat{p}}}{\partial x} = \frac{\sigma_{ij}^{\hat{p}_{x+}} - \sigma_{ij}^{\hat{p}_{x-}}}{2d_x} \quad \text{und} \quad \frac{\partial \sigma_{ij}^{\hat{p}}}{\partial y} = \frac{\sigma_{ij}^{\hat{p}_{y+}} - \sigma_{ij}^{\hat{p}_{y-}}}{2d_y}, \tag{4.9}$$

wobei mit $\hat{p}_{x-}, \hat{p}_{x+}, \hat{p}_{y-}, \hat{p}_{y+}$, wie in Abbildung 4.3 dargestellt, die Nachbarpunkte eines Punktes $\hat{p}$ in $x-$ beziehungsweise y-Richtung im Gitter bezeichnet werden. Die Fehlerfunktionen $L_{div,x}$ und $L_{div,y}$ werden anhand der gemessenen Kraft F^t zum Zeitpunkt t und dem Anfangsvolumen V_0 normiert, sodass sie unabhängig der Spannungs- oder Kraftgröße sind. Für eine weitere Fehlerfunktion wird der kraftfreie Rand der Probe und die dadurch hergeleitete zug- beziehungsweise kraftfreie Randbedingung

$$\sigma \cdot n = 0$$

betrachtet, wobei durch $n \in \mathbb{R}^3$ der nach außen gerichteten Einheitsnormalenvektor auf dem kraftfreien Rand bezeichnet wird. Unter der Annahme einer flachen Probe lassen sich zwei unterschiedliche Bedingungen ableiten, die zu den folgenden Fehlerfunktionen

$$L_z(\sigma) = \frac{1}{T \cdot P} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p=1}^{P} \left(d\left(0, \sigma_{xz}^{p,t}\right) + d\left(0, \sigma_{yz}^{p,t}\right) + d\left(0, \sigma_{zz}^{p,t}\right)\right) \tag{4.10}$$

und

$$\begin{aligned} L_{bnd}(\sigma) = & \frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} \left(d\left(0, \sigma_{xx}^{p_b,t} n_x^{p_b,t} + \sigma_{yx}^{p_b,t} n_y^{p_b,t}\right)\right. \\ & \left. + d\left(0, \sigma_{xy}^{p_b,t} n_x^{p_b,t} + \sigma_{yy}^{p_b,t} n_y^{p_b,t}\right) + d\left(0, \sigma_{xz}^{p_b,t} n_x^{p_b,t} + \sigma_{yz}^{p_b,t} n_y^{p_b,t}\right)\right) \end{aligned} \tag{4.11}$$

führen. Der Fehler L_z ergibt sich aus der Forderung eines ebenen Spannungszustands durch

die Abwesenheit von Kräften in z-Richtung basierend auf (4.3) mit dem Normalenvektor $n = (0, 0, 1)^T$. Diese Bedingung soll für alle Punkte $p \in P$ erfüllt sein. L_{bnd} berücksichtigt den zweidimensionalen kraftfreien Rand und muss an jedem kraftfreien Messpunkt p_b eingehalten werden, wobei mit P_b die Anzahl aller Messpunkte an den kraftfreien Rändern bezeichnet wird. n_x und n_y bezeichnen die individuell berechneten Komponenten des nach außen gerichteten Einheitsnormalenvektors bezogen auf den betrachteten Messpunkt. Da davon auszugehen ist, dass die Probe flach in der $x-y-$Ebene liegt, ist n_z immer null. Beide Fehler werden mit der gemessenen Kraft F^t und der Ausgangsfläche A_0 normiert.

4.2.2 Fehlerfunktion für die 2D-Formulierung von Schalenelementen

Im Fall von Schalenelementen ohne Berücksichtigung der $\varepsilon_3 = \varepsilon_{zz}, \varepsilon_5 = \varepsilon_{yz}, \varepsilon_6 = \varepsilon_{zx}$ sowie $\sigma_3 = \sigma_{zz}, \sigma_5 = \sigma_{yz}, \sigma_6 = \sigma_{zx}$ Komponenten beziehungsweise einer separaten Berechnung dieser vereinfachen sich die einzelnen Fehlerfunktionen wie folgt:

- Die Summe über alle Komponenten $\sum_{i=1}^{6} \sigma_i^p \cdot d\varepsilon_i^p$ in der Fehlerfunktion L_{force} für die gesamte Kraft kann wegen $\varepsilon_3 = \sigma_3 = 0$ vereinfacht werden zu

$$\sum_{\substack{i=1 \\ i \neq 3}}^{6} \sigma_i^p \cdot \varepsilon_i^p .$$

  Die Komponenten ε_i, σ_i für $i = 5, 6$ müssen allerdings hier mit berücksichtigt werden, da diese im Allgemeinen ungleich null sind.

- Die Fehlerfunktionen für die Divergenz $L_{div,x}$ und $L_{div,y}$ ändern sich nicht, da auch für den allgemeinen Fall ein (Zug-)versuch einer flachen Probe betrachtet und daher von einem zweidimensionalem Divergenzfeld ausgegangen wird.

- Die Fehlerfunktion L_z ist als Gesamtes konstant bezüglich der trainierbaren Parameter des ML-Materialmodells und muss daher nicht berücksichtigt werden.

- Bei L_{bnd} ist der letzte additive Term $d\left(0, \sigma_{xz}^{p_b,t} n_x^{p_b,t} + \sigma_{yz}^{p_b,t} n_y^{p_b,t}\right)$ unabhängig von den trainierbaren Parametern des ML-Materialmodells und die Fehlerfunktion vereinfacht sich daher zu

$$\begin{aligned} L_{bnd} = & \frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} \left(d\left(0, \sigma_{xx}^{p_b,t} n_x^{p_b,t} + \sigma_{yx}^{p_b,t} n_y^{p_b,t}\right) \right. \\ & \left. + d\left(0, \sigma_{xy}^{p_b,t} n_x^{p_b,t} + \sigma_{yy}^{p_b,t} n_y^{p_b,t}\right) \right) . \end{aligned} \tag{4.12}$$

4.3 Optimierungsstrategien für das Training von neuronalen Netzen mit den Fehlerfunktionen basierend auf Versuchsdaten

Im Folgenden liegt der Fokus für die Trainingsalgorithmen der ML-Materialmodelle mit Daten aus Versuchen auf neuronalen Netzen und kann für unterschiedliche Netzarten und Netzstrukturen (Feed Forward neuronale Netze wie auch rekurrente neuronale Netze) angewendet werden. Darüber hinaus ist eine Anwendung der vorgestellten Trainingsmethoden direkt für jede Art von ML-Modell möglich, welches durch eine fest vorgegebene Anzahl an trainierbaren, reellwertigen Parametern eindeutig bestimmt ist. Dazu gehören unter anderem auch Transformer Modelle und Regressionsmodelle, wie beispielsweise lineare oder Spline Regression, nicht jedoch baumbasierte Verfahren wie Random Forests.

4.3.1 Training der ML-Modelle als mehrkriterielles Optimierungsproblem vs. Skalarisierung

Die verschiedenen vorgestellten Gleichungen, welche für die vorhergesagten Spannungen $\sigma(x)$ eines trainierten ML-Materialmodells mit n gegebenen trainierbaren Parametern $x \in \mathbb{R}^n$ gelten sollen, können durch eine Abweichung der Gleichheit wie in Abschnitt 4.2 beschrieben als Fehlerfunktionen $L_{force}, L_{div,x}, L_{div,y}, L_z$ und L_{bnd} für das Training betrachtet werden. Da mehrere unterschiedliche Zielfunktionen minimiert werden sollen, kann das Training basierend auf Versuchsdaten als multikriterielles oder auch mehrkriterielles Optimierungsproblem der Form

$$\begin{aligned} \min \quad & f(x) = (L_{force}(\sigma(x)), L_{div,x}(\sigma(x)), L_{div,y}(\sigma(x)), L_z(\sigma(x)), L_{bnd}(\sigma(x))) \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned} \tag{4.13}$$

mit $f : \mathbb{R}^n \to \mathbb{R}^5$ beschrieben werden. Während bei einem einkriteriellen Optimierungsproblem eindeutig definiert ist, welche Lösung des Problems durch einen größeren beziehungsweise kleineren Zielfunktionswert (je nachdem ob minimiert oder maximiert wird) besser oder schlechter ist, werden in der multikriteriellen Optimierung Pareto-optimale Lösungen betrachtet:

Definition 4.3.1. Eine zulässige Lösung $\hat{x} \in X$ für ein multikriterielles Optimierungsproblem der Form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X, \end{aligned} \tag{4.14}$$

mit einer Zielfunktion $f : X \subset \mathbb{R}^n \to \mathbb{R}^k$ heißt

- *Pareto-optimal* oder *effizient*, falls keine weitere zulässige Lösung $x \in X \setminus \{\hat{x}\}$ mit $f_i(x) \leq f_i(\hat{x})$ für $i = 1, ..., k$ und $f(x) \neq f(\hat{x})$ existiert. Man sagt in diesem Fall auch, dass x die Lösung $\hat{x}$ *dominiert*.
- *schwach effizient*, falls keine Lösung $x \in X$ mit $f_i(x) < f_i(\hat{x})$ für $i = 1, ..., k$ existiert.

Aus Definition 4.3.1 geht insbesondere hervor, dass jede schwach effiziente Lösung auch Pareto-optimal (effizient) ist. Multikriterielle Optimierungsprobleme können durch unterschiedliche Skalarisierungsmethoden in ein einkriterielles Optimierungsproblem umgewandelt werden. Eine häufig genutzte Skalarisierungsmethode ist die Skalarisierung als gewichtete Summe

$$\begin{aligned} \min \quad & L_{total}(x) \coloneqq w_f L_{force}(\sigma(x)) + w_{d,x} L_{div,x}(\sigma(x)) + w_{d,y} L_{div,y}(\sigma(x)) \\ & \qquad + w_{b,z} L_z(\sigma(x)) + w_b L_{bnd}(\sigma(x)) \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned} \tag{4.15}$$

mit vordefinierten Gewichten $w_f, w_{d,x}, w_{d,y}, w_{b,z}, w_b \in \mathbb{R}$. Werden die Gewichte der Skalarisierung nicht negativ und ungleich null gewählt, so ist eine optimale Lösung der entsprechenden Skalarisierung schwach effizient (siehe [24]), auch wenn im Allgemeinen Fall umgekehrt nicht jede schwach effiziente Lösung des multikriteriellen Optimierungsproblems durch die Lösung der Skalarisierung mit einer entsprechenden Gewichtung gefunden werden kann. Ein Nachteil der einkriteriellen Variante ist daher einerseits, dass möglicherweise eine effiziente Lösung auch mit einer passenden Gewichtsauswahl nicht gefunden wird und dass weiterhin die gefundene Lösung stark von einer passenden Wahl der Gewichtung abhängig sein kann.
Gilt bei einem mehrkriteriellen Optimierungsproblem wie in (4.14) für zwei Komponenten $f_i : \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x)_i$ und $f_j : \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x)_j$ der Zielfunktion die folgende Bedingung

$$f_i(x^1) \leq f_i(x^2) \Rightarrow f_j(x^1) \leq f_j(x^2), \forall x^1, x^2 \in X, \tag{4.16}$$

so können die beiden Zielfunktionskomponenten zusammengefasst werden und müssen nicht separat in der Optimierung berücksichtigt werden. Dass dies für das Training von ML-Materialmodellen mit Versuchsdaten für keine Kombination der verschiedenen Fehlerfunktionen der Fall ist, zeigt das folgende Gegenbeispiel. In Tabelle 4.1 ist zu sehen, dass es im Allgemeinen nicht genügt nur einen Teil der verschiedenen Fehlerfunktionen zu betrachten, da sich diese paarweise für jede Kombination in einer Optimierung gegenläufig verhalten können. Zu sehen sind die einzelnen Fehlerfunktionswerte für vier

Beispielmodelle $ML_1^{36}, ..., ML_4^{36}$ für unterschiedliche trainierbare Parameter für das ML^{36} neuronale Netz aus Abschnitt 3.1 ohne zwischenliegende Schichten für ein linear elastisches Materialmodell bezogen auf Daten aus einem simulierten Patch-Zugversuch, wie er näher in Abschnitt 4.4.1 betrachtet wird für ein E-Modul von 210 und eine Querkontraktionszahl von $0,3$. Die Modelle ML_1^{36} und ML_2^{36} zeigen, dass die Zielfunktionspaare $(L_{force}, L_{div,x}), (L_{force}, L_{bnd}), (L_{div,x}, L_{div,y}), (L_{div,x}, L_z), (L_{div,y}, L_{bnd})$ sowie (L_{bnd}, L_z) die Bedingung (4.16) nicht erfüllen. Betrachtet man ML_2^{36} und ML_3^{36}, so ist zu sehen, dass auch die Funktionspaare $(L_{force}, L_{div,y}), (L_{force}, L_z)$ und $(L_{bnd}, L_{div,x})$ gegenläufig sein können. Dass zudem für $L_{div,y}$ und L_z (4.16) nicht gilt, zeigen die Fehlerwerte für ML_3^{36} und ML_4^{36}.

	ML_1^{36}		ML_2^{36}		ML_3^{36}		ML_4^{36}
L_{force}	0.260	$>$	0.119	$<$	0.184	$<$	0.195
$L_{div,x}$	$1.271 \cdot 10^{-4}$	$<$	$1.336 \cdot 10^{-4}$	$<$	$1.387 \cdot 10^{-4}$	$>$	$1.365 \cdot 10^{-4}$
$L_{div,y}$	$2.955 \cdot 10^{-5}$	$>$	$2.835 \cdot 10^{-5}$	$>$	$2.793 \cdot 10^{-5}$	$>$	$2.762 \cdot 10^{-5}$
L_z	0.413	$>$	0.119	$>$	$7.692 \cdot 10^{-2}$	$<$	0.117
L_{bnd}	$9.058 \cdot 10^{-2}$	$<$	1.078	$>$	$9.416 \cdot 10^{-2}$	$>$	$8.608 \cdot 10^{-2}$

Tabelle 4.1: Beispiel bezüglich der Gegenläufigkeit der verschiedenen Fehlerfunktionen

Für die Anwendung der Methode für das Training der ML-Materialmodelle mit Versuchsdaten werden in den Abschnitten 4.4 und 4.5 sowohl der mehrkriterielle Optimierungsansatz als auch das Training durch die Skalarisierung als gewichtete Summe betrachtet.

4.3.2 Training der ML-Materialmodelle durch evolutionäre Algorithmen ohne Gradientenberechnung

Üblicherweise wird für das Training von neuronalen Netzen ein gradientenbasiertes Optimierungsverfahren mit der Berechnung der Gradienten durch Backpropagation, wie in Kapitel 2.3 beschrieben, genutzt. Auch wenn Backpropagation mit einer angepassten Berechnung der partiellen Ableitungen für die Gewichte in der letzten Schicht des neuronalen Netzes (wie es später in Abschnitt 4.3.3 beschrieben wird) möglich ist, werden hier zusätzlich alternative, gradientenfreie Optimierungsalgorithmen für das Training betrachtet. Die betrachteten Algorithmen gehen von einem Optimierungsproblem ohne

Nebenbedingungen der Form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathbb{R}^n, \end{aligned} \tag{4.17}$$

mit n reellwertigen Variablen aus und benutzten im Laufe der Optimierung die Zielfunktion f ausschließlich über die Funktionsauswertungen $f(x)$ für gegebene Lösungen $x \in \mathbb{R}^n$. Dieser Optimierungsansatz hat gegenüber der klassischen Trainingsmethode den Vorteil, dass die Verfahren bei Veränderungen und Ergänzungen der Fehlerfunktionen direkt ohne Anpassung des Optimierungsverfahren angewendet werden können. Weiterhin können auch nicht differenzierbare Fehlerfunktionen oder Ergebnisse einer FE-Simulation in das Training der ML-Materialmodelle integriert werden. Für die Anwendung als Trainingsalgorithmus eines ML-Modells entspricht eine Lösung $x \in \mathbb{R}^n$ einem Vektor, welcher die trainierbaren Parameter des ML-Modells enthält und der Zielfunktionswert beziehungsweise die Zielfunktionswerte $f(x)$ den Fehlerwerten, die durch die vorhergesagten Ausgabewerte des ML-Modells (Spannungswerte des ML-Materialmodells) mit diesen gegebenen Parameter berechnet werden.

In dieser Arbeit werden verschiedene Optimierungsalgorithmen aus dem Bereich der evolutionären Algorithmen für das Training der ML-Materialmodelle betrachtet. *Evolutionäre Algorithmen* (siehe [61] [65]) sind iterative, randomisierte Optimierungsverfahren, die auf den Mechanismen der biologischen Evolution basieren und können für Optimierungsprobleme wie in (4.17) angewendet werden. In jeder Iteration werden neue Lösungen durch stochastische Variationen vorhandener Lösungen generiert. Die verschiedenen Lösungskandidaten werden evaluiert, indem der Zielfunktionswert (in diesem Zusammenhang oft auch als *Fitnesswert* der Lösung bezeichnet) berechnet wird. Aus der Menge an betrachteten Lösungen (auch als *Population* bezeichnet) werden basierend auf den Fitnesswerten Kandidaten ausgewählt (selektiert) und in die nächste Iteration übernommen. Abbildung 4.4 zeigt den Vorgang bei einer Anwendung dieser Art von Algorithmen für das Training der ML-Materialmodelle. Die ML-Modelle werden hierbei extern optimiert, was bedeutet, dass die trainierbaren Gewichte der neuronalen Netze innerhalb des Optimierungsprozesses als Vektoren in $\mathbb{R}^n$ extrahiert werden, um neue Lösungen zu generieren. Für die Evaluierung der ML-Modelle werden die neuronalen Netze mit den gegebenen Gewichten aufgebaut, um die Modelle für eine Spannungsvorhersage, mit welcher wiederum die Fehlerwerte (Fitnesswerte) bestimmt werden, zu benutzen. Für das Training der neuronalen Netze mittels der Skalarisierung in (4.15) wird der *CMA* (*Covariance Matrix Adaptation*) Algorithmus aus [31] genutzt. Der Optimierungsalgorithmus ist eine evolutionäre Strategie (ES), bei der neue Lösungen auf Basis einer multivariaten Normalverteilung generiert werden. Der

CMA Algorithmus stellt eine Methode dar, die Kovarianzmatrix der Verteilung basierend auf der Population zu schätzen und zu aktualisieren.

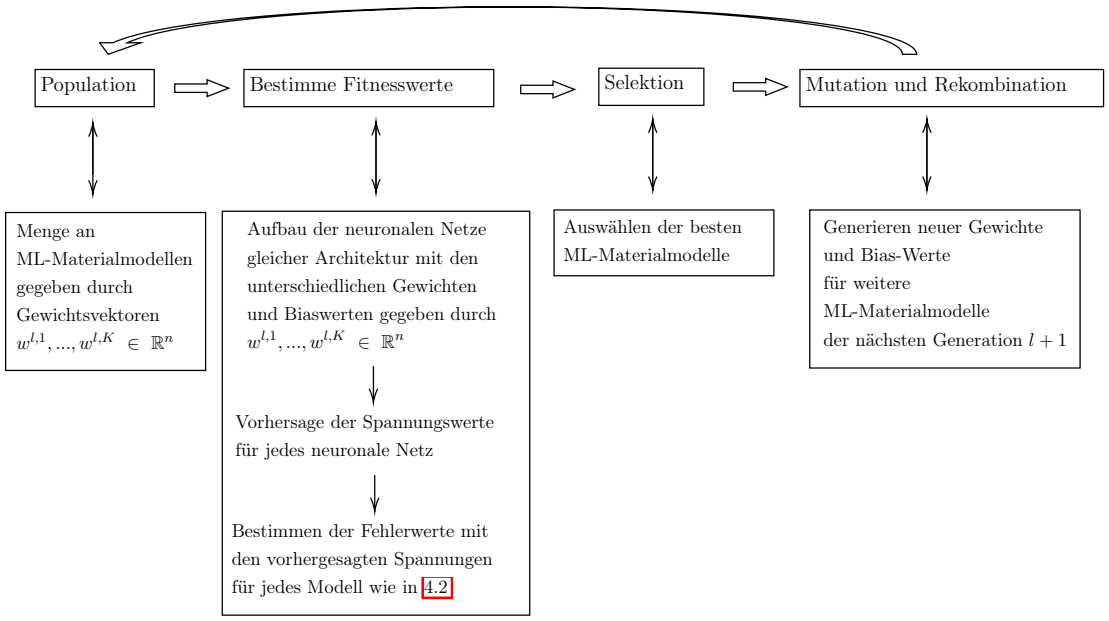


Abbildung 4.4: Schematischer Ablauf einer Iteration l des Trainings von ML-Materialmodellen mittels evolutionärer Algorithmen

Für den mehrkriteriellen Optimierungsfall in (4.13) werden die beiden elitären genetischen Algorithmen *NSGA-II* (*Non-dominated Sorting Genetic Algorithm*) aus [21] und *NSGA-III*, vorgestellt in [22] und aufbauend auf einer Implementierung aus [7], betrachtet. Beim NSGA-II Verfahren werden in einer Iteration auf Basis einer aktuellen Population der Größe k durch Selektion (Binary tournament selection), Rekombination und Mutation eine Nachkommengeneration der gleichen Größe erstellt. Aus diesen $2k$ Lösungen bilden die besten k Lösungen bezüglich ihres Ranges der Pareto-Ordnung eine neue Elterngeneration für die nächste Iteration ($k + k$-Selektion). Bei gleichem Rang entscheidet gegebenenfalls zusätzlich die *Crowding-Distance*. Die Crowding-Distance dient dazu die Diversität der Lösungsmenge bezüglich der verschiedenen Zielfunktionen sicherzustellen als approximatives Maß für die Größe des größten Quaders um den Punkt im Lösungsraum, welcher keine weitere Lösung innerhalb der Population enthält. Der Ablauf des NSGA-III ist ähnlich zu dem des NSGA-II. Die Erhaltung der Diversität der Lösungsmenge unterscheidet sich jedoch, indem beim NSGA-III statt der Crowding-Distance eine Menge an sogenannten *Referenzpunkten* betrachtet wird. Zusätzlich zur Identifikation von nicht-dominierten Lösungen wird darauf geachtet Lösungen zu betrachten, welche den (entweder strukturiert

weit im Raum verteilten oder benutzerdefiniert verteilten) Referenzpunkten zugeordnet werden, um eine Diversität der Population zu gewährleisten.

4.3.3 Training mit angepasstem Backpropagation-Verfahren

Da die diskretisierte, approximative Berechnung der Fehlerfunktionen, wie in Abschnitt 4.2 beschrieben, jeweils aus Verkettungen differenzierbarer Funktionen bezüglich der Spannungsausgabe und damit bezüglich den Gewichten und Bias-Werten der neuronalen Netze besteht, kann das Training von ML-Materialmodellen trotz deutlich veränderter Fehlerfunktion mittels des Backpropagation-Verfahrens durchgeführt werden. Der Unterschied zwischen dem Training von neuronalen Netzen mit den üblichen Fehlerfunktionen, die auf der Differenz zwischen den korrekten und den vom ML-Modell vorhergesagten Ausgabewerten basieren, besteht in der Berechnung des Gradienten der Fehlerfunktion bezüglich den trainierbaren Parametern der letzten Schicht des neuronalen Netzes. Der Fehler kann zum einen nur für die gesamte, festgelegte Menge an Datenpunkten, die sich aus allen im Versuch gemessenen Dehnungen ergibt, bestimmt werden und nicht für einzelne Datenpunkte unabhängig. Zudem unterscheidet sich die Berechnung des Gradienten je nach Gewicht, abhängig davon zu welchem Ausgabeneuron ein Gewicht gehört, wie im Folgenden ersichtlich wird. Die angepassten partiellen Ableitungen eines Gewichts in der letzten Schicht eines neuronalen Netzes werden für die in Abschnitt 4.2 vorgestellten Fehlerfunktionen mit einer quadratischen Abweichung $d(x, y) = (x - y)^2$ bestimmt. Die partiellen Ableitungen bezüglich der Gewichte in den übrigen, vorherigen Schichten des Netzes ergeben sich aufbauend darauf wie gewohnt durch die Kettenregel.

Sei w_j^i das Gewicht der Verbindung zwischen dem j-ten Neuron in der vorletzten Schicht des Netzes und dem i-ten Ausgangsneuron, b_j der Bias-Wert, k die Anzahl an Neuronen in der vorletzten Schicht, und g die (differenzierbare) Aktivierungsfunktion in der letzten Schicht. Weiterhin wird mit $h_j^{p,t}$ der Wert, welcher für die Eingabe $\varepsilon^{p,t}$ am Neuron j in der vorletzten Schicht bestimmt wurde, bezeichnet (siehe auch Abbildung 4.5). Die Vorhersage eines Ausgabewerts $\sigma_{i*}^{p,t}$ lässt sich durch

$$\sigma_{i*}^{p,t} = g(\sum_{j=1}^{k} w_j^{i^*} \cdot h_j^{p,t} + b_j) =: \phi(w^{i^*}, b, h^{p,t}).$$

berechnen. Die partielle Ableitung der Fehlerfunktion L_{force} nach dem Gewicht w_{j^*},

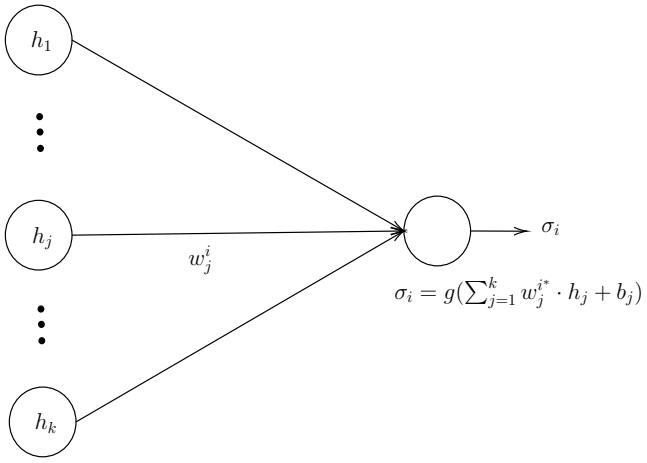


Abbildung 4.5: Schematische Darstellung eines Ausgabeneurons mit den verwendeten Notationen

welches zu einem Ausgabewert σ_{i*} führt, lässt sich damit wie folgt bestimmen:

$$\begin{aligned}
&\frac{\partial L_{force}}{\partial w_{j^*}^{i^*}} = \frac{\partial}{\partial w_{j^*}^{i^*}} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{F^t} \left(\frac{1}{du^t} \sum_{p=1}^{P} \left(\sum_{i=1}^{6} \sigma_i^{p,t} \cdot \varepsilon_i^{p,t} \right) V^{p,t} - F^t \right)^2 \\
=&\frac{1}{T} \sum_{t=1}^{T} \frac{1}{F^t} \frac{\partial}{\partial w_{j^*}^{i^*}} \left(\frac{1}{du^t} \sum_{p=1}^{P} \left(\sum_{i=1}^{6} \sigma_i^{p,t} \cdot \varepsilon_i^{p,t} \right) V^{p,t} - F^t \right)^2 \\
=&\frac{1}{T} \sum_{t=1}^{T} \frac{1}{F^t} \frac{\partial}{\partial w_{j^*}^{i^*}} \left(\frac{1}{du^t} \sum_{p=1}^{P} \left(\sum_{\substack{i=1 \\ i \neq i^*}}^{6} \sigma_i^{p,t} \cdot \varepsilon_i^{p,t} + \phi(w^{i^*}, b, h^{p,t}) \cdot \varepsilon_{i^*}^{p,t} \right) V^{p,t} - F^t \right)^2 \\
=&\frac{1}{T} \sum_{t=1}^{T} \frac{1}{F^t} \frac{2}{du^t} \left(\sum_{p=1}^{P} \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \varepsilon_{i^*}^{p,t} \cdot V^{p,t} \cdot \frac{1}{du^t} \right) \left(\sum_{p=1}^{P} \sum_{i=1}^{6} \sigma_i^{p,t} \cdot \varepsilon_i^{p,t} \cdot V^{p,t} - F^t \right)
\end{aligned}$$

mit

$$\begin{aligned}
&\frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \\
=&\frac{\partial}{\partial w_{j^*}^{i^*}} \; g(\sum_{\substack{j=1 \\ j \neq j^*}}^{k} w_j^{i^*} \cdot h_j^{p,t} + b_j + w_{j^*}^{i^*} \cdot h_{j^*}^{p,t} + b_{j^*}) \\
=&h_{j^*}^{p,t} \cdot g'(\sum_{j=1}^{k} w_j^{i^*} \cdot h_j^{p,t} + b_j).
\end{aligned}$$

Die partielle Ableitung der Fehlerfunktion $L_{div,x}$ ergibt sich durch

$$\frac{\partial L_{div,x}}{\partial w_{j^*}^{i^*}} = \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} \frac{\partial}{\partial w_{j^*}^{i^*}} \left(\underbrace{\frac{\sigma_1^{\hat{p}_{i+1},t} - \sigma_1^{\hat{p}_{i-1},t}}{2d_x} + \frac{\sigma_4^{\hat{p}_{i+1},t} - \sigma_4^{\hat{p}_{i-1},t}}{2d_y}}_{=:div_x(\hat{p})} \right)^2$$

$$= \begin{cases} \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} \frac{1}{2d_x} \left(\frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i+1},t})}{\partial w_{j^*}^{i^*}} - \frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i-1},t})}{\partial w_{j^*}^{i^*}} \right) \cdot div_x(\hat{p}) & \text{für } i^* = 1 \\ \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} \frac{1}{2d_y} \left(\frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i+1},t})}{\partial w_{j^*}^{i^*}} - \frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i-1},t})}{\partial w_{j^*}^{i^*}} \right) \cdot div_x(\hat{p}) & \text{für } i^* = 4 \\ 0 & \text{sonst} \end{cases}$$

und analog für $L_{div,y}$ durch

$$\frac{\partial L_{div,y}}{\partial w_{j^*}^{i^*}} = \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p} \in \hat{P}} \frac{\partial}{\partial w_{j^*}^{i^*}} \left(\underbrace{\frac{\sigma_4^{\hat{p}_{x+},t} - \sigma_4^{\hat{p}_{x-},t}}{2d_x} + \frac{\sigma_2^{\hat{p}_{y+},t} - \sigma_2^{\hat{p}_{y-},t}}{2d_y}}_{=:div_y(\hat{p})} \right)^2$$

$$= \begin{cases} \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} \frac{1}{2d_x} \left(\frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i+1},t})}{\partial w_{j^*}^{i^*}} - \frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i-1},t})}{\partial w_{j^*}^{i^*}} \right) \cdot div_y(\hat{p}) & \text{für } i^* = 4 \\ \frac{1}{T \cdot \hat{P}} \sum_{t=1}^{T} \frac{V_0}{F^t} \sum_{\hat{p}=1}^{\hat{P}} \frac{1}{2d_y} \left(\frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i+1},t})}{\partial w_{j^*}^{i^*}} - \frac{\partial \phi(w^{i^*}, b, h^{\hat{p}_{i-1},t})}{\partial w_{j^*}^{i^*}} \right) \cdot div_y(\hat{p}) & \text{für } i^* = 2 \\ 0 & \text{sonst.} \end{cases}$$

Für die Fehlerfunktion L_z kann die partielle Ableitung durch

$$\frac{\partial L_z}{\partial w_{j^*}^{i^*}} = \frac{\partial}{\partial w_{j^*}^{i^*}} \frac{1}{T \cdot P} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p=1}^{P} \left(\sigma_3^{p,t}\right)^2 + \left(\sigma_5^{p,t}\right)^2 + \left(\sigma_6^{p,t}\right)^2$$

$$= \begin{cases} \frac{1}{T \cdot P} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p=1}^{P} 2\sigma_{i^*} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} & \text{falls } i^* \in \{3, 5, 6\} \\ 0 & \text{sonst.} \end{cases}$$

berechnet werden. Die partielle Ableitung von L_{bnd} ergibt sich abhängig von dem jeweiligen

Ausgabeneuron i^* durch

$$
\begin{aligned}
&\frac{\partial L_{bnd}}{\partial w_{j^*}^{i^*}} \\
=&\frac{\partial}{\partial w_{j^*}^{i^*}} \frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} \Bigg(\left(\sigma_1^{p_b,t} n_1^{p_b,t} + \sigma_4^{p_b,t} n_2^{p_b,t} \right)^2 \\
&+ \left(\sigma_4^{p_b,t} n_1^{p_b,t} + \sigma_2^{p_b,t} n_2^{p_b,t} \right)^2 + \left(\sigma_6^{p_b,t} n_1^{p_b,t} + \sigma_5^{p_b,t} n_2^{p_b,t} \right)^2 \Bigg) \\
=&\begin{cases}
\frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} 2 n_1^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_1^{p_b,t} n_1^{p_b,t} + \sigma_4^{p_b,t} n_2^{p_b,t} \right) & \text{für } i^* = 1 \\
\frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} 2 n_2^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_4^{p_b,t} n_1^{p_b,t} + \sigma_2^{p_b,t} n_2^{p_b,t} \right) & \text{für } i^* = 2 \\
\frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} 2 n_2^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_6^{p_b,t} n_1^{p_b,t} + \sigma_5^{p_b,t} n_2^{p_b,t} \right) & \text{für } i^* = 5 \\
\frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} 2 n_1^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_6^{p_b,t} n_1^{p_b,t} + \sigma_5^{p_b,t} n_2^{p_b,t} \right) & \text{für } i^* = 6 \\
\frac{1}{T \cdot P_b} \sum_{t=1}^{T} \frac{A_0}{F^t} \sum_{p_b=1}^{P_b} 2 n_2^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_1^{p_b,t} n_1^{p_b,t} + \sigma_4^{p_b,t} n_2^{p_b,t} \right) & \\
+ 2 n_1^{p_b,t} \cdot \frac{\partial \phi(w^{i^*}, b, h^{p,t})}{\partial w_{j^*}^{i^*}} \cdot \left(\sigma_4^{p_b,t} n_1^{p_b,t} + \sigma_2^{p_b,t} n_2^{p_b,t} \right) & \text{für } i^* = 4 \\
0 & \text{für } i^* = 3.
\end{cases}
\end{aligned}
$$

Abschließend bekommt man ausgehend von einer Skalarisierung durch die gewichtete Summe wie in (4.15) den Gradienten des Gesamtfehlers durch

$$
\frac{\partial L_{total}}{\partial w_{j^*}^{i^*}} = w_f \frac{\partial L_{force}}{\partial w_{j^*}^{i^*}} + w_{d,x} \frac{\partial L_{div,x}}{\partial w_{j^*}^{i^*}} + w_{d,y} \frac{\partial L_{div,y}}{\partial w_{j^*}^{i^*}} + w_{b,z} \frac{\partial L_z}{\partial w_{j^*}^{i^*}} + w_b \frac{\partial L_{bnd}}{\partial w_{j^*}^{i^*}} \tag{4.18}
$$

für $w_{j^*}^{i^*}$ mit $i^* \in \{1, ..., 6\}$, $j^* \in \{1, ..., k\}$.

4.4 Validierung der Methode anhand des linear elastischen Materialmodells

Die vorgestellte Methode für das Training von ML-Materialmodellen basierend auf Daten aus Versuchen wird in diesem Abschnitt zunächst anhand eines linear elastischen Materialmodells wie in Kapitel 3.1 beschrieben mittels simulierter Versuche angewendet und validiert. Die Verwendung von simulierten Versuchen bedeutet, dass Simulationen der Versuche als FE-Modell aufgebaut und durchgeführt werden. Die genutzten Daten werden aus den Simulationsergebnissen extrahiert, wobei ausschließlich auf jene Daten (globale Kraft und Dehnungsfeld) zurückgegriffen wird, welche auch dem entsprechenden realen

Versuch entnommen werden können.
Das Ziel-Materialmodell, mit welchem die simulierten Zugversuche durchgeführt werden, ist jenes linear elastische Materialmodell, welches in Abschnitt 3.1 betrachtet wird mit einem Elastizitätsmodul von 210 sowie einer Querkontraktionszahl von $0,3$. Als Startmodell für die Optimierung dient ein anhand von Dehnungs-Spannungspaaren vortrainiertes ML-Materialmodell mit einem geringeren Elastizitätsmodul von 180 und einer Querkontraktionszahl von $0,2$. Dieses vortrainierte ML-Modell soll durch die unterschiedlichen in Abschnitt 4.3 vorgestellten Trainingsmethoden auf das Ziel-Materialmodell weitertrainiert und auf die korrekten Materialparameter angepasst werden. Für das linear elastische Materialmodell werden die beiden gleichen Netzstrukturen eines Feed Forward neuronalen Netzes wie in Abschnitt 3.1 betrachtet. Die vortrainierten ML-Materialmodelle, bestehend aus 36 beziehungsweise 384 trainierbaren Parametern, werden im Folgenden mit $ML^{linel,36}_{180-0,2}$ und $ML^{linel,384}_{180-0,2}$ bezeichnet. In Abschnitt 4.4.1 wird die Optimierung zunächst für das minimale, lineare ML-Modell anhand eines rechteckigen Ausschnitts (Patch) des Zugversuches beschrieben und anschließend auf das größere, nichtlineare Modell angewendet. In Abschnitt 4.4.2 wird der Ausschnitt des Zugversuchs erweitert, um einen etwas vielfältigeren Datensatz für das Training zu ermöglichen (siehe Abbildung 4.6).

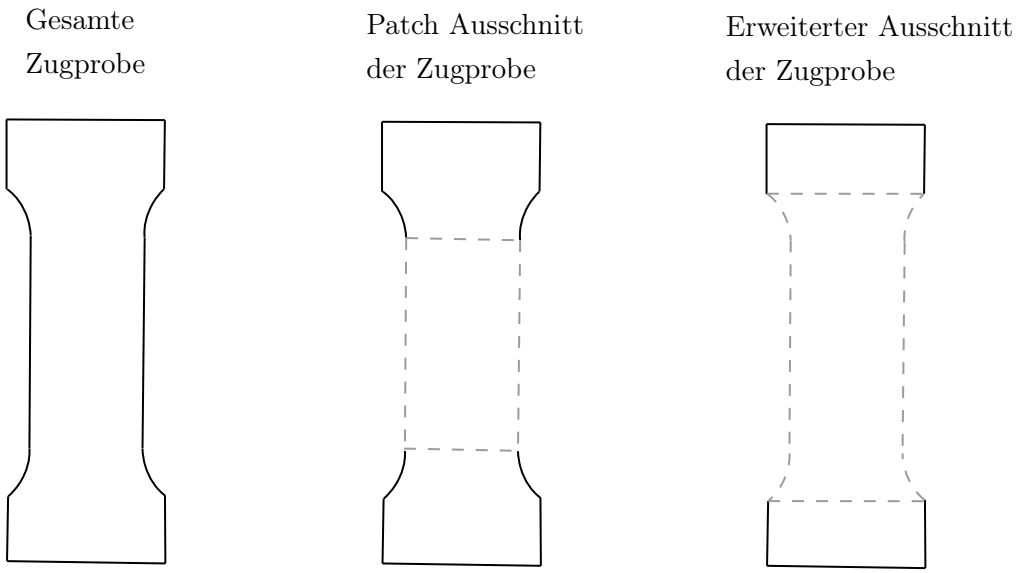


Abbildung 4.6: Versuchsproben für das Training mit Versuchsdaten, Links: Gesamte Zugprobe mit dem Rand für die Einspannung, Mittig: rechteckiger Patch Ausschnitt in der Mitte (grau gestrichelt), Rechts: Erweiterung des Ausschnittes (grau gestrichelt)

Um die Methode zu evaluieren werden die ML-Modelle, welche auf Basis des simulierten

Versuchs angepasst wurden, zu jenen verglichen, welche direkt auf zufällig generierten Daten in Form von Dehnungs-Spannungspaaren, wie in Kapitel 3.1 beschrieben, trainiert wurden. Diese direkt auf einem Elastizitätsmodul von 210 und einer Querkontraktionszahl von $0,3$ trainierten Zielmodelle werden mit $ML^{linel,36}_{210-0,3}$ und $ML^{linel,384}_{210-0,3}$ bezeichnet.
Um zu validieren inwiefern die alternativen Fehlerfunktionen für das Training mit Versuchsdaten funktionieren und zu evaluieren, wie gut das ML-Modell die Spannungen aus dem gesehenen Versuch lernt, wird die mittlere quadratische Abweichung zwischen den vom Modell vorhergesagten Spannungen aus dem Versuch und den korrekten, nicht im Training genutzten Spannungen, gegeben durch das klassische Materialmodell bestimmt

$$\mathrm{MSE}(V_\varepsilon, ML) := \frac{1}{6 \cdot |V_\varepsilon|} \sum_{i=1,\dots,6} \sum_{\varepsilon \in V_\varepsilon} (ML(\varepsilon)_i - Kl(\varepsilon)_i)^2, \tag{4.19}$$

wobei mit V_ε die Menge aller Dehnungen in dem entsprechenden Versuch bezeichnet wird und mit $ML(\varepsilon)$ der Vektor, welcher die vom ML-Materialmodell vorhergesagten Spannungen bei Eingabe der Dehnungen ε enthält sowie mit $Kl(\varepsilon)$ die korrekten Spannungen des klassischen Materialmodells. Für den Patch Ausschnitt wird die Menge der Dehnungen mit V_p und für den erweiterten Ausschnitt mit V_{mfz} bezeichnet. Die Abweichung in (4.19) zwischen den korrekten und den vom ML-Modell vorhergesagten Spannungen dient lediglich als Validierung der vorgestellten Methodik für das Training aus Versuchsdaten und zeigt inwiefern die vorgestellten Fehlerfunktionen als Alternative zu der direkten Abweichung zwischen vorhergesagten und korrekten Spannungen verwendet werden können. Für den Fall, dass ein ML-Materialmodell für einen neuartigen Werkstoff auf Basis von Versuchen trainiert werden soll, kann diese Metrik nicht herangezogen werden, da das klassische Materialmodell nicht vorhanden ist. Für die Validierung können hierfür stattdessen Vergleiche von Simulationsergebnissen mit dem trainierten ML-Modell und dem entsprechenden Versuch herangezogen werden.
Da der Trainingsdatensatz durch die Versuche limitiert ist und lediglich einfache Zugversuche betrachtet werden, wird abschließend in Abschnitt 4.4.3 eine Methode vorgestellt, um die Verallgemeinerungsfähigkeit der ML-Modelle anhand zusätzlich erzeugter Daten zu verbessern.

4.4.1 Training anhand des Patch Ausschnitts eines Zugversuches

Zunächst wird ein neuronales Netz trainiert anhand von Daten aus einer rechteckigen Zugprobe (Patch) unter Zugbelastung, wie man sie erhält, indem man die Außenkante einer Zugprobe, welche die Einspannung enthält, vollständig weglässt (siehe Abbildung

4.6). Abbildung 4.7 zeigt den Aufbau des simulierten Versuchs. Die Probengeometrie besteht aus 80 Schalenelementen, ist 100 mm lang, 20 mm breit und 1 mm dick. Die Knoten am linken Rand der Probe werden festgehalten, während für die Knoten am rechten Rand eine Verschiebung vorgegeben wird.

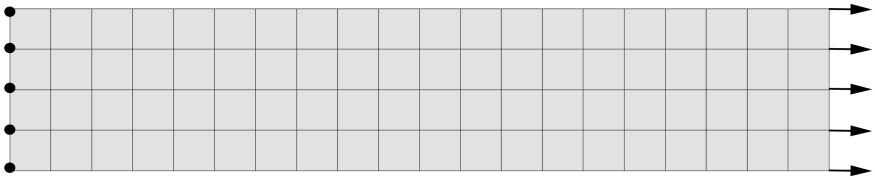


Abbildung 4.7: Aufbau Simulation Patch-Ausschnitt eines Zugversuchs

Als vortrainiertes Startmodell wird zunächst das Modell $ML^{linel,36}_{180-0,2}$ mit 36 trainierbaren Parametern betrachtet. Die Abbildung 4.8 zeigt den Verlauf des Gesamtfehlers $L_{total}(w)$ bei einer Optimierung der Fehlerfunktionen mittels der Skalarisierung wie in (4.15) mit einer gleichen Gewichtung $w = (1, 1, 1, 1, 1)$ aller Fehlerfunktionen im Laufe der Iterationen. Die Anpassung der trainierbaren Gewichte des Netzes erfolgt hierbei durch ein Gradientenabstiegsverfahren mit angepasster Berechnung der Gradienten, wie in Abschnitt 4.3.3 beschrieben sowie der Nutzung des Adam Optimierers (siehe (2.43)).

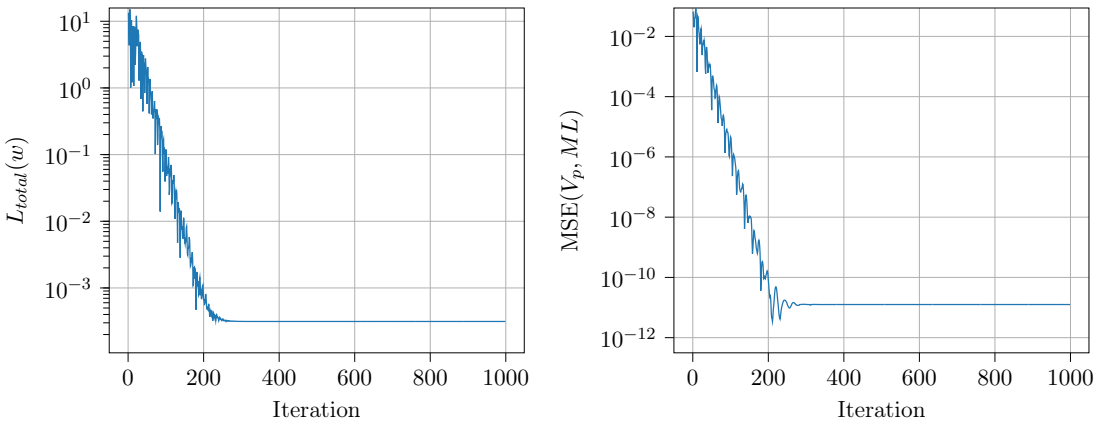


Abbildung 4.8: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ bei einer Gewichtung $w = (1, 1, 1, 1, 1)$ über die Iterationen bei einer Optimierung durch die Gradientenberechnung mit Lernrate $\eta = 1$ und dem Adam Optimierer, Rechts: Verlauf der mittleren quadratischen Abweichung $\mathrm{MSE}(V_p, ML)$ zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen des simulierten Versuchs

Das vortrainierte Startmodell $ML^{linel,36}_{180-0,2}$ beginnt mit einem Gesamtfehler von $L_{total}(w) =$

13.14 und erreicht bei einer konstanten Lernrate $\eta = 1$ einen Fehler von $L_{total}(w) = 3.142 \cdot 10^{-4}$ nach 1000 Iterationen. Der Gesamtfehler von $3.15 \cdot 10^{-4}$ des direkt auf Dehnungs-Spannungspaaren trainierten Zielmodells $ML^{linel,36}_{210-0,3}$ wird bereits nach 294 Iterationen übertroffen. Der ebenfalls abnehmende Verlauf der Abweichung MSE(V_p, ML) zwischen den vom optimierten ML-Modell (in der jeweiligen Iteration) vorhergesagten und den korrekten Spannungen des simulierten Versuchs für den Patch Ausschnitt in Abbildung 4.8 rechts zeigt, dass auch mithilfe der alternativen Fehlerfunktionen die im Versuch gesehenen Spannungen vom ML-Modell gut erlernt werden, ohne dass diese für das Training des Modells genutzt wurden.
Abbildung 4.9 zeigt den Fehlerverlauf des Trainings mittels Gradient und mit dem CMA-Algorithmus im direkten Vergleich für verschiedene Lernraten η für die Anwendung des Gradienten beziehungsweise unterschiedliche initiale Schrittgrößen σ_0 für die Standardabweichung der multivariaten Normalverteilung des CMA-Algorithmus.

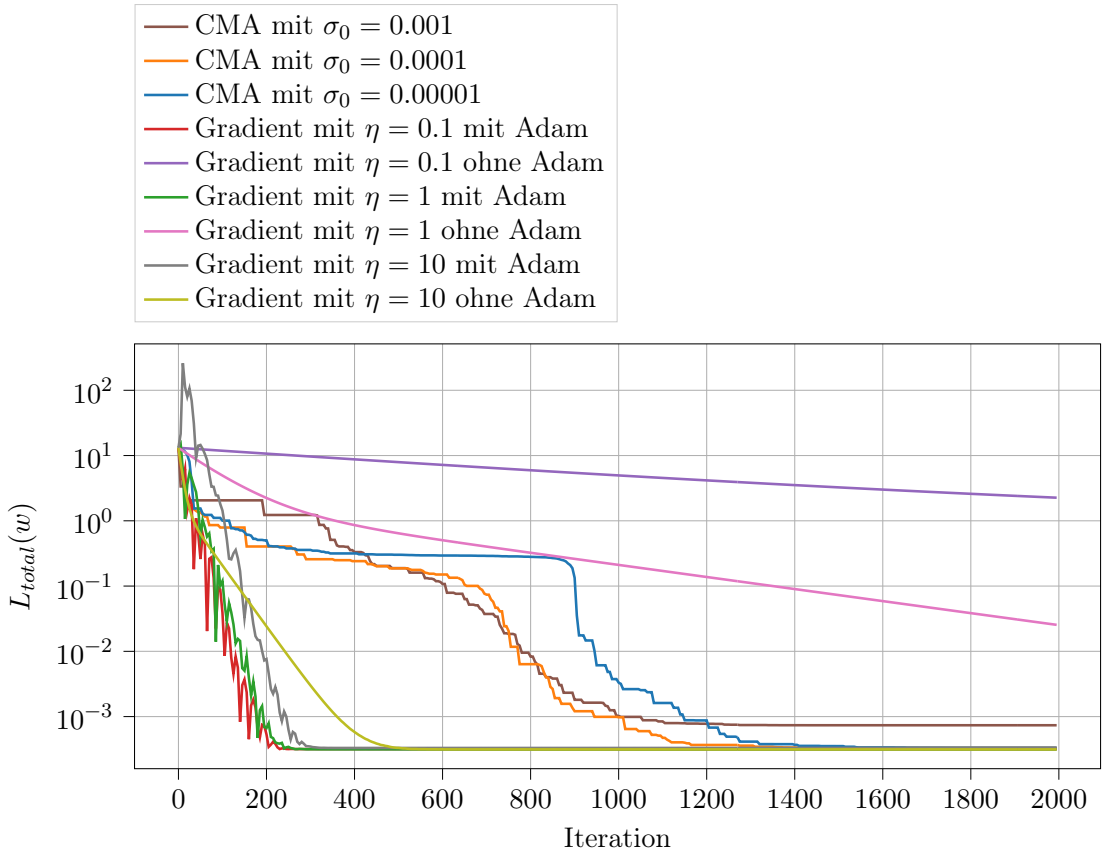


Abbildung 4.9: Vergleich der Optimierungsverläufe von $L_{total}(w)$ für die Gewichtung $w = (1, 1, 1, 1, 1)$ mittels verschiedener Optimierungsstrategien bei unterschiedlichen Lernraten η beziehungsweise Schrittgrößen σ_0

Bei der Anwendung des CMA-Algorithmus wurden je Konfiguration (je gewählter initialer Schrittgröße σ_0) 20 verschiedene Optimierungsläufe durchgeführt. Für jede Konfiguration wird jener Lauf auswählt und die entsprechenden Ergebnisse aufgezeigt, welcher vom Verlauf des Mittelwerts aller 20 Optimierungsläufe die kleinste mittlere quadratische Abweichung bezogen auf den Verlauf des erreichten Zielfunktionswertes über die Iterationen der Optimierung hat.

Die Varianz im Laufe der Iterationen der CMA-Optimierungsläufe sowie der Verlauf des Mittelwerts ist in Abbildung 4.10 zu sehen. Die Varianz der unterschiedlichen Optimierungsläufe nimmt bei allen gewählten initialen Schrittgrößen im Laufe der Iterationen deutlich ab und erreicht bezüglich des niedrigsten in der Optimierung erzielten Zielfunktionswerts je nach initialer Schrittgröße einen sehr geringen Wert zwischen $1.659 \cdot 10^{-10}$ und $1.599 \cdot 10^{-8}$ (siehe auch Tabelle 4.2). Für die niedrigeren initialen Schrittweiten von 0.0001 und 0.00001 ist die Varianz bezüglich des erzielten Zielfunktionswertes in der Optimierung geringer als bei der höher gewählten Schrittgröße von 0.001.

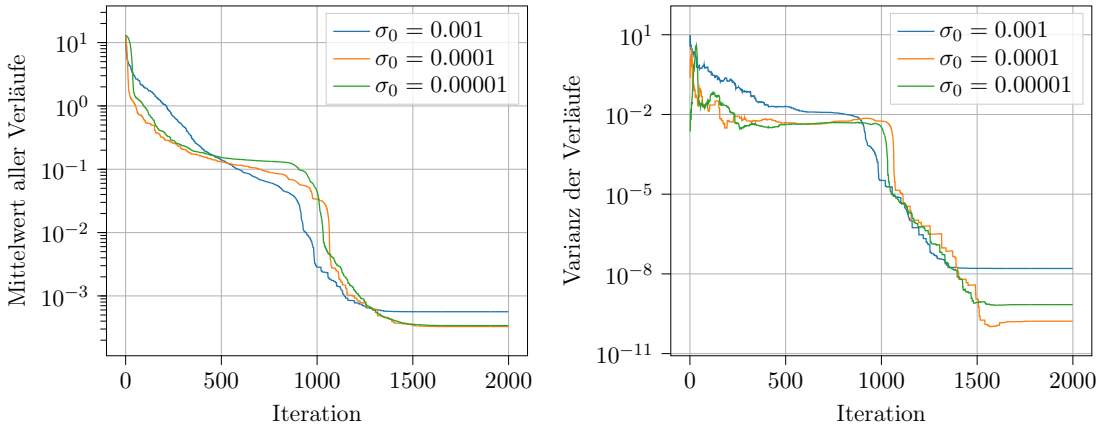


Abbildung 4.10: Links: Verlauf des Mittelwertes aller 20 CMA-Optimierungsläufe mit der jeweiligen initialen Schrittgröße σ_0 je Iteration, wobei je Iteration der kleinste, bisher erreichte Zielfunktionswert in der Optimierung betrachtet wird, Rechts: Varianz der 20 Optimierungsläufe im Laufe der Iterationen

Beim Training mittels des Gradienten wird je Iteration eine Berechnung der Gradienten mit den Spannungsvorhersagen des ML-Modells mit den aktuell betrachteten trainierbaren Parametern bestimmt. Beim CMA-Verfahren werden in je einer Iteration die Auswertungen der Fehlerfunktionen mit den vorhergesagten Spannungen von 14 verschiedenen ML-Modellen berechnet. Auch mit einer Optimierung durch CMA wird mit hinreichend kleiner initialer Schrittgröße (kleiner gleich 0.0001) ein Zielfunktionswert im Bereich des direkt trainierten ML-Zielmodells $ML_{210-0,3}^{linel,36}$ erreicht. Es werden im Vergleich zu

einer Optimierung durch den Gradienten (mit hinreichend großer Lernrate $\eta = 10$ oder durch Anwendung des Adam Optimierers) jedoch etwas mehr Iterationen benötigt. Die genauen Fehlerwerte (inklusive derer für die einzelnen Fehlerfunktionen) unterschiedlicher optimierter Modelle sowie des Ziel- und Startmodells finden sich in einer Übersicht in Tabelle 4.5.

Initiale Schrittgröße σ_0	Varianz gesamter Verlauf	Varianz der erreichten Zielfunktionswerte	Mittelwert der erreichten Zielfunktionswerte	Minimaler erreichter Zielfunktionswert	Maximaler erreichter Zielfunktionswert
0.001	$9.241 \cdot 10^{-1}$	$1.599 \cdot 10^{-8}$	$5.627 \cdot 10^{-4}$	$3.581 \cdot 10^{-4}$	$9.053 \cdot 10^{-4}$
0.0001	$5.743 \cdot 10^{-1}$	$1.659 \cdot 10^{-10}$	$3.277 \cdot 10^{-4}$	$3.081 \cdot 10^{-4}$	$3.433 \cdot 10^{-4}$
0.00001	$2.013 \cdot 10^{0}$	$6.967 \cdot 10^{-10}$	$3.395 \cdot 10^{-4}$	$3.043 \cdot 10^{-4}$	$4.138 \cdot 10^{-4}$

Tabelle 4.2: Übersicht über die Varianz der 20 unterschiedlichen CMA-Optimierungsverläufe für die jeweilige Wahl der initialen Schrittgröße σ_0 bezüglich des gesamten Verlaufs und bei der ausschließlichen Betrachtung des nach 2000 Iterationen erreichten minimalen Zielfunktionswertes sowie der Mittelwert, Maximal- und Minimalwert der erreichten minimalen Zielfunktionswerte aller 20 Optimierungsläufe. Die geringen Varianzen der erreichten Zielfunktionswerte sowie die nahe aneinanderliegenden Minimum- und Maximumwerte sprechen hierbei für eine hohe Robustheit des Optimierungsalgorithmus bezogen auf das vorliegende Trainingsproblem.

Die Abbildung 4.11 zeigt, dass eine alleinige Minimierung der einzelnen Fehlerfunktionen nicht ausreicht, um eine gute Spannungsprognose des ML-Materialmodells bezüglich der im simulierten Versuch betrachteten Daten zu erreichen. Zu sehen ist der Verlauf des $\mathrm{MSE}(V_p, ML)$ während der Optimierung mittels einer Anwendung des Gradienten (mit Lernrate $\eta = 1$) für ausschließlich die einzelnen Fehlerfunktionen $L_{div,x}, L_{div,y}, L_z, L_{bnd}$ und L_{force} sowie der Kombination aller Zielfunktionen $L_{total}(w)$ bei einer gleichen Gewichtung $w = (1, 1, 1, 1, 1)$, wobei sich ausschließlich bei Letzterem der Wert $\mathrm{MSE}(V_p, ML)$ signifikant verbessert.
Da die Lösung bei einer Skalarisierung durch die gewichtete Summe grundsätzlich abhängig von der Wahl der Gewichte sein kann, werden unterschiedliche Kombinationen für die Werte der verschiedenen Gewichte $w = (w_f, w_{d,x}, w_{d,y}, w_{b,z}, w_b)$ untersucht. Für die Auswertung und Vergleichbarkeit der verschiedenen, durch die Optimierung mit den unterschiedlichen Gewichtungen erhaltenen ML-Modelle, wird die entsprechende, gewichtete Fehlerfunktion $L^{\star}_{total}(w)$ des Modells $ML^{linel,36}_{210-0,3}$ als zu erreichender Zielfunktionswert betrachtet. Sobald für den erreichten Wert $L_{total}(w)$ die Bedingung $0.9 \cdot L_{total}(w) \leq L^{\star}_{total}(w)$ innerhalb der Optimierung erzielt wird, wird die Abweichung $\mathrm{MSE}(V_p, ML)$ als Bewertungskriterium für die entsprechende Gewichtung betrachtet. Die Endergebnisse der Optimierungen mit

den unterschiedlichen Gewichten sind in Tabelle 4.3 zu sehen.

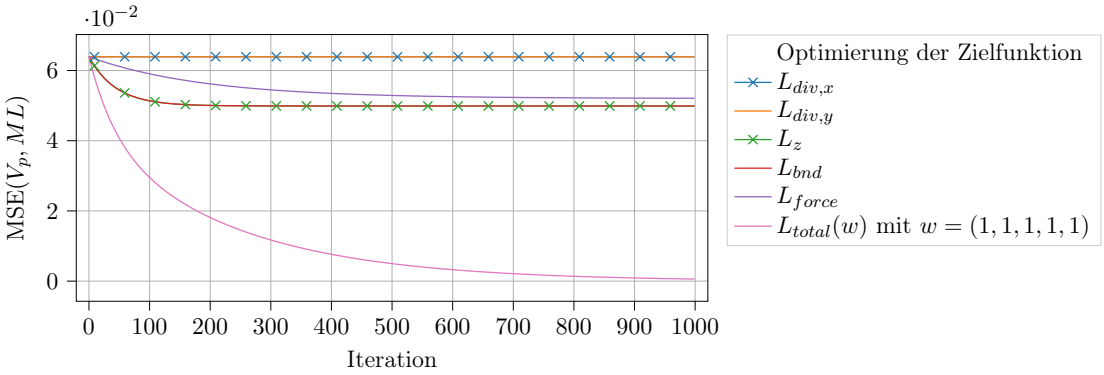


Abbildung 4.11: Verlauf des MSE(V_p, ML) bei einer Fehleroptimierung der einzelnen Fehlerfunktionen im Vergleich zu einer Kombination aller Fehlerfunktionen $L_{total}(w)$ als gewichtete Summe mit der Gewichtung $w = (1, 1, 1, 1, 1)$. Der MSE(V_p, ML)-Wert bezieht sich auf das beste bis zur jeweiligen Iteration in der Optimierung erhaltene ML-Modell bezüglich der in der Optimierung berücksichtigten Zielfunktion (Fehlerfunktion). Eine signifikante Verbesserung der Vorhersagefähigkeit des optimierten ML-Materialmodells für die im Versuch vorhandenen Daten ist nur bei der Kombination der unterschiedlichen Fehlerfunktionen und nicht durch die Optimierung der einzelnen Fehlerfunktionen zu sehen.

Tabelle 4.3 zeigt zudem ergänzend zu Abbildung 4.11, dass selbst falls nur einzelnen Fehlerfunktionen eine Gewichtung gleich null zugewiesen wird, die korrekten Spannungen aus dem Versuch nicht mehr erlernt werden. Die Abweichung MSE(V_p, ML) bleibt für diese Optimierungsläufe in einer ähnlichen Größenordnung wie beim Startmodell $ML^{linel,36}_{180-0,2}$ (siehe Tabelle 4.5). Eine Ausnahme bilden die Fehlerfunktionen $L_{div,x}$ und $L_{div,y}$ für die Divergenzen im Falle der hier betrachteten Patch-Simulation. Werden die Gewichte dieser beiden Fehlerfunktionen gleich null gesetzt, so hat dies kaum Auswirkungen auf die Abweichung MSE(V_p, ML) des optimierten ML-Modells. Erklären lässt sich dies für diesen Spezialfall, da durch die rechteckige Probengeometrie (ohne Löcher im Inneren der Probe) ein sehr homogenes Dehnungsfeld (kleine Unterschiede in den Dehnungswerten zwischen den räumlichen Punkten der Probe) vorliegt, was wiederum unabhängig der Qualität des Modells zu ähnlichen Werten in der Spannungsvorhersage in den Nachbarschaftselementen eines Punktes führt. Die Fehlerwerte für die Divergenzen sind daher nur marginal abhängig von den Parametern des ML-Materialmodells, somit grundsätzlich sehr klein und können für diesen Versuch in der Optimierung vernachlässigt werden. Werden einzelne Gewichtungen höher (zehnfach) gesetzt, so hat dies kaum Auswirkungen

auf den MSE(V_p, ML)-Wert, welcher auf einem ähnlich niedrigen Niveau wie bei einer Optimierung mit einer gleichen Gewichtung der Fehlerfunktionen bleibt.

Gewichtung $(w_f, w_{d,x}, w_{d,y}, w_{b,z}, w_b)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	$L_{total}(w)$	$L^{\star}_{total}(w)$	$\frac{L^{\star}_{total}(w)}{L_{total}(w)}$	MSE(V_p, ML)
(1,1,1,1,1)	$6.162 \cdot 10^{-5}$	$1.650 \cdot 10^{-4}$	$2.723 \cdot 10^{-5}$	$6.896 \cdot 10^{-5}$	$1.397 \cdot 10^{-5}$	$3.368 \cdot 10^{-4}$	$3.171 \cdot 10^{-4}$	0.941	$2.896 \cdot 10^{-10}$
(10,1,1,1,1)	$6.036 \cdot 10^{-5}$	$1.652 \cdot 10^{-4}$	$2.748 \cdot 10^{-5}$	$5.884 \cdot 10^{-5}$	$1.343 \cdot 10^{-5}$	$8.686 \cdot 10^{-4}$	$8.632 \cdot 10^{-4}$	0.993	$1.223 \cdot 10^{-11}$
(1,1,1,10,1)	$6.031 \cdot 10^{-5}$	$1.651 \cdot 10^{-4}$	$2.723 \cdot 10^{-5}$	$4.489 \cdot 10^{-5}$	$1.292 \cdot 10^{-5}$	$7.144 \cdot 10^{-4}$	$6.495 \cdot 10^{-4}$	0.909	$1.263 \cdot 10^{-11}$
(1,1,1,1,10)	$6.031 \cdot 10^{-5}$	$1.650 \cdot 10^{-4}$	$2.654 \cdot 10^{-5}$	$5.737 \cdot 10^{-5}$	$1.839 \cdot 10^{-5}$	$4.932 \cdot 10^{-4}$	$4.763 \cdot 10^{-4}$	0.965	$3.140 \cdot 10^{-11}$
(1,0,0,1,1)	$6.032 \cdot 10^{-5}$	$1.650 \cdot 10^{-4}$	$2.724 \cdot 10^{-5}$	$4.968 \cdot 10^{-5}$	$1.295 \cdot 10^{-5}$	$1.229 \cdot 10^{-4}$	$1.153 \cdot 10^{-4}$	0.938	$3.001 \cdot 10^{-11}$
(0,1,1,1,1)	0.190	$1.335 \cdot 10^{-4}$	$2.724 \cdot 10^{-5}$	$8.445 \cdot 10^{-5}$	$1.414 \cdot 10^{-5}$	$2.594 \cdot 10^{-4}$	$2.564 \cdot 10^{-4}$	0.988	$3.588 \cdot 10^{-2}$
(1,1,1,0,1)	$7.763 \cdot 10^{-5}$	$1.573 \cdot 10^{-4}$	$2.725 \cdot 10^{-5}$	15.56	$1.571 \cdot 10^{-5}$	$2.779 \cdot 10^{-4}$	$2.802 \cdot 10^{-4}$	1.008	$2.596 \cdot 10^{-2}$
(1,1,1,1,0)	$6.295 \cdot 10^{-5}$	$1.573 \cdot 10^{-4}$	$2.865 \cdot 10^{-5}$	$4.967 \cdot 10^{-5}$	1.405	$2.986 \cdot 10^{-4}$	$2.994 \cdot 10^{-4}$	1.002	$2.596 \cdot 10^{-2}$

Tabelle 4.3: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ und MSE(V_p, ML) der optimierten Modelle für unterschiedliche Gewichtungen der Skalarisierung

Als nächstes wird eine simultane Optimierung der verschiedenen Fehlerfunktionen (mittels des NSGA-III) ohne die vorherige Wahl einer Gewichtung vorgenommen. Mit der obigen Begründung werden dabei die Fehlerfunktionen $L_{div,x}$ und $L_{div,y}$ nicht berücksichtigt und somit nur die drei Zielfunktionen L_{force}, L_z und L_{bnd} betrachtet. Die Referenzpunkte für den NSGA-III werden nach dem Ansatz von Das und Dennis aus [20] gesetzt mit fünf Punkten in jeder Richtung der Zielfunktionsachse mit jeweils gleichem Abstand. Abbildung 4.12 und Tabelle 4.4 zeigen die verschiedenen Fehlerfunktionswerte für die nicht-dominierten Lösungen der finalen Population der Optimierung nach 500 Iterationen.

	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	L_{total} mit $w = (1,1,1,1,1)$	MSE(V_p, ML)
$ML^{patch,36}_{opt,multi,best}$	$6.022 \cdot 10^{-5}$	$1.640 \cdot 10^{-4}$	$1.900 \cdot 10^{-5}$	$1.356 \cdot 10^{-4}$	$8.587 \cdot 10^{-6}$	$3.875 \cdot 10^{-4}$	$1.665 \cdot 10^{-11}$
	$6.022 \cdot 10^{-5}$	$1.639 \cdot 10^{-4}$	$1.900 \cdot 10^{-5}$	$1.353 \cdot 10^{-4}$	$8.587 \cdot 10^{-6}$	$3.871 \cdot 10^{-4}$	$1.671 \cdot 10^{-11}$
	$6.022 \cdot 10^{-5}$	$1.637 \cdot 10^{-4}$	$1.901 \cdot 10^{-5}$	$1.324 \cdot 10^{-2}$	$8.587 \cdot 10^{-6}$	$1.349 \cdot 10^{-2}$	$1.690 \cdot 10^{-8}$
	$6.022 \cdot 10^{-5}$	$1.640 \cdot 10^{-4}$	$2.135 \cdot 10^{-5}$	$1.346 \cdot 10^{-4}$	$4.742 \cdot 10^{-1}$	$4.751 \cdot 10^{-1}$	$2.717 \cdot 10^{-3}$
$ML^{patch,36}_{opt,multi,worst}$	$1.650 \cdot 10^{-1}$	$1.367 \cdot 10^{-4}$	$1.900 \cdot 10^{-5}$	$1.352 \cdot 10^{-4}$	$8.563 \cdot 10^{-6}$	$1.660 \cdot 10^{-1}$	$2.705 \cdot 10^{-2}$

Tabelle 4.4: Funktionswerte der durch die mehrkriterielle Optimierung erhaltenen Lösungen, welche sortiert nach der gewichteten Summe $L_{total}(w)$ mit $w = (1,1,1,1)$ aufgeführt sind

Die Werte MSE(V_p, ML) dieser finalen Lösungen reichen von $1.665 \cdot 10^{-11}$ für das beste Modell $ML^{patch,36}_{opt,multi,best}$ bis zu $2.705 \cdot 10^{-2}$ für das Modell $ML^{patch,36}_{opt,multi,worst}$ in der finalen Population mit der größten Abweichung. Auch ohne eine Gewichtung zu definieren, kann somit ein geeignetes ML-Materialmodell erzielt werden. Allerdings kann auch eine in der

Optimierung nicht-dominierte, gefundene Lösung einen hohen Fehler in der Vorhersage aufweisen. Auch hier wäre daher eine finale Evaluierung durch die Anwendung der verschiedenen, erhaltenen ML-Modelle notwendig.

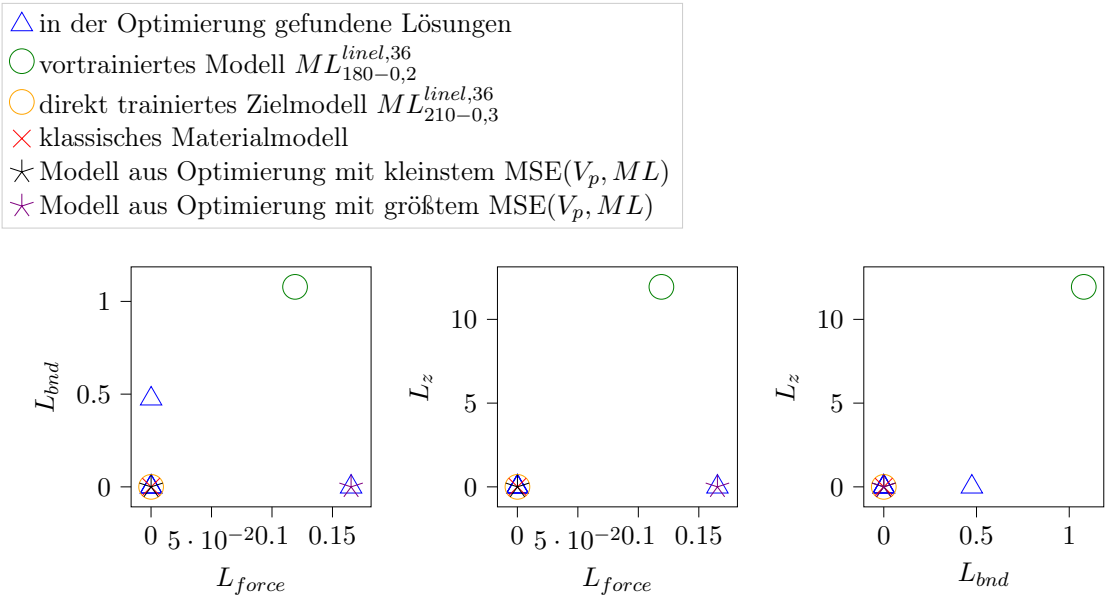


Abbildung 4.12: Die Fehlerfunktionswerte für die finalen, nicht-dominierten Lösungen des NSGA-III mit einer Populationsgröße von 30 sowie für das Start- und Zielmodell und das klassische Materialmodell. Die beiden in der Optimierung erhaltenen Lösungen mit den niedrigsten MSE(V_p, ML)-Werten (siehe auch Zeile 1 und 2 in Tabelle 4.4) weisen in jeder in der Optimierung berücksichtigten Fehlerfunktionen ähnlich niedrige Werte wie das direkt trainierte Zielmodell $ML^{linel,36}_{210-0,3}$ auf. Eine Verringerung einzelner Zielfunktionswerte dem gegenüber geht deutlich zu Lasten einer Erhöhung anderer Zielfunktionskomponenten (siehe auch Zeile 3-5 in Tabelle 4.4) und verbunden damit einem Anstieg der MSE(V_p, ML)-Werte.

Werden die Werte $L_{total}(w)$ für $w = (1, 1, 1, 1, 1)$ in Tabelle 4.4 für die final erhaltenen ML-Materialmodelle der mehrkriteriellen Optimierung zusammen mit den Werten für MSE(V_p, ML) betrachtet (auch wenn der Wert $L_{total}(w)$ in der Optimierung nicht genutzt wurde), so ist auch hier zu erkennen, dass eine Skalarisierung mit einer gleichen Gewichtung der unterschiedlichen Zielfunktionen für das auf diesen Anwendungsfall bezogene Optimierungsproblem einen guten Ansatzpunkt für das Training der ML-Materialmodelle bietet.

	$L_{total}(w)$ mit $w = (1,1,1,1,1)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	$\text{MSE}(V_p, ML)$
*MAT_ELASTIC	$2.77 \cdot 10^{-4}$	$6.06 \cdot 10^{-5}$	$1.64 \cdot 10^{-4}$	$3.62 \cdot 10^{-5}$	0	$1.69 \cdot 10^{-5}$	-
$ML^{linel,36}_{180-0,2}$	13.14	$1.19 \cdot 10^{-1}$	$1.33 \cdot 10^{-4}$	$2.84 \cdot 10^{-5}$	11.94	1.08	$6.39 \cdot 10^{-2}$
$ML^{linel,36}_{210-0,3}$	$3.15 \cdot 10^{-4}$	$6.08 \cdot 10^{-5}$	$1.64 \cdot 10^{-4}$	$3.62 \cdot 10^{-5}$	$3.69 \cdot 10^{-5}$	$1.72 \cdot 10^{-5}$	$1.51 \cdot 10^{-12}$
$ML^{patch,36}_{opt,CMA(\sigma_0=0.0001)}$	$3.33 \cdot 10^{-4}$	$6.03 \cdot 10^{-5}$	$1.64 \cdot 10^{-4}$	$2.93 \cdot 10^{-5}$	$6.48 \cdot 10^{-5}$	$1.44 \cdot 10^{-5}$	$1.69 \cdot 10^{-11}$
$ML^{patch,36}_{opt,Grad(\eta=10)}$	$3.14 \cdot 10^{-4}$	$6.03 \cdot 10^{-5}$	$1.65 \cdot 10^{-4}$	$2.73 \cdot 10^{-5}$	$4.89 \cdot 10^{-5}$	$1.29 \cdot 10^{-5}$	$1.26 \cdot 10^{-11}$
$ML^{patch,36}_{opt,multi,best}$	$3.88 \cdot 10^{-4}$	$6.02 \cdot 10^{-5}$	$1.64 \cdot 10^{-4}$	$1.90 \cdot 10^{-5}$	$1.35 \cdot 10^{-4}$	$8.58 \cdot 10^{-6}$	$1.66 \cdot 10^{-11}$
$ML^{patch,36}_{opt,multi,worst}$	$1.66 \cdot 10^{-1}$	$1.65 \cdot 10^{-1}$	$1.37 \cdot 10^{-4}$	$1.90 \cdot 10^{-5}$	$1.35 \cdot 10^{-4}$	$8.56 \cdot 10^{-6}$	$2.71 \cdot 10^{-2}$

Tabelle 4.5: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ bei einer gleichen Gewichtung $w = (1,1,1,1,1)$ und der Abweichung $\text{MSE}(V_p, ML)$ für die unterschiedlich optimierten Modelle sowie des vortrainierten Startmodells $ML^{linel,36}_{180-0,2}$, des direkt auf Spannungs-Dehnungsdaten trainierten Zielmodells $ML^{linel,36}_{210-0,3}$ und für die Spannungen des klassischen Materialmodells *MAT_ELASTIC

Da das neuronale Netz $ML^{linel,36}_{180-0,2}$ einen Sonderfall eines ML-Modells darstellt, in der Hinsicht, dass die Anzahl an trainierbaren Parameter minimal für die zu erlernende Aufgabe ist, wird als nächstes das nichtlineare Startmodell $ML^{linel,384}_{180-0,2}$ mit 384 trainierbaren Parametern betrachtet. Der gewichtete Gesamtfehler $L_{total}(w)$ des Startmodells $ML^{linel,384}_{180-0,2}$ liegt hier bei 13.165 für eine gleiche Gewichtung der einzelnen Fehlerfunktionen ($w = (1,1,1,1,1)$) und für das direkt trainierte Ziel-ML-Modell bei $L_{total}(w) = 0.161$. Das Modell übertrifft diesen Wert in der Optimierung mit dem CMA-Algorithmus (mit einer initialen Schrittweite von $\sigma_0 = 10^{-7}$) nach 1200 Iterationen (mit 21 Funktionsauswertungen je Iteration) und endet nach 5000 Iterationen mit einem Fehlerwert von $1.534 \cdot 10^{-2}$. Das optimierte ML-Materialmodell erreicht bezüglich seiner Prognose bei den Daten aus der Patch-Simulation eine Abweichung von $\text{MSE}(V_p, ML) = 2.969 \cdot 10^{-6}$ nach 5000 Iterationen, was zwar nicht an den Wert des optimierten minimalen, linearen Modells $ML^{patch,36}_{opt,CMA(\sigma_0=0.0001)}$ von $1.69 \cdot 10^{-11}$ herankommt, allerdings trotzdem noch einer sehr guten Prognose entspricht und in einem ähnlichen Bereich wie das direkt trainierte ML-Materialmodell $ML^{linel,384}_{210-0,3}$ mit der gleichen Netzarchitektur liegt, welches einen $\text{MSE}(V_p, ML)$-Wert von $1.397 \cdot 10^{-6}$ aufweist. Somit lässt sich schließen, dass auch dieses ML-Materialmodell mit deutlich mehr trainierbaren Parametern als nötig, um ein linear elastisches Materialmodell zu modellieren, auf Basis des betrachteten Patch-Zugversuchs angepasst werden kann, um die im Versuch gesehenen Spannungen vorherzusagen. Der gesamte Verlauf von $L_{total}(w)$ und $\text{MSE}(V_p, ML)$ der Optimierung ist in Abbildung 4.13 zu sehen.

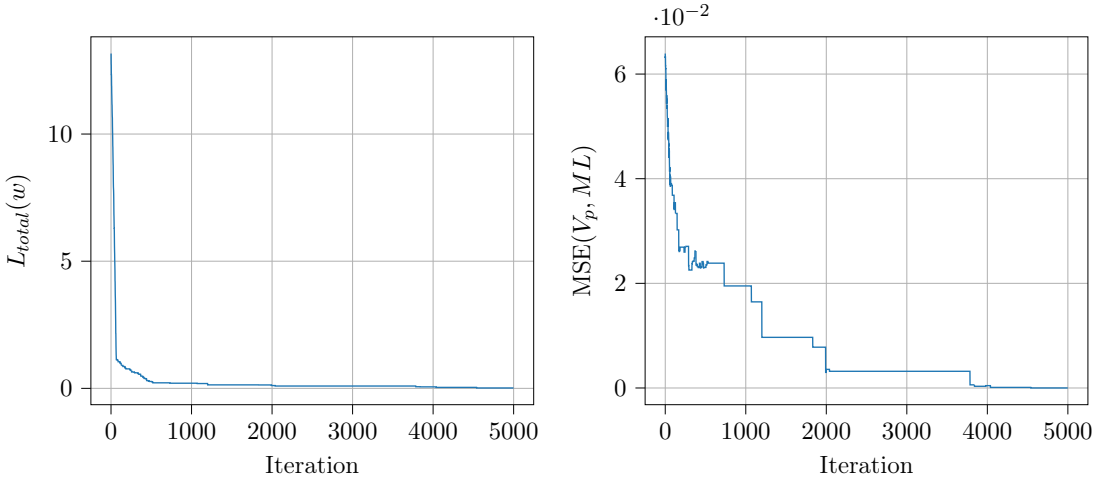


Abbildung 4.13: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ bei einer Gewichtung $w = (1, 1, 1, 1, 1)$ über die Iterationen bei einer Optimierung mittels CMA mit $\sigma_0 = 10^{-7}$ und dem Startmodell $ML^{linel,384}_{180-0,2}$, Rechts: Verlauf der mittleren quadratischen Abweichung MSE(V_p, ML) zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen des simulierten Versuchs

Da die ML-Materialmodelle für den allgemeinen Anwendungsfall trainiert werden sollen, ist es wichtig zu überprüfen, wie gut die Vorhersage der ML-Materialmodelle bei Dehnungs-Spannungsdaten, welche nicht im Versuch des Trainings enthalten sind, ist. Hierfür wird ein Validierungsdatensatz bestehend aus 10.000 zufällig generierten Dehnungswerten je Komponente nach einer Normalverteilung mit einem Mittelwert von 0 und einer Varianz von 0.001, angelehnt an in Simulationen beobachtbaren Werten, generiert. Die korrespondierenden Spannungen werden mithilfe des klassischen Materialmodells, wie in Abschnitt 3.1 beschrieben, berechnet. Abbildung 4.14 zeigt den Verlauf der Abweichung für diesen Testdatensatz, bezeichnet durch $\text{MSE}^{rand,rel}$ und den Verlauf von den im Versuch gesehenen Daten $\text{MSE}^{rel}(V_p, ML)$ in der Optimierung für die beiden verschiedenen Netzarchitekturen mit 36 beziehungsweise 384 Parametern jeweils normiert durch die Abweichung (MSE) des Startmodells $ML^{linel,36}_{180-0,2}$ beziehungsweise $ML^{linel,384}_{180-0,2}$, um die Fehlerwerte vergleichbar zu machen. In beiden Fällen ist deutlich zu erkennen, dass der Fehler für die im Training gesehenen Daten sinkt, während die Abweichung für ungesehene, zufällig gewählte Validierungsdaten auf einem ähnlich hohen Niveau bleibt, wie bei den nicht angepassten Startmodellen $ML^{linel,36}_{180-0,2}$ beziehungsweise $ML^{linel,384}_{180-0,2}$. Dies zeigt, dass der hier betrachtete Datensatz eines Zugversuchs mit einer viereckigen Probengeometrie nicht ausreicht, um ein allgemeingültiges ML-Materialmodell für einen linear elastischen Werkstoff zu trainieren. Aus diesem Grund wird in den folgenden Abschnitten zunächst

die Probengeometrie des Zugversuchs etwas erweitert und anschließend eine Möglichkeit aufgezeigt (abhängig des betrachteten Werkstoffs) weitere Trainingsdaten für das Training der ML-Materialmodelle basierend auf Versuchsdaten zu erzeugen.

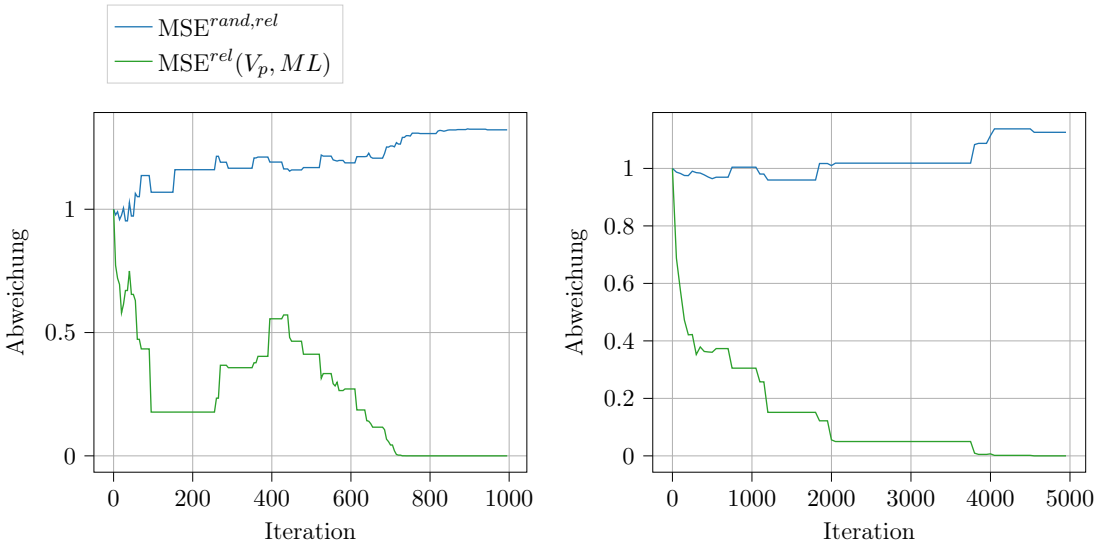


Abbildung 4.14: Fehlerverläufe für die im Training gesehenen Daten MSE$^{rel}(V_p, ML)$ und die zufällig erzeugten Testdaten MSE^{rand,rel} Links: Optimierung anhand des Startmodells $ML^{linel,36}_{180-0,2}$ mittels CMA für $\sigma_0 = 0.0001$, Rechts: Optimierung anhand des Startmodells $ML^{linel,384}_{180-0,2}$ mittels CMA für $\sigma_0 = 10^{-7}$

4.4.2 Training anhand eines erweiterten Ausschnitts eines Zugversuches

Der bisher betrachtete, rechteckige Ausschnitt der Zugprobe wird in diesem Abschnitt erweitert (siehe Abbildung 4.6 und 4.15), um eine leicht komplexere Probengeometrie zu betrachten und damit auf einen etwas erweiterten Trainingsdatensatz zurückzugreifen.

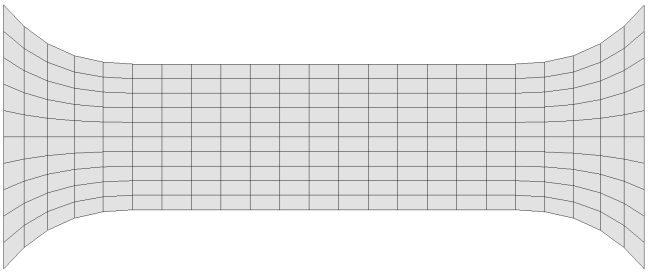


Abbildung 4.15: Aufbau Simulation erweiterter Ausschnitt des Zugversuchs

Auch bei dieser Probe werden die Knoten am linken äußeren Rand fest eingespannt und eine Verschiebung der rechten Randpunkten vorgegeben (vorgegebene Verschiebungsrandbedingung).

Abbildung 4.16 zeigt die Häufigkeitsverteilung der Spannungszustände der beiden Probegeometrien (Patch V_p und erweiterter Ausschnitt V_{mfz}). Während der Patch Ausschnitt bezüglich der Triaxialität lediglich aus einem einzigen Zustand besteht, liegt bei dem erweiterten Ausschnitt diesbezüglich ein etwas vielfältigerer Datensatz vor, welcher allerdings auch noch deutlich entfernt von einer vollständigen Abdeckung ist, unter anderem da ausschließlich Spannungszustände im Zugbereich vorliegen.

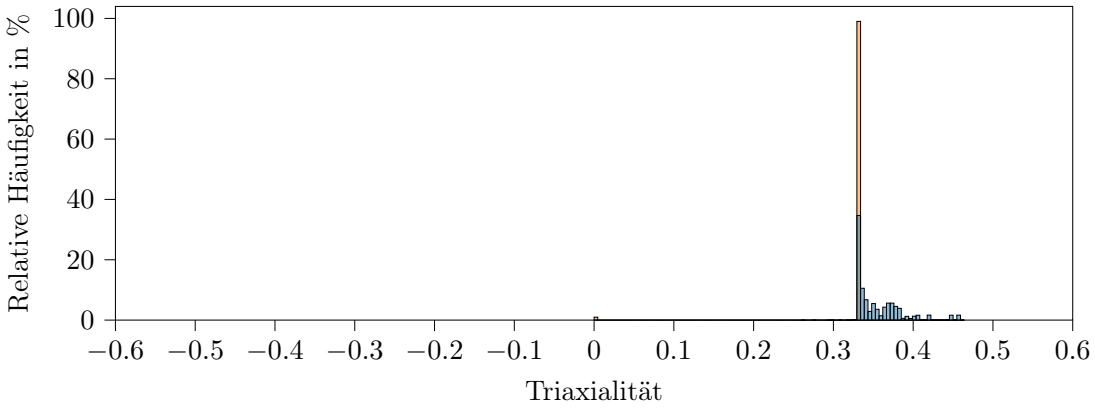


Abbildung 4.16: Häufigkeitsverteilung der Triaxialität der Spannungszustände (wie in (3.23) definiert) für die beiden Zugversuchsgeometrien Patch (orange) und den erweiterten Ausschnitt (blau)

Abbildung 4.17 zeigt auf der linken Seite den Verlauf der Optimierung mit dieser erweiterten Zugprobe für $L_{total}(w)$ mit $w = (1, 1, 1, 1, 1)$ mittels des CMA (bei einer initialen Schrittgröße von $\sigma_0 = 0.001$) und auf der rechten Seite den ebenfalls abnehmenden Verlauf der Abweichung $\text{MSE}(V_{mfz}, ML)$ zwischen den vorhergesagten und korrekten Spannungen des entsprechenden Versuchs. Auch für diesen Anwendungsfall ist zu sehen, dass das ML-Materialmodell die im Versuch enthaltenen Daten erlernt. Im Laufe der Optimierung wird eine Abweichung von $\text{MSE}(V_{mfz}, ML) = 7.064 \cdot 10^{-4}$ und damit ein deutlich besserer Wert als beim Startmodell $ML^{linel,36}_{180-0,2}$ mit einer Abweichung von $\text{MSE}(V_{mfz}, ML) = 2.025$ erreicht.

In Abbildung 4.18 sind die Fehlerverläufe des Trainings für $L_{total}(w)$ mit $w = (1, 1, 1, 1, 1)$ mittels des Gradienten und mit dem CMA-Algorithmus für verschiedene Lernraten η beziehungsweise unterschiedliche initiale Schrittgrößen σ_0 zu sehen.

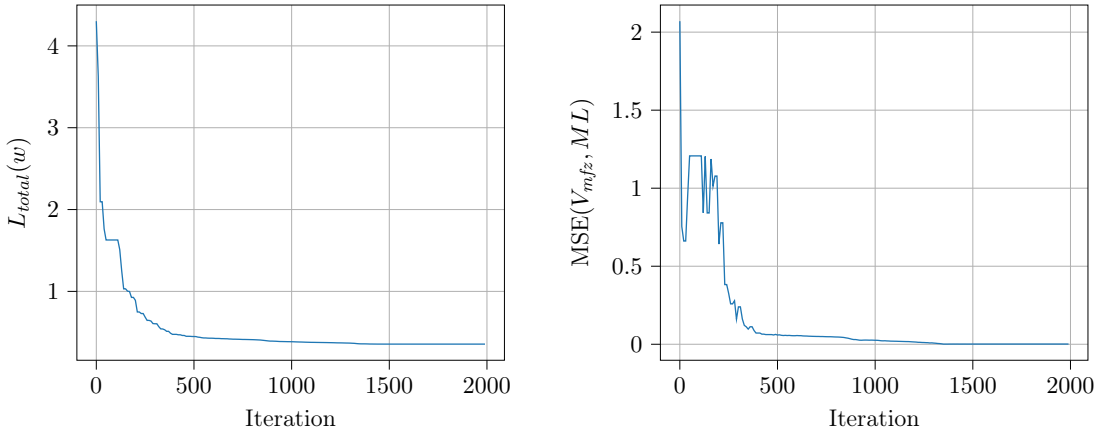


Abbildung 4.17: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ bei einer Gewichtung $w = (1, 1, 1, 1, 1)$ über die Iterationen bei einer Optimierung durch CMA mit $\sigma_0 = 0.001$, Rechts: Verlauf der mittleren quadratischen Abweichung MSE(V_{mfz}, ML) zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen des simulierten Versuchs

Für die in Abbildung 4.18 gezeigten CMA-Optimierungsverläufe wird wie auch im bisherigen Abschnitt der Optimierungsverlauf aus 20 durchgeführten Verläufen ausgewählt, welcher vom Verlauf des Mittelwerts aller 20 Optimierungsläufe die kleinste mittlere quadratische Abweichung bezogen auf den Verlauf des erreichten Zielfunktionswertes über die Iterationen der Optimierung hat.

Die Varianz der verschiedenen CMA-Verläufe im Laufe der Iterationen der Optimierung sowie der Verlauf des Mittelwerts sind in Abbildung 4.19 zu sehen. Die Varianz der unterschiedlichen Optimierungsläufe nimmt bei allen gewählten initialen Schrittgrößen im Laufe der Iterationen deutlich ab und erreicht je nach initialer Schrittgröße bezüglich des niedrigsten in der Optimierung erzielten Zielfunktionswerts einen sehr geringen Wert zwischen $2.341 \cdot 10^{-8}$ und $4.174 \cdot 10^{-12}$ (siehe auch Tabelle 4.6). Dies deutet auch für diesen simulierten Versuch darauf hin, dass der CMA-Algorithmus sehr stabile Ergebnisse liefert.

Im Unterschied zu der Optimierung mit dem Patch Ausschnitt der Zugprobe im vorherigen Abschnitt, wird hier ausschließlich mit dem CMA-Optimierungsverfahren der gewichtete Fehlerwert $L_{total}(w)$ des Zielmodells $ML^{linel,36}_{210-0,3}$ von 0.3597 (innerhalb der betrachteten Anzahl an Iterationen) erreicht. Bei einer initialen Schrittgröße von $\sigma_0 = 0.001$, $\sigma_0 = 0.0001$ und $\sigma_0 = 0.00001$ ist dies bei den betrachteten Verläufen nach 1344, 1235 beziehungsweise 1382 Iterationen der Fall. Im Unterschied dazu wird mittels des kombinierten Gradienten im besten Fall (für $\eta = 10$ ohne Adam-Optimierer) lediglich ein Wert von $L_{total} = 0.4479$

nach 6000 Iterationen erreicht, wie es auch deutlich in Abbildung 4.18 zu sehen ist. Die genauen Fehlerwerte (inklusive derer für die einzelnen Fehlerfunktionen) der unterschiedlich optimierten Modelle sowie des Ziel- und Startmodells finden sich in einer Übersicht in Tabelle 4.9.

Bei der Optimierung von einzelnen Fehlerfunktionen separat durch die Anwendung des Gradienten können im Unterschied zu der Optimierung von L_{total} die Zielfehlerwerte (gegeben durch die Werte des Modells $ML^{linel,36}_{210-0,3}$), wie in Abbildung 4.20 zu sehen ist, durch diese Art der Optimierung erreicht werden. Der Zielfunktionswert wird nur bei der Kombination der verschiedenen Fehlerfunktionen bei einer gradientenbasierten Optimierung nicht erreicht.

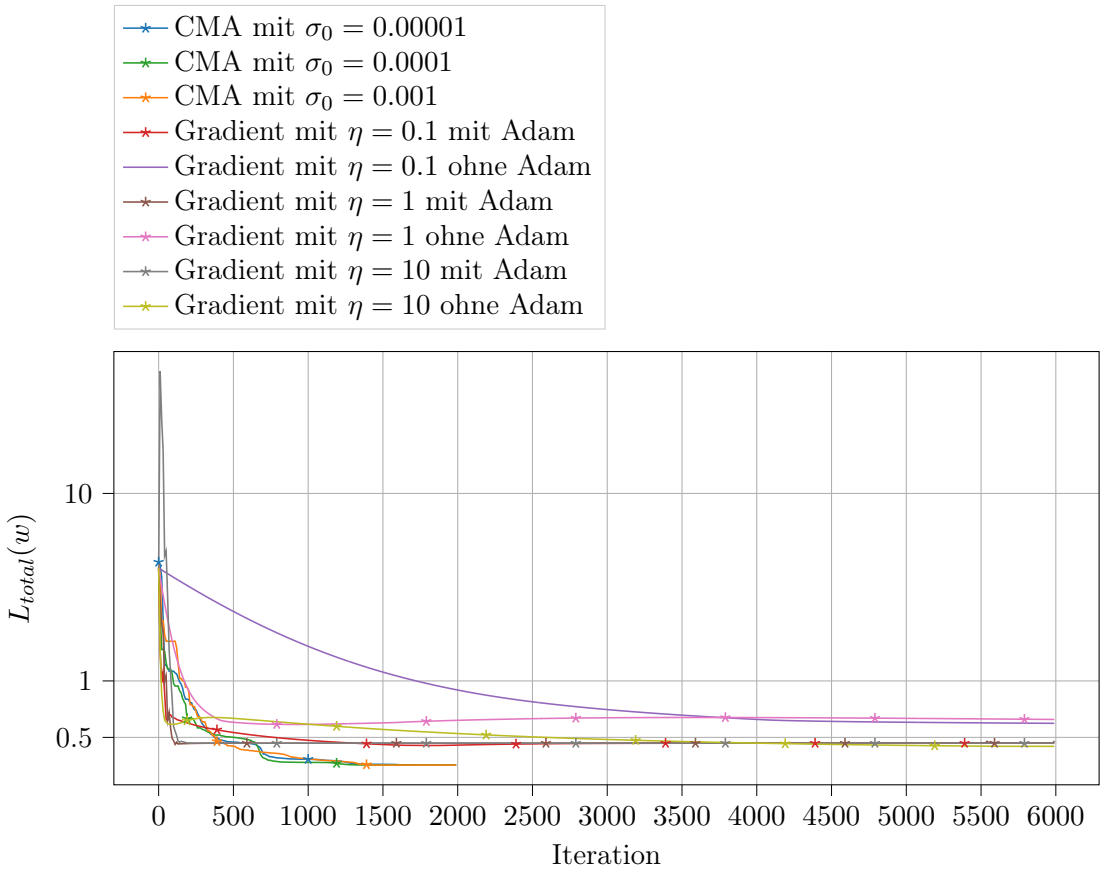


Abbildung 4.18: Vergleich der Optimierungsverläufe von $L_{total}(w)$ für die Gewichtung $w = (1, 1, 1, 1, 1)$ mittels verschiedener Optimierungsstrategien (CMA und gradientenbasierte Optimierung) bei unterschiedlichen Lernraten η beziehungsweise Schrittgrößen σ_0

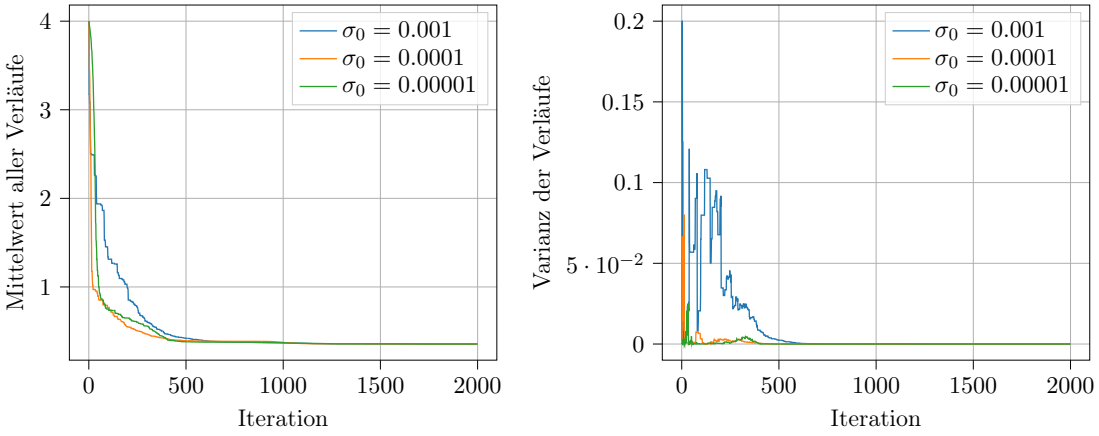


Abbildung 4.19: Links: Verlauf des Mittelwertes aller 20 Optimierungsläufe von $L_{total}(w)$, $w = (1, 1, 1, 1, 1)$ mittels des CMA mit der jeweiligen initialen Schrittgröße σ_0 je Iteration, wobei je Iteration der kleinste, bisher erreichte Zielfunktionswert in der Optimierung betrachtet wird, Rechts: Varianz der 20 Optimierungsläufe im Laufe der Iterationen

Initiale Schrittgröße σ_0	Varianz gesamter Verlauf	Varianz der erreichten Zielfunktionswerte	Mittelwert der erreichten Zielfunktionswerte	Minimaler erreichter Zielfunktionswert	Maximaler erreichter Zielfunktionswert
0.001	$1.897 \cdot 10^{-1}$	$3.263 \cdot 10^{-9}$	$3.557 \cdot 10^{-1}$	$3.557 \cdot 10^{-1}$	$3.558 \cdot 10^{-1}$
0.0001	$6.475 \cdot 10^{-2}$	$2.341 \cdot 10^{-8}$	$3.558 \cdot 10^{-1}$	$3.557 \cdot 10^{-1}$	$3.560 \cdot 10^{-1}$
0.00001	$1.773 \cdot 10^{-1}$	$4.174 \cdot 10^{-12}$	$3.557 \cdot 10^{-1}$	$3.557 \cdot 10^{-1}$	$3.557 \cdot 10^{-1}$

Tabelle 4.6: Übersicht über die Varianz der 20 unterschiedlichen CMA-Optimierungsverläufe für die jeweilige Wahl der initialen Schrittgröße σ_0 bezüglich des gesamten Verlaufs und bei der ausschließlichen Betrachtung des nach 2000 Iterationen erreichten minimalen Zielfunktionswertes sowie der Mittelwert, Maximal- und Minimalwert der erreichten minimalen Zielfunktionswerte aller 20 Optimierungsläufe. Die niedrigen Varianzen der erreichten Zielfunktionswerte sowie die sehr eng beieinander liegenden Minimal- und Maximalwerte deuten auf eine hohe Robustheit des Optimierungsansatzes in Bezug auf das vorliegende Trainingsproblem hin.

Abbildung 4.21 zeigt den Verlauf von $\text{MSE}(V_{mfz}, ML)$ bei einer Optimierung der einzelnen Fehlerfunktionen separat sowie bei der Kombination der Fehlerfunktionen $L_{total}(w)$ mit $w = (1, 1, 1, 1, 1)$ im direkten Vergleich. Wie auch bei der Optimierung mittels des Patch Ausschnitts sinkt die Abweichung $\text{MSE}(V_{mfz}, ML)$ lediglich bei der Kombination aller Fehlerfunktionen in ausreichendem Maße, was zeigt, dass dem ML-Modell die einzelnen Fehlerfunktionen alleine nicht ausreichen, um die Daten aus dem Versuch zu erlernen. Die

Optimierung der Fehlerfunktionen in Abbildung 4.21 erfolgte jeweils mittels Anwendung des Gradienten mit einer Lernrate $\eta = 1$.

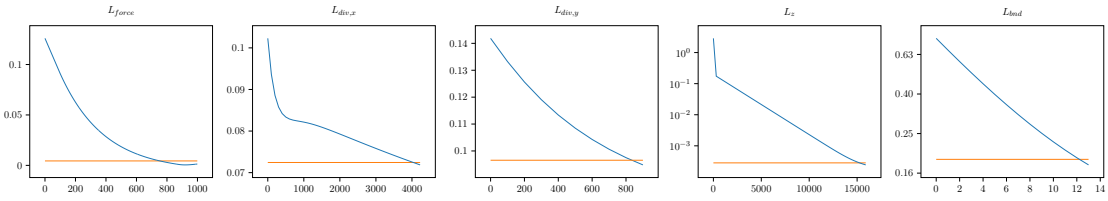


Abbildung 4.20: Optimierungsverläufe mittels Anwendung des Gradienten bei der Optimierung der einzelnen Fehlerfunktionen separat (blau) und die Werte der einzelnen Fehlerfunktionen für das Ziel-ML-Modell $ML^{linel,36}_{210-0,3}$ (orange)

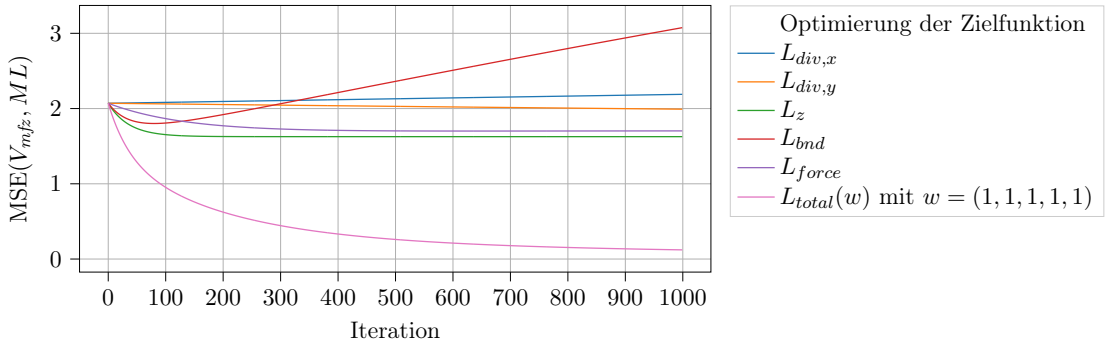


Abbildung 4.21: Verlauf des MSE(V_{mfz}, ML) für die ML-Materialmodelle im Laufe der Optimierung anhand einzelner Fehlerfunktionen im Vergleich zu einer Kombination aller Fehlerfunktion als gewichtete Summe mit Gewichtung $w = (1, 1, 1, 1, 1)$. Der MSE(V_{mfz}, ML)-Wert bezieht sich auf das beste bis zur jeweiligen Iteration in der Optimierung erhaltene ML-Modell bezüglich der in der Optimierung berücksichtigten Zielfunktion (Fehlerfunktion). Die Vorhersagefähigkeit des optimierten ML-Materialmodells für die im Versuch vorhandenen Daten verbessert sich in signifikantem Maß lediglich bei der Kombination der unterschiedlichen Fehlerfunktionen und nicht durch die Optimierung der einzelnen Fehlerfunktionen.

Tabelle 4.7 zeigt die Ergebnisse von Optimierungen der Skalarisierung, welche mit unterschiedlichen Gewichtungen der einzelnen Fehlerfunktionen durchgeführt wurden. Im Gegensatz zum Training anhand des Patch Ausschnitts sind (teilweise deutliche) Unterschiede bezüglich des MSE(V_{mfz}, ML) zu sehen, allerdings durchgehend mit abnehmender Prognosefähigkeit der Modelle bei einer höheren Gewichtung einzelner Zielfunktionen. Insgesamt ist zu sehen, dass hier die Optimierungsergebnisse weniger robust bezüglich variierender Gewichtungen sind. Weiterhin sind die Fehlerfunktionen für die Divergenz

$L_{div,x}$ und $L_{div,y}$ mitentscheidend für eine gute Prognose des trainierten ML-Modells, da sich ohne Beachtung dieser die Abweichung MSE(V_{mfz}, ML) deutlich erhöht, wie in der letzten Zeile in Tabelle 4.7 ersichtlich wird.

Gewichtung $w = (w_f, w_{d,x}, w_{d,y}, w_{b,z}, w_b)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	$L_{total}(w)$	$L^{\star}_{total}(w)$	$\frac{L^{\star}_{total}(w)}{L_{total}(w)}$	MSE(V_{mfz}, ML)
(1,1,1,1,1)	$4.226 \cdot 10^{-3}$	$7.389 \cdot 10^{-2}$	$9.001 \cdot 10^{-2}$	$1.986 \cdot 10^{-4}$	0.187	0.356	0.360	1.011	$7.064 \cdot 10^{-4}$
(10,1,1,1,1)	$4.068 \cdot 10^{-3}$	$8.155 \cdot 10^{-2}$	$8.806 \cdot 10^{-2}$	$2.109 \cdot 10^{-4}$	0.188	0.398	0.399	1.003	$5.880 \cdot 10^{-3}$
(1,1,1,10,1)	$4.236 \cdot 10^{-3}$	$7.375 \cdot 10^{-2}$	$8.835 \cdot 10^{-2}$	$2.426 \cdot 10^{-4}$	0.190	0.359	0.362	1.010	$2.436 \cdot 10^{-3}$
(1,1,1,1,10)	$2.446 \cdot 10^{-1}$	$1.275 \cdot 10^{-1}$	$8.226 \cdot 10^{-2}$	$3.212 \cdot 10^{-1}$	0.122	1.999	2.034	1.018	1.739
(1,10,10,1,1)	$2.369 \cdot 10^{-1}$	$5.504 \cdot 10^{-2}$	$7.085 \cdot 10^{-2}$	$1.708 \cdot 10^{-1}$	0.171	1.837	1.881	1.024	1.615
(1,0,0,1,1)	$4.421 \cdot 10^{-3}$	$1.575 \cdot 10^{-1}$	$1.443 \cdot 10^{-1}$	$2.945 \cdot 10^{-2}$	0.151	0.185	0.191	1.032	$2.051 \cdot 10^{-2}$

Tabelle 4.7: Funktionswerte der verschiedenen Fehlerfunktionen und des Gesamtfehlers $L_{total}(w)$ der trainierten ML-Modelle bei einer Optimierung mit unterschiedlichen Gewichtungen

Für den mehrkriteriellen Optimierungsansatz werden daher für die hier betrachtete Versuchsprobe alle fünf Fehlerfunktionen berücksichtigt. Verwendet wird für die Optimierung auch hier der NSGA-III. Die finale Population nach 1000 Iterationen besteht aus 24 Lösungen mit MSE(V_{mfz}, ML)-Werten, welche im besten Fall von $5.447 \cdot 10^{-4}$ für das Modell $ML^{mfz,36}_{opt,multi,best}$ bis 1.756 für die Lösung $ML^{mfz,36}_{opt,multi,worst}$ mit dem höchsten Wert reichen. Tabelle 4.8 zeigt die einzelnen Werte der besten zwei Lösungen (Zeile 1 und 2) sowie der zwei Lösungen mit dem höchsten MSE(V_{mfz}, ML)-Wert (Zeile 3 und 4). Auch hier ist ein niedriger Wert der Summe $L_{total}(w)$ mit einer gleichen Gewichtung der einzelnen Zielfunktionen ein guter Indikator für die Prognosefähigkeit der optimierten Modelle (wie es in der vorletzten Spalte von Tabelle 4.8 erkennbar wird). Allerdings hat die beste Lösung der multikriteriellen Optimierung einen sehr hohen Wert für L_{total} im Vergleich zu der Lösung $ML^{mfz,36}_{opt,CMA(\sigma_0=0.001)}$ aus der Optimierung von $L_{total}(w)$ mit $w = (1,1,1,1,1)$ mit gleichzeitig einer sogar etwas kleineren Abweichung MSE(V_{mfz}, ML), wie in Tabelle 4.9 zu sehen ist.

	Rang	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	L_{total} mit $w = (1,1,1,1,1)$	MSE(V_p, ML)
$ML^{mfz,36}_{opt,multi,best}$	1	0.0115	$1.601 \cdot 10^{-4}$	$1.360 \cdot 10^{-6}$	3.618	$5.692 \cdot 10^{-3}$	3.635	$5.447 \cdot 10^{-4}$
	2	0.0163	$1.599 \cdot 10^{-4}$	$3.671 \cdot 10^{-5}$	9.754	0.223	9.994	$4.663 \cdot 10^{-3}$
	23	0.4153	$2.612 \cdot 10^{-5}$	$8.089 \cdot 10^{-6}$	77.717	5.994	84.127	1.617
$ML^{mfz,36}_{opt,multi,worst}$	24	0.2988	$4.045 \cdot 10^{-5}$	$3.461 \cdot 10^{-5}$	115.249	6.204	121.752	1.756

Tabelle 4.8: Funktionswerte der beiden Lösungen mit dem kleinsten MSE(V_{mfz}, ML) (Rang 1 und 2) und dem höchsten MSE(V_{mfz}, ML) (Rang 23 und 24) in der finalen Population der mehrkriteriellen Optimierung

	$L_{total}(w)$ mit $w = (1,1,1,1,1)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_z	L_{bnd}	$\text{MSE}(V_p, ML)$
⋆MAT_ELASTIC	0.3595	$4.406 \cdot 10^{-3}$	$7.254 \cdot 10^{-2}$	$9.652 \cdot 10^{-2}$	0.0	0.186	-
$ML^{linel,36}_{180-0,2}$	4.0	$1.262 \cdot 10^{-1}$	$1.023 \cdot 10^{-1}$	$1.420 \cdot 10^{-1}$	2.869	0.761	2.025
$ML^{linel,36}_{210-0,3}$	0.3597	$4.407 \cdot 10^{-3}$	$7.244 \cdot 10^{-2}$	$9.655 \cdot 10^{-2}$	$2.828 \cdot 10^{-4}$	0.186	$1.056 \cdot 10^{-7}$
$ML^{mfz,36}_{opt,CMA(\sigma_0=0.001)}$	0.356	$4.226 \cdot 10^{-3}$	$7.389 \cdot 10^{-2}$	$9.001 \cdot 10^{-2}$	$1.986 \cdot 10^{-4}$	0.187	$7.064 \cdot 10^{-4}$
$ML^{mfz,36}_{opt,Grad(\eta=1)}$	0.466	$1.442 \cdot 10^{-2}$	$2.051 \cdot 10^{-1}$	$1.132 \cdot 10^{-1}$	$1.539 \cdot 10^{-2}$	0.118	$3.377 \cdot 10^{-2}$
$ML^{mfz,36}_{opt,multi,best}$	3.635	$1.201 \cdot 10^{-2}$	$1.601 \cdot 10^{-4}$	$1.360 \cdot 10^{-6}$	3.618	$5.692 \cdot 10^{-3}$	$5.447 \cdot 10^{-4}$
$ML^{mfz,36}_{opt,multiworst}$	121.752	$2.993 \cdot 10^{-1}$	$4.045 \cdot 10^{-5}$	$3.461 \cdot 10^{-5}$	115.249	6.204	1.756

Tabelle 4.9: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ bei einer gleichen Gewichtung $w = (1,1,1,1,1)$ und der Abweichung $\text{MSE}(V_{mfz}, ML)$ für die unterschiedlich optimierten Modelle sowie des vortrainierten Startmodells $ML^{linel,36}_{180-0,2}$, des direkt auf Spannungs-Dehnungsdaten trainierten Zielmodells $ML^{linel,36}_{210-0,3}$ sowie für die Spannungen des klassischen Materialmodells ⋆MAT_ELASTIC

Um zu überprüfen, wie gut das anhand des erweiterten Ausschnitts des Zugversuches optimierten ML-Materialmodells ungesehene Spannungen vorhersagen kann, wird der zufällig erzeugte Testdatensatz, welcher auch schon im vorherigen Abschnitt betrachtet wurde, herangezogen. Abbildung 4.22 zeigt den Verlauf der relativen Abweichung $\text{MSE}^{rand,rel}$ für diesen Testdatensatz im Laufe der Optimierung von $L_{total}(w)$ mit $w = (1,1,1,1,1)$. Links in der Abbildung ist der Vergleich zwischen der relativen Abweichung der ungesehenen Testdaten $\text{MSE}^{rand,rel}$ und den im Versuch gesehenen Daten $\text{MSE}^{rel}(V_{mfz}, ML)$ zu sehen. Während der Fehler $\text{MSE}^{rel}(V_{mfz}, ML)$ im Laufe der Optimierung deutlich abnimmt, kann nur ein leichter Rückgang für $\text{MSE}^{rand,rel}$ verzeichnet werden. Allerdings ist im Vergleich zu der Optimierung anhand des Patch Ausschnitts (V_p) im vorherigen Abschnitt immerhin eine leichte Verbesserung des $\text{MSE}^{rand,rel}$ erkennbar, wie rechts in der Abbildung 4.22 zu sehen ist. Für ein allgemeingültiges ML-Materialmodell reichen jedoch beide Versuchsproben nicht aus, weswegen im folgenden Abschnitt noch eine weitere Möglichkeit betrachtet wird, für bestimmte Materialmodelle den Trainingsdatensatz zu erweitern und in den Abschnitten 4.5.2 und 4.5.3 für den allgemeinen Fall zusätzlich komplexere Versuchsproben sowie das Training anhand mehrerer, kombinierter Versuche betrachtet wird. Auch wenn der erweiterte Datensatz des in diesem Abschnitt betrachteten Versuchs noch nicht für eine gute Verallgemeinerung des ML-Modells ausreicht, zeigt dieser Anwendungsfall einerseits, dass im Allgemeinen alle fünf betrachteten Fehlerfunktionen kombiniert berücksichtigt werden müssen, damit die Spannungen beim Training mit Versuchsdaten vom ML-Materialmodell erlernt werden. Weiterhin ist zu sehen, dass die Gewichtung eine entscheidende Rolle spielen kann und dass für das Training der

ML-Modelle anhand der Kombination der verschiedenen Fehlerfunktionen evolutionäre Strategien vorteilhafter als gradientenbasierte Optimierungsmethoden sein können.

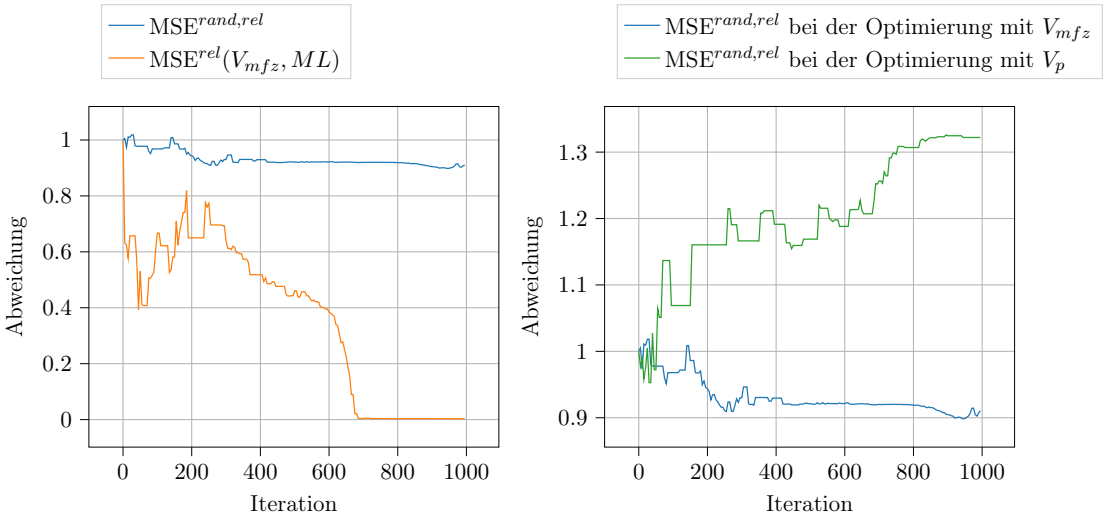


Abbildung 4.22: Fehlerverläufe für MSE^{rand,rel} Links: Vergleich der Verläufe für MSE^{rand,rel} und MSE$^{rel}(V_{mfz}, ML)$ bei der Optimierung mittels CMA für $\sigma_0 = 0.001$ anhand V_{mfz}, Rechts: Vergleich der Verläufe von MSE^{rand,rel} für die Optimierung anhand V_{mfz} und V_p

4.4.3 Verbesserung der Verallgemeinerungsfähigkeit durch Datenaugmentierung mithilfe Transformationen

Im Folgenden wird eine Methode gezeigt, welche für eine bestimmte Art von Materialmodellen genutzt werden kann, um eine verbesserte Verallgemeinerungsfähigkeit der anhand von Versuchen trainierten ML-Materialmodellen zu erreichen. Die in diesem Abschnitt thematisierte Idee zur Verbesserung der Verallgemeinerungsfähigkeit der ML-Materialmodelle durch Datenaugmentierung wurde bereits in [9] für einen Anwendungsfall gezeigt und wird im Folgenden verallgemeinert dargestellt und auf das linear elastische Materialmodell angewendet.

Aus der inneren Struktur von Werkstoffen können sich Eigenschaften ableiten, welche im klassischen Materialmodell $f : \mathbb{R}^6 \to \mathbb{R}^6, \varepsilon \mapsto \sigma = f(\varepsilon)$ durch Transformationsinvarianz mit einer Menge an bestimmten Transformationsabbildungen $\mathcal{T} = \{T \mid T : \mathbb{R}^6 \to \mathbb{R}^6\}$ (beispielsweise Symmetrie- oder Rotationstransformationen) berücksichtigt werden und formal durch die Bedingung

$$f(T(\varepsilon)) = T(f(\varepsilon)) \quad \forall \varepsilon \in \mathbb{R}^6 \tag{4.20}$$

beschrieben werden kann. Gilt diese Bedingung für einen Werkstoff mit einer gegebenen Menge an Transformationsabbildungen $\mathcal{T}$, so kann diese Eigenschaft zur Datenaugmentierung bei dem Training mit Versuchsdaten genutzt werden, um zusätzliche, nicht im Versuch enthaltene Trainingsdaten zu generieren. Hierfür werden die Dehnungen $\varepsilon \in V_\varepsilon$ aus dem Versuch anhand dessen das ML-Materialmodell trainiert wurde und die vorhergesagten Spannungen $\sigma \in V_\sigma \coloneqq \{\sigma \mid \sigma = ML(\varepsilon) \text{ mit } \varepsilon \in V_\varepsilon\}$ vom ML-Modell genutzt. Unter der Annahme, dass das ML-Modell durch das Training gelernt hat diese Menge der Spannungen hinreichend genau vorherzusagen, können Transformationsabbildungen erzeugt werden, um die Dehnungen und vorhergesagten Spannungen gleichermaßen zu transformieren

$$D_V^{transf} \coloneqq \{(\varepsilon^{transf}, \sigma^{transf}) \mid \varepsilon^{transf} = T(\varepsilon), \sigma^{transf} = T(ML(\varepsilon)) \text{ mit } \varepsilon \in V_\varepsilon, T \in \mathcal{T}\} \tag{4.21}$$

und so eine neue Trainingsdatenmenge D_V^{transf} mit nicht im Versuch betrachteten Dehnungen und hinreichend genauer Spannungsprognose erzeugen zu können. Die im Versuch trainierten ML-Materialmodelle können mit den zusätzlichen Daten auf konventionelle Art (MSE Fehler der Spannungswerte und Backpropagation) weitertrainiert werden, um eine bessere Verallgemeinerungsfähigkeit der ML-Modelle zu erreichen.

Für das hier betrachtete linear elastische Materialmodell können aufgrund der Isotropie (siehe [6]), wodurch ein richtungsunabhängiges Materialverhalten bezeichnet wird, Transformationen mittels Rotationsmatrizen $Q(\alpha, v) \in \mathbb{R}^{3\times3}$ um einen zufällig gewählten Winkel α und eine Rotationsachse $v = (v_1, v_2, v_3)^T$ (welcher als normierter Einheitsvektor vorausgesetzt wird) genutzt werden. Die Matrizen können durch

$$Q(\alpha,v) = \begin{pmatrix} v_1^2+(v_2^2+v_3^2)cos(\alpha) & v_1v_2(1-cos(\alpha))-v_3sin(\alpha) & v_1v_3(1-cos(\alpha))+v_2sin(\alpha) \\ v_1v_2(1-cos(\alpha))+v_3sin(\alpha) & v_2^2+(v_1^2+v_3^2)cos(\alpha) & v_2v_3(1-cos(\alpha))+v_1sin(\alpha) \\ v_1v_3(1-cos(\alpha))-v_2sin(\alpha) & v_2v_3(1-cos(\alpha))+v_3sin(\alpha) & v_3^2+(v_1^2+v_2^2)cos(\alpha) \end{pmatrix} \tag{4.22}$$

berechnet werden (siehe [19]), wobei der Winkel α zufällig, gleichverteilt aus dem Intervall $[0, 2\pi]$ erzeugt wird. Für die Rotationsachse werden Werte $x_1, x_2, x_3 \sim \mathcal{N}(0, 1)$ zufällig generiert und anschließend durch

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot \frac{1}{\sqrt{\sum_{i=1}^{3} x_i^2}} \tag{4.23}$$

normiert.

Als Anwendungsfall wird das auf Basis des einfachen Patch-Zugversuchs trainierte ML-Modell $ML^{patch,36}_{opt,CMA(\sigma_0=0.0001)}$ betrachtet, welches sich, wie in Abbildung 4.14 zu sehen, innerhalb der Optimierung bezüglich gänzlich ungesehener Daten nicht verbessert. Das ML-Modell weist einen Fehler von $\text{MSE}(V_p, ML) = 1.69 \cdot 10^{-11}$ bezüglich der Daten aus dem Versuch und einen Fehler von $\text{MSE}^{rand} = 0.115$ für die zufällig normalverteilten Validierungsdaten auf, was deutlich höher als das direkt an Dehnungs-Spannungspaaren trainierte Zielmodell $ML^{linel,36}_{210-0,3}$ mit einem Fehler von $\text{MSE}^{rand} = 1.811 \cdot 10^{-13}$ ist. Zu sehen ist dies auch im direkten Vergleich der trainierbaren Gewichte ($C_{ML^{patch,36}_{opt,CMA(\sigma_0=0.0001)}}$) zu der zu erreichenden Abbildung des klassischen Modells ($C^{210}_{0,3}$) in Matrixdarstellung:

$$C^{210}_{0,3} = \begin{pmatrix} 282.69 & 121.15 & 121.15 & 0 & 0 & 0 \\ 121.15 & 282.69 & 121.15 & 0 & 0 & 0 \\ 121.15 & 121.15 & 282.69 & 0 & 0 & 0 \\ 0 & 0 & 0 & 80.77 & 0 & 0 \\ 0 & 0 & 0 & 0.00 & 80.77 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80.77 \end{pmatrix} \tag{4.24}$$

$$C_{ML^{patch,36}_{opt,CMA(\sigma_0=0.0001)}} = \begin{pmatrix} 225.70 & 71.49 & 70.99 & 1.71 & 5.38 & -0.79 \\ 48.90 & 201.97 & 46.99 & 16.01 & 10.07 & 7.14 \\ 3.44 & 36.33 & 189.65 & -10.32 & 7.88 & -9.77 \\ -3.18 & -12.01 & 4.95 & 61.98 & 20.56 & 13.05 \\ 57.45 & -10.13 & 10.53 & -49.18 & 69.18 & -4.47 \\ 29.02 & 34.98 & -7.61 & -28.09 & 31.00 & 66.74 \end{pmatrix}. \tag{4.25}$$

Um den zusätzlichen Trainingsdatensatz $D^{transf}_{V_p}$ zu erzeugen, werden alle 8160 Dehnungen und die entsprechenden vorhergesagten Spannungen aus der Patch Simulation mittels 200 Rotationsmatrizen $Q \in \mathcal{Q}$ wie folgt transformiert:

$$\varepsilon^Q = \begin{pmatrix} \varepsilon^Q_{xx} & \varepsilon^Q_{xy} & \varepsilon^Q_{xz} \\ \varepsilon^Q_{yx} & \varepsilon^Q_{yy} & \varepsilon^Q_{yz} \\ \varepsilon^Q_{zx} & \varepsilon^Q_{zy} & \varepsilon^Q_{zz} \end{pmatrix} = Q \begin{pmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{pmatrix} Q^T. \tag{4.26}$$

Abbildung 4.23 zeigt den Verlauf der mittleren quadratischen Abweichungen MSE^{rand} über die Epochen des Trainings mit dem erzeugten Trainingsdatensatz $D^{transf}_{V_p}$ und dem $\mathcal{N}(0, 0.001)$-verteilten zufällig erzeugten Validierungsdatensatz, wobei hier die reguläre MSE-Fehlerfunktion und der Backpropagation-Algorithmus ohne Anpassungen verwendet werden können, da die Spannungen gegeben sind. Der Fehler für den Validierungsdatensatz

sinkt bei diesem zusätzlichen Trainingsschritt nach 20 Epochen auf $\text{MSE}^{rand} = 3.11 \cdot 10^{-13}$.

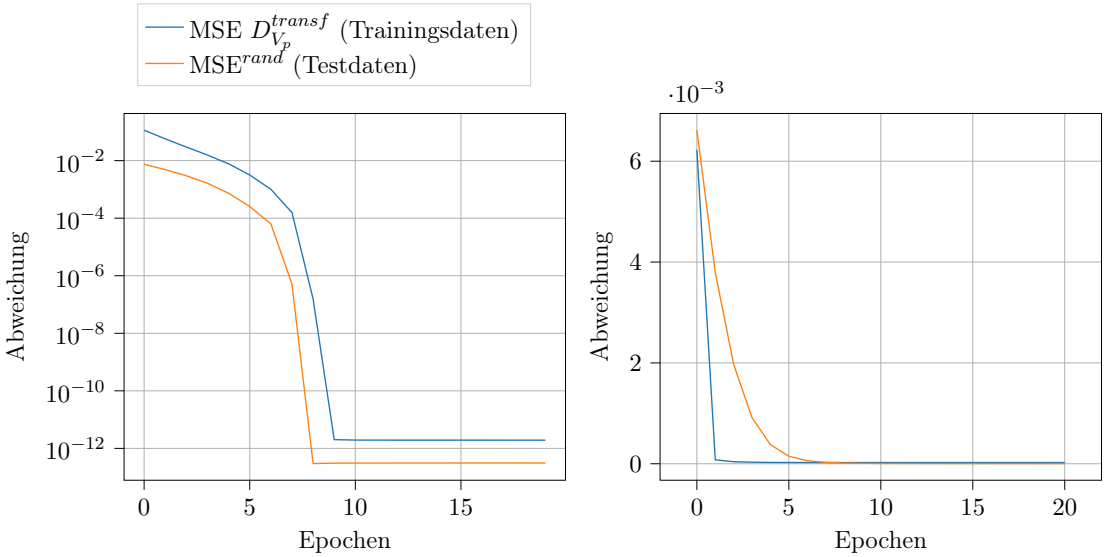


Abbildung 4.23: Fehlerverlauf für das anschließende Training der Modelle mit dem zusätzlich erzeugten Trainingsdatensatz $D_{V_p}^{transf}$ Links: Weitertraining des ML-Modells $ML_{opt,CMA(\sigma_0=0.0001)}^{patch,36}$, Rechts: Weitertraining des ML-Modells mit 384 trainierbaren Parametern anhand des Patch-Versuches und mit der CMA-Optimierung

Dass mit diesem abschließenden Trainingsschritt ein allgemeingültiges ML-Materialmodell erzielt wurde, ist für das ML-Modell $ML_{opt,CMA(\sigma_0=0.0001),rot}^{patch,36}$ mit 36 trainierbaren Parametern auch an den Gewichten zu sehen

$$C_{ML_{opt,CMA(\sigma_0=0.0001),rot}^{patch,36}} = \begin{pmatrix} 282.69 & 121.15 & 121.15 & 0 & 0 & 0 \\ 121.15 & 282.69 & 121.15 & 0 & 0 & 0 \\ 121.15 & 121.15 & 282.69 & 0 & 0 & 0 \\ 0 & 0 & & 80.77 & 0 & 0 \\ 0 & 0 & 0 & 0 & 80.77 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80.77 \end{pmatrix}. \quad (4.27)$$

Fazit

Die Validierung zeigt anhand eines linear elastischen Materialmodells, dass auch ohne die lokalen Spannungswerte, welche in Charakterisierungsversuchen nicht messbar sind, ein ML-Materialmodell trainiert werden kann. Anhand alternativer Fehlerfunktionen können die im Versuch gesehenen Spannungswerte erlernt werden, ohne dass diese für das Training

benutzt werden. Hierfür können unterschiedliche Optimierungsansätze zielführend sein. Für den einfachen, rechteckigen Patch-Ausschnitt eines Zugversuchs bei einer Optimierung der Skalarisierung als gewichtete Summe ermöglicht ein gradientenbasierter Optimierungsansatz durch eine angepasste Berechnung der Gradienten (vorausgesetzt einer geeigneten Wahl der Lernrate) eine schnelle Anpassung der Modelle. Jedoch kann durch einen Optimierungsansatz mittels eines gradientenfreien, randomisierten Optimierungsalgorithmus durch Covariance Matrix Adaptation für eine erweiterte, etwas komplexere Zugversuchsprobe ein besseres Ergebnis erzielt und damit im Unterschied zum gradientenbasierten Ansatz der Referenz-Zielfunktionswert innerhalb der betrachteten Iterationen erreicht werden.
Da das erhaltene, optimierte ML-Materialmodell im Falle der Skalarisierung von der Wahl der Gewichtung der Zielfunktionen abhängt, wurden sowohl unterschiedliche Gewichtungen als auch ein mehrkriterieller Optimierungsansatz für das Training untersucht. Hierbei wird ersichtlich, dass bei beiden betrachteten Versuchen zum einen alle Fehlerfunktionen (abgesehen von den Fehlerfunktionen für die Divergenzen im Spezialfall der rechteckigen Patch-Probe) für das Training benötigt werden und dass weiterhin die Skalarisierung als gewichtete Summe mit einer gleichen Gewichtung aller Zielfunktionen einen gut geeigneten Optimierungsansatz für das Training darstellt.
Während die Spannungen, welche auf den im Versuch enthaltenen Dehnungen beruhen, vom optimierten ML-Materialmodell im Training sehr gut erlernt werden, reichen die betrachteten, einfachen Zugversuchsproben alleine nicht aus, um ein allgemeingültiges ML-Materialmodell zu erhalten, welches auch nicht im Training gesehene Spannungen vorhersagen kann. Für das hier betrachtete linear elastische Materialmodell kann eine verbesserte Verallgemeinerungsfähigkeit der ML-Materialmodelle durch zusätzliche Transformationen der im Versuch berücksichtigten Daten aus dem Training erreicht werden.

4.5 Anwendung der Methode für ein Materialmodell mit Plastizität

Die Methode für das Training der ML-Materialmodelle basierend auf Versuchen wird in diesem Abschnitt angewendet auf das in Abschnitt 3.2 betrachtete elastoplastische Materialmodell und rekurrenter neuronaler Netze, anhand derer das Materialmodell modelliert wird. Als Startmodell dient ein ML-Materialmodell, welches für die Materialparameter eines DP600 Stahls trainiert wurde. Die beiden Stähle DP600 und DP800 unterscheiden sich bezüglich ihrer Mindestzugfestigkeit von 600 beziehungsweise 800 Megapascal. Betrachtet

man die Parameter der passenden, klassischen Materialmodelle für die verschiedenen Werkstoffe, so weichen diese hinsichtlich des Elastizitätsmoduls, welches für den DP600 Stahl 190 und für den DP800 Stahl 210 beträgt, voneinander ab. Weiterhin variieren die Fließkurven der verschiedenen Werkstoffe, wie in Abbildung 4.24 zu erkennen ist.

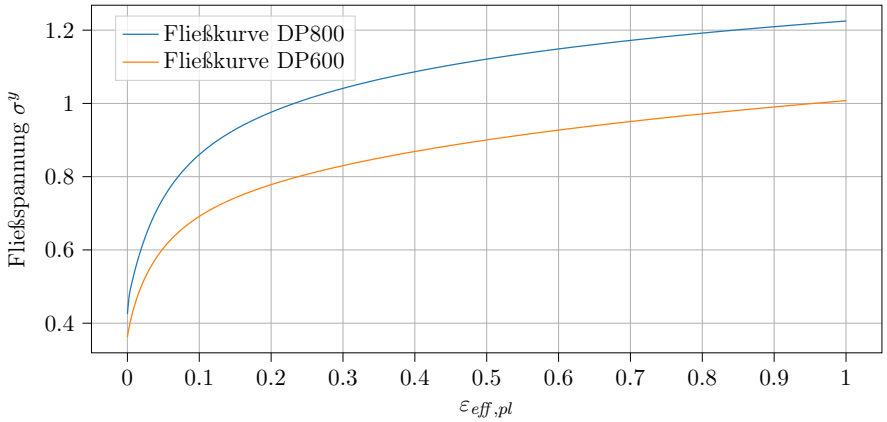


Abbildung 4.24: Die Fließkurven für das Start-Materialmodell DP600 sowie des Zielmodells DP800

Da für das LMSC-Modell ML^{LMSC} mit 8710 trainierbaren Parametern deutlich weniger Variablen in der Optimierung als bei dem LSTM Modell mit 805604 trainierbaren Parametern zu berücksichtigen sind und dieses Modell zudem in den Demonstratorsimulationen in Abschnitt 3.2.7 in den meisten Fällen am besten abgeschnitten hat, wird diese Netzarchitektur für das Training ausgewählt.
Zunächst wird in Abschnitt 4.5.1 für die Anpassung des auf den Stahl DP600 vortrainierten ML-Materialmodells auf die Parameter des DP800 Stahls der bereits betrachtete Patch-Ausschnitt eines simulierten Zugversuchs genutzt. Für das Erreichen einer besseren Verallgemeinerungsfähigkeit des ML-Materialmodells wird anschließend in Abschnitt 4.5.2 das Training anhand einer neu entwickelten Zugprobe aus [39] betrachtet. Abschließend wird in Abschnitt 4.5.3 der Trainingsdatensatz um weitere Versuchsproben erweitert und das Training anhand einer Kombination unterschiedlicher Versuche betrachtet mit dem Ziel eine verbesserte Verallgemeinerungsfähigkeit des ML-Materialmodells zu erreichen.
Da mit dem Materialmodell `*MAT_024` eine Schalenformulierung betrachtet wird, muss wie in Abschnitt 4.2.2 näher erläutert die Zielfunktion L_z nicht berücksichtigt werden. Der Gewichtsvektor für die Skalarisierung $L_{total}(w)$ wird daher im Folgenden durch $w = (w_f, w_{d,x}, w_{d,y}, w_b)$ ersetzt.

4.5.1 Training anhand des Patch Ausschnitts eines Zugversuches

Als Versuchsprobe für das Training wird im Folgenden der Patch-Ausschnitt eines Zugversuchs wie in Abschnitt 4.4.1 (siehe Abbildung 4.7) betrachtet. Der Aufbau des simulierten Versuchs bleibt dabei bis auf das genutzte Materialmodell, mit welchem die Simulation durchgeführt wird, unverändert. Das zuvor genutzte Materialmodell `*MAT_ELASTIC` wird in der Simulation in diesem Abschnitt durch das Modell `*MAT_024` mit den Materialparametern für den DP800 Stahl ersetzt. Das in Abschnitt 3.2.4 trainierte LMSC-Modell für den DP800 Stahl dient als Ziel-ML-Materialmodell für den Vergleich und die Auswertung des in den Optimierungen erhaltenen ML-Modells und wird in diesem Kapitel als ML_{DP800}^{LMSC} bezeichnet. Das für den DP600 Stahl vortrainierte ML-Modell bestehend aus der gleichen Netzarchitektur wird mit ML_{DP600}^{LMSC} bezeichnet.
Abbildung 4.25 zeigt auf der linken Seite einen Optimierungsverlauf für die Anpassung der Gewichte des LMSC-Modells ML_{DP600}^{LMSC} anhand des CMA-Algorithmus bei einer gleichen Gewichtung aller vier Zielfunktionen.

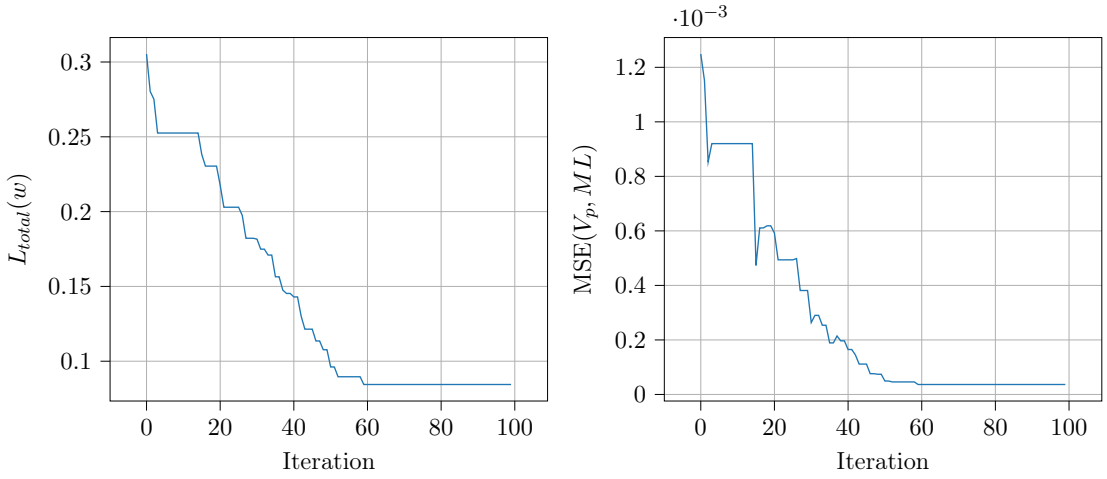


Abbildung 4.25: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ bei einer Gewichtung $w = (1, 1, 1, 1)$ über die Iterationen bei einer Optimierung mittels CMA bei $\sigma_0 = 0.0001$, Rechts: Verlauf der mittleren quadratischen Abweichung MSE(V_p, ML) zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen des simulierten Versuchs

Auch hier sind die Ergebnisse jenes CMA-Verlaufs zu sehen, welcher dem Mittelwert von 20 verschiedenen Verläufen am nächsten liegt (gemessen an der mittleren quadratischen Abweichung der erreichten Zielfunktionswerte im Laufe der Iterationen). Die gesamte Varianz der verschiedenen Verläufe liegt bei $4.009 \cdot 10^{-3}$. Wird nur der nach 100 Iterationen erreichte Zielfunktionswert der verschiedenen Optimierungsläufe berücksichtigt, so liegt

die Varianz lediglich bei $9.776 \cdot 10^{-5}$ mit einem Mittelwert von $7.998 \cdot 10^{-2}$. In jedem der 20 Verläufe wird nach den berücksichtigten 100 Iterationen der Gesamtfehler $L_{total} = 1.455 \cdot 10^{-1}$ des Ziel-Modells ML_{DP800}^{LMSC} übertroffen und liegt je nach Optimierungslauf zwischen $6.540 \cdot 10^{-2}$ und $9.917 \cdot 10^{-2}$. Für den in Abbildung 4.25 betrachteten Verlauf wird der Gesamtfehler des Ziel-Modells ML_{DP800}^{LMSC} bereits nach 38 Iterationen in der Optimierung erreicht. Abbildung 4.25 zeigt auf der rechten Seite zudem, dass auch für dieses neuronale Netz mit rekurrenten Verbindungen und deutlich mehr trainierbaren Parametern (als die bisher betrachteten 36 beziehungsweise 384) die Spannungs-Dehnungsbeziehung des komplexeren elastoplastischen Materialmodells erlernt wird. Die Fehlerwerte der einzelnen Fehlerfunktionen des optimierten Modells (bezeichnet durch $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,patch}$) sowie des Ziel- und Startmodells und die genauen Werte für die Abweichung $\text{MSE}(V_p, ML)$ finden sich in einer Übersicht in Tabelle 4.10.

	$L_{total}(w)$ mit $w = (1,1,1,1,1)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_{bnd}	$\text{MSE}(V_p, ML)$
`*MAT_024`	$1.863 \cdot 10^{-4}$	$3.727 \cdot 10^{-5}$	$1.209 \cdot 10^{-4}$	$1.574 \cdot 10^{-5}$	$1.243 \cdot 10^{-5}$	-
ML_{DP600}^{LMSC}	$3.053 \cdot 10^{-1}$	$1.679 \cdot 10^{-1}$	$1.025 \cdot 10^{-4}$	$1.538 \cdot 10^{-5}$	$1.373 \cdot 10^{-1}$	$1.249 \cdot 10^{-3}$
ML_{DP800}^{LMSC}	$1.455 \cdot 10^{-1}$	$1.576 \cdot 10^{-2}$	$1.220 \cdot 10^{-4}$	$1.794 \cdot 10^{-5}$	$1.296 \cdot 10^{-1}$	$3.040 \cdot 10^{-5}$
$ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,patch}$	$8.444 \cdot 10^{-2}$	$2.435 \cdot 10^{-2}$	$1.221 \cdot 10^{-4}$	$1.526 \cdot 10^{-5}$	$5.996 \cdot 10^{-2}$	$3.654 \cdot 10^{-5}$
$ML_{opt,multi,worst}^{LMSC,DP800,patch}$	$2.055 \cdot 10^{-2}$	$6.159 \cdot 10^{-3}$	$1.134 \cdot 10^{-4}$	$1.033 \cdot 10^{-5}$	$1.427 \cdot 10^{-2}$	$8.403 \cdot 10^{-6}$
$ML_{opt,multi,best}^{LMSC,DP800,patch}$	$1.962 \cdot 10^{-2}$	$3.966 \cdot 10^{-3}$	$1.131 \cdot 10^{-4}$	$1.005 \cdot 10^{-5}$	$1.553 \cdot 10^{-2}$	$2.381 \cdot 10^{-6}$

Tabelle 4.10: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ bei einer gleichen Gewichtung $w = (1,1,1,1,1)$ und der Abweichung $\text{MSE}(V_p, ML)$ für die unterschiedlich optimierten Modelle sowie des vortrainierten Startmodells ML_{DP600}^{LMSC}, des direkt auf Spannungs-Dehnungsdaten trainierten Zielmodells ML_{DP800}^{LMSC} und für die Spannungen des klassischen Materialmodells `*MAT_024`

Da für das hier betrachtete Materialmodell die Zielfunktion L_z nicht berücksichtigt werden muss und für den Patch-Ausschnitt, wie in Abschnitt 4.4.1 näher erläutert wird, die Fehlerfunktionen für die Divergenzen $L_{div,x}$ und $L_{div,y}$ nicht relevant sind, kann das Training für diesen Anwendungsfall als bikriterielles Optimierungsproblem mit den beiden Zielfunktionen L_{force} und L_{bnd} betrachtet werden. Abbildung 4.26 zeigt die Zielfunktionswerte der beiden Fehlerfunktionen für die finalen Lösungen nach 300 Iterationen des NSGA-II. Die Position eines Punktes in der Abbildung 4.26, welcher jeweils zu einer Lösung (ML-Modell) gehört, wird durch die Zielfunktionswerte L_{force} sowie L_{bnd} festgelegt und die Größe des Durchmessers eines Punktes orientiert sich (logarithmisch) an der (wie auch bisher nicht in der Optimierung berücksichtigten) Abweichung $\text{MSE}(V_p, ML)$. Je kleiner

der Durchmesser, desto genauer ist die Vorhersage des ML-Materialmodells bezüglich der im Versuch gesehenen Daten. Man sieht, dass alle finalen ML-Modelle das direkt trainierte Zielmodell dominieren, dieses auch bezüglich des MSE(V_p, ML) übertreffen und dementsprechend eine geringere Abweichung aufweisen. Die beiden Modelle mit dem größten und kleinsten MSE(V_p, ML) innerhalb der finalen Population werden mit $ML_{opt,multi,worst}^{LMSC,DP800,patch}$ und $ML_{opt,multi,best}^{LMSC,DP800,patch}$ bezeichnet, unterscheiden sich allerdings kaum in ihrer Prognosefähigkeit bezüglich der Trainingsdaten und übertreffen auch den MSE(V_p, ML)-Wert des ML-Modells $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,patch}$, welches anhand der Skalarisierung $L_{total}(w)$ mit $w = (1, 1, 1, 1)$ optimiert wurde. Die einzelnen Fehlerfunktionswerte und die Abweichungen MSE(V_p, ML) der unterschiedliche optimierten ML-Modelle finden sich in Tabelle 4.10.

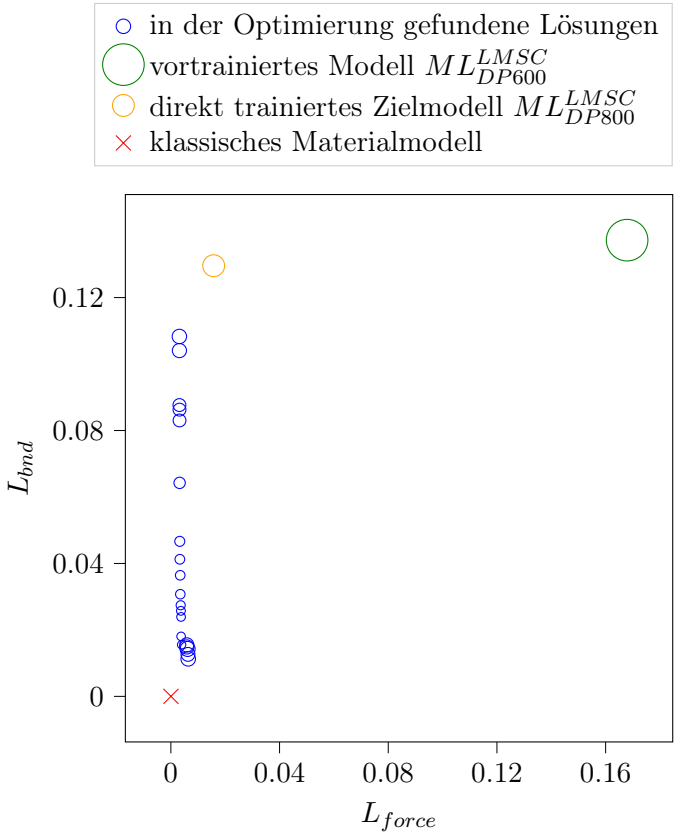


Abbildung 4.26: Fehlerfunktionswerte für die finalen Lösungen des NSGA-II bei einer bikriteriellen Optimierung anhand der Patch-Simulation

Wie zu erwarten, erlernt das ML-Modell lediglich die Daten des Versuchs, nicht jedoch zufällig erzeugte Dehnungs-Spannungspaare, wie es analog beim linear elastischen Material-

modell in Abschnitt 4.4.1 erkennbar war. Abbildung 4.27 zeigt den Verlauf $\text{MSE}^{rel}(V_p, ML)$ der im Versuch enthaltenen Daten und der Abweichung $\text{MSE}^{rand,rel}$ bezüglich des bereits in Kapitel 3 betrachteten Testdatensatzes bestehend aus zufällig generierten Dehnpfaden und den zugehörigen Spannungsverläufen (auch hier jeweils normiert durch die jeweilige Abweichung des Startmodells ML_{DP600}^{LMSC}).

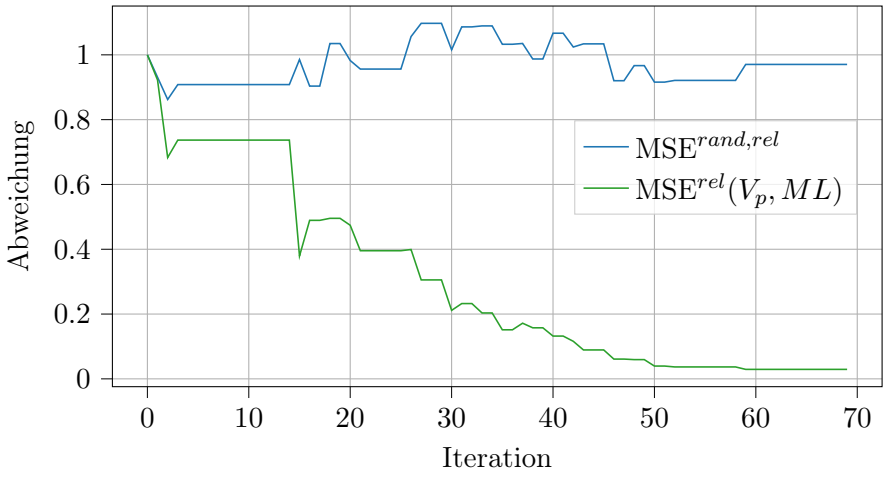


Abbildung 4.27: Fehlerverläufe für $\text{MSE}^{rel}(V_p, ML)$ und $\text{MSE}^{rand,rel}$ für die Optimierung von $L_{total}(w)$ mit $w = (1, 1, 1, 1)$ anhand des Startmodells ML_{DP600}^{LMSC} mittels CMA für $\sigma_0 = 0.0001$

4.5.2 Training basierend auf neu entwickelter Zugversuchsprobe

Da der rechteckige Patch Ausschnitt eines Zugversuchs nur einen sehr kleinen Datensatz darstellt und darüber hinaus aus sehr ähnlichen Datenpunkten (Dehnungs-Spannungsverläufen) besteht, welche ausschließlich im Zugbereich liegen (siehe Abbildung 4.28 links), wird das Training basierend auf einer neu konstruierten Zugversuchsprobe aus [39] (im Folgenden auch als Lochzugprobe bezeichnet) mit zusätzlichen Aussparungen (Löchern) im inneren Bereich der Geometrie betrachtet. Die Versuchsprobe wurde mit dem Ziel entwickelt einen vielfältigeren Datensatz für das Training der ML-Modelle basierend auf Versuchsdaten bereitzustellen. Der Aufbau der Probe sowie die von-Mises-Spannungen des letzten Zeitschrittes sind in Abbildung 4.29 zu sehen. Konzipiert wurde die Probe in Hinblick auf einer möglichst großen Abdeckung der Triaxialität (insbesondere auch im Druckbereich) und der plastischen Dehnung (wie es bereits in Abschnitt 3.2.3 betrachtet wurde), wobei für die Entwicklung der Versuchsgeometrie die genaue Form, Anzahl und Größe der Aussparungen innerhalb der Probe als Variablen im Rahmen einer Optimierung betrachtet wurden.

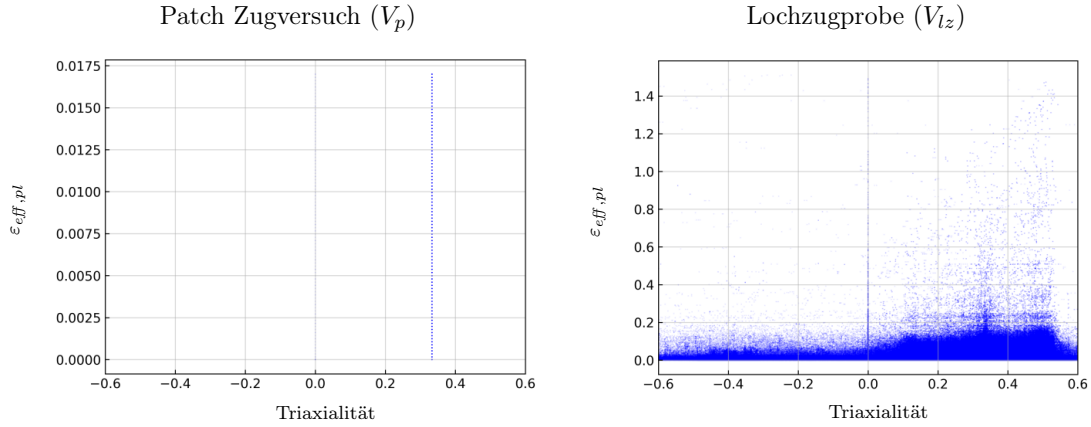


Abbildung 4.28: Abdeckung der Spannungszustände gemessen an der Triaxialität (siehe (3.23)) und der plastischen Dehnung $\varepsilon_{eff,pl}$ für den Patch Zugversuch (links) und der Lochzugprobe (rechts) im Vergleich

Bei dieser Optimierung zum Finden einer geeigneten Probengeometrie wurden, wie auch in [58] erläutert wird, hauptsächlich symmetrische Versuchsproben betrachtet, da ansonsten an der Einspannung der Probe horizontale Kräfte entstehen, welche von einem Großteil der Versuchsanlagen nicht erfasst werden können.

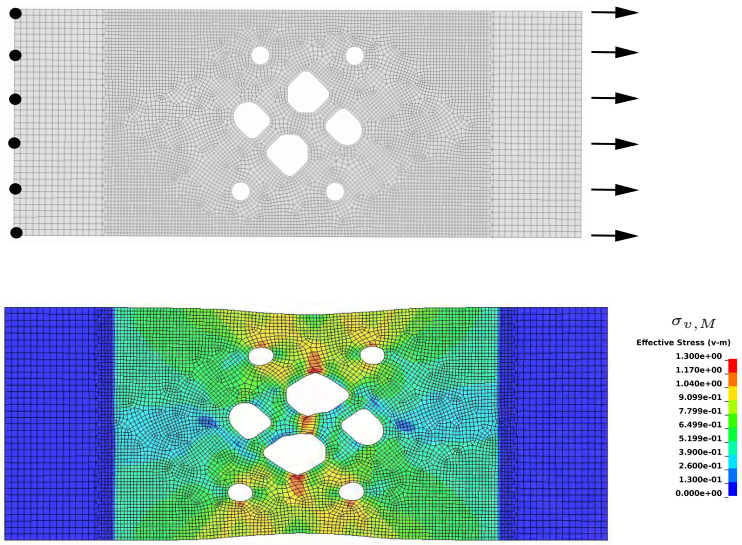


Abbildung 4.29: Neuartige Versuchsprobe (Lochzugprobe) für das Training von ML-Materialmodellen basierend auf Versuchsdaten, Oben: Aufbau des simulierten Versuchs, Unten: Verteilung der von-Mises-Spannungen $\sigma_{v,M}$ auf der Geometrie im letzten Zeitschritt des simulierten Versuchs

Wie auch bei den bisher berücksichtigten Zugversuchen werden bei der hier betrachteten Versuchsprobe die Knoten an einem Rand der Geometrie festgehalten und eine Verschiebung auf die gegenüberliegenden Randknoten aufgebracht (siehe Abbildung 4.29). Dass diese Zugversuchsprobe einen deutlich größeren Bereich der plastischen Dehnung $\varepsilon_{eff,pl}$ sowie der Triaxialität abdeckt und insbesondere Spannungszustände im Druckbereich enthält, was durch die Aussparungen innerhalb der Probe erreicht wird, zeigt die Abbildung 4.28.

Auf der linken Seite von Abbildung 4.30 ist der Optimierungsverlauf für das Training anhand der Lochzugprobe mittels der Skalarisierung $L_{total}(w)$ bei einer gleichen Gewichtung aller vier Zielfunktionen durch den CMA-Algorithmus zu sehen. Wie bisher sind auch hier die Ergebnisse des Optimierungsverlaufs zu sehen, welcher dem Mittelwert von allen durchgeführten Verläufen am nächsten liegt. Die Varianz aller durchgeführten Optimierungsläufe liegt bei diesem Versuch unter Berücksichtigung des gesamten Verlaufs bei $1.787 \cdot 10^{-1}$ und bei Berücksichtigung des nach 100 Iterationen erreichten Zielfunktionswerts lediglich bei $2.501 \cdot 10^{-5}$ mit einem Mittelwert von $2.610 \cdot 10^{-1}$. Nach 100 Iterationen verringert sich beim betrachteten Optimierungslauf der Gesamtfehler L_{total} von 6.11 des Startmodells ML_{DP600}^{LMSC} auf $L_{total} = 0.26$. Die Abweichung $\text{MSE}(V_{lz}, ML)$ zwischen den korrekten und den vorhergesagten Spannungen des optimierten ML-Modells nimmt dabei von $1.510 \cdot 10^{-3}$ ab auf $2.577 \cdot 10^{-4}$, wie auch der Verlauf auf der rechten Seite von Abbildung 4.30 zeigt.

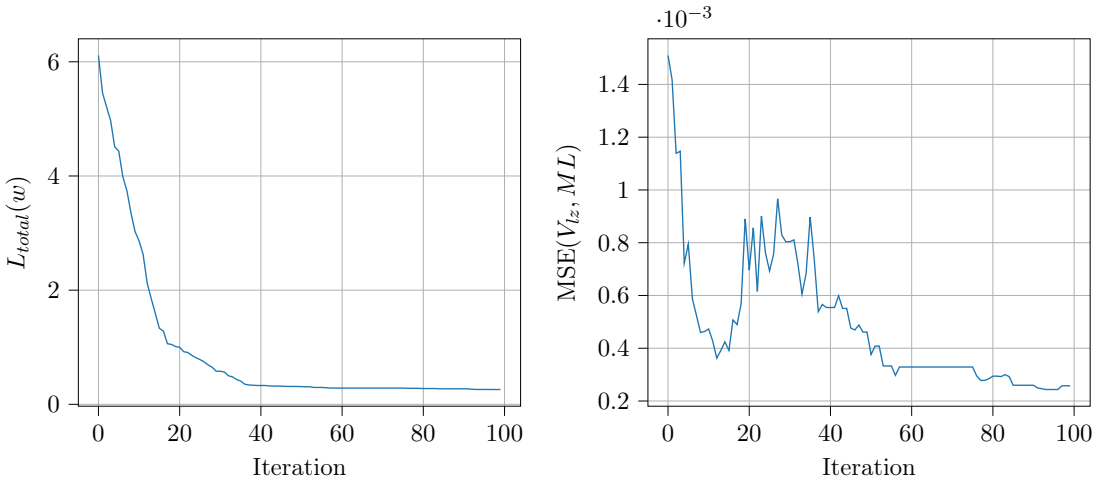


Abbildung 4.30: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ bei einer Gewichtung $w = (1, 1, 1, 1)$ über die Iterationen bei einer Optimierung mittels CMA bei $\sigma_0 = 0.0001$, Rechts: Verlauf der mittleren quadratischen Abweichung $\text{MSE}(V_{lz}, ML)$ zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen des simulierten Versuchs

Tabelle 4.11 zeigt die Ergebnisse von Optimierungen, bei denen jeweils die Fehlerfunktionen für die Divergenzen $L_{div,x}$, $L_{div,y}$, die Fehlerfunktionen für die globale Kraft L_{force} und für den kraftfreien Rand L_{bnd} deutlich höher (zehnfach) gewichtet werden. Wie auch in den bisherigen Anwendungsfällen sind die besten Ergebnisse (bezüglich der geringsten MSE(V_{lz}, ML)-Werte) bei einer gleichen Gewichtung aller Zielfunktionen zu sehen, wobei sich jedoch die MSE(V_{lz}, ML)-Werte in jedem Fall gegenüber dem Startmodell ML_{DP600}^{LMSC} verbessern und die Ergebnisse für diesen Anwendungsfall damit recht robust bezüglich der unterschiedlichen Gewichtungen sind.

Gewichtung $(w_f, w_{d,x}, w_{d,y}, w_b)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_{bnd}	$L_{total}(w)$	$L^{\star}_{total}(w)$	$\frac{L^{\star}_{total}(w)}{L_{total}(w)}$	MSE(V_{lz}, ML)
(1,1,1,1)	$1.220 \cdot 10^{-1}$	$2.161 \cdot 10^{-2}$	$9.185 \cdot 10^{-3}$	$1.069 \cdot 10^{-1}$	0.260	0.325	1.250	$2.577 \cdot 10^{-4}$
(10,1,1,1)	$2.620 \cdot 10^{-2}$	$2.218 \cdot 10^{-2}$	$9.340 \cdot 10^{-3}$	$1.244 \cdot 10^{-1}$	0.418	0.419	1.002	$5.456 \cdot 10^{-4}$
(1,1,1,10)	$6.488 \cdot 10^{-2}$	$2.244 \cdot 10^{-2}$	$1.076 \cdot 10^{-2}$	$4.346 \cdot 10^{-2}$	0.533	0.510	0.958	$3.998 \cdot 10^{-4}$
(1,10,10,1)	$1.922 \cdot 10^{-0}$	$1.967 \cdot 10^{-2}$	$9.065 \cdot 10^{-1}$	$1.095 \cdot 10^{-1}$	2.319	2.973	1.239	$3.306 \cdot 10^{-4}$

Tabelle 4.11: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ und der Abweichung MSE(V_{lz}, ML) der optimierten Modelle bei einer Optimierung der Skalarisierung mit unterschiedlichen Gewichtungen der verschiedenen Fehlerfunktionen

Die Eignung der Optimierung durch die Skalarisierung $L_{total}(w)$ mit einer gleichen Gewichtung $w = (1, 1, 1, 1)$ aller Zielfunktionen bestätigt sich auch bei der Betrachtung der Ergebnisse eines mehrkriteriellen Optimierungsansatzes. Die unterschiedlichen Fehlerfunktionswerte von Lösungen der finalen Population eines NSGA-III Optimierungslaufs nach 300 Iterationen sind in Abbildung 4.31 und Tabelle 4.12 zu sehen. Tabelle 4.12 zeigt, sortiert nach der Prognosefähigkeit der ML-Modelle gemessen am MSE(V_{lz}, ML) und beginnend mit dem besten Modell $ML_{opt,multi,best}^{LMSC,DP800,lz}$ bis hin zum Modell $ML_{opt,multi,worst}^{LMSC,DP800,lz}$, welches die höchste Abweichung MSE(V_{lz}, ML) aufweist, die Fehlerfunktionswerte sowie die Abweichung MSE(V_{lz}, ML) der finalen ML-Modelle aus dem mehrkriteriellen Optimierungslauf. Wie es auch bereits in den Abschnitten 4.4.1 und 4.4.2 für das linear elastische Materialmodell erkennbar war, ist ein starker Zusammenhang zwischen einem niedrigen $L_{total}(w)$-Wert mit $w = (1, 1, 1, 1)$ und einer geringen Abweichung MSE(V_{lz}, ML) zu erkennen. Geringere Werte in den Zielfunktionskomponenten $L_{div,x}, L_{div,y}$ und L_{bnd} bei den nicht-dominierten, finalen Lösungen des Optimierungslaufs gehen (teilweise auch sehr deutlich) zulasten eines höheren L_{force}-Fehlerwertes verbunden mit einer höheren MSE(V_{lz}, ML)-Abweichung.

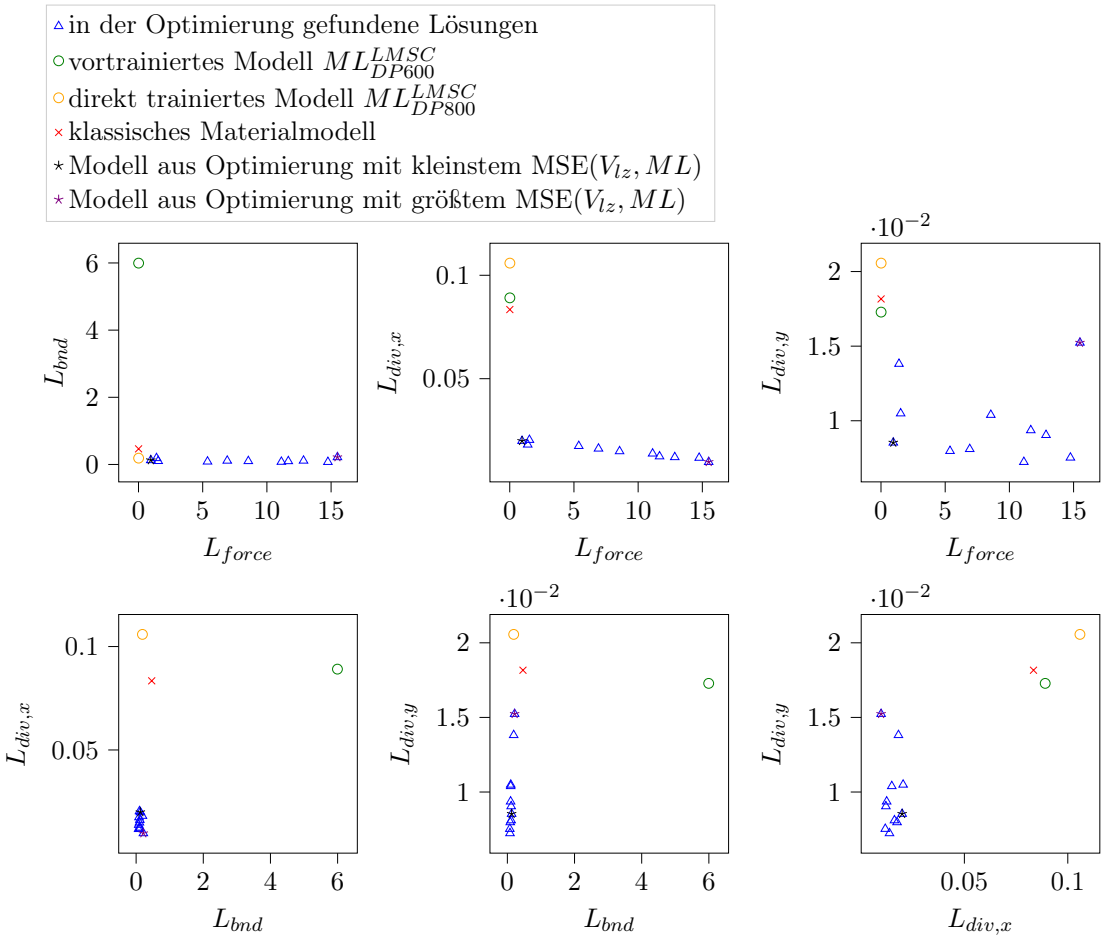


Abbildung 4.31: Fehlerfunktionswerte für die finalen Lösungen des NSGA-III beim Training des ML-Materialmodells anhand der Lochzugprobe

Die erhoffte Verbesserung eines ML-Modells, welches anhand der neuartigen Lochzugprobe trainiert wurde, ist zumindest tendenziell erkennbar. Abbildung 4.32 zeigt den Verlauf $\text{MSE}^{rel}(V_{lz}, ML)$ der im Versuch enthaltenen Daten und der normierten Abweichung $\text{MSE}^{rand,rel}$ bezüglich des bisher betrachteten Testdatensatzes bestehend aus zufällig generierten Dehnpfaden und den zugehörigen Spannungsverläufen. Links in der Abbildung ist der Vergleich zwischen der normierten Abweichung der ungesehenen Testdaten $\text{MSE}^{rand,rel}$ und den im Versuch gesehenen Daten $\text{MSE}^{rel}(V_{lz}, ML)$ zu sehen. Abbildung 4.32 zeigt auf der rechten Seite den Vergleich des Rückgangs der Abweichung der Testdaten $\text{MSE}^{rand,rel}$ für das Training anhand des Patch Versuchs und der in diesem Abschnitt betrachteten Lochzugprobe (jeweils bei einer Optimierung mit einer gleichen Gewichtung der Fehlerfunktionen). Für die Optimierung mittels der Lochzugprobe V_{lz} nimmt die Ab-

weichung deutlicher ab, als bei einer Optimierung mit dem einfachen Patch Zugversuch V_p. Nach wie vor ist allerdings auch bei der hier betrachteten Lochzugprobe der Rückgang des $\text{MSE}^{rand,rel}$ im Laufe der Optimierung noch geringer als die Abweichung $\text{MSE}^{rel}(V_{lz}, ML)$ für die im Versuch gesehenen Daten. Aus diesem Grund wird im folgenden Abschnitt das Training anhand mehrerer, kombinierter Versuche betrachtet.

	L_{force}	$L_{div,x}$	$L_{div,y}$	L_{bnd}	$L_{total}(w)$ mit $w = (1,1,1,1)$	$\text{MSE}(V_{lz}, ML)$
$ML^{LMSC,DP800,lz}_{opt,multi,best}$	0.957	$1.991 \cdot 10^{-2}$	$8.535 \cdot 10^{-3}$	$1.269 \cdot 10^{-1}$	1.113	$7.725 \cdot 10^{-4}$
	1.533	$2.039 \cdot 10^{-2}$	$1.049 \cdot 10^{-2}$	$1.014 \cdot 10^{-1}$	1.665	$9.128 \cdot 10^{-4}$
	5.376	$1.745 \cdot 10^{-2}$	$7.973 \cdot 10^{-3}$	$8.573 \cdot 10^{-2}$	5.487	$2.244 \cdot 10^{-3}$
	6.912	$1.619 \cdot 10^{-2}$	$8.101 \cdot 10^{-3}$	$1.118 \cdot 10^{-1}$	7.048	$2.781 \cdot 10^{-3}$
	1.402	$1.813 \cdot 10^{-2}$	$1.381 \cdot 10^{-2}$	$1.881 \cdot 10^{-1}$	1.622	$3.432 \cdot 10^{-3}$
	8.554	$1.491 \cdot 10^{-2}$	$1.039 \cdot 10^{-2}$	$9.909 \cdot 10^{-2}$	8.679	$6.006 \cdot 10^{-3}$
	11.127	$1.380 \cdot 10^{-2}$	$7.237 \cdot 10^{-3}$	$7.841 \cdot 10^{-2}$	11.226	$6.957 \cdot 10^{-3}$
	11.668	$1.244 \cdot 10^{-2}$	$9.362 \cdot 10^{-3}$	$9.681 \cdot 10^{-2}$	11.787	$9.452 \cdot 10^{-3}$
	14.753	$1.175 \cdot 10^{-2}$	$7.528 \cdot 10^{-3}$	$7.363 \cdot 10^{-2}$	14.846	$1.064 \cdot 10^{-2}$
	12.854	$1.205 \cdot 10^{-2}$	$9.042 \cdot 10^{-3}$	$1.126 \cdot 10^{-1}$	12.987	$1.119 \cdot 10^{-2}$
$ML^{LMSC,DP800,lz}_{opt,multi,worst}$	15.495	$9.733 \cdot 10^{-3}$	$1.523 \cdot 10^{-2}$	$2.160 \cdot 10^{-1}$	15.736	$2.298 \cdot 10^{-2}$

Tabelle 4.12: Funktionswerte der einzelnen Fehlerfunktionskomponenten, des aufsummierten Gesamtfehlers $L_{total}(w)$ für $w = (1,1,1,1)$ sowie die $\text{MSE}(V_{lz}, ML)$-Werte für die finalen Lösungen der mehrkriteriellen Optimierung

	$L_{total}(w)$ mit $w = (1,1,1,1,1)$	L_{force}	$L_{div,x}$	$L_{div,y}$	L_{bnd}	$\text{MSE}(V_{lz}, ML)$
`*MAT_024`	0.574	$7.312 \cdot 10^{-3}$	$4.647 \cdot 10^{-1}$	$8.342 \cdot 10^{-2}$	$1.816 \cdot 10^{-2}$	-
ML^{LMSC}_{DP600}	6.110	$8.836 \cdot 10^{-3}$	$5.995 \cdot 10^{0}$	$8.908 \cdot 10^{-2}$	$1.728 \cdot 10^{-2}$	$1.510 \cdot 10^{-3}$
ML^{LMSC}_{DP800}	0.325	$1.040 \cdot 10^{-2}$	$1.883 \cdot 10^{-1}$	$1.059 \cdot 10^{-1}$	$2.056 \cdot 10^{-2}$	$8.534 \cdot 10^{-5}$
$ML^{LMSC,DP800,lz}_{opt,CMA(\sigma_0=0.0001)}$	0.260	$1.220 \cdot 10^{-1}$	$2.161 \cdot 10^{-2}$	$9.185 \cdot 10^{-3}$	$1.069 \cdot 10^{-1}$	$2.577 \cdot 10^{-4}$
$ML^{LMSC,DP800,lz}_{opt,multi,worst}$	15.736	15.495	$9.733 \cdot 10^{-3}$	$1.523 \cdot 10^{-2}$	$2.160 \cdot 10^{-1}$	$2.298 \cdot 10^{-2}$
$ML^{LMSC,DP800,lz}_{opt,multi,best}$	1.665	$9.572 \cdot 10^{-1}$	$1.991 \cdot 10^{-2}$	$8.535 \cdot 10^{-3}$	$1.269 \cdot 10^{-1}$	$9.128 \cdot 10^{-4}$

Tabelle 4.13: Funktionswerte der unterschiedlichen Fehlerfunktionen, des Gesamtfehlers $L_{total}(w)$ bei einer gleichen Gewichtung $w = (1,1,1,1)$ und der Abweichung $\text{MSE}(V_{lz}, ML)$ für die unterschiedlich optimierten Modelle sowie des vortrainierten Startmodells ML^{LMSC}_{DP600}, des direkt auf Spannungs-Dehnungsdaten trainierten Zielmodells ML^{LMSC}_{DP800} und für die Spannungen des klassischen Materialmodells `*MAT_024`

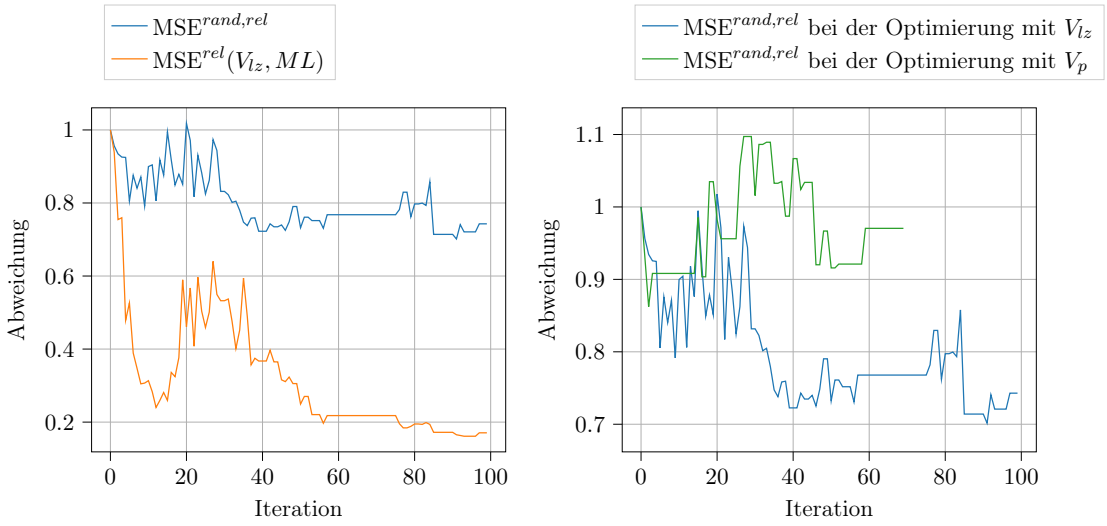


Abbildung 4.32: Fehlerverläufe für MSE^{rand,rel} Links: Vergleich der Verläufe für MSE^{rand,rel} und MSE$^{rel}(V_{lz}, ML)$ bei der Optimierung von $L_{total}(w)$ mit $w = (1, 1, 1, 1)$ mittels CMA für $\sigma_0 = 0.001$ anhand der Lochzugprobe V_{lz}, Rechts: Vergleich der Verläufe von MSE^{rand,rel} für die Optimierung von $L_{total}(w)$ mit $w = (1, 1, 1, 1)$ anhand des Patch-Zugversuchs V_p und der Lochzugprobe V_{lz} im direkten Vergleich

4.5.3 Verbesserung der Verallgemeinerungsfähigkeit durch die Kombination mehrerer Versuchsproben

Für eine verbesserte Verallgemeinerungsfähigkeit des trainierten ML-Materialmodells werden in diesem Abschnitt sieben unterschiedliche Versuche im Training kombiniert, um einen größeren und vielfältigeren Trainingsdatensatz zu erhalten. Die Lochzugprobe aus dem vorherigen Abschnitt wird weiterhin betrachtet, wobei zusätzlich zu der Zugbelastung die gleiche Probengeometrie unter Druck und als Scherversuch betrachtet wird, wie es in Abbildung 4.34 ersichtlich ist. Die Menge aller Dehnungen des Zugversuchs werden mit V_{lz}, die des Druckversuchs mit V_{ld} und die des Scherversuchs mit V_{ls} bezeichnet. Zusätzlich werden vier verschiedene Standard Gissmo-Proben (siehe [33]) ergänzt, welche in Abbildung 4.33 zu sehen sind. Bei allen vier Versuchsproben handelt es sich um Zugversuche, bei denen für den Simulationsaufbau die Randpunkte auf einer Seite festgehalten werden, während für die Punkte auf der gegenüberliegenden Seite eine Verschiebung vorgegeben wird. Die Menge der Dehnungen wird für den Kerbzug mit V_{kez}, für den Miniflachzug mit V_{gmfz} und für die beiden Scherzugversuche mit V_{sv0} beziehungsweise V_{sv45} bezeichnet.

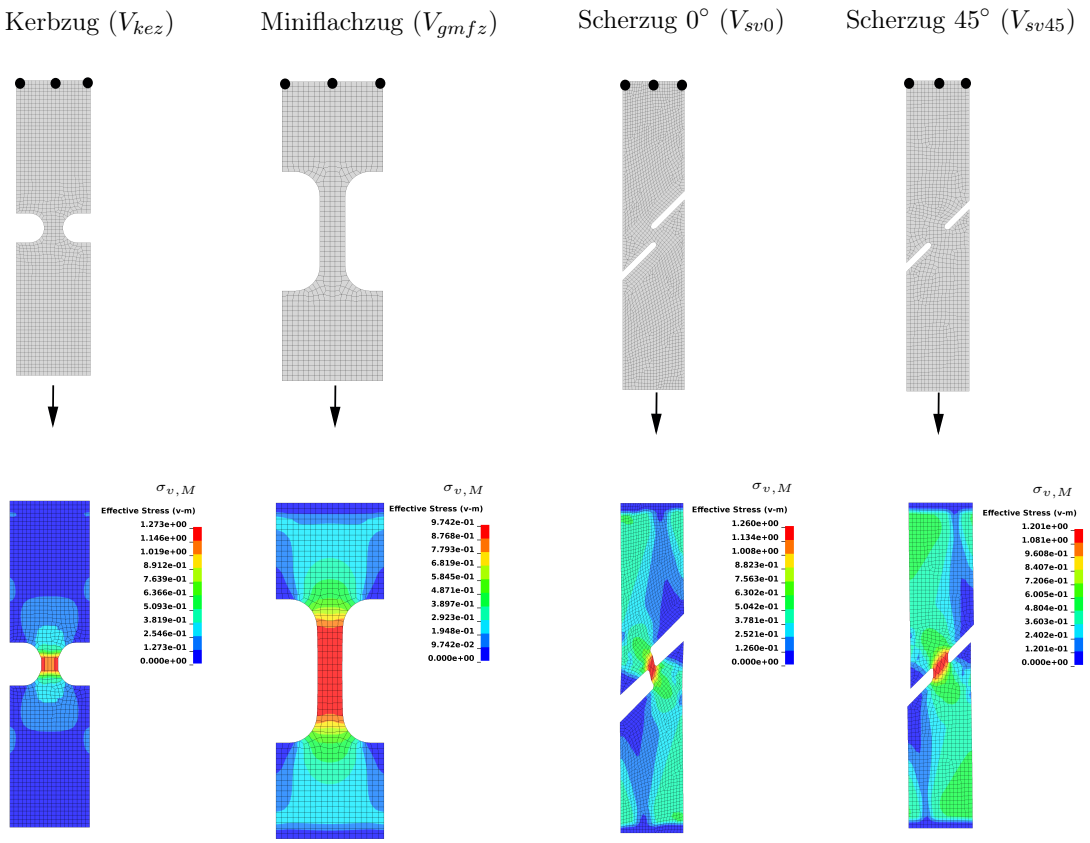


Abbildung 4.33: Zusätzlich betrachtete Gissmo-Zugproben, Oben: Aufbau des jeweiligen simulierten Versuchs, Unten: Verteilung der von-Mises-Spannungen $\sigma_{v,M}$ auf der Geometrie im letzten Zeitschritt des simulierten Versuchs

Abbildung 4.35 zeigt die Triaxialität mit den jeweiligen Werten für die plastische Dehnung für jeden Datenpunkt (Spannungskomponenten eines einzelnen Zeitschritts und räumlichen Punktes) der im vorherigen Abschnitt 4.5.2 genutzten, einzelnen Lochzugprobe und für den in diesem Abschnitt betrachteten, gesamten Datensatz bestehend aus allen sieben Versuchen. Hierbei ist erkennbar, dass die Abdeckung mit den zusätzlichen Versuchen sowohl im Druck- als auch im Zugbereich etwas größer ist. Ein wesentlicher Unterschied zu den bisher betrachteten Datensätze gegeben durch die jeweils einzelnen Versuche ergibt sich jedoch bezüglich der Anzahl der zur Verfügung stehenden Datenpunkte (Dehnpfade), welche zusätzlich zur Diskretisierung der Proben (grobes oder feines Netz) natürlich abhängig der Anzahl der betrachteten Versuchsproben ist. Während der einfache Patch-Zugversuch aus Abschnitt 4.5.1 aus lediglich 80 Dehnpfaden besteht, vergrößert sich für die einzelne Lochzugprobe in Abschnitt 4.5.2 die Anzahl der Pfade auf 9052 und wurde in diesem Abschnitt mittels der sieben kombinierten Versuche nochmals deutlich auf 32809

erhöht.

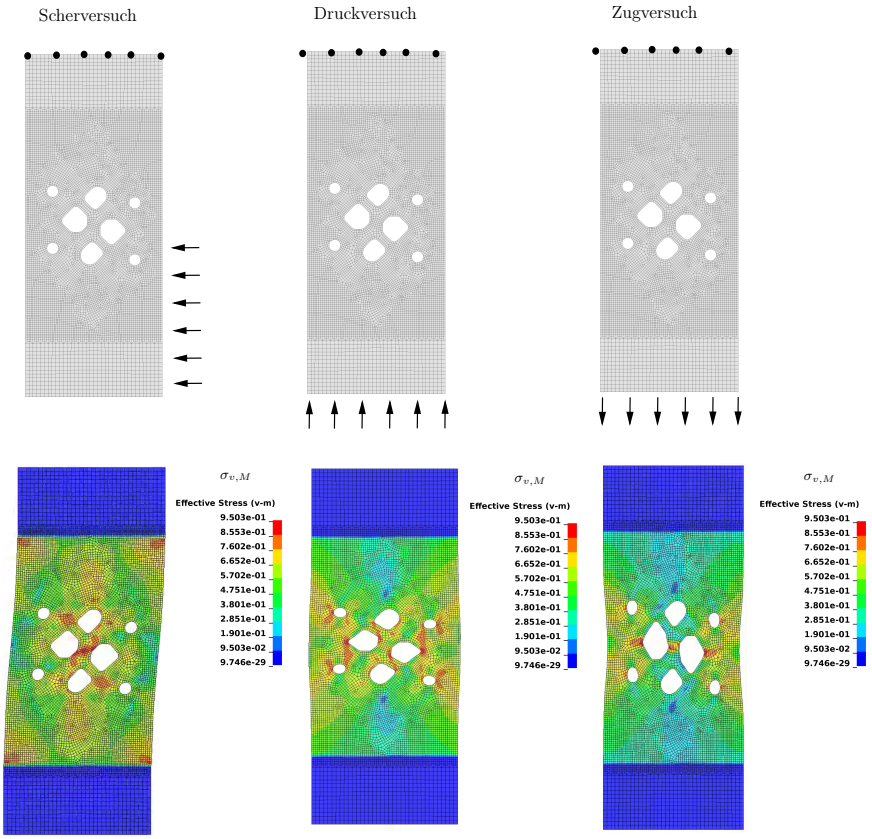


Abbildung 4.34: Neuartige Lochzugprobe unter verschiedenen Belastungen, Oben: Aufbau des jeweiligen simulierten Versuchs, Unten: Verteilung der von-Mises-Spannungen $\sigma_{v,M}$ auf der Geometrie im letzten Zeitschritt des simulierten Versuchs

Für das Training des ML-Materialmodells anhand aller dieser Versuche kombiniert werden für den zu minimierenden Gesamtfehler $L_{total}(w)$ die gewichteten Fehlerwerte $L_{total}^{V_{ld}}(w), ..., L_{total}^{V_{sv45}}(w)$ aller einzelnen Versuche aufsummiert:

$$L_{total}(w) = L_{total}^{V_{ls}}(w)+L_{total}^{V_{ld}}(w)+L_{total}^{V_{lz}}(w)+L_{total}^{V_{kez}}(w)+L_{total}^{V_{gmfz}}(w)+L_{total}^{V_{sv0}}(w)+L_{total}^{V_{sv45}}(w).$$

Die Menge der Dehnungen aller sieben Versuche wird mit V_{kombi} und die Abweichung der vom ML-Materialmodell vorhergesagten und korrekten Spannungen aller Versuche dementsprechend mit MSE(V_{kombi}, ML) bezeichnet.

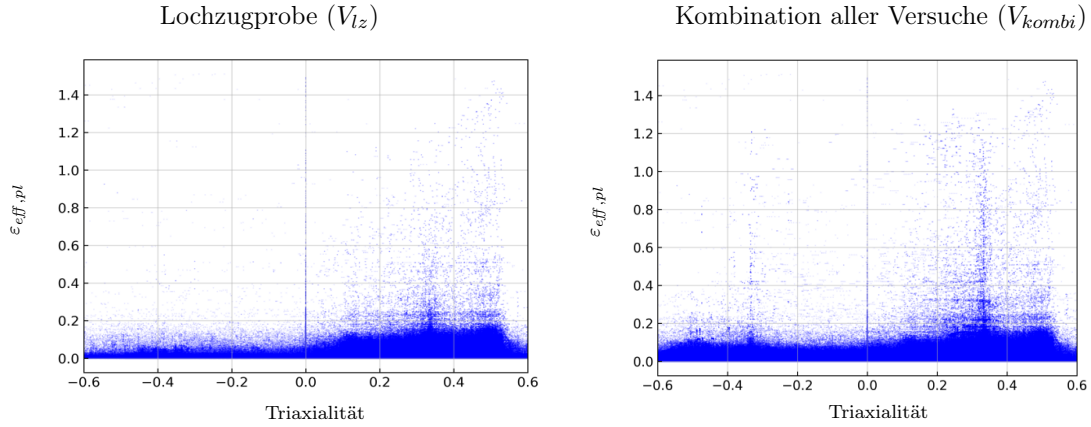


Abbildung 4.35: Abdeckung der Spannungszustände gemessen an der Triaxialität (siehe (3.23)) und der plastischen Dehnung $\varepsilon_{eff,pl}$ für den einzelnen Zugversuch V_{lz} (links) und den gesamten Datensatz V_{kombi} bestehend aus allen sieben Versuchen (rechts)

Der typische Verlauf der Optimierung mit der Gewichtung $w = (1, 1, 1, 1)$ durch den CMA-Algorithmus, welcher wie bisher als derjenige Optimierungsdurchlauf ausgewählt wird, der am wenigsten vom Verlauf des Mittelwerts aller durchgeführten CMA-Optimierungsläufe abweicht, ist in der Abbildung 4.36 auf der linken Seite zu sehen.

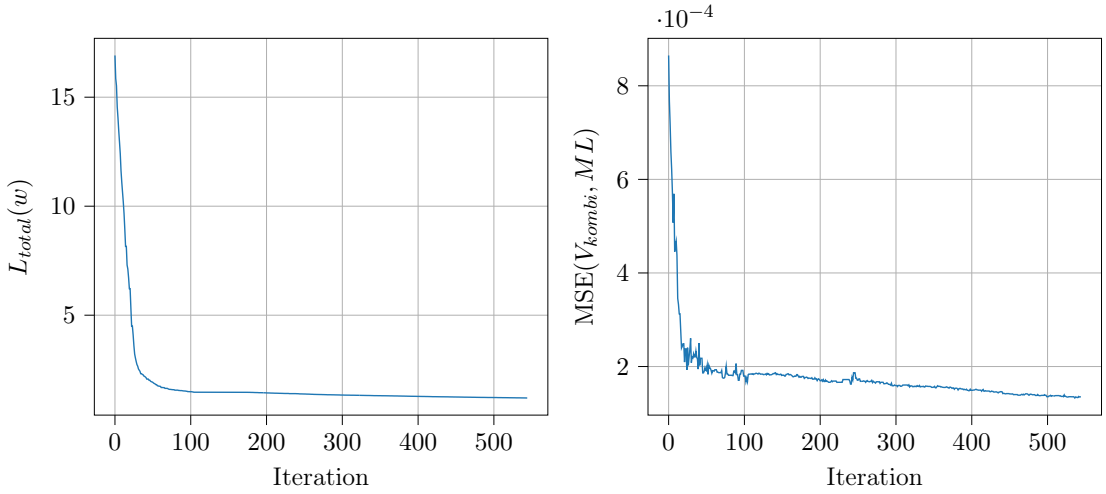


Abbildung 4.36: Links: Verlauf des gewichteten Gesamtfehlers $L_{total}(w)$ aller simulierter Versuche aufsummiert bei einer Gewichtung $w = (1, 1, 1, 1)$ über die Iterationen bei einer Optimierung mittels CMA bei $\sigma_0 = 0.0001$, Rechts: Verlauf der mittleren quadratischen Abweichung MSE(V_{kombi}, ML) zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen aller simulierten Versuche

Die Varianz des erreichten Zielfunktionswerts aller durchgeführten Optimierungsläufe ist auch hier mit $2.774 \cdot 10^{-4}$ sehr gering. Um den Referenzfehler des direkt trainierten Ziel-ML-Materialmodells ML_{DP800}^{LMSC} zu erreichen, werden hier deutlich mehr Iterationen als bei den Optimierungen mit den einzelnen Versuchen benötigt. Nach 530 Iterationen wird allerdings auch hier bei dem betrachteten Optimierungslauf der Referenzfehler von 1.209 durch einen Fehlerwert von $L_{total}(w) = 1.208$ übertroffen. Das nach 530 Iterationen erhaltene ML-Materialmodell wird im Folgenden mit $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$ bezeichnet. Auf der rechten Seite von Abbildung 4.36 ist die ebenfalls abnehmende mittlere Abweichung $\text{MSE}(V_{kombi}, ML)$ der vom optimierten ML-Materialmodell vorhergesagten und korrekten Spannungen aller betrachteten Versuche zu sehen. Die Verläufe der Abweichungen der einzelnen Versuche separat zeigt die Abbildung 4.37. Während über den gesamten Verlauf der 530 Iterationen die abnehmende Tendenz für jede Versuchsprobe separat deutlich zu erkennen ist, unterscheiden sich lokal betrachtet in erster Linie zu Beginn der Optimierung die Verläufe teilweise leicht. So steigt beispielsweise die $\text{MSE}(V_{sv45}, ML)$-Abweichung zwischen der 50. und 100. Iteration, während der $\text{MSE}(V_{gmfz}, ML)$-Wert im gleichen Bereich sinkt. Dies deutet darauf hin, dass die Datenpunkte aus einem Versuch nicht oder nur unzureichend durch einen anderen abgedeckt werden. Dass die Kombination der unterschiedlichen Versuche einen Mehrwert bringt, ist außerdem in der Übersichtstabelle 4.14 anhand des Modells $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{lz}}$ zu erkennen.

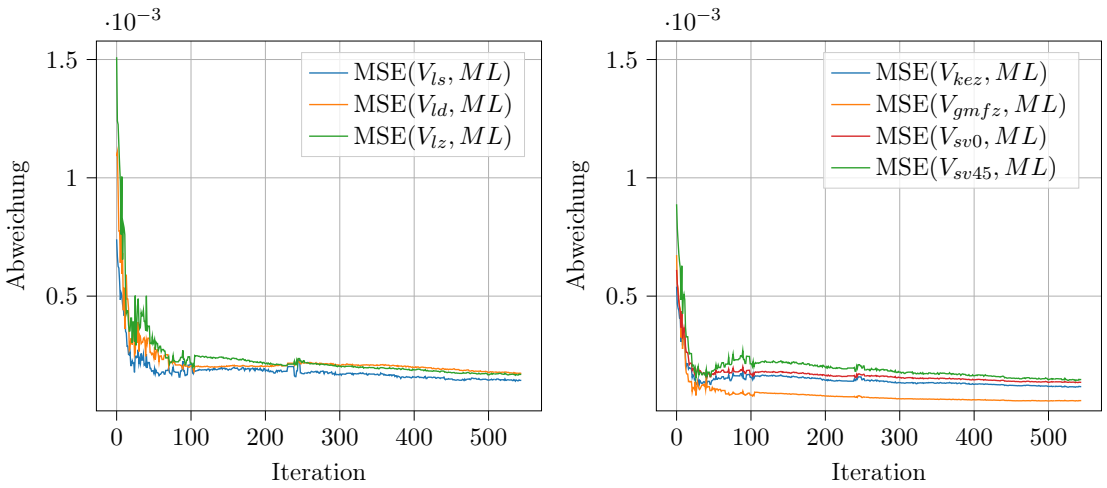


Abbildung 4.37: Verläufe der mittleren quadratischen Abweichung zwischen den vom optimierten ML-Modell vorhergesagten und den korrekten Spannungen der einzelnen simulierten Versuche Links: Verläufe für die Lochproben, Rechts: Verläufe für die Gissmo-Versuchsproben

Auch bei dem Training basierend auf den in diesem Abschnitt betrachteten sieben unterschiedlichen Versuchen ist eine andere Wahl der Gewichte der einzelnen Fehlerfunktionen möglich. Allerdings wird aufgrund der bisherigen Untersuchung unterschiedlicher Gewichtungen (siehe Tabellen 4.3, 4.7 und 4.11) sowie der Betrachtung der Ergebnisse eines mehrkriteriellen Optimierungsansatzes (siehe Tabellen 4.4, 4.8 und 4.12), auf Basis welcher eine gleiche Gewichtung der verschiedenen Zielfunktionen sich durchweg als sehr guter Indikator für eine Lösung (ML-Materialmodell) mit geringen MSE-Werten herausgestellt hat, die gewichtete Summe $L_{total}(w)$ mit $w = (1, 1, 1, 1)$ als Zielfunktion in diesem Abschnitt ausgewählt.

	ML_{DP600}^{LMSC}	ML_{DP800}^{LMSC}	$ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$	$ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{lz}}$
$L_{total}(w)$ mit $w = (1, 1, 1, 1, 1)$	16.914	1.209	1.208	8.922
$L_{total}^{V_{ls}}$	2.269	0.184	0.174	1.781
$L_{total}^{V_{ld}}$	5.518	0.213	0.247	6.234
$L_{total}^{V_{lz}}$	6.110	0.325	0.262	0.260
$L_{total}^{V_{kez}}$	0.704	0.081	0.104	0.119
$L_{total}^{V_{gmfz}}$	0.716	0.062	0.077	0.095
$L_{total}^{V_{sv0}}$	0.574	0.151	0.199	0.252
$L_{total}^{V_{sv45}}$	1.023	0.193	0.144	0.181
$\text{MSE}(V_{kombi}, ML)$	$8.648 \cdot 10^{-4}$	$6.445 \cdot 10^{-5}$	$1.355 \cdot 10^{-4}$	$4.583 \cdot 10^{-4}$
$\text{MSE}(V_{lz}, ML)$	$1.510 \cdot 10^{-3}$	$8.534 \cdot 10^{-5}$	$1.695 \cdot 10^{-4}$	$2.577 \cdot 10^{-4}$
MSE^{rand}	$8.727 \cdot 10^{-3}$	$1.526 \cdot 10^{-4}$	$2.930 \cdot 10^{-3}$	$6.483 \cdot 10^{-3}$

Tabelle 4.14: Funktionswerte für die unterschiedlichen Versuche und des Gesamtfehlers $L_{total}(w)$ aller Versuche jeweils bei einer gleichen Gewichtung $w = (1, 1, 1, 1)$ sowie die Abweichungen $\text{MSE}(V_{kombi}, ML)$, $\text{MSE}(V_{lz}, ML)$ und MSE^{rand} für die unterschiedlich optimierten Modelle sowie des vortrainierten Startmodells ML_{DP600}^{LMSC} und des direkt auf Spannungs-Dehnungsdaten trainierten Zielmodells ML_{DP800}^{LMSC}. Die Fehlerwerte $L_{total}^{V_{ls}}, ..., L_{total}^{V_{sv45}}$ der jeweiligen Versuche beziehen sich jeweils auf eine gleiche Gewichtung der einzelnen Fehlerfunktionen.

In Tabelle 4.14 wird erkennbar, dass das einzig anhand der Lochzugprobe optimierte Modell $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{lz}}$ bei allen Versuchen, welche nicht in der Optimierung enthalten waren (alle außer V_{lz}) einen höheren Fehlerwert zeigt als das Modell $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$, bei welchem die entsprechenden Versuche in der Optimierung enthalten waren, auch wenn

sich in einigen Fällen der Wert gegenüber dem Startmodell zumindest leicht verbessert. Am deutlichsten ist der höhere Fehlerwert des Modells $ML^{LMSC,DP800,V_{lz}}_{opt,CMA(\sigma_0=0.0001)}$ bei dem Druckversuch V_{ld} zu sehen, für welchen sich beim Modell $ML^{LMSC,DP800,V_{lz}}_{opt,CMA(\sigma_0=0.0001)}$ während der Optimierung der Fehlerwert $L^{V_{ld}}_{total}$ gegenüber dem Startmodell sogar erkennbar verschlechtert. Dies unterstreicht die Wichtigkeit der Hinzunahme eines solchen Versuchs in das Training der ML-Materialmodelle.

Vergleicht man die Verläufe der Abweichung für den zufällig generierten Validierungsdatensatz $\text{MSE}^{rand,rel}$ (siehe Abbildung 4.38 rechts) für die verschiedenen ML-Materialmodelle, welche anhand der unterschiedlichen simulierten Versuche trainiert wurden, so ist eine deutliche Verbesserung für das Modell $ML^{LMSC,DP800,V_{kombi}}_{opt,CMA(\sigma_0=0.0001)}$ zu erkennen, welches in diesem Abschnitt anhand mehrerer Versuche V_{kombi} trainiert wurde gegenüber den Modellen, welche als Datenbasis den Patch-Zugversuch V_p beziehungsweise die Lochzugprobe V_{lz} zur Verfügung bekommen haben.

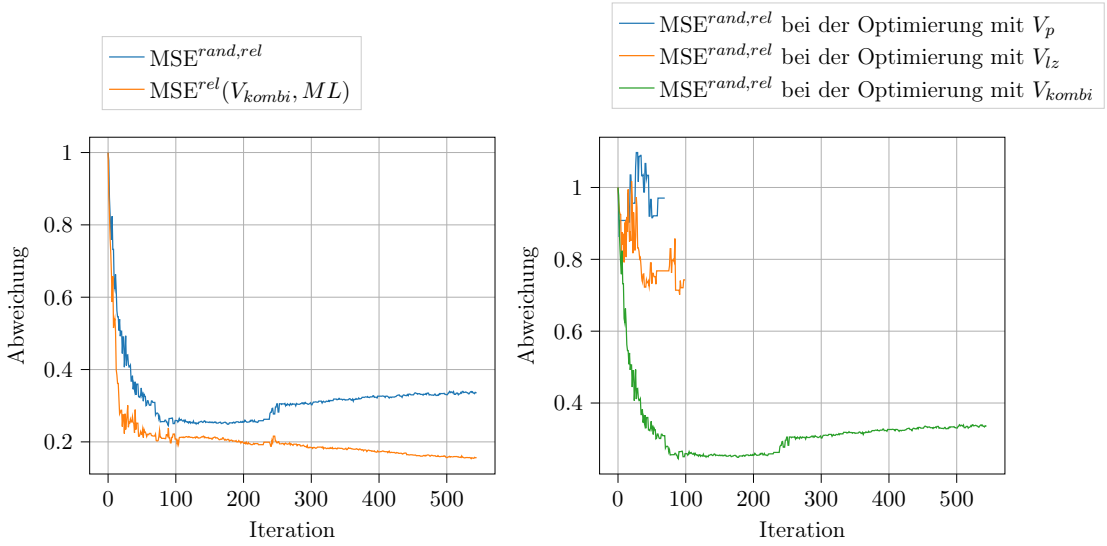


Abbildung 4.38: Fehlerverläufe für $\text{MSE}^{rand,rel}$ Links: Vergleich der Verläufe für $\text{MSE}^{rand,rel}$ und $\text{MSE}(V_{kombi}, ML)$ bei der Optimierung von $L_{total}(w), w = (1,1,1,1)$ mittels CMA für $\sigma_0 = 0.0001$ anhand V_{kombi}, Rechts: Vergleich der Verläufe von $\text{MSE}^{rand,rel}$ für die Optimierung von $L_{total}(w), w = (1,1,1,1)$ anhand der Versuche V_p, V_{lz} und für die Kombination V_{kombi} der sieben Versuchsproben

Die deutliche Verbesserung der Prognosefähigkeit des in diesem Abschnitt trainierten ML-Materialmodells für nicht im Training gesehene, zufällig generierte Daten ist auch beim Vergleich der relativen Abweichung der Trainingsdaten $\text{MSE}^{rel}(V_{kombi}, ML)$ mit

dem Verlauf von $\text{MSE}^{rand,rel}$ in Abbildung 4.38 auf der linken Seite sichtbar. Die bisher beobachtbare Diskrepanz der beiden Verläufe ist deutlich geringer als bei den trainierten Modellen der vorherigen Abschnitte. Allerdings sind im Laufe des Trainings Anzeichen einer Überanpassung in den späteren Iterationen erkennbar, da die Abweichung bezüglich der im Training berücksichtigten Daten $\text{MSE}^{rel}(V_{kombi}, ML)$ weiter sinkt, während die Abweichung für die Validierungsdaten $\text{MSE}^{rand,rel}$ wieder leicht ansteigt.
Die Eignung des optimierten ML-Modells in Hinblick auf ungesehene, komplexere Anwendungen zeigt sich auch im Einsatz des ML-Materialmodells $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{lz}}$ innerhalb der in dieser Arbeit betrachteten Validierungssimulationen (siehe auch Abschnitt 3.2.7). Das optimierte ML-Modell läuft in allen Simulationen ohne Fehler durch. Die Abweichungen der Spannungsausgaben MSE_σ sowie der Knotenpositionen MSE_{pl} werden in allen Anwendungssimulationen (teilweise auch deutlich) kleiner im Vergleich zum Startmodell, wie in Tabelle 4.15 ersichtlich wird.

	ML_{DP600}^{LMSC}	$ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$	ML_{DP800}^{LMSC}
Zugversuch MSE_{pl}	$2.074 \cdot 10^{-4}$	$1.374 \cdot 10^{-4}$	$2.459 \cdot 10^{-5}$
Zugversuch MSE_σ	$8.287 \cdot 10^{-4}$	$8.559 \cdot 10^{-5}$	$2.605 \cdot 10^{-5}$
Hutprofil MSE_{pl}	$1.313 \cdot 10^{-1}$	$3.613 \cdot 10^{-2}$	$1.374 \cdot 10^{-2}$
Hutprofil MSE_σ	$4.985 \cdot 10^{-2}$	$3.647 \cdot 10^{-2}$	$1.297 \cdot 10^{-2}$
Crashbox MSE_{pl}	$6.979 \cdot 10^{-1}$	$3.031 \cdot 10^{-1}$	$1.401 \cdot 10^{-2}$
Crashbox MSE_σ	$4.478 \cdot 10^{-2}$	$4.083 \cdot 10^{-2}$	$1.737 \cdot 10^{-2}$
Kreuznapf MSE_{pl}	$8.964 \cdot 10^{-1}$	$5.745 \cdot 10^{-1}$	$8.350 \cdot 10^{-2}$
Kreuznapf MSE_σ	$1.822 \cdot 10^{-1}$	$9.377 \cdot 10^{-2}$	$3.30 \cdot 10^{-2}$

Tabelle 4.15: Übersicht der mittleren quadratischen Abweichungen innerhalb der Simulationen bezüglich der Knotenpositionen MSE_{pl} und Spannungsausgaben MSE_σ für das optimierte Modell $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$ im Vergleich zum Startmodell ML_{DP600}^{LMSC} und Zielmodell ML_{DP800}^{LMSC}

Zu erkennen ist dies zudem besonders deutlich an den von-Mises-Spannungen des Zugversuchs in Abbildung 4.39, welche für das optimierte Modell $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{lz}}$ deutlich näher am Zielmaterial DP800 sind, als beim Zugversuch des Startmodells ML_{DP600}^{LMSC}. Dies ist auch daran zu erkennen, dass die Verschiebungen der Randknoten der Probe für das nicht angepasste Startmodell deutlich größer sind und die Probe weiter auseinandergezogen wird, als es beim klassischen Zielmaterialmodell sowie dem mittels der unterschiedlichen

simulierten Versuche optimierten ML-Materialmodell $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$ der Fall ist.

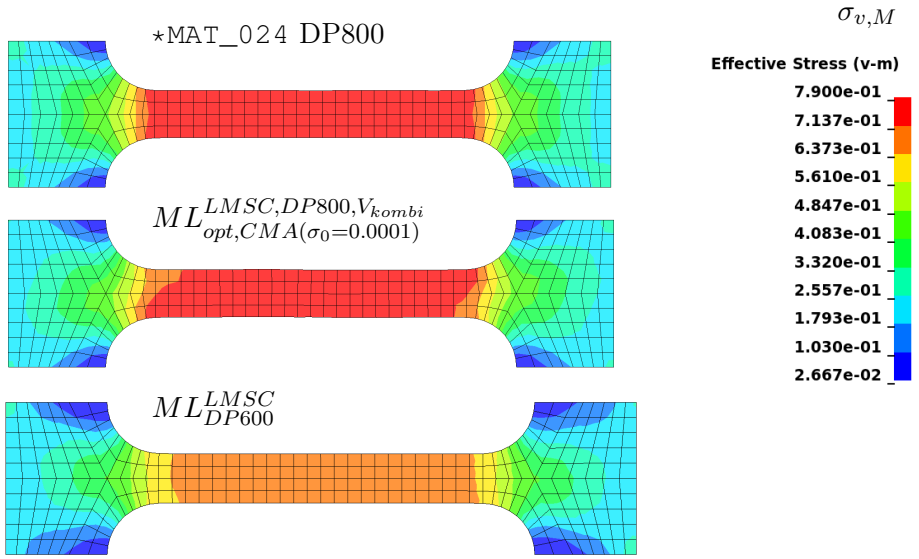


Abbildung 4.39: Vergleich D3plot des Zugversuchs bezüglich der von-Mises-Spannungen $\sigma_{v,M}$ des klassischen Zielmaterialmodells *MAT_024 für den DP800 Stahl, des ML-Materialmodells für den DP600 Stahl (Startmodell ML_{DP600}^{LMSC}) und des anhand der unterschiedlichen, simulierten Versuche optimierten ML-Materialmodells $ML_{opt,CMA(\sigma_0=0.0001)}^{LMSC,DP800,V_{kombi}}$

Fazit

Es wurde gezeigt, dass auch für das in diesem Abschnitt betrachtete, deutlich komplexere elastoplastische Materialmodell ein rekurrentes ML-Materialmodell auf einen weiteren Werkstoff angepasst werden kann, ohne das zugrunde liegende klassische Materialmodell zu benutzten. Die ausschließliche Verwendung der globalen Kraft und der Dehnungen machen die Methode auch in der Praxis anwendbar, um ML-Materialmodelle auf Basis von Trainingsdaten aus Charakterisierungsversuchen für einen Werkstoff zu erstellen, für welchen kein klassisches Materialmodell erstellt wurde. Wie auch schon beim linear elastischen Materialmodell stellt hierfür ein Optimierungsansatz mittels der Skalarisierung als gewichtete Summe mit einer gleichen Gewichtung aller Zielfunktionen einen sehr guten Trainingsansatz dar. Dies kann auf Basis der Ergebnisse von Optimierungsläufen mit einer unterschiedlichen Wahl von Gewichtungen sowie auf den Ergebnissen basierend auf einem mehrkriteriellen Optimierungsansatz gefolgert werden. Eine Verallgemeinerungsfähigkeit der optimierten ML-Materialmodelle kann für das elastoplastische Materialmodell durch den Einsatz einer neu entwickelten, komplexeren Versuchsprobe verbessert und in aus-

reichendem Maße schließlich durch die Kombination mehrerer Versuchsproben erreicht werden. Mithilfe Letzterem kann das auf Basis der simulierten Versuche optimierte ML-Materialmodell auch in den unterschiedlichen Validierungssimulationen eingesetzt werden, bei welchen sich die Ergebnisse gegenüber dem Startmodell verbessern.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Die vorliegende Arbeit zeigt und untersucht verschiedene Möglichkeiten, um ein linear elastisches und elastoplastisches Materialmodell durch Methoden des maschinellen Lernens zu modellieren und zu ersetzen. Während das linear elastische Materialmodell durch ein einfaches lineares Regressionsmodell oder auch durch Feed Forward neuronale Netze mit einer einfachen Netzstruktur direkt ersetzt werden kann, muss für das nichtlineare elastoplastische Materialmodell beachtet werden, dass die Spannungsausgabe vom bisherigen zeitlichen Verlauf der aufgetretenen Dehnungskomponenten abhängig ist. Um dies zu berücksichtigen, wurden zwei Möglichkeiten vorgestellt. Die erste Möglichkeit ergibt sich durch die Betrachtung der Dehnungs- und Spannungsverläufe als multivariate Zeitreihe und den Einsatz von rekurrenten ML-Methoden wie rekurrenten neuronalen Netzen. Da gezeigt wurde, dass Modifikationen in einzelnen Dehnungskomponenten auch noch viele Zeitschritte später zu Änderungen in der Spannungsausgabe führen können und Simulationen aus Dehnungsverläufen mit hunderttausenden an Zeitschritten und mehreren Millionen von Elementen bestehen können, wurden ausschließlich ML-Methoden betrachtet, welche die Daten (zeitlichen Verläufe) sequentiell verarbeiten und die Historie über zwischengespeicherte Zustandsvariablen weitergeben, ohne die Notwendigkeit mit Zeitfenstern zu arbeiten, wie es beispielsweise bei Transformer-Modellen oder auch Convolutional Neural Networks notwendig wäre. Eine weitere Modellierungsmöglichkeit ergibt sich durch die Hinzunahme und Integration der internen Variablen des klassischen Materialmodells in das Training der ML-Modelle. Dadurch kann das klassische Materialmodell als Regressionsproblem für einen beliebig festen gegebenen Zeitschritt betrachtet werden. Für das elastoplastische Materialmodell und die Materialparameter eines DP800 Stahls wurden in diesem Rahmen unterschiedliche ML-Materialmodelle trainiert und für jede Methode eine breite Variation an Hyperparametern untersucht. Durch die Betrachtung des jeweiligen ML-Modells mit der besten Konfiguration für jede Methode wurden die verschiedenen Ansätze miteinander verglichen. Die Evaluierung und Validierung umfasst dabei die Vorhersagegenauigkeit für unterschiedliche Testdatensätze, welche sowohl

zufällige, generisch erstellten Daten einbeziehen als auch Daten, welche einer Simulation entnommen wurden. Als finaler Validierungsschritt werden die unterschiedlichen ML-Materialmodelle in verschiedenen Anwendungssimulationen innerhalb der Simulationssoftware LS-Dyna (angefangen mit einem einfachen Zugversuch bis hin zu gesamten Bauteil-Simulationen im Bereich Umformen und Crash) integriert. Die Ergebnisse der verschiedenen ML-Materialmodelle werden ausgewertet und zu den korrespondierenden Simulationsergebnissen mit dem klassischen Materialmodell verglichen und validiert. Hierbei kann festgestellt werden, dass ML-Materialmodelle grundsätzlich für die betrachteten Anwendungen in der Lage sind die klassischen Materialmodelle zu ersetzen. Je nach ML-Modell treten nur geringe Abweichungen auf, jedoch akkumuliert sich der Fehler über die Zeit. In Simulationen mit vielen Zeitschritten kann dies zu größeren Abweichungen oder sogar zum Abbruch führen. Im Allgemeinen ist eine sehr hohe Vorhersagegenauigkeit der ML-Modelle notwendig, um stabile Simulationen und vergleichbare Ergebnisse zu den Simulationen mit den klassischen Materialmodellen zu gewährleisten. Wichtig in der Auswertung hinsichtlich einer Bewertung der ML-Materialmodelle in Hinblick auf die Einsatzfähigkeit in Simulationen ist die Berücksichtigung der Fortpflanzung des Fehlers im Laufe der Zeitschritte. Diese Fehlerfortpflanzung ist bei den ML-Materialmodellen, welche mit Hinzunahme der internen Variablen des klassischen Materialmodells trainiert werden deutlich ausgeprägter als bei den rekurrenten ML-Modellen, welche anhand der gesamten zeitlichen Verläufe trainiert werden, auch wenn diese ML-Modelle einen sehr niedrigen Fehler im Training und anhand der Testdatensätze ohne Berücksichtigung der Fehlerfortpflanzung aufwiesen. Aus diesem Grund sind die Simulationsergebnisse mit den rekurrenten ML-Modellen deutlich genauer und stabiler (hinsichtlich Abbrüchen der Simulationen) als die Simulationsläufe, welche mittels der ML-Modellen durchgeführt werden, die mit Hinzunahme der Geschichtsvariablen trainiert worden sind. Eine weitere Voraussetzung, welche sich als entscheidend für ein gutes Simulationsergebnis erweist, ist die Fähigkeit eines ML-Materialmodells den Nullzustand, welcher zu Beginn einer Simulation bei einem Großteil der Integrationspunkte und teilweise auch für sehr viele aufeinanderfolgende Zeitschritte vorkommt, korrekt vorherzusagen. Im Gegensatz zu baumbasierten ML-Methoden ist dies bei konventionellen Architekturen von neuronalen Netzen weniger gut möglich. Um eine exakte Vorhersage von Nullzuständen bei einer Abwesenheit von Dehnungen (Dehnungswerte gleich null) zu gewährleisten, wird die rekurrente Netzarchitektur (LMSC-Modelle) aus [11] betrachtet und dieser Ansatz auf Feed Forward neuronale Netze erweitert mit der zusätzlichen Garantie der Nicht-Negativität der plastischen Dehnung als interne Variable des ML-Modells, wie es auch im klassischen Materialmodell der Fall ist. Durch diese Erweiterung in der Architektur der neuronalen

Netze kann die Fehlerfortpflanzung deutlich verringert und bessere Simulationsergebnisse erzielt werden.

Als weiterführender Schritt zu den in Kapitel 3 trainierten ML-Materialmodellen werden Methoden vorgestellt und untersucht, um ein ML-Materialmodell für einen Werkstoff auf Basis von Daten aus Versuchen zu trainieren, ohne ein passendes, bestehendes klassisches Materialmodell entwickeln zu müssen. Die Schwierigkeit des Trainings der ML-Modelle stellt hierbei die Tatsache dar, dass die benötigten Ausgabewerte gegeben durch die lokalen Spannungen nicht in Versuchen messbar sind. Um die Abwesenheit der normalerweise benötigten korrekten Ausgabegrößen zu kompensieren, werden für das Training der ML-Materialmodelle mehrere physikalisch motivierte Gleichungen aus [9] genutzt, welche für die Spannungsvorhersagen des angepassten ML-Materialmodells möglichst genau erfüllt werden sollen. Um die trainierbaren Parameter der neuronalen Netze anzupassen, werden unterschiedliche Optimierungsmethoden im Bereich der evolutionären Algorithmen betrachtet sowie eine Anpassung des üblicherweise für das Training von neuronalen Netzen verwendeten Backpropagation Algorithmus vorgenommen.

Die verschiedenen Optimierungsansätze werden anhand unterschiedlicher, simulierter Versuche für die beiden betrachteten Materialmodelle (linear elastisch und elastoplastisch) angewendet und die Methode für das Training mit Versuchsdaten validiert. Der Fehler der alternativen Fehlerfunktionen kann dabei in jedem Anwendungsfall auf das Fehlerniveau eines direkt anhand Dehnungs-Spannungspaaren trainierten Referenz-ML-Modells minimiert werden. Dadurch kann erreicht werden, dass in den unterschiedlichen Anwendungsfällen die im Versuch gesehenen Daten vom ML-Modell durch das Training anhand der alternativen Fehlerfunktionen erlernt werden. Während der Fehler beim linear elastischen Materialmodell beim Patch Zugversuch mittels des gradientenbasierten Optimierungsansatzes schnell minimiert werden kann und insbesondere dadurch der Referenz-Zielfehlerwert im Vergleich zum Training mittels evolutionären Strategien (durch CMA) früher erreicht wird, ist letzterer Optimierungsansatz bei einer etwas komplexeren Versuchsgeometrie (Miniflachzug) besser geeignet. Da für das Training mit Versuchsdaten mehrere unterschiedliche Fehlerfunktionen betrachtet werden, stellt das Training der ML-Materialmodelle ein mehrkriterielles Optimierungsproblem dar. Dies wird zum einen durch mehrkriterielle Optimierungsalgorithmen gelöst (NSGA-II und NSGA-III) und des Weiteren durch einen Skalarisierungsansatz mittels einer gewichteten Summe modelliert. Bei Variationen der Gewichtungen sowie Vergleichen zwischen den Ergebnissen mittels der Skalarisierung und dem mehrkriteriellen Optimierungsansatz kann festgestellt werden, dass die Skalarisierung mit einer gleichen Gewichtung aller Zielfunktionen (abgesehen

vom Patch Ausschnitt bei dem die Fehlerfunktionen für die Divergenz nicht berücksichtigt werden müssen) sehr gut für das Erlernen der im Versuch gesehenen Spannungen geeignet ist. Eine weitere Herausforderung beim Training der ML-Materialmodelle auf Basis von Versuchsdaten stellt die begrenzte Datenbasis gegeben durch die jeweiligen Versuche und limitiert durch die Messanlagen und Messmethoden dar. In diesem Zusammenhang werden zwei Möglichkeiten für die Verbesserung der Verallgemeinerungsfähigkeit gezeigt. Ein im Allgemeinen anwendbarer Ansatz ergibt sich durch den Einsatz einer Kombination vieler verschiedener Versuche sowie die Nutzung komplexerer Versuchsproben, um die Größe und Diversität des Trainingsdatensatzes zu erhöhen. Eine weitere Möglichkeit ergibt sich durch eine künstliche Generierung weiterer Daten ohne die Notwendigkeit viele komplexe Versuche zu benötigen, indem materialspezifische Eigenschaften durch gültige Transformationen, welche für das Materialmodell als geltend vorausgesetzt werden können, beschrieben werden. Beim Vorhandensein solcher Transformationen kann es sogar vorteilhafter sein das ML-Modell anhand eines sehr einfachen Versuchs (beispielsweise Patch-Zugversuch) zu trainieren und mit den wenigen Daten Augmentierungen zu benutzten um eine gute Verallgemeinerungsfähigkeit des ML-Materialmodells zu erreichen statt anhand vieler Versuche zu trainieren, da in diesem Fall die Anpassung des ML-Modells langwieriger und tendenziell mit einer höheren Abweichung in der Vorhersage verbunden ist, was sich wiederum auf die Genauigkeit der augmentierten Daten auswirkt. Dies hat sich beim linear elastischen Fall gezeigt.
Aber auch durch die Kombination verschiedener Versuche kann ein in diversen Simulationen einsetzbares ML-Materialmodell angepasst werden mit einer eindeutigen Verbesserung gegenüber dem Startmodell auch ohne Voraussetzungen an Materialeigenschaften zu stellen, wie es anhand des elastoplastischen Materialmodells gezeigt wurde.

5.2 Ausblick

In Hinblick auf die Existenz sehr vieler unterschiedlicher Materialmodelle, was unter anderem allein durch hunderte in LS-Dyna implementierten, zur Verfügung stehenden klassischen Modelle sichtbar ist (siehe [49]), können Anwendungen der in dieser Arbeit vorgestellten und untersuchten Methoden auf weitere Werkstoffe mit unterschiedlichsten Materialeigenschaften wie beispielsweise viskoelastischem, viskoplastischem oder hyperelastischem Materialverhalten von zukünftigem Interesse sein.
Die vorgestellte und anhand von simulierten Versuchen validierte Methode für das Training von ML-Materialmodellen anhand von Versuchsdaten ist in der Praxis stark von den vorhandenen Messtechniken und Messanlagen abhängig, welche Beschränkungen unter

anderem bezüglich der möglichen, verwendbaren Versuchsarten, Versuchsaufbauten und Versuchsgeometrien sowie der Genauigkeit der gemessenen Daten auferlegen. Mit der zukünftigen Weiterentwicklung und Verbesserung von Messmethoden und Messanlagen ergeben sich auch weitere Möglichkeiten für das Training von ML-Materialmodellen aus Versuchsdaten. Beispielsweise gibt es bereits erste Ansätze durch Digital Volume Correlation (siehe [14]), um im dreidimensionalen Raum die Dehnungskomponenten zu erfassen, was die in dieser Arbeit vorgestellte Methode für das Training mit Versuchsdaten in Zukunft auch für Volumenelemente beziehungsweise Schalenformulierungen ohne separate Berechnungen der σ_{zx} und σ_{zy} Komponenten anwendbar machen würde.

Literatur

[1] Martín Abadi u. a. „Tensorflow: A system for large-scale machine learning“. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. Software available from tensorflow.org. 2016, S. 265–283. arXiv: 1603.04467.

[2] Fethi Abbassi u. a. „Parameter identification of a mechanical ductile damage using Artificial Neural Networks in sheet metal forming“. In: *Materials & Design* 45 (2013), S. 605–615. ISSN: 0261-3069. DOI: https://doi.org/10.1016/j.matdes.2012.09.032.

[3] Holm Altenbach. *Kontinuumsmechanik - Einführung in die materialunabhängigen und materialabhängigen Gleichungen*. Berlin Heidelberg New York: Springer-Verlag, 2012. ISBN: 978-3-642-24119-2. DOI: https://doi.org/10.1007/978-3-642-24119-2.

[4] Dieter Arendes. „Grundlagen der Umformtechnik“. In: *Einführung in die Umformtechnik: Ein Lehrbuch zum Einstieg für Technikinteressierte*. Wiesbaden: Springer Fachmedien Wiesbaden, 2023, S. 1–30. ISBN: 978-3-658-42034-5. DOI: 10.1007/978-3-658-42034-5_1.

[5] Panagiotis Asteris, Panayiotis Roussis und Maria Douvika. „Feed-Forward Neural Network Prediction of the Mechanical Properties of Sandcrete Materials“. In: *Sensors* 17 (Juni 2017). DOI: 10.3390/s17061344.

[6] O. A. Bauchau und J. I. Craig. „Basic equations of linear elasticity“. In: *Structural Analysis*. Hrsg. von O. A. Bauchau und J. I. Craig. Dordrecht: Springer Netherlands, 2009, S. 3–51. ISBN: 978-90-481-2516-6. DOI: 10.1007/978-90-481-2516-6_1.

[7] Julian Blank und Kalyan Deb. „Pymoo: Multi-Objective Optimization in Python“. In: *IEEE Access* PP (Apr. 2020). DOI: 10.1109/ACCESS.2020.2990567.

[8] Frederic E. Bock u. a. „A Review of the Application of Machine Learning and Data Mining Approaches in Continuum Materials Mechanics“. In: *Frontiers in Materials* 6 (2019). ISSN: 2296-8016. DOI: 10.3389/fmats.2019.00110.

[9] Pauline Böhringer u. a. „A strategy to train machine learning material models for finite element simulations on data acquirable from physical experiments“. In: *Computer Methods in Applied Mechanics and Engineering* 406 (2023), S. 115894. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.115894.

[10] Gaurav Bokil u. a. *Accelerating elastoplastic material models with sparse nonlinear regression: A hybrid approach*. 14th European LS-DYNA Conference, Baden-Baden, Germany. Okt. 2023. URL: https://www.researchgate.net/publication/377776842_Accelerating_elastoplastic_material_models_with_sparse_nonlinear_regression_A_hybrid_approach.

[11] Colin Bonatti und Dirk Mohr. „On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids“. In: *Journal of the Mechanics and Physics of Solids* 158 (2022), S. 104697. ISSN: 0022-5096. DOI: https://doi.org/10.1016/j.jmps.2021.104697.

[12] Colin Bonatti und Dirk Mohr. „One for all: Universal material model based on minimal state-space neural networks“. In: *Science Advances* 7.26 (2021). DOI: 10.1126/sciadv.abf3658.

[13] Leo Breiman u. a. *Classification and Regression Trees*. Wadsworth, 1984. ISBN: 0-534-98053-8.

[14] Ante Buljac u. a. „Digital Volume Correlation: Review of Progress and Challenges“. In: *Experimental Mechanics* 58 (Juni 2018). DOI: 10.1007/s11340-018-0390-7.

[15] Guang Chen. „Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity“. In: *Computational Mechanics* 67 (März 2021). DOI: 10.1007/s00466-021-01981-y.

[16] Qiang Chen, Ruijian Jia und Shanmin Pang. „Deep long short-term memory neural network for accelerated elastoplastic analysis of heterogeneous materials: An integrated data-driven surrogate approach“. In: *Composite Structures* 264 (2021), S. 113688. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2021.113688.

[17] Kyunghyun Cho u. a. „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Okt. 2014, S. 1724–1734. DOI: 10.3115/v1/D14-1179.

[18] François Chollet. *Deep Learning with Python*. Manning, 2021. ISBN: 9781617296864.

[19] Ian R. Cole. „Modelling CPV“. Dissertation. Loughborough University, 2015. URL: https://repository.lboro.ac.uk/articles/thesis/Modelling_CPV/9523520.

[20] Indraneel Das und J. E. Dennis. „Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems“. In: *SIAM Journal on Optimization* 8.3 (1998), S. 631–657. DOI: 10.1137/S1052623496307510.

[21] K. Deb u. a. „A fast and elitist multiobjective genetic algorithm: NSGA-II“. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), S. 182–197. DOI: 10.1109/4235.996017.

[22] Kalyanmoy Deb und Himanshu Jain. „An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints“. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), S. 577–601. DOI: 10.1109/TEVC.2013.2281535.

[23] Peter Wriggers Dietmar Gross Werner Hauger. *Technische Mechanik 4 - Hydromechanik, Elemente der Höheren Mechanik, Numerische Methoden*. Heidelberg: Springer-Verlag, 2014. ISBN: 978-3-642-16828-4. DOI: https://doi.org/10.1007/978-3-642-41000-0.

[24] Matthias Ehrgott. „The Weighted Sum Method and Related Topics“. In: *Multicriteria Optimization*. Springer Berlin Heidelberg, 2005, S. 65–95. ISBN: 978-3-540-27659-3. DOI: 10.1007/3-540-27659-9_3.

[25] G. W. Ellis u. a. „Stress-Strain Modeling of Sands Using Artificial Neural Networks“. In: *Journal of Geotechnical Engineering* 121.5 (1995), S. 429–435. DOI: 10.1061/(ASCE)0733-9410(1995)121:5(429).

[26] Jan Niklas Fuhg u. a. „Modular machine learning-based elastoplasticity: Generalization in the context of limited data“. In: *Computer Methods in Applied Mechanics and Engineering* 407 (2023), S. 115930. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2023.115930.

[27] A. Géron, K. Rother und T. Demmig. *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. O'Reilly, 2020. ISBN: 9783960103394.

[28] J. Ghaboussi, J. H. Garrett und X. Wu. „Knowledge-Based Modeling of Material Behavior with Neural Networks“. In: *Journal of Engineering Mechanics* 117.1 (1991), S. 132–153. DOI: 10.1061/(ASCE)0733-9399(1991)117:1(132).

[29] J. Ghaboussi und D.E. Sidarta. „New nested adaptive neural networks (NANN) for constitutive modeling“. In: *Computers and Geotechnics* 22.1 (1998), S. 29–52. ISSN: 0266-352X. DOI: https://doi.org/10.1016/S0266-352X(97)00034-7.

[30] Ghassem Habibagahi und Alireza Bamdad. „A neural network framework for mechanical behavior of unsaturated soils“. In: *Canadian Geotechnical Journal* 40.3 (2003), S. 684–693. DOI: 10.1139/t03-004.

[31] Nikolaus Hansen und Andreas Ostermeier. „Completely Derandomized Self-Adaptation in Evolution Strategies“. In: *Evolutionary Computation* 9.2 (2001), S. 159–195. DOI: 10.1162/106365601750190398.

[32] Trevor Hastie, Robert Tibshirani und Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[33] André Haufe u. a. *Recent Enhancements to the GISSMO Failure Model in LS-DYNA*. 8th European LS-DYNA Conference, Strasbourg, France. Mai 2011. URL: https://www.researchgate.net/publication/312046173_Recent_Enhancements_to_the_GISSMO_Failure_Model_in_LS-DYNA.

[34] Peter Haupt. *Continuum Mechanics and Theory of Materials*. Springer-Verlag, 2002. ISBN: 978-3-540-43111-4. DOI: https://doi.org/10.1007/978-3-662-04775-0.

[35] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Computation* 9.8 (1997), S. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[36] G.A. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. Wiley, 2000. ISBN: 9780471823049.

[37] Victor Iacob. „Materialmodelle für FE-Simulationen basierend auf künstlichen Neuronalen Netzen“. Masterarbeit. Institut für Technische Mechanik Karlsruher Institut für Technologie, 2023.

[38] Christian Ilg u. a. „Displacement based Simulation and Material Calibration based on Digital Image Correlation Part II - Application“. In: *IOP Conference Series: Materials Science and Engineering* 1284 (Juni 2023), S. 012056. DOI: 10.1088/1757-899X/1284/1/012056.

[39] Christian Ilg u. a. „Parameter Identification Applying Full-Field Calibration (FFC) Techniques“. In: *Proceedings of the 14th International Conference on the Technology of Plasticity - Current Trends in the Technology of Plasticity*. Hrsg. von Katia Mocellin u. a. Cham: Springer Nature Switzerland, 2024, S. 578–585. ISBN: 978-3-031-42093-1.

[40] Dong Phill Jang, Piemaan Fazily und Jeong Whan Yoon. „Machine learning-based constitutive model for J2- plasticity“. In: *International Journal of Plasticity* 138 (2021), S. 102919. ISSN: 0749-6419. DOI: https://doi.org/10.1016/j.ijplas.2020.102919.

[41] Akbar Javadi und Mohammad Rezania. „Applications of artificial intelligence and data mining techniques in soil modeling“. In: *Geomechanics and Engineering* 1 (März 2009). DOI: 10.12989/gae.2009.1.1.053.

[42] Arash Jenab u. a. „The Use of genetic algorithm and neural network to predict rate-dependent tensile flow behaviour of AA5182-O sheets“. In: *Materials & Design* 94 (2016), S. 262–273. ISSN: 0264-1275. DOI: https://doi.org/10.1016/j.matdes.2016.01.038.

[43] Diederik P. Kingma und Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[44] B. Klein. *FEM: Grundlagen und Anwendungen der Finite-Element-Methode im Maschinen- und Fahrzeugbau*. Springer Vieweg Studium. Maschinenelemente und Konstruktion. Vieweg+Teubner Verlag, 2012. ISBN: 9783834821348. DOI: https://doi.org/10.1007/978-3-8348-2134-8.

[45] David Kracker u. a. *Automatic Analysis of Crash Simulations with Dimensionality Reduction Algorithms such as PCA and t-SNE*. 16th International LS-DYNA Conference. Okt. 2020.

[46] Siddhant Kumar und Dennis M. Kochmann. „What Machine Learning Can Do for Computational Solid Mechanics“. In: *Current Trends and Open Problems in Computational Mechanics*. Hrsg. von Fadi Aldakheel u. a. Cham: Springer International Publishing, 2022, S. 275–285. ISBN: 978-3-030-87312-7. DOI: 10.1007/978-3-030-87312-7_27.

[47] M. Lefik und B.A. Schrefler. „Artificial neural network as an incremental non-linear constitutive model for a finite element code“. In: *Computer Methods in Applied Mechanics and Engineering* 192.28 (2003). Multiscale Computational Mechanics

for Materials and Structures, S. 3265–3283. ISSN: 0045-7825. DOI: https://doi.org/10.1016/S0045-7825(03)00350-5.

[48] Xueyang Li, Christian C. Roth und Dirk Mohr. „Machine-learning based temperature- and rate-dependent plasticity model: Application to analysis of fracture experiments on DP steel“. In: *International Journal of Plasticity* 118 (2019), S. 320–344. ISSN: 0749-6419. DOI: https://doi.org/10.1016/j.ijplas.2019.02.012.

[49] *Livermore Software Technology, An ANSYS Company, LS-DYNA® R12 Keyword User's Manual*. Livermore, California, USA. 2020.

[50] J. Lubliner. *Plasticity Theory*. Dover Books on Engineering. Dover Publications, 2013. ISBN: 9780486318202.

[51] Andreas Lutz. „Crash“. In: *Methodische Werkstoff- und Prozessentwicklung für die additive Serienproduktion von automobilen Strukturkomponenten*. Springer Berlin Heidelberg, 2023, S. 93–110. ISBN: 978-3-662-66532-9. DOI: 10.1007/978-3-662-66532-9_7.

[52] Michael Reck Manfred Hahn. *Kompaktkurs Finite Elemente für Einsteiger*. Heidelberg: Springer-Verlag, 2021. ISBN: 978-3-658-33410-9. DOI: https://doi.org/10.1007/978-3-658-33411-6.

[53] Matthias Merzkirch. *Mechanical Characterization Using Digital Image Correlation: Advanced Fibrous Composite Laminates*. Jan. 2022. ISBN: 978-3-030-84039-6. DOI: 10.1007/978-3-030-84040-2.

[54] D. Millar und E. Clarici. „Investigation of back-propagation artificial neural networks in modelling the stress-strain behaviour of sandstone rock“. In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*. Bd. 5. 1994, 3326–3331 vol.5. DOI: 10.1109/ICNN.1994.374770.

[55] Francisco J. Montáns u. a. „Data-driven modeling and learning in science and engineering“. In: *Comptes Rendus Mécanique* 347.11 (2019). Data-Based Engineering Science and Technology, S. 845–855. ISSN: 1631-0721. DOI: https://doi.org/10.1016/j.crme.2019.11.009.

[56] Mojtaba Mozaffar u. a. „Deep learning predicts path-dependent plasticity“. In: *Proceedings of the National Academy of Sciences of the United States of America* 116 (2019), S. 26414–26420. URL: https://api.semanticscholar.org/CorpusID:209391265.

[57] Yacoub Najjar und Chune Huang. „Simulating the stress–strain behavior of Georgia kaolin via recurrent neuronet approach“. In: *Computers and Geotechnics* 34 (Sep. 2007), S. 346–361. DOI: 10.1016/j.compgeo.2007.06.006.

[58] Pauline Böhringer, Joachim Sprave, Said Jamei, Norbert Dölle, David Koch, Christian Ilg, André Haufe, Thomas Haase, Thomas Soot, Julian Kleih, Daniel Sommer, Dominik Platzer, Peter Middendorf, Celalettin Karadogan, Mathias Liewald, Markus Stoll, Stefan Suwelack. *Verbundprojekt AIMM: Artificial Intelligence for Material Models : gemeinsamer Schlussbericht zum Verbundprojekt, Laufzeit vom 01.01.2021 bis 31.12.2023*. Stuttgart, 2024.

[59] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[60] R. M. V. Pidaparti und M. J. Palakal. „Material model for composites using neural networks“. In: *AIAA Journal* 31.8 (1993), S. 1533–1535. DOI: 10.2514/3.11810.

[61] Ingo Rechenberg. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, 1973.

[62] D. E. Rumelhart, G. E. Hinton und R. J. Williams. „Learning internal representations by error propagation“. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, S. 318–362. ISBN: 026268053X.

[63] Karlheinz Schiebold. „Mechanische Prüfverfahren“. In: *Zerstörende Werkstoffprüfung: Mechanisch-technologische Verfahren*. Springer Berlin Heidelberg, 2018, S. 16–116. ISBN: 978-3-662-57797-4. DOI: 10.1007/978-3-662-57797-4_2.

[64] Scott Schoenfeld und Dave Benson. „Quickly convergent integration methods for plane stress plasticity“. In: *Communications in Numerical Methods in Engineering* 9.4 (1993), S. 293–305. DOI: https://doi.org/10.1002/cnm.1640090403.

[65] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Jan. 1995. ISBN: 978-0-471-57148-3.

[66] Daniel Sommer u. a. *Using Data from Physical Experiments to Train Machine Learning Material Models*. 14th European LS-DYNA Conference, Baden-Baden, Germany. Okt. 2023.

[67] J. Sprave, T. Erhart und A. Haufe. „An Interprocess Communication based Integration of AI User Materials into LS-DYNA“. In: *14th European LS-Dyna Konferenz, Baden-Baden, Germany*. 15. Okt. 2023.

[68] C.C. Tasan u. a. „An Overview of Dual-Phase Steels: Advances in Microstructure-Oriented Processing and Micromechanically Guided Design“. In: *Annual Review of Materials Research* 45.Volume 45, 2015 (2015), S. 391–431. ISSN: 1545-4118. DOI: https://doi.org/10.1146/annurev-matsci-070214-021103.

[69] Nick Vlassis und Waiching Sun. „Component-based machine learning paradigm for discovering rate-dependent and pressure-sensitive level-set plasticity models“. In: *Journal of Applied Mechanics* (Nov. 2021). DOI: 10.1115/1.4052684.

[70] Marcus Wagner. *Lineare und nichtlineare FEM - Eine Einführung mit Anwendungen in der Umformsimulation mit LS-DYNA*. Heidelberg: Springer-Verlag, 2022. ISBN: 978-3-658-36521-9. DOI: https://doi.org/10.1007/978-3-658-36522-6.

[71] Marcus Wagner. „Materielle Nichtlinearität“. In: *Lineare und nichtlineare FEM: Eine Einführung mit Anwendungen in der Umformsimulation mit LS-DYNA®*. Springer Fachmedien Wiesbaden, 2022, S. 197–228. ISBN: 978-3-658-36522-6. DOI: 10.1007/978-3-658-36522-6_10.

[72] Paul Werbos. „Generalization of Backpropagation with Application to a Recurrent Gas Market Model“. In: *Neural Networks* 1 (Dez. 1988), S. 339–356. DOI: 10.1016/0893-6080(88)90007-X.

[73] Kailai Xu, Daniel Z. Huang und Eric Darve. „Learning constitutive relations using symmetric positive definite neural networks“. In: *Journal of Computational Physics* 428 (2021), S. 110072. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.110072.

[74] Hongbo Zhao, Zenghui Huang und Zhengsheng Zou. „Simulating the Stress-Strain Relationship of Geomaterials by Support Vector Machine“. In: *Mathematical Problems in Engineering* 2014 (Aug. 2014). DOI: 10.1155/2014/482672.