Technische Hochschule Ulm
University of Applied Sciences

Bachelorarbeit

im Bachelorstudiengang

Wirtschaftsingenieurwesen

Fakultät Produktionstechnik

an der Technischen Hochschule Ulm

Wasser- und Rauscherkennung in der Shearographie-Reifenprüfung mit KI

Erstkorrektor:	Prof. Dr.-Ing. Klaus Schlickenrieder
Zweitkorrektor:	Prof. Dr.-Ing. Hartwig Baumgärtel
Betreuer:	M.Eng. Manuel Friebolin

Vorgelegt von:	Philipp Mario Knauer
Matrikel-Nr.:	299321

Thema erhalten:	02.12.2024
Arbeit abgegeben:	01.04.2025

Eidesstattliche Erklärung

„Diese Abschlussarbeit wurde von mir selbstständig verfasst. Es wurden nur die angegebenen Quellen und Hilfsmittel verwendet. Alle wörtlichen und sinngemäßen Zitate sind in dieser Arbeit als solche kenntlich gemacht."

Ort, Datum Ulm, 23.03.25 P. Kuhn

Unterschrift

Inhalt

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation und Problemstellung

Diese Arbeit entsteht im Rahmen eines Forschungsprojekts zwischen der Technischen Hochschule Ulm und der SDS Systemtechnik GmbH. Ziel des Projekts ist die Digitalisierung und Automatisierung der Shearographie-gestützten Reifenprüfung, um den Prüfprozess effizienter und zuverlässiger zu gestalten.

Die Shearographie ist ein etabliertes Verfahren zur zerstörungsfreien Prüfung von Reifen. Sie macht Defekte und strukturelle Schwachstellen sichtbar, indem sie Deformationen aufgrund von Druckänderungen analysiert. Aktuell erfolgt die Auswertung der Shearographie-Bilder jedoch ausschließlich manuell. Ein Mitarbeiter muss jede Aufnahme einzeln betrachten und bewerten, was nicht nur zeitaufwändig ist, sondern auch eine gewisse subjektive Fehleranfälligkeit mit sich bringt.

Ein besonderes Problem bei der Bildauswertung ist das Auftreten von Artefakten, die die Interpretation der Shearographie-Bilder erschweren können. Insbesondere Wasser und Rauschen treten in unterschiedlichen Ausprägungen auf und können die Sichtbarkeit anderer relevanter Strukturen beeinträchtigen.

Die zunehmenden Anforderungen an Qualität, Effizienz und Automatisierung in der Industrie machen es erforderlich, die Reifenprüfung weiterzuentwickeln. Mit modernen KI-Methoden soll die Shearographie-Analyse nicht nur automatisiert, sondern auch präziser und skalierbar gestaltet werden. In dieser Arbeit wird daher einen Ansatz zur KI-gestützten Erkennung von Wasser und Rauschen untersucht, um eine objektive und zuverlässige Bewertung der Shearographie-Bilder zu ermöglichen.

1.2 Zielsetzung der Arbeit

Ziel dieser Arbeit ist die Entwicklung und Evaluierung eines KI-basierten Klassifikationsmodells, das Wasser und Rauschen in Shearographie-Bildern zuverlässig erkennt. Der aktuelle Prüfprozess erfordert eine manuelle Bildanalyse durch Fachkräfte, was zeitaufwendig ist und zu subjektiven Abweichungen führen kann. Die KI soll diesen Prozess unterstützen, indem sie die betroffenen Bildbereiche automatisch klassifiziert und farblich markiert. Dadurch soll langfristig eine effizientere, objektivere und kostengünstigere Reifenprüfung ermöglicht werden.

Das Modell soll zwischen vier Klassen unterscheiden:

- Water – Bereiche mit eingedrungenem Wasser
- Good – unauffällige, nicht betroffene Bildbereiche
- Medium Noise – mittelstarkes Rauschen
- Strong Noise – stark ausgeprägtes Rauschen

Während bei Rauschen eine Differenzierung nach Intensität erfolgen soll, soll Wasser nur binär klassifiziert werden. Vorerst soll das trainierte KI-Modell als Entscheidungshilfe dienen, indem es Bildbereiche farblich hervorhebt, sodass ein Prüfer die Ergebnisse validieren kann. Eine vollständig automatische Bewertung ohne menschliches Eingreifen wird als langfristiges Ziel angestrebt.

Um die Qualität des Modells zu bewerten, sollen die KI-Ergebnisse mit manuell klassifizierten Bildern verglichen werden. Dabei wird eine quantitative Analyse erfolgen, die aufzeigt, inwiefern die KI mit der menschlichen Bewertung übereinstimmt. Eine visuelle Prüfung der eingefärbten Bildbereiche unterstützt zusätzlich die Bewertung der praktischen Anwendbarkeit.

1.3 Aufbau der Arbeit

Diese Arbeit ist in sieben Hauptkapitel gegliedert.

Kapitel 1 bildet die Einleitung und führt in das Thema ein.

Kapitel 2 behandelt die theoretischen Grundlagen, die für das Verständnis der Arbeit wichtig sind. Zunächst wird die Shearographie in der Reifenprüfung erklärt, wobei ein besonderer Fokus auf die Entstehung und Bedeutung der Artefakte Wasser und Rauschen gelegt wird. Anschließend folgt eine Einführung in Deep Learning und neuronale Netze, in der grundlegende Konzepte des maschinellen Lernens vorgestellt werden. Abschließend wird im Abschnitt Stand der Technik der aktuelle Einsatz von KI in der Bildverarbeitung gesichtet, um den Inhalt dieser Arbeit in bestehende Forschung und industrielle Anwendungen einzuordnen.

Kapitel 3 beschreibt die Methodik, die zur Umsetzung des Modells angewandt wurde. Zunächst wird die Datenerhebung erläutert, also die Auswahl und Beschaffung von Shearographie-Bildern. Daraufhin folgt die Vorverarbeitung der Daten, bei der die Bilddaten gefiltert, normalisiert und annotiert werden, um sie für das Training des neuronalen Netzes vorzubereiten. Danach wird der Aufbau und das Training des Deep-Learning-Modells beschrieben, bevor schließlich die konkrete Implementierung der Klassifikation erläutert wird. Abschließend wird die Validierung des Modells vorgestellt, um die Leistungsfähigkeit der KI anhand definierter Kriterien zu überprüfen.

Kapitel 4 umfasst die Implementierung und die experimentellen Ergebnisse. Hier wird zunächst die Software-Architektur und der Code beschrieben, um einen Einblick in die technische Umsetzung zu geben. Danach wird der Trainingsprozess und die angewandten Optimierungsstrategien erläutert. Ein zentraler Bestandteil dieses Kapitels ist die Analyse der Modellperformance, in der die Genauigkeit des Modells sowie Kennwerte wie Verlust und Genauigkeit untersucht und auf Herausforderungen und Lösungsansätze eingegangen wird. Abschließend erfolgt die Erhebung der Validierungsdaten eines fertig trainierten und optimierten Modells.

Kapitel 5 widmet sich der Diskussion und dem Ausblick. Zunächst werden die Ergebnisse interpretiert, um die Stärken und Schwächen des Modells zu bewerten. Danach folgt eine Analyse der Limitationen der Studie, in der mögliche Einschränkungen der Methodik oder der verwendeten Daten reflektiert werden. Abschließend werden Verbesserungspotenziale

und zukünftige Forschungsperspektiven aufgezeigt, um auf mögliche Weiterentwicklungen des Modells hinzuweisen.

Kapitel 6 fasst die wichtigsten Erkenntnisse der Arbeit zusammen. Zunächst wird ein Fazit gezogen, in dem die wesentlichen Resultate und deren Bedeutung für die Forschung und Praxis zusammengefasst werden. Anschließend wird die Relevanz der Ergebnisse für die Praxis diskutiert, insbesondere im Hinblick auf die Integration des entwickelten Systems in bestehende Prüfprozesse.

Die Arbeit schließt mit dem Literaturverzeichnis (**Kapitel 7**), in dem alle verwendeten Quellen aufgeführt sind.

2 Theoretische Grundlagen

2.1 Grundlagen der Shearographie in der Reifenprüfung

Die Shearografie, in der deutschen Sprache auch als Scherografie bekannt, ist ein optisches, zerstörungsfreies Prüfverfahren zur Erkennung von Materialdefekten. Sie basiert auf der Laser-Speckle-Shearing-Interferometrie und ermöglicht die Analyse mechanischer Spannungen und Deformationen in Bauteilen. In der Reifenprüfung wird die Shearografie eingesetzt, um strukturelle Unregelmäßigkeiten zu identifizieren (Ettenmeyer, 1991, S. 247; Honlet & Walz, 2002, S. 39).

Funktionsweise der Shearografie:

Das Verfahren nutzt kohärentes Laserlicht, um die Oberfläche des Prüfobjekts zu beleuchten. Das reflektierte Licht wird mit einer Kamera aufgenommen, die ein spezielles optisches Element enthält, das eine versetzte Überlagerung (Shearing) der Bildinformationen erzeugt. Dadurch werden minimale Formänderungen sichtbar gemacht (Ettenmeyer, 1991, S. 247–248).

Die Shearografie arbeitet mit einer Doppelbelichtung, bei der zwei Bilder unter unterschiedlichen mechanischen Belastungen aufgenommen werden. Dies kann durch Vakuum, thermische Einflüsse oder mechanische Schwingungen erfolgen. Durch den Vergleich der beiden Bilder entsteht ein Differenzbild, das ausschließlich Veränderungen zwischen den beiden Zuständen zeigt (Ettenmeyer, 1991, S. 248).

Die resultierenden Interferenzmuster erscheinen als helle und dunkle Linien und visualisieren kleinste Deformationen des Materials. Ein gleichmäßiges Streifenmuster deutet auf ein intaktes Material hin, während Unregelmäßigkeiten oder Flecken auf Defekte, Lufteinschlüsse oder Fremdstoffe hindeuten können (Ettenmeyer, 1991, S. 247 & 249).

Anwendung im Bezug auf die Reifenprüfung:

Die Shearographie wird für die Prüfung verschiedenster Reifen eingesetzt. Sie findet beispielsweise bei der Prüfung von Reifen für Schwerlastflugzeuge, Militärjets, Concorde, NASCAR-Rennwagen und anderen schnellen PKW´s Anwendung. Sie wird aber auch zunehmend für Lastkraftwagenreifen und mehrfach runderneuerte Reifen verwendet, um auftretende Sicherheitsrisiken zu senken (Honlet & Walz, 2002, S. 41).

Die Anwendung dieses Verfahrens soll dabei helfen, Artefakte in Materialverbundprodukten aufzudecken, die mit anderen Verfahren möglicherweise nur schwer zu finden wären (siehe Abbildung 1). Es bietet die Gelegenheit, eine materialunabhängige und berührungslose Prüfung durchzuführen, die im losen Zustand des Reifen erfolgt. Die Entscheidung für den Einsatz der Shearografie in der Qualitätskontrolle eines Reifenherstellers oder -erneuerers ist eine reine wirtschaftliche Abwägung (Honlet & Walz, 2002, S. 41)

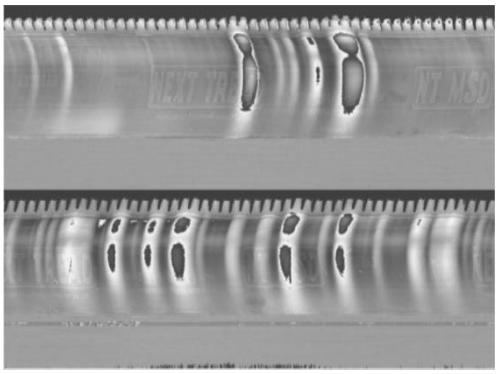

Abbildung 1: Shearographiebild eines druckgeprüften Reifens von SDS (Arno Borchers, 2009)

Im Rahmen dieser Arbeit geht es ausschließlich um die Detektion der Artefakte Wasser und Rauschen. Diese weisen nicht auf einen Missstand am Reifen hin, sondern entstehen in der Bildgebung. Für den Fall, das ein Reifen beispielsweise bei Minusgraden draußen zwischengelagert wird, kann es vorkommen, dass sich Feuchtigkeit am Reifen festsetzt und durch die niedrigen Temperaturen gefriert. Sobald der Reifen nun zur Prüfung in den Prüfraum gebracht wird, kann die festgefrorene Schicht schmelzen und Wasserrückstände am Reifen hinterlassen (siehe Abbildung 2).

Abbildung 2: Shearographiebild mit Wasserrückständen (SDS, 2024)

Ein Rauschen im Bild kann ebenfalls durch einen nassen Bereich im Reifen, oder durch einen kalten beziehungsweise warmen Reifen auftreten (siehe Abbildung 3). Diese Artefakte, die das Shearographie Verfahren hier erkennt, stellen wie bereits erwähnt, keine Missstände am Reifen dar, können aber eventuell andere Artefakte oder Befunde, die zu einem Sicherheitsrisiko führen könnten, verdecken.

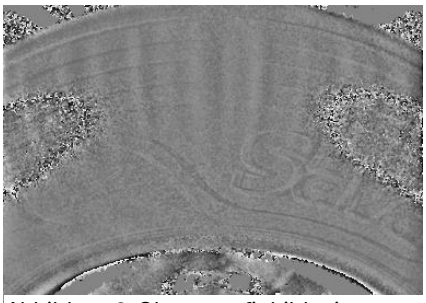

Abbildung 3: Shearografiebild mit starkem Rauschen (SDS, 2024)

2.2 Einführung in Deep Learning und neuronale Netze

Deep Learning gehört zu dem Bereich des maschinellen Lernens, der auf tiefen neuronalen Netzen basiert. Diese mehrschichtigen Netzwerke sind darauf ausgelegt, komplexe Muster und Strukturen zu erkennen und nachzuahmen, ähnlich den Entscheidungsprozessen im menschlichen Gehirn. Es handelt sich um eine Form der künstlichen Intelligenz (KI), deren Ziel es ist, physische und analytische Aufgaben ohne Mitwirkung des Menschen zu automatisieren. Die neuronalen Netze haben im Deep Learning also die wesentliche Aufgabe, das menschliche Gehirn nachzuahmen, indem sie Bias, Gewichte und Dateneingaben (die als Siliziumneuronen fungieren) nutzen, um Gegenstände in den Daten zu erfassen, zu analysieren und zu klassifizieren (Holdsworth & Scapicchio, 2024).

Deep Learning findet Anwendung in verschiedenen Bereichen, in denen große Datenmengen analysiert werden, um zukünftige Ereignisse vorherzusagen oder Prozesse zu automatisieren. Dazu gehören beispielsweise die Spracherkennung, die Bildklassifikation und Segmente der Empfehlungssysteme (Wuttke, 2023a).

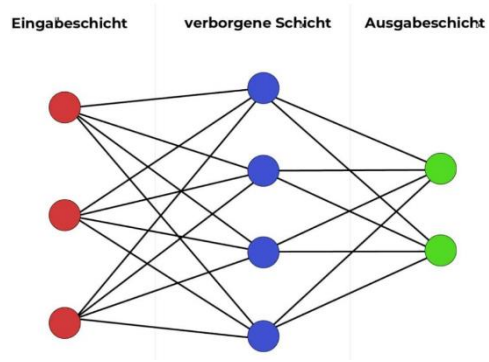

Abbildung 4: Künstliches Neuronales Netzwerk (Wuttke, 2023)

Die neuronalen Netze bilden, wie bereits angesprochen, die Basis des Deep Learnings. Sie setzen sich aus unterschiedlichen vielen verbundenen Schichten von Knoten zusammen (siehe Abbildung 4), die zur Verarbeitung und Übertragung von Informationen dienen. Durch die Eingabeschicht werden die Daten an das neuronale Netz übertragen. Die Ausgabeschicht gibt die vom Netz ermittelte Vorhersage aus und klassifiziert somit den Datensatz. Das sind die beiden Schichten, mit denen der Nutzer interagieren kann. Die verborgene Schicht ist, wie der Name bereits vermuten lässt, für den Nutzer nicht sichtbar. Jeder Knoten (bzw. jedes Neuron), der Bestandteil einer verborgenen Schicht (Hidden Layer) ist, besitzt eine bestimmte Gewichtung und einen Schwellenwert (Holdsworth & Scapicchio, 2024). Einfach ausgedrückt besteht ihre Aufgabe darin, verschiedene Input-Signale zu empfangen und diesen jeweils ein Output-Ergebnis zuzuordnen (Wuttke, 2023a).

Falls der Output einer der Knoten, den besagten Schwellenwert überschreitet, werden durch die Aktivierung dieses Knotens Informationen an die nächste Schicht des Netzes weitergegeben. Falls der zugewiesene Schwellenwert nicht überschritten wird, aktiviert sich der Knoten nicht und sendet somit auch keine Daten weiter (Holdsworth & Scapicchio, 2024).

Der grundlegende Unterschied von Deep Learning zu maschinellem Lernen besteht hauptsächlich aus der Anzahl der verwendeten Schichten. Während beim maschinellen Lernen durchschnittlich ein bis zwei verborgene Schichten zum Einsatz kommen, verwendet ein Deep Learning Netz mindestens drei, meistens aber deutlich mehr Schichten zum Training des Modells. Deshalb reichen einem Deep Learning Modell theoretisch völlig unstrukturierte Eingabedaten, um trotzdem genaue Ergebnisse zu erzielen. Den Vorgang, bei dem das Modell den Eingabedaten wichtige Beziehungen und Merkmale entnimmt, nennt man unüberwachtes Lernen. Beim überwachten Lernen, wie es bei klassischen maschinellen Lernmodellen angewendet wird, sind klar strukturierte und beschriftete Daten erforderlich, um brauchbare Ergebnisse zu erzielen. Je nach Anwendungsfall finden verschiedene neuronale Netze im Deep Learning Anwendung. Es gibt beispielsweise Transformer, Autoencoder, Recurrend Neural Networks (RNNs) und Convolutional Neural Networks (CNNs) (Holdsworth & Scapicchio, 2024).

Von den vier genannten neuronalen Netzen ist das CNN am besten für Objekterkennungs- und Bildklassifizierungsprojekte geeignet. Grundlegend sieht der Aufbau dieses neuronalen Netzes sehr ähnlich aus, wie bei klassischen neuronalen Netzen. Der Unterschied besteht allerdings darin, dass die Rohdaten hier als Bildinformationen angenommen werden. Sie werden von 2D-Matrizen dargestellt, welche die Pixelwerte der Rohdaten beinhalten. Je nach Aufbau des Netzes kommt nun ein Filter, mit einer vorher festgelegten Kerngröße, zum Einsatz, der die Inputs filtert und an die darauffolgende verborgene Schicht weitergibt. Dabei besitzt allerdings nicht jeder Eingang eine Verbindung zu jedem Neuron in der verborgenen Schicht, sondern mehrere Inputs (je nach Kerngröße) werden in lokalen rezeptiven Feldern zusammengefasst und jeweils von einem Neuron der Hidden Layer repräsentiert. Die grundlegenden Bestandteile der verborgenen Schicht eines CNN sind die „Convolutional“, „Normalization“, „Pooling“ und „Activation“ -schichten. Auf eine weiterführende Analyse dieser Schichten wird verzichtet, da sie über den Umfang dieser Arbeit hinausgehen würde (Sönke, 2018, S. 5).

2.3 Architektur und Eigenschaften von Resnet50

Unter einem ResNet50-Modell versteht man ein tiefes Convolutional Neural Network (CNN) das extra für die Bilderkennung entwickelt wurde. ResNet steht dabei für Residual Network, welches einem Deep Learning Modell entspricht, dass in der Computer Vision zum Einsatz kommt. Die Fünfzig steht für die Anzahl der Schichten, die das Netzwerk besitzt (Praveen et al., 2023, S. 322 & 324).

Der grundlegende Unterschied zwischen einem ResNet50 und anderen CNNs ist die Architektur der einzelnen Blöcke und die zusätzliche Verwendung von sogenannten Skip-Verbindungen (He et al., 2015, S. 2). Ein normales CNN besteht aus einer Stapelung von Convolutional Layern, Aktivierungsfunktionen und Pooling Layern. Dabei lernt jede Schicht direkt aus der Ausgabe der vorherigen Schicht (Narwade & Kolhe, 2023, S. 280). ResNet50 verwendet stattdessen Residualblöcke als grundlegende Bausteine. Jeder dieser Blöcke besteht normalerweise aus zwei oder drei Convolutional Layern, einer Batch-Normalisierung und ReLu-Aktivierungen. Außerdem enthalten sie eine Skip Connection (oder auch Shortcut Connection), die Informationen aus einer früheren Schicht direkt an eine spätere weitergeben kann, ohne zuerst durch alle dazwischenliegenden Schichten laufen zu müssen. Sie addiert prinzipiell den Input eines Blocks direkt zu dessen Output (He et al., 2015, S. 2 & 3; Narwade & Kolhe, 2023, S. 280 & 287).

Während normale CNNs die Informationen also sequenziell durch jede Schicht fließen lassen, kann das ResNet50 die Informationen aus früheren Schichten direkt zu den späteren Schichten schicken, ohne das die dazwischenliegenden Transformationen vollständig durchlaufen werden müssen. Das sorgt dafür, dass das Problem des verschwindenden Gradienten umgangen werden kann. Während des Trainings wird stetig der Fehler zwischen der vorhergesagten Ausgabe und der tatsächlichen Ausgabe berechnet. Dieser wird dann rückwärts propagiert damit der Fehler in Zukunft, durch die Anpassung der Gewichte jeder Schicht, reduziert werden kann. Bei tiefen neuronalen Netzen wird der Gradient durch die Ableitung der Aktivierungsfunktionen jeder Schicht multipliziert. Die Werte dieser Ableitungen liegen typischerweise zwischen 0 und 1. Wenn nun also viele dieser kleinen Werte miteinander multipliziert werden, kann das Ergebnis exponentiell klein werden, oder in anderen Worten „verschwinden“. In diesem Fall können die früheren Schichten des Netzwerks keine sinnvollen Merkmale mehr aus den Eingabedaten ziehen. Nur noch die hinteren Schichten erhalten ausschlagegebende Updates während die vorderen stagnieren.

Dieser Fakt kann dazu führen, dass tiefere Netzwerke kaum besser oder sogar schlechter abschneiden als weniger tiefe. Gegenteilig gibt es auch „explodierende“ Gradienten, welche nach und nach exponentiell groß werden. Fakt ist, dass beide dieser Probleme das Training tieferer Netzwerke erschwert, bzw. unmöglich gemacht haben (He et al., 2015, S. 1 & 2; Narwade & Kolhe, 2023, S. 282). Dieses grundlegende Problem konnte durch die Erfindung des Residual Neural Networks beseitigt werden.

Einen weiteren Vorteil stellt die höhere Genauigkeit dar, die ResNet50 oft im Verhältnis zu seinen Vorgängern erzielt. Einige Experimente zeigten, dass das ResNet50 ein einfaches CNN, vor allem bei größeren Datensätzen, in Bezug auf die Klassifikationsgenauigkeit durchweg übertrifft (Sheikh et al., 2024, S. 4610).

Abgesehen davon ist die Nutzung von Rechenressourcen, im Vergleich zu anderen tiefen Netzwerken, beim ResNet50 effizienter. Durch die etablierte Architektur, die bereits für viele Computer-Vision-Projekte genutzt wurde, ist außerdem die Zuverlässigkeit dieses Netzes gewährleistet (Praveen et al., 2023, S. 322 & 326).

Zusammenfassend lässt sich also sagen, dass das ResNet50, durch seine hohe Genauigkeit, die effizient genutzte Rechenleistung, seine hohe Zuverlässigkeit und letztendlich natürlich durch dessen Lösung des Gradienten Problems, bestens für jegliche Aufgaben in der Bildverarbeitung geeignet ist.

2.4 Stand der Technik: Einsatz von KI in der Bildverarbeitung

Künstliche Intelligenz (KI) in der Bildverarbeitung befasst sich mit der Identifikation von Gegenständen oder Mustern in Bildern. Sie gehört zur Computer Vision und ist ein Teilgebiet des Überbegriffs KI (Wuttke, 2023b). Durch Computer Vision erlernen Computer die Fähigkeit zur visuellen Wahrnehmung, indem sie Bilder oder Videos analysieren, um Entscheidungen zu treffen oder Informationen über ihre Umgebung zu erlangen (Buxmann & Schmidt, 2021, S. 65). Der Einsatz von KI in der Bildverarbeitung hat in der letzten Zeit erhebliche Fortschritte gemacht und erreicht in einigen Bereichen bereits ein Leistungsniveau, das mit dem menschlicher Experten vergleichbar oder sogar überlegen ist. (Pouly et al., 2020, S. 667–668).

Die technologischen Grundlagen der Bildverarbeitung mit KI, also neuronale Netze und Deep Learning, wurden im vorherigen Kapitel ausführlich behandelt, jedoch nicht der Trainingsprozess. Für das Trainieren der KI-Modelle sind sehr umfangreiche und

zuverlässige Trainingsdaten erforderlich. Je nach Eigenschaften der eingespeisten Trainingsdaten, kann die Güte und die Entscheidungsqualität der Algorithmen später variieren (Buxmann & Schmidt, 2021, S. 67). Es gibt einige Aspekte, die einen direkten Einfluss auf die Genauigkeit des fertig trainierten Modells haben. Das Datenvolumen ist dabei die offensichtlichste Stellschraube. Ein neuronales Netz benötigt in der Regel sehr viele Bilder, um gute Ergebnisse zu erzielen. Das Modell kann also umso genauer trainiert werden, je mehr beschriftete Bilder es zur Verfügung hat (Marquardt, 2020, S. 731; Wuttke, 2023b). Auch die Datenqualität spielt eine entscheidende Rolle, um aussagekräftige KI-Modelle zu erhalten. Bilder deren Qualität stark schwankt, stellen beim Training des Modells nach wie vor ein großes Problem dar (Buxmann & Schmidt, 2021, S. 69). Die richtige Annotation der Trainingsdaten ist ebenso ein wichtiger Aspekt. Falls dabei bereits Fehler gemacht wurden, ziehen diese sich bis zum Ende durch das gesamte Modell. Deshalb sollte jedes Bild mit einer aussagekräftigen Information gelabelt werden. Ja nach Ziel der Anwendung, kann sich die Art der Annotation unterscheiden. Die Daten können beispielsweise einer binären Klassifikation, also einer Aufteilung in Gut- und Schlechtteile unterzogen werden, oder einer sogenannten multiklassen Klassifikation, bei der nicht nur gut und schlecht existiert, sondern verschiedene Güte-, bzw. Fehlerklassen. Damit das Modell sich selbst überprüfen kann, werden die Eingabedaten schlussendlich automatisch in Trainings- und Testdaten aufgeteilt. (Buxmann & Schmidt, 2021, S. 67).

Bildverarbeitende Verfahren, die auf KI aufbauen, kommen vor allem in der Industrie und in der Medizin häufig zum Einsatz, da sie die geforderte Genauigkeit erreichen, bzw. bereits übertreffen. In der Industrie werden sie beispielsweise für die Qualitätskontrolle (Marquardt, 2020, S. 731), für visuelle Inspektionen (Buxmann & Schmidt, 2021, S. 75), für die Unterstützung von Produktionsprozessen, als Instrument der Industrie 4.0 und für etliche weitere Aufgabenfelder genutzt (Buxmann & Schmidt, 2021, S. 205–206).

In der Medizin spielen KI gestützte Bildverarbeitungsverfahren bereits auch eine sehr große Rolle. Sie dienen unter anderem der Erkennung etlicher Krankheiten, wie zum Beispiel Brustkrebs (Pouly et al., 2020, S. 660). Es werden auch bereits einige Verfahren bei der Behandlung von Kindern angewendet, welche vielversprechende Ansätze zur Erleichterung der Arbeit und potenziellen Erhöhung der Befundsicherheit bieten. Wie auch in der Industrie, gibt es hier noch unzählbar viele weitere Anwendungsbereiche (Mentzel, 2021, S. 694 & 697). Allerdings profitieren nicht nur die Industrie und die Medizin von der Nutzung von KI. Etliche andere Bereiche setzen diese ebenfalls ein, um Aufgaben zu übernehmen, bei

denen herkömmliche, regelbasierte Systeme an ihre Grenzen stoßen (Pouly et al., 2020, S. 660). Dazu zählen beispielsweise das Bildungswesen, die Kriminalitätsbekämpfung und der Einzelhandel (Wuttke, 2023b).

Die Anwendung von KI-basierter Bildverarbeitung unterliegt natürlich auch einigen Grenzen und Herausforderungen. Ein wesentlicher Faktor ist die Bereitstellung von Trainingsdaten. Welche Eigenschaften diese erfüllen müssen wurde bereits geklärt. Um ein Ergebnis mit hoher Genauigkeit zu erzielen, muss also oft erst einmal sehr viel Vorarbeit geleistet werden und es muss außerdem eine sehr große Menge an Bildern zum Trainieren zur Verfügung stehen (Pouly et al., 2020, S. 665). Eine weitere Hürde stellt die Erklärbarkeit der Modelle dar. Die Ergebnisse setzen sich aus Millionen von Parameterwerten, oft tagelanger Rechenzeit und einer endlosen Anzahl an Eingabedaten zusammen. Für den Menschen ist es häufig sehr schwierig oder sogar unmöglich, nachzuvollziehen, warum das Modell bestimmte Entscheidungen trifft. Es ist zu diesem Zeitpunkt noch nicht möglich nachzuvollziehen, wie es zu einem bestimmten Resultat gekommen ist. Es wird allerdings sehr aktiv an einer Lösung dieses Problems, genauer gesagt einer Explainable Artificial Intelligence (XAI) geforscht (Pouly et al., 2020, S. 666). Auch die Aktualisierung der Algorithmen spielt eine große Rolle. Da momentan so aktiv an KI weitergeforscht wird, kommen ständig neue Innovationen auf den Markt, die einen Prozess unter Umständen deutlich erleichtern. Um also wettbewerbsfähig zu bleiben und weiterhin solide Resultate zu generieren, sollten die Algorithmen stehts auf einem aktuellen Stand gehalten werden (Wuttke, 2023b). Es gilt außerdem zu gewährleisten, dass personenbezogene Daten niemals in die falschen Hände geraten. Aufgrund der zuvor angesprochenen Intransparenz der Vorgänge in der Bildverarbeitung mit KI, ist das jedoch keine Selbstverständlichkeit (Buxmann & Schmidt, 2021, S. 53). Abschließend ist es wichtig zu erwähnen, dass KI in einigen Bereichen, Stand jetzt, nicht als Allheilmittel betrachtet werden sollte, sondern eher als ergänzendes Werkzeug (Marquardt, 2020, S. 733).

Zusammenfassend lässt sich sagen, dass die KI-basierte Bildverarbeitung in der Industrie und in jeglichen anderen Bereichen, zukünftig eine immer größer werdende Rolle spielen wird. Die außerordentlichen Möglichkeiten, die die künstliche Intelligenz mit sich bringt, überwiegen die bestehenden Risiken durchaus. Unter Berücksichtigung einiger ethischen Aspekte und der Förderung der Zusammenarbeit zwischen Mensch und Maschine, kann sie ohne Zweifel gewinnbringend eingesetzt werden (Buxmann & Schmidt, 2021, S. 200 & 204).

3 Methodik

3.1 Datenerhebung: Shearographie-Bilder von Reifen

Die Shearographie-Bilder, die für diese Arbeit verwendet werden, stammen direkt aus der Datenbank der SDS Systemtechnik GmbH. Es handelt sich hierbei nicht um speziell generierte Testbilder, die nur zu Forschungszwecken erstellt wurden, sondern um reale anonymisierte Aufnahmen aus der Praxis, die bereits von Mitarbeitern gesichtet und bewertet wurden. Die zur Verfügung gestellte Datenmenge ist mit etwa 150 000 Bilder enorm groß und stellt somit ein gutes Verhältnis zwischen Gutbildern und Bildern mit Artefakten und Befunden dar. Falls der Bedarf bestehen würde, könnten jederzeit weiter Bilddaten aus einem Pool von über zwei Millionen Bildern bezogen werden.

Bildaufnahmeprozess und Randbedingungen:

Die Aufnahme der Bilder findet in einer speziell dafür vorgesehenen Prüfmaschine statt. Der Reifen wird in dieser positioniert und rotiert anschließend um die eigene Achse, um alle verschiedenen Bereiche des Reifens und deren Oberfläche erfassen zu können. Die Kamera (gegebenenfalls auch mehrere), die sich an einem beweglichen Arm befindet, wird an einen Punkt über oder innerhalb des Reifens positioniert und während der Aufnahme der Bilder nicht weiterbewegt. Dadurch kann gewährleistet werden, dass die Bilder alle aus demselben Winkel und im selben Abstand zum Reifen entstehen.

Von jedem Reifen werden insgesamt 48 Bilder erstellt. 16 davon stellen die Seitenwand (Sidewall) des Reifens dar und 8 die Krone (Crown). Die restlichen 24 Bilder spielen für diese Arbeit keine weitere Rolle.

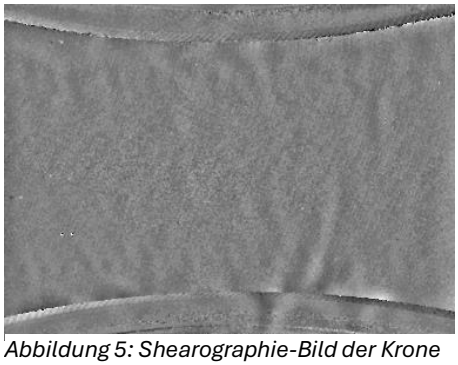

Abbildung 5: Shearographie-Bild der Krone eines Reifens (SDS, 2024)

Krone: Sie stellt die Innenseite der Lauffläche des Reifens dar. Die Aufnahme erfolgt durch eine oder mehrere Kameras, die in die Öffnung des Reifens gefahren und fest dort positioniert werden. Durch das Drehen des Reifens um die eigene Achse, kann die gesamte Krone auf insgesamt 8 Bildern dargestellt werden. Auf einigen Bildern kann man, den Übergang zur Seitenwand erkennen (siehe Abbildung 5).

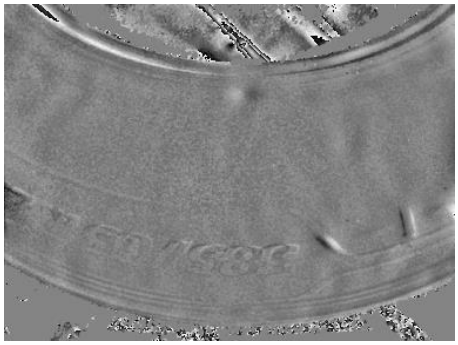

Abbildung 6: Shearographie-Bild der Seitenwand eines Reifens (SDS, 2024)

Seitenwand: Wie der Name schon vermuten lässt, wird hier der Reifen von der Seite abgelichtet. Da der Reifen in der Prüfmaschine liegt, wird die Kamera dementsprechend über dem Reifen positioniert. Es ist gut zu erkennen, dass auf dieser Art von Bild, ein Teil vom Hintergrund dargestellt wird, der gar nicht zum Reifen gehört (siehe Abbildung 6). Dieser Umstand erweist sich im Verlauf dieser Arbeit als Problem, das jedoch in den folgenden Kapiteln ausführlich erläutert und gelöst wird.

Die Aufnahmen der Reifen erfolgen in der Qualitätskontrolle. Der Prüfer, der die erstellten Bilder analysiert, muss selbstverständlich auf Bilder, die Artefakte oder Befunde aufweisen reagieren.

Datenverwaltung und Dateistruktur:

Die zur Verfügung gestellten Bilder liegen im PNG-Format vor und besitzen eine Auflösung von 284x287 Pixeln. Wie genau die Speicherung und Verwaltung der Daten firmenintern abläuft, ist nicht bekannt. Es wurde allerdings sichergestellt, dass jedes Bild eine Beschriftung erhält, die so kein zweites Mal vorkommt. Diese Beschriftung setzt sich folgendermaßen zusammen:

- **trb oder trc** (internes Identifikationsmerkmal)
- **390** (internes Identifikationsmerkmal)
- **Jahr, Monat, Tag, Stunde, Minute, Sekunde** der Aufnahme
- **Eine „3“ als fester Bestandteil** (internes Identifikationsmerkmal)
- **Aufsteigende Zahlen von 1 bis 48**
- **Dateiendung .png**

Die Beschriftung eines Bildes sieht also beispielsweise aus wie folgt: „trb_390_240108153445_3_41.png“. Das Bild wurde dementsprechend am 08.01.2024 um 15:34 Uhr und 45 Sekunden aufgenommen. Es handelt sich um das 41. Von insgesamt 48 Bildern, die von dem Reifen erstellt wurden.

Datenmenge und Diversität

Wie bereits angesprochen, ist die zur Verfügung gestellte Datenmenge, mit knapp 150 000 Bildern recht groß. Der Datensatz enthält überwiegend Bilder ohne sichtbare Artefakte oder Befunde. Das liegt selbstverständlich an dem Fakt, dass ein Großteil der geprüften Reifen kein Sicherheitsrisiko aufweist. Die Bilder, die sich nach der visuellen Überprüfung als unproblematisch herausgestellt haben, spielen in dieser Arbeit keine Rolle.

Durch die große Menge an Daten, kann eine hohe Variabilität in den Bildaufnahmen gewährleistet werden. Bei so vielen Bildern, sollte eine breite Palette von Umgebungsbedingungen, Reifenmodellen und anderen Einflussfaktoren bei der Bildaufnahme abgedeckt sein. Auf diesem Weg können beim Trainieren des Modells beispielsweise minimale Schwankungen in der Beleuchtung, Unterschiede in der Materialbeschaffenheit, oder Reifen mit unterschiedlichen Ausgangstemperaturen, indirekt berücksichtigt werden.

Außerdem enthalten die Bilder dadurch eine Vielzahl möglicher Artefakte, die bei den Shearographie-Aufnahmen der Reifen auftreten können. Die exakte Verteilung dieser Artefakte innerhalb der bereitgestellten Datenmenge ist nicht bekannt. Diese ist für die weitere Vorgehensweise allerdings auch nicht von Belang, da ausreichend Bilder mit Wasser und Rauschen zur Verfügung stehen.

Manuelle Annotation der Bilder:

Bisher läuft die Klassifikation der Bilder, wie zuvor beschrieben, über eine Sichtprüfung durch einen Mitarbeiter. Dieser kennzeichnet Reifen mit erkannten Artefakten und sorgt für die Einleitung von Maßnahmen, um den Reifen bei risikofreien Artefakten (wie beispielsweise Wasser oder Rauschen auf dem Bild) erneut zu fotografieren oder zu vermerken, dass die Prüfung nicht ausgeführt werden konnte. Bei Befunden, die ein Sicherheitsrisiko darstellen (wie z.B. Separationen), wird der Reifen entsorgt, bzw. kenntlich gemacht, dass dieser unbrauchbar ist. Bei Fehlinterpretationen entsteht dementsprechend ein Mehrausschuss.

Um eine belastbare Datengrundlage für die weitere Analyse und das Training des neuronalen Netzes zu schaffen, ist eine gezielte Auswahl relevanter Bilder notwendig. Die Kriterien für diese Auswahl sowie die durchgeführten Vorverarbeitungsschritte werden im nächsten Kapitel ausführlich behandelt.

3.2 Vorverarbeitung der Daten (Filterung, Normalisierung, Annotieren)

Die richtige Vorabverarbeitung der Eingabedaten ist ein dringend notwendiger Schritt, um die Shearographie-Bilder für das Training eines neuronalen Netzes vorzubereiten. In diesem Abschnitt wird die Auswahl relevanter Bilder, die Filterung und Bereinigung des Datensatzes, die Skalierung und Normalisierung sowie die Annotation der Bilder und deren Aufteilung in Trainings- und Testdaten beschrieben.

Auswahl relevanter Bilder:

Die für diese Arbeit relevanten Bilder werden nach der subjektiven Einschätzung des Autors ausgewählt. Es dienen allerdings Vorlagen, die die charakteristischen Merkmale der Artefakte Water und Noise zeigen, als Referenz. Die Vorselektion der Bilder erfolgt rein visuell, ohne den Einsatz einer speziellen Software.

Die ursprünglichen Bilddaten werden vorerst in 4 verschiedene Kategorien aufgeteilt:

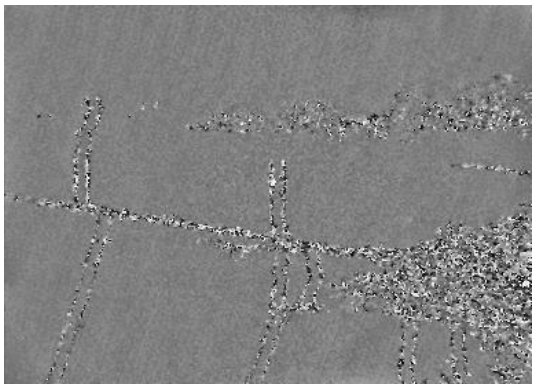

Abbildung 7: Shearographie-Bild des Artefakts "Water" (SDS, 2024)

Water: Wasser ist hauptsächlich in der Reifenkrone zu finden. Aufnahmen, die Wasser enthalten, sind einfach zu identifizieren. Man kann das Wasser durch die schwarzen und weißen Pixel, die sich stark vom Grauton des restlichen Bildes abheben, erkennen (siehe Abbildung 7). Es handelt sich oft um Wassertropfen, die sich am Gummi des Reifens lang bewegen und dabei Wasserrückstände hinterlassen. Auf einigen Bildern sind aber auch Wasserlachen zu erkennen, die beinahe das gesamte Bild einnehmen. Da die Bilder nach Datum sortiert sind und außerdem die 8 Bilder, die von der Krone gemacht werden, immer direkt aufeinander folgen, haben sich die Bilder, die Wasser enthalten, meistens an einigen Stellen in den Ursprungsdaten gebündelt. Vor allem am Anfang des Jahres 2024, also noch während der Wintermonate, befinden sich zwischen dem 18.01. und dem 01.03 sehr viele Bilder mit Wasserrückständen in dem Datensatz. Falls Reifen also draußen zwischengelagert werden und beispielsweise Wasser an diesen festfriert, taut dieses im Prüfraum wieder auf und erzeugt dadurch die Artefakte in den Shearographie-Bildern. Es ist davon auszugehen, dass diese Reifen nacheinander

geprüft wurden und die Prüfergebnisse dieser Tage dementsprechend mehrere Bilder mit Wasserrückständen beinhalten.

Bis zu diesem Zeitpunkt ist das Artefakt Rauschen nicht weiter untergliedert. Es besteht allerdings die Anforderung, dass das Vorkommen von Rauschen, in seiner Stärke unterschieden wird. Deshalb wird es vorerst in folgende drei Klassen unterteilt:

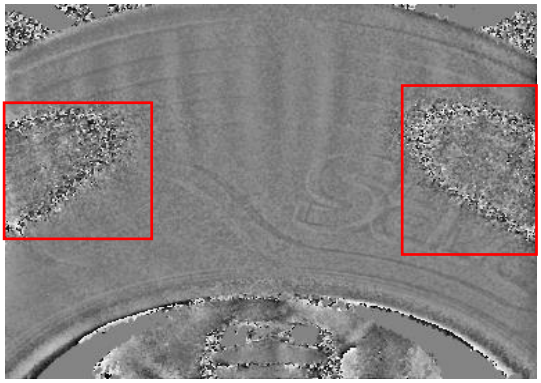

Abbildung 8: Shearographie-Bild des Artefakts "Strong Noise" (Eigene Darstellung in Anlehnung an SDS, 2024)

Strong Noise: Als starkes Rauschen werden Stellen im Bild definiert, die meistens mittig, auf der rechten und linken Seite der Bilder der Seitenwand auftreten (siehe Abbildung 8). Wie beim Wasser sieht man an diesen Stellen im Bild, wesentlich dunklere bzw. hellere Pixel als im restlichen Bild. Trotz der prinzipiellen Ähnlichkeit der Pixel, zu den Bildern mit Wasservorkommen, ist ein deutlicher Unterschied in der Struktur des Vorkommens zu erkennen.

Abbildung 9: Shearographie-Bild des Artefakts "Medium Noise" (Eigene Darstellung in Anlehnung an SDS, 2024)

Medium Noise: Als mittleres Rauschen werden die Stellen im Bild definiert, die ebenfalls meist auf der linken und rechten Seite, mittig im Bild zu sehen und deutlich dunkler als der sonstige Grauton des Bildes sind (siehe Abbildung 9). Oft gibt es auf Bildern mit starkem Rauschen einen fließenden Übergang in mittleres Rauschen. Andersrum ist das allerdings nicht der Fall.

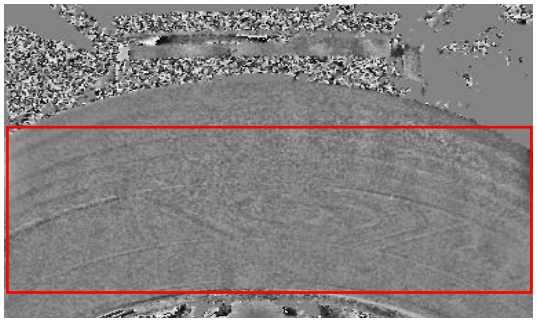

Abbildung 10: Shearographie-Bild des Artefakts "Slight Noise" (Eigene Darstellung in Anlehnung an SDS, 2024)

Slight Noise: Eine leichte Verpixelung, die sich durch das gesamte Bild zieht, wird als schwaches Rauschen definiert (siehe Abbildung 10). Da dieses allerdings in fast allen Bildern vertreten ist und letztendlich keinen negativen Einfluss auf die Sichtbarkeit anderer Artefakte hat, wird diese Klasse aus der Arbeit entfernt.

Der Ursprungsdatensatz wird nun manuell nach 3 (Water, Medium Noise und Strong Noise) der ursprünglichen 4 oben erläuterten Kategorien abgesucht. Falls ein Bild eines der genannten Artefakte beinhaltet, wird es in einen extra Ordner verschoben. Es werden allerdings nicht alle 150 000 Bilder manuell analysiert. Sobald eine ausreichende Anzahl an Bildern pro Artefakt gespeichert ist (etwa 100 pro Klasse), kann auf die weitere Suche nach diesem Artefakt verzichtet werden.

Bildfilterung und Bereinigung:

Wie bereits angesprochen, werden nicht alle Bilder für die weitere Verarbeitung genutzt. Die Bilder, die andere Artefakte oder Befunde als „Water“ oder „Noise“ beinhalten, sind für diese Arbeit irrelevant und werden ausgeschlossen, da sie sonst zu Problemen bei der Klassifikation führen können.

Dadurch, dass die Dateien, anhand des Datums, der genauen Uhrzeit und einer Nummer zwischen 1 und 48, eindeutig identifizierbar sind, ist sichergestellt, dass kein Bild in den Trainingsdaten doppelt vorkommt. Der Computer hätte für den Fall, dass eine Datei bereits im Zielordner vorhanden ist, automatisch eine Meldung ausgegeben, die davor warnt, dass dieser Dateiname bereits existiert. Während der Auswahl der Bilder wird die ursprüngliche Benennung nicht verändert. Erst beim Annotieren dieser, wird ein weiteres Identifikationsmerkmal hinzugefügt. Das wird aber im nächsten Abschnitt genauer erläutert.

Es existieren ein paar Bilder, die durch eine fehlerhafte Aufnahme unbrauchbar sind. Außerdem gibt es einige Aufnahmen, auf denen die Artefakte nicht eindeutig identifizierbar, bzw. nicht direkt als solche erkennbar sind. Diese Aufnahmen werden nicht weiter berücksichtigt, da sonst Probleme beim Training des neuronalen Netzes auftreten können.

Annotation der Bilder:

Zu Annotation der Bilder wird ein extra Programm in Python entwickelt, welches das Ursprungsbild in 16 Kacheln aufteilt und jede dieser Kacheln einzeln durch die manuelle Auswahl des Benutzers labelt. Dabei werden alle Kacheln, die nicht manuell ausgewählt werden, automatisch als „Good“ abgespeichert. Die ausgewählten Kacheln, werden je nach Artefakt mit der Dateiendung „Water“, „Strongnoise“ oder „Mediumnoise“ versehen. Durch das Klicken auf einen Button, wird also die Auswahl des Nutzers gespeichert und die umbenannten Kacheln in einem neuen Ordner abgespeichert. Der neue Dateiname setzt sich ab diesem Zeitpunkt also aus dem ursprünglichen Namen, einer weiteren Zahl von 1 bis 16 und der Benennung des vorliegenden Artefakts zusammen. Um das Beispiel des

Dateinamens von Kapitel 3.1 noch einmal aufzugreifen, sieht dieser nach der Annotation aus wie folgt: „trb_390_240108153445_3_41_4_Water.png“. Dadurch kann sichergestellt werden, dass das neuronale Netz die Kachel beim Einlesen der Daten für den Trainingsprozess, eindeutig mit dem Artefakt Wasser in Verbindung bringt.

Nach der Annotation der ersten Bilder, ist schnell aufgefallen, dass die Unterteilung des Originalbildes in sechzehn Einzelbilder noch wesentlich zu grob ist. Da die Klassifikation der Bilder später je Kachel erfolgen soll, ist der Anspruch, dass diese so genau wie möglich ist und die Kacheln dementsprechend klein sind. Andernfalls würden Kacheln, die vielleicht zu neunzig Prozent „Good“ sind, anhand von beispielsweise zehn Prozent „Water“ auf der Kachel, auch als „Water“ klassifiziert werden. Um die Artefakte also genauer erfassen zu können, wird die Kachelanzahl der Bilder auf vierundsechzig erhöht. Die Anpassungen, die dafür an dem erstellten Annotier Programm vorgenommen werden müssen, sind gering. Es müssen lediglich die Zahlen, durch die die Höhe und Breite des Bildes geteilt werden, jeweils von 4 auf 8 umgeändert werden. Die Zusammensetzung des Dateinamens bleibt gleich, bis auf die neu eingeführte Identifikationszahl, die nun nicht mehr von 1 bis 16, sondern von 1 bis 64 zählt.

Da die Auflösung der einzelnen Bilder selbst bei einer Aufteilung in vierundsechzig Kacheln noch völlig ausreichend zum Trainieren eines neuronalen Netzes ist, ist ziemlich schnell die Entscheidung gefallen, das Gitter noch feiner zu gestalten. Um Artefakte später möglichst genau lokalisieren und hervorheben zu können, wird die Kachelanzahl pro Bild von vierundsechzig auf zweihundertsechsundfünfzig angehoben. Die Vorgehensweise beim Abändern des Annotier Programms ist dabei identisch zu der vorherigen Anpassung. Die neu entstandenen Kacheln haben nun eine Auflösung von 24x17, was jeweils der ursprünglichen Bildhöhe, bzw. Breite, geteilt durch 16 entspricht. Die Auflösung der entstehenden Kacheln ist nach wie vor ausreichend für das folgende Training. Die Artefakte können bei dieser Kachelgröße wesentlich genauer eingegrenzt werden (siehe Abbildung 11). Jede Kachel, die vom Nutzer ausgewählt wird, wird unter der Bezeichnung des Artefakts auf dem Bild abgespeichert. Die nicht ausgewählten folglich als „Good“.

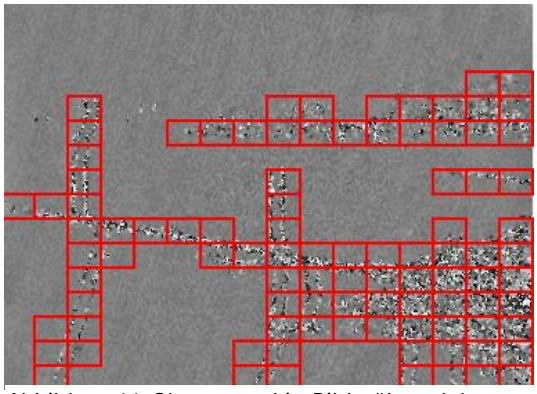

Abbildung 11: Shearographie-Bild während der Annotation (Eigene Darstellung in Anlehnung an SDS, 2024)

Bildskalierung und Normalisierung:

Das für das Training verwendete ResNet50-Modell erfordert eine Eingabegröße von 224x224 Pixeln. Um diese Werte zu erreichen, müssen die einzeln eingespeisten Kacheln (24x17 Pixel) entsprechend skaliert werden. Da die Kacheln nicht quadratisch sind, wird eine Zero-Padding-Methode angewendet, die oben und unten einen schwarzen Hintergrund einfügt. Falls allerdings bereits quadratische Bilder importiert werden, wird dieser Schritt automatisch übersprungen. Anschließend wird das quadratische Bild auf die geforderte Größe 224x224 Pixel skaliert.

Durch die Skalierung kommt es zu einem gewissen Qualitätsverlust, dieser stellt jedoch für das Training des neuronalen Netzes kein Problem dar. Da das ResNet50-Modell für das Modelltraining Bilder erwartet, die einen RGB-Farbmodus als Eingabeformat besitzen, werden die Graustufenbilder durch das Programm entsprechend angepasst.

Um die Bilder zu normalisieren, wird für jeden Farbkanal der Mittelwert abgezogen und durch die Standardabweichung geteilt. Dadurch sind die Kompatibilität der Daten mit vortrainierten Modellen und die Stabilität des neuronalen Netzes sichergestellt.

Datenaufteilung für das Training:

Zum optimalen Training des Modells, gibt es eine Unterteilung der Eingangsdaten in Trainings- und Test-/Validierungsdaten. Dabei werden achtzig Prozent der Daten für das Training des Modells verwendet und die restlichen zwanzig Prozent für die anschließende Validierung der aktuellen Epoche.

Um eine gute Balance sicherzustellen, wird ein besonderes Augenmerk auf die gleichmäßige Verteilung der verschiedenen Klassen gelegt. Keine der Klassen soll überrepräsentiert werden, weswegen darauf geachtet wird, dass alle Kategorien in einem vergleichbaren Umfang in den Datensatz eingehen. Dadurch wird das Risiko der Verzerrung im Training minimiert.

Dass keine Bilder vom selben Reifen in unterschiedlichen Datensätzen auftauchen, wird durch die Beibehaltung der ursprünglichen Benennung gewährleistet. Da alle Bilder eindeutig beschriftet sind, kann ausgeschlossen werden, dass sich identische Bilder sowohl im Trainings- als auch im Testdatensatz befinden.

3.3 Aufbau und Training des Deep-Learning-Modells

Für das Trainieren der Bilder wird ein bereits vortrainiertes ResNet50-Modell über Py Torch (Version 2.5.1) implementiert. Diese Auswahl wurde vorgegeben, da diese Arbeit einen Teil eines größeren Projekts darstellt, in welchem einheitlich das ResNet50 als neuronales Netz genutzt werden soll. Es gehört zu den tieferen neuronalen Netzen, ist aber dank seiner Residual-Verbindungen trotzdem stabil zu trainieren. Im Vergleich zu ResNet101 oder ResNet152 benötigt es weniger Speicher und Rechenzeit, besitzt aber dennoch eine hohe Genauigkeit und Effizienz. ResNet-Architekturen haben bereits in Wettbewerben wie beispielsweise ImageNet ihre Leistung unter Beweis gestellt. Aus genannten Gründen ist das ResNet50 eine gute Wahl für die erfolgreiche Umsetzung dieser Arbeit.

Dadurch, dass das Modell bereits vortrainiert ist, kann sicherstellt werden, dass der Rechenaufwand während des Trainings minimiert wird und es von Merkmalen profitieren kann, die es bereits gelernt hat. Das erleichtert das Feintuning für die spezifische Aufgabe und führt zu einer schnelleren Konvergenz. Das vortrainierte Modell wird nahezu unverändert übernommen. Da die „Fully Connected Layer“, also die Klassifikationsschicht, aber bisher für 1000 Klassen ausgelegt ist, muss diese letzte Schicht, auf vier Neuronen, also die bereits angesprochenen Klassen „Good, Water, Medium Noise und Strong Noise“ reduziert werden.

Anschließend wird das vortrainierte Modell auf den verfügbaren Rechner übertragen, um es auf der CPU (oder falls vorhanden der GPU) ausführen zu können. Es handelt sich dabei um einen „AMD Ryzen 7 2700x Eight-Core Processor“.

Um das Modell darauf zu trainieren, möglichst präzise Wahrscheinlichkeitsverteilungen für die unterschiedlichen Klassen auszugeben, wird eine Cross-Entropy-Verlustfunktion eingesetzt. Diese kommt vor allem in Verbindung mit neuronalen Netzen, die mit der Softmax-Funktion arbeiten, zum Einsatz. Softmax ist eine Methode zum Umwandeln von Netzwerkausgaben in Wahrscheinlichkeiten, die oft in der Mehrklassen-Klassifikation angewendet wird. Im Grunde wird durch Cross-Entropy die Unähnlichkeit zwischen der vorhergesagten und der tatsächlichen Wahrscheinlichkeitsverteilung über die verschiedenen Klassen gemessen. Wenn die Cross-Entropy minimiert wird, kann die vorhergesagte Wahrscheinlichkeit der korrekten Klasse maximiert werden (Mao et al., 2023, S. 3).

Als Optimierer verwendet das Netz Adam (Adaptive Moment Estimation). Das ist ein adaptiver Optimierungsalgorithmus, der sehr häufig zum Training von tiefen neuronalen Netzen eingesetzt wird. Ohne nun zu genau auf die Funktionsweise dieses Optimierers einzugehen, sorgt dieser für die individuelle Anpassung der Lernraten für jedes einzelne Gewicht in einem neuronalen Netz, basierend auf der Richtung und der Größe vergangener Gradienten. Dabei erhalten Parameter mit großen Gradienten tendenziell kleine Lernraten und umgekehrt (Ajani & Bharadwaj, 2019, S. 1). Adam hat bereits in einigen Szenarien eine wesentlich bessere Optimierungsleistung als beispielsweise der alternative Optimierer Stochastic Gradient Descent (SGD) gezeigt (Gupta et al., 2021, S. 1). Dieser steht zwar als Option im Programm zur Verfügung, wurde jedoch von einem Tool zu Hyperparameterfindung (Optuna), auf welches im Laufe dieser Arbeit noch genauer eingegangen wird, als weniger effizient bewertet als Adam. Da er adaptive Lernraten verwendet, ist er bestens für nicht stationäre Probleme geeignet und funktioniert gut, ohne dass die Lernrate stark verfeinert werden muss. Deshalb wird das Training dementsprechend mit Adam als Optimierer durchgeführt.

Um die Konvergenz des Modells zu verbessern, wird ein Lernraten-Scheduler eingesetzt, der nach jeweils 5 Epochen die Lernrate auf zehn Prozent der bisherigen Lernrate senkt. Falls also beispielsweise mit einer Lernrate von 0.1 in den Trainingsprozess gestartet wird, wird diese mit Abschluss der Epoche 5 auf 0.01 gesenkt. Das sorgt dafür, dass erst schnellere Updates erfolgen und später eine präzisere Optimierung möglich ist.

Um Overfitting zu vermeiden und Rechenzeit zu sparen, wird eine Early-Stopping-Bedingung eingebaut, die nach 5 Epochen ohne Verbesserung der Verlustrate den Trainingsprozess abbricht. Overfitting kann sich beispielsweise durch die Verschlechterung der Leistung auf dem Validierungsdatensatz äußern. In diesem Fall ist das Konvergenzkriterium erfüllt und das Modell lernt die Trainingsdaten nur noch auswendig (Goodfellow et al., 2016, S. 273 & 274).

Zur Lösung des Problems der „explodierenden“ Gradienten, wird eine „Gradient Clipping“ Funktion eingebaut, welche die Gradienten beschneidet und sie damit begrenzt. Dabei wird die Norm (der Betrag) der Gradienten, für alle Parameter des Modells überprüft. Falls der ausgegebene Wert größer ist als 1.0, wird die Norm demensprechend so skaliert, dass sie genau 1.0 beträgt. Dadurch können die Gradienten nicht zu groß werden und das Modell kann stabiler und effizienter trainiert werden.

Beim Importieren der Bilder in das Programm, muss eine Batchgröße vom Nutzer angegeben werden, die bestimmt, in wie viele verschiedene Batches die Eingangsbilder aufgeteilt werden. Es wird von Anfang an mit einer Batchgröße von 16 gearbeitet, da diese einen optimalen Kompromiss zwischen Effizienz und Performance darstellt. Sie erfordert weniger Speicher und ist besser für CPUs geeignet. Größere Batch-Sizes könnten zwar die Konvergenz stabilisieren, würden dafür aber auch wesentlich mehr Speicher benötigen.

Beim Festlegen der Lernrate kann es vorkommen, dass das Training durch eine zu niedrige Lernrate unnötig verlangsamt wird, oder das Model durch eine zu hohe Lernrate nicht konvergieren kann. Deshalb wird eine Startlernrate von 0.001 angesetzt, welche gängig für tiefe neuronale Netze in Verbindung mit dem Adam Optimierer ist.

Das Training wird mit zwanzig Epochen ausgeführt. Eine zu niedrige Anzahl an Epochen, könnte dafür sorgen, dass das Model nicht vollständig trainiert wird, eine zu hohe Anzahl wiederum würde das Training unnötig in die Länge ziehen und die Gefahr von Overfitting erhöhen.

Nach jeder Epoche erfolgt direkt anschließend die Validierung dieser. Die Genauigkeit und der Verlust während des Trainings sind zwar wichtige Kennzahlen, aber die Kennzahlen Validierungsverlust pro Epoche und Validierungsgenauigkeit pro Epoche sind ebenso wichtig, um die Performance des Modells auf Bildern zu testen, die dem Netz bisher noch nicht bekannt sind. Dabei wird verglichen, welche Klassifizierung das Modell den Bildern des Testbatches zugewiesen hat und welche manuelle Klassifikation ursprünglich durch die Person festgelegt wurde, die das Bild gelabelt hat. Umso mehr Klassifikationen übereinstimmen, desto höher ist die Genauigkeit. Falls die Genauigkeit nach einigen Epochen anfängt zu sinken, ist das klar ein Zeichen dafür, dass das Netz die Trainingsbilder auswendig lernt und unbekannte Bilder nicht mehr akkurat klassifizieren kann.

Während des gesamten Trainingsprozesses werden verschiedene Kennzahlen an das Tensorboard übertragen. Dazu gehören der Verlust pro Batch, der Verlust pro Epoche, die Genauigkeit pro Epoche und die Lernrate pro Epoche. Diese Werte werden nach jeder Epoche bzw. nach jedem Batch, neu berechnet und im jeweiligen Schaubild vom Tensorboard visualisiert. Die Kennzahlen der anschließenden Validierung, also der Validierungsverlust und die Validierungsgenauigkeit pro Epoche, können ebenfalls über das Tensorboard eingesehen werden.

3.4 Implementierung der Klassifikation

Nach dem Training des Deep Learning Modells gilt es nun die Klassifizierung der Shearographie-Bilder umzusetzen. Das Ziel ist, dass Artefakte, die auf den Bildern zu sehen sind, klassifiziert und visuell gekennzeichnet werden.

Dazu werden direkt zu Beginn die Bilder aus dem vom Nutzer ausgewählten Ordner importiert. Anschließend wird zufällig eines der Bilder ausgewählt, das für den Klassifikationsprozess geöffnet wird. Die zufällige Auswahl sorgt dafür, dass die Bilder komplett durchgemischt klassifiziert werden und nicht zuerst alle Bilder mit Wasser und anschließen alle Bilder, die ein Rauschen aufweisen.

Die Shearographie-Bilder, die in der Praxis verwendet werden, haben wie bereits erwähnt eine Größe von 384x287 Pixeln. Da die Bilder aber für den Trainingsprozess in 256 verschiedene Kacheln aufgeteilt werden und das Modell deshalb natürlich auch nur diese Art von Bild klassifizieren kann, müssen die Ursprungsbilder für den Klassifikationsprozess dieselben Eigenschaften besitzen. Dazu werden die Höhe und die Breite des Bildes jeweils durch 16 geteilt, was der Anzahl der Kacheln pro Reihe und Spalte entspricht. Genau wie die Eingangsdaten für das Training haben die Bilder nun eine Größe von jeweils 24x17 Pixeln.

Da die ursprünglichen Trainingsbilder noch einigen anderen Transformationen unterzogen wurden, gilt es nun die zu klassifizierenden Bilder in das exakt gleiche Format zu bringen. Dazu werden die Kacheln jeweils durch ein Zero Padding, das oben und unten einen schwarzen Rand einfügt, quadratisch gemacht und anschließend auf die geforderte Größe 224x224 Pixel skaliert. Darauf folgt die Umwandlung des Graustufenbildes in ein RGB-Bild und die anschließende Transformation in einen Tensor. Zum Schluss wird für jeden Farbkanal der Mittelwert abgezogen und durch die Standardabweichung geteilt. Damit haben die Kacheln für die Klassifikation die exakt selben Eigenschaften, wie die ursprünglichen Trainingsdaten.

Nach der Transformation der Kacheln kann nun die eigentliche Klassifikation dieser erfolgen. Jede der Kacheln wird einzeln klassifiziert. Das Klassifizieren dieser in Batches, also ähnlich wie beim Trainingsprozess, würde zwar einige Vorteile, wie beispielsweise eine höhere Effizienz, eine höhere Geschwindigkeit und konsistentere Modellvorhersagen mit sich bringen, aber gleichzeitig wesentlich mehr Speicherplatz verbrauchen. Da der

Referenzrechner nicht die aktuellste Hardware verbaut hat, wird auf diesen Schritt also vorerst verzichtet. Es wird nun also jeder Tensor einzeln auf die CPU verschoben und durch das neuronale Netz geleitet. Dieses gibt die rohen Vorhersagewerte für jede Klasse zurück. Durch die Anwendung einer Softmax Funktion können anschließend die Wahrscheinlichkeiten berechnet werden. Diese werden extra in einer Liste abgespeichert. Unter einer Variable wird dann der Index der Klasse mit dem höchsten Score gespeichert. Um nun aber keinen Tensor Wert auszugeben, sondern das Label der Kachel, wird dieser zuerst in eine normale Zahl umgewandelt und anschließend als Index für die Klassenliste genutzt, die mit "water", "good", "mediumnoise" und "strongnoise" gefüllt ist. Da Listen in Python bei 0 beginnen, weist ein Index von 0 der zu klassifizierenden Kachel das Label „water“ zu. Ein Index von 1 bis 3 dementsprechend die darauffolgenden Klassen.

Die Zuweisung eines Labels sorgt allerdings noch nicht dafür, dass Artefakte einfach als solche erkannt werden können. Um wirklich ersichtlich zu machen, dass das Netz eine Kachel, als eine der 3 Artefakt Klassen „water, mediumnoise oder strongnoise“ klassifiziert hat, muss diese also visuell angepasst werden. Dazu wird sie nach der Übergabe aller Labels farblich gekennzeichnet. Die Farbcodierung der Klassifikation sieht wie folgt aus: Kacheln die als „water“ klassifiziert werden, erhalten eine rote Einfärbung, „mediumnoise“ Kacheln eine gelbe und „strongnoise“ Kacheln eine grüne. Die Kacheln ohne Artefakte bekommen keine farbliche Markierung. Durch diese Vorgehensweise wird gewährleistet, dass die Klassifikationsergebnisse direkt visuell überprüft werden können.

Da das Ergebnis der Klassifikation allerdings nicht in einzelnen Kacheln vorliegen soll, die einander nicht direkt zuordenbar sind, müssen diese wieder zu der ursprünglichen Form des Eingangsbildes zusammengesetzt werden. Aus den Kacheln, die reihenweise in einer Liste Zwischengespeichert sind, wird nun wieder ein vollständiges Bild.

Die Kacheln werden zur Klassifikation, wie bereits erklärt, zwar einigen Transformationen unterzogen, diese gelten allerdings ausschließlich dem Klassifikationsprozess im neuronalen Netz. Die Veränderungen, die am Bild vorgenommen werden, haben also keinen direkten Einfluss auf die eigentlichen Kacheln, sondern nur auf deren Erscheinungsbild während des Label Prozesses des ResNet50 Modells. Eingefärbt werden also nicht die größer skalierten Kacheln mit ZeroPadding, sondern die ursprünglichen Kacheln der Größe 24x17 Pixel. Andernfalls wäre eine exakte Wiederherstellung des Eingabebildes nicht möglich gewesen.

Um die Ergebnisse der Klassifikation visuell darzustellen, wird ein Fenster über ein Graphical User Interface (GUI) erzeugt, dass das klassifizierte, wieder zusammengesetzte Bild anzeigt (siehe Abbildung 12).

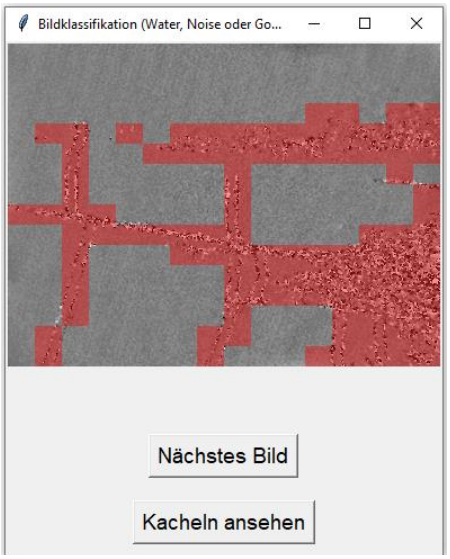

Abbildung 12: Klassifiziertes Shearographie-Bild (Eigene Darstellung in Anlehnung an SDS,2024)

Abbildung 13: Klassifiziertes Shearographie-Bild in Kachelansicht (Eigene Darstellung in Anlehnung an SDS, 2024)

Da auf dem zusammengesetzten Bild nicht immer direkt zu erkennen ist, welche farbig markierte Kachel, zu welchem Teil des Artefakts gehört, sollen die Kacheln noch einmal einzeln dargestellt werden. Durch Auswahl des Buttons „Kacheln ansehen“ wird das Bild in der Kachelansicht, also dem Stadium der Daten vor der Zusammensetzung angezeigt (siehe Abbildung 13). In dieser Ansicht ist bestens zu erkennen, in welchem Ausmaß das Artefakt in jeder einzelnen Kachel zu sehen ist.

Das Ergebnis lebt von der visuellen Darstellung, weshalb es dem Nutzer lediglich angezeigt und nicht extra in einer Datei abgespeichert wird. Eine zukünftige Erweiterung könnte allerdings darin bestehen, die Ergebnisse zusätzlich in einer CSV oder JSON-Datei abzuspeichern, oder die farbcodierten Bilder zu exportieren.

Der Klassifikationsvorgang für ein Bild (dementsprechend 256 Kacheln) dauert auf dem Referenzrechner etwa 15 Sekunden. Dieser Wert ist nicht besonders niedrig, lässt sich aber durch die etwas ältere CPU erklären. Durch die Nutzung von Batch-Processing, also der Klassifikation der Kacheln in Batches und nicht sequenziell, könnte die Verarbeitungszeit verkürzt werden. Ebenso würde der Einsatz einer GPU den Prozess stark beschleunigen.

3.5 Validierung des Modells

Um sicherzustellen, dass das trainierte Modell nicht nur die Trainingsdaten gut verarbeitet, sondern auch auf bisher unbekannten Daten verlässlich funktioniert, wird es validiert. Die Validation, die während des Trainingsprozesses stattfindet, dient als erster Eindruck, wie gut das Model auf Bilder reagiert, die es beim Training noch nicht gesehen hat. Um nun aber Bilder zu testen, auf die das Netz selbst noch gar keinen Zugriff hatte, erfolgt eine zufällige Auswahl von Daten aus dem zur Verfügung gestellten Datensatz. Um festzustellen, welche Bilder für die Validation in Frage kommen, wird nach den Kriterien vorgegangen, die bereits in Kapitel 3.1 erklärt wurden. Es wird dabei selbstverständlich darauf geachtet, dass die Bilder nicht bereits in anderen Datensätzen, die für das Training genutzt wurden vorkommen und außerdem, dass die verschiedenen Klassen jeweils ausreichend repräsentiert werden.

Um die Modelleistung objektiv bewerten zu können, werden die Genauigkeit (Accuracy), die Precision, der Recall und der F1-Score als Bewertungskriterien herangezogen. Die Accuracy gibt die richtig klassifizierten Kacheln im Verhältnis zu allen Kacheln an. Falls also beispielsweise 100 von 200 Kacheln richtig klassifiziert werden, entspräche das einer Accurency von 50%.

Die anderen drei genannten Werte geben stattdessen nicht nur wieder, wie viele Vorhersagen richtig waren, sondern wie gut das Modell bestimmte Klassen erkennt. Die Precision des Modells berechnet sich wie folgt:

$$\frac{\text{Richtig Positive (TP)}}{\text{Richtig Positive (TP)} + \text{Falsch Positive (FP)}}$$

Diese Kennzahl beantwortet die Frage, wie viele der als positiv vorhergesagten Kacheln, tatsächlich auch positiv sind (Goutte & Gaussier, 2005, S. 346–347). Für diesen Anwendungsfall also: Wenn das Modell eine Kachel als „water“ klassifiziert, wie oft liegt es dann wirklich richtig.

Der Recall beschreibt, wie viele der tatsächlich positiven Kacheln korrekt erkannt werden. Diese Kennzahl berechnet sich durch:

$$\frac{\text{Richtig Positive (TP)}}{\text{Richtig Positive (TP)} + \text{Falsch Negative (FN)}}$$

Sie prüft also, ob das Modell, Kacheln die Wasser beinhalten, auch zuverlässig als „water“ kennzeichnet (Goutte & Gaussier, 2005, S. 346–347). Ein niedriger Recall-Wert würde dementsprechend bedeuten, dass das Modell viele Kacheln fälschlicherweise als negativ klassifiziert und dementsprechend das Wasser auf ihnen nicht erkennt.

Die letzte relevante Kennzahl, der F1-Score, bildet das harmonische Mittel zwischen Precision und Recall: $2 \times \frac{\text{Precision x Recall}}{\text{Precision + Recall}}$

Ein sehr hoher F1-Score zeigt, dass eine gute Balance zwischen Precision und Recall besteht. Der F1-Score ist vor allem aussagekräftig, wenn eine große Differenz zwischen Precision und Recall vorliegt (Goutte & Gaussier, 2005, S. 346–347). Also beispielsweise für den Fall, dass das Modell zwar nahezu alle „water"-Kacheln erkennt, allerdings auch einige „good"-Kacheln fälschlicherweise als „water" klassifiziert.

Die Accurency wird für die einzelnen Epochen, wie bereits erwähnt, während des Trainings kontinuierlich ausgegeben. Dabei klassifiziert das neuronale Netz nach jeder Epoche die Kacheln aus dem Testdatensatz und vergleicht sie anschließend mit den manuell annotierten Werten. Diese Vorgehensweise gilt es nun auch außerhalb des Trainingsprozesses für Bilder umzusetzen, auf welche das Netz bisher noch keinen Zugriff hatte.

Um eine umfängliche Validierung umzusetzen, wird ein Schwellenwert von etwa fünfhundert Kacheln pro Artefakt festgelegt. Als Validierungsgrundlage dienen also beispielsweise 5 Bilder, die zusammen etwa fünfhundert Kacheln mit Wasser aufweisen. Diese 5 Bilder, die das Netz noch nie gesehen hat, werden von diesem, gemäß dem bisherigen Ablauf, klassifiziert. Die Klassifikationsergebnisse werden vom Programm in einer Liste gespeichert. Dazu werden zwei Schleifen eingesetzt, die über die verschiedenen Zeilen und Spalten in einem 16x16 Raster iterieren. Die Variable y steht dabei für die Zeilen und die Variable x für die Spalten. Die Speicherung der Labels in der Liste beginnt in der obersten Zeile ganz links (also y=1 und x=1) und endet bei Kachel 256 unten rechts (also y=16 und x=16). Dabei werden die Klassifikationsergebnisse nach jeder Zeile, also nach jeweils 16 Kacheln, in einer Liste gespeichert, welche anschließend an die endgültige Klassifikationsliste übergeben wird. Es handelt sich dabei also um eine Liste, die weitere 16 Listen (also die Zeilen) beinhaltet. Dann kann die Datei, als JSON-Datei auf den Rechner exportiert werden. Die zugewiesenen Labels befinden sich nun also in einer extra Datei, um sie später mit weiteren Daten vergleichen zu können.

Da die vom neuronalen Netz klassifizierten Bilder nun mit manuell klassifizierten verglichen werden sollen, müssen dieselben Bilder jetzt von Hand gelabelt werden. Um diese Aufgabe umzusetzen, wird ein weiteres Programm erstellt, welches ziemliche Ähnlichkeiten zu dem Programm hat, das zum Annotieren genutzt wurde. Nachdem ein Bild ausgewählt wird, besteht die Möglichkeit, manuell Kacheln auf diesem hervorheben. Falls nun also einige Kacheln auf dem Bild beispielsweise Wasser aufzeigen, können diese per Mausklick

ausgewählt werden. Nachdem alle kritischen Kacheln ausgewählt sind, werden diese, genau wie die Ergebnisse vom ResNet50-Modell, in einer JSON-Datei abgespeichert. Es wird dabei darauf geachtet, dass die Datei denselben Dateinamen zugewiesen bekommt, wie das bearbeitete Bild.

Bevor die Ergebnisse aber anhand der erstellten Listen vergleichen werden, kann vorerst eine visuelle Überprüfung der Ergebnisse durchgeführt werden. Da die Kacheln, die vom ResNet50 Netz klassifiziert werden, vom Programm eingefärbt werden, ist gut erkenntlich, welche Kacheln wie klassifiziert wurden (siehe Abbildung 14). Die roten Rahmen bei der manuellen Klassifizierung (siehe Abbildung 15), können nun für den direkten Vergleich herangezogen werden.

Abbildung 14: Von ResNet50 klassifiziertes Shearographie-Bild (Eigene Darstellung in Anlehnung an SDS, 2024)

Abbildung 15: Manuell klassifiziertes Shearographie-Bild (Eigene Darstellung in Anlehnung an SDS, 2024)

Wie beim beispielhaften visuellen Vergleich der beiden Abbildung oben ersichtlich wird, sind die Kacheln, die farblich markiert wurden, bei beiden Bildern nahezu identisch.

Um nun aber wirklich aussagekräftige Kennzahlen zu generieren, müssen die bereits angesprochenen Listen miteinander verglichen werden. Da der händische Vergleich zwischen zwei Listen mit jeweils 256 Einträgen allerdings viel zu zeitaufwändig und fehleranfällig wäre, erfolgt dieser Schritt durch die Erstellung eines weiteren Python-Programms. Durch dieses können zwei JSON-Dateien ausgewählt und deren Einträge miteinander verglichen werden. Das Ergebnis dieses Vergleichs, das der Klassifizierungsgenauigkeit des Modells entspricht, wird automatisch in einem Ordner, unter dem Dateinamen des klassifizierten Bildes, als Textdatei abgespeichert und kann aus dieser dann abgelesen werden. Bei dem Vergleich der beiden Klassifizierungsergebnisse oben, konnte eine Genauigkeit von 98,05%, eine Precision von 97%, ein Recall von 97,98% und ein F1-Score von 97,49% erreicht werden. Die Zahlen können folgendermaßen

interpretiert werden: Eine Genauigkeit von 98,05% sagt aus, dass die meisten Klassifikationen, also in diesem Fall 251 von 256 Kacheln, bei beiden Modellen identisch sind. Durch eine Precision von 97% kann festgestellt werden, dass das Model wenige „good“ Kacheln fälschlicherweise als „Water“ klassifiziert hat. Ein Recall von 97,98 % hingegen zeigt, dass wenige Kacheln, die Wasser beinhalten, fälschlicherweise als in Ordnung angenommen werden. Der F1-Score von 97,49% ist wenig überraschend, da die Precision und der Recall bereits gut ausbalanciert sind und das harmonische Mittel zwischen ihnen dementsprechend ebenfalls hoch ausfällt.

Zur Validierung des Modells für die anderen Klassen Mediumnoise und Strongnoise muss lediglich das Label Programm auf die jeweilige Klasse umgestellt werden. Ansonsten ist die Vorgehensweise identisch.

Um den Durchschnittswert der jeweiligen Kennwerte für alle verwendeten Bilder zu ermitteln, wird ein weiteres einfaches Programm erstellt, dass die verschiedenen Kennwerte zeilenweise pro Dokument miteinander addiert und anschließend durch die Anzahl der Bilder teilt. Dadurch erhält man aussagekräftige Zahlen zur Bewertung der Modellleistung.

Für die Art der Bilderkennung, die diese Arbeit erfordert, gibt es online keine vergleichbaren Modelle, weshalb die Klassifikationsergebnisse auch ausschließlich mit der manuellen Klassifikation verglichen werden können. Es ist nicht zu leugnen, dass dabei selbstverständlich die subjektive Meinung desjenigen, der die Bilder bewertet, hineinspielt. Es gibt bisher allerdings keine Definition, wie die Artefakte genau aussehen und welche Strukturen oder Pixelabweichungen im Bild noch als solche zu verstehen sind. Deswegen kann die manuell erstellte Klassifikation in diesem Fall als gewünschtes Resultat angesehen werden, auf welches hingearbeitet werden kann. Im besten Fall stimmt das Ergebnis des neuronalen Netzes schlussendlich zu annähernd 100% mit diesem überein. Falls Abweichungen auftreten, kann analysiert werden, woran dies liegt, um das neuronale Netz gezielt weiterzuentwickeln.

Im Laufe dieses Abschnitts wurde nun der Aufbau der Validierungsmethode ausführlich erläutert. Die Ergebnisse, die das trainierte Modell erzielen kann, werden in Kapitel 4.3.3 genauer analysiert.

4 Implementierung und Experimentelle Ergebnisse

4.1 Beschreibung der Software-Architektur und des Codes

Um das Deep Learning gestützte Klassifikationsprogramm umzusetzen, wird Python als Programmiersprache gewählt. Sie bietet zahlreiche Vorteile für verschiedenste Deep Learning Anwendungen. Unter anderem besitzt Python etliche leistungsstarke Bibliotheken, wie beispielsweise PyTorch, Tensorflow und Keras, die extra für das Deep Learning entwickelt wurden. Sie ermöglichen eine effiziente Modellentwicklung. Des Weiteren ist diese Programmiersprache wesentlich benutzerfreundlicher als beispielsweise die beiden Alternativen Java und C++. Durch die klare Syntax wird dem Benutzer die Entwicklung und die Wartung des Codes deutlich erleichtert. Es konvertiert den Quellcode vorerst in eine Zwischensprache und diese wird anschließend in Maschinensprache übersetzt. Die verwendete Anordnung von Wörtern ist deshalb greifbarer und einfacher zu lernen (B. Rathod, & Gadhavi, 2013, S. 83–84).

Abgesehen davon verfügt Python durch seine große Community und die umfangreichen Ressourcen über eine Vielzahl von Tutorials und Foren, die zur Lösung von gängigen Problemen und der Integration von bereits bewährten Methoden beitragen. Durch CUDA bietet Python außerdem die Möglichkeit Berechnungen auf der GPU auszuführen (falls eine existiert) und kann dadurch Deep Learning Modelle erheblich beschleunigen. Aufgrund dieser Vorteile kommt Python für die Umsetzung des Klassifikationsprogramms zum Einsatz.

Das erstellte Programm folgt einer modularen Architektur, die aus mehreren Hauptkomponenten besteht. Die Umsetzung erfolgt über Python unter Zuhilfenahme verschiedener Bibliotheken, auf die später genauer eingegangen wird. Die gesamte Interaktion mit dem User erfolgt über die Konsole, oder über das Graphical User Interface. Der Code zur Klassifizierung umfasst ein Main-Skript und drei weitere Skripte:

1. Main-Skript: Hat die Aufgabe den Workflow zu steuern
2. Import Images Klasse: Bilder importieren und Transformieren
3. Train Klasse: Trainieren des ResNet50-Modells
4. Result Picture Klasse: Klassifikation durchführen und Ergebnisse visualisieren

Main-Skript

Das Hauptskript hat die Aufgabe, den Workflow des Programms zu steuern. Zur Nutzung des Codes müssen einige Module bzw. Klassen importiert werden. Die meisten Module führen hauptsächlich Aufgaben aus, die nichts zum eigentlichen Deep Learning Modell beitragen, deshalb wird auf eine nähere Ausführung dieser verzichtet. Sie werden beispielsweise zum Öffnen eines PopUp-Fensters oder des Tensorboards verwendet. Die Module Optuna und Pytorch werden ausschließlich zur Optimierung der Hyperparameter importiert. Am wichtigsten sind selbstverständlich die drei Klassen Import_Images, Train und Resultpicture, die jeweils aus den gleichnamigen Skripten importiert werden. Sie enthalten die Funktionen rund um das Deep Learning Modell.

Das Skript besteht aus sechs verschiedenen Funktionen. Beim Ausführen wird jedoch nur die main-Funktion durchlaufen, die wiederum die anderen fünf Funktionen aufrufen kann. Durch sie lässt sich das Programm steuern. Direkt nach der Initialisierung der Funktion **main** wird die Funktion **start_Tensorboard** aufgerufen. Diese startet TensorBoard zur Überprüfung der Trainingserfolge und gibt anschließend den Link dazu aus. Danach wird der User gefragt, welche Aktion als nächstes ausgeführt werden soll, und kann aus sieben verschiedenen Möglichkeiten wählen (siehe Abbildung 16).

```
print("1: Bilder importieren und Batches erstellen")
print("2: Modell trainieren")
print("3: Modell speichern")
print("4: Modell laden")
print("5: Klassifikation und Interaktive Bildanzeige")
print("6: Hyperparameter automatisch optimieren")
print("7: Programm beenden")
```

Abbildung 16: Codeausschnitt aus dem Main-Skript (eigene Darstellung)

- Durch Auswahl der ersten Option, wird erstens die Funktion **select_folder** aufgerufen welche den Datei-Explorer öffnet und den User einen Ordner mit den zu verwendenden Bildern auswählen lässt und zweitens die Klasse **Import_Images**, welche im Laufe dieses Kapitels erklärt wird. Außerdem erfolgt hier die Aufteilung der Inputdaten in Training und Test.
- Um das Modell zu trainieren, muss Option zwei ausgewählt werden. Nach der manuellen Eingabe zweier Hyperparameter (Anzahl der Epochen und Lernrate) durch den User, werden die vorher importierten Trainingsdaten übergeben und die Klasse **Train** wird aufgerufen. Auch dieser wird im folgenden Teil ein extra Abschnitt gewidmet.

- Optionen drei und vier sind für die Speicherung und das Laden eines trainierten Modells zuständig. Wenn Option drei gewählt wird, wird das trainierte Modell unter dem Dateinamen „water_noise_model_YYYYMMDD_HHMMSS.pth“ abgespeichert. Dieser könnte also beispielsweise so aussehen: „water_noise_model_20250203_024949.pth“. Option vier wiederum, lässt den User ein bereits gespeichertes Modell auswählen, welches anschließend in das Programm geladen wird. Dazu wird die Funktion **select_file** aufgerufen, welche den User über den Datei-Explorer zu der Auswahl des Modells auffordert.
- Option fünf beinhaltet die Klassifikation von neuen Bildern. Es erfolgt abermals der Aufruf der Funktion **select_folder**, um den Ordner, in dem sich die zu klassifizierenden Bilder befinden, auszuwählen. Anschließend wird dieser an die Klasse **ResultPicture** übergeben, deren Funktionsweise weiter unten erklärt wird.
- Durch Option sechs, können die optimalen Hyperparameter für das Training des Modells ermittelt werden. Es wird erneut die Funktion **select_folder** aufgerufen, um einen Testdatensatz auszuwählen, anhand welchem das Modell die Hyperparameter ermitteln kann. Zu diesen gehören die Lernrate, die Anzahl der Epochen, die Größe der Batches und der Name des Optimierers (Adam oder SGD). Anschließend wird die Funktion **objective** aufgerufen, welche über Optuna zehn verschiedene Hyperparameter-Kombinationen testet. Der anschließende Aufruf der Funktion **evaluate** sorgt für die Bewertung der Performance des Modells auf den Testdaten. Die optimalen Parameter werden danach in der Konsole ausgegeben.
- Option sieben beendet die Auswahl und damit das gesamte Programm.

Klasse Import_Images:

Diese Klasse hat die Aufgabe, die Trainingsdaten so vorzubereiten, dass sie den Eingabeanforderungen des ResNet50-Modells entsprechen. Dafür werden erneut einige Module importiert. Einige dieser Imports erledigen nebensächliche Aufgaben wie beispielsweise die Visualisierung von Bildmaterial, das Generieren einer zufälligen Auswahl von Daten oder das Ermöglichen von Datei- und Ordneroperationen. Die anderen Module tragen direkt zum Deep Learning Modell bei. Der Import von PyTorch, wird zur Berechnung der Tensoren verwendet. Das Modul „transforms“ sorgt dafür, dass die Bilder transformiert werden können. PIL (Pillow) bietet die Möglichkeit, Bilder zu manipulieren, beispielsweise durch Spiegeln oder Invertieren. Zudem ermöglicht es dem Nutzer, Bilder zu öffnen und zu verarbeiten.

Die Klasse besteht aus drei verschiedenen Methoden und ruft während der Ausführung eine weitere Klasse „ZeroPaddingToSquare“ auf. Durch die **__init__** Methode wird die Klasse initialisiert. Nach der Eingabe der gewünschten Batch-Größe durch den User, werden dann zwei Bilder, durch den Aufruf der Methode **show_example_image** visuell dargestellt. Die Auswahl des Bildes erfolgt zufällig. Das erste zeigt das ursprüngliche Bild, genau wie es hochgeladen worden ist, das Zweite zeigt dieses, wie es nach der Transformation aussehen würde.

Anschließend wird die Methode **load_data** aufgerufen, welche die Reihenfolge der Daten im Eingabeordner vermischt und diese daraufhin mithilfe einer Schleife einzeln in das Programm lädt. Darauf folgt die Transformierung des Bildes. Wie bereits im Kapitel 3.2 ausführlich erklärt wurde, wird das Bild oben und unten mit einem ZeroPadding auf eine quadratische Form erweitert. Danach wird es auf 224x224 Pixel hochskaliert, in RGB umgewandelt und in einen Tensor transformiert. Nach der Normalisierung ist der Transformationsprozess abgeschlossen und das Bild wird in Form seiner Tensorwerte dem aktuellen Batch zugewiesen. Die Targets, die später dafür sorgen, dass das Modell erkennen kann, welcher Klasse ein Bild angehört, werden basierend auf den Dateinamen erstellt. Falls im Dateinamen folglich ein „water“ vorkommt, wie es bei den manuell annotierten Bildern der Fall ist, wird das Target Water auf 1 gesetzt. Da die anderen Klassen nicht im Dateinamen enthalten sind, behalten diese den Targetwert 0. Die vier Werte werden anschließend an eine Liste übergeben, die in einen Tensor verwandelt wird. Der Tensor würde für diesen Fall dementsprechend so aussehen: (1, 0, 0, 0). Dadurch weiß das Programm, dass es dieses Bild als Wasser interpretieren soll.

Sobald der aktuelle Batch der vorher festgelegten Batchgröße entspricht, wird er den Trainings-Batches hinzugefügt. Nachdem alle Bilder als Tensoren mit zugehörigen Targets gespeichert sind, werden sie an das Main-Skript übergeben.

Klasse Train:

Über die Klasse Train erfolgt das Training des ResNet50-Modells. Die importierten Module werden in dieser Klasse fast alle im direkten Zusammenhang mit dem Deep-Learning-Modell verwendet und werden aus der PyTorch-Bibliothek importiert, welche das Arbeiten mit Tensoren und neuronalen Netzen erlaubt und zusätzlich den Zugriff auf eine GPU ermöglichen kann. Das Modul „torch.nn“ wird dabei für das beschneiden der Gradienten genutzt. Das Modul „optim“ zum Implementieren der Optimierer Adam und SGD. Um Zugriff auf das vortrainierte ResNet50-Modell zu erhalten, wird das Modul „models“

importiert. Für die Anpassung der Lernrate während des Trainingsprozesses, wird „StepLR“ importiert, über welches das Programm Zugriff auf den gleichnamigen Lernraten-Scheduler erhält. Der „SummaryWriter“ kann die Ergebnisse des Trainingsprozesses im TensorBoard loggen.

Die Klasse besteht aus fünf Methoden, von denen als erstes die **__init__** Methode aufgerufen wird. Vorerst erfolgt die Übergabe einiger Daten und Werte, wie beispielsweise den Trainings- und Testbatches, der festgelegten Lernrate und der Anzahl der Epochen. Anschließend wird das vortrainierte ResNet50 Modell initialisiert, die Klassifikationsschicht wird angepasst und die Verlustfunktion und der Optimierer werden festgelegt. Abschließend wird als Lernraten Scheduler StepLR ausgewählt, welcher die Lernrate in geregelten Abständen reduziert.

Daraufhin wird über das Main-Skript die Methode **train** aufgerufen. Sie ist für die Durchführung des Trainings zuständig. Dazu wird eine Schleife verwendet, die über die Anzahl der Epochen iteriert. Eine weitere Schleife, die sich innerhalb dieser befindet, iteriert nun über die Trainings-Batches, welche die Bilder und deren zugewiesene Targets jeweils als Datenpaare an das Programm übergeben.

Darauf folgt der eigentliche Trainingsprozess, also der Teil des Codes, in dem das Modell wirklich lernt. Dieser beginnt mit dem Vorwärtsdurchlauf, in dem das Modell die Eingabebilder nimmt und Vorhersagen über diese ausgibt. Der Verlust wird anschließend mit der Verlustfunktion berechnet. Anschließend kommt es zum Rückwärtsdurchlauf, bei dem die Gradienten berechnet werden, die bestimmen, wie stark die Gewichte des Modells angepasst werden müssen. Zum Schluss werden die Gewichte des Modells, basierend auf den berechneten Gradienten aktualisiert. Das ist der Schritt, in dem das Modell wirklich lernt. Für den reibungslosen Ablauf des Trainings werden die Gradienten für jedes Batch zurückgesetzt und nach dem Rückwärtsdurchlauf beschnitten, was ein „Explodieren“ dieser verhindern soll.

Nachdem das Modell mit einem Batch trainiert worden ist, werden einige Kennwerte, wie beispielsweise der Verlust die Genauigkeit und die aktuelle Lernrate berechnet, ausgegeben und zusätzlich im Tensorboard geloggt. Um Overfitting zu vermeiden, vergleicht das Programm den durchschnittlichen Verlust des aktuellen Batchs, mit dem bisher niedrigsten Verlust innerhalb des Trainingsprozesses. Falls der aktuelle Verlust niedriger ist, passiert nichts weiter, aber für den Fall, dass dieser fünf Mal in Folge größer ist, wird das Early Stopping aktiviert und der Trainingsprozess wird abgebrohen.

Am Ende jeder Epoche wir auf die Methode **validate** zugegriffen, welche die Leistung des Modells auf einem Validierungsdatensatz bewertet. Dabei wird durch eine Schleife, nach Abschluss der aktuellen Trainingsepoche, über die Bilder und deren Targets iteriert. Jedes Bild wird vom neuronalen Netz klassifiziert und anschließend mit dem Wert des Targets verglichen. Falls die Klassifikation des Modells mit dem Target übereinstimmt, wird eine Variable um eins erhöht. Nachdem alle Bilder bewertet worden sind, wird die Anzahl der korrekt klassifizierten Bilder durch die Anzahl aller klassifizierten Bilder geteilt. Daraus ergibt sich dann die Genauigkeit des Modells nach der aktuellen Epoche. Die Ergebnisse werden dem User in der Konsole ausgegeben und zusätzlich im Tensorboard geloggt. Danach gilt der Trainingsprozess als beendet, und das Programm kehrt wieder in das Haupt-Skript zurück.

Die beiden Methoden **save_model** und **load_model** befinden sich ebenfalls in der Train Klasse und können jeweils über das Main-Skript aufgerufen werden**. save_model** speichert das aktuell fertig trainierte Modell unter einem festgelegten Dateipfad, während **load_model** dieses, oder ein früher trainiertes Modell über den Datei Explorer ins Programm laden kann.

Klasse ResultPicture:

Die ResultPicture Klasse beinhaltet den Code für die Klassifizierung der Shearographie-Bilder. Von den importierten Modulen, sind nur wenige direkt an der Klassifizierungsaufgabe beteiligt. Dazu gehören das bereits bekannte „torch“ Modul, das etliche nützliche Tools für Deep Learning Modelle bereitstellt, das ebenfalls bereits bekannte „transforms“, welches die nötigen Tools zur Vorverarbeitung der Bilder liefert und das Modul „Image“, durch das direkte Veränderungen am Originalbild vorgenommen werden können. Die restlichen Module erfüllen nur nebensächliche Aufgaben. Aus dem Import_Images-Skript wird außerdem die Klasse ZeroPaddingToSquare importiert, welche auch hier denselben Zweck erfüllen soll.

Die Klasse besteht aus sieben verschiedenen Methoden, von denen die **__init__** Methode wieder als erstes ausgeführt wird. Sie übergibt ein trainiertes Modell an die CPU (oder falls vorhanden an die GPU) des Rechners und setzt dieses in den Evaluierungsmodus. Daraufhin werden alle Bilddateien, die eine der gängigen Dateiendungen „png, jpg, oder jpeg“ besitzen, aus dem übergebenen Ordner in das Programm geladen. Anschließend wird das Graphical User Interface (GUI) initialisiert. Das GUI ist eine visuelle Schnittstelle, die es dem User erlaubt, mit dem Programm zu interagieren, ohne die Kommandozeile zu nutzen. Durch den Aufruf der Methode **setup_ui** werden verschiedene Elemente des GUI erstellt. Dazu gehören

beispielsweise der Bereich, in dem das Bild angezeigt werden soll und die Buttons, durch welche der User die Möglichkeit hat, mit dem Ablauf des Codes zu interagieren.

Durch den folgenden Aufruf der Methode **show_random_image** wählt das Programm zufällig ein Bild aus dem übergebenen Ordner und klassifiziert es für den User. Dafür wird wiederum die Methode **process_image** herangezogen. Diese Teilt das geladene Bild in 256 gleich große Teile, indem sie Die Höhe und die Breite des Bildes durch 16 teilt und anschließend mit jeweils einer Schleife über die Zeilen und die Spalten im Bild iteriert. Zum Klassifizieren der Kacheln wird die Methode **classify_tile** aufgerufen, welche vorerst jede Kachel den exakt selben Transformationen unterzieht, wie bereits bei der Klasse Import_Images, um dieselben Ausgangsbedingungen für das Training genauso wie für die Klassifikation zu schaffen. Dafür wurde aus dem Import_Images-Skript die Klasse ZeroPaddingToSquare importiert und auch sonst derselbe Code verwendet. Nach der Transformation jede Kachel in das Modell geladen und von diesem klassifiziert, indem die Wahrscheinlichkeiten für jede Klasse ausgegeben und durch die max-Funktion aus Pytorch, die wahrscheinlichste Klasse ausgewählt wird. Die vorhergesagte Klasse wird an die **process_image** Methode zurückgegeben. Danach werden die Kacheln zeilenweise in einer Liste gespeichert. Wie der Code mögliche Fehlklassifikationen korrigiert, wird in Kapitel 4.3.2 ausführlich erläutert.

Nachdem alle Kacheln klassifiziert sind, wird zum Zusammensetzen des ursprünglichen Bildes wieder über zwei Schleifen iteriert, die einmal die Zeilen und einmal die Spalten im 16x16 Raster darstellen. Die ursprüngliche Position jeder Kachel wird bestimmt und die zugehörige Klassifikation wird in der Variable „label“ zwischengespeichert. Über das Label wird daraufhin entschieden, ob die Kachel visuell verändert werden soll. Das ist der Fall, wenn das Label den Klassen „water, mediumnoise oder strongnoise“ entspricht. Wenn das zutrifft, wird die Methode **add_color_overlay** aufgerufen, welche die Kachel je nach Label rot, gelb, oder grün einfärbt. Dafür wird ein RGBA-Kanal genutzt, der für Rot, Grün, Blau und Alpha (Transparenz) steht. Damit die ursprünglichen Inhalte der Kachel nicht komplett verschwinden, wird mit einer Transparenz von etwa 40% gearbeitet. Bei 100%, was im RGBA-Kanal der Zahl 255 entspricht, wäre der Inhalt der Kachel nicht mehr zu sehen. Nachdem alle 256 Kachel klassifiziert und falls nötig eingefärbt sind, werden sie nach und nach in das ursprüngliche Bild eingefügt, bis dieses wieder vollständig ist und die Artefakte, die auf diesem zu sehen sind, farblich hervorhebt.

Dieser gesamte Prozess läuft ab, während die Methode **show_random_image** das Bild für den User anzeigen will. Sobald sie das zusammengesetzte Bild von der **process_image** Methode zurückbekommt, wird dieses im GUI angezeigt (siehe Abbildung 17).

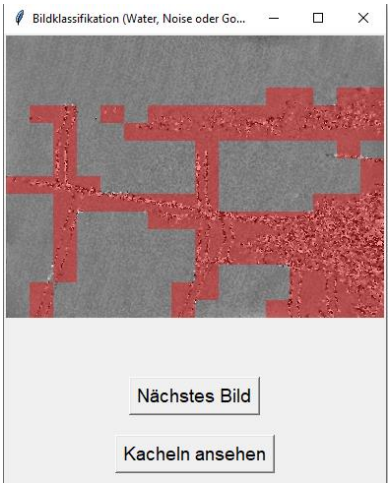

Abbildung 17: Ausgabe des Graphical User Interface nach dem Klassifikationsvorgang (Eigene Darstellung in Anlehnung an SDS, 2024)

Durch ein Drücken auf den Knopf „Nächstes Bild“, wird die **show_random_image** Methode erneut aufgerufen und es wird ein weiteres Bild klassifiziert und angezeigt.

Wenn wiederum auf „Kacheln ansehen“ geklickt wird, wird die Methode **view-tiles** aufgerufen. Sie sorgt durch den Einsatz von zwei Schleifen für die richtige Reihenfolge der Kacheln und fügt jeweils einen horizontalen und vertikalen Abstand von 5 Pixeln zur nächsten Kachel ein. Nachdem alle Kacheln angeordnet wurden, geht ein weiteres Fenster auf, welches die Kacheln einzeln darstellt.

Der Code zur Klassifizierung neuer Bilder läuft so lange weiter, bis der User das Graphical User Interface durch das Schließen des Fensters beendet. Dadurch kehrt man wieder in das Main-Skript zurück.

4.2 Trainingsprozess und Optimierungsstrategien

Nachdem in Kapitel 3.3 der Aufbau und der prinzipielle Ablauf des Trainings vom neuronalen Netz erklärt und in Kapitel 4.1 auf die Umsetzung im Code eingegangen wurde, folgen nun die Anwendung und die Optimierung des Trainingsprozesses auf echten Bildern.

Um zu testen, wie das neuronale Netz auf die in 256 Kacheln aufgeteilten Shearographie-Bilder reagiert, erfolgen die ersten Tests mit sehr wenig Bildmaterial und vorerst ausschließlich mit den Klassen „Good“ und „Water“. Dazu wird die Klassifizierungsschicht des ResNet50-Netzes auf zwei statt den eigentlich vier Klassen reduziert. Da bekannt ist, dass neuronale Netze, für eine solides Ergebnis, im Normalfall eine große Anzahl an Trainingsdaten erfordern, sahen die Ergebnisse dieser Testdurchläufe auch nicht besonders vielversprechend aus (siehe Abbildung 18). Wie zu erkennen ist, griff bereits nach der zweiten Epoche die Early-Stopping-Funktion, die nach der fünften Epoche ohne Verbesserung das Training beendete. Dieser Trainingsdurchlauf wurde mit jeweils hundert „Good“ und „Water“ Bildern durchgeführt. Der Trainingsprozess wurde bei so wenigen Bildern in etwa 5 Minuten abgeschlossen. Die höchste Genauigkeit, die bei der Validierung erreicht werden konnte, betrug bei 200 Eingabebildern ca. 76%. Dieser Wert ist wesentlich zu gering und würde unter der Berücksichtigung der zwei weiteren Klassen noch wesentlich schlechter ausfallen. Aus diesem Grund fiel die Entscheidung, den Trainingsdatensatz erheblich zu erweitern.

```
Epoche 5/10 abgeschlossen.
Verlust: 1.9385, Genauigkeit: 81.25%, Lernrate: 0.000100
Keine Verbesserung: 4/5 Epochen ohne Verbesserung.
Validierung gestartet...
Validierung: Verlust: 2.0763, Genauigkeit: 76.47%
Epoche 6/10 abgeschlossen.
Verlust: 1.0256, Genauigkeit: 81.82%, Lernrate: 0.000100
Keine Verbesserung: 5/5 Epochen ohne Verbesserung.
Early Stopping aktiviert. Training wird beendet.
Training abgeschlossen.
```

Abbildung 18: Konsolenausgabe während des Trainingsprozesses (eigene Darstellung)

Der nächste Test wurde also mit 2500 Bildern durchgeführt, allerdings immer noch begrenzt auf die zwei Klassen „Good“ und „Water“. Das Training dauerte in diesem Fall natürlich wesentlich länger, dafür waren aber auch die Ergebnisse entscheidend besser. Die kompletten 10 Epochen wurden ohne ein Early-Stopping durchlaufen. Die Validierungsgenauigkeit stieg von Epoche zu Epoche und betrug nach der letzten Epoche etwa 86%. Die Erhöhung der Anzahl der Trainingsbilder sorgte also bereits für ein deutlich besseres Ergebnis.

Da das ResNet50-Netz aber nicht nur zwei Klassen vorhersagen muss, sondern vier, gilt es, die anderen beiden Klassen ebenfalls in den Trainingsprozess mit aufzunehmen. Dazu wird die letzte Schicht des neuronalen Netzes wieder von zwei auf vier Neuronen erhöht. Die Trainingsdaten werden um die zwei Klassen „Mediumnoise“ und „Strongnoise“ erweitert.

Zur Erstellung der Trainingsdaten von den beiden Klassen muss auf den Originalbildern, wie bereits in Kapitel 3.1 kurz angesprochen, der Hintergrund des Bildes angepasst werden. Die Seitenwand des Reifens macht zwar einen Großteil des Bildes aus, aber im Hintergrund sind verschiedene Störfaktoren zu sehen, die einen negativen Einfluss auf die Funktionsweise des neuronalen Netzes hätten. Deshalb wird der Hintergrund mit schwarzer Farbe überdeckt. Genaueres zur Vorgehensweise findet sich in Kapitel 4.3.2. Es existierten nun also Bilddaten zu jeder der vier Klassen.

Um zu überprüfen, wie das Modell auf die zwei zusätzlichen Klassen reagiert, wird ein erster Test mit etwa zweihundert Bildern pro Klasse durchgeführt. Dafür werden eine Batch-Größe von 16, eine Lernrate von 0,001 und eine Epochenanzahl von 15 eingestellt. Wie zu erwarten war, sahen die Ergebnisse allerdings noch nicht besonders vielversprechend aus. Das Modell erreichte eine Validierungsgenauigkeit von etwa 78 %, was für den Anspruch dieser Arbeit jedoch noch kein zufriedenstellendes Ergebnis darstellt. Zur Verbesserung dieses Wertes kommen zwei Optionen in Frage: erstens die Erweiterung des Eingabedatensatzes und zweitens die Optimierung der Hyperparameter.

Zunächst wird die erste Option umgesetzt: die Erweiterung des Eingabedatensatzes. Um dem ResNet50-Modell genügend Bilder für den Trainingsprozess zur Verfügung zu stellen, wird für jede Klasse ein Richtwert von 1.500 Bildern festgelegt. Beim Ausbalancieren der Anzahl der Bilder pro Klasse ist jedoch schnell aufgefallen, dass die Klasse ‚Mediumnoise‘ wesentlich unterrepräsentiert ist, da dieses Artefakt im ursprünglichen Datensatz nur selten vorkommt. Um die mangelnde Balance auszugleichen, wird für die Klasse „Mediumnoise“ eine Datenaugmentation implementiert, die jedes dieser Bilder um 90°, 180° und 270° dreht und in der jeweiligen Position abspeichert. Dadurch kann die ursprüngliche Anzahl der Bilder von 375 auf die geforderten 1.500 angehoben werden.

Nach der Erstellung des ausgewogenen Trainingsdatensatzes, wird erneut getestet, wie das neuronale Netz auf diese Veränderung reagiert. Die Hyperparameter werden für diesen Durchlauf noch nicht angepasst. Das Modell durchlief alle 15 Epochen, ohne Early-Stopping und generierte eine Validierungsgenauigkeit von etwa 85%.

Um das bereits deutlich bessere Ergebnis weiter zu optimieren, gilt es nun, die Parameter des Trainingsprozesses anzupassen, auf die der Nutzer einen Einfluss hat. Dazu zählen die Lernrate, die Anzahl der Trainingsepochen, die Batch-Größe und der Optimierer. Diese Parameter werden im maschinellen Lernen als Hyperparameter bezeichnet. Im Gegensatz zu normalen Parametern, wie beispielsweise den Gewichten eines neuronalen Netzes, werden die Hyperparameter vor dem Training festgelegt und beeinflussen direkt, wie das Modell trainiert wird. Sie werden also nicht durch das neuronale Netz während des Trainings angepasst, sondern einmalig durch ein Verfahren wie Hyperparameter-Optimierung ermittelt und dem Modell übergeben.

Um die angesprochene Hyperparameter-Optimierung durchzuführen, kommt Optuna zum Einsatz. Das ist eine Python Bibliothek, die verschiedene Kombinationen von Parameterwerten ausprobiert, um am Ende die optimalen Parameter für das neuronale Netz auszugeben. Für diese Arbeit werden zehn verschiedene Kombinationen von Optuna getestet. Dabei wurden für die Lernrate Werte von 0,01 bis 0,00001, für die Batch-Größe die Werte 16, 32 und 64, für die Anzahl der Epochen Werte zwischen 5 und 20 und für den Optimierer Adam und SGD getestet. Der Optimierungsprozess läuft völlig automatisch ab und gibt nach der Testung aller zehn Kombinationen, die bestmöglichen Werte für jeden Parameter aus (Siehe Abbildung 19). Für das optimale Training des ResNet50-Modells wird nun also die Lernrate auf 0,0001 reduziert und die Anzahl der Trainingsepochen auf 20 angehoben. Die anderen beiden Werte können so übernommen werden, wie sie sind.

```
✅ Beste gefundene Hyperparameter:
Beste Lernrate: 0.0001712465549447676
Beste Anzahl an Epochen: 20
Beste Batch-Größe: 16
Bester Optimizer: Adam
```

Abbildung 19: Konsolenausgabe von Optuna (eigene Darstellung)

Das Training dauerte über die 20 Epochen und mit so vielen Bilder auf dem Referenzrechner etwa 7 Stunden. Die Ergebnisse, die das Modell erzielen konnte, waren wesentlich besser als vor der Hyperparameter-Optimierung (siehe Abbildung 20). Da die Genauigkeit des Modells auf den Trainingsbatches nach 15 Epochen bereits sehr hoch war (99,47%), hat sie sich in den letzten 5 Epochen nur noch um 0,04%

```
Epoche 20/20 abgeschlossen.
Verlust: 0.0212, Genauigkeit: 99.51%, Lernrate: 0.000000
Verbesserung erzielt: Verlust: 0.0212
Validierung gestartet...
Validierung: Verlust: 0.7568, Genauigkeit: 89.86%
Training abgeschlossen.
```

Abbildung 20: Konsolenausgabe des Trainings-Skriptes (eigene Darstellung)

verbessert. Die Lernrate wurde aufgrund der so hohen Genauigkeit so stark verringert, dass sie zahlenmäßig gar nicht mehr vom Programm ausgegeben wurde. Normalerweise sollte für diesen Fall die Early-Stopping-Funktion greifen, aber durch die minimale Verbesserung des Verlusts in den letzten Epochen, wurde diese nicht aufgerufen.

Die Genauigkeit des Modells auf den Validierungsdaten blieb über fünf Epochen hinweg konstant. Da sie in diesem Zeitraum aber nicht sank und die Differenz zur Trainingsgenauigkeit gering war, gab es keinen eindeutigen Hinweis auf Overfitting.

Der genaue Verlauf der Kurven der einzelnen Kennzahlen wird durch das Tensorboard während des gesamten Trainingsprozesses aufgezeichnet und kann in den zugehörigen Schaubildern abgelesen werden (siehe Abbildung 21).

An den beiden Kurven, die die Accurency darstellen (rechte Seite) ist gut zu erkennen, dass das Modell das Konvergenzkriterium nach den 20 Epochen erfüllt hat und ein Fortführen des Trainingsprozesses keinen Mehrwert generiert hätte. Die beiden Schaubilder auf der linken Seite zeigen, wie die Lernrate angepasst wurde und wie sich der Trainingsverlust im Laufe des Trainings verringert hat.

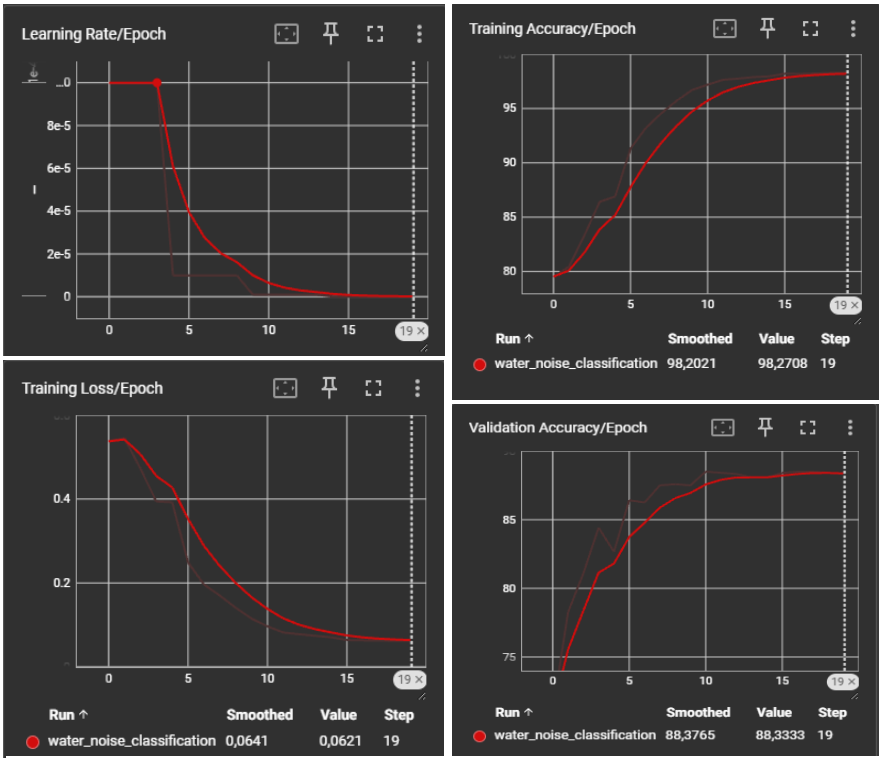

Abbildung 21: Ausgabe des Tensorboards nach Abschluss des Trainingsprozesses (eigene Darstellung)

Um zu analysieren, wie das trainierte Modell auf bisher unbekannte Bilder reagiert, wird im folgenden Kapitel eine Analyse der Modellperformance durchgeführt.

4.3 Modellperformance und Optimierungspotenziale

In diesem Kapitel wird die Effizienz des Deep-Learning-Modells, insbesondere im Hinblick auf die Klassifizierungsergebnisse analysiert. Ziel der Analyse ist es, mögliche Schwachstellen des Modells zu identifizieren, und zu bewerten, wie genau es auf neue, bisher unbekannte Bilder reagiert. Dies dient dazu, Optimierungspotenziale zu identifizieren und die Gesamtleistung zu bewerten. Im Mittelpunkt steht die Frage, ob die Leistung des Modells für die vorgesehene Anwendung ausreichend ist.

Um die Leistung des Modells zu bewerten und zu validieren, kommen die bereits genannten Metriken Genauigkeit, Präzision, Recall und F1-Score zum Einsatz. Diese Metriken repräsentieren alle wesentlichen Aspekte der Klassifikationsergebnisse. Dafür werden die manuell klassifizierten Bilder als Referenzdaten genutzt.

Es werden Testdaten verwendet, auf die das ResNet50-Modell während des Trainingsprozesses noch keinen Zugriff hatte. Die Bilder werden zufällig aus dem verfügbaren Datensatz ausgewählt, um sicherzustellen, dass die Stichprobe möglichst repräsentativ ist. Es kommen verschiedene Methoden zum Einsatz, um die Leistung des Modells zu untersuchen und zu verbessern. Erstens die Visualisierung der Klassifikationsergebnisse, an welcher man Fehlklassifikationen direkt erkennen kann. Darauf folgen mögliche Verbesserungsansätze und deren Umsetzung und abschließend werden die bereits im Kapitel 3.5 vorgestellten Programme zur endgültigen Ermittlung der Validierungswerte des Modells genutzt.

4.3.1 Visueller Vergleich der Klassifikationsergebnisse

Bevor die Klassifizierungsergebnisse anhand der genannten Metriken bewertet werden, werden sie einer optischen Kontrolle unterzogen. Dafür werden einige Bilder ausgewählt, anhand welcher überprüft werden sollte, wie genau das Modell die Bilder bereits klassifizieren kann. Nach der Auswahl eines zu klassifizierenden Bildes, braucht das neuronale Netz etwa 15 Sekunden, um den Klassifizierungsvorgang abzuschließen. Danach wird das Bild über das Graphical User Interface angezeigt.

Das neuronale Netz erzielte beim Klassifizieren folgende Ergebnisse (siehe Abbildung 22):

Das erste Bild stellt das Artefakt Water (rot) dar, während das zweite und das dritte Bild die Klassen Strongnoise (grün) und Mediumnoise (gelb) repräsentieren.

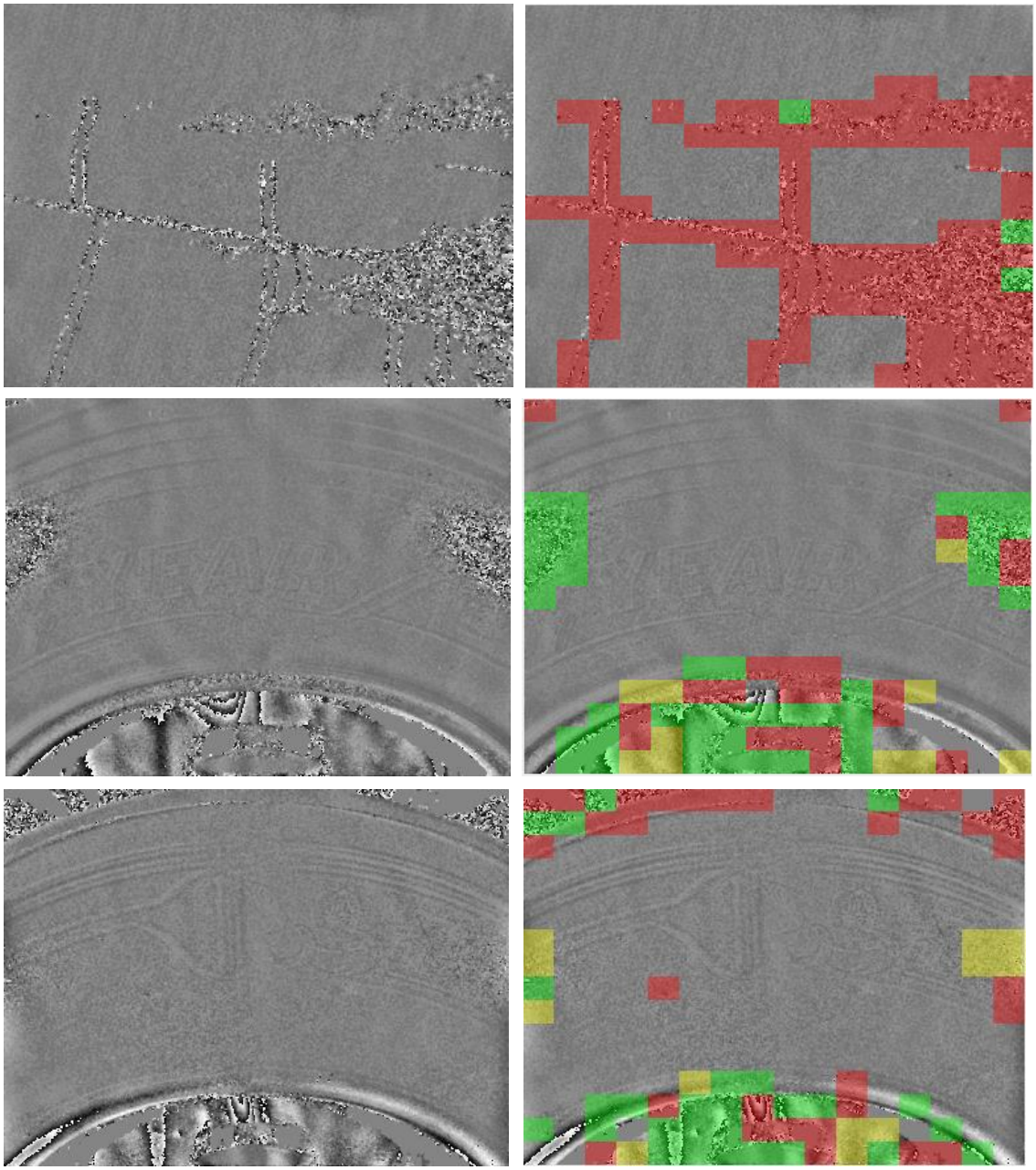

Abbildung 22: Vergleich von Originalbild (links) mit klassifiziertem Bild (rechts) (Eigene Darstellung in Anlehnung an SDS, 2024)

Die Artefakte werden bereits gut vom Modell erkannt. Allerdings ist unschwer zu erkennen, dass sich an einigen Stellen mehrere Farben mischen oder einzelne Kacheln mitten im Bild als Artefakte gekennzeichnet werden. Auch der Hintergrund der Sidewall, bereitet wie bereits bei der Annotation der Bilder, auch bei der Klassifikation Schwierigkeiten. Wie mit diesen Erkenntnissen umgegangen wird, wird im nächsten Kapitel erläutert.

4.3.2 Herausforderungen und Lösungsansätze

Die Klassifikationsergebnisse sind, wie erwähnt, bereits recht zufriedenstellend. Alle Artefakte werden vom ResNet50-Modell gekennzeichnet, auch wenn die Farbe noch nicht immer korrekt gewählt wird. Allerdings wird zusätzlich auch der Hintergrund der Bilder, der Seitenwand des Reifens, als Artefakt klassifiziert. Das liegt daran, dass das Modell die im Hintergrund vorhandenen Strukturen während des Trainingsprozesses nicht zu sehen bekommen hatte. Da eine Einbindung dieser in die Trainingsdaten nur zu zukünftigen Klassifikationsfehlern führen würde, ist die einzige Option, den Hintergrund, genau wie bei der Annotation der Seitenwandbilder, schwarz zu hinterlegen.

Zur Maskierung der Bilder kommt Labelstudio zum Einsatz. Die Bilder, die beispielsweise für die Validierung des Modells genutzt werden sollen, werden dementsprechend in Labelstudio hochgeladen und alle einzeln von Hand mit einem Pinsel maskiert. Da das Maskieren ausschließlich auf den Bildern der Seitenwand nötig ist, müssen nur die Bilder mit den Klassen Mediumnoise und Strongnoise maskiert werden. Die Bilder mit dem Artefakt Water, erfordern keine weiteren Vorverarbeitungen.

Nach dem Erstellen der Maske, kann diese als PNG-Datei heruntergeladen werden. Das Originalbild und die Maske werden anschließend in ein selbst erstelltes Python-Programm geladen und von diesem miteinander kombiniert. Das maskierte Bild wird daraufhin in einem extra Ordner abgespeichert.

Diese Anpassung sorgte dafür, dass das neuronale Netz die Kacheln des Hintergrunds nicht mehr falsch zuordnet und dementsprechend auch keine Fehlklassifikationen mehr in diesem Bereich auftreten können. Im direkten Vergleich zu den zuvor klassifizierten Bildern ohne Maskierung, kann hier schon eine deutliche Verbesserung festgestellt werden (siehe Abbildung 23).

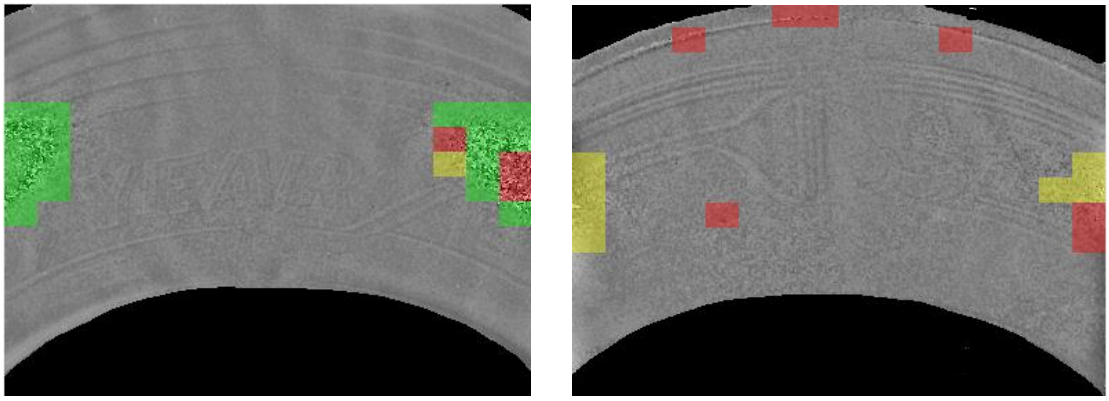

Abbildung 23: Zur Klassifikation maskierte Shearographie-Bilder (Eigene Darstellung in Anlehnung an SDS, 2024)

Wie erwähnt kann das Modell Stellen im Bild, die sich vom Rest abheben, bereits gut erkennen. Nur mit der richtigen Klasse hat es noch etwas Probleme. Da das Artefakt Wasser auf den Kacheln sehr ähnlich aussieht, wie starkes Rauschen, vertauscht das neuronale Netz diese gelegentlich. Dabei wird starkes Rauschen hin und wieder als Wasser vorhergesagt und umgekehrt. Außerdem werden sehr kleine und meistens alleinstehende „Unebenheiten“ im Bild manchmal als Wasser klassifiziert, obwohl diese kaum zu erkennen sind.

Um diese beiden Probleme in den Griff zu bekommen, wird das Skript, das für den Klassifikationsvorgang zuständig ist, um einige Funktionen erweitert. Es kommen dafür drei Methoden zur Verbesserung der Segmentierung zum Einsatz.

Vor der Ausführung der drei Methoden werden einige Informationen benötigt. Dazu zählen die vorhergesagten Wahrscheinlichkeiten der einzelnen Klassen für jede Kachel, die Anzahl der Kacheln je Klasse und die Gesamtzahl aller Kacheln, die nicht als „Good“ klassifiziert werden.

Die Wahrscheinlichkeiten werden während des Klassifizierungsprozesses in einer extra Liste abgespeichert. Die Anzahl an Kacheln je Klasse wird durch das Aufsummieren der jeweiligen Klassen in der Liste, die die Klassifizierungsergebnisse enthält, ermittelt. Um nun die Anzahl aller Kacheln mit Artefakten zu erhalten, werden die drei vorher ermittelten Werte aufsummiert.

Nachdem alle notwendigen Werte zur Verfügung standen, konnte mit der Anpassung der Überempfindlichkeit des Modells begonnen werden. Damit das neuronale Netz nicht mehr jede kleine Anomalie direkt als Artefakt kennzeichnet, werden die benachbarten Kacheln jeder Kachel auf weitere Artefakte untersucht. Falls in einem Radius von 1 keine weiteren Kacheln desselben Artefakts existieren, wird die Kachel entsprechend den Klassen ihrer Nachbarn angepasst. Falls sich allerdings ein weiteres Artefakt im Umkreis befindet, bleibt die Klassifikation bestehen, um keine wirklich existierenden Artefakte außer Acht zu lassen.

Um die Fehlklassifikationen von Wasser und starkem Rauschen zu verhindern, wird ein logischer Vergleich eingebaut, der überprüft, ob ein Großteil der Kacheln einer speziellen Klasse entspricht. Falls das der Fall ist, werden die wenigen anders klassifizierten Kacheln entsprechend angepasst. Für den Fall, dass eine Kachel mit Rauschen fälschlicherweise als Wasser vorhergesagt wird, wird diese anhand der bestimmten Wahrscheinlichkeiten pro Klasse umklassifiziert. Falls also die nächsthöchste Wahrscheinlichkeit die Klasse

„Mediumnoise“ ist, wird das Label dieser Kachel dementsprechend zu dieser umgeschrieben.

Diese Vorgehensweise ist nur möglich, da die Artefakte „Water“ und „Noise“ stehts getrennt voneinander auftreten. Auf Bildern mit Wasser, befindet sich kein Rauschen und umgekehrt. Die beiden Arten von Rauschen können allerdings sehr wohl im gleichen Bild aufeinandertreffen.

Um die umgesetzten Optimierungen des Klassifikationsprozesses auf ihre Funktion zu überprüfen, werden dieselben Bilder, die bereits klassifiziert wurden, erneut durch das neuronale Netz geschickt. Die Klassifikationsergebnisse sahen dieses Mal wesentlich vielversprechender aus (siehe Abbildung 24).

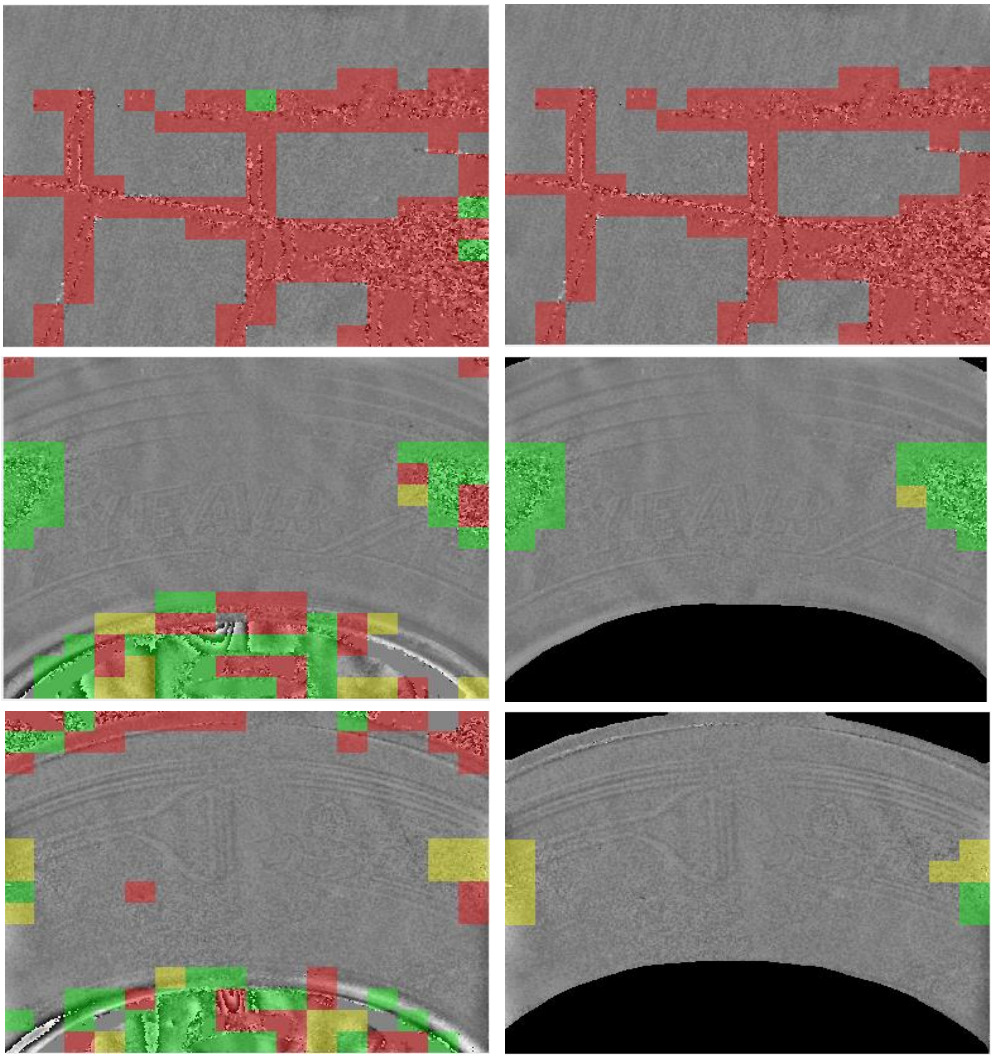

Abbildung 24: Vergleich der klassifizierten Bilder vor (links) und nach (rechts) den vorgenommenen Optimierungen (Eigene Darstellung in Anlehnung an SDS, 2024)

Das Problem der falsch klassifizierten Kacheln ist damit behoben und der Klassifizierungsprozess kann offiziell validiert werden.

4.3.3 Ermittlung der Validierungsergebnisse

Um die Performance des Modells nun anhand fester Zahlenwerte zu bewerten, erfolgt eine Validierungsdatenerhebung. Die gesamte Vorgehensweise wurde bereits in Kapitel 3.5 erläutert und wird in diesem Kapitel nun durchgeführt.

Um das Modell zu validieren, wird für jedes Artefakt eine gewisse Anzahl an Kacheln überprüft. Die Bilder, die Wasser enthalten werden einzeln validiert, während Bilder mit mittlerem und starkem Rauschen zusammen validiert werden können. Das liegt daran, dass starkes Rauschen meistens in mittleres Rauschen übergeht und deshalb beide Artefakte häufig auf demselben Bild vorkommen.

Die ganzen zufällig ausgewählten Bilder werden manuell durch das selbst erstellte Label-Programm klassifiziert. Die subjektive Meinung der Person, die die Bilder manuell labelt, spielt hier also eine große Rolle. Nachdem dieser Vorgang für alle Bilder durchgeführt ist, kann die automatische Klassifizierung durch das ResNet50-Modell erfolgen.

Wenn alle notwendigen Daten gesammelt sind, können sie von dem Validierungsprogramm miteinander verglichen werden. Dabei entsteht für jedes Bild eine Textdatei, die die vier bereits genannten Kennwerte enthält. Diese sehen beispielsweise so aus (siehe Abbildung 25):

Water: Accuracy: 0.95 Precision: 0.9798 Recall: 0.9700 F1-Score: 0.9749 Mediumnoise: Accuracy: 0.00 Precision: 0.0000 Recall: 0.0000 F1-Score: 0.0000 Strongnoise: Accuracy: 0.00 Precision: 0.0000 Recall: 0.0000 F1-Score: 0.0000	Water: Accuracy: 0.00 Precision: 0.0000 Recall: 0.0000 F1-Score: 0.0000 Mediumnoise: Accuracy: 0.68 Precision: 0.7143 Recall: 0.7692 F1-Score: 0.7407 Strongnoise: Accuracy: 0.97 Precision: 1.0000 Recall: 0.9688 F1-Score: 0.9841

Abbildung 25: Ausgabe des Validierungsprogramms (eigene Darstellung)

Bei dem linken Block handelt es sich um ein Bild mit Wasser und bei dem rechten Block um ein Bild mit mittlerem und starkem Rauschen. Die Werte der jeweils anderen Klassen werden dabei nicht berücksichtigt und geben dementsprechend eine „0“ aus.

Um nun aber ein zusammengefasstes und aussagekräftiges Ergebnis zu erhalten, muss aus den einzelnen Zahlwerten der Durchschnitt gebildet werden. Dafür kommt ein weiteres sehr simples Programm zum Einsatz. Es addiert die Werte, die jeweils in derselben Zeile stehen und teilt anschließend die Summe durch die Anzahl der bewerteten Bilder. Durch dieses Vorgehen kann bestimmt werden, wie genau das Modell auf die Artefakte der Validierungsbilder reagiert und was für eine Klassifizierungsleistung es erbringt.

Die endgültigen Ergebnisse der Klassen „Water, Mediumnoise und Strongnoise““ sehen aus wie folgt (siehe Abbildung 26):

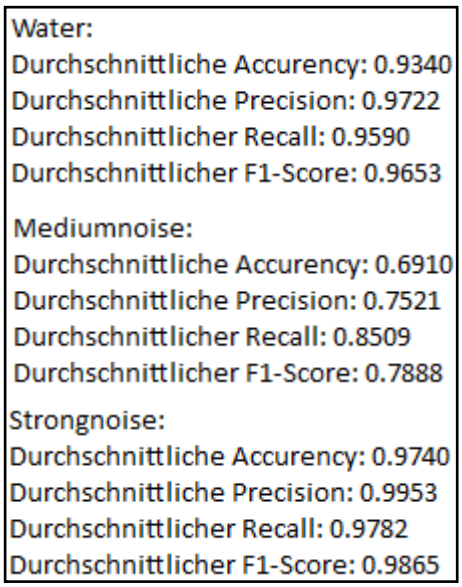

Abbildung 26: Evaluationsmetriken des ResNet50-Modells (eigene Darstellung)

Es lässt sich direkt erkennen, dass die Ergebnisse der Klassen Water und Strongnoise deutlich besser sind als die der Klasse Mediumnoise.

Um den Grund dafür zu ermitteln und einordnen zu können, erfolgt im nächsten Kapitel eine detaillierte Interpretation der Ergebnisse und deren Bedeutung für die Gesamtleistung des Modells.

5 Diskussion und Ausblick

5.1 Interpretation der Validierungsergebnisse

Durch das Durchführen von Analysen und Optimierungen konnte gezeigt werden, dass das ResNet50-Modell insbesondere bei den Artefakten „Water“ und „Strongnoise“ eine hohe Klassifikationsleistung erreichte. Da jedoch trotzdem Unterschiede in der Genauigkeit und den weiteren Kennwerten bestehen, werden diese im folgenden Abschnitt ausgiebig untersucht.

Water:

Bei der Erkennung von Wasser konnte eine Genauigkeit von 93,4% erreicht werden. Unter der Berücksichtigung der beiden anderen Werte Precision und Recall lässt sich über das Modell sagen, dass die meisten Kacheln richtig klassifiziert werden. Die als „Water“ klassifizierten Kacheln beinhalten zu über 97% auch wirklich Wasser und die Kacheln, die tatsächlich Wasser aufweisen werden auch vom Modell zu nahezu 96% richtig erkannt. Das sind sehr gute Ergebnisse.

Beim Vergleich des automatisch klassifizierten Bildes zu dem manuell Klassifizierten, lässt sich erkennen, wodurch sich die geringe Abweichung zum Optimalfall (100%) zusammensetzt (siehe Abbildung 27). Das neuronale Netz erkennt in seltenen Fällen kleinste Wasserrückstände direkt als Artefakt und lässt ab und zu eine leicht befallene Kachel ohne Einfärbung durchgehen.

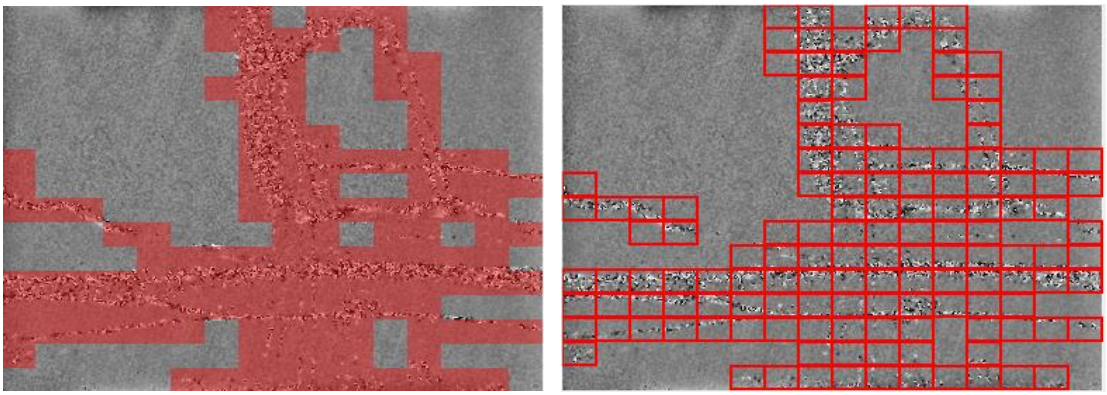

Abbildung 27: Vergleich von automatisch klassifiziertem Wasser-Bild (links) mit manuell Klassifiziertem (rechts) (Eigene Darstellung in Anlehnung an SDS, 2024)

Für den Zweck dieses Klassifikationsverfahrens ist das allerdings irrelevant, weil feinste Details für diese Arbeit keine größere Rolle spielen. Solange die wirklich auffälligen und gut sichtbaren Artefakte erkannt werden, sind die Anforderungen an den Klassifikationsprozess erfüllt.

Mediumnoise: Die Genauigkeit der Klasse Mediumnoise fiel mit knapp 70% etwas gering aus. Bei Betrachtung des Precision-Wertes lässt sich feststellen, dass einige (ca. 25%) vom neuronalen Netz als positiv vorhergesagten Kacheln , im manuell klassifizierten Bild nicht positiv waren. Der Recall-Wert von ca. 85% besagt, dass viele tatsächlich positve Kacheln auch vom Modell als solche erkannt wurden (siehe Abbildung 28). Da mittleres Rauschen sich durch einen dunkleren Grauton im Bild bemerkbar macht, kann durchaus beim manuellen Klassifizierungsvorgang eine Kachel mit diesen Beschaffenheiten übersehen werden. Das Modell sucht währenddessen systematisch das gesammte Bild nach dieser Struktur ab und erkennt das Artefakt eventuell an einigen Stellen genauer, als das ein Mensch könnte.

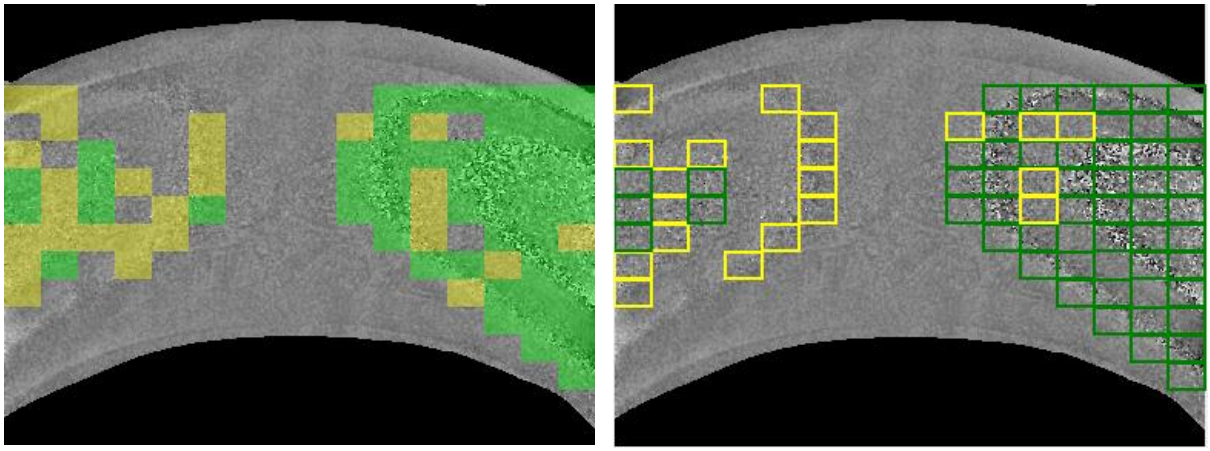

Abbildung 28: Vergleich von automatisch klassifiziertem Rausch-Bild (links) mit manuell Klassifiziertem (rechts) (Eigene Darstellung in Anlehnung an SDS, 2024)

Mittleres Rauschen tritt oft als Ausläufer von starkem Rauschen in Bildern auf. Es stellt praktisch eine Zwischenstufe zwischen starkem Rauschen und dem normalen Grauton der Shearographie-Bilder dar. Es verdeckt keine anderen eventuell sicherheitsgefärdenden Befunde oder Artefakte, sondern stellt meistens das Ende eines Strongnoise-Artefaks dar. Aus diesem Grund und dadurch, dass die subjektive Meinung beim manuellen Klassifizieren eine sehr große Rolle spielt, kann dieser Wert als akzeptabel angesehen werden und muss nicht weiter optimiert werden.

Strongnoise:

Beim Artefakt Strongnoise konnten die bisher besten Ergebnisse erzielt werden. Alle vier Kennzahlen weichen nur minimal von 100% ab. Da starkes Rauschen deutlich zu erkennen ist und durchaus andere Befunde überdecken kann, ist die präzise Erkennung dieses Artefakts ausschlaggebend. Da die Struktur von starkem Rauschen sich deutlich vom Rest

des Bildes abhebt, hat das neuronale Netz keine Probleme die betroffenen Kacheln (grün) zu identifizieren (siehe Abbildung 29).

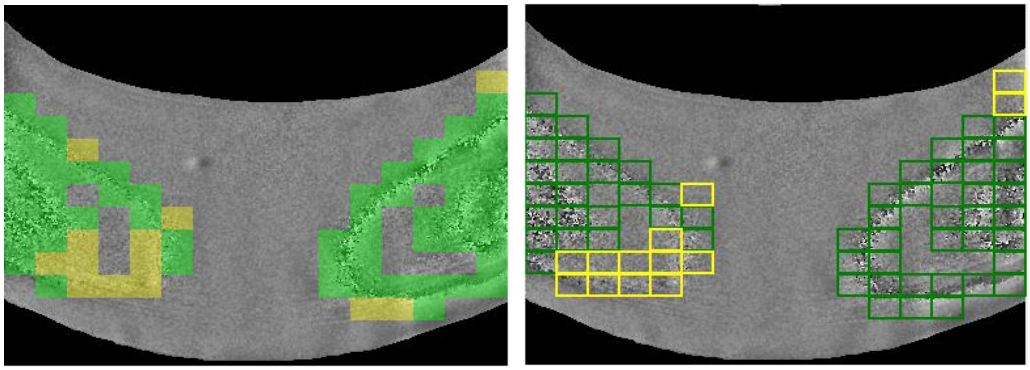

Abbildung 29: Vergleich von automatisch klassifiziertem Rausch-Bild (links) mit manuell Klassifiziertem (rechts) (Eigene Darstellung in Anlehnung an SDS, 2024)

Bei allen Bildern, die für die Validierung genutzt wurden, konnte das Modell problemlos nahezu alle von Strongnoise betroffenen Kacheln richtig klassifizieren. Für den seltenen Fall, dass eine Kachel falsch klassifiziert wurde, handelte es sich dabei meistens um ein nur schwaches Vorkommen der Struktur von starkem Rauschen, oder aus der Sicht des Recall-Wertes um Kacheln, die nur am äußersten Rand von starkem Rauschen befallen waren. Das ist für diese Arbeit allerdings irrelevant.

Damit ist die Frage, ob das Modell den Anforderungen dieser Arbeit entspricht, beantwortet. Alle Artefakte, die einen anderen Befund überdecken, oder aus anderen Gründen ein Sicherheitsrisiko darstellen können, werden zuverlässig von dem ResNet50-Modell als solche gekennzeichnet.

5.2 Limitationen der Studie

Trotz der zufriedenstellenden Ergebnisse des Klassifikationsprozesses, gibt es einige limitierende Faktoren, die während der Erstellung des Klassifikationsprogramms aufgetreten sind.

Ein wesentlicher limitierender Faktor ist der begrenzte Datensatz. Aufgrund der eingeschränkten Rechenleistung des verwendeten Referenzrechners konnte nur eine begrenzte Anzahl an Bildern für den Trainingsprozess genutzt werden. Der Trainingsprozess dauert bereits bei 6000 Bildern über 5 Stunden und würde bei Erweiterung der Eingabedaten dementsprechend noch mehr Zeit in Anspruch nehmen.

Auch die Berechnung der Klassifikation dauert aktuell ca. 15 Sekunden pro Bild. Das könnte für eine Echtzeitanwendung eventuell zu hoch sein. Da in der Praxis leistungsfähigere Hardware zum Einsatz kommen könnte, wäre eine weitere Evaluierung unter realen Bedingungen sinnvoll.

Zudem wurde die Generalisierbarkeit des trainierten Modells auf bisher nicht berücksichtigte Artefakte oder Befunde nicht untersucht, was eine Einschränkung hinsichtlich seiner Anwendbarkeit in anderen Kontexten darstellt.

Einen weiteren limitierenden Faktor stellen die manuelle Annotation und Maskierung dar. Da sowohl die Annotation als auch die Maskierung manuell erfolgen, unterliegen sie subjektiven Einflüssen. Die manuelle Annotation kann eine Verzerrung enthalten, da verschiedene Personen das gleiche Bild unterschiedlich interpretieren könnten. Außerdem ist dieser Prozess zeitaufwändig und kann zu Ungenauigkeiten im Klassifikationsprozess führen.

Der letzte relevante Faktor bezieht sich auf die auftretenden Fehlklassifikationen. Wasser und starkes Rauschen sehen, solange die Bilder noch in ihrer Originalgröße vorliegen, von ihrer Struktur her sehr unterschiedlich aus. Da die Bilder aber in 256 Kacheln aufgeteilt werden und die Gesamtstruktur der Artefakte auf diesen nicht mehr zu erkennen ist, hat das Modell Schwächen bei der Unterscheidung zwischen den beiden Klassen. Diese Schwäche wird zwar durch die Optimierung des Klassifikationsskripts ausgeglichen, die Ursache konnte jedoch nicht beseitigt werden.

Diese Limitationen sollten bei der Interpretation der Ergebnisse berücksichtigt werden.

5.3 Mögliche Verbesserungen und zukünftige Forschungsperspektiven

Das trainierte Modell und die erstellte Software erfüllen ihren Zweck und erzielen bereits überzeugende Ergebnisse. Nichtsdestotrotz existieren einige Faktoren, die zukünftig noch einer Verbesserung bedürfen bzw. an denen in Zukunft weiter geforscht werden kann.

Automatisches Maskieren der Seitenwände:

Die Maskierung der Bilder der Reifenseitenwände erfolgte manuell. Dies ist bei einer großen Anzahl an Bildern mit einem enormen Zeitaufwand verbunden. Für die Zukunft wäre es sinnvoll diesen Prozess zu automatisieren. Dazu wäre wiederum ein weiteres Bildverarbeitungsprogramm denkbar, das die Bilder der Seitenwände analysiert und ab der Stelle, an der der Hintergrund beginnt, das restliche Bild schwärzt.

Fehlermanagement und Unsicherheiten:

Das Modell liefert für jede Kachel eine Klassifizierung, ohne dass eine Unsicherheitsschwelle oder eine besondere Behandlung von schwer zu klassifizierenden Kacheln berücksichtigt wird. Wenn die Wahrscheinlichkeiten mehrerer Klassen gleich hoch sind, werden sie nicht gesondert behandelt. Eine mögliche Erweiterung könnte eine visuelle Kennzeichnung von Kacheln sein, bei denen das Modell unsicher ist, oder eine Mindestwahrscheinlichkeitsschwelle, unterhalb derer eine Kachel als „unbestimmt“ markiert wird.

Batch-Processing während dem Klassifizierungsprozess:

Im Trainingsprozess des ResNet50-Modells kommt bereits eine Batch-Verarbeitung zum Einsatz. Diese Vorgehensweise könnte auch für den Klassifikationsprozess umgesetzt werden. Das würde für eine effizientere Nutzung der GPU oder CPU sorgen, ein allgemein stabileres Training und eine schnellere Konvergenz des Modells ermöglichen und außerdem die Hardware-Ressourcen optimal ausnutzen.

Erweiterung und Diversifizierung des Datensatzes:

Durch die Erweiterung des genutzten Datensatzes um neue Trainingsdaten, kann die Generalisierbarkeit des Modells sich verbessern. Umso mehr Trainingsdaten dem neuronalen Netz zur Verfügung stehen, desto besser sind in der Regel die Ergebnisse bei der Klassifikation. Um keine neuen Bilder annotieren zu müssen, kann eine Datenaugmentation

auf alle Bilder angewendet werden, die durch Rotation, Skalierung oder Helligkeitsanpassungen der Bilder die Robustheit des Modells erhöhen kann.

Manuelles Klassifizieren von Validierungsbildern durch mehrere Personen

Für diese Arbeit erfolgte die manuelle Klassifizierung der Validierungsdaten durch eine einzige Person. Die Validierungsgrundlage bezog sich dementsprechend ausschließlich auf die subjektive Meinung dieser Person. Falls in Zukunft weiterhin die in dieser Arbeit beschriebene Validierungstechnik angewandt werden soll, könnten mehrere Personen dieselben Bilder einzeln klassifizieren und aus diesen der Durchschnitt ermittelt werden. Das würde für eine repräsentativere Validierungsgrundlage sorgen.

Ausblick:

In einem nächsten Schritt kann die entwickelte Software in die Prüfmaschinen von SDS-Systemtechnik integriert werden. Sobald sie auf die Maschinen gespielt wurde, können Feldtests starten, die die Software unter realen Bedingungen auf die Probe stellen. Dabei kann ihre Funktionalität und Leistungsfähigkeit unter realistischen Einsatzszenarien validiert werden, um ihre Praxistauglichkeit sicherzustellen.

6 Fazit

6.1 Zusammenfassung der Erkenntnisse

In dieser Arbeit wird ein Klassifikationsmodell zur automatisierten Erkennung der Artefakte ‚Wasser' sowie ‚mittleres' und ‚starkes' Rauschen auf Shearographie-Bildern entwickelt und evaluiert. Der Fokus liegt dabei auf der Effizienz des Trainings- und Klassifikationsprozesses sowie den Herausforderungen, die bei der Implementierung dieser entstehen. Die genutzte Modellarchitektur (vortrainiertes ResNet50-Modell) und die Trainingsstrategie erwiesen sich als geeignet für die gestellte Aufgabe. Die Ergebnisse zeigen, dass durch den Einsatz moderner KI-Methoden die Qualität und Verlässlichkeit der Reifenprüfung verbessert werden kann. Abgesehen davon spart die Automatisierung dieses Prozesses auf längere Sicht gesehen zusätzlich Zeit und Geld.

Durch die abschließende Analyse der Modelleistung konnte festgestellt werden, dass das trainierte Modell eine hohe Genauigkeit bei der Erkennung der Artefakte Wasser und Rauschen erreichen konnte. Die Bereiche im Bild, auf denen ein Artefakt zu sehen ist, werden zuverlässig vom Modell erkannt und farblich hervorgehoben. Die Ergebnisse der Analyse zeigen, dass die Anforderungen an das Programm erfüllt wurden und es zuverlässig und exakt arbeitet. Dennoch bestehen Herausforderungen hinsichtlich des Annotationsprozesses, der Erkennung von mittlerem Rauschen und der vom Modell benötigten Rechenzeit. Zudem könnte eine noch größere Datenmenge zu einer Verbesserung der Klassifikationsergebnisse beitragen.

Zukünftige Forschungen könnten sich darauf konzentrieren, die Genauigkeit des Modells durch neue, umfangreichere Trainingsdaten weiter zu optimieren, den Maskierungsprozess der Bilder der Seitenwand des Reifens zu automatisieren, ein geeignetes Fehlermanagement in den Prozess einzubinden und die weiteren Artefakte und Befunde in den Trainings- und Klassifikationsprozess einzubinden.

Die Implementierung des vorgestellten Programms in Echtzeit-Prüfsysteme, ist abschließend ein wichtiger Schritt zur Validierung des Modells auf seine industrielle Anwendbarkeit.

6.2 Relevanz der Ergebnisse für die Praxis

Das in dieser Arbeit entwickelte Deep-Learning-Modell hat das Potenzial, gewinnbringend in die automatisierte Analyse von Bilddaten integriert zu werden. Die vielversprechenden Ergebnisse des Klassifikationsprozesses bilden eine solide Grundlage für die Umsetzung eines automatisierten Prüfprozesses für Reifen jeglicher Art.

Durch die Einbindung des trainierten Modells in den Prüfprozess kann die Analyse großer Bilddatensätze automatisiert und erheblich beschleunigt werden. Dies spart nicht nur Zeit und Kosten, sondern ermöglicht zudem eine konsistentere und objektivere Bewertung.

Insgesamt zeigen die erzielten Ergebnisse, dass die automatisierte Wasser- und Rauscherkennung eine vielversprechende Grundlage für den praktischen Einsatz bietet. Weitere Optimierungen, insbesondere hinsichtlich der Verarbeitungszeit sowie der Integration zusätzlicher Artefakte und Befunde, könnten den Übergang in die Praxis weiter erleichtern.

7 Literaturverzeichnis

Ajani, B., & Bharadwaj, A. (2019, April 6). *Adaptive Moment Estimator (Adam) Optimizer in ITKv3*.

B. Rathod, D., & Gadhavi, V. (2013). A Review on Deep Learning Using Python Libraries and Packages. In *INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS* (S. 83–87). https://ijcrt.org/papers/IJCRT1134596.pdf

Borchers, A. (2009). *Von der Shearografie zur Druckprüfung* [Post]. https://reifenpresse.de/2009/03/16/von-der-shearografie-zur-druckpruefung/

Buxmann, P., & Schmidt, H. (2021). *Künstliche Intelligenz—Mit Algorithmen zum wirtschaftlichen Erfolg* (2. Aufl.). Springer Gabler.

Ettenmeyer, A. (1991). Shearografie—Ein optisches Verfahren zur zerstörungsfreien Werkstoffprüfung / Shearography—An optical method for nondestructive testing. *tm - Technisches Messen*, *58*, 247–252.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. https://www.deeplearningbook.org/contents/optimization.html

Goutte, C., & Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In *Advances in Information Retrieval* (S. 345–359). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-31865-1_25

Gupta, A., Ramanath, R., Shi, J., & Keerthi, S. S. (2021). *Adam vs. SGD: Closing the generalization gap on image classification*.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. https://doi.org/10.48550/arXiv.1512.03385

Holdsworth, J., & Scapicchio, M. (2024, Juni 17). *Was ist Deep Learning? | IBM*. https://www.ibm.com/de-de/topics/deep-learning

Honlet, M., & Walz, T. (2002). Shearografie Stand der Technik eines optischen Verfahrens für die ZfP. *ZfP-Zeitung*, 39–41.

Mao, A., Mohri, M., & Zhong, Y. (2023). *Cross-Entropy Loss Functions: Theoretical Analysis and Applications*. https://doi.org/10.48550/ARXIV.2304.07288

Marquardt, E. (2020). Künstliche Intelligenz in optischen Mess- und Prüfsystemen: Chance oder Hype? *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, *115*(10), 731–733. https://doi.org/10.1515/zwf-2020-1151019

Mentzel, H.-J. (2021). *Künstliche Intelligenz bei Bildauswertung und Diagnosefindung*. *8*, 694–704.

Narwade, D., & Kolhe, V. (2023). Signature Verification using ResNet-50 Model. *International Journal of Scientific Research in Science, Engineering and Technology*, 278–291. https://doi.org/10.32628/IJSRSET2310612

Pouly, M., Koller, T., & Lionetti, S. (2020). *Künstliche Intelligenz in der Bildanalyse – Grundlagen und neue Entwicklungen*. 660–668.

Praveen, T. N. V. S., Sivathmika, D., Jahnavi, G., & Bolledu, J. (2023). An In-depth Exploration of ResNet-50 for Complex Emotion Recognition to Unraveling Emotional States. *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, 1–5. https://doi.org/10.1109/InCACCT57535.2023.10141774

SDS. (2024). *Anonymisierter Testdatensatz aus der Praxis* [Dataset].

Sheikh, L. M. K., Shaikh, A., Sandupatla, A., Pudale, R., Bakare, A., & Chavan, Prof. M. (2024). Classification of Simple CNN Model and ResNet50. *International Journal for Research in Applied Science and Engineering Technology*, *12*(4), 4606–4610. https://doi.org/10.22214/ijraset.2024.60677

Sönke, M. (2018). *Automatisiertes Lernen von Neuronalen Netzen zur Objektklassifikation auf heterogenen Systemen*. Technischen Universität Carolo-Wilhelmina zu Braunschweig.

Wuttke, L. (2023a, Mai 24). *Deep Learning: Was ist es und warum wird es eingesetzt?* Datasolut GmbH. https://datasolut.com/was-ist-deep-learning/

Wuttke, L. (2023b, August 18). *Bilderkennung: Definition und Anwendungsbereiche*. Datasolut GmbH. https://datasolut.com/bilderkennung/