Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Erweiterung eines Convolutional Neural Network auf 3D zur Modellierung von Grundwasserwärmepumpen

Johanna Trick

Studiengang: Informatik

Prüfer/in: Prof. Dr. Miriam Schulte

Betreuer/in: M.Sc. Julia Pelzer

Beginn am: 6. Mai 2024

Beendet am: 6. November 2024

Kurzfassung

In dieser Arbeit untersuchen wir die Skalierung von Datenbedarf und Leistung zwischen 2D und 3D Convolutional Neural Networks (CNNs). Dazu erweitern wir ein U-Net-basiertes 2D CNN zur Modellierung von stationären Grundwasserwärmefahnen [PS24] um eine räumliche Dimension. Unser optimiertes Netzwerk trainieren wir auf PFLOTRAN-simulierten 3D Daten mit 100 und 1000 Datenpunkten (dp). Die verwendeten 2D Modelle basieren auf der Arbeit von Pelzer und Schulte [PS24]. Alle Netzwerke werden von Grund auf trainiert. Für den Vergleich berücksichtigen wir Trainingszeit, Inferenzzeit und Speicherbedarf sowie die Metriken Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), zellbasierter Averaged Absolute Error (AAE) und Percentage Above Threshold (PAT), den prozentualen Anteil an Fehlerzellen über 0.1°C. Zusätzlich geben wir Varianz und Standardabweichung des Vorhersagefehlers an. Unser bestes 3D Modell zeigt einen vierfachen RMSE, doppelten MAE und neunfachen PAT bei zehnfach höherer Trainingszeit als das beste 2D Modell. Die Inferenzzeit ist im Vergleich zum 2D Modell um das 26-fache erhöht. Unser Modell zeigt Schwierigkeiten bei der Vorhersage von länglichen Wärmefahnen entlang der Strömungsrichtung und weist hohe Fehlerwerte im Bereich des Injektionspunktes auf. Die Ergebnisse zeigen, dass im 3D Fall eine Datenmenge von 1000 dp für effizientes Lernen und Konvergenz notwendig ist. Unsere Daten sind auf DaRUS verfügbar (https://doi.org/10.18419/darus-4533, https://doi.org/10.18419/darus-4534), unseren Code haben wir auf Github veröffentlicht (https://github.com/ja-tk/1HP_NN).

Inhaltsverzeichnis

1. Einleitung **13**

2. Grundlagen von CNNs und U-Net **15**
2.1. 2D CNN 15
2.2. 2D U-Net CNN 17
2.3. 2D vs. 3D Convolution 18

3. Verwandte Arbeiten **21**
3.1. Modellierung von Wärmefahnen 21
3.2. Vergleiche zwischen 2D und 3D CNNs 23

4. Methodik **25**
4.1. Implementierung 25
4.2. Hyperparameteroptimierung 25
4.3. Datensatz 27
4.4. Datenvorbereitung 28
4.5. Architektur und Training 29
4.6. Bewertungsmetriken 31
4.7. 2D Modelle und Unterschiede 33

5. Resultat und Diskussion **35**
5.1. Leistungsanalyse der 2D und 3D Modelle 35
5.2. Ergebnisse mit SmoothL1 Loss und Leistungsanalyse des 3D-1000-L1 39
5.3. Vergleich des Rechenaufwands der 2D und 3D Modelle 42
5.4. Evaluierung des Lernfortschritts im Trainingsverlauf 44
5.5. Zusammenführung der Ergebnisse zu Leistung, Rechenaufwand und Effizienz . 47

6. Gesamtbewertung und Zukunftsausblick **49**
6.1. Limitierungen 49
6.2. Einbettung in den wissenschaftlichen Kontext 50
6.3. Zusammenfassung und Ausblick 50

Literaturverzeichnis **53**

A. Anhang **59**
A.1. Hardware 59
A.2. 2D Architektur und Trainingskonfiguration 59
A.3. 2D Datensätze 59
A.4. Fehlermetriken der 3D CNNs in ZY- und XY-Ebene 60
A.5. Varianz und Standardabweichung des zellbasierten AEs 60

A.6. PFLOTRAN-Einstellungen in 2D und 3D 61
A.7. Lossfunktionen 61

Abbildungsverzeichnis

2.1. ReLU Aktivierungsfunktion 16
2.2. 2D vs. 3D Convolution 19

4.1. 3D Visualisierung einer PFLOTRAN-simulierten Wärmefahne mit ParaView . . 28
4.2. 3D Inputparameter 29
4.3. Netzwerkarchitektur des 3D CNN 30

5.1. Visualisierung des Training und Validation Loss der 2D und 3D CNNs 36
5.2. Zellbasierter AAE des 3D-100 in ZY- und XY-Ebene 38
5.3. Zellbasierter AAE der 2D und 3D CNNs 39
5.4. Zellbasierter AAE der 3D CNNs in X-Richtung gemittelt 40
5.5. Validation Loss verschiedener 3D Modelle mit SmoothL1 Loss 41
5.6. Training und Validation Loss des 3D-1000-L1 42
5.7. Zellbasierter AAE des 3D-1000-L1 in ZY-Ebene und in X-Richtung gemittelt . . 43
5.8. Validation Loss über die Trainingszeit der 2D und 3D CNNs 46
5.9. Metrikenverlauf über ausgewählte Epochen und Trainingszeitpunkte 47

Tabellenverzeichnis

4.1. Konfiguration des LR-Scheduler ReduceLROnPlateau 26
4.2. Einstellungen für die Hyperparameteroptimierung mit Optuna 27
4.3. Trainingskonfiguration des 3D CNN 31

5.1. Erreichtes Training und Validation Loss der 2D und 3D CNNs 36
5.2. Fehlermetriken der 2D und 3D CNNs 37
5.3. Fehlermetriken mit SmoothL1 Loss 40
5.4. Fehlermetriken des 3D-1000-L1 41
5.5. Trainingszeit der 2D und 3D CNNs 44
5.6. Speicheranforderungen der 2D und 3D CNNs 44
5.7. Inferenzzeit der 2D und 3D CNNs 45

A.1. Architektur und Training des 2D CNN 59
A.2. Fehlermetriken der 3D CNNs in ZY- und XY-Ebene 60
A.3. Varianz und Standardabweichung des zellbasierten AEs der 2D und 3D CNNs . . 60
A.4. PFLOTRAN-Einstellungen in 2D und 3D 61

Abkürzungsverzeichnis

2D-1000 auf 1000 dp trainiertes 2D CNN. 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 48, 60

2D-100 auf 100 dp trainiertes 2D CNN. 35, 36, 37, 38, 39, 42, 43, 44, 45, 48, 60

3D-1000 auf 1000 dp trainiertes 3D CNN. 35, 36, 37, 39, 40, 41, 42, 44, 45, 60

3D-100 auf 100 dp trainiertes 3D CNN. 7, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 60

3D-1000-L1 auf 1000 dp trainiertes 3D CNN mit SmoothL1 Loss und $\beta = 0.1$. 5, 7, 9, 35, 39, 40, 41, 42, 43, 44, 45, 48, 50, 60

AAE Averaged Absolute Error. 3

Adam Adaptive Moment Estimation. 26

AE Absolute Error. 5, 9, 31, 32

BatchNorm Batch Normalisierung. 15

CNN Convolutional Neural Network. 3

Conv1 Convolution Kernel der Größe $1 \times 1 \times 1$. 29

Conv5 Convolution Kernel der Größe $5 \times 5 \times 5$. 29

CPU Central Processing Unit. 32

DCNN Deep Convolutional Neural Network. 16

Dim Dimension. 44

dp Datenpunkt. 3

FEFLOW Finite Element Subsurface FLOW Simulation System. 21

GPU Graphics Processing Unit. 14

LAHM Linear Advective Heat transport Model. 21

LBM Lattice Boltzmann Method. 22

LR Learning Rate. 9, 26

MAE Mean Absolute Error. 3

MaxPool Max-Pooling. 16

MIMO Multiple Input Multiple Output. 23

MSE Mean Squared Error. 25

PAHM Planar Advective Heat transport Model. 21

PAT Percentage Above Threshold. 3

PINN Physics-Informed Neural Network. 22

PINO Physics-Informed Neural Operator. 22

ReLU Rectified Linear Unit. 7, 16

RMSE Root Mean Squared Error. 3

RMSprop Root Mean Square Propagation. 26

SGD Stochastic Gradient Descent. 26

SHEMAT Simulator for HEat and MAss Transport. 21

TOM Thermische Online Modell. 21

UpConv Upconvolution. 29

VRAM Video Random Access Memory. 33

1. Einleitung

Der Klimawandel wird aufgrund seiner weitreichenden Auswirkungen auf Ökosysteme, menschliche Gesellschaften und die globale Wirtschaft als eine zentrale Herausforderung unserer Zeit gesehen [Cli22]. Zur Bekämpfung des Klimawandels bilden erneuerbare Energien unverzichtbare Ressourcen, um CO_2-Emissionen einzusparen und das Freisetzen von Treibhausgasen zu verhindern [Gio21]. Geothermische Energie zählt – neben Sonnenenergie, Windenergie, Wasserkraft und Biomasse – zu den erneuerbaren Energiequellen, da sie auf der Nutzung von Erdwärme basiert, die durch kontinuierlichen natürlichen Zerfall radioaktiver Elemente im Erdinneren stetig erneuert wird [US 24a]. Natürliche Warmwasservorkommen im Untergrund sowie relativ konstante Untergrundtemperaturen speisen sich aus dieser kontinuierlichen Wärmeproduktion [US 24a]. Für die Nutzung dieser geothermischen Ressourcen kommen spezielle Systeme zum Einsatz, wobei Wärmepumpensysteme eine zentrale Rolle einnehmen [US 24a]. Geothermische Wärmepumpen können die im Erdreich gespeicherte Energie sowohl zum Heizen als auch zum Kühlen von Gebäuden nutzen, indem sie Wärme zu- oder abführen. Da die Temperaturen im erdoberflächennahen Untergrund relativ konstant sind, eignen sich Wärmepumpen zur ganzjährigen Temperaturregulierung [US 24b].

Die in der vorliegenden Arbeit betrachteten Wärmepumpen sind Grundwasserwärmepumpen in Open-Loop-Systemen. Diese Wärmepumpen nutzen ein Zwei-Brunnen-System, das mit einem Förderbrunnen Grundwasser entnimmt und erwärmtes oder abgekühltes Wasser über einen Schluckbrunnen in das Grundwasser zurückführt. Über einen Wärmetauscher entziehen die Wärmepumpen dem Wasser dabei Wärmeenergie. Ein solches System wird als Open-Loop-System bezeichnet, da es direkt mit dem Grundwasser interagiert [US 24b]. Für die effiziente Nutzung von Grundwasserwärmepumpen sollten oberflächennahe, gut durchlässige Bodenschichten vorhanden sein, die ausreichende Mengen an Grundwasser führen und relativ konstante Temperaturen aufweisen [Min09]. Sind diese lokalen Voraussetzungen gegeben, können ganze Gemeinden mit geothermischer Energie versorgt werden [US 24a], indem ein umfangreiches Netzwerk aus Grundwasserwärmepumpen installiert wird. Zusätzlich ist die passende Positionierung der Brunnen bereits bei der Infrastruktur-Planung von großer Bedeutung [MP13]. Zu dicht platzierte Brunnen können zu unerwünschten Interferenzen führen. Das zurückgeleitete erwärmte Wasser könnte in Richtung des Förderbrunnens zurückfließen oder den Förderbrunnen eines benachbarten Systems in Fließrichtung des Grundwassers beeinflussen [MP13]. Dies würde die Effizienz der Wärmepumpen verringern, sofern das benachbarte System das Grundwasser für denselben Zweck nutzt. Daher ist es notwendig, die Ausbreitung und Interaktion von Wärmefahnen mit zu berücksichtigen.

Wie in der Arbeit von Pelzer und Schulte [PS24] dargelegt, stehen zwar verschiedene Ansätze zur Modellierung von Wärmefahnen zur Verfügung, jedoch sind diese mit bestimmten Einschränkungen verbunden. Simulationsbasierte Ansätze erfordern, abhängig von der modellierten Zeitdauer und dem Umfang des betrachteten Areals, aufwendige numerische Simulationen. Analytische Methoden hingegen sind auf die Modellierung einzelner Wärmefahnen beschränkt und berücksichtigen nicht deren Interaktionen.

Pelzer und Schulte [PS24] entwickelten für diese Herausforderung eine zweistufige Methode zur Vorhersage der Interaktion von Wärmepumpen unter dem Einsatz von Convolutional Neural Networks (CNNs) mit U-Net-basierter Architektur. In der ersten Stufe schätzt ein Netzwerk die lokale Temperatur ausgehend von einer einzelnen Wärmepumpe. Basierend auf dieser Vorhersage prognostiziert ein weiteres Netzwerk die Interaktion mehrerer Wärmepumpen. Die optimierten Netzwerke erhalten Inputdaten in 2D und erreichen in beiden Stufen einen Root Mean Squared Error von etwa 0.1 °C.

Eine Erweiterung dieses Ansatzes auf eine dritte räumliche Dimension ermöglicht die Erfassung von Tiefeninformationen benachbarter Zellen, die im zweidimensionalen Bereich verloren gehen [HES21]. Durch diese zusätzliche Dimension kann der dynamische Prozess der Wärmeausbreitung im Grundwasser realistischer modelliert werden, da sich dreidimensionale Inputparameter und spezielle 3D Effekte wie Niederschlag einbeziehen lassen.

CNNs in drei Dimensionen sind enorm leistungsfähig, jedoch aufwendiger zu optimieren und trainieren als ein 2D Modell, da die zusätzliche Dimension deutlich größere Datenmengen und eine höhere Rechenleistung erfordert [SWG+20]. Angesichts der kontinuierlichen Entwicklung moderner Graphics Processing Units (GPUs) und der Möglichkeit, Berechnungen effizient auf die GPU auszulagern, kann das Potential von 3D Netzwerken umfassender untersucht werden.

Daher analysieren wir in unserer Arbeit die Genauigkeit und den erforderlichen Rechenaufwand eines 3D Netzwerks im Kontext der Wärmefahnenvorhersage von Grundwasserwärmepumpen in Open-Loop-Systemen. Dazu erweitern wir das 2D CNN aus der Arbeit von Pelzer und Schulte [PS24] um eine räumliche Dimension, beschränkt auf die erste Stufe ohne Interaktion von Wärmefahnen.

Die 3D Inputdaten für unser Netzwerk sind mit der Software PFLOTRAN [LHL+15] simuliert. Das optimierte 3D CNN wird mit zwei Datensätzen mit 100 und 1000 Datenpunkten (dp) trainiert. Unser Fokus liegt auf dem Vergleich zwischen dem 2D Modell aus der Arbeit von Pelzer und Schulte [PS24] und unserem 3D Modell hinsichtlich des Speicherbedarfs, der Trainings- und Inferenzzeiten sowie der Metriken Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Averaged Absolute Error (AAE) und Percentage Above Threshold (PAT), dem prozentualen Anteil an Fehlerzellen über 0.1°C.

Die Arbeit beginnt in Kapitel 2 mit den relevanten Grundlagen zu CNNs und der speziellen Architektur U-Net. In Kapitel 3 befassen wir uns mit verwandten Arbeiten, die Vergleiche zwischen 2D und 3D CNNs sowie die Modellierung von Wärmefahnen behandeln. Anschließend erläutern wir in Kapitel 4 unsere Implementierung und Vorgehensweise bei der Netzwerkoptimierung. Wir gehen auf unsere verwendeten 3D Rohdatensätze ein, erklären die Datenvorbereitung für das Training und stellen Architektur und Trainingskonfiguration unseres 3D Modells vor. Zudem geben wir die verwendeten Auswertungsmetriken an und legen die grundlegenden Unterschiede in Architektur, Training und Datensätzen zwischen den 2D und 3D Modellen dar. In Kapitel 5 präsentieren wir die Ergebnisse unseres Experiments und diskutieren diese. In Kapitel 6 analysieren wir die Einschränkungen unserer Studie und setzen unsere Ergebnisse in den Kontext der aktuellen Forschung. Abschließend fassen wir die wichtigsten Erkenntnisse zusammen und stellen Anknüpfungspunkte für zukünftige Arbeiten vor. Im Anhang A sind zusätzliche Daten, Grafiken und Materialien enthalten, die die Ergebnisse der Arbeit unterstützen.

2. Grundlagen von CNNs und U-Net

In diesem Kapitel erklären wir anhand des Aufbaus und des Trainingsablaufs die Funktionsweise eines allgemeinen Convolutional Neural Networks (CNNs). Insbesondere gehen wir auf die einzelnen Komponenten eines CNNs ein. Im Anschluss erklären wir die spezielle Architektur U-Net [RFB15] und schließen mit einem kurzen Vergleich der 2D und 3D Convolution.

2.1. 2D CNN

Wir betrachten in diesem Abschnitt den Aufbau eines CNNs und beschreiben die verschiedenen Layer, die zur Merkmalsextraktion und Datenverarbeitung beitragen. Zudem erläutern wir den Trainingsprozess, bei dem ein CNN die Gewichte seiner Kernel lernt.

2.1.1. Aufbau

Der erste Layer eines CNNs ist der Input Layer, der die Inputdaten enthält. Der letzte Layer, der als Output Layer bezeichnet wird, erzeugt den Output eines Netzwerkes. Dazwischen befinden sich die Hidden Layer, die aus den verschiedenen Komponenten Convolutional Layer, Activation Layer und Pooling Layer bestehen [GBC16]. Die kategorisierende Komponente in einem CNN ist der Convolutional Layer, was seinen Namen erklärt. Ein Netzwerk wird als CNN klassifiziert, wenn es mindestens einen Convolutional Layer verwendet [GBC16].

Häufig werden in einem CNN auch Batch Normalisierungs (BatchNorms) Layer und finale Fully Connected Layer eingesetzt [AZH+21].

Convolutional Layer verwenden die lineare mathematische Operation Convolution. Wird die Convolution mit einem 2D Kernel der Größe 5×5 auf diskrete 2D Datenpunkte (x, y) angewandt, entspricht das einer diskreten mathematischen Faltung im 2D Raum mit dem Operator $*$ und $(f * g)(x, y) = \sum_{k=-2}^{2} \sum_{\ell=-2}^{2} f(x-k, y-\ell) g(k, \ell)$. Dabei ist g eine diskrete Gewichtsfunktion, der Kernel, und f eine diskrete Funktion der 2D Inputdaten. Um die Gesamtgröße der Inputdaten zu erhalten, werden zusätzliche Randwerte hinzugefügt. Dieses Verfahren wird als Padding bezeichnet.

Häufig wird zusätzlich ein gelernter Bias-Term zu den gewichteten Inputdaten addiert. Dadurch wird der Output des Aktivierungslayers verschoben, wodurch das Modell flexibler auf Inputdaten reagieren kann. Die Parameter eines Modells umfassen die Anzahl der Kernelgewichte und Bias-Terme, die während des Trainings optimiert werden [GBC16].

Eine Convolution bewirkt das Extrahieren eines Merkmals der Inputdaten, wie Kanten oder Texturen. Ihr Ergebnis ist eine Merkmalskarte [GBC16]. Sowohl in Netzwerken mit wenigen als auch mit vielen Convolutional Layern werden lokale Merkmale extrahiert. Deep Convolutional Neural Networks (DCNNs), die eine Multi-Layer Architektur haben, können zusätzlich globale Eigenschaften lernen [LLY+22].

Einem Convolutional Layer folgt typischerweise ein **Activation Layer**, der eine nichtlineare Aktivierungsfunktion auf die Werte der Merkmalskarte anwendet, um komplexe nichtlineare Beziehungen in den Daten zu modellieren [GBC16]. Ohne diese Nichtlinearität könnte das Netzwerk nur lineare Abhängigkeiten erkennen und selbst ein DCNN würde letztlich nur eine Linearkombination der Inputdaten erzeugen [LLY+22].

Ein Beispiel für eine Aktivierungsfunktion ist die Rectified Linear Unit (ReLU) [NH10], dargestellt in Abbildung 2.1. Im Vergleich zu traditionellen Aktivierungsfunktionen wie Sigmoid oder Tanh bietet die ReLU mehrere Vorteile. Da sie positive Werte beibehält und negative Werte einfach auf Null setzt, ist sie rechnerisch effizient und beschleunigt dadurch den Lernprozess. Zudem unterstützt sie die Konvergenz des Modells und wirkt in tiefen Netzwerken dem Verschwinden des Gradienten entgegen [LLY+22]. Ein Nachteil der ReLU ist ihre Abbildung von negativen Werten auf genau Null. Dies kann dazu führen, dass Kerneloutputs dauerhaft Null bleiben, wenn die Inputwerte konstant negativ sind. Aus solchen Regionen kann das Netzwerk dann keine Merkmale mehr extrahieren [LLY+22].

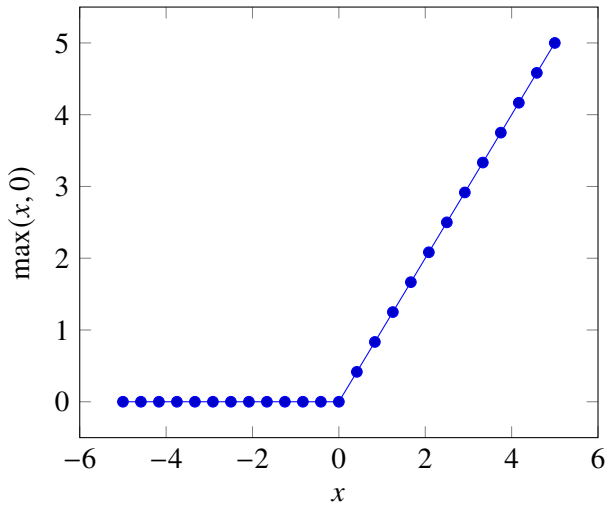

Abbildung 2.1.: ReLU Aktivierungsfunktion

Einem Convolutional Layer mit Aktivierung folgt typischerweise ein **Pooling Layer**, der die räumlichen Dimensionen des Inputs reduziert, um die Anzahl der Parameter und Rechenaufwand zu verringern [GBC16]. So extrahiert Max-Pooling (MaxPool) beispielsweise den Maximalwert innerhalb einer bestimmten Nachbarschaft. Alternative Methoden sind Min-Pooling oder Durchschnittsbildung. Die Komprimierung des Inputs beschleunigt den Lernprozess, wirkt Überanpassung entgegen [DV20] und bietet eine gewisse Invarianz bei kleinen Verschiebungen im Input [GBC16]. Ein Nachteil ist jedoch der Verlust an räumlicher Information.

Die **BatchNorm** [IS15] ist für das Training tiefer Netzwerke oft unerlässlich [TTC18]. Sie wirkt dem Verschwinden des Gradienten entgegen, indem der Output eines Convolutional Layer mithilfe von Mittelwert und Varianz eines Batches normalisiert wird [IS15]. Dabei wird das Netzwerk nicht nur in seinen Vorhersagen deutlich genauer, es benötigt auch ein Vielfaches weniger Trainingszeit und beschleunigt daher die Konvergenz [IS15]. Durch den Einsatz von BatchNorm reduziert sich zusätzlich der Einfluss der Initialisierung und es können größere Lernraten eingesetzt werden, ohne die Konvergenz zu beeinträchtigen. Zudem hat sie auch einen regularisierenden Effekt [IS15].

2.1.2. Training

Beim Training lernt ein CNN die Gewichte seiner Kernel, die aus dem Input eines Convolutional Layer eine gewichtete Summe bilden und so die Merkmalskarte erzeugen. Es gibt verschiedene Lernverfahren, die beim Training von CNNs angewandt werden können. In dieser Arbeit verwenden wir Supervised Learning. Bei diesem Verfahren erhält das Modell während des Trainings eine Grundwahrheit, die als Referenz für seine Vorhersagen dient. Die Abweichung zwischen der Grundwahrheit und den Vorhersagen des Modells wird von einer Lossfunktion gemessen. Dieser Fehler wird durch Backpropagation über das Netzwerk verteilt, um die Kernelgewichte und Bias-Terme zu aktualisieren [GBC16]. Dabei werden die Ableitungen der Lossfunktion nach den Gewichten und Bias-Termen berechnet [GBC16]. Basierend auf diesem Gradienten, passt ein Optimierungsalgorithmus, meist Gradient Descent, die Gewichte und Bias-Terme nach jedem Durchlauf eines Batches so an, dass der Fehler minimiert wird [GBC16]. Mit der Batchgröße kann daher gesteuert werden, wie oft diese Aktualisierungen innerhalb eines Trainingszyklus erfolgen [LLY+22].

Ziel beim Trainieren eines CNNs ist es, möglichst genaue Vorhersagen für beliebige, aber problemspezifische Inputdaten zu erhalten. Dabei wird die Fähigkeit eines Modells, bei unbekannten Daten präzise Vorhersagen zu liefern, Generalisierung genannt. Dafür ist oft eine zeitaufwendige Optimierung des CNNs erforderlich.

2.2. 2D U-Net CNN

Das von Ronneberger et al. [RFB15] vorgestellte 2D U-Net ist ein CNN mit U-förmiger Architektur und Skip Connections. Es besteht aus den in Abschnitt 2.1.1 erwähnten Basis-Layern, besitzt jedoch keine Fully Connected Layer, da diese räumliche Informationen nicht erhalten. Die charakteristische U-Form ergibt sich aus zwei komplementären Seiten: Auf der Kontraktionsseite (Encoder) wird Pooling angewendet, um die räumliche Größe der Merkmalskarten schrittweise durch Datenkomprimierung zu reduzieren. Auf der Expansionsseite (Decoder) wird Upconvolution, d. h. ein transponierter Convolutional Kernel eingesetzt, um die Auflösung des Outputs wieder zu verbessern [RFB15]. Die große Anzahl an Merkmalen, die Ronneberger et al. [RFB15] beim Upsampling einsetzen, ergibt eine fast symmetrische Architektur. Skip Connections verbinden die Kontraktions- mit der Expansionsseite und ermöglichen es, detaillierte Merkmale aus früheren Layern direkt an spätere Layer weiterzugeben. Diese Merkmale werden mit den Kontextinformationen aus tieferen Layern kombiniert, wodurch eine präzise Lokalisierung der Objekte möglich wird [RFB15]. Daher ist das 2D U-Net besonders effizient für Bildsegmentierungsaufgaben [RFB15].

2.3. 2D vs. 3D Convolution

Bei der Anwendung der Convolution wird ein Convolutional Kernel über gitterartig strukturierte Inputdaten bewegt [GBC16]. Ein 2D Convolution Kernel hat eine Höhe und Breite. Er wird daher, wie in Abbildung 2.2 oben zu sehen, in zwei Richtungen über ein Bild bewegt. Bei einem 3D Convolutional Kernel kommt die Dimension Tiefe hinzu. Er ist typischerweise würfelförmig und seine Gewichte werden mit Voxel- anstatt Pixeldaten trainiert. Hierzu wird der 3D Kernel in alle drei Dimensionen über die Voxeldaten bewegt und an jeder Position der Inputdaten eine gewichtete Summe der Voxelwerte im Bereich des Kernels berechnet. Die Gleichung aus Abschnitt 2.1.1 erhält damit eine zusätzliche Dimension: $(f * g)(x, y, z) = \sum_{k=-2}^{2} \sum_{\ell=-2}^{2} \sum_{m=-2}^{2} f(x-k, y-\ell, z-m) g(k, \ell, m)$. Der Kernel g ist nun dreidimensional mit Größe $5 \times 5 \times 5$ und f ist ein 3D Volumen, das an den Koordinaten (x, y, z) abgetastet wird.

Bei der 3D Convolution wird Information in alle drei Dimensionen verarbeitet. Damit erhalten auf 3D Daten operierende 3D CNNs die Fähigkeit Tiefeninformation zu erlernen, welche ein 2D CNN nicht oder nur gemittelt erlernen kann. Ein 3D CNN arbeitet nicht nur mit 3D Convolution, sondern auch mit 3D Pooling und einer 3D Aktivierungsfunktion. Bei der Erweiterung eines 2D Netzwerkes werden daher die Operationen von 2D auf 3D umgestellt.

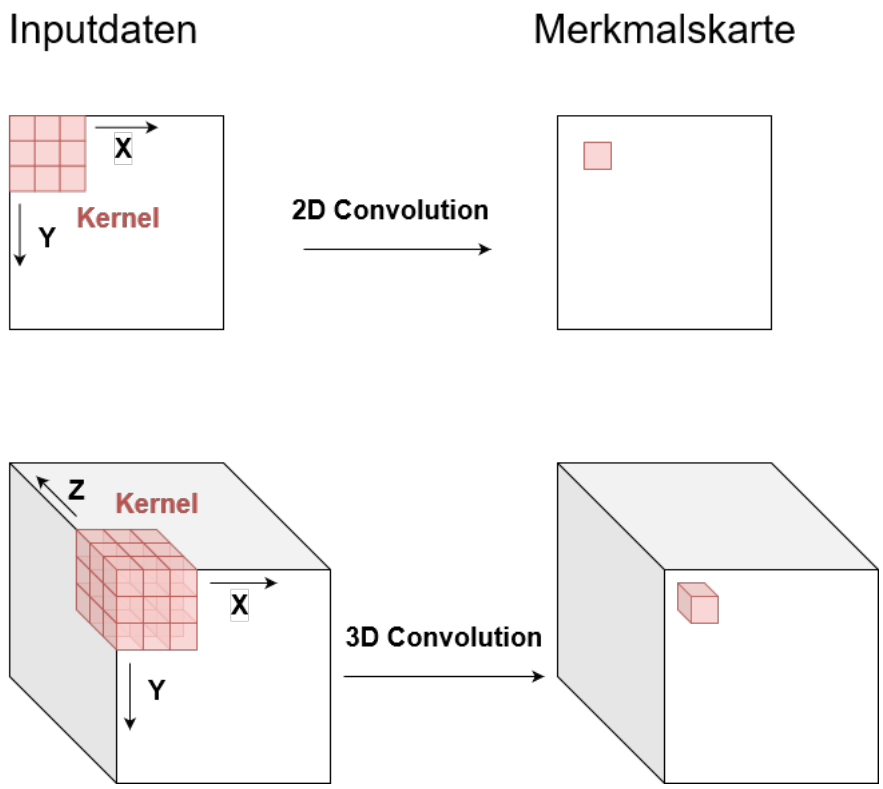

Abbildung 2.2.: Vergleich der Convolution in 2D und 3D: Auf der linken Seite sind die Inputdaten zu sehen, rechts das Gesamtergebnis der Convolution als Merkmalskarte, nachdem der Kernel über die vollständige Inputdaten geschoben wurde. Der Kernel mit Beispielgröße 3 pro Dimension ist auf der linken Seite in rosa dargestellt. Das Ergebnis einer einzelnen Convolution ist auf der rechten Seite in rosa dargestellt. Die Pfeile X, Y und Z geben die Dimensionsrichtungen an, in die der Kernel geschoben wird. Um die Gesamtgröße der Inputdaten zu erhalten sind zusätzliche Randwerte nötig.

3. Verwandte Arbeiten

In diesem Kapitel widmen wir uns dem aktuellen Forschungsstand zur Modellierung von Wärmefahnen im Grundwasser im Kontext von Grundwasserwärmepumpen in Open-Loop-Systemen. Anschließend betrachten wir Studien, die sowohl CNNs als auch die U-Net Architektur für einen Vergleich der Leistung zwischen 2D und 3D Modellen heranziehen.

3.1. Modellierung von Wärmefahnen

Zur Modellierung von Wärmefahnen durch Grundwasserwärmepumpen können analytische und numerische Ansätze verwendet werden. Auch datengetriebene Modellierungsansätze haben in jüngster Zeit Erfolge erzielt und werden zunehmend untersucht.

Zu den analytischen Modellen gehören unter anderem das Linear Advective Heat transport Model (LAHM) [Kin87], welches das Temperaturfeld einer kontinuierlichen Punktquelle beschreibt und das Planar Advective Heat transport Model (PAHM) [DR85], das von einer planaren Wärmequelle ausgeht. Beide Transportmodelle sind traditionell für 2D Ansätze ausgelegt. Das PAHM wurde von Hähnlein [HMB+10] weiterentwickelt, sodass es auch für 3D Anwendungen eingesetzt werden kann [OKK+22]. Diese Modelle basieren auf vereinfachten Annahmen und neigen dazu, die Ausbreitung der Wärmefahnen im Vergleich zu numerischen Modellen zu überschätzen [OKK+22].

Ein beliebtes numerisches Simulationsmodell für Grundwasserströmungen und Wärmetransport ist das Finite Element Subsurface FLOW Simulation System (FEFLOW) [Die13]. Das Modell nutzt eine mehrdimensionale Finite Elemente Methode und erlaubt die Modellierung komplexer geologischer Systeme. Es kommt in 2D und 3D Grundwassersimulationen zum Einsatz, unter anderem in [LC09; NO10; PAB+20; PBT+20]. Auch OpenGeoSys [KBB+12] nutzt die Finite Element Methode zur Berechnung gekoppelter multiphysikalischer Prozesse in porösen und geklüfteten Medien. Simulator for HEat and MAss Transport (SHEMAT) [Cla03] und seine Weiterentwicklung SHEMAT-Suite [KRB+20] verwenden die Finite Differenzen Methode zur numerischen Lösung von Strömungs- und Transportprozessen. PFLOTRAN [LHL+15] ist ein weiteres numerisches Simulationsprogramm zur Modellierung von Strömungen und reaktiven Transportprozessen in porösen Materialien. Es nutzt eine Finite Volumen Methode und ist speziell für parallele Berechnungen optimiert.

Ein anderes Modell stellt einen Kompromiss zwischen vereinfachten analytischen Modellen und rechenintensiven Planungsmodellen dar. Das in Baden-Württemberg entwickelte webbasierte Thermische Online Modell (TOM) ist ein stark vereinfachtes numerisches Grundwassermodell zur Berechnung von Temperaturfahnen im Grundwasser [OKK+22]. Es verwendet DuMu^x [FDE+11], einen numerischen Simulator für Strömungs- und Transportprozesse in porösen Medien. Eine Studie von Ohmer et al. [OKK+22] zeigt, dass TOM die analytischen Modelle LAHM und PAHM deutlich übertrifft und vergleichbare Ergebnisse erzielt wie das FEFLOW Modell. Für kleinere Wärmepumpanlagen mit Förderraten $< 10\frac{\ell}{s}$ wird TOM daher als ausreichend genau empfohlen.

Insgesamt sind jedoch einfache analytische Modelle und hochpräzise numerische Grundwassersimulationen entweder zu ungenau oder zu rechenintensiv für eine alltägliche Nutzung [DLPS23]. Um dieser Herausforderung zu begegnen, werden datengetriebene Modellierungsansätze mit Neural Networks untersucht, die eine schnellere und dennoch präzise Modellierung ermöglichen können.

Eine Methode sind Physics-Informed Neural Networks (PINNs) [RPK17], die in diesem Zusammenhang eingesetzt werden können. PINNs beziehen physikalische Gesetze in Form von Differentialgleichungen in den Lernprozess von Neural Networks ein und verringern so die benötigte Trainingsdatenmenge. In der Arbeit von [Sch22] werden sie im 2D Bereich verwendet, um die Übergänge zwischen zusammengefügten lokalen Wärmefahnenvorhersagen zu korrigieren. Physics-Informed Neural Operators (PINOs) [LZK+24] erweitern den Ansatz der PINNs, indem sie anstelle einer einzelnen Lösung einen gesamten Lösungsoperator lernen. Dadurch können sie besser auf größere Bereiche generalisieren. In der Arbeit von Göhring [Göh23] wird ein PINO für die dreidimensionale Modellierung einer Wärmefahne implementiert und dessen Genauigkeit sowie Datenbedarf untersucht. Ein datengesteuerte Ansatz erweist sich mit nur 50 Datenpunkten und ausreichender Genauigkeit als effizient, während ein rein physikgetriebener Ansatz diese Genauigkeit nicht erreichen kann.

Andere Arbeiten setzen CNNs zur Vorhersage der Temperaturverteilung von Wärmefahnen ein. Davis et al. [DLPS23] präsentieren in ihrer Studie drei Varianten einer modifizierten 2D U-Net Architektur, die Darcy-Geschwindigkeitsfelder als Eingabedaten erhalten. Diese CNNs sind in der Lage realistische Temperaturfelder unter geringerem Rechenaufwand als hochpräzise Löser vorherzusagen. Allerdings stellt die genaue Vorhersage der Form von Wärmefahnen entlang der Strömungsrichtung in einigen Fällen eine Herausforderung für die Netzwerke dar. Pelzer und Schulte [PS24] entwickeln ebenfalls ein 2D U-Net, das auf Basis messbarer Eigenschaften aus dem Untergrund wie Druckgradient und Permeabilität die Temperaturfelder vorhersagt. In einem zweistufigen Ansatz wird zunächst eine lokale Wärmefahne modelliert, anschließend prognostiziert ein weiteres CNN die Interaktion zwischen den Wärmefahnen. Die Autoren erzielen dabei eine hohe Genauigkeit.

Die vorgestellten Arbeiten sind auf den 2D Bereich beschränkt. Eine verwandte Arbeit zur Modellierung von Temperaturfeldern mit einem 3D CNN, ist die Arbeit von Wang et al. [WWY+22]. Die Autoren verwenden eine U-Net-basierte 3D Architektur sowie Datensätze, die mit der Lattice Boltzmann Method (LBM) [QCZK20] simuliert wurden, um Temperaturfelder und die zugehörige effektive Wärmeleitfähigkeit in allgemeinen porösen Medien unter verschiedenen Randbedingungen vorherzusagen. Das entwickelte CNN zeigt eine hohe Lernfähigkeit und erzielt geringe relative Fehler bei deutlich kürzerer Trainingszeit im Vergleich zur LBM-Simulation. Im Gegensatz dazu fokussieren wir uns auf den konkreten Anwendungsfall zur Wärmefahnenvorhersage von Grundwasserwärmepumpen in Open-Loop-Systemen und verwenden hydrogeologische Eingabeparameter wie Permeabilität und Druckgradient.

3.2. Vergleiche zwischen 2D und 3D CNNs

Dem effizienten Training von 2D CNNs mit geringerem Ressourcenaufwand steht das Potenzial von 3D CNNs gegenüber, die Genauigkeit aufgrund der Tiefeninformation von 3D Daten signifikant zu steigern [EGB+23; PGT+22]. In den letzten Jahren wurde zunehmend untersucht, inwiefern 3D CNNs in der Videoklassifikation, Objekterkennung und weiteren Bereichen die Leistung von 2D Ansätzen übertreffen können – mit unterschiedlichen Ergebnissen.

Tran et al. [TBF+15] zeigen, dass 3D CNNs in der Videoklassifikation 2D Modelle bei der Genauigkeit übertreffen. Von ähnlichen Ergebnissen berichten Li [Li17] im Bereich der Objekterkennung aus Punktwolken, Marinberg et al. [MCBP20] beim Schätzen von Multiple Input Multiple Output (MIMO) Kanälen und Yu et al. [YYW+20] bei der Klassifizierung von Lungenknötchen anhand von CT-Daten.

Neuere Studien zeigen die verbesserte Genauigkeit der 3D CNNs bei der Hirntumorsegmentierung aus MRT-Bildern [PGT+22] und bei der Klassifikation von Druckstellen auf Orangen mit hyperspektralen Bilddaten [PSZ+23].

Zudem wurde die Überlegenheit von 3D U-Nets bei der volumetrischen Bildsegmentierung demonstriert: In der Studie von Çiçek et al. [ÇAL+16] lernt das 3D U-Net eine dichte 3D Segmentierung aus der Markierung weniger Bildebenen und übertrifft sein 2D Pendant. Hassanzadeh et al. [HES21] zeigen die hohe Präzision eines 3D U-Nets, dessen Netzwerkstruktur zunächst in 2D mit einem evolutionären Algorithmus optimiert und anschließend durch den Austausch von 2D mit 3D Operationen in den 3D Bereich konvertiert wird. Dieser Ansatz führt bei der Optimierung zu erheblichen Zeit- und Recheneinsparungen.

Allerdings sind die Ergebnisse in der Literatur nicht konsistent und in anderen Anwendungsfällen erreichen 3D CNNs keine Genauigkeitsverbesserung gegenüber 2D Architekturen.

So stellen Kurmanji und Ghaderi [KG19] bei der Klassifizierung von Handgesten aus RGB-D-Daten und Stamoulakatos et al. [SCM+21] bei der Bildklassifizierung von Unterwasservideos fest, dass 2D Modelle überlegen sind. Diese Ergebnisse bestätigen sich auch bei der Emotionserkennung in Videos mit flachen CNNs [DHT23] sowie bei der semantischen Segmentierung auf MRT- und CT-Daten mithilfe eines auf ResUnet-basierten Modells [CLS22].

Ein grundlegendes Problem für 3D Modelle ist die begrenzte Verfügbarkeit von Trainingsdaten [CLS22; EGB+23]. Zudem erhalten 2D Modelle, die in einigen Vergleichen bessere Ergebnisse erzielen, eine deutlich größere Anzahl an Inputdaten, da sie einzelne 2D Bildebenen aus 3D Volumina nutzen und dadurch einen zahlenmäßig größeren Datensatz aufbauen können [CLS22; DHT23]. Insgesamt zeigen die Vergleiche, dass 3D Modelle erheblich mehr Rechenaufwand, längere Trainingszeiten und höhere Speicherkapazitäten erfordern, da die zusätzliche Dimension größere Modelle und Datensätze bedingt. Ein Zugewinn an Genauigkeit rechtfertigt daher nicht immer den hohen Ressourcenverbrauch [YYW+20]. Es bedarf weiterer Forschung, um das Potential von 3D Ansätzen effizienter auszuschöpfen [EGB+23].

4. Methodik

In diesem Abschnitt geben wir zunächst einen kurzen Überblick über die Implementierung unseres 3D CNNs. Anschließend erläutern wir unser Vorgehen bei der Hyperparameteroptimierung. Es folgt eine Beschreibung der von uns verwendeten Daten sowie deren Vorbereitung. Weiterhin werden die Architektur und das Training unseres 3D CNNs detailliert beschrieben, ebenso wie die angewendeten Metriken zur Evaluation.

4.1. Implementierung

Wir erweitern den Code von Pelzer [Pel24b] für ein 2D CNN um eine weitere Dimension. Dazu gehören die Erstellung eines 3D Netzwerks, die Datenvorbereitung sowie die Visualisierung und das Messen von 3D Daten. Die Implementierung ist nur auf der ersten Stufe anwendbar, d. h. für das Erlernen von Wärmefahnen ohne Interaktion.

Um ein 3D Modell zu erstellen, haben wir die 2D Operationen durch ihre entsprechenden 3D Varianten ersetzt. Für die Implementierung verwenden wir die Programmierbibliothek PyTorch [AYH+24], die speziell für die Entwicklung neuronaler Netze konzipiert ist. Dieses Python-Package nutzt GPU-Beschleunigung, um die Leistungsfähigkeit und Effizienz bei komplexen Berechnungen zu steigern.

Zur Visualisierung der 3D Vorhersagen schneiden wir die XY- und ZY-Ebene auf Höhe des Injektionspunktes aus. Dafür verwenden wir die für 2D Grafiken ausgelegte Python-Bibliothek Matplotlib [Hun07]. Für die Messung von Metriken auf 3D Daten extrahieren wir ebenfalls die XY- und ZY-Ebene auf Höhe des Injektionspunktes. Diese werden für einige Metriken zugeschnitten, wie in Abschnitt 4.6.1 erläutert.

Unser Code ist auf Github `https://github.com/ja-tk/1HP_NN/tree/release_24` verfügbar.

4.2. Hyperparameteroptimierung

Als Grundlage für unsere 3D Netzwerkarchitektur verwenden wir die 2D U-Net-basierte Architektur aus der Arbeit von Pelzer und Schulte [PS24], deren Hyperparameter wir anschließend optimieren. Für die Optimierung nutzen wir den Datensatz mit 100 dp und übernehmen die Netzwerkanpassungen anschließend für das Training mit 1000 dp. Als Lossfunktion wählen wir das Mean Squared Error (MSE) Loss wie im 2D Fall mit Definition im Anhang A.7 und eine Epochenanzahl von 200. Wir behalten die Modellkonfiguration mit dem besten erreichten Validation Loss aus den jeweiligen Optimierungen bei.

Die maximale Batchgröße wird zu Beginn auf 6 festgelegt, da sie durch den GPU-Speicher limitiert ist. Eine Zweierpotenz kann zu einer besseren Ausnutzung der GPU-Leistung führen [GBC16; LLY+22], daher testen wir zusätzlich die Batchgröße 4. Diese schneidet sowohl bei der Trainingszeit pro Epoche als auch beim Validation Loss schlechter ab. Die typischerweise eingesetzten Zweierpotenzen ab 32 [GBC16] sind speicherbedingt nicht umsetzbar.

Eine gleichzeitige Optimierung aller Hyperparameter ist aufgrund des exponentiell wachsenden Suchraums und der damit verbundenen enormen Anzahl an Experimenten ineffizient [LLY+22]. Nach Li et al. [LLY+22] sollte die Learning Rate (LR) zuerst optimiert werden, da sie das Training maßgeblich beeinflusst. Daher suchen wir zu Beginn nach einer geeigneten initialen LR. Aufgrund der geringen Batchgröße erhöht sich die Varianz der Gradienten [GBC16]. Aus Stabilitätsgründen und aufgrund von Lernschwierigkeiten des Modells reduzieren wir die initiale LR von $1 \cdot 10^{-4}$ auf $5 \cdot 10^{-5}$.

Für das weitere LR-Scheduling setzen wir den LR-Scheduler ReduceLROnPlateau aus Pytorch [AYH+24] ein. Dieser Scheduler senkt die LR, wenn das Validation Loss über eine festgelegte Anzahl von Epochen stagniert. Wir experimentieren mit verschiedenen Einstellungen, die finalen Werte des LR-Schedulers sind in Tabelle 4.1 aufgeführt. Dabei gibt der Factor an, mit welchem Wert die aktuelle LR multipliziert wird, falls sich keine Besserung ergibt. Cooldown ist die Anzahl der Epochen, die nach einer LR-Reduzierung abgewartet wird, bevor der LR-Scheduler wieder aktiv wird. Patience gibt die Anzahl der Epochen an, die der LR-Scheduler überwacht. Tritt keine Verbesserung ein, reduziert der LR-Scheduler die LR. Min-LR setzt eine untere Grenze der LR für alle Parameter.

Parameter	**Wert**
Factor	0.5
Patience	10
Cooldown	10
Min-LR	$1 \cdot 10^{-6}$

Tabelle 4.1.: Einstellungen für den LR-Scheduler ReduceLROnPlateau von PyTorch [AYH+24]

Zur Optimierung der Hyperparameter Tiefe, initiale Merkmalsanzahl, Kernelgröße und Optimierungsverfahren nutzen wir das Framework Optuna [ASY+19]. Dabei handelt es sich um eine Python-Bibliothek, die eine automatische und effiziente Hyperparametersuche für benutzerdefinierte Suchräume ausführt. Unter Verwendung von PyTorch [AYH+24] erweitern wir die vorhandene 2D Implementierung von Pelzer [Pel24a] auf Github. Die Erweiterung ist auf `https://github.com/ja-tk/1HP_NN/tree/optuna` verfügbar.

Die mit Optuna untersuchten Hyperparameterbereiche sind in Tabelle 4.2 aufgeführt. Zur Optimierung des Optimierungsverfahrens wählen wir Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop) und Stochastic Gradient Descent (SGD) [Sun20]. Wir merken an, dass das Optimierungsverfahren SGD in unserem Fall konstant deutlich schlechtere Ergebnisse liefert als die anderen beiden Optimierungsverfahren, sodass wir es im weiteren Verlauf der Hyperparameteroptimierung ausschließen.

Parameter	Wert
Optimierungsverfahren	[Adam, SGD, RMSprop]
Tiefe	[1, 2, 3]
Anzahl an Merkmalen, initial	[8, 16, 32]
Kernelgröße	[3, 4, 5]

Tabelle 4.2.: Einstellungen für die Hyperparameteroptimierung mit Optuna

Um eine geeignete initiale Anzahl von Merkmalen zu finden, setzen wir Zweierpotenzen ein. Aufgrund hardwareseitiger Einschränkungen durch die GPU begrenzen wir die maximale initiale Merkmalsanzahl auf 32.

Insgesamt erzielen wir mit Kernelgröße 5, Tiefe 3, Merkmalsanzahl 16 und dem Optimierungsverfahren Adam das geringste MSE Loss auf den Validationsdaten. Diese Konfiguration wählen wir für unser Netzwerk.

Eine weitere wichtige Optimierung für tiefe Netzwerke ist eine geeignete Gewichtsinitialisierung (Weight Initialization), um zu verhindern, dass sich Teile des Netzwerkes ungleichmäßig beteiligen [RFB15]. Unsere verwendete Aktivierungsfunktion ist ReLU. Da es sich um keine symmetrische Funktion handelt, wird bei der Verwendung einer ReLU die Kaiming-Normal-Initialisierung, auch He-Initialisierung, empfohlen [HZRS15]. Ronneberger et al. [RFB15] empfehlen diese Initialisierung ebenfalls für ihre U-Net Architektur. Es handelt sich um die Normalverteilung $N(0, \frac{2}{n})$, wobei n die Anzahl der Gewichte ist, die ein Filter auf die Inputdaten eines Convolutional Layers anwendet. Zur Berechnung wird die Anzahl der eingehenden Kanäle C mit den Dimensionen X, Y, Z des Kernels multipliziert: $n = C \times X \times Y \times Z$. Bei einer Kernelgröße von $5 \times 5 \times 5$ und vier Inputkanälen sind das zu Beginn $5 \cdot 5 \cdot 5 \cdot 4 \ = \ 500$ Gewichte, was der Normalverteilung $\mathcal{N}(0, 0.004)$ entspricht.

Wir untersuchen die Kaiming-Normal-Initialisierung, deren Varianz zwischen den Convolutional Layern variieren kann sowie die Verteilung $\mathcal{N}(0, 0.004)$ mit konstanter Varianz, wie es ähnlich im 2D Fall gemacht wurde. Die Versuche werden mehrfach durchgeführt, führen aber zu keiner Verbesserung des Validation Loss, sodass wir an der ursprünglichen Einstellung von $\mathcal{N}(0, 0.02)$ für alle Layer festhalten.

4.3. Datensatz

Die verwendeten 3D Datensätze sind mit der Simulationssoftware PFLOTRAN [LHL+15] erstellt und werden von Pelzer zur Verfügung gestellt. Die beiden Datensätze mit 100 und 1000 dp sind auf DaRUS verfügbar (https://doi.org/10.18419/darus-4533).

Eine Simulation hat fünf Zeitstempel und geht über 27.5 Jahre, um sicherzustellen, dass die Wärmefahne einen Gleichgewichtszustand erreicht hat. Der erste Zeitstempel liegt vor dem Eintritt des erwärmten Wassers ins Grundwasser und dient zum Einlesen der Inputdaten. Die Temperatur aus dem letzten Zeitstempel nutzen wir als Grundwahrheit. Es handelt sich um eine 3D Box mit $64 \times 256 \times 64$ Zellen und den Temperaturwerten der Wärmefahne, wie in Abbildung 4.1 zu sehen. Alle Achsenlängen einer Zelle entsprechen 5 m in der Simulation. Daher deckt eine Box einen 3D Bereich von $320\ m \times 1280\ m \times 320\ m$ ab.

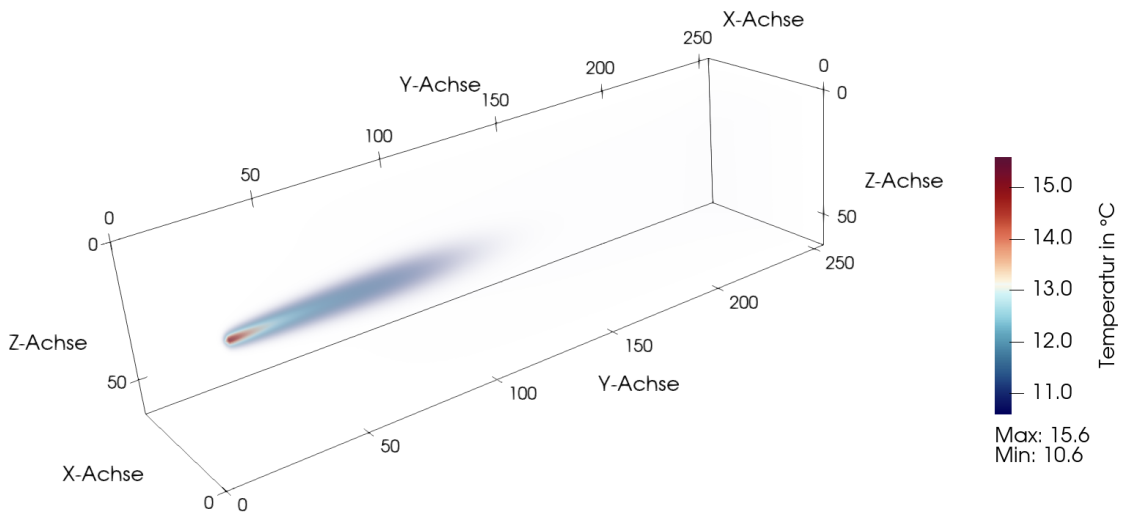

Abbildung 4.1.: Visualisierung einer PFLOTRAN-simulierten Wärmefahne mit ParaView [AGL05]: Die Achsen geben die Anzahl der Zellen in jede Richtung an. Die Box mit $64 \times 256 \times 64$ Zellen entspricht einer Größe von $320\ m \times 1280\ m \times 320\ m$ im Raum. Die Temperaturskala geht von dunkelrot bis dunkelblau über einen Bereich von 10.6 bis 15.6 °C und gibt die Temperaturwerte der Wärmefahne an. Im dunkelroten Bereich an Zellenposition $31 \times 21 \times 31$ liegt der Injektionspunkt des erwärmten Wassers.

Unsere Datensätze sind wie im 2D Fall [PS24] pysikalisch konsistent und mit realistischen Wertebereichen simuliert. Die Einstellungen sind aber einfach gehalten: Der Injektionspunkt innerhalb der Boxen ist über die Datensätze hinweg konstant. Permeabilität und Druckgradient sind je Datenpunkt konstant, variieren aber zwischen den Datenpunkten. Die Permeabilität nimmt Werte zwischen $1.05 \cdot 10^{-11}$ und $4.71 \cdot 10^{-9}\ m^2$ an, der Druckgradient variiert zwischen -0.0035 und -0.0015. Die Injektionsrate beträgt $0.24\ \cdot 10^{-3}\ \frac{\ell}{s}$ und die Injektionstemperatur 15.6 °C. Weitere Details können den Rohdaten auf DaRUS entnommen werden (https://doi.org/10.18419/darus-4533).

4.4. Datenvorbereitung

Um das Training zu optimieren und unabhängige Leistungsmessungen sicherzustellen, wird jeder Datensatz in Trainings-, Validations- und Testdatensatz im Verhältnis [0.7, 0.2, 0.1] aufgesplittet. Die Daten werden in Tensorform gebracht und normalisiert, d. h. auf das Intervall [0,1] abgebildet. Der Input für das CNN ist ein 5D Tensor mit den Dimensionen Batchgröße, Anzahl an Kanälen, Anzahl Zellen in X-, in Y- und in Z-Richtung.

Die Inputparameter sind in Abbildung 4.2 beispielhaft an einem Datenpunkt dargestellt. Sie umfassen den One-Hot-codierten Injektionspunkt des zurückgeführten Grundwassers ("i"), Permeabilität ("k"), Druckgradient ("g") sowie eine ausgewertete Signed-Distance-Function ("s"). Letztere gibt den Abstand von jeder Zelle zum Injektionspunkt an. Die Abkürzungen innerhalb der Klammern stehen für die jeweiligen Inputparameter.

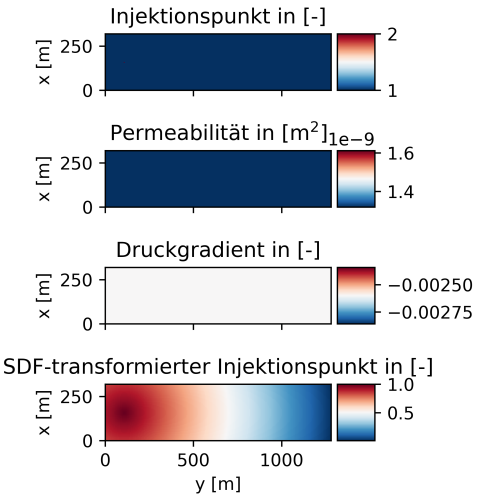

Abbildung 4.2.: Inputparameter des 3D CNNs als Schnitt in der XY-Ebene auf Z-Höhe des Injektionspunktes. Die X-Achse gibt die Breite, die Y-Achse die Länge der 3D Wärmefahnenbox in [m] an.

4.5. Architektur und Training

In diesem Kapitel stellen wir die optimierte 3D Netzwerkarchitektur vor und gehen auf unsere Trainingskonfiguration ein.

4.5.1. Netzwerkarchitektur

In unserer Arbeit verwenden wir ein Modell mit U-Net Architektur [RFB15]. Die Struktur unseres 3D CNN wird in Abbildung 4.3 dargestellt. Das Netzwerk hat eine U-Net Block-Tiefe von drei und Convolution Kernel der Größe $5 \times 5 \times 5$ (Conv5) mit Stride $1 \times 1 \times 1$. Zur Aktivierung wird ReLU eingesetzt. Das gesamte Netzwerk umfasst sieben U-Net Blöcke. Jeder U-Net Block ist aus mehreren Layern aufgebaut: Conv5 mit ReLU, Conv5 mit BatchNorm und ReLU, Conv5 mit ReLU. Das CNN hat vier Inputkanäle und einen Outputkanal. Zu Beginn wird die Merkmalsanzahl auf 16 gesetzt und mit jedem U-Net Block verdoppelt, beim Upsampling halbiert. Jedem Encoderblock folgt ein MaxPool mit Kernelgröße $2 \times 2 \times 2$ und Stride $2 \times 2 \times 2$, jedem Decoderblock geht eine Upconvolution (UpConv) voraus. Die Encoderseite ist mit der Decoderseite über einen U-Net Block verbunden. Insgesamt besteht das Netzwerk aus 21 Convolutional Layer und einem zusätzlichen Convolution Kernel der Größe $1 \times 1 \times 1$ (Conv1), um die am Ende erhaltenen 16 Merkmalskarten zur räumlichen Temperatur zusammenzufassen.

Um Vergleichbarkeit zu gewährleisten, belassen wir möglichst viele Einstellungen auf den Werten des ursprünglichen 2D CNN [PS24]. Für den Rand verwenden wir Null-Padding, um die Dimensionen der Daten beizubehalten. Die Dilation ist auf 1 eingestellt und der Bias aktiviert.

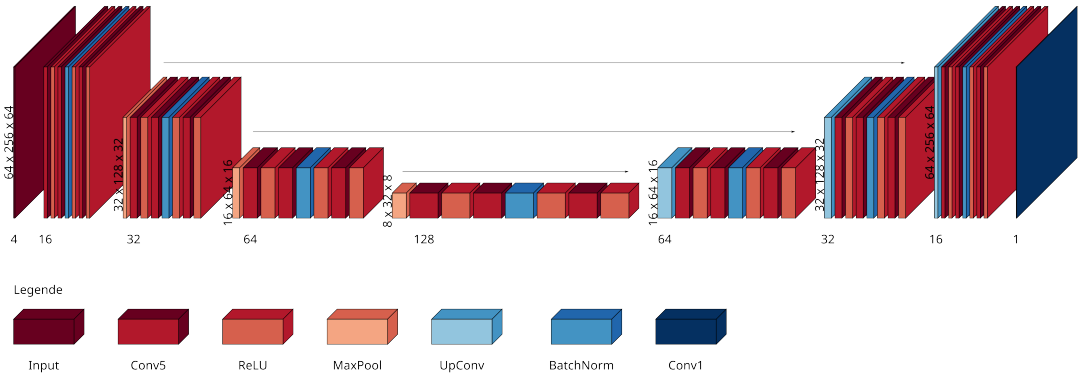

Abbildung 4.3.: Netzwerkarchitektur des 3D CNN. Das U-Net besteht aus sieben U-Net Blöcken. Jeder U-Net Block ist aus mehreren Layern aufgebaut, die der Legende entnommen werden können. Nach jedem Encoder U-Net Block wird MaxPool eingesetzt, vor jedem Decoder U-Net Block UpConv. Die Pfeile entsprechen Skip Connections. Die Zahlen unterhalb der Blöcke stehen für die Anzahl an Merkmalen oder der Input- und Outputkanäle. Die Zahlen links der Blöcke zeigen die Dimensiongrößen in X-, Y- und Z-Richtung an. Um die Anschaulichkeit zu erhöhen, wurde auf die Darstellung der dritten Dimension Z verzichtet. (based on: [PS24])

4.5.2. Training

Eine Übersicht der Trainingskonfiguration befindet sich in Tabelle 4.3. Wir trainieren mit dem Optimierungsverfahren Adam und einer Batchgröße von 6. Die initiale LR beträgt $5 \cdot 10^{-5}$ und wird schrittweise während des Trainings reduziert. Dazu verwenden wir den LR-Scheduler ReduceLROnPlateau mit den Einstellungen aus Tabelle 4.1. Die zu minimierende Lossfunktion ist anfänglich MSE. Für das 3D Modell mit 1000 dp wechseln wir im späteren Verlauf auf SmoothL1 mit $\beta = 0.1$. Die Definitionen der Lossfunktionen befinden sich im Anhang A.7. Pro Epoche vergleichen wir das neu erreichte Validation Loss mit dem bisher erreichten Bestwert. Ein Modell mit besserem Validation Loss wird abgespeichert. Für unsere Analyse wählen wir das zuletzt gespeicherte Modell, sofern nichts anderes angegeben. Für das Training legen wir eine maximale Epochenanzahl von 5400 für 100 dp und 600 für 1000 dp fest und setzen Early Stopping ein, sobald keine signifikante Verbesserung im Validation Loss mehr zu beobachten ist.

Lossfunktion und Anpassungen. Da hohe Fehlerwerte die Gesamtleistung des Modells negativ beeinflussen, testen wir in einem zusätzlichen Trainingsdurchlauf die SmoothL1 Lossfunktion mit $\beta \in \{0.1, 1\}$, um das Modell unempfindlicher gegenüber Ausreißern zu machen und wählen das SmoothL1 Loss mit $\beta = 0.1$ in der finalen Trainingskonfiguration für unsere 3D Modell mit 1000 dp.

Parameter	Wert
Batchgröße	6
Epochen	fallabhängig (siehe Kapitel 5)
Loss	MSE (zu Beginn) SmoothL1, $\beta = 0.1$ (finale Anpassung für 3D, 1000 dp)
Learning Rate	$5 \cdot 10^{-5}$ initial LR-Scheduler
Optimierungsverfahren	Adam
Betas	(0.9, 0.999)
Weight Decay	$1 \cdot 10^{-4}$
Weight Initialization	$\mathcal{N}(0, 0.02)$
BatchNorm	$\mathcal{N}(1, 0.02)$
Bias Initialization	0

Tabelle 4.3.: Trainingskonfiguration des 3D CNN

4.6. Bewertungsmetriken

Um die Leistungsfähigkeit eines CNNs zu bewerten, evaluieren wir die Vorhersagen auf den Testdaten unter Verwendung der Metriken Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), zellbasierter Averaged Absolute Error (AAE) sowie Percentage Above Threshold (PAT), die nachfolgend detailliert beschrieben werden. Zusätzlich untersuchen wir den Root Mean Squared Error (RMSE) und Mean Absolute Error (MAE) auch auf den Trainings- und Validationsdaten, um Rückschlüsse auf Überanpassung und Generalisierungfähigkeit des Modells zu ziehen. Zudem geben wir für die Testdaten Varianz und Standardabweichung des zellbasierten Absolute Errors (AEs) im Anhang A.5 an.

Zur Analyse des Rechenaufwands erfassen wir Trainings- und Inferenzzeiten, den Speicherbedarf sowie die Anzahl der Modellparameter. Die Metriken werden im Folgenden erläutert.

4.6.1. Genauigkeit

In ihrer Arbeit geben Naser und Alavi [NA23] eine Übersicht über oft verwendete Metriken für Regression und Klassifikation. Sie empfehlen die Verwendung mehrerer Metriken, um Einschränkungen einzelner auszugleichen. Beispielsweise gewichtet der RMSE Ausreißer stärker und die Einheit [°C] ist intuitiver als [°C^2]. Daher verwenden wir mehrere Metriken, die wir nachfolgend definieren.

Für die Leistungsanalyse messen wir den Root Mean Squared Error in [°C] mit $RMSE = \sqrt{\frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(t_{i,j} - \hat{t}_{i,j}\right)^2}$ und den Mean Absolute Error in [°C] mit $MAE = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left|t_{i,j} - \hat{t}_{i,j}\right|$ mit der wahren Temperatur $t_{i,j}$, der vorhergesagten Temperatur $\hat{t}_{i,j}$, der Anzahl n an Zellen

im Tensor und der Anzahl m an Datenpunkten. Den AAE pro Zelle bestimmen wir in [°C], indem wir den Absolute Error (AE) einer Zelle j über den gesamten Datensatz summieren und anschließend seinen Duchschnitt ermitteln: $\bar{e}_j = \frac{1}{m} \sum_{i=1}^{m} \left| t_{i,j} - \hat{t}_{i,j} \right|, \; j \in \{1, ..., n\}$. Dieser unterstützt dabei, die Vorhersagezellen zu identifizieren, die im Durchschnitt die größte Abweichung von der Grundwahrheit aufweisen. Mit PAT messen wir den Prozentanteil der Fehlerzellen bezüglich eines Datensatzes, deren AE die Schwelle von 0.1 °C überschreitet. Dies ermöglicht eine intuitive Interpretation: Bei typischen Messungen der Grundwassertemperatur kann eine Genauigkeit von 0.1 °C erreicht werden [UKB24]. Eine niedrige Prozentzahl des PAT kann daher als Indikator für eine hohe Genauigkeit betrachtet werden.

Unseren Metriken stimmen mit den Definitionen aus der Arbeit von Pelzer und Schulte [PS24] überein.

Auswertung der 3D Modelle. Eine Auswertung der Fehlermetriken im gesamten 3D Bereich könnte im Vergleich zu zweidimensionalen Analysen zu einer Verzerrung führen, da das Volumen der Wärmefahne im Verhältnis zum Gesamtvolumen klein ist. Der Großteil des Volumens wird kaum von der Wärmefahne beeinflusst. Dadurch fallen Fehler im Wärmefahnenbereich weniger stark ins Gewicht, was sich positiv auf die Metriken auswirkt und ihre Aussagekräftigkeit bezüglich der Genauigkeit einschränkt.

Um einen fairen Vergleich der 2D und 3D Modelle zu ermöglichen, extrahieren wir für unsere Messungen die XY- und ZY-Ebene auf Höhe des Injektionspunktes aus den 3D Vorhersagevolumina. Diese Ebenen werden auf 16×256 Zellen reduziert und entsprechen damit der Outputgröße der 2D Modelle, die wir für den späteren Vergleich heranziehen. Für die Analyse wählen wir die schlechter abschneidene Ebene, da sie eine konservative Leistungsabschätzung erlaubt und zeigt, wie das Modell im ungünstigeren Fall performt. Dies verhindert eine Verzerrung zugunsten der 3D Netzwerke und bietet eine realistischere Grundlage für die Gesamtbewertung des Modells.

Weiterhin berechnen wir den Wert der Zelle mit dem größten AAE und mitteln zudem den AAE in X-Richtung, um das Randverhalten der 3D Modelle zu visualisieren.

4.6.2. Rechenaufwand

Um den Rechenaufwand unserer Modelle zu messen, definieren wir uns die Metriken Trainingszeit, Inferenzzeit und Speicherbedarf.

Trainingszeit. Sie gibt die Dauer des Trainings auf einem Trainingsdatensatz an. Die Messung beginnt nach der Datenvorbereitung und -initialisierung und endet nach dem Durchlauf der letzten Epoche. Wir messen die Trainingszeit des Modells mit dem besten Validation Loss in [s] sowie dessen benötigte Epochenanzahl. Aus den Messungen berechnen wir die benötigte Trainingszeit pro Epoche in [s], pro Datenpunkt in [ms] und pro Zelle in [μs].

Inferenzzeit. Sie umfasst die Zeit, die das Modell benötigt, um einen einzelnen Datenpunkt zu verarbeiten. Die Zeitmessung wird exklusiv auf der GPU durchgeführt, beginnt unmittelbar bevor dem Modell die Daten übergeben werden und endet nach Verarbeitung eines einzelnen Datenpunktes, wobei zwischen GPU und Central Processing Unit (CPU) synchronisiert wird, um das exakte Zeitende zu messen. Das Laden und Übertragen der Datenpunkte und des Modells findet vor der Zeitmessung statt. Wir messen die benötigte GPU-Zeit für alle Datenpunkte eines kompletten

Datensatzes einzeln und bilden anschließend den Mittelwert der Inferenzzeit für den gesamten Datensatz. Zusätzlich geben wir die Inferenzzeit inklusive Ladezeit und Datenübertragung in den Video Random Access Memory (VRAM) der GPU an. Die gemessene Einheit beträgt [s].

Speicherbedarf. Zur Ermittlung des Speicherbedarfs berechnen wir die Inputspeichergröße eines einzelnen vorbereiteten Datenpunktes, gemittelt über 1000 dp. Dies spiegelt die tatsächliche Datengröße wider, die ein Modell nach dem Training erhält.

Darüber hinaus geben wir die Modellspeichergröße sowie die Anzahl der Modellparameter an. Alle Speichergrößen werden in [KB] angegeben.

4.7. 2D Modelle und Unterschiede

In diesem Kapitel stellen wir die verwendeten 2D Modelle vor, die als Basis für unseren Vergleich dienen. Im Anschluss werden die wichtigsten Unterschiede zwischen den 2D und 3D Modellen hinsichtlich Architektur, Training und Datensatzverwendung beschrieben. Diese Unterschiede betreffen unter anderem die Anzahl der initialen Merkmale, die Lossfunktion, die Batchgröße sowie die Art und den Umfang der Inputdaten.

4.7.1. Verwendete 2D Modelle

Da sich unsere Hardware, wie im Anhang A.1 beschrieben, von der ursprünglich für das 2D Modell eingesetzten Hardware unterscheidet, führen wir das Training sowie alle Messungen neu durch. Dabei behalten wir die ursprünglichen Einstellungen aus Anhang A.2 und die 2D Datensätze aus Anhang A.3 bei. Das 2D Modell trainieren wir auf den Datensatzgrößen 100 dp und 1000 dp mit den Inputkanälen "gksi"(siehe Abschnitt 4.3). Das Training wird dreimal durchgeführt.

In jedem Trainingsdurchlauf vergleichen wir pro Epoche das neu erreichte Validation Loss mit dem bisher in diesem Training erreichten Bestwert. Ein Modell mit besserem Validation Loss wird abgespeichert. Für unsere Analyse wählen wir das zuletzt gespeicherte Modell, sofern nichts anderes angegeben.

4.7.2. Unterschiede zu 3D

Architektur. Die Architektur der 2D Modelle verwendet 32 im Vergleich zu 16 Merkmalen in den 3D Modellen und damit eine doppelt so hohe Anzahl initialer Merkmale. Diese Merkmalsanzahl beeinflusst auch die Verdopplungs- bzw. Halbierungsstruktur der Merkmale zwischen den U-Net Blöcken, wodurch die 2D Modelle insgesamt eine höhere Kapazität zur Merkmalserkennung besitzen.

Training. Die 3D Modelle werden sowohl mit dem MSE Loss als auch mit dem SmoothL1 Loss trainiert. Das SmoothL1 Loss wird dabei jedoch nur in einem einzigen Trainingsdurchlauf angewendet, während beim MSE Loss für die 2D und 3D Modelle drei Trainingseinheiten durchgeführt werden.

Begrenzte Speicherkapazitäten führen außerdem dazu, dass die 3D Modelle mit einer Batchgröße von 6 trainiert werden, während die Batchgröße bei den 2D Modellen 1000 beträgt. Dadurch werden im Training der 3D Modelle die Gewichte auf Basis einer kleineren Anzahl von Datenpunkten häufiger aktualisiert, was ein Gradientenrauschen verursachen und das Lernen erschweren kann, wie in Abschnitt 4.2 erläutert. Deshalb legen wir für unsere 3D Modelle eine geringere initiale LR fest.

Ein weiterer Unterschied besteht in der Anpassung der LR: Für die 3D Modelle wird ein automatischer LR-Scheduler eingesetzt, während das Training der 2D CNNs einem festgelegten LR-Schedule folgt. Da ein 3D CNN aufgrund der höheren Rechenanforderungen deutlich weniger Epochen trainiert wird als ein 2D CNN, ist eine frühzeitige Anpassung der LR notwendig, um die Trainingsqualität zu sichern. Ein langsamer Lernprozess fördert die Präzision, erhöht jedoch auch das Risiko, dass das Modell in einem lokalen Minimum verbleibt. Die 2D Modelle können mit ihrem festgelegten LR-Schedule und der höheren Batchgröße leichter ein globales Minimum erreichen, obwohl eine hohe LR in diesem Fall die Gefahr birgt, lokale Minima zu überspringen, was das Lernen feiner Details erschweren kann. Insgesamt wird die LR im 3D Training schneller reduziert als bei den 2D Modellen.

Zudem verwenden wir im Gegensatz zu den 2D Modellen im 3D Fall Early Stopping, was zu einem vorzeitigen Abbruch des Trainings führen kann.

Datensätze Die 2D und 3D Datensätze wurden aus physikalisch konsistenten und realistischen Wertebereichen generiert und sind in ihren Parametereinstellungen sehr ähnlich. Es bestehen jedoch Abweichungen in der Permeabilität, die in Anhang A.6 beschrieben werden. Zusätzlich gibt es Unterschiede in den Dimensionsgrößen der Datenpunkte: ein 2D Datenpunkt hat eine Größe von 16×256 Zellen, während ein 3D Datenpunkt aus $64 \times 256 \times 64$ Zellen besteht. Ein 3D Datenpunkt enthält somit 256-mal so viele Zellen wie ein 2D Datenpunkt und bezieht daher eine wesentlich größere Umgebung mit ein, was sich auch auf die Berechnung der Metriken auswirkt. Um diesem Problem zu begegnen, extrahieren wir eine Ebene aus dem 3D Volumen.

Die Testdatensätze werden, wie in Abschnitt 4.4 beschriebenen, als prozentualer Anteil eines gesamten Datensatzes entnommen und weichen damit ebenfalls voneinander ab. Verschiedene Variationen in den Testdatensätze können potentiell zu unterschiedlichen Ergebnissen führen und dadurch die Evaluierung der Modelle beeinflussen.

5. Resultat und Diskussion

In diesem Kapitel führen wir eine Analyse der 2D und 3D Modelle durch, die auf den Datensätzen mit 100 und 1000 dp trainiert wurden. Diese Modelle zeichnen sich durch das jeweils niedrigste erreichte Validation Loss aus und werden im Folgenden häufig als "beste"Modelle bezeichnet. Insgesamt erhalten wir vier Modelle auf Basis des MSE Losses: zwei 2D CNNs und zwei 3D CNNs trainiert auf je 100 und 1000 dp. Im Folgenden bezeichnen wir die 2D Netzwerke als 2D-100 und 2D-1000 gemäß ihrer Datensatzgröße sowie die 3D Modelle als 3D-100 respektive 3D-1000.

Zudem testen wir zusätzlich die SmoothL1 Lossfunktion mit $\beta \in \{0.1, 1\}$. Unser auf 1000 dp trainiertes 3D Modell mit der SmoothL1 Lossfunktion bezeichnen wir als 3D-1000-L1.

Zunächst untersuchen wir die Leistung unserer 2D und 3D Modelle, die auf dem MSE Loss trainiert wurden, indem wir die Genauigkeit und Generalisierungsfähigkeit der Modelle analysieren und vergleichen. Im nächsten Kapitel gehen wir in Kürze auf die Ergebnisse der Modelle mit SmoothL1 Lossfunktion ein und betrachten die Genauigkeit des 3D-1000-L1. Anschließend vergleichen wir den Rechenaufwand aller Modelle, indem wir Trainings- und Inferenzzeiten sowie Speicheranforderungen gegenüberstellen. Zudem untersuchen wir die Lerngeschwindigkeit und fassen die Ergebnisse in einem finalen Überblick zusammen.

5.1. Leistungsanalyse der 2D und 3D Modelle

In diesem Abschnitt untersuchen wir Überanpassung und Generalisierungsfähigkeit unserer Modelle mithilfe von Fehlermessungen auf Trainings-, Validations- und Testdaten. Wir vergleichen die erreichte Genauigkeit der 2D und 3D Modelle unter Verwendung der Fehlermetriken auf den Testdatensätzen.

Generalisierungsfähigkeit. Der Verlauf des Training und Validation Loss ist in Abbildung 5.1 dargestellt, die Epochenanzahl und erreichten besten MSE-Werte sind in Tabelle 5.1 aufgeführt.

Die Kurven des Training und Validation Loss liegen im 3D Fall als auch beim 2D-1000 in Abbildung 5.1 eng beieinander, was darauf hindeutet, dass es zu keiner Überanpassung gekommen ist. Im Gegensatz dazu weisen die Graphen des 2D-100 von Beginn an eine erhebliche Differenz zwischen den beiden Losswerten auf, was auf eine verstärkte Anpassung an die Trainingsdaten hindeutet. Dafür sprechen auch die gemessenen Fehlermetriken aus Tabelle 5.2. Das 2D-100 schneidet in allen Metriken auf den Test- und Validationsdaten deutlich schlechter ab als auf den Trainingsdaten. Eine mögliche Erklärung dafür könnte die geringe Menge an Trainingsdaten im Verhältnis zur Modellkomplexität sein, was zu einer schlechten Generalisierung führt.

Das 3D-1000 ist das einzige Modell, das ein klares Konvergenzverhalten zeigt. Beim 3D-100 sind zunehmende Schwankungen im Validation Loss erkennbar. Es ist in Tabelle 5.1 zu beachten, dass das 3D-100 aufgrund von wiederholten, aber geringfügigen Verbesserungen im Validation Loss

eine deutlich höhere Anzahl an Epochen als das 3D-1000 trainiert wurde. Das 3D-100 könnte sich auf einem Plateau befinden, auf dem es nur minimale, aber zunehmend instabile Fortschritte erzielt. Die Metriken RMSE und MAE zeigen für das 3D-100 ähnliche Werte über alle drei Datensätze hinweg. Der Anteil an Fehlerzellen über 0.1 °C steigt jedoch auf den Testdaten. Diese Abweichung bleibt aber mit einer Differenz von 1.63 % bezüglich der Trainingsdaten gering und deutet darauf hin, dass das Modell nicht überangepasst ist.

Die 3D Modelle und das 2D-1000 schneiden damit in allen Metriken auf ihren Testdatensätzen ähnlich gut ab wie auf den Trainingsdatensätzen. Doch nur die geringen Werte des 2D-1000 deuten auf eine gute Generalisierungsfähigkeit hin.

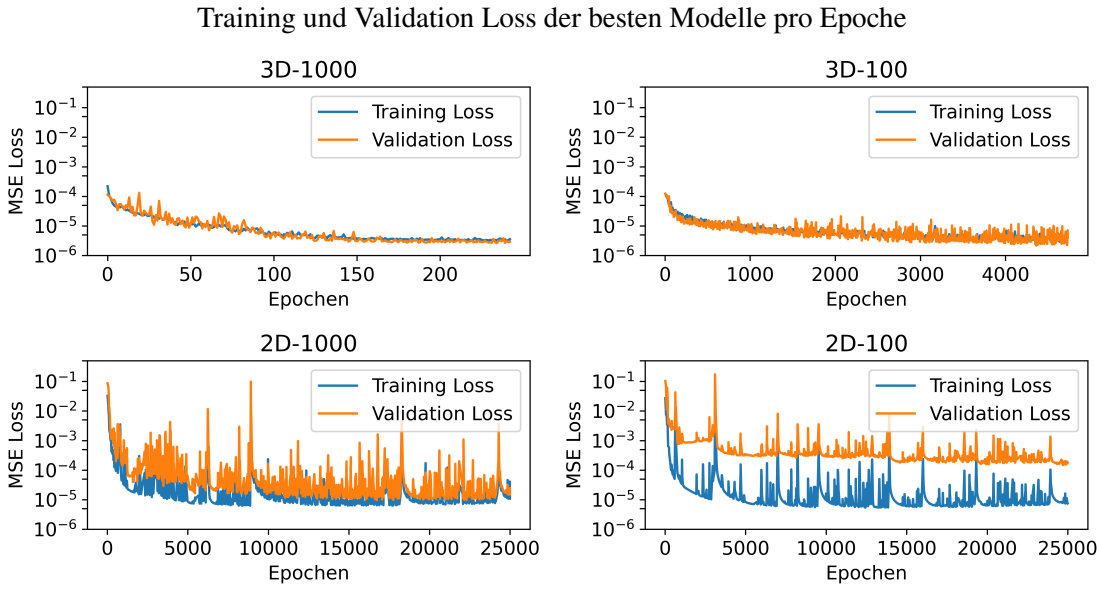

Abbildung 5.1.: Graphen zum Training und Validation Loss pro Epoche der besten 2D und 3D Modelle: In Farbe blau für das Training Loss und in Farbe orange für das Validation Loss. Der MSE ist normiert.

Modell	**Epochen**	**MSE normalisiert** in $[10^{-6}]$	
		train	**val**
2D-100	24607	9.49	144.82
2D-1000	14250	6.75	9.17
3D-100	4418	3.57	2.14
3D-1000	234	2.64	3.29

Tabelle 5.1.: Erreichtes Validation Loss und zugehöriges Training Loss mit jeweiliger Epochenanzahl der besten Modelle. Dabei handelt es sich um das normierte MSE Loss, das im 3D Fall auf den gesamten 3D Bereich angewendet wird.

Modell	RMSE in [°C]			MAE in [°C]			PAT in [%]		
	train	val	test	train	val	test	train	val	test
2D-100	0.0242	0.0602	0.1800	0.0096	0.0215	0.0466	1.29	4.92	9.51
2D-1000	0.0133	0.0151	0.0142	0.0061	0.0069	0.0063	0.14	0.23	0.20
3D-100	0.0820	0.0813	0.0811	0.0260	0.0259	0.0286	5.51	5.85	7.14
3D-1000	0.0800	0.0803	0.0810	0.0300	0.0305	0.0309	7.92	8.09	8.39

Tabelle 5.2.: Berechnung der Metriken RMSE, MAE und PAT der besten Netzwerke, trainiert mit den Datensatzgrößen von 100 und 1000 dp und ausgewertet an Trainings-, Validations- und Testdatensatz.

Genauigkeit 2D vs. 3D. Für die Genauigkeitsanalyse der 3D Modelle wählen wir die schlechter abschneidene ZY-Ebene. Die Werte beider Ebenen sind im Anhang A.4 dargestellt. Der Unterschied zwischen den Ebenen ist in Bezug auf den RMSE und MAE sehr gering, jedoch verdoppeln sich die Fehlerzellen über 0.1°C etwa. Die 3D Modelle haben Schwierigkeiten die vertikale Ausbreitung der Wärmefahnen genau zu erfassen.

Im Vergleich der Fehlermetriken RMSE, MAE und PAT schneidet das 3D-100 insgesamt leicht besser ab als das 3D-1000. Dies ist in Tabelle 5.5 vermutlich auf die fast doppelt so lange Trainingszeit des 3D-100 zurückzuführen. Der RMSE des 2D-1000 ist zehnmal und sein MAE sechsmal besser als der des 2D-100. Zudem weist es einen 31-mal besseren PAT als der des 2D-100 auf. Die Verwendung eines zehnmal größeren Datensatzes im 2D Bereich führt damit zu einer signifikanten Genauigkeitsverbesserung. Im 3D Fall hingegen tritt diese Verbesserung nicht ein.

Die 3D Modelle übertreffen in allen Metriken das 2D-100, liegen mit ihren Werten jedoch deutlich näher am 2D-100 als am 2D-1000. Das beste 3D Modell 3D-100 zeigt 24-mal mehr Zellenfehlers über 0.1 °C als das 2D-1000. Zudem zeigt es einen fünfmal höheren RMSE und einen viermal höheren MAE als das 2D-1000.

Den geringsten Unterschied sowohl in der Differenz als auch im Verhältnis zwischen RMSE und MAE zeigt das 2D-1000. Dies könnte darauf hindeuten, dass es insgesamt stabiler und weniger anfällig für große Fehlvorhersagen ist.

Zur Visualisierung des dreidimensionalen zellweisen AAE extrahieren wir die XY- und ZY-Ebene auf Höhe des Injektionspunktes und wählen die schlechter abschneidene ZY-Ebene für die weitere Analyse. Zum Vergleich sind beide Ebenen am Beispiel des 3D-100 in Abbildung 5.2 dargestellt. Zusätzlich betrachten wir alle ZY-Ebenen des dreidimensionalen AAE der 3D Modelle und stellen fest, dass sich viele Zellen mit hohen Fehlerwerten auf Höhe des Injektionspunktes befinden. Dies stützt unsere Wahl die Höhe des Injektionspunktes für unsere Messungen zu verwenden.

In Abbildung 5.3 ist zu sehen, dass alle Modelle Schwierigkeiten bei der Temperaturvorhersage am und um den Injektionspunkt aufweisen. Dieses Verhalten könnte auf Glättungseffekte zurückzuführen sein, die durch die Funktionsweise der Convolution entstehen. Der maximale AAE bleibt beim 3D-100 unter 1.6 °C, beim 3D-1000 unter 1.7 °C. Bei den 2D Modellen liegt der maximale AAE unter 0.2 °C für das 2D-1000 und unter 0.3 °C für das 2D-100. Das leistungsstärkere 3D Modell

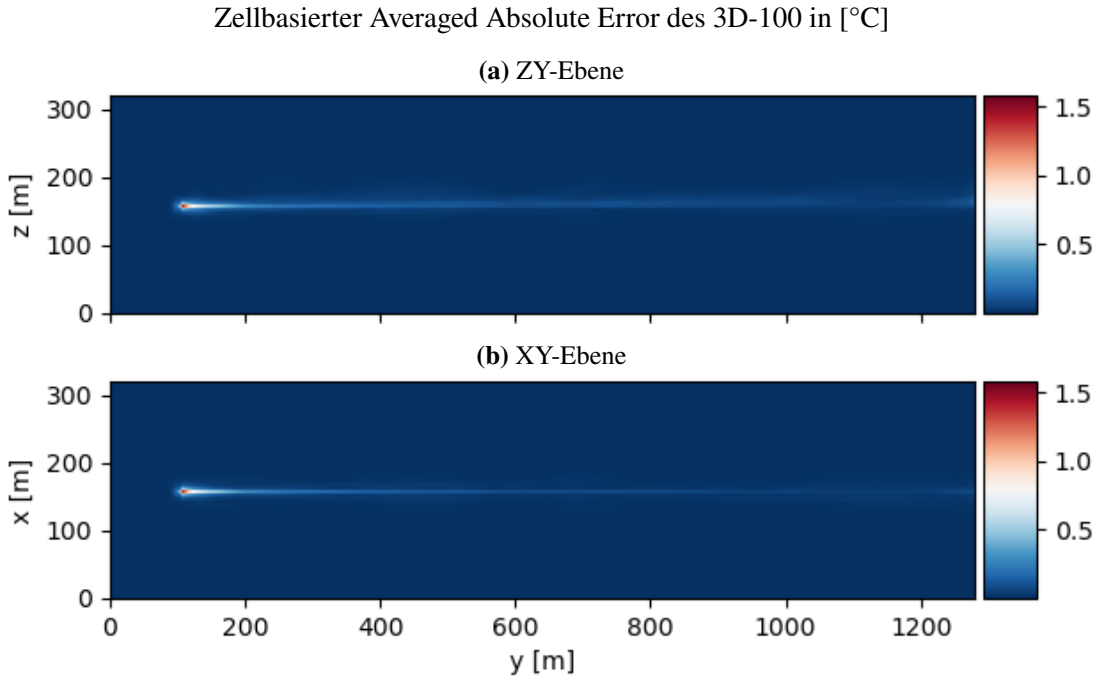

Abbildung 5.2.: Visualisierung des Averaged Absolute Errors (AAEs) pro Zelle am Beispiel des 3D-100: Oben wird der Schnitt in X-Höhe und unten in Z-Höhe des Injektionspunkts angezeigt.

weist somit einen maximalen AAE auf, der achtmal höher ist als der des besten 2D Modells. Die längliche weiße Fahne, die besonders gut bei den 3D Modellen zu erkennen ist, deutet auf Schwierigkeiten bei der Vorhersage von Wärmefahnen entlang der Strömungsrichtung hin.

Das 2D-100 weist zusätzliche Vorhersageschwierigkeiten an den Rändern der Wärmefahnen auf, was sich in den weißen ovalen Bereichen niederschlägt. Weiße Stellen am Rand und in den Ecken der 2D Modell-Bilder lassen vermuten, dass die Vorhersage von Randwerten ebenfalls herausfordernd ist, wahrscheinlich bedingt durch das verwendete Padding. In Abbildung 5.4 sind ähnliche Schwierigkeiten auch bei den 3D Modellen erkennbar. Den Beobachtungen zufolge liegen die Fehlerwerte an den Rändern beim 2D-1000 unter 0.1 °C und beim besten 3D Modell, in X-Richtung gemittelt, unter 0.02 °C. Diese Werte erscheinen insgesamt sehr gering, stützen sich jedoch auf die grafische Auswertung der Bilder in Abbildung 5.3 und Abbildung 5.4.

Die häufigeren Fehlerzellen der 3D Modelle scheinen besonders auf die Vorhersagen von Wärmefahnen entlang der Strömungsrichtung und auf den steilen Temperaturgradienten rund um den Injektionspunkt zurückzuführen sein.

Insgesamt übertrifft das 2D-1000 alle anderen Modelle in der Genauigkeit mit sowohl niedrigeren durchschnittlichen Abweichungen als auch weniger großen Fehlvorhersagen. Das beste 3D Modell 3D-100 hat nicht nur wesentlich häufiger Abweichungen oberhalb des Grenzwertes von 0.1 °C als das 2D-1000, sondern vereinzelt auch deutlich höhere Fehlerwerte, was auf eine erhebliche Ungenauigkeit hindeutet.

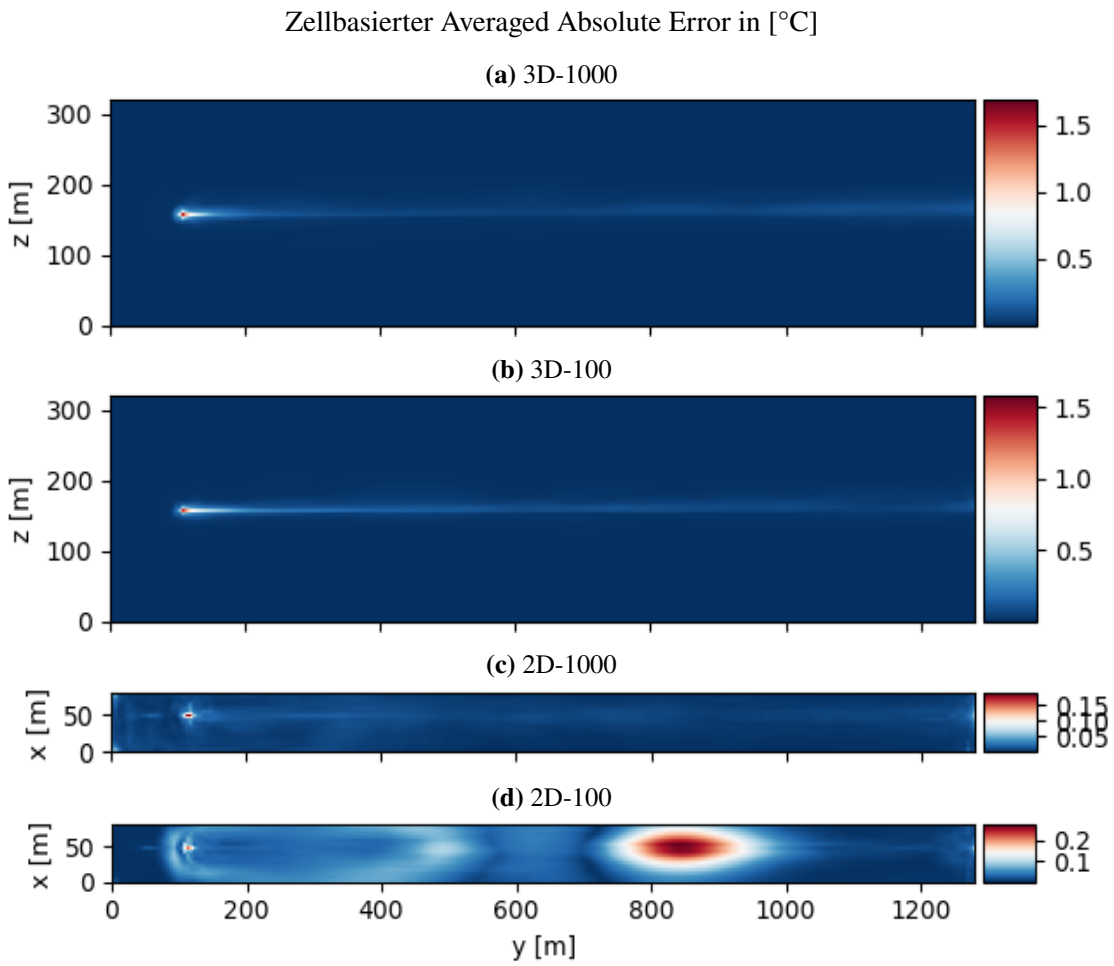

Abbildung 5.3.: Visualisierung des Averaged Absolute Error (AAE) pro Zelle mit gemeinsamer Y-Achse in Meter. Für den zellbasierten 3D AAE wählen wir die ZY-Ebene auf X-Höhe des Injektionspunktes.

5.2. Ergebnisse mit SmoothL1 Loss und Leistungsanalyse des 3D-1000-L1

In diesem Abschnitt präsentieren wir der Vollständigkeit halber in Kürze Ergebnisse aus verschiedenen Einstellungen der SmoothL1 Lossfunktion, widmen uns im Weiteren aber der Leistungsanalyse des 3D-1000-L1.

Wir trainieren die 3D Modelle mit dem SmoothL1 Loss und $\beta \in \{0.1, 1\}$. Das Validation Loss während des Trainings ist in Abbildung 5.5 visualisiert. Nur das 3D Modell, das mit 1000 dp und $\beta = 0.1$ trainiert wurde, zeigt eine signifikante Verbesserung gegenüber den 3D Modellen mit MSE Loss, wie in Tabelle 5.3 zu sehen ist. Dieses Modell bezeichen wir im Folgenden als 3D-1000-L1.

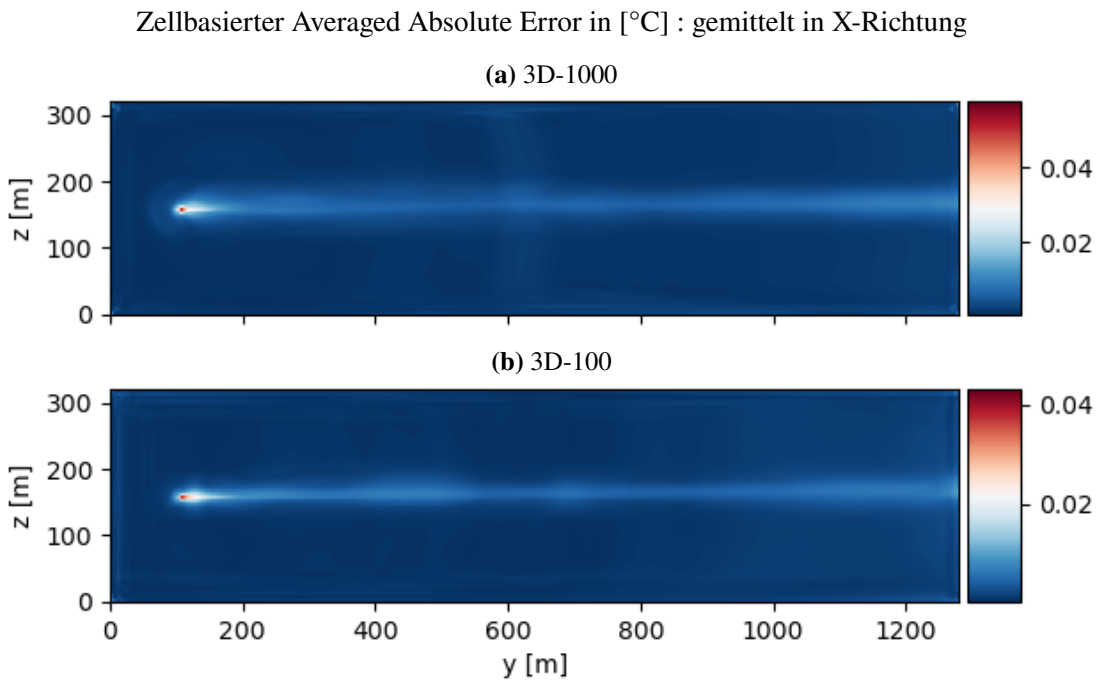

Abbildung 5.4.: Visualisierung des zellbasierten Averaged Absolute Error (AAE) der 3D Modelle mit gemeinsamer Y-Achse. Der AAE wird in X-Richtung gemittelt.

Da die anderen Modelle keine Verbesserungen aufweisen, werden sie im Weiteren nicht näher betrachtet. Dennoch sei darauf hingewiesen, dass auch hier bei 3D Modellen mit 1000 dp Konvergenz erkennbar ist, während dies bei 3D Modellen mit 100 dp nicht der Fall ist. Bei 100 dp ist zwar eine langsame Verbesserung festzustellen, jedoch mit zunehmenden Schwankungen im Validation Loss. Diese Instabilität, die auch beim MSE Loss auftritt, deutet auf die Schwierigkeit hin, eine stabile Lösung zu finden, was durch die geringe Datenmenge bedingt sein könnte.

β	dp	Epochen	RMSE in [°C]	MAE in [°C]	PAT in [%]
0.1	100	1073	0.127	0.050	14.4
0.1	1000	305	0.053	0.015	1.9
1	100	4410	0.100	0.035	10.7
1	1000	309	0.144	0.497	13.2

Tabelle 5.3.: Werte der Metriken RMSE, MAE und PAT der 3D CNNs mit SmoothL1 Lossfunktion, ausgewertet am Testdatensatz und in ZY-Richtung.

Trainingsverlauf und Generalisierungsfähigkeit. Die Graphen in Abbildung 5.6 nähern sich an und schwanken nur noch leicht, das Modell zeigt Konvergenz.

Die Auswertung von RMSE, MAE und PAT in Tabelle 5.4 zeigen für das 3D-1000-L1 ähnliche Werte auf allen drei Datensätze, daher ist keine Überanpassung anzunehmen. Die niedrigeren Werte im Vergleich zum 3D-1000 zeigen eine deutlich bessere Generalisierungsfähigkeit.

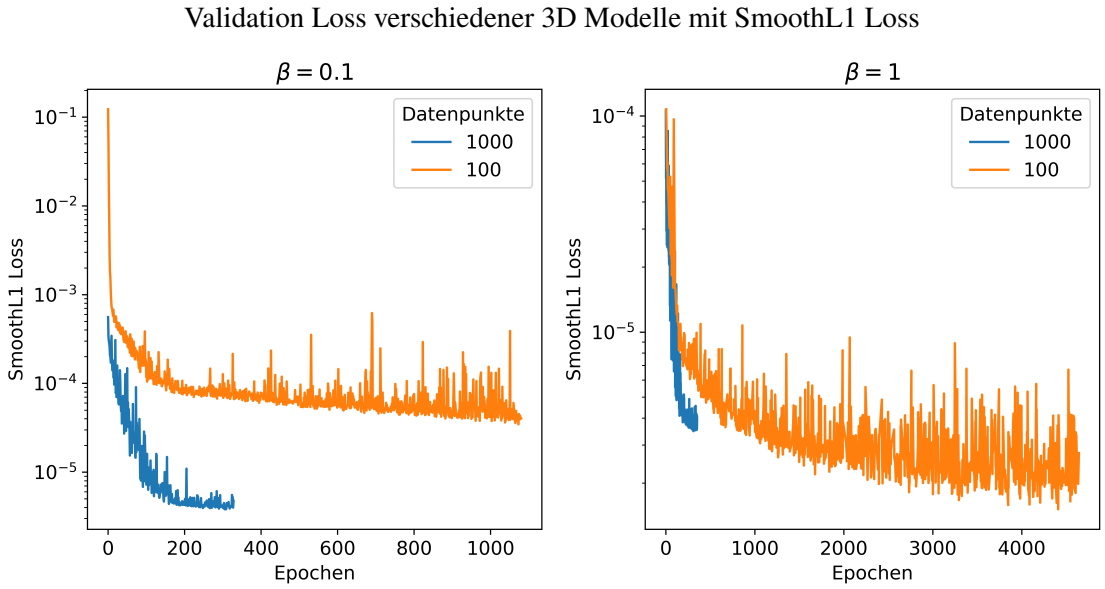

Abbildung 5.5.: Validation Loss pro Epoche der 3D Modelle mit SmoothL1 Loss: In Farbe blau für das Training Loss und in Farbe orange für das Validation Loss. Das SmoothL1 Loss ist normiert.

Modell	**RMSE in [°C]**			**MAE in [°C]**			**PAT in [%]**		
	train	**val**	**test**	**train**	**val**	**test**	**train**	**val**	**test**
3D-1000-L1	0.0528	0.0527	0.0530	0.0145	0.0149	0.0147	1.81	1.91	1.86

Tabelle 5.4.: Berechnung der Metriken RMSE, MAE und PAT in ZY-Ebene des 3D-1000-L1, ausgewertet an Trainings-, Validations- und Testdatensatz.

Genauigkeit 2D vs. 3D Für die Analyse der Genauigkeit verwenden wir die Fehlermetriken Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) und Percentage Above Threshold (PAT) aus Tabelle 5.4 sowie den zellbasierten Averaged Absolute Error (AAE) mit Visualisierung in Abbildung 5.7. Dabei beziehen wir uns auf die am Testdatensatz gemessenen Werte in Bezug auf die ZY-Ebene, wie in Abschnitt 4.6.1 beschrieben.

Das 3D-1000-L1 hat einen 1.5-fach besseren RMSE, 1.9-fach besseren MAE und einen 3.8-fach besseren PAT als das 3D-100. Der RMSE des 2D-1000 ist 3.7-mal, sein MAE 2.3-mal und sein PAT 9.3-mal besser als der des 3D-1000-L1. Damit verbessert sich wie im 2D Fall auch im 3D Fall die Genauigkeit signifikant bei Erhöhung des Datensatzes. Die Genauigkeit des 2D-1000 erreicht das 3D-1000-L1 jedoch nicht.

In Abbildung 5.7 ist zu erkennen, dass auch das 3D-1000-L1 Schwierigkeiten bei der Temperaturvorhersage am und um den Injektionspunkt hat. Der maximale AAE des 3D Modells ist mit etwa 1.5 °C nach wie vor hoch. Die Vorhersage von Wärmefahnen entlang der Strömungsrichtung scheint sich im Vergleich zum 3D-1000 verbessert zu haben, da die weiße längliche Wolke weniger deutlich

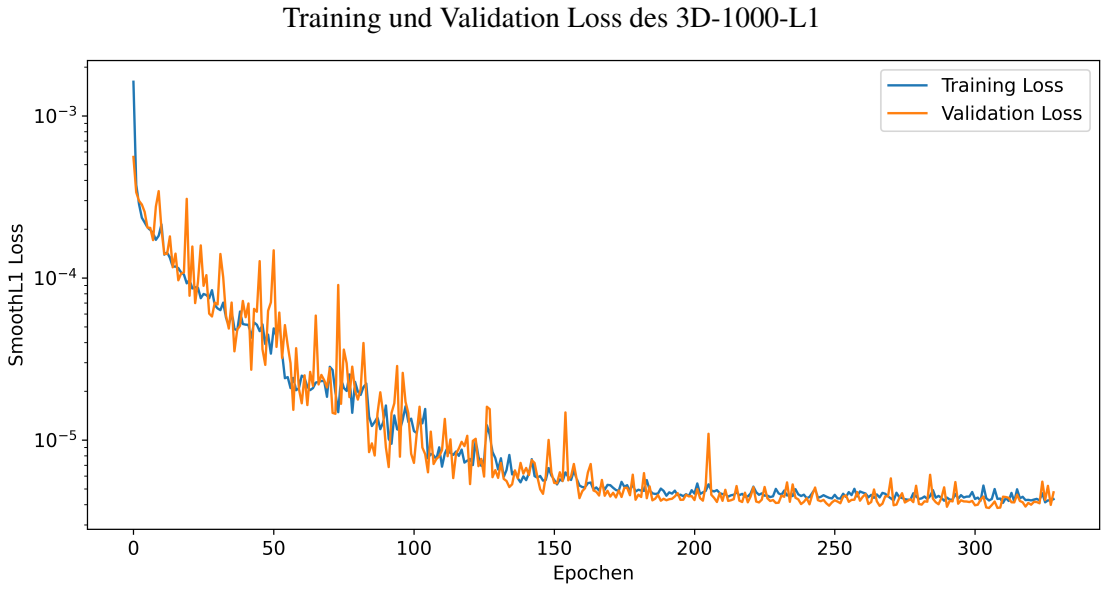

Abbildung 5.6.: Training und Validation Loss des 3D-1000-L1 pro Epoche : In Farbe blau für das Training Loss und in Farbe orange für das Validation Loss. Das SmoothL1 Loss ist normiert.

sichtbar ist. Dennoch bleibt die Herausforderung bestehen. Im unteren Bild sind weiße Stellen an den Rändern und in den Ecken erkennbar, der Fehler scheint jedoch gering zu sein. Demnach stellt vor allem der steile Temperaturgradient in der Nähe des Injektionspunktes eine Schwierigkeit bei der Vorhersage dar.

5.3. Vergleich des Rechenaufwands der 2D und 3D Modelle

In diesem Abschnitt vergleichen wir den Rechenaufwand der 2D und 3D Modelle anhand der Kriterien Trainingszeit, Speicherbedarf und Inferenzzeit. Zunächst wird die Trainingszeit analysiert und zwischen den Modellen verglichen. Anschließend betrachten wir die Speicheranforderungen, die sowohl von der Modellgröße und der Anzahl der Parameter als auch von der Größe der Inputdaten eines einzelnen Datenpunktes abhängen. Abschließend vergleichen wir die Inferenzzeit, sowohl ohne als auch mit Berücksichtigung des Ladens und der Übertragung der Daten auf die GPU.

Trainingszeit. Die gemessene Trainingszeit ist in Tabelle 5.5 dargestellt. Wir berechnen die Trainingszeit pro Epoche, Datenpunkt und Zelle. Das 2D-1000 weist eine 5.1-mal längere Trainingszeit auf als das 2D-100, während das 3D-100 1.9-mal so lange trainiert wird wie das 3D-1000 und 1.4-mal länger als das 3D-1000-L1. Im 2D Fall steigt die Trainingszeit bei größeren Datenmengen also verhältnismäßig stärker an.

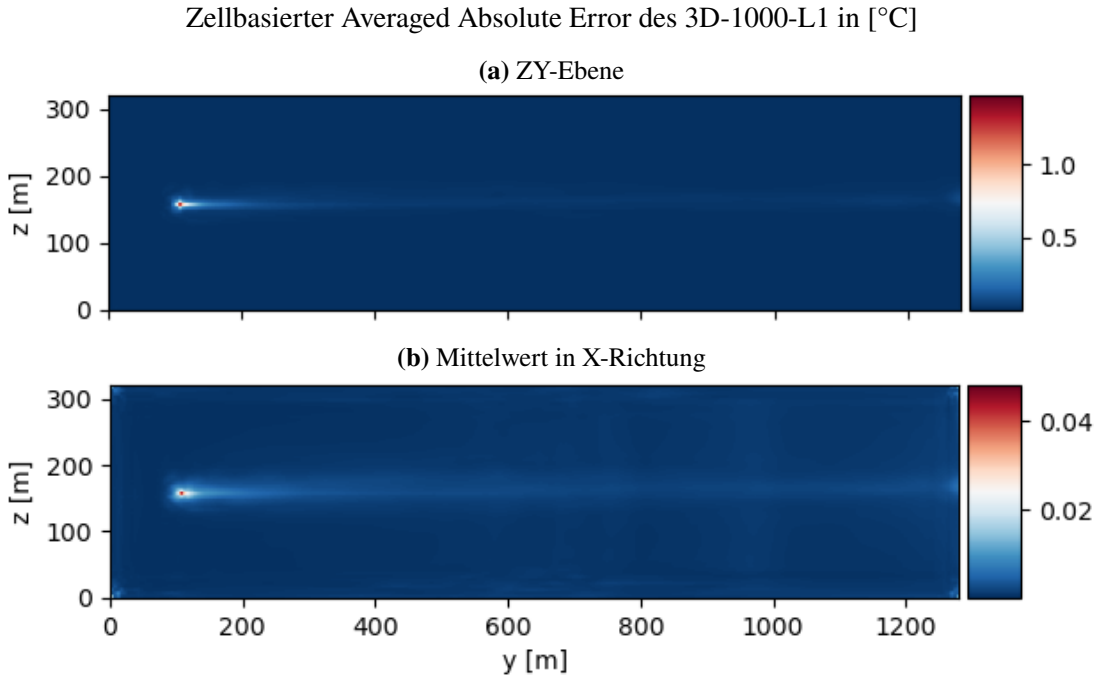

Abbildung 5.7.: Visualisierung des zellbasierten Averaged Absolute Error (AAE) mit gemeinsamer Y-Achse in Meter: oben die ZY-Ebene auf X-Höhe des Injektionspunktes, unten der Mittelwert in X-Richtung.

Beim Vergleich zwischen den 2D und 3D Modellen ergibt sich eine acht bis 77.3-mal längere Trainingszeit für die 3D Modelle. Dabei zeigt das 3D-1000-L1 eine 10.4-mal längere Trainingszeit als das 2D-1000, das 3D-100 eine 77.3-mal längere Trainingszeit als das 2D-100. Mit steigender Datenmenge verringert sich damit der relative Trainingszeitunterschied zwischen den 2D und 3D Modellen.

Die Trainingszeit pro Datenpunkt ist für das 2D-1000 um 0.1 ms niedriger als für das 2D-100. Insgesamt skaliert jedoch die Trainingszeit pro Epoche für alle CNN mit der Datenmenge: Modelle die auf der zehnfachen Datenmenge trainiert wurden, benötigen etwa die zehnfache Zeit pro Epoche. Die Trainingsdauer für eine einzelne Zelle bleibt damit für ein CNN unabhängig von der Datenmenge etwa gleich.

Bei gleicher Datensatzgröße ist die Trainingszeit pro Epoche für ein 3D Modell bis zu 489-mal länger als für ein 2D Modell. Auf Zellebene verdoppelt sich die Rechenzeit mit den Faktoren 1.7 für 100 dp und 1.9 für 1000 dp nahezu.

Speicherbedarf. Die erhobenen Speichergrößen der Modelle und Inputdaten sowie die Anzahl der Modellparameter können Tabelle 5.6 entnommen werden. Das 3D Modell benötigt 7243 KB mehr Speicher als das 2D Modell und ist damit um 23.3 % größer. Die Speichergröße eines 3D Inputs hingegen ist etwa 251-mal höher als bei einem 2D Input. Da ein 3D Input jedoch 256-mal mehr Zellen umfasst als ein 2D Input, verringert sich im 3D Fall die Speichergröße pro Zelle um 2.2 %.

Modell	Trainingszeit in			
	[h:min:s]	[s pro Epoche]	[ms pro dp]	[μs pro Zelle]
2D-100	00:34:49	0.08	0.85	0.21
2D-1000	02:58:53	0.75	0.75	0.18
3D-100	44:49:42	36.53	365.28	0.35
3D-1000	23:47:46	366.09	366.09	0.35
3D-1000-L1	31:04:37	366.81	366.81	0.35

Tabelle 5.5.: Berechnete Trainingszeit aller Modelle in verschiedenen Einheiten, sortiert nach Dimension und Größe des Datensatzes.

Dim	Parameteranzahl	Speichergröße in [KB]	
		Modell	vorbereiteter Datenpunkt
2D	7756609	31061	67
3D	9567905	38304	16778

Tabelle 5.6.: Speichergröße der 2D und 3D Modelle sowie eines Datenpunktes nach der Vorbereitung, gemittelt über die jeweiligen Inputdaten. Die Angabe erfolgt in Kilobyte. Zusätzliche Angabe der Parameteranzahl.

Inferenzzeit. Die in Tabelle 5.7 dargestellten Messwerte zeigen, dass ein 3D Modell für die Verarbeitung eines Volumens bis zu 45 ms in Anspruch nimmt. Im Vergleich benötigt ein 3D CNN somit bis zu 26-mal mehr Zeit zur Verarbeitung eines einzelnen Datenpunktes als ein 2D CNN.

Unter Einbeziehung des Ladens und der Übertragung auf die GPU benötigt das 3D-1000-L1 68 ms und damit neunmal mehr Zeit als das 2D-1000. Das 3D-100 benötigt 48 ms und damit 18-mal länger als das 2D-100. Dies deutet darauf hin, dass die Effizienz der 3D Modelle in Bezug auf die Inferenzzeit im Vergleich zu den 2D Modellen erheblich verringert ist.

5.4. Evaluierung des Lernfortschritts im Trainingsverlauf

Wir betrachten in diesem Kapitel die Effizienz der Modelle pro Epoche und Trainingszeit. Die Effizienz pro Epoche gibt Aufschluss darüber, wie schnell ein Modell aus Daten lernt, während die Trainingszeit zeigt, wie effizient die eingesetzte Rechenzeit für den Lernfortschritt genutzt wird.

In Abbildung 5.8 ist das Validation Loss der Modelle in Bezug auf die Trainingszeit und die Anzahl der Epochen dargestellt. Ein Vergleich zwischen 2D und 3D ist hier jedoch nur eingeschränkt möglich, da das MSE Loss im 3D Fall auf dem gesamten 3D Volumen berechnet wird und dies, wie in Abschnitt 4.6.1 erläutert, die Metrik positiv beeinflusst. Zudem werden das MSE und SmoothL1 Loss unterschiedlich berechnet, was ebenfalls einen direkten Vergleich zwischen

Modell	Inferenzzeit in [s]	
	ohne Laden & Übertragen	mit Laden & Übertragen
2D-100	0.0018	0.0026
2D-1000	0.0017	0.0076
3D-100	0.0441	0.0476
3D-1000	0.0441	0.0640
3D-1000-L1	0.0450	0.0683

Tabelle 5.7.: Gemessene Inferenzzeit der 2D und 3D CNNs, gemittelt über die Datensatzgrößen 100 dp und 1000 dp gemäß der in Abschnitt 4.6.2 beschriebenen Vorgehensweise.

den 3D Modellen erschwert. Um diese Vergleiche dennoch zu ermöglichen, wählen wir aus den gespeicherten Zwischenmodellen während des Trainings eine gezielte Teilmenge aus, die in bestimmten Trainingsphasen definierte Schwellenwerte im Validation Loss[1] unterschreitet. Diese Teilmenge werten wir anhand der Metriken RMSE, MAE und PAT auf den jeweiligen Testdatensätzen aus. Eine Visualisierung dieser Metriken zu bestimmten Epochen und über Trainingszeitpunkten ist in Abbildung 5.9 dargestellt.

Effizienz pro Epoche. Die Kurven der Modelle mit 1000 dp zeigen innerhalb ihrer jeweiligen Dimensionalität in Abbildung 5.8 insgesamt einen schnelleren Abfall als die Kurven der Modelle mit 100 dp. Dies deutet darauf hin, dass eine höhere Lerneffizienz bei einer größeren Datenmenge vorliegt. In Abbildung 5.9 ist zu erkennen, dass das 3D-1000 und 3D-1000-L1 über die ausgewählten Epochen hinweg die höchste Lerneffizienz aufweisen. Beide Modelle zeigen jedoch nach etwa 170 Epochen kaum noch Fortschritte. Das 2D-1000 lernt ebenfalls effizient, kann seine Lernleistung aber über eine größere Anzahl an Epochen aufrechterhalten, was zu einer deutlich höheren Endgenauigkeit führt.

Effizienz über die Trainingszeit. Zu Beginn des Trainings zeigen beide Modelle mit 100 dp in Abbildung 5.8 eine kurzfristig erhöhte Lerngeschwindigkeit, was vermutlich auf die geringere Datenmenge zurückzuführen ist. Allerdings werden sie im Laufe des Trainings von den Modellen mit der zehnfachen Datenmenge überholt: Das 3D-1000 übertrifft das 3D-100 nach etwa 6 h Trainingszeit, während das 2D-1000 das 2D-100 bereits nach etwa 6 min überholt. Insbesondere erreicht das 3D-100 das Validation Loss des 3D-1000 erst nach der 1.4-fachen Trainingszeit. Die Modelle mit 1000 dp und MSE Loss zeigen somit insgesamt eine effizientere Lernleistung in Bezug auf die Trainingszeit. Das 3D-1000-L1 bleibt im Vergleich in Bezug auf Abbildung 5.8 unberücksichtigt, da es auf einer anderen Lossfunktion basiert.

Um alle Modelle miteinander zu vergleichen, betrachten wir Abbildung 5.9. Zwischen den 2D und 3D Modelle ist erkennbar, dass die Metriken beim 2D-1000 über die Trainingszeit hinweg deutlich schneller sinken als bei allen anderen Modellen. Das 2D-100 zeigt zu Beginn ebenfalls einen

[1] $\{10^{-n}, 5 \cdot 10^{-n} \mid n = 1, 2, ..., 6\}$

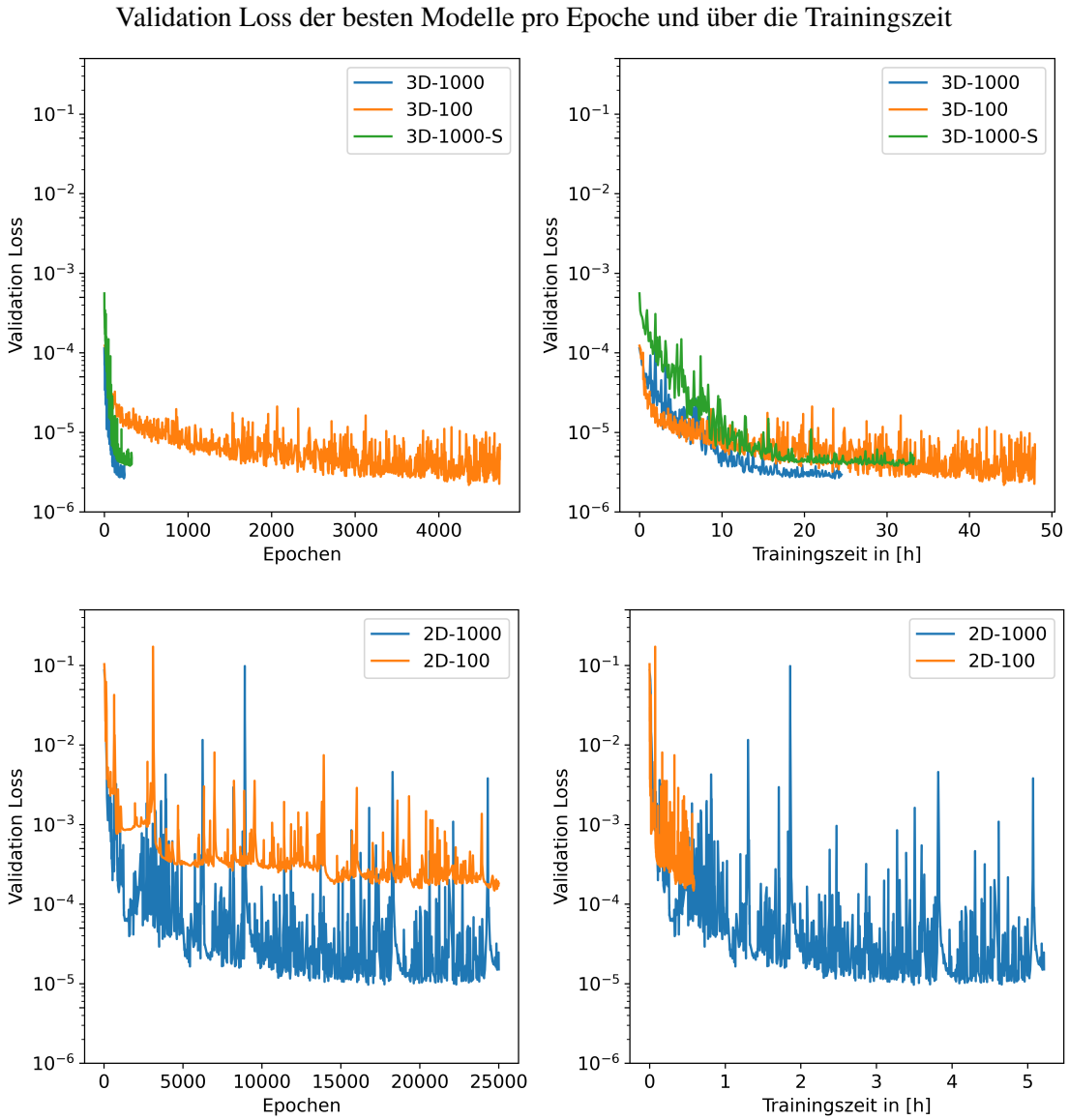

Abbildung 5.8.: Graphen der 2D und 3D Modelle zum normierten Validation Loss in Abhängigkeit von der Epochenanzahl und der Trainingszeit.

schnelleren Lernprozess als die 3D Modelle, stagniert jedoch schließlich, vermutlich aufgrund der zu geringen Datenmenge im Verhältnis zur Modellkomplexität. Auch die 3D Modelle zeigen nach einigen Stunden nur noch moderate Verbesserungen. Dabei führt im 3D Fall mit MSE Loss die zehnfache Datenmenge zu keiner wesentlichen Genauigkeitsverbesserung während des Trainings, verkürzt jedoch die Trainingszeit. Mit der SmoothL1 Lossfunktion hingegen ist besonders in der ersten Hälfte des Trainings ein schnelleres Absinken der Metriken zu beobachten.

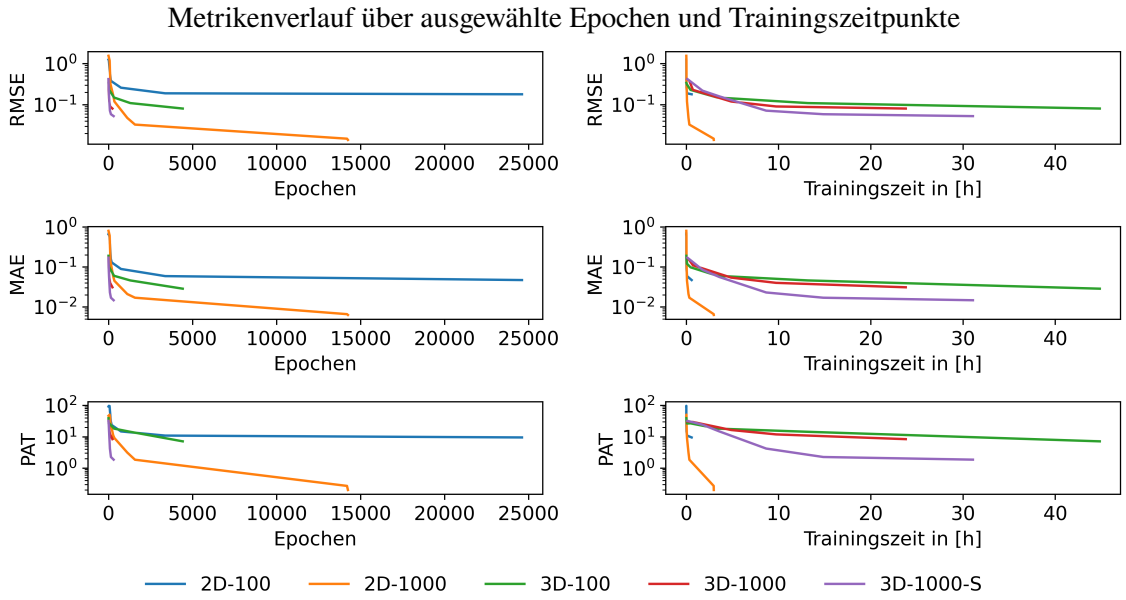

Abbildung 5.9.: Graphen der 2D und 3D Modelle zu RMSE, MAE und PAT: pro Epoche auf der linken Seite, in Abhängigkeit von der Trainingszeit auf der rechten Seite.

Insgesamt scheint die größere Datenmenge von 1000 dp das Erkennen von Mustern in den Daten zu beschleunigen, diese Modelle benötigen weniger Epochen und haben eine höhere Trainingsgeschwindigkeit. Zudem lernen die 3D Modelle bei gleicher Datenmenge im Durchschnitt pro Epoche schneller, was auf die zusätzlichen Informationen aus der dritten Dimension zurückzuführen sein könnte. Über die Zeit hinweg sind jedoch die 2D Modelle durchschnittlich schneller.

5.5. Zusammenführung der Ergebnisse zu Leistung, Rechenaufwand und Effizienz

In diesem Kapitel fassen wir wesentliche Erkenntnisse unserer Analyse zusammen und diskutieren Leistung, Rechenaufwand und Effizienz der Modelle.

Die Ergebnisse zeigen, dass eine Datenmenge von 100 dp sowohl für den 2D Fall als auch für 3D Modelle nicht ausreicht. Im 2D Fall führt die geringe Datenmenge zu einer Überanpassung, da vermutlich das Modell im Verhältnis zu den Daten zu komplex ist. Im 3D Fall zeigt das Modell bei 100 dp sowohl mit MSE Loss als auch mit SmoothL1 Loss zunehmend instabiles Verhalten im Training und keine Konvergenz, was ebenfalls auf eine unzureichende Datenmenge zurückzuführen sein könnte. Das 3D Modell mit 100 dp erreicht eine bessere Genauigkeit als das 2D Modell mit 100 dp, weist jedoch mit einem PAT von 7.14 % immer noch viele Fehlerzellen über 0.1 °C auf.

Bei einer größeren Datenmenge von 1000 dp sehen wir im 2D Fall eine hohe Genauigkeit mit einem PAT von 0.2 %. Das beste 3D Modell mit 1000 dp erreicht mit einem 9.3-mal höheren PAT diese Genauigkeit nicht. Es zeigt jedoch mit einem PAT von 1.86 % eine Verbesserung von 5.28 % im Vergleich zum PAT von 7.14 % des 3D-100. Das entspricht einer Reduktion der Fehlerzellen über 0.1 °C um 73.9 %. Zudem zeigen die 3D Modelle mit 1000 dp deutliches Konvergenzverhalten.

Die größere Datenmenge von 1000 dp beschleunigt das Erkennen von Mustern in den Daten und führt insgesamt zu einer höheren Trainingsgeschwindigkeit. Bei gleicher Datenmenge lernen die 3D Modelle pro Epoche schneller, was auf die zusätzlichen Informationen aus der dritten Dimension zurückzuführen ist. Über die Trainingszeit hinweg sind jedoch die 2D Modelle durchschnittlich schneller.

Der Wechsel der Lossfunktion hat die Genauigkeit des 3D Modells mit 1000 dp verbessert. Das Ausbleiben dieses Effekts beim 3D Modell mit 100 dp könnte auf die zu geringe Datenmenge sowie den einmaligen Trainingsdurchlauf zurückzuführen sein.

Bezüglich des Rechenaufwandes zeigt sich, dass die Trainingszeit pro Epoche etwa mit der Datenmenge skaliert. Bei steigender Datenmenge verringert sich der relative Trainingszeitunterschied zwischen 2D und 3D Modellen. Dies ist auf die geringe Epochenanzahl durch schnelle Konvergenz bei den 3D Modellen mit 1000 dp zurückzuführen. Dennoch benötigt ein 3D Modell pro Zelle nahezu die doppelte Trainingszeit im Vergleich zu einem 2D Modell.

Die Speichergröße des 3D Modells ist um 7243 KB höher als die des 2D Modells. Ein 3D Inputdatenpunkt benötigt insgesamt 251-mal mehr Speicher als ein 2D Inputdatenpunkt, erfordert jedoch auf Zellebene etwa 2 % weniger Speicher. Daher könnte der Speicher bei 3D Inputdatenpunkten effizienter genutzt werden.

Die Inferenzzeit bei 3D Modellen beträgt bis zu 45 ms, was bis zu 26-mal höher ist als im 2D Fall. Nimmt man Lade- und Übertragungsprozesse hinzu, benötigt das 3D-1000-L1 neunmal mehr Zeit als das 2D-1000 und das 3D-100 18-mal länger als das 2D-100. Die Effizienz der 3D Modelle in Bezug auf die Inferenzzeit ist daher im Vergleich zu den 2D Modellen erheblich verringert. Dies zeigt, dass die Verarbeitung von 3D-Daten deutlich zeitintensiver ist.

Die Vorhersage von Wärmefahnen entlang der Strömungsrichtung bleibt für das 3D Modell auch nach Losswechsel eine Herausforderung, obwohl hier eine deutliche Verbesserung eintritt. Die hohen Fehlerwerte rund um den Injektionspunkt bleiben jedoch bestehen. Sie könnten auf Glättungseffekten beruhen, die sich möglicherweise durch die zusätzliche Dimension viel stärker auswirken als es in 2D der Fall ist. Anpassungen in der Architektur, des β-Parameters der Lossfunktion oder eine Erhöhung der Datenmenge könnten Genauigkeitsverbesserungen bringen.

Insgesamt zeigt die Analyse, dass im 3D Fall eine Datenmenge von mindestens 1000 dp für Konvergenz und gute Genauigkeit notwendig ist. Das beste 3D Modell zeigt eine Vervierfachung des RMSE, Verdopplung des MAE und eine Verneunfachung der Fehlerzellen über 0.1 °C bei einer etwa zehnfachen Trainingszeit im Vergleich zum 2D-1000. Wir vermuten, dass diese Abweichungen vor allem durch den steilen Temperaturgradienten um den Injektionspunkt bedingt sind.

6. Gesamtbewertung und Zukunftsausblick

In diesem Kapitel wird eine umfassende Bewertung der Ergebnisse dieser Arbeit vorgenommen, gefolgt von einem Ausblick auf mögliche zukünftige Entwicklungen im Bereich der Temperaturvorhersage von Wärmefahnen in Grundwasserwärmepumpensystemen. Unsere Ergebnisse werden im Kontext ihrer Limitierungen und ihrer Bedeutung innerhalb des aktuellen Forschungsstandes betrachtet. Abschließend werden potenzielle Verbesserungen und Forschungsperspektiven vorgeschlagen, die zur Optimierung der entwickelten Modelle beitragen und die Genauigkeit der Vorhersagen weiter erhöhen können.

6.1. Limitierungen

Große Modelle wie 3D CNNs erfordern aufgrund ihrer zahlreichen Parameter umfangreiche Datenmengen und lange Trainingszeiten, was den Umfang und die Tiefe möglicher Optimierungen einschränkt.

Ein weiteres Limitierungspotenzial liegt in den zugrunde liegenden Trainingsdaten, die hinsichtlich Permeabilität und besonders in der abgedeckten Dimensionalität variieren. Die 3D Daten umfassen einen größeren Bereich um die Wärmefahne als die 2D Daten, was den Vergleich der Modelle möglicherweise verfälscht haben könnte.

Zudem stellt die Leistungsmessung anhand einer 2D Ebene, die aus den 3D Volumina extrahiert wurde, eine potenzielle Einschränkung dar. Bei diesem Ansatz bleiben große Teile des 3D Volumens unberücksichtigt. Um eine repräsentative Ebene auszuwählen, vergleichen wir verschiedene Ebenen und wählen jeweils die weniger präzise aus, um die Robustheit unserer Messung zu erhöhen. Es ist wichtig zu beachten, dass unser 3D Modell möglicherweise benachteiligt wurde, da die Genauigkeit der XY-Ebene die der ausgewählten ZY-Ebene übertrifft, wie in Anhang A.4 demonstriert.

Die Testdaten basieren auf simulierten Datensätzen, die keiner zusätzlichen Validierung unterzogen wurden, was die Generalisierbarkeit der Ergebnisse potenziell einschränken könnte.

Schließlich wurde die Lossfunktion für das 3D Modell mit 1000 dp zu einem späten Zeitpunkt geändert, was neue Optimierungen erforderlich machen könnte, um die Anpassung des Modells an die Zielmetrik zu verbessern. Zudem könnte eine erneute Anpassung des Schwellenwerts β der SmoothL1 Lossfunktion die Leistungsfähigkeit weiter verbessern.

6.2. Einbettung in den wissenschaftlichen Kontext

Frühere Arbeiten, wie jene von Pelzer und Schulte [PS24], zeigen, das 2D CNNs bei der Vorhersage bemerkenswerte Ergebnisse erzielen können. Mit der Erweiterung um eine dritte Dimension kann die räumliche Ausbreitung von Wärme realistischer erfasst werden, was potenziell zu genaueren Vorhersagen führt. Trotz einzelner hoher Zellenfehlers im Bereich des Injektionspunktes erreicht unser 3D-1000-L1 eine gute Genauigkeit mit einem PAT von 1.86 %. Es bleibt jedoch hinter dem auf 1000 dp trainierten 2D CNN von Pelzer und Schulte [PS24], das einen PAT von 0.2 % erzielt. Damit leistet unsere Arbeit einen wichtigen Beitrag zur Optimierung der Grundwasserwärmepumpen-Infrastruktur und zur Weiterentwicklung von Modellierungsansätzen in diesem Bereich.

Wie bereits in wissenschaftlichen Arbeiten in Abschnitt 3.2 dokumentiert, erfordern 3D CNNs eine aufwendige Optimierung sowie erheblich höhere Trainingszeiten und Rechenressourcen im Vergleich zu 2D Ansätzen. Die Uneinigkeit in der Literatur zur Leistungsfähigkeit von 2D und 3D CNNs ist vermutlich auf die Spezifität der jeweiligen Anwendungsfälle und die damit verbundene Optimierung zurückzuführen. In unserem Szenario konnten wir zeigen, dass ein 3D CNNs mindestens die Datenmenge eines leistungsfähigen 2D CNNs benötigt. Wir sind zudem zuversichtlich, dass eine Erweiterung des Datensatzes weitere Genauigkeitsverbesserungen bringen könnte.

Der erhöhte Rechenaufwand gegenüber 2D Modellen könnte sich dadurch rechtfertigen lassen, dass ein 3D CNN die räumliche Struktur der Daten präziser erfassen und dreidimensionale Parameter realistischer abbilden kann, da Grundwasserströmungen und Wärmetransportprozesse dreidimensional verlaufen.

6.3. Zusammenfassung und Ausblick

In dieser Arbeit entwickeln wir ein 3D CNN mit U-Net Architektur [RFB15] als Erweiterung eines 2D CNN, um die Temperatur von Wärmefahnen in Open-Loop-Systemen von Grundwasserwärmepumpen vorherzusagen. Das optimierte Netzwerk wird auf PFLOTRAN-simulierten [LHL+15] Inputdaten trainiert. Unser bestes 3D Modell zeigt einen vierfachen Root Mean Squared Error (RMSE), doppelten Mean Absolute Error (MAE) und neunmal mehr Fehlerzellen über 0.1°C bei zehnfach höherer Trainingszeit als das beste auf der Arbeit von Pelzer und Schulte [PS24] basierende 2D Modell. Die Inferenzzeit ist im Vergleich zum 2D Modell um das 26-fache erhöht. Unser Modell zeigt Schwierigkeiten bei der Vorhersage von Wärmefahnen entlang der Strömungsrichtung und weist insbesondere hohe Fehlerwerte im Bereich des Injektionspunktes auf. Um diese hohe Fehlerwerte zu senken, könnten weitere Anpassungen des Modells notwendig sein. Unsere Ergebnisse zeigen, dass im 3D Fall eine Datenmenge von mindestens 1000 dp für effizientes Lernen und Konvergenz notwendig ist. Der erhöhte Rechenaufwand gegenüber 2D Modellen könnte durch die realistischere Modellierung von Wärmefahnen gerechtfertigt sein.

Ausblick

Diese Arbeit unterstützt die Optimierung der Grundwasserwärmepumpen-Infrastruktur durch die Entwicklung eines 3D CNN zur Temperaturvorhersage von Wärmefahnen im Grundwasser. Die zusätzliche räumliche Dimension erlaubt eine realitätsnähere Erfassung der Wärmeausbreitung, was die Entscheidungsfindung bei der optimalen Platzierung von Grundwasserwärmepumpen verbessern kann. Zukünftige Arbeiten sollten die Einbeziehung weiterer Inputparameter sowie eine Vergrößerung der Datenbasis prüfen, um die Modellgenauigkeit weiter zu steigern. Eine Erweiterung auf die zweite Stufe aus der Arbeit von Pelzer und Schulte [PS24] könnte zudem die Wechselwirkungen von Wärmefahnen in den Modellierungsansatz integrieren und so die Vorhersagequalität weiter verbessern.

Literaturverzeichnis

[AGL05] J. Ahrens, B. Geveci, C. Law. „ParaView: An End-User Tool for Large Data Visualization“. In: *Visualization Handbook* (Jan. 2005). URL: https://www.researchgate.net/publication/247111133_ParaView_An_End-User_Tool_for_Large_Data_Visualization (zitiert auf S. 28).

[ASY+19] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama. „Optuna: A Next-generation Hyperparameter Optimization Framework“. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, S. 2623–2631. ISBN: 9781450362016. DOI: 10.1145/3292500.3330701 (zitiert auf S. 26).

[AYH+24] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, S. Chintala. „PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation“. In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*. ACM, Apr. 2024. DOI: 10.1145/3620665.3640366 (zitiert auf S. 25, 26).

[AZH+21] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, L. Farhan. „Review of deep learning: concepts, CNN architectures, challenges, applications, future directions“. In: *Journal of Big Data* 8.1 (2021). ISSN: 2196-1115. DOI: 10.1186/s40537-021-00444-8 (zitiert auf S. 15).

[ÇAL+16] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger. „3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation“. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*. Hrsg. von S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, W. Wells. Cham: Springer International Publishing, 2016, S. 424–432. ISBN: 978-3-319-46723-8. DOI: 10.1007/978-3-319-46723-8_49 (zitiert auf S. 23).

[Cla03] C. Clauser. *Numerical simulation of reactive flow in hot aquifers: SHEMAT and processing SHEMAT*. Springer, 2003 (zitiert auf S. 21).

[Cli22] I. P. on Climate Change (IPCC). *Climate Change 2022: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge, UK und New York, NY, USA: Cambridge University Press, 2022, S. 3056. DOI: 10.1017/9781009325844 (zitiert auf S. 13).

[CLS22] L. Crespi, D. Loiacono, P. Sartori. „Are 3d better than 2d convolutional neural networks for medical imaging semantic segmentation?“ In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2022, S. 1–8. DOI: 10.1109/IJCNN55064.2022.9892850 (zitiert auf S. 23).

[DHT23] A. Dhoot, N. B. Hadj-Alouane, M. Turki-Hadj Alouane. „2D CNN vs 3D CNN: An Empirical Study on Deep Learning-based Facial Emotion Recognition“. In: *2023 International Conference on Modeling, Simulation & Intelligent Computing (MoSICom)*. 2023, S. 138–143. DOI: 10.1109/MoSICom59118.2023.10458797 (zitiert auf S. 23).

[Die13] H.-J. G. Diersch. *FEFLOW: finite element modeling of flow, mass and heat transport in porous and fractured media*. Springer, 2013 (zitiert auf S. 21).

[DLPS23] K. Davis, R. Leiteritz, D. Pflüger, M. Schulte. „Deep learning based surrogate modeling for thermal plume prediction of groundwater heat pumps“. In: *arXiv preprint arXiv:2302.08199* (2023) (zitiert auf S. 22).

[DR85] P. Domenico, G. Robbins. „A new method of contaminant plume analysis“. In: *Groundwater* 23.4 (1985), S. 476–485. DOI: 10.1111/j.1745-6584.1985.tb01497.x (zitiert auf S. 21).

[DV20] A. Dhillon, G. K. Verma. „Convolutional neural network: a review of models, methodologies and applications to object detection“. In: *Progress in Artificial Intelligence* 9.2 (2020), S. 85–112. ISSN: 2192-6360. DOI: 10.1007/s13748-019-00203-0 (zitiert auf S. 16).

[EGB+23] B. Emek Soylu, M. S. Guzel, G. E. Bostanci, F. Ekinci, T. Asuroglu, K. Acici. „Deep-Learning-Based Approaches for Semantic Segmentation of Natural Scene Images: A Review“. In: *Electronics* 12.12 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12122730 (zitiert auf S. 23).

[FDE+11] B. Flemisch, M. Darcis, K. Erbertseder, B. Faigle, A. Lauser, K. Mosthaf, S. Müthing, P. Nuske, A. Tatomir, M. Wolff, R. Helmig. „DuMu^x: DUNE for multi-phase,component,scale,physics,. . . flow and transport in porous media“. In: *Advances in Water Resources* 34.9 (2011). New Computational Methods and Software Tools, S. 1102–1112. DOI: 10.1016/j.advwatres.2011.03.007 (zitiert auf S. 21).

[GBC16] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (zitiert auf S. 15–18, 26).

[Gio21] F. Giovannetti. „Solar- und Umweltenergie für effiziente Wärme- und Kälteerzeugung“. In: *Forschung für den European Green Deal*. Hrsg. von Forschungsverbund Erneuerbare Energien. Berlin: FVEE, 2021, S. 71–77. URL: https://www.fvee.de/wp-content/uploads/2022/01/th2020.pdf#page=53 (zitiert auf S. 13).

[Göh23] M. Göhring. *Modelling of Groundwater Flow including a Heatpump with the Physics Informed Neural Network framework MODULUS from NVIDIA*. Research Module Simtech, University of Stuttgart. 2023 (zitiert auf S. 22).

[HES21] T. Hassanzadeh, D. Essam, R. Sarker. „2D to 3D Evolutionary Deep Convolutional Neural Networks for Medical Image Segmentation“. In: *IEEE Transactions on Medical Imaging* 40.2 (2021), S. 712–721. DOI: 10.1109/TMI.2020.3035555 (zitiert auf S. 14, 23).

[HMB+10] S. Hähnlein, N. Molina-Giraldo, P. Blum, P. Bayer, P. Grathwohl. „Ausbreitung von Kältefahnen im Grundwasser bei Erdwärmesonden“. In: *Grundwasser* 15.2 (2010), S. 123–133. ISSN: 1432-1165. DOI: 10.1007/s00767-009-0125-x (zitiert auf S. 21).

[Hun07] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), S. 90–95. DOI: 10.1109/MCSE.2007.55 (zitiert auf S. 25).

[HZRS15] K. He, X. Zhang, S. Ren, J. Sun. „Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dez. 2015, S. 1026–1034. URL: https://openaccess.thecvf.com/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html (zitiert auf S. 27).

[IS15] S. Ioffe, C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. 2015. DOI: 10.48550/arXiv.1502.03167. arXiv: 1502.03167 [cs.LG] (zitiert auf S. 17).

[KBB+12] O. Kolditz, S. Bauer, L. Bilke, N. Böttcher, J.-O. Delfs, T. Fischer, U. J. Görke, T. Kalbacher, G. Kosakowski, C. McDermott et al. „OpenGeoSys: an open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media“. In: *Environmental Earth Sciences* 67 (2012), S. 589–599. DOI: 10.1007/s12665-012-1546-x (zitiert auf S. 21).

[KG19] M. Kurmanji, F. Ghaderi. „A Comparison of 2D and 3D Convolutional Neural Networks for Hand Gesture Recognition from RGB-D Data“. In: *27th Iranian Conference on Electrical Engineering (ICEE2019)*. IEEE. 2019, S. 2022–2027. DOI: 10.1109/IranianCEE.2019.8786671 (zitiert auf S. 23).

[Kin87] W. Kinzelbach. *Numerische Methoden zur Modellierung des Transports von Schadstoffen im Grundwasser*. Oldenbourg, 1987 (zitiert auf S. 21).

[KRB+20] J. Keller, V. Rath, J. Bruckmann, D. Mottaghy, C. Clauser, A. Wolf, R. Seidler, H. M. Bücker, N. Klitzsch. „SHEMAT-Suite: An open-source code for simulating flow, heat and species transport in porous media“. In: *SoftwareX* 12 (2020), S. 100533. ISSN: 2352-7110. DOI: 10.1016/j.softx.2020.100533 (zitiert auf S. 21).

[LC09] S. Lo Russo, M. V. Civita. „Open-loop groundwater heat pumps development for large buildings: A case study“. In: *Geothermics* 38.3 (2009), S. 335–345. ISSN: 0375-6505. DOI: 10.1016/j.geothermics.2008.12.009 (zitiert auf S. 21).

[LHL+15] P. C. Lichtner, G. E. Hammond, C. Lu, S. Karra, G. Bisht, B. Andre, R. Mills, J. Kumar. *PFLOTRAN User Manual. A Massively Parallel Reactive Flow and Transport Model for Describing Surface and Subsurface Processes*. Techn. Ber. U.S. Department of Energy, 2015. URL: https://www.osti.gov/biblio/1168703 (zitiert auf S. 14, 21, 27, 50).

[Li17] B. Li. „3d fully convolutional network for vehicle detection in point cloud“. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, S. 1513–1518. DOI: 10.1109/IROS.2017.8205955 (zitiert auf S. 23).

[LLY+22] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou. „A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects“. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (Dez. 2022), S. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827 (zitiert auf S. 16, 17, 26).

[LZK+24] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar. „Physics-Informed Neural Operator for Learning Partial Differential Equations“. In: *ACM / IMS J. Data Sci.* 1.3 (Mai 2024). DOI: 10.1145/3648506 (zitiert auf S. 22).

[MCBP20] B. Marinberg, A. Cohen, E. Ben-Dror, H. H. Permuter. „A study on MIMO channel estimation by 2D and 3D convolutional neural networks“. In: *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE. 2020, S. 1–6. DOI: 10.1109/ANTS50601.2020.9342797 (zitiert auf S. 23).

[Min09] K. u. E. B.-W. Ministerium für Umwelt. *Leitfaden zur Nutzung von Erdwärme mit Grundwasserwärmepumpen*. 2009. URL: https://um.baden-wuerttemberg.de/fileadmin/redaktion/m-um/intern/Dateien/Dokumente/2_Presse_und_Service/Publikationen/Energie/1_Leitfaden_Erdwaerme_Grundwasserwaermepumpen.pdf (besucht am 12. 10. 2024) (zitiert auf S. 13).

[MP13] E. Milnes, P. Perrochet. „Assessing the impact of thermal feedback and recycling in open-loop groundwater heat pump (GWHP) systems: a complementary design tool“. In: *Hydrogeology journal* 21.2 (2013), S. 505–514. DOI: 10.1007/s10040-012-0902-y (zitiert auf S. 13).

[NA23] M. Z. Naser, A. H. Alavi. „Error Metrics and Performance Fitness Indicators for Artificial Intelligence and Machine Learning in Engineering and Sciences“. In: *Architecture, Structures and Construction* 3.4 (Dez. 2023), S. 499–517. DOI: 10.1007/s44150-021-00015-8 (zitiert auf S. 31).

[NH10] V. Nair, G. E. Hinton. „Rectified linear units improve restricted boltzmann machines“. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, S. 807–814. URL: https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf (zitiert auf S. 16).

[NO10] Y. Nam, R. Ooka. „Numerical simulation of ground heat and water transfer for groundwater heat pump system based on real-scale experiment“. In: *Energy and Buildings* 42.1 (2010). International Conference on Building Energy and Environment (COBEE 2008), S. 69–75. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2009.07.012 (zitiert auf S. 21).

[OKK+22] M. Ohmer, A. Klester, A. Kissinger, S. Mirbach, H. Class, M. Schneider, M. Lindenlaub, M. Bauer, T. Liesch, K. Menberg, P. Blum. „Berechnung von Temperaturfahnen im Grundwasser mit analytischen und numerischen Modellen“. In: *Grundwasser - Zeitschrift der Fachsektion Hydrogeologie* 27 (2 2022), S. 113–129. DOI: 10.1007/s00767-022-00509-2 (zitiert auf S. 21).

[PAB+20] W. Pophillat, G. Attard, P. Bayer, J. Hecht-Méndez, P. Blum. „Analytical solutions for predicting thermal plumes of groundwater heat pump systems“. In: *Renewable Energy* 147 (2020). Shallow Geothermal Energy Systems, S. 2696–2707. ISSN:

0960-1481. DOI: https://doi.org/10.1016/j.renene.2018.07.148. URL: https://www.sciencedirect.com/science/article/pii/S0960148118309455 (zitiert auf S. 21).

[PBT+20] W. Pophillat, P. Bayer, E. Teyssier, P. Blum, G. Attard. „Impact of groundwater heat pump systems on subsurface temperature under variable advection, conduction and dispersion“. In: *Geothermics* 83 (2020), S. 101721. ISSN: 0375-6505. DOI: 10.1016/j.geothermics.2019.101721. URL: https://www.sciencedirect.com/science/article/pii/S0375650519300574 (zitiert auf S. 21).

[Pel24a] J. Pelzer. *Heat-Plume-Prediction Optuna*. Besucht am: 2024-10-01. 2024. URL: https://github.com/JuliaPelzer/Heat-Plume-Prediction/%20tree/optuna (zitiert auf S. 26).

[Pel24b] J. Pelzer. *Heat-Plume-Prediction Release_24*. Besucht am: 2024-10-01. 2024. URL: https://github.com/JuliaPelzer/Heat-Plume-Prediction/tree/release_24 (zitiert auf S. 25).

[PGT+22] S. Pnev, V. Groza, B. Tuchinov, E. Amelina, E. Pavlovskiy, N. Tolstokulakov, M. Amelin, S. Golushko, A. Letyagin. „Multi-Class Brain Tumor Segmentation via 3d and 2d Neural Networks“. In: *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. 2022, S. 1–5. DOI: 10.1109/ISBI52829.2022.9761538 (zitiert auf S. 23).

[PS24] J. Pelzer, M. Schulte. „Efficient two-stage modeling of heat plume interactions of geothermal heat pumps in shallow aquifers using convolutional neural networks“. In: *Geoenergy Science and Engineering* 237 (2024), S. 212788. ISSN: 2949-8910. DOI: 10.1016/j.geoen.2024.212788 (zitiert auf S. 3, 13, 14, 22, 25, 28–30, 32, 50, 51, 59).

[PSZ+23] R. Pourdarbani, S. Sabzi, R. Zohrabi, G. García-Mateos, R. Fernandez-Beltran, J. M. Molina-Martínez, M. H. Rohban. „Comparison of 2D and 3D convolutional neural networks in hyperspectral image analysis of fruits applied to orange bruise detection“. In: *Journal of Food Science* 88.12 (2023), S. 5149–5163. DOI: 10.1111/1750-3841.16801 (zitiert auf S. 23).

[QCZK20] X. Qin, J. Cai, Y. Zhou, Z. Kang. „Lattice Boltzmann simulation and fractal analysis of effective thermal conductivity in porous media“. In: *Applied Thermal Engineering* 180 (2020), S. 115562. ISSN: 1359-4311. DOI: 10.1016/j.applthermaleng.2020.115562 (zitiert auf S. 22).

[RFB15] O. Ronneberger, P. Fischer, T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597 (zitiert auf S. 15, 17, 27, 29, 50).

[RPK17] M. Raissi, P. Perdikaris, G. E. Karniadakis. „Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations“. In: *arXiv preprint arXiv:1711.10561* (2017). DOI: 10.48550/arXiv.1711.10561 (zitiert auf S. 22).

[Sch22] S. Scheurer. *A deep learning approach for large-scale groundwater heat pump temperature prediction*. Master thesis, University of Stuttgart. 2022. DOI: 10.18419/opus-12902 (zitiert auf S. 22).

[SCM+21] A. Stamoulakatos, J. Cardona, C. Michie, I. Andonovic, P. Lazaridis, X. Bellekens, R. Atkinson, M. M. Hossain, C. Tachtatzis. „A Comparison of the Performance of 2D and 3D Convolutional Neural Networks for Subsea Survey Video Classification“. In: *OCEANS 2021: San Diego – Porto*. 2021, S. 1–10. DOI: 10.23919/OCEANS44145.2021.9706125 (zitiert auf S. 23).

[Sun20] R.-Y. Sun. „Optimization for deep learning: An overview“. In: *Journal of the Operations Research Society of China* 8.2 (2020), S. 249–294. DOI: 10.1007/s40305-020-00309-6 (zitiert auf S. 26).

[SWG+20] S. P. Singh, L. Wang, S. Gupta, H. Goli, P. Padmanabhan, B. Gulyás. „3D Deep Learning on Medical Images: A Review“. In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185097 (zitiert auf S. 14).

[TBF+15] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri. „Learning Spatiotemporal Features With 3D Convolutional Networks“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, S. 4489–4497. URL: https://openaccess.thecvf.com/content_iccv_2015/html/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.html (zitiert auf S. 23).

[TTC18] V. Thakkar, S. Tewary, C. Chakraborty. „Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data“. In: *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*. 2018, S. 1–5. DOI: 10.1109/EAIT.2018.8470438 (zitiert auf S. 17).

[UKB24] UKB System Technology. *Kabellichtlot Typ 120 - LTC elektrische Leitfähigkeit bis 100m*. 2024. URL: https://geotechnik-shop.de/Kabellichtlot-Typ-120-LTC-elektrische-Leitfaehigkeit-bis-100m (besucht am 11. 10. 2024) (zitiert auf S. 32).

[US 24a] U.S. Department of Energy. *Geothermal Basics*. Office of Energy Efficiency & Renewable Energy. 2024. URL: https://www.energy.gov/eere/geothermal/geothermal-basics (besucht am 13. 09. 2024) (zitiert auf S. 13).

[US 24b] U.S. Department of Energy. *Geothermal Heat Pumps. Energy Saver*. 2024. URL: https://www.energy.gov/energysaver/geothermal-heat-pumps (besucht am 12. 10. 2024) (zitiert auf S. 13).

[WWY+22] M. Wang, H. Wang, Y. Yin, S. Rahardja, Z. Qu. „Temperature field prediction for various porous media considering variable boundary conditions using deep learning method“. In: *International Communications in Heat and Mass Transfer* 132 (2022), S. 105916. DOI: 10.1016/j.icheatmasstransfer.2022.105916 (zitiert auf S. 22).

[YYW+20] J. Yu, B. Yang, J. Wang, J. Leader, D. Wilson, J. Pu. „2D CNN versus 3D CNN for false-positive reduction in lung cancer screening“. In: *Journal of Medical Imaging* 7.5 (2020), S. 051202–051202. DOI: 10.1117/1.JMI.7.5.051202 (zitiert auf S. 23).

Alle URLs wurden zuletzt am 5. 11. 2024 geprüft.

A. Anhang

A.1. Hardware

Alle Messungen werden auf der CPU AMD EPYC 9274F mit RAM 384GB DDR5, 48 Kernen und 96 Threads ausgeführt. Das 3D CNN wird auf der GPU NVIDIA A30 24GB über die Grafikschnittstelle CUDA trainiert.

A.2. 2D Architektur und Trainingskonfiguration

Parameter	**Wert**
Merkmale	32 initial
Kernelgröße	5×5
Tiefe	3
Batchgröße	1000
Epochen	25000
Loss	MSE
Optimierungsverfahren	Adam
Learning Rate	$1 \cdot 10^{-4}$ initial $5 \cdot 10^{-5}$ ab Epoche 1200 $3 \cdot 10^{-5}$ ab Epoche 5000
Weight Decay	$1 \cdot 10^{-4}$

Tabelle A.1.: Architektur und Trainingskonfiguration des 2D CNN

A.3. 2D Datensätze

Eingesetzt werden die 2D Datensätze "benchmark_dataset_2d_100datapoints"(https://doi.org/10.18419/darus-3649) und "dataset_2d_small_1000dp"(https://doi.org/10.18419/darus-3650). Die Originale der vorbereiteten 2D Daten und die Modelle aus der Arbeit von Pelzer und Schulte [PS24] sind auf https://doi.org/10.18419/darus-3690 verfügbar. Die vorbereiteten 2D Daten und Modelle aus unserem Experiment werden der Betreuerin separat ausgehändigt.

A.4. Fehlermetriken der 3D CNNs in ZY- und XY-Ebene

Modell	Ebene	RMSE in [°C]			MAE in [°C]			PAT in [%]		
		train	val	test	train	val	test	train	val	test
3D-100	ZY	0.082	0.081	0.081	0.026	0.026	0.029	5.51	5.85	7.14
	XY	0.077	0.076	0.075	0.020	0.020	0.021	3.05	3.38	3.88
3D-1000	ZY	0.080	0.080	0.081	0.030	0.031	0.031	7.92	8.09	8.39
	XY	0.073	0.074	0.074	0.023	0.024	0.024	3.75	3.87	3.97
3D-1000-L1	ZY	0.053	0.053	0.053	0.015	0.015	0.015	1.81	1.91	1.86
	XY	0.051	0.052	0.052	0.011	0.012	0.011	1.28	1.35	1.34

Tabelle A.2.: Werte der Metriken RMSE, MAE und PAT in der XY- und ZY-Ebene auf Höhe des Injektionspunkts der 3D Netzwerke, trainiert mit den Datensatzgrößen von 100 und 1000 dp und ausgewertet an Trainings-, Validation- und Testdatensatz.

A.5. Varianz und Standardabweichung des zellbasierten AEs

Die Varianz V und die Standardabweichung σ des zellbasierten AEs berechnen wir mit $V = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\left| t_{i,j} - \hat{t}_{i,j} \right| - \bar{e}_j \right)^2$ und $\bar{e}_j = \frac{1}{m} \sum_{i=1}^{m} \left| t_{i,j} - \hat{t}_{i,j} \right|,\ j \in \{1, ..., n\}$ sowie $\sigma = \sqrt{V}$.

Modell	Ebene	Varianz in [10^{-3} °C^2]			Standardabweichung in [°C]		
		train	val	test	train	val	test
2D-100	XY	0.4	3.0	27.9	0.021	0.055	0.167
2D-1000	XY	0.1	0.2	0.1	0.011	0.012	0.012
3D-100	XY	2.3	2.4	2.0	0.048	0.049	0.045
	ZY	2.4	2.5	2.0	0.049	0.050	0.045
3D-1000	XY	2.3	2.4	2.3	0.048	0.049	0.048
	ZY	2.3	2.3	2.3	0.048	0.048	0.048
3D-1000-L1	XY	0.9	1.0	0.9	0.031	0.031	0.030
	ZY	0.9	0.9	0.9	0.030	0.031	0.030

Tabelle A.3.: Varianz und Standardabweichung des zellbasierten Absolute Errors (AEs), gemessen an Trainings-, Validation- und Testdatensatz.

A.6. PFLOTRAN-Einstellungen in 2D und 3D

Parameter	2D 100 dp	2D 1000 dp	3D 100 und 1000 dp
Dimensionen	2	2	3
Gebiet $[m \times m \times m]$	$100 \times 1280 \times 5$	$100 \times 1280 \times 5$	$320 \times 1280 \times 320$
Position $[m \times m \times m]$	$50 \times 120 \times 1$	$50 \times 120 \times 1$	$161 \times 111 \times 161$
Injektionsrate $[\ell/s]$	unbekannt	unbekannt	0.00024
Injektionstemperatur [°C]	15.6	15.6	15.6
Permeabilität (case)	perlin_noise	perlin_v2	perlin_v2
Permeabilitätsfrequenz	[2,8]	[9,9]	[1.8,1.8]
Permeabilität max $[m^2]$	$6.65 \cdot 10^{-9}$	$5.10 \cdot 10^{-9}$	$4.71 \cdot 10^{-9}$
Permeabilität min $[m^2]$	$1.36 \cdot 10^{-12}$	$1.02 \cdot 10^{-11}$	$1.05 \cdot 10^{-11}$
Druckgradient (min,max)	(-0.0035, -0.0015)	(-0.0035, -0.0015)	(-0.0035, -0.0015)

Tabelle A.4.: PFLOTRAN-Einstellungen der Simulationen für 2D und 3D. Die Simulationen unterscheiden sich in Permeabilität und Gebietsgröße.

A.7. Lossfunktionen

Wir verwenden die Lossfunktionen SmoothL1 und MSE Loss mit folgenden Definitionen: $SmoothL1 = \frac{1}{N}\sum_{i=1}^{N} SmoothL1(y_i, \hat{y}_i)$ wobei $SmoothL1(y_i, \hat{y}_i) = \begin{cases} \frac{0.5}{\beta} \cdot (y_i - \hat{y}_i)^2, \text{ für } |y_i - \hat{y}_i| < \beta \\ |y_i - \hat{y}_i| - 0.5 \cdot \beta, \text{ sonst} \end{cases}$ mit der Batch Size N, dem wahren Wert y_i und der Vorhersage $\hat{y}_i$ des i-ten Datenpunktes. Der Schwellenwert β steuert den Übergang vom quadratischen zum linearen Term. Der quadratische Term glättet den AE nahe Null und fördert damit eine genaue Modellanpassung. Für größere Fehler geht die Funktion in den linearen Bereich über und reagiert so weniger empfindlich auf Ausreißer.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Warmbronn, 6.11.24 J. Trick

Ort, Datum, Unterschrift