HAW HAMBURG

BACHELORARBEIT

Künstliche neuronale Netze für die Bildklassifikation von Fledermausrufen

vorgelegt am 29. Juli 2022
Kilian Mauson, Matrikelnummer: [redacted]

Erstprüferin: Prof. Dr. Tessa Taefi
Zweitprüfer: Marc Roswag

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
22081 Hamburg

Zusammenfassung

In dieser Arbeit wird untersucht, inwiefern sich vortrainierte künstliche neuronale Netze für die automatische Klassifizierung von Sonagrammen bei der Auswertung von erfassten Fledermausrufen mittels UAS eignen. Dafür werden Convolutional Neural Networks und Vision Transformer als vielversprechende Bildklassifizierungsmethoden vorgestellt und mit Daten aus Fledermausrufen sowie Störsignalen trainiert. Die Ergebnisse werden anhand von Metriken des maschinellen Lernens miteinander verglichen. Bei der Durchführung des Experiments wurden Klassifikationsgenauigkeiten von über 90% erreicht.

Abstract

This thesis investigates the extent to which pre-trained artificial neural networks are suitable for the automatic classification of sonagrams when evaluating recorded bat calls using UAS. For this purpose, convolutional neural networks and vision transformers are presented as promising image classification methods and trained with data from bat calls and noise signals. The results are compared using machine learning metrics. When the experiment was carried out, classification accuracies of over 90% were achieved.

Inhaltsverzeichnis

1 Einleitung

1.1 Problematik und Zielsetzung

Laut einer Studie des Leibniz-Instituts aus dem Jahre 2021 sterben in Deutschland jährlich mehrere hunderttausend Fledermäuse durch den Betrieb von Windenergieanlagen (WEA) (Kruszynski et al., 2022). Da diese Tiere aufgrund Ihrer Echoortung die Rotoren der WEA nicht wahrnehmen können, sterben sie entweder durch eine Kollision mit den sich schnell bewegenden Rotorblättern oder durch den, hinter den Blättern auftretenden, starken Druckabfall. Das hat verheerende Folgen sowohl für die Populationsentwicklung als auch für die Ökosysteme, in denen die Fledermäuse leben, weil zum einen Fledermäuse eine geringe Fortpflanzungsrate haben und zum anderen, weil sie wichtige regulierende Funktionen in den Ökosystemen übernehmen (Kruszynski et al., 2022). Zudem zählen Fledermäuse zu den am strengsten geschützten Arten in Deutschland (Voigt et al., 2015).

Damit der Aufbau neuer WEA den Artenschutz nicht gefährdet, werden Abschaltzeiten bei der Planung neuer Anlagen und dem Betrieb bestehender Anlagen eingeführt, die die Todesrate von Fledermäusen verringern sollen (Schreiber et al., 2016). Weiterhin werden vom Boden der WEA aus Messungen durchgeführt, die in der Umgebung Fledermausrufe erfassen (Hurst et al., 2020). Die Erfassung wird mithilfe von akustischen Detektoren durchgeführt, die eine begrenzte Reichweite besitzen. Die Messergebnisse haben Einfluss auf die Standortwahl neuer WEA (Hurst et al., 2020). Bei dieser Art der Erfassung besteht das Problem, dass aufgrund der geringen Reichweite nur ein kleiner Bereich um den WEA erfasst werden kann (Hurst et al., 2020). Deshalb kann unter Umständen über juristischem Wege der Ausbau von neuen WEA gestoppt werden, da nicht gewährleistet werden kann, dass sich in einem ausreichend großen Abstand um den WEA herum keine gefährdeten Fledermausarten befinden.

Eine Verbesserung der Messtechnik kann durch den Einsatz von Unmanned Ariel Systems (UAS) erreicht werden. UAS sind unbemannte Flugkörper mit eigenem Antrieb, die entweder von einem Menschen ferngesteuert werden oder autonom fliegen und anwendungsspezifische Technologien besitzen wie zum Beispiel Kameras oder Sensoren (*Unmanned Aircraft Systems*, o. D.). Um Fledermäuse zu erkennen, werden UAS mit Ultraschallmikrofonen ausgestattet. Diese Mikrofone sind für den Ultraschallfrequenzbereich zwischen 20 und 150 kHz konzipiert und können das Aktivitätsniveau eines Fledermausrufes identifizieren. Die Aufzeichnung von Rufsequenzen wird dabei jedoch nur dann ausgelöst, wenn ein Fledermausruf erkannt wird. Sobald dieser Ruf vorbei ist, wird die Aufzeichnung wieder unterbrochen.

Der Vorteil der akustischen Erfassung von Fledermäusen mittels UAS ist der größere Einsatzbereich. Anstelle von stationären Messmasten oder Gutachtern, die zu Fuß mit einem Mikrofon den Boden ablaufen, können UAS einerseits das Untersuchungsgebiet um die vertikale Ebene erweitern und andererseits horizontal einen größeren Radius abdecken.

Beim Einsatz der UAS kann jedoch folgendes Problem auftreten: Das angebundene Ultraschall Mikrofon interpretiert Störgeräusche, die durch den Flugantrieb entstehen, als vermeintliche Fledermausrufe. In diesem Fall werden Audiosequenzen aufgenommen, ohne dass in diesen Fledermausrufe wiederzufinden sind. Insofern könnte fälschlicherweise das Vorhandensein von gefährdeten Fledermausarten gemessen werden, obwohl sich dort in Wirklichkeit keine aufhalten. Um dies zu vermeiden, müssen die fehlerhaften Audiosequenzen aussortiert werden. Durch die Interpretation von Sonagrammen kann festgestellt werden, ob es sich um ein Störsignal handelt oder um einen Ruf. Doch ein manuelles Aussortieren der Sonagramme ist mit einem erheblichen Zeitaufwand verbunden. Die grundlegende Hypothese dieser Arbeit ist, dass dieser Prozess mithilfe von künstlicher Intelligenz gestützter Bilderkennungssoftware um ein Vielfaches beschleunigt und automatisiert werden kann. Insbesondere vortrainierte künstliche neuronale Netze haben dahingehend in den letzten Jahren für verschiedene Anwendungsbereiche ähnliche Probleme gelöst (Jmour et al., 2018). Dementsprechend stellt sich die Frage, ob vortrainierte künstliche neuronale Netze geeignet sind, um das Aussortieren der Sonagramme zu übernehmen. Dafür soll in dieser Arbeit erforscht werden, inwiefern sich vortrainierte künstliche neuronale Netze für die automatische Klassifizierung von Sonagrammen bei der Auswertung von erfassten Fledermausrufen mittels UAS eignen.

1.2 Vorgehensweise

Um zu untersuchen, inwiefern sich vortrainierte künstliche neuronale Netze für die beschriebene Problematik eignen, wird zunächst im zweiten Kapitel mithilfe wissenschaftlicher Veröffentlichungen herausgearbeitet, welche Methoden in der Lage sind einzelne Sonagramme zu verarbeiten und die Unterschiede zwischen Fledermausruf und Störsignal erlernen zu können. Darauf aufbauend wird dargestellt, inwieweit sich vortrainierte künstliche neuronale Netze für diesen Anwendungsfall besser oder schlechter eignen als alternative Methoden. Zudem werden Metriken aufgezeigt, mithilfe derer die Eignung der Netze gemessen werden kann. Abschließend wird anhand von Literatur, die sich bereits mit Fledermausrufen und neuronalen Netze auseinandergesetzt hat, erarbeitet, welche Erkenntnisse noch fehlen, um die Forschungsfrage zu beantworten.

In dieser Arbeit wird ein Experiment durchgeführt, um die Eignung von vortrainierten künstlichen neuronalen Netzen für die automatische Klassifizierung von Sonagrammen bei der Auswertung von erfassten Fledermausrufen mittels UAS zu untersuchen. Dabei werden vier unterschiedlichen vortrainierten künstlichen neuronalen Netzen mittels Transfer Learning beigebracht, zwischen Störsignalen und Fledermausrufen zu unterscheiden. Als Trainingsdaten werden Aufnahmen des Projektes Drones4Bats von der HAW Hamburg verwendet, die bereits in gesonderten Feldversuchen entstanden sind. Drones4Bats ist ein Forschungsprojekt, das sich mit der Problematik aus dem Kapitel 1.1 auseinandersetzt und als Lösungsansatz UAS entwickelt. Die erhobenen Daten aus diesem Projekt stehen dem Experiment dieser Thesis zur Verfügung und werden den Anforderungen der neuronalen Netze entsprechend vorbereitet. Die Ergebnisse des Experimentes werden im vierten Kapitel präsentiert. Anhand dieser Ergebnisse wird

im fünften Kapitel eine Einschätzung für die Eignung der neuronalen Netze erarbeitet. Des Weiteren wird auf zusätzliche Fragestellungen, die sich durch den Verlauf des Experiments bzw. der Interpretation der Ergebnisse ergeben, eingegangen und Empfehlungen für die weitere Forschung werden ausgesprochen. Am Ende werden die wesentlichen Aussagen aus der Literatur und des Experimentes zusammengefasst.

2 Grundlagen

In diesem Kapitel wird der aktuelle Forschungsstand zur computergestützten Bildklassifikation und insbesondere von vortrainierten künstlichen neuronalen Netzen dargestellt. Dabei besteht das Ziel, die Eignung von diesen neuronalen Netzen anhand der Literatur zu erarbeiten und herauszustellen, welche Aspekte zur Beantwortung der Forschungsfrage in der Literatur nicht betrachtet worden sind. Die technischen Grundlagen sollen den oder die Leser*in bemächtigen das Experiment sowie die Ergebnisse und deren Interpretation nachvollziehen zu können.

2.1 Bildklassifikation

Die Untersuchungen in dieser Thesis stützen sich zum einen auf verwandte Literatur und zum anderen auf akustische Aufnahmen aus dem in Kapitel 1.2 vorgestellten Projekt Drones4Bats. Diese Aufnahmen stammen aus Feldversuchen, die den Zweck verfolgten, mithilfe von UAS Fledermausrufe aufzunehmen. Als Aufnahmegerät diente ein batCoder der ecoObs GmbH. Die Aufnahmen des batCoders wurden durch die Analysesoftware bcAdmin der ecoObs GmbH als Sonagramme bereitgestellt. Bei einem Sonagramm handelt es sich um eine „grafische Darstellung einer akustischen Struktur“ (Dudenredaktion, 2022, "Bedeutung" Abschnitt). Die einzelnen Frequenzen eines akustischen Signals werden bei einem Sonagram mithilfe der Kurzzeit-Fourier-Transformation im zeitlichen Verlauf dargestellt (Kiencke, 2009).

Um Sonagramme automatisch einordnen zu können, werden computergestützte Bildklassifizierungsmethoden verwendet (Chiao et al., 2019; Dominguez-Morales et al., 2018; Zeng et al., 2019). Bildklassifizierungsmethoden sind Algorithmen, die dem Zweck dienen einem Bild eine Klasse zuzuordnen. „Die Klasse bezeichnet typischerweise das Objekt, das im Bild angezeigt wird“ (Steinwendner & Schwaiger, 2019, S. 370). Im Rahmen dieser Thesis wäre die Klasse ein Fledermausruf oder ein Störsignal. Das Hauptziel von Bildklassifizierungsmethoden ist die exakte Identifizierung der in einem Bild vorhandenen Merkmale (Nath et al., 2014). Die Unterscheidung dieser Bildklassifizierungsmethoden richtet sich nach der Art und Weise, wie die Mustererkennung erlernt wird. Steinwendner & Schwaiger (2019) unterschieden zwischen überwachtem, unüberwachtem, und verstärkendem Lernen. Für die Klassifikation von Bildern wird vor allem das überwachte und unüberwachte Lernen eingesetzt (Schmarje et al., 2021), weshalb an dieser Stelle nicht weiter auf das verstärkende Lernen eingegangen wird.

Beim überwachten Lernen erhält ein Algorithmus gekennzeichnete Daten. Das bedeutet, dass für jede Eingabe eine passende Ausgabe übergeben wird. Der Algorithmus lernt daraufhin Muster, die sich aus den Merkmalen der Ein- und Ausgabe ergeben. Bildklassifizierungsmethoden, die überwacht Lernen, sind zum Beispiel Decision Trees, Support Vector Machines oder Naive Bayers Classifier (Alzubi et al., 2018). Das unüberwachte Lernen zeichnet sich hingegen dadurch aus, dass ein Algorithmus keine passenden Ausgaben erhält, sondern nur Eingaben, und sich mit diesen selbst trainiert. Das Ziel ist dabei,

dass der Algorithmus Muster in den Daten erkennt und anhand dieser die Daten eigenständig in verschiedene Gruppen einteilt. Beliebte Algorithmen dieser Kategorie sind unter anderem K-Means Clustering, K-Nearest-Neighbours oder Hierarchical Clustering (Glielmo et al., 2021).

Es existieren viele wissenschaftliche Veröffentlichungen, die sich in verschiedenen Anwendungsfällen mit Bildklassifizierungsmethoden auseinandersetzen. Gavali und Banu (2019) identifizieren Neural Networks, Support Vector Machines, Fuzzy Logic und Genetic Algorithms als vielversprechende Methoden für die Bildklassifizierung. Diese Methoden werden vor allem im medizinischen Bereich eingesetzt, da eine korrekte Klassifikation von Krankheitsbildern wesentlich für klinische Diagnosen und darauf basierenden Krankheitsbehandlungen sein kann. Ein weit verbreiteter Anwendungsfall ist zum Beispiel das automatische Auswerten von Bildern des menschlichen Gehirns, die durch Magnetresonanztomographie (MRT) entstanden sind. Für derartige Daten wurde durch Verwendung eines Support Vector Machine Algorithmus eine Klassifikationsgenauigkeit von 65% erreicht (Abdullah et al., 2011). Das bedeutet, dass der Support Vector Machine Algorithmus 65% der gesamten Datenmenge zur korrekten Klasse zugewiesen hat. Mit einem Fuzzy C-Means Algorithmus haben Ahmed et al. (2002) je nach Datensatz eine Klassifikationsgenauigkeit zwischen 78,9% und 98,92% erreicht. Mit einem Ansatz, der ein vielschichtiges neuronales Netzwerk beinhaltete, wurde eine Klassifikationsgenauigkeit von 96,97% erreicht (Mohsen et al., 2018).

In den letzten Jahren sind immer mehr wissenschaftliche Artikel erschienen, die insbesondere künstliche neuronale Netze für die Klassifikation von medizinischen Bildern untersucht haben (Zhang et al., 2022). Vor allem sogenannte Convolutional Neural Networks (CNNs) stechen dabei hervor. Qing Li et al. (2014) veröffentlichten den Artikel „Medical Image Classification with Convolutional Neural Network“, in dem die Vorteile von CNNs für die Klassifikation von Lungenbildern aufgezeigt wurden. Wie aus einer Veröffentlichung von T. Fujioka et al. (2020) hervorgeht, werden auch im Bereich der Brustultraschallbilder CNNs verwendet. Zudem haben auch Aykanat et al. (2017) verschiedene CNNs mit einem Support Vector Machine Algorithmus bezüglich der Klassifikationsgenauigkeit von Lungen Bildern verglichen. Das Resultat war, dass die CNNs eine höhere Klassifikationsgenauigkeit besaßen als der Support Vector Machine Algorithmus. Daneben haben Alnajjar und Abu-Naser (2022) CNNs für die Klassifikation von Herzgeräuschen benutzt, um Herz-Kreislauf-Störungen diagnostizieren zu können. Doch nicht nur im medizinischen Bereich werden CNNs eingesetzt. Boddapati et al. (2017) haben sie in Verbindung mit Recurrent Neural Networks benutzt um Umweltgeräusche zu klassifizieren. Müller et al. (2021) konnten akustische Anomalien von Maschinen durch neuronale Netze detektieren und eine Klassifikation von Schnarchgeräuschen mittels Convolutional Neural Networks ist ebenfalls möglich (Amiriparian et al., 2017).

2.2 Convolutional Neural Networks

Convolutional Neural Networks sind mehrschichtige, vorwärts gerichtete, tiefe neuronale Netze und gehören somit zum Bereich des maschinellen Lernens. Ein neuronales Netz ist ein Algorithmus, der aus einer Verkettung von vielen sogenannten mathematischen Neuronen besteht. Das Modell eines mathematischen Neurons (Scheid & Vogl, 2021) ist in der folgenden Abbildung dargestellt:

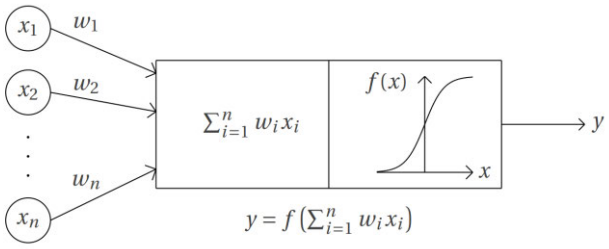

Abbildung 2-1: Modell eines mathematischen Neurons
Quelle: Scheid & Vogl 2021, S. 331

Für die Eingabe (Inputs), die das Neuron erhält, steht $x_i, i = 1, \dots, n$. Diese werden mit den Gewichten $w_i \in \mathbb{R}, i = 1, \dots, n$ multipliziert und aufsummiert (Scheid & Vogl, 2021). Die Summe wird anschließend in die Aktivierungsfunktion f eingesetzt. Wenn mit dem Ergebnis aus f ein bestimmter Schwellenwert überschritten worden ist, wird nur dann der Ausgabenwert γ (Output) ausgegeben (Scheid & Vogl, 2021).

Mehrere Neuronen lassen sich in Schichten (Layer) zusammenfassen (Aggarwal, 2018). Abbildung 2-2 zeigt ein einfaches neuronales Netz mit Input Layer, Hidden Layer und Output Layer. Die einzelnen Kreise stehen jeweils für ein Neuron. Außerdem können mehrere Hidden Layer als Sequenz hintereinander gereiht werden. Die Ausgabe eines Neurons ist dann die Eingabe eines anderen Neurons. Derart können komplexe mathematische Operationen in kleine Aufgaben geteilt werden.

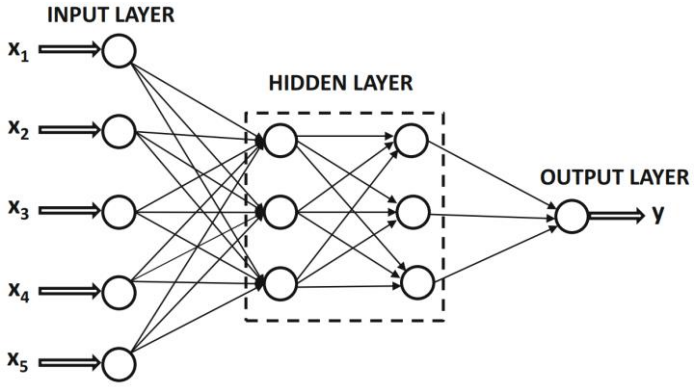

Abbildung 2-2: Modell eines einfachen neuronalen Netzes
Quelle: Aggarwal 2018, S. 14

Je nach Aufgabenbereich eines Netzes unterscheiden sich die Anzahl und die Funktionen der Schichten. Der Aufgabenbereich von Convolutional Neural Networks ist das Verarbeiten von Daten, die eine Gitter-Struktur besitzen (Aggarwal, 2018). Ein digitales Bild ist ein Beispiel derartiger Daten, da es aus

einer mehrdimensionalen Matrix besteht. Die spezifische Aneinanderreihung von verschiedenen Schichten ermöglicht den CNNs die Verarbeitung dieser Matrixstrukturen.

Eine bekannte Convolutional Neural Network Architektur ist das LeNet-5, das von Lecun et al. (1998) für die Klassifizierung von handgeschrieben Symbolen entworfen wurde. Es besteht neben einem Input Layer aus den namensgebenden Convolutional Layern, die Merkmale aus Bildern mithilfe von Filtern extrahieren können. Sie erstellen dazu sogenannte Feature Maps. Pooling (Subsampling) Layer reduzieren die Auflösung dieser, damit der Rechenaufwand verringert wird. Am Ende werden für die Klassifikation alle Informationen durch sogenannte Fully Connected Layer zusammengefasst und an einen Output Layer übergeben, der die finale Klassifikation übernimmt (Lecun et al., 1998).

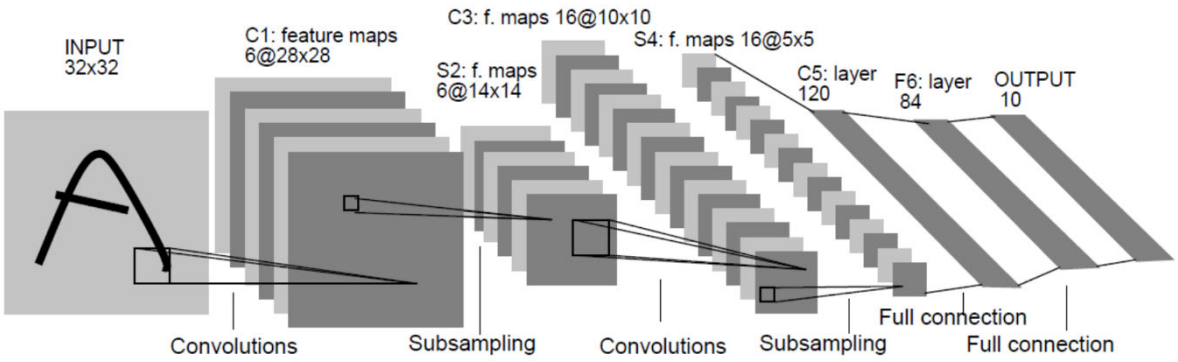

Abbildung 2-3: Architektur des LeNet-5
Quelle: Lecun et al. 1998, S. 7

Die CNN Architekturen haben sich seit 1998 weiterentwickelt und besitzen heutzutage in der Regel mehr aufeinanderfolgende Schichten als die LeNet-5 Architektur. Das Zusammenspiel zwischen Convolutional Layern, Pooling Layern und Fully Connected Layern bestimmt aber immer noch die Charakteristik eines CNN. Durch diesen speziellen Aufbau sind CNNs in der Lage, Merkmale aus Bildern zu extrahieren und in Klassen einzuordnen. Um zielgerichtete Ergebnisse zu erhalten, müssen diese Netze jedoch vor dem Einsatz trainiert werden. Mithilfe eines Datensatzes, der Bilder aller Klassen enthält, lernt ein Netz, die Bilder zu unterscheiden und einer Klasse zuzuordnen.

Den allgemeinen Trainingsprozess für mehrschichtige neuronale Netze haben Paaß & Hecker (2021) detailliert beschrieben. Für Sie besteht der Ablauf des Lernens aus drei Schritten, wie in der folgenden Abbildung zu erkennen ist:

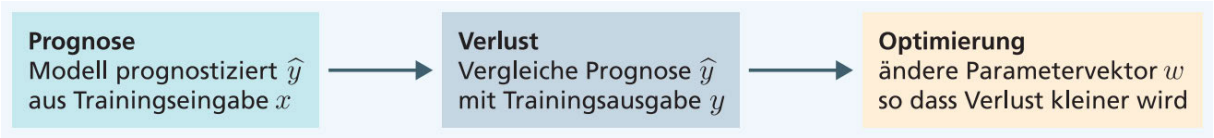

Abbildung 2-4: Grundlegende Schritte für das Training von maschinellen Lernverfahren
Quelle: Paaß & Hecker 2021, S. 53

Für ein Modell, das zwischen zwei Klassen unterscheiden soll (auch binäre Klassifikation genannt) bedeutet das Folgendes:

Als erstes erhält das Modell ein Bild als Trainingseingabe x und entwickelt anhand der gewichteten Parameter w eine Prognose $\hat{y}$, die angibt, inwiefern x zur ersten oder zweiten Klasse gehört. In einem zweiten Schritt wird ein Verlust berechnet, indem der Prognosewert mit der tatsächlichen Klasse y verglichen wird. Der Verlust bestimmt sich somit aus der Differenz zwischen $\hat{y}$ und y. Im Normalfall ist dieser Wert nach dem ersten Durchlauf sehr hoch, weil initial die Gewichte eines Modells zufällig sind. Im dritten Schritt werden beginnend mit der Ausgabenschicht bis hin zur Eingabeschicht alle Gewichte geändert, sodass beim nächsten Durchlauf der Verlust kleiner wird. Dieser Prozess wird mithilfe des Gradientenabstiegs realisiert (Böttcher et al., 2019). Ein Trainingsdurchlauf wird auch Epoche genannt. Das Training für ein Modell besteht zumeist aus mehreren Epochen und verfolgt das Ziel, nach jeder einzelnen Epoche den Verlustwert zu minimieren (Scheid & Vogl, 2021).

Ob künstliche neuronale Netze ein Bild korrekt klassifizieren, hängt also von den Gewichten ab, die während des Trainings auf den Datensatz abgestimmt werden und somit auch vom Datensatz an sich. Als Indikator, ob die Gewichte einen geeigneten Wert besitzen, fungiert der berechnete Verlust jeder Trainingsepoche. Dieser Parameter ist somit elementar für die Bewertung eines neuronalen Netzes. Zusammenfassend lässt sich sagen, dass Convolutional Neural Networks höhere Klassifikationsgenauigkeiten als andere Bildklassifizierungsmethoden aufweisen, sofern eine korrekte Trainingsstrategie angewendet wurde.

2.3 Vision Transformer

Für einen anderen Aufgabenbereich von künstlicher Intelligenz, dem Natural Language Processing (NLP), hat sich die Transformer Architektur durchgesetzt (Brown et al., 2020; Vaswani et al., 2017). Diese Architektur haben Dosovitskiy et al. (2020) mit einer großen Anzahl an Bildern trainiert um die Vision Transformer (ViT) Architektur zu entwickeln. Zusätzlich haben Sie in einem Experiment ViT und CNNs die gleichen Klassifizierungsprobleme lösen lassen und miteinander verglichen. Im Ergebnis haben ViT Resultate erreicht, die ähnlich gut denen moderner Convolutional Neural Networks oder sogar besser waren. Seither gelten ViT als relevante Alternative zu CNNs. Es hat sich herausgestellt, dass ViT vor allem für große Datenmengen besser funktionieren. Für kleine Datenmengen eignen sich entsprechend eher CNNs (Han et al., 2022).

Der Unterschied von ViT zu CNNs ist, dass ViT ein Bild in sogenannte Patches von 16x16 Pixeln aufteilen und parallel verarbeiten (Dosovitskiy et al., 2020). Diese werden durch eine lineare Projektion zu Vektoren transformiert und dienen als Eingabe für den Transformer Encoder, der in der Veröffentlichung von Vaswani et al. (2017) vorgestellt wurde. Die letzte Schicht des Vision Transformers ist für die endgültige Klassifizierung zuständig, wie es auch bei den CNNs der Falls ist.

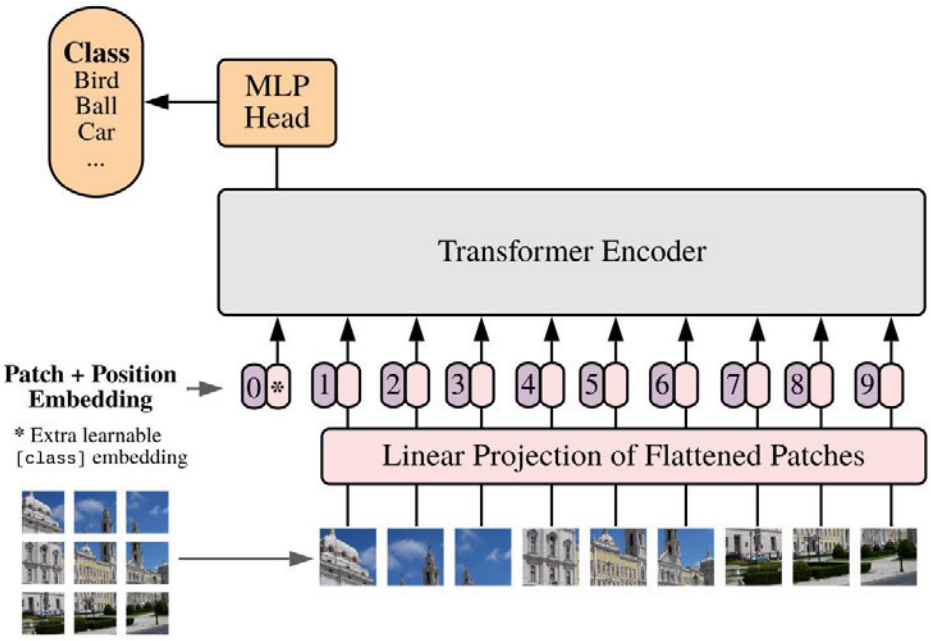

Abbildung 2-5: Übersicht des Vision Transformer Modells
Quelle: Dosovitskiy et al. 2020, S. 3

Während CNNs das Eingangsbild Stück für Stück abtasten und die Informationen des Bildes so immer weiter bündeln, können Transformer Architekturen von Anfang an auf alle Informationen des Bildes zugreifen. Der Nachteil von Transformer Architekturen ist jedoch, dass für das Training eine große Menge an Daten und Rechenleistung benötigt wird (Han et al., 2022). Eine Möglichkeit diesen Nachteil zu umgehen, könnten Architekturen bieten, die die Vorteile von CNN und ViT vereinen (Guo et al., 2022).

Die Webseite paperswithcode.com führt eine Rangliste von Architekturen, die die besten Klassifikationsgenauigkeiten auf dem ImageNet Datensatz besitzen (Image Classification on ImageNet, 2022). ImageNet ist eine Datenbank mit aktuell 1.281.167 Trainingsbildern und 150.000 Bildern zur Auswertung, die von der Princeton und Standford Universität für Forschungszwecke veröffentlicht worden ist (ImageNet, o. D.). Zur Abgabe dieser Thesis stellen neun von den zehn Architekturen, die die höchste Klassifikationsgenauigkeit auf dem ImageNet Datensatz besitzen, Transformer Architekturen dar (Image Classification on ImageNet, 2022). Die insgesamt höchste Klassifikationsgenauigkeit von 91% hat die Transformer Architektur CoCa, welche von Yu et al. (2022) entworfen wurde.

2.4 Vortrainierte Modelle und Transfer Learning

Mit der Weiterentwicklung von mehrschichtigen neuronalen Netzen ist die Anzahl der Gewichte - nachfolgend Parameter genannt - stetig gewachsen (Qiu et al., 2020). Damit wird der Trainingsprozess immer komplexer und rechenintensiver. Zudem verstärken neue Arten der Informationsverarbeitung diesen Effekt, wie es zum Beispiel bei der Einführung des Vision Transformers der Fall war (Han et al., 2022). Das führt zu einer immer höheren Menge an benötigten Daten für das Training. Da in der Regel für das

eigene Klassifizierungsproblem nicht so viele Daten zur Verfügung stehen, können vortrainierte künstliche neuronale Netze benutzt werden, die mittels Transfer Learning an ein spezifisches Problem angepasst werden. Beim Transfer Learning wird nur die letzte Schicht oder eine geringe Anzahl an Schichten des vortrainierten Modells neu trainiert. Dadurch wird außerdem der Trainingsprozess beschleunigt.

Vortrainierte Modelle sind Architekturen, die mit einem großen Datensatz, wie dem JFT oder ImageNet Datensatz, trainiert worden sind und somit bereits optimierte Parameter besitzen. Neben den bereits Vorgestellten sind MNIST, CIFAR-10 und STL-10 weitere bekannte Datensätze für die Bildklassifikation. Der Vorteil von vortrainierten Modellen ist, dass sie durch das Training angepasste Parameter besitzen und dadurch schneller für ein Klassifizierungsproblem trainiert werden können als mit zufälligen Werten bei einem neu initialisierten Modell. Darüber hinaus zeigen Experimente von Sahu et al. (2020), dass die Klassifikationsgenauigkeit für eine binäre Klassifikation von einem gänzlich neu trainierten Modell etwa 28% geringer ist, als die eines vortrainierten Modells. Für sehr spezielle Anforderungen an ein neuronales Netz könnte sich der Einsatz von vortrainierten Modellen jedoch nachteilig auswirken, wenn zum Beispiel die Parameter durch den Datensatz unvorteilhaft angepasst worden sind.

2.5 Bewertung von künstlichen neuronalen Netzen

Beim Trainieren eines künstlichen neuronalen Netzes kann der Effekt der sogenannten Überanpassung entstehen. Überanpassung bedeutet, dass das Netz den Trainingsdatensatz memoriert. Die Parameter des Netzes passen sich dementsprechend optimal auf den Datensatz an. Wenn das Netz anschließend jedoch Bilder erhält, die dem Netz unbekannt sind, fällt die Klassifikationsgenauigkeit stark ab. Das Netz kann dann neue Merkmale, die in dem Trainingsdatensatz nicht vorhanden waren, nicht mehr eindeutig zuordnen. Um dies zu verhindern, wird das Netz nach jeder Trainingsepoche mit einem Validierungsdatensatz getestet. Steigt der Verlust (Fehler) mit diesem Datensatz, bedeutet das, dass das Netz anfängt auswendig zu lernen und dass das Training beim globalen Minimum des Validierungsfehlers beendet werden sollte. Diese Methode wird auch Early Stopping genannt (Scheid & Vogl, 2021).

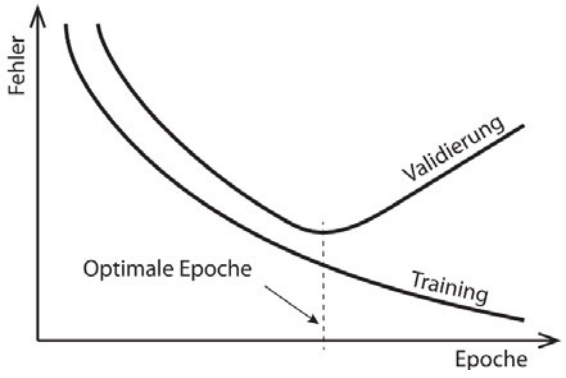

Abbildung 2-6: Early Stopping
Quelle: Scheid & Vogl 2021, S. 342

Um sowohl eine Überanpassung zu verhindern als auch das Training schneller konvergieren zu lassen, können Werte geändert werden, die für den Lernprozess verantwortlich sind. Sie werden auch Hyperparameter genannt. Dazu zählen zum Beispiel die Anzahl der Epochen, die Lerngeschwindigkeit, welche durch die Lernrate gesteuert wird, sowie die sogenannte Batch Size (Chargengröße), also die Anzahl an Bildern, nach denen die Gewichte angepasst werden. Die Lernrate ist eine wichtige Variable für den Trainingsprozess, weil sie definiert, wie stark der Einfluss der Verlustwerte auf die neuen Gewichte ist. Durch die Betrachtung des Trainings- sowie Validierungsverlustes lässt sich einschätzen, inwiefern das künstliche neuronale Netz aus den verfügbaren Daten lernt. Der Verlauf des Trainings- sowie Validierungsverlustes in der folgenden Abbildung zeigt beispielhaft, wie ein ideales Lernverhalten eines neuronalen Netzes aussieht:

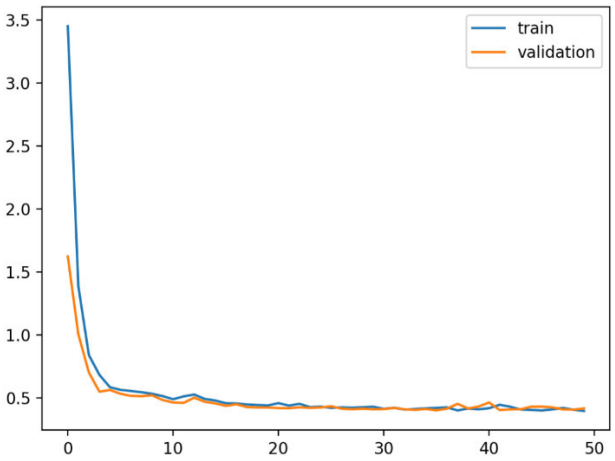

Abbildung 2-7: Verlust beim Trainieren eines künstlichen neuronalen Netzes
Quelle: Brownlee 2019

Die vertikale Achse stellt den Wert für den Verlust dar und die horizontale Achse die Anzahl der Epochen. Zu Beginn nehmen sowohl der Trainings- also auch der Validierungsverlust stark ab und nähern sich mit fortschreitender Epoche einem Wert an. Dabei ist der Validierungsverlust meistens knapp unter dem des Trainings. Doch aus dieser Darstellung wird nicht ersichtlich wie viele Bilder das Netz korrekt klassifiziert hat. Um dafür einen Überblick dafür zu bekommen, eignet sich eine Darstellung als Konfusionsmatrix.

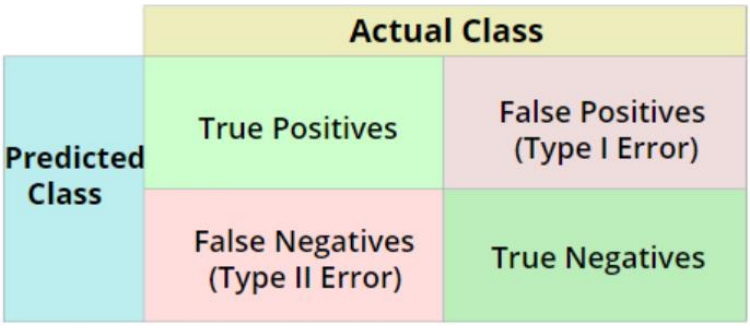

Abbildung 2-8: Konfusionsmatrix für binäre Klassifizierung
Quelle: Oppermann 2021

Mit der Konfusionsmatrix wird die tatsächliche Anzahl der klassifizierten Bilder aufgezeigt. An ihr kann abgelesen werden, wie viele Bilder aus einer Klasse vom Netz korrekt klassifiziert worden sind (True Positives und True Negatives) oder einer falschen Klasse zugeordnet wurden (False Positves und False Negatives). Anhand dieser Werte lassen sich Metriken ableiten, mit denen die Ergebnisse von künstlichen neuronalen Netzen bewertet werden können. Die Klassifikationsgenauigkeit (Accuracy) gibt den Anteil der korrekt klassifizierten Bilder bezogen dem gesamten Datensatz an.

$$Klassifikationsgenauigkeit = \frac{TP + TN}{TP + TN + FP + FN}$$

Diese Metrik wird für die Bewertung der generellen Klassifikationsgenauigkeit benutzt. Prioritäten für eine bestimme Klasse lassen sich mit dieser Metrik nicht darstellen und es werden zudem keine Datensätze berücksichtigt, die eine unausgewogene Verteilung von Bildern für eine Klasse haben (Chicco & Jurman, 2020). Diese Eigenschaften lassen sich mithilfe der Trefferquote und Präzision besser darstellen. Die Trefferquote (Recall) gibt den Anteil der korrekt klassifizierten Bilder für eine bestimmte Klasse bezogen auf die Menge der tatsächlichen Klasse an. Das ist hilfreich, falls das Auffinden einer bestimmten Klasse eine hohe Priorität hat, wie zum Beispiel beim Identifizieren von lebensgefährlichen Krankheiten. Die Präzision (Precision) gibt den Anteil der korrekt klassifizierten Bilder für eine bestimmte Klasse bezogen auf die Menge aller als dieser Klasse eingeordneten Bilder an.

$$Trefferquote = \frac{TP}{TP + FN}$$

$$Präzision = \frac{TP}{TP + FP}$$

Der Nachteil der Trefferquote und der Präzision ist, dass die Anzahl der True Negatives nicht berücksichtigt wird, denn die Summe der True Negatives könnte jeglichen Wert annehmen und die Metriken würden sich nicht verändern. Um eine allgemeine Aussage über die Qualität von binären Klassifizierern treffen zu können, eignet sich vielmehr der Matthews Correlation Coefficient (MCC), der von Matthews (1975) veröffentlicht wurde. Speziell für die binäre Klassifikation besitzt diese Metrik Vorteile gegenüber der generellen Klassifikationsgenauigkeit und der F1 Metrik, die auch im Bereich des maschinellen Lernens verwendet wird (Chicco & Jurman, 2020). MCC bezieht alle Ergebnisse des Klassifizierers ein und nimmt einen Wert zwischen 1 und -1 an. Der Wert 1 steht dann für ein perfektes Klassifizieren ohne Fehler und -1 steht dafür, dass alle Daten der falschen Klasse zugeordnet worden sind (Chicco & Jurman, 2020).

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

2.6 Verwandte Arbeiten

Es wurden einige wissenschaftliche Veröffentlichungen dargestellt, die Convolutional Neural Networks und Transformer Architekturen erfolgreich zur Klassifizierung in unterschiedlichen Bereichen angewendet haben, wie zum Beispiel im medizinischen Bereich oder in der Spracherkennung. Im Bereich der Klassifizierung von Fledermausrufen haben Schwab et al. (2022) CNN Architekturen mit einem großen Datensatz aus Sonagrammen trainiert um Fledermausarten anhand ihrer Rufe automatisiert bestimmen zu können. Dabei wurden die Sonagramme zuvor bearbeitet, sodass die relevanten Informationen zur Bestimmung eines Rufes oder Störsignales besser für die CNNs zu lesen waren. Danach wurden die Sonagramme in Fledermausruf oder Störsignal eingeteilt, damit im Anschluss die Fledermausart durch die Charakteristik des Rufes bestimmt werden kann. Die Einteilung in Ruf oder Störsignal wird zwar auch in der vorliegenden Arbeit verwendet, jedoch sind die Störsignale durch UAS entstanden und nicht durch Insekten oder technische Impulse von elektronischen Weidezäunen (Schwab et al., 2022). Mit dieser Klassifizierungsmethode haben sie erreicht, dass alle Störsignale herausgefiltert worden sind und bei der Bestimmung der Fledermausart haben sie zudem Klassifikationsgenauigkeiten von bis zu 96,13% erreicht. Dafür haben Schwab et al. (2022) drei CNN Architekturen selbst entwickelt und zwei modifizierte Modelle der Inception-v1, sowie ReNet-50 Architektur benutzt.

Aodha et al. (2018) untersuchten einen ähnlichen Ansatz, der Fledermausrufe in Stadt- und Naturgeräuschen mittels Bildklassifizierungsmethoden identifiziert hat. Dazu wurden unterschiedliche Bildklassifizierungsmethoden angewendet und anschließend die Trefferquote und Präzision untereinander verglichen. Unter anderen wurden ein Random Forest Algorithmus mit zwei unterschiedlichen CNN Architekturen verglichen. Im Ergebnis haben die CNNs ebenfalls die besten Ergebnisse hervorgebracht. Jennings et al. (2008) haben Unterschiede zwischen Menschen und künstlichen neuronalen Netzen dahingehend untersucht, inwiefern sie Fledermausarten anhand ihrer Rufe für die Echoortung erkennen. Dafür haben 26 Experten Fledermausrufe verschiedenen Fledermausarten zugeordnet. Die gleichen Daten wurden anschließend von künstlichen neuronalen Netzen (Parsons & Jones, 2000) zugeordnet. Das Ergebnis war, dass die künstlichen neuronalen Netze höhere Klassifikationsgenauigkeiten besaßen als 75% der Experten. Manche der 26 Experten waren individuell zwar wesentlich besser in der Zuordnung der korrekten Fledermausart, aber die meisten Experten konnten weniger genau klassifizieren als die künstlichen neuronalen Netze.

Zualkernan et al. (2020) haben Convolutional Neural Networks benutzt um zu untersuchen, welche Visualisierungsart sich am besten für die Klassifikation von Fledermausrufen eignet. Sonagramme, entstanden durch eine Kurzzeit-Fourier-Transformation, wurden mit Visualisierungen durch Mel-Scaled Filter banks (MFSB) und Mel-Frequency Cepstral Coefficientes (MFCC) verglichen. MFSB erzielten mit 97,51% die höchste Klassifikationsgenauigkeit, aber Kurzzeit-Fourier-Transformationen waren mit Genauigkeiten zwischen 91,72% und 96,02% ähnlich erfolgreich. Die Klassifikationsgenauigkeit wurde dabei wieder durch die korrekte Zuordnung von Rufen zur jeweiligen Fledermausart gemessen.

Die verwandten Arbeiten haben ausschließlich die Klassifizierung der Fledermausart als übergeordnetes Ziel. Schwab et al. (2022) und Aodha et al. (2018) nutzten zusätzlich CNNs um die Rufe von Störsignalen durch Insekten, technische Impulse oder Stadtgeräusche zu unterscheiden. Die vorliegende Arbeit setzt den Schwerpunkt jedoch auf die Unterscheidung von Fledermausrufen und Störsignalen, die durch UAS hervorgerufen werden. Des Weiteren wurden vorwiegend CNNs genutzt. Es wurden aber keine Transformer Architekturen in die Untersuchungen aufgenommen, obwohl Sie, wie in Kapitel 2.3 beschrieben, in den letzten Jahren bessere Klassifikationsgenauigkeiten aufweisen konnten als CNNs. Ebenso wurden alle eingesetzten künstlichen neuronalen Netze von Grund auf neu trainiert. Keine der benannten Arbeiten hat vortrainierte Modelle eingesetzt.

Dementsprechend werden für die vorliegende Arbeit vortrainierte Convolutional Neural Networks und Vision Transformer untersucht. Sie werden prototypisch trainiert und anschließend miteinander verglichen mit dem Ziel zwischen Fledermausrufen und Störsignalen unterscheiden zu können. Dabei werden als Datensatz zufällig ausgewählte Fledermausrufe verschiedener Arten und Störsignale von UAS verwendet. Es ist zudem anzumerken, dass die Störsignale nicht ausschließlich von UAS verursacht sein müssen, sondern auch durch andere Quellen, wie die aus den vorgestellten Arbeiten entstanden sein können. Durch einen Vergleich lassen sich Rückschlüsse für den Einsatz und somit auch der Eignung von vortrainierten künstlichen neuronalen Netzen ziehen.

3 Methodik

Im zweiten Kapitel wurde erklärt, wieso sich vortrainierte CNNs und ViT für die Klassifizierung von Fledermausrufen in Form von Sonagrammen eignen. Darauf aufbauend wird in diesem Kapitel beschrieben, wie die Eignung für eine konkrete Anwendung experimentell bewertet werden kann. Die Durchführung und Auswertung des Experimentes werden in diesem Kapitel detailliert beschrieben, um die Ergebnisse reproduzieren zu können. Dabei orientiert sich die Methodik an Sahu et al. (2020), Sarraf & Tofighi (2017) sowie an der vorgestellten Literatur in Kapitel 2.6.

Um nicht nur theoretisch, sondern auch praktisch die Eignung von vortrainierten Modellen zu untersuchen, werden mehrere Modelle, die ein möglichst weites Spektrum von Architekturen abdecken, mit einem Datensatz, bestehend aus Fledermausrufen und Störsignalen, trainiert und anschließend ausgewertet. Da der Schwerpunkt auf vortrainierten Modellen liegt, wird jeweils nur die Klassifizierungsschicht der Modelle trainiert. Dafür ist der Einsatz einer geeigneten Programmbibliothek unabdingbar. Im Rahmen dieser Arbeit wurde eine relativ leicht erlernbare und gut dokumentierte Programmbibliothek benutzt. Für eventuelle weitere Entwicklungen der Modelle ist es von Vorteil, wenn die Programmbibliothek Open Source ist und das ONNX Format unterstützt, weil dadurch eine große Community um die Programmbibliothek entstehen kann. Zudem müssen vortrainierte Modelle für die Bildklassifizierung in der Bibliothek enthalten sein, um das Training überhaupt durchführen zu können. Aus diesen Gründen wurden TensorFlow, PyTorch, MxNet und Deeplearning4j analysiert und die Merkmale dieser Bibliotheken in Tabelle 1 aufgelistet.

Tabelle 1: Auswahl Programmbibliothek
Quelle: eigene Darstellung

Merkmal	TensorFlow	PyTorch	MxNet	Deeplearning4j
Open Source	Ja	Ja	Ja	Ja
detaillierte Dokumentation	Ja	Ja	Nein	Ja
Programmiersprache	Python	Python	Python	Java
ONNX Format	Ja	Ja	Ja	Nein
vortrainierte Modelle für die Bildklassifizierung	Ja	Ja	Ja	Ja
Komplexität	Sehr hoch	hoch	niedrig	*keine Einschätzung*

Aufgrund dieser Merkmale wurde sich für den Einsatz von PyTorch entschieden, weil es alle Anforderungen erfüllt hat und sich besser für die Entwicklung von kleinen Projekten eignet als TensorFlow. PyTorch wurde extra für die Anwendung von maschinellem Lernen und in der Programmiersprache Python entwickelt. Außerdem lässt sich der Code auf einer GPU ausführen, wodurch die Trainingsdauer verkürzt wird.

PyTorch stellt mehrere vortrainierte Modelle zur Verfügung (*Models and pre-trained weights,* o. D.). In dieser Arbeit werden die Architekturen AlexNet, ResNet50, ConvNeXt_Large und Vision Transformer_1_32 untersucht, die allesamt mit dem ImageNet Datensatz trainiert worden sind. Bei der Auswahl dieser vortrainierten künstlichen neuronalen Netze wurde darauf geachtet, dass sich die Architekturen voneinander unterscheiden, um ein möglichst breites Spektrum zu untersuchen. AlexNet ist ein Convolutiona Neural Network und von Krizhevsky et al. (2012) veröffentlicht worden. Die implementierte Architektur in PyTorch ist eine modifizierte Version von Krizhevsky (2014) und stellt somit die älteste Architektur, sowie die Architektur mit der geringsten Anzahl an aufeinanderfolgenden Schichten in der Programmbibliothek dar. Neben diesem CNN soll mit ConvNeXt auch ein CNN aus PyTorch untersucht werden, das erst vor Kurzem veröffentlicht wurde. ConvNeXt wurde im Januar 2022 veröffentlicht und stellt somit zum Zeitpunkt der Auswahl für das Experiment das modernste vortrainierte neuronale Netz dar, das PyTorch integriert hatte. Überdies weist es die meisten Schichten auf. Als drittes CNN dient ResNet, das von He et al. (2015) vorgestellt wurde. Einige Architekturen, die auf der Grundlage von ResNet aufbauen, haben in den letzten Jahren auf den ImageNet Datensatz die höchsten Klassifikationsgenauigkeiten erreicht (Image Classification on ImageNet, 2022). Aufgrund dieser vielversprechenden Ergebnisse ist ResNet ausgewählt worden. Neben des CNNs wird die in den Grundlagen vorgestellte Vision Transformer Architektur trainiert und mit den anderen Modellen verglichen.

Wie bereits beschrieben, hängt die Fähigkeit dieser Modelle, zwischen Fledermausruf und Störsignal unterscheiden zu lernen, von dem Datensatz und den Hyperparametern ab. Deswegen muss im Idealfall für jedes Modell der Datensatz und die Hyperparameter individuell abgestimmt werden. Dies führt jedoch dazu, dass die Modelle mit anderen Konfigurationen bessere Lernerfolge erzielen könnten. Aus diesem Grund muss ein Rahmen geschaffen werden, der bestimmt, in welcher Weise das Training ablaufen und beendet werden soll. Dieser Rahmen wird im Folgenden vorgestellt.

Für dieses Experiment steht ein kleiner Datensatz zur Verfügung, der in Kapitel 3.2 näher beschrieben wird. Allen vortrainierten Modellen wird mit diesem Datensatz mittels Transfer Learning beigebracht zwischen Fledermausruf und Störsignal zu unterscheiden. Damit die Ergebnisse am Ende vergleichbar und reproduzierbar sind, werden die Modelle außerdem mit den gleichen Hyperparametern trainiert. Aus vor dem eigentlichen Experiment testweise vollzogenen Testdurchläufen hat sich ergeben, dass 80 Epochen für das Training der Modelle mehr als ausreichend sind. Für die Lernrate und die Chargengröße lässt sich nicht bedingungslos ein Wert festlegen, da beide Größen auf den Trainingsprozess jedes Modells unterschiedliche Auswirkungen haben. Deshalb wurden diese Hyperparameter auf das AlexNet Modell optimiert und einheitlich für alle Modelle festgelegt. Die Wahl fiel auf AlexNet, da es durch die älteste Architektur voraussichtlich die schlechtesten Ergebnisse liefern würde und mit der Optimierung der Hyperparameter somit ein Ausgleich für die Ergebnisse geschaffen wird.

Das Training für die Modelle wird analog zu Sahu et al. (2020) sowie Sarraf & Tofighi (2017) optimiert, die verschiedene Convolutional Neural Networks für die Klassifikation Pflanzenkrankheiten und Obstarten verglichen haben. Insofern wurde als Metrik für das Training die Klassifikationsgenauigkeit auf dem Validierungsdatensatz verwendet. Das Training wird für ein Modell beendet, wenn die globale Klassifikationsgenauigkeit erreicht worden ist. Das hat den Vorteil, dass dadurch ein Modell entwickelt wird, welches die meisten Bilder der korrekten Klasse zugeordnet hat. Der Nachteil ist wiederum, dass bestimmte Prioritäten für eine Klasse nicht berücksichtigt werden. Bei der Aufnahme von Fledermausrufen mittels UAS ist auch eine Optimierung des Trainings auf die Trefferquote in Erwägung zu ziehen. Dadurch spezialisiert sich das Modell auf das Aufwinden von Fledermausrufen und bei Grenzentscheidungen, ob sich in einem gefährdeten Gebiet Fledermäuse befinden oder nicht, besitzt dieses Modell mehr Aussagekraft. Wenn das Training abgebrochen wird, sobald die Trefferquote am höchsten ist, ist es möglich, dass sowohl alle Rufe als auch alle Störsignale als Ruf klassifiziert werden. So hat das Modell nicht gelernt Störsignale zu identifizieren und der Einsatz von Bildklassifizierung wäre unbegründet. Eine bessere Variante, eine erhöhte Trefferquote zu erreichen, ist es, die Funktion anzupassen, die den Verlust jeder Trainingsepoche ermittelt. In PyTorch besteht die Möglichkeit, der Verlustfunktion eine eigene Gewichtung zu Gunsten einer Klasse zu vermitteln, wodurch das Optimieren auf die Trefferquote dieser Klasse in einem geringen Maße möglich ist. Da für diese Variante aber Fehloptimierungen beim Trainingsprozess nicht auszuschließen waren, wurde der Ansatz, die Verlustfunktion zugunsten der Trefferquote zu verändern, für diese Arbeit nicht weiterverfolgt.

3.1 Datenaufbereitung

Die Audioaufnahmen aus Feldversuchen von dem Projekt Drones4Bats lassen sich mithilfe der Rufanalyse des Programms bcAdmin von der ecoObs GmbH als Sonagramme visualisieren. Diese Sonagramme sind durch das Standard MacOS Ausschneidewerkzeug abfotografiert worden.

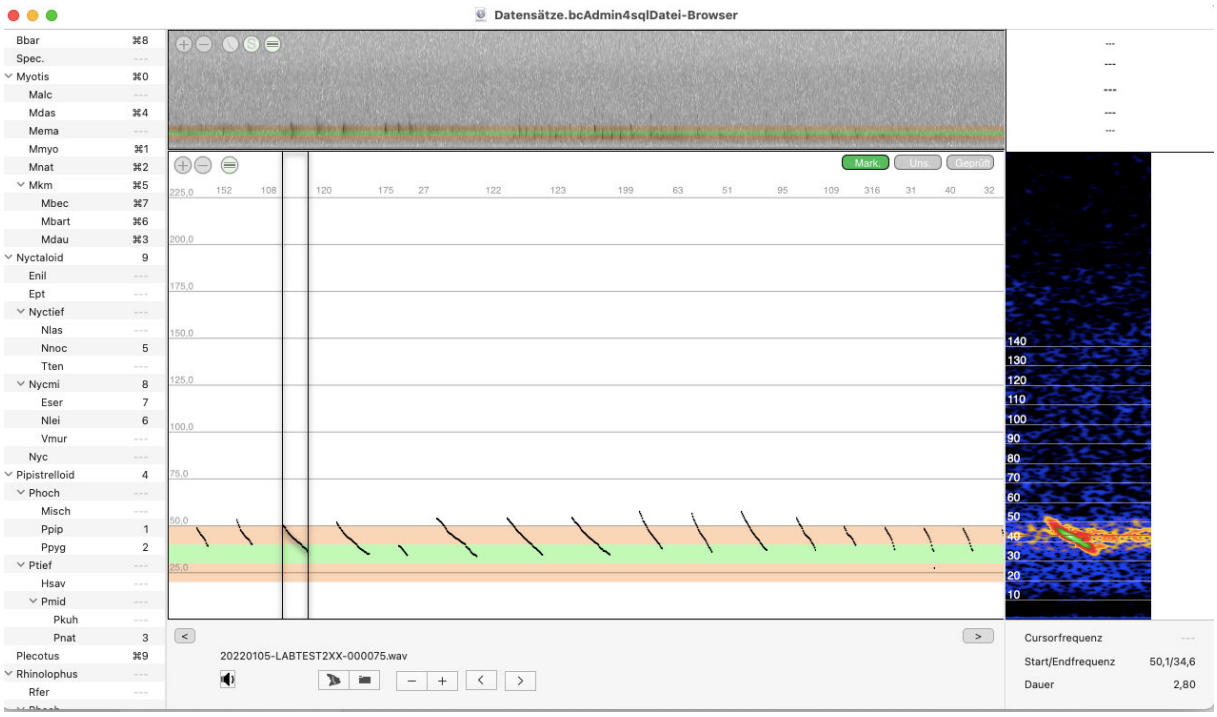

Abbildung 3-1: bcAdmin Rufanalyse
Quelle: bcAdmin

Beim Ausschneiden ist zu beachten, dass irrelevante Informationen nicht mitausgeschnitten werden. Das bedeutet, dass zum einen die gesamte Visualisierung des Rufes und Störsignals in dem Ausschnitt enthalten ist und zum anderen, dass darüber hinaus nur Bestandteile des Sonagramms sichtbar sind. In einem späteren Aufbereitungsschritt wird die Größe der Bilder auf 224x224 Pixel angepasst. Sollte der Ausschnitt dieser Pixelgröße in mindestens einer Dimension um ein Vielfaches der 224 Pixel übersteigen, kann es sein, dass nach der Größenanpassung ein Fledermausruf sehr klein dargestellt wird. Um daraus resultierende negative Folgen für das Training zu vermeiden, sollte daher darauf geachtet werden, den Ausschnitt so klein wie möglich zu wählen.

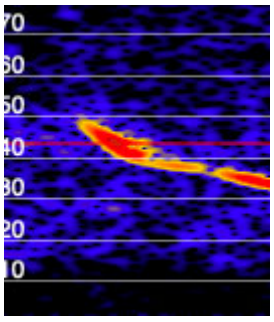

Abbildung 3-2: Sonagramm eines Fledermausrufs
Quelle: bcAdmin

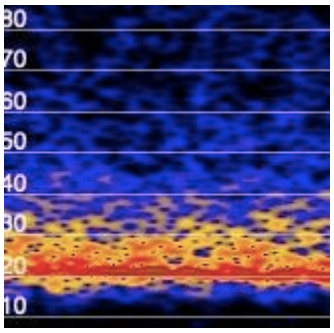

Abbildung 3-3: Sonagramm eines Störsignals
Quelle: bcAdmin

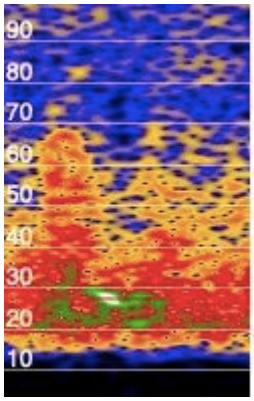

Abbildung 3-4: Sonagramm eins starken Störsignals
Quelle: bcAdmin

Abbildung 3-2 ist ein Beispiel für einen Fledermausruf, der in dem Datensatz enthalten ist. Die geschwungene Form in rötlichen Farben ist in diesem Beispiel ausschlaggebend für die Klassifizierung als Ruf. Abbildung 3-3 zeigt ein Störsignal. Bei sehr vielen Störgeräuschen, wird ein Bild als Störsignal eingeordnet, weil dann die Qualität der Aufnahme so ungenügend gewesen ist, dass die Aufnahme an sich nicht aussagekräftig ist. Abbildung 3-4 ist ein Beispiel für eine Aufnahme vieler Störgeräusche. Diese Aufnahme ist in dem Datensatz als Störsignal klassifiziert worden, obwohl die grüne Form in dem Sonagramm auf einen Fledermausruf hindeuten könnte.

Alle Ausschnitte werden somit je nach Charakteristik als Fledermausruf oder Störsignal klassifiziert und als Datensatz zusammengefasst. Auf diese Weise wurde ein Datensatz erstellt, der insgesamt 800 Bilder beinhaltet. Jeweils 400 für die Klasse Fledermausruf und 400 für die Klasse Störsignal. Damit ist dieser Datensatz gegenüber öffentlichen Datensätzen wie JFT oder ImageNet vergleichsweise klein. Für die Anwendung mit Transfer Learning reicht er jedoch aus.

Aufgrund der geringen Anzahl an Daten wurden in diesem Experiment 60% der Daten für das Training und 40% der Daten für die Evaluierung benutzt. Die 40% werden dann nochmals gleichmäßig aufgeteilt in eine Validierungsmenge und eine Testmenge. Die Validierungsmenge wird für die Bewertung der

Trainingsqualität genutzt und mit der Testmenge wird das trainierte Modell ausgewertet. Für das vorliegende Experiment ergibt das dementsprechend folgende Aufteilung:

Tabelle 2: Aufteilung Datensatz
Quelle: eigene Darstellung

Datensatz	Fledermausrufe	Störsignale	Summe
Training	240	240	480
Validierung	80	80	160
Test	80	80	160

Da die Modelle mit dem ImageNet Datensatz vortrainiert worden sind und dessen Bilder 224x224 Pixel groß sind, müssen alle Bilder des Experimentdatensatzes in diese Größe formatiert werden. Zudem müssen die Pixelinformationen zu einem PyTorch-Tensor formatiert werden, damit die Modelle diese verarbeiten können. Wenn die Pixelintensität der Bilder untereinander angeglichen wird, können die Bilder schneller von den Modellen verarbeitet werden, was wiederum das Training beschleunigt. Das erreicht man, indem der Mittelwert und die Standardabweichung aller Pixelwerte aus dem Datensatz berechnet wird und anschließend jeder Tensor T mit T = (T-Mittelwert) / (Standardabweichung) normalisiert wird.

Die in PyTorch integrierte DataLoader-Klasse übergibt die Bilder zur weiteren Verarbeitung an die Modelle. Sie teilt Datensatz dazu in zufällige Mengen auf, die gleich der Chargengröße sind und überreicht dem Modell diese Mengen nacheinander. Nach jeder abgeteilten Menge werden die Verluste berechnet und die Gewichte neu angepasst. Alle drei Datensätze werden dementsprechend jeweils einer DataLoader-Klasse zugeteilt.

Um das im zweiten Kapitel vorgestellte Problem der Überanpassung zu umgehen, werden für die abgeteilten Mengen des Trainingsdatensatzes jedes Mal zufällig kleine Veränderungen an den Bildern vorgenommen. So sehen die Modelle nach jeder Epoche nicht immer wieder die gleichen Bilder, sondern die zufällig modifizierten. Dadurch wird das Auswendiglernen verzögert. Im Experiment wurden die PyTorch Funktionen RandomHorizontalFlip(), ColorJitter() und RandomGreyscale() dafür verwendet.

3.2 Durchführung

In diesem Kapitel wird die Durchführung des Trainings beschrieben und technisch erklärt. Da das Training mithilfe von PyTorch umgesetzt wurde, ist der Prozess spezifisch auf dieses Framework ausgerichtet. Mit anderen Frameworks kann das Training anders ablaufen.

Als Entwicklungsumgebung wird Google Colab eingesetzt. Dabei handelt es sich um eine cloudbasierte Umgebung, mit der Jupiter-Notebooks ausgeführt werden können. Durch den Zugriff über Internetbrowser ist Google Colab zugänglich. Zudem stellt Google GPUs und TPUs zur Verfügung, auf denen die Modelle trainiert werden können. Damit ist das Training nicht auf eigene technische Ressourcen beschränkt. Zusätzlich lassen sich verschiedene Bibliotheken wie PyTorch oder Numpy integrieren. Die Notebooks werden mit Python geschrieben. Für das Training der vortrainierten Modelle wird eine GPU

als Hardwarebeschleuniger benutzt. Für die Auswertung der Modelle werden sowohl TensorBoard als auch Matplotlib eingesetzt. TensorBoard ist ein Hilfsprogramm zur Visualisierung von Experimenten im Bereich des maschinellen Lernens. Entwickelt wurde es zwar ursprünglich für TensorFlow, jedoch ist es auch für PyTorch und Google Colab einsetzbar. Es wird im Rahmen der vorliegenden Arbeit hauptsächlich für die Optimierung der Hyperparameter eingesetzt, da es das interaktive Anpassen von Parametern ermöglicht. Mithilfe von Matplotlib, einer Programmbibliothek für die Visualisierung mathematischer Darstellungen, werden Diagramme für die Auswertung des Trainingsprozesses erstellt.

Bevor das Training beginnen kann, muss sowohl die Batch Size als auch die Lernrate bestimmt werden. Dafür wird das AlexNet Modell mehrmals mit dem vorgestellten Datensatz neu trainiert. Bei jedem neuen Trainingsdurchlauf wird jeweils die Chargengröße und Lernate geändert. Dabei werden die Werte 8, 16, 32 und 64 für die Batch Size und die Werte 0.1, 0.01, 0.001, 0.0001 sowie 0.00001 für die Lernrate getestet. Zusätzlich wird schrittweise alle sieben Epochen die Lernrate um 0.5 verringert, um eine abfallende Lernrate zu erhalten, wie es von T. Yu & Zhu (2020) empfohlen wird. Mithilfe von TensorBoard wird anschließend der Verlust auf dem Trainingsdatensatz ausgewertet.

Abbildung 3-5: Ergebnisse der Hyperparameteroptimierung
Quelle: TensorBoard

Die Lernrate 0,01 in Verbindung mit der Chargengröße 32 erzielt die besten Ergebnisse. Zum einen nimmt der Verlust schnell ab und nähert sich mit steigender Epochenanzahl einem niedrigen Wert an und zum anderen ist der Wert mit etwa 0,146 der geringste aus allen Messungen. In Abbildung 3-5 sind die Ergebnisse aus TensorBoard dargestellt. Die Graphen sind für eine bessere Lesbarkeit mit dem Smoothing Parameter von TensorBoard bearbeitet. Der graue Graph steht für die Lernrate mit der Batch Size 32. Die anderen Graphen stehen für die gleiche Lernrate mit jeweils unterschiedlichen Werten für die Chargengröße. Damit stehen die Hyperparameter fest. Die Anzahl der Epochen ist 80, die Lernrate 0,01 und die Chargengröße ist 32. Zudem werden die Datensätze entsprechend des Kapitels 3.2 vorbereitet. Zusätzlich müssen die vortrainierten Modelle an den Datensatz angepasst werden. Da die Modelle mit dem ImageNet Datensatz, der 1000 Klassen besitzt, trainiert worden sind, geben die Modelle auch 1000 Werte für die Prognose aus. Die Anzahl der auszugebenen Werte wird dementsprechend mit der

Anzahl der Klassen aus dem individuellen Datensatz gleichgesetzt. Für die vorliegende Arbeit wird die Anzahl auf zwei reduziert. Für das Neutrainieren der Parameter aus der Klassifizierungsschicht werden zudem alle übrigen Parameter für PyTorch derart markiert, dass diese beim Gradientenverfahren ignoriert werden. Nach diesen Vorbereitungen beginnt das eigentliche Training. In Abbildung 3-6 ist der Trainingsprozess für jede Epoche als UML Aktivitätsdiagramm dargestellt. Dieser Prozess wird insgesamt 80-mal durchlaufen.

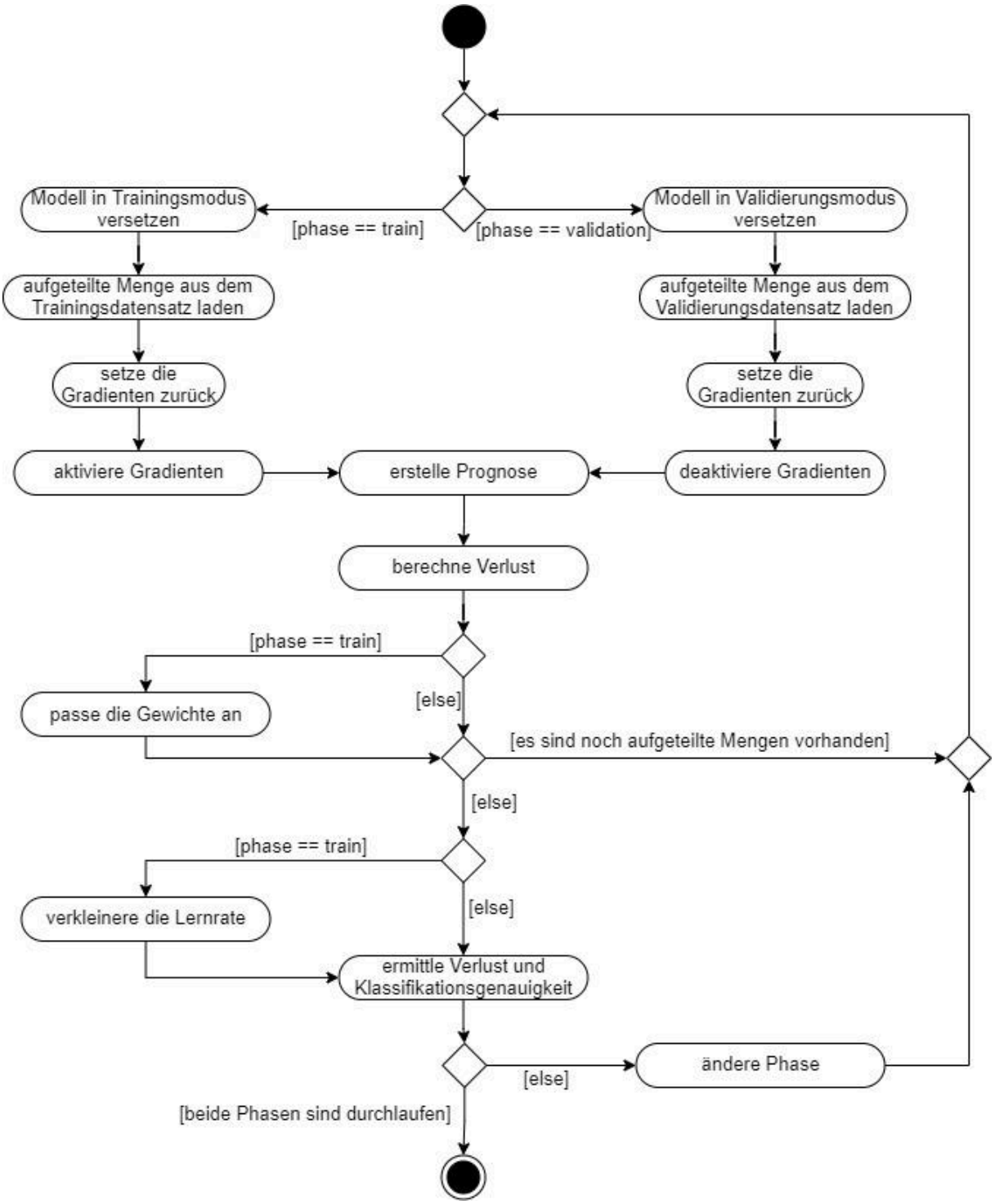

Abbildung 3-6: Aktivitätsdiagramm des Trainingsprozesses pro Epoche (UML Version 2.5.1)
Quelle: eigene Darstellung

Die Trainings- und Validierungsphasen laufen immer nacheinander ab. Gestartet wird mit der Trainingsphase. Dafür wird das Modell durch eine Funktion in den Trainingsmodus versetzt. Dies hat Auswirkungen auf Bestandteile des Modells, die Aufgaben speziell für das Training übernehmen, wie sogenannte Dropout- oder BatchNorm-Schichten.

Als nächstes wird die erste Menge an Bildern mit den Informationen der dazugehörigen Klassen aus dem Trainings-DataLoader geladen. Ebenso werden die Gradienten für das Gradientenverfahren auf null zurückgesetzt und PyTorch mit der Berechnung der Gradienten beauftragt.

Anschließend erstellt das Modell die Prognosen für die Bilder und berechnet den Verlust mithilfe der nn.CrossEntropyLoss Funktion von PyTorch. Diese Funktion ist speziell für Klassifizierungsprobleme entwickelt worden und optimiert das Modell auf alle Klassen gleichermaßen. Falls für den Anwendungszweck des Modells eine Optimierung auf das Erkennen der Fledermausrufe erwünscht ist, stellt der Einsatz der Funktion nn.BCEWithLogitsLoss mit einer geeigneten Besetzung des pos_weight Parameters eine Alternative dar.

Im Anschluss werden die Parameter durch das Gradientenverfahren angepasst. Dafür ist ein Optimierer in PyTorch zuständig. Für das Experiment wird der Adam Optimierer benutzt, weil Adam im Vergleich zu anderen Optimierern die besten Forschungsergebnisse erzielt hat (Kingma & Ba, 2017; Yaqub et al., 2020). Das ist auf die Fähigkeit von Adam zurückzuführen, die Lernrate selbständig anzupassen. Diese Anpassung erfolgt jedoch nur im Rahmen der maximalen Lernrate. Dieser Vorgang wird so lange wiederholt, bis jedes Bild aus dem Datensatz verarbeitet wurde. Danach wird der Klasse, die für die Verringerung der maximalen Lernrate zuständig ist, mitgeteilt, dass eine Trainingsphase durchlaufen worden ist, damit nach jedem siebten Durchlauf die Lernrate um 0.5 verringert werden kann. In dem Experiment wird dafür die lr_scheduler.StepLR Klasse verwendet.

Nach jeder Phase wird für die Auswertung des Trainings die berechnete Klassifikationsgenauigkeit und der berechnete Verlust ausgegeben. Auf die Trainingsphase folgt die Validierungsphase. Sind beide Phasen abgeschlossen, wird der Trainingsprozess so lange wiederholt, bis alle 80 Epochen durchlaufen sind. Für die Validierungsphase wird das Modell in den Validierungsmodus versetzt. Dadurch werden alle Bestandteile des Modells, die nur für Aufgaben des Trainings zuständig sind, von PyTorch ignoriert. Somit wird Rechenzeit eingespart. Zudem ist keinerlei Anpassung der Parameter nötig. Deshalb werden die Gradienten deaktiviert. Anschließend wird mit einer Menge aus dem Validierungsdatensatz wieder der Verlust zwischen der Prognose und der korrekten Klasse berechnet. Ebenso wie in der Trainingsphase wird zum Ende der Verlust sowie die Klassifikationsgenauigkeit ausgegeben. Das Modell, das nach einer Epoche die höchste Klassifikationsgenauigkeit auf den Validierungsdatensatz besitzt, wird zwischengespeichert. So wird das Modell erstellt, das mithilfe des Trainingsdatensatzes ausgewertet werden soll.

3.3 Auswertung

Um die generelle Eignung der vortrainierten Modelle für die Klassifikation von Fledermausruf und Störsignalen zu untersuchen, wird zum einen ausgewertet, wie gut die Modelle den Datensatz erlernen können. Zum anderen wird die Fähigkeit zur Klassifikation mit dem Testdatensatz überprüft. Diese Methode wählten auch Sahu et al. (2020) und Sarraf & Tofighi (2017).

Für die Bewertung der Fähigkeit des Erlernens werden die Werte des Verlustes und der Klassifikationsgenauigkeit aus der Test- und Validierungsphase miteinander verglichen. Dafür werden sie als Graphen in einem Diagramm dargestellt. Sowohl die Klassifikationsgenauigkeiten der beiden Phasen als auch die Verluste werden gegenübergestellt. Dadurch lässt sich erkennen, inwiefern die Modelle die Merkmale aus dem Trainingsdatensatz erlernen konnten. Damit lassen sich wiederum Rückschlüsse auf die Vor- und Nachteile der vortrainierten Modelle für diesen Anwendungsfall ziehen.

Um den Einsatz von vortrainierten Modellen für die Ruferfassung mittels UAS zu bewerten, wird mithilfe des Testdatensatzes ein Anwendungsfall für die Modelle simuliert. Die Modelle erhalten einen Datensatz mit Bildern, die sie noch nicht erfasst haben und klassifizieren diese. Die Durchführung verläuft grundsätzlich entsprechend der Validierungsphase im Training, jedoch wird anstelle des Validierungsdatensatzes der Testdatensatz verwendet und dieser wird nur einmal von dem Modell verarbeitet (anstelle von 80-mal wie im Training).

Die Ergebnisse eines Modells werden in einer Konfusionsmatrix dargestellt. Daraus werden die Klassifikationsgenauigkeit, Trefferquote und Präzision für die Fledermausrufe berechnet. Jedoch haben die Metriken für sich einzeln betrachtet nicht viel Aussagekraft über die Eignung der Modelle. In diesem Experiment wird für eine allgemeingültigere Aussagekraft die MCC Metrik verwendet. Sie dient schlussendlich im Rahmen der Interpretation als Entscheidungsgrundlage, für die Frage, welche vortrainierten Modelle die besten Ergebnisse erzielt haben.

Darüber hinaus wird die Zeit gemessen, die die Modelle für die Klassifizierung des Testdatensatzes benötigen. Es ist jedoch zu beachten, dass diese Metrik anhängig von den technischen Ressourcen ist, die für das Experiment zur Verfügung stehen. Außerdem ist die relative Rechenzeit ein geeigneter Indikator, um herauszufinden, wie rechenintensiv ein neuronales Netz im Vergleich zu anderen ist.

4 Ergebnisse

Nachfolgend sind alle Ergebnisse aus der Durchführung des Experimentes aufgelistet. Dabei wurden alle Werte auf drei Nachkommastellen gerundet. Für jedes Modell sind die Konfusionsmatrizen der Klassifikation des Testdatensatzes dargestellt sowie die Kurven der Klassifikationsgenauigkeiten und Verluste von dem des Trainings- und Validierungsdatensatz. In der nachfolgenden Tabelle sind alle Ergebnisse aus dem Testdatensatz zu sehen. In der linken Spalte sind die Metriken für die Auswertung aufgelistet und in den rechten Spalten die jeweiligen berechneten Werte für die trainierten Modelle.

Tabelle 3: Metriken der Modelle auf dem Testdatensatz
Quelle: eigene Darstellung

Metrik	AlexNet	ResNet	ConvNeXt	Vision Transformer
benötigte Zeit	0.911s	0.917s	2.966s	1.638s
Klassifikationsgenauigkeit	0.875	0.906	0.925	0.944
Präzision	0.841	0.891	0.886	0.927
Trefferquote	0.925	0.925	0.975	0.962
MCC	0.754	0.813	0.854	0.888

Abbildung 4-1 zeigt den Verlauf des Verlustes aller Modelle auf dem Trainingsdatensatz über alle 80 Epochen. Die Graphen wurden durch TensorBoard erstellt und besitzen kein Smoothing. Die hellblaue Kurve stellt die Werte des AlexNet Modells dar und besitzt in der letzten Epoche den Wert 0,2866. Die rote Kurve ist die des ResNet Modells und endet mit dem Wert 0,1914. Die grüne Kurve gehört zum ConvNeXt Modell mit einem abschließenden Wert von 0,1036 und die pinke Kurve stellt den Verlust des Vision Transformer Modells dar. Dieser weist in der letzten Epoche einen Wert von 0,02641 aus.

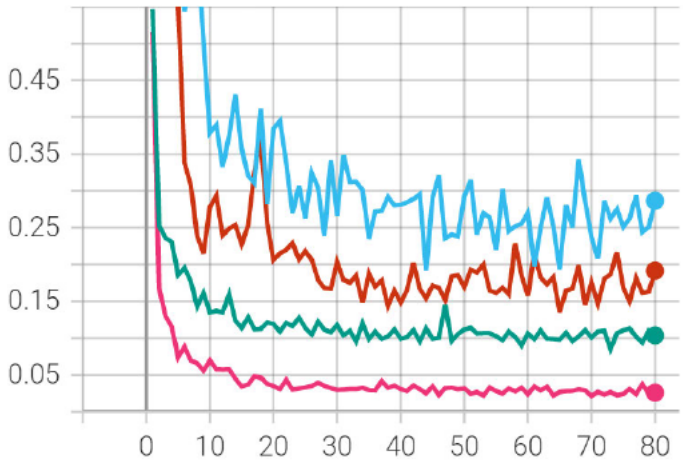

Abbildung 4-1: Vergleich des Trainingsverlustes aller Modelle
Quelle: eigene Darstellung

4.1 AlexNet

Im Nachfolgenden sind die Ergebnisse des AlexNet aus dem Experiment aufgeführt:

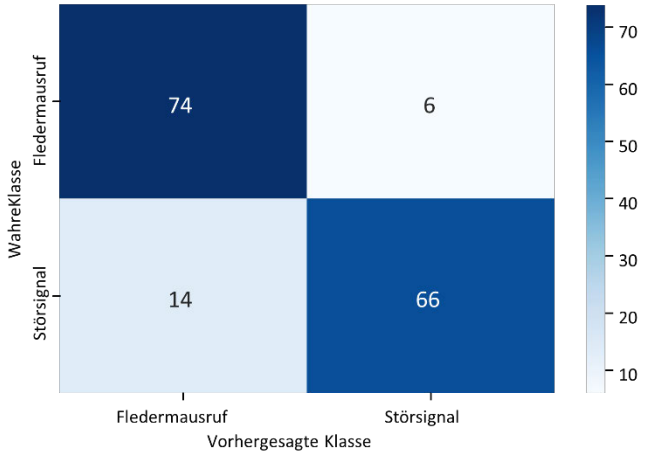

Abbildung 4-2: AlexNet Konfusionsmatrix des Testdatensatzes
Quelle: eigene Darstellung

Die Anzahl an Fledermausrufen und Störsignalen aus dem Testdatensatz, die durch das AlexNet Modell sowohl der korrekten als auch der inkorrekten Klasse zugordnet worden sind, können in der Konfusionsmatrix der Abbildung 4-2 abgelesen werden. Die Matrizen sind für alle Modelle entsprechend gleich aufgebaut. Durch das AlexNet Modell wurden insgesamt 74 Fledermausrufe und 66 Störsignale aus jeweils 80 Bildern korrekt zugeordnet. Dazu wurden sechs Fledermausrufe als Störsignal und 14 Störsignale als Fledermausruf klassifiziert.

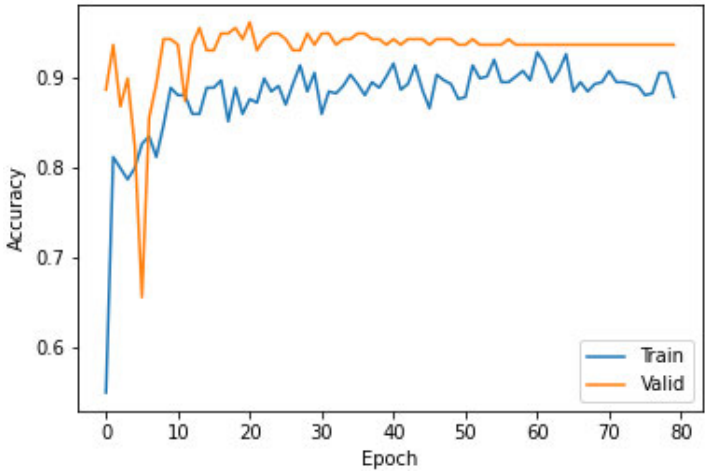

Abbildung 4-3: AlexNet Klassifikationsgenauigkeiten auf dem Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Klassifikationsgenauigkeit auf dem Validierungsdatensatz: 0,937
- Klassifikationsgenauigkeit auf dem Trainingsdatensatz: 0,8729

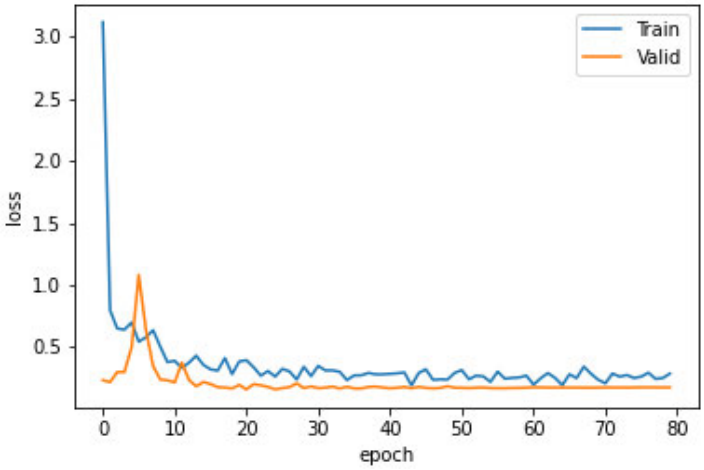

Abbildung 4-4: AlexNet Verlust auf dem Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Verlust auf dem Validierungsdatensatz: 0,175
- Verlust auf dem Trainingsdatensatz: 0,287

4.2 ResNet

Im Nachfolgenden sind die Ergebnisse des ResNet aus dem Experiment aufgeführt:

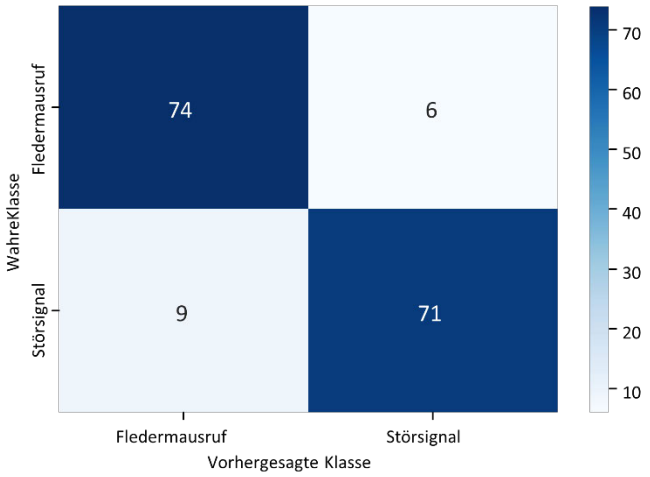

Abbildung 4-5: ResNet Konfusionsmatrix des Testdatensatzes
Quelle: eigene Darstellung

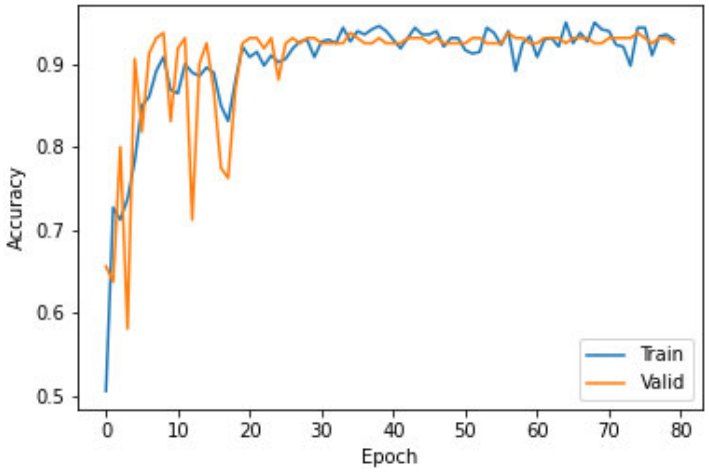

Abbildung 4-6: ResNet Klassifikationsgenauigkeit des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Klassifikationsgenauigkeit auf dem Validierungsdatensatz: 0,925
- Klassifikationsgenauigkeit auf dem Trainingsdatensatz: 0,929

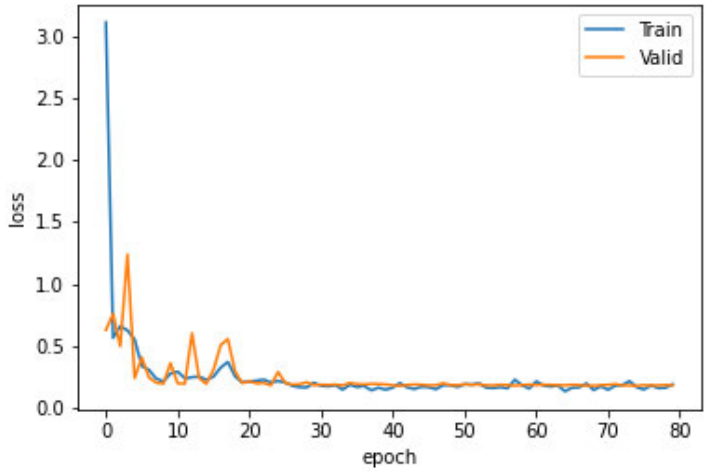

Abbildung 4-7: ResNet Verlust des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Verlust auf dem Validierungsdatensatz: 0,183
- Verlust auf dem Trainingsdatensatz: 0,191

4.3 ConvNeXt

Im Nachfolgenden sind die Ergebnisse des ConvNeXt aus dem Experiment aufgeführt:

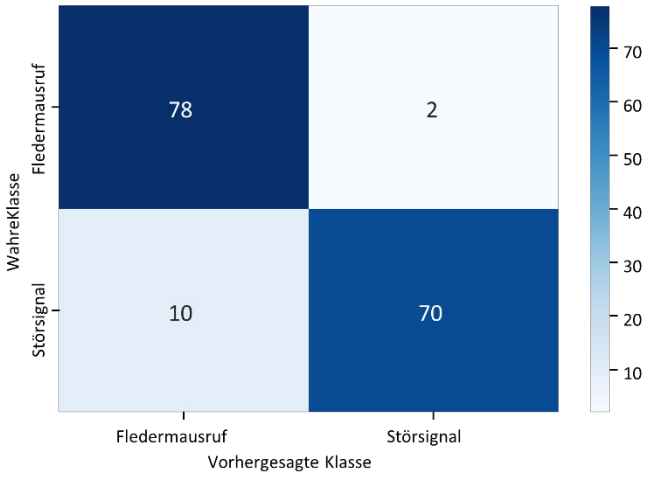

Abbildung 4-8: ConvNeXt Konfusionsmatrix des Testdatensatzes
Quelle: eigene Darstellung

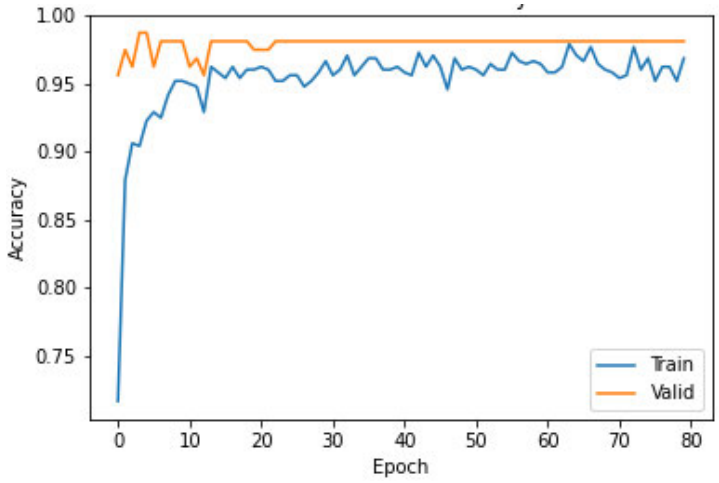

Abbildung 4-9: ConvNeXt Klassifikationsgenauigkeit des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Klassifikationsgenauigkeit auf dem Validierungsdatensatz: 0,981
- Klassifikationsgenauigkeit auf dem Trainingsdatensatz: 0,969

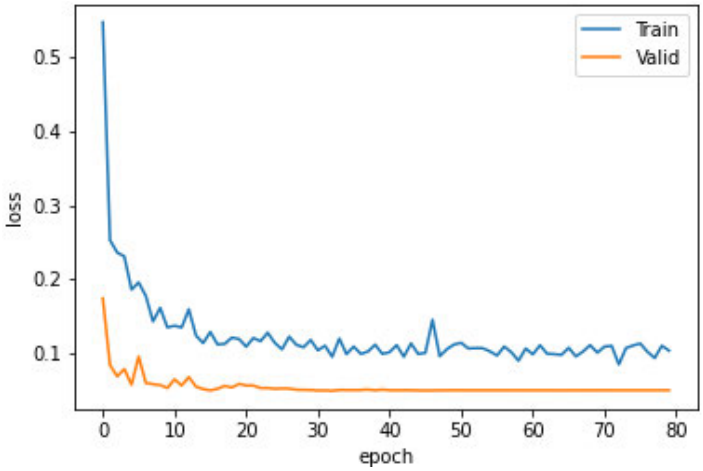

Abbildung 4-10: ConvNeXt Verlust des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Verlust auf dem Validierungsdatensatz: 0,05
- Verlust auf dem Trainingsdatensatz: 0,103

4.4 Vision Transformer

Im Nachfolgenden sind die Ergebnisse des Vision Transformers aus dem Experiment aufgeführt:

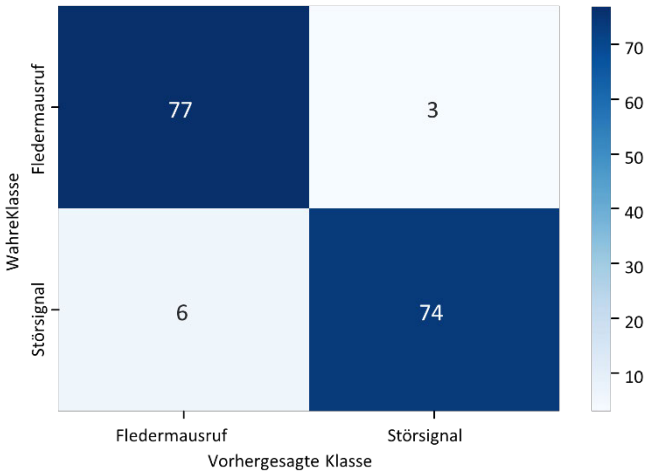

Abbildung 4-11: Vision Transforner Konfusionsmatrix des Testdatensatzes
Quelle: eigene Darstellung

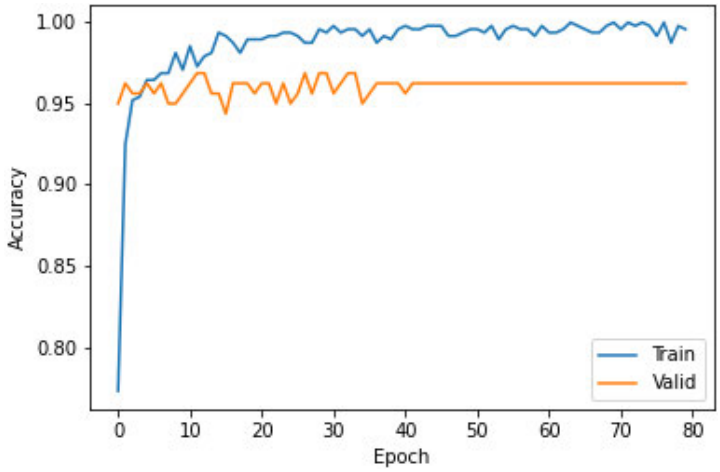

Abbildung 4-12: Vision Transformer Klassifikationsgenauigkeit des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Klassifikationsgenauigkeit auf dem Validierungsdatensatz: 0,963
- Klassifikationsgenauigkeit auf dem Trainingsdatensatz: 0,996

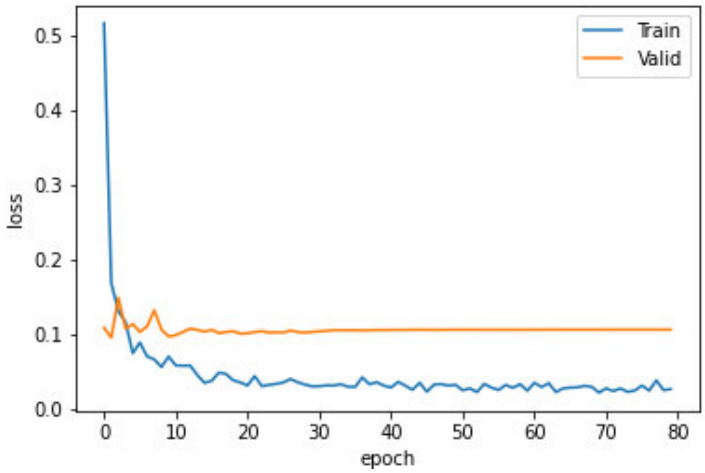

Abbildung 4-13: Vision Transformer Verlust des Trainings- und Validierungsdatensatz
Quelle: eigene Darstellung

- Verlust auf dem Validierungsdatensatz: 0,106
- Verlust auf dem Trainingsdatensatz: 0,026

5 Diskussion

In diesem Kapitel werden die Ergebnisse aus dem Experiment interpretiert, in den wissenschaftlichen Kontext eingeordnet und der Gültigkeitsbereich definiert. Abschließend werden Empfehlungen für weiterführende Forschungen ausgesprochen, die sich aus nicht berücksichtigten Fragestellungen und der Interpretation der Ergebnisse ergeben haben.

5.1 Interpretation

Aus Tabelle 3 wird ersichtlich, dass das Vision Transformer Modell den höchsten Wert für die MCC Metrik erzielt hat. Insofern weist dieses vortrainierte neuronale Netz die besten Klassifikationseigenschaften im Rahmen der hier verwendeten Modelle auf. Das ConvNeXt Modell besitzt als modernstes CNN aus dem Experiment die zweitbesten Klassifikationseigenschaften. Darauf folgt das ResNet Modell und anschließend das AlexNet Modell. Dieses Ergebnis stimmt auch mit dem Ergebnis aus dem Vergleich der Trainingsverluste in Abbildung 4-1 überein. Vision Transformer weist ab spätestens der fünften Epoche den niedrigsten Verlust auf, woraus abgeleitet werden kann, dass dieses Modell die Merkmale von Fledermausrufen und Störsignalen am erfolgreichsten erlernt hat. Gleichermaßen sind die Verluste der CNNs - der Anzahl an enthaltenen Schichten entsprechend - aufsteigend. Auch die generelle Klassifikationsgenauigkeit der Modelle spiegelt diese Rangfolge wider.

Dass Vision Transformer in diesem Experiment die besten Resultate präsentiert hat, stimmt auch mit dem Forschungsergebniss von Dosovitskiy et al. (2020) überein, dass Transformer Architekturen als Weiterentwicklung von CNNs höhere Klassifikationsgenauigkeiten besitzen können. Ebenso deuten die Ergebnisse der CNNs daraufhin, dass sich mit steigender Anzahl an Schichten die Klassifikationseigenschaften positiv entwickeln. Bei genauerer Betrachtung der Präzision und Trefferquote, sind Abweichungen zu erkennen. Bei der Präzision besitzt Vision Transformer zwar wieder den höchsten Wert, aber ResNet hat mit 89,1% eine höhere Präzision als ConvNeXt mit 88,6%. Die höchste Trefferquote hat das ConvNext Modell mit 97,5%. Es hat demnach nur zwei aus insgesamt 80 Fledermausrufen falsch klassifiziert. Entsprechend wäre ConvNeXt das geeignetste Modell, wenn die Trefferquote für einen Anwendungsfall ausschlaggebend ist. Die Trefferquote des AlexNet Modells und ResNet Modells sind mit 92,5% identisch. Es ist davon auszugehen, dass diese Werte mit einem größeren Datensatz divergieren. Zudem sind die Trefferquoten aller Modelle höher als die generellen Klassifikationsgenauigkeiten. Dies entspricht nicht den Erwartungen, weil nicht notwendigerweise davon ausgegangen werden konnte, dass die Fledermausrufe insgesamt öfter der korrekten Klasse zugeordnet werden. Es lässt sich also festhalten, dass alle Modelle die Merkmale von Fledermausrufen besser erlernen konnten als die der Störsignale. Das liegt wahrscheinlich daran, dass - im Gegensatz zu den Störsignalen - die Fledermausrufe bestimmten Charakteristiken folgen. Die Störsignale können in den Sonagrammen jedoch jegliche Form annehmen und besitzen dadurch wahrscheinlich weniger wiedererkennbare Merkmale.

Ein weiterer Aspekt ist, dass zu Beginn des Experimentes die Lernrate und Chargengröße des Trainings auf das AlexNet Modell optimiert wurde und es sich von allen neuronalen Netzen am wenigsten für das Klassifikationsproblem eignet, da es den geringsten MCC Wert aufweist. Wesentlich für die Klassifikationseigenschaften ist demnach die Auswahl eines Modells, dessen Architektur dem aktuellen Forschungsstand entspricht. Eine tiefgehende Optimierung der Hyperparameter hingegen ist zu vernachlässigen. Darüber hinaus zeigt Abbildung 4-1, dass der Trainingsverlust von AlexNet am stärksten ausschlägt. Dies könnte auf eine zu hohe Lernrate hindeuten. Eine niedrigere Lernrate oder eine Anpassung dieser während des Trainings könnten das Ausschlagen verringern.

In Hinblick auf die Zeit, die jedes Modell benötigt hat, um die 160 Bilder zu klassifizieren, hat das ConvNeXt Modell mit knapp drei Sekunden deutlich länger gebraucht als die anderen Modelle. Hinsichtlich der Rechenintensität eignen sich die Modelle AlexNet und ResNet sehr gut, da sie mit 0,911s und 0.917s am wenigsten Rechenzeit in Anspruch genommen haben. Insofern sind in diesem Experiment die Modelle mit der geringsten Anzahl an Schichten die effizientesten.

Die Graphen der Trainings- und Validierungsphasen zeigen, dass mit jeder weiteren Epoche die Gewichtsanpassungen zunehmend geringer werden. Ungefähr ab Epoche 40 sind die Unterschiede zur nächsten Epoche nur noch marginal. Der Verlauf des Trainings und des Validierungsverlustes von AlexNet und Resnet bestätigt, dass der Trainingsprozess sowie die Datensätze für diese Modelle passend abgestimmt sind. Bei ConvNeXt ist der Validierungsverlust erkennbar geringer als der Trainingsverlust. Entsprechend konnte ConvNext den Validierungsdatensatz mit einer höheren Genauigkeit klassifizieren als den Trainingsdatensatz. Es bietet sich an zu überprüfen, inwieweit sich die beiden Datensätze voneinander unterscheiden. Beim Vision Transformer hingegen ist der Validierungsverlust wiederum höher als der Trainingsverlust. Der minimale Validierungsfehler liegt in der zweiten Epoche. Mit jeder weiteren Epoche nimmt der Verlust leicht zu, was dahingehend interpretiert werden kann, dass das Modell mit dem Trainingsdatensatz zur Überanpassung tendiert. Wie in den Grundlagen beschrieben, könnte für Vision Transformer mit einem größeren Datensatz das Risiko des Auftretens einer Überanpassung verringert werden.

Die Ergebnisse aus dem Experiment lassen mit hoher Wahrscheinlichkeit darauf schließen, dass vortrainierte neuronale Netze mit mehr Schichten geeigneter für die Merkmalerkennung sind als Netze mit einer kleineren Architektur. Dies haben W. Yu et al. (2014) ebenso festgestellt. Die erreichten Klassifikationsgenauigkeiten stimmen überwiegend mit den vorgestellten Klassifikationsgenauigkeiten aus anderen Forschungsergebnissen überein. Sie übersteigen zwar die Ergebnisse von Sahu et al. (2020) für die Neuanpassung der letzten Schicht, die Klassifikationsgenauigkeiten von bis zu 96,13% (Schwab et al., 2022) wurden jedoch nicht ganz erreicht.

5.2 Beschränkungen

Die Ergebnisse aus dem Experiment sind nicht allgemeingültig. Sie sind vielmehr von den verwendeten Datensätzen und der Gestaltung des Trainingsprozesses abhängig. Es ist davon auszugehen, dass die verglichenen vortrainierten Modelle mit einem anderen Datensatz und anderen Hyperparametern andere Ergebnisse erzielen werden. Zudem lässt sich auch nicht ausschließen, dass sich bei der Verwendung verschiedener Trainingsmethoden auch die Bewertung hinsichtlich der Stärken und Schwächen der Modelle verändert. Zum Beispiel könnte AlexNet mit einem optimal abgestimmten Training die höchste generelle Klassifikationsgenauigkeit von allen Modellen erreichen. Doch nicht nur die Trainingsmethode hat einen großen Einfluss auf die Ergebnisse. Auch die Anzahl und Funktionen der Schichten sowie die initialen Gewichte vom ImageNet Datensatz haben einen nicht zu vernachlässigenden Einfluss. Erwartungsgemäß ist die allgemeine Laufzeit abhängig von der Entwicklungsumgebung und den technischen Ressourcen, die für das Training zu Verfügung stehen. Außerdem können Verbesserungen in der Programmbibliothek PyTorch zu erhöhten Klassifikationseigenschaften der Modelle führen. Aus diesen Gründen sind die Ergebnisse nur im Rahmen der in Kapitel 3 beschriebenen Methodik gültig. Gleichwohl zeigt die Interpretation, dass die Ergebnisse sich mit den Forschungsergebnissen aus der verwendeten Literatur decken.

5.3 Ausblick

Da das Training eines neuronalen Netzes wie dargestellt komplex ist, konnte im Rahmen dieser Arbeit keine allgemeingültige Antwort auf die eigentliche Forschungsfrage gefunden werden. Die theoretisch und praktisch herausgearbeiteten Ergebnisse können aus diesem Grund nur eine Tendenz für die Eignung der hier behandelten vortrainierten künstlichen neuronalen Netzen aufzeigen. Weiterführende Forschungen sollten die Erkenntnisse aus dieser Arbeit ergänzen.

Für zukünftige Studien ist deshalb zu empfehlen, das Experiment mit mehr vortrainierten neuronalen Netzen zu erweitern. PyTorch bietet weitere, hier nicht beachtete Architekturen an, die untersucht werden können. Mit der neusten Version 1.12 (veröffentlicht am 28.06.2022) sind die EfficientNet und Swin Transformer Architekturen integriert worden (*Releases · Pytorch/Vision*, o. D.). Mit Swin Transformer als alternative Transformer Architektur wäre ein interessanter Vergleich möglich, um die Ergebnisse des Vision Transformers genauer einordnen zu können. Aber auch andere Programmbibliotheken wie TensorFlow bieten vortrainierte Modelle an und gehen somit über die Funktionalitäten von PyTorch hinaus. Insbesondere diejenigen Modelle, die aktuell die höchsten Klassifikationsgenauigkeiten auf dem ImageNet Datensatz nachweisen, erscheinen vielversprechend. Zudem wurde in dem Experiment die Lernrate und Chargengröße nur auf ein Modell optimiert. Das Optimieren dieser Hyperparameter für jedes einzelne Modell erscheint zwar naheliegend, die Ergebnisse wären dadurch aber nur beschränkt vergleichbar. Um die Vergleichbarkeit zu erhalten, könnte eine Methode erforscht werden, wie man die

Hyperparameter für diesen Anwendungsfall passend abstimmen kann. Des Weiteren wäre es überlegenswert, das Anpassen der Gewichte auf mehrere Schichten auszuweiten und nicht nur auf die letzte Schicht zu beschränken. Mit diesem Ansatz konnten Sahu et al. (2020) bei der Klassifikation von Pflanzenkrankheiten eine Klassifikationsgenauigkeit von 96% erreichen.

Da in den verwandten Arbeiten keine vortrainierten neuronalen Netze verwendet wurden, um Fledermausrufe zu klassifizieren, schließt sich die Frage an, inwiefern sich nicht vortrainierte neuronale Netze für die Problematik aus der Einleitung grundsätzlich eignen. So könnte gesondert untersucht werden unter welchen Voraussetzungen derartige Modelle bessere Ergebnisse erzielen als vortrainierte Modelle. Ferner sollten zusätzlich Bildklassifizierungsmethoden untersucht werden, die nicht zu CNNs oder ViT gehören, wie zum Beispiel ein Fuzzy C-Means Algorithmus oder Support Vector Machines.

Unabhängig von den Architekturen ist es für die Ausgestaltung eines Trainingsprozesses sehr wichtig, die Anforderungen an ein neuronales Netz zu definieren. Im Rahmen dieser Thesis war gefordert, dass die Netze sowohl die Merkmale von Fledermausrufen also auch die Merkmale von Störsignalen eines UAS erlernen können. Da in der Literatur unzureichende Forschungsergebnisse für diesen speziellen Anwendungsfall vorgefunden wurden, lag der Schwerpunkt auf der technischen Umsetzung, um diese Anforderungen zu erreichen. Darauf aufbauend könnte erarbeitet werden, welche Anforderungen bei der konkreten automatischen Klassifizierung von Sonagrammen von erfassten Fledermausrufen mittels UAS wichtig sind. Die Auswahl der Architektur, das Training und der Datensatz sollten in Bezug auf die spezielle Fragestellung angepasst werden. Die eigentliche Forschungsfrage sollte demnach um die gewünschten Metriken erweitert werden.

6 Fazit

In dieser Arbeit wurde herausgearbeitet, dass sich Convolutional Neural Networks und Vision Transformer grundsätzlich für Bildklassifizierungsprobleme gut eignen. Um den konkreten, dieser Arbeit zugrundeliegenden Anwendungsfall der automatischen Klassifizierung von Sonagrammen bei der Auswertung von erfassten Fledermausrufen mittels UAS zu untersuchen, ist die Verwendung von vortrainierten Modellen zu empfehlen. Mittels Transfer Learning lassen sich diese Modelle ressourcenschonend trainieren und testen. Um die konkreten Vor- und Nachteile von vortrainierten künstlichen neuronalen Netzen für die Klassifizierung von Fledermausrufen und Störsignalen zu erörtern, wurden vortrainierte Modelle der AlexNet, ResNet, ConvNeXt und Vision Transformer Architektur mit einem eigens erstellten Datensatz aus Audioaufnahmen des Projektes Drones4Bats von der HAW Hamburg trainiert. Anschließend wurden die Ergebnisse aus dem Training sowie aus einem simulierten Einsatz der Modelle ausgewertet.

Die Ergebnisse aus dem Vergleich und aus anderen Forschungen haben gezeigt, dass Klassifikationsgenauigkeiten von über 90% erreichbar sind und dass sich im Allgemeinen moderne Architekturen für die automatische Klassifikation besser eignen als ältere Architekturen mit einer geringeren Anzahl an Schichten. Zudem wiesen die Modelle im Verlauf des Experimentes eine höhere Klassifikationsgenauigkeit von Fledermausrufen als von Störsignalen auf. Aus diesen Gründen erscheint die Eignung von vortrainierten künstlichen neuronalen Netzen für die automatische Klassifizierung von Sonagrammen bei der Auswertung von erfassten Fledermausrufen mittels UAS vielversprechend zu sein. Vor allem Transformer Architekturen können in der konkreten Anwendung eine hohe Klassifikationsgenauigkeit erreichen.

Die Forschungsfrage lässt sich jedoch nicht vollumfänglich beantworten, da dazu eine tiefere Eingrenzung der Anforderungen an die Netze notwendig erscheint. Sollen alle Störsignale aus den Aufnahmen herausgefiltert werden, kann es sein, dass Fledermausrufe verloren gehen. Sollen auf jeden Fall alle Rufe erkannt werden, werden mit hoher Wahrscheinlichkeit wieder Störsignale als Ruf klassifiziert. Die gewünschten Metriken sollten also klar definiert werden, da aufgrund der Komplexität von neuronalen Netzen der Trainingsprozess auf diese optimiert werden sollte.

Abbildungsverzeichnis

Literaturverzeichnis

Abdullah, N., Ngah, U. K., & Aziz, S. A. (2011). Image classification of brain MRI using support vector machine. *2011 IEEE International Conference on Imaging Systems and Techniques*, 242–247. https://doi.org/10.1109/IST.2011.5962185

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer.

Ahmed, M. N., Yamany, S. M., Mohamed, N., Farag, A. A., & Moriarty, T. (2002). A modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data. *IEEE Transactions on Medical Imaging*, *21*(3), 193–199. https://doi.org/10.1109/42.996338

Alnajjar, M. K., & Abu-Naser, S. S. (2022). *Heart Sounds Analysis and Classification for Cardiovascular Diseases Diagnosis using Deep Learning*. http://dspace.alazhar.edu.ps/xmlui/handle/123456789/3534

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms: An Overview. *Journal of Physics: Conference Series*, *1142*, 012012. https://doi.org/10.1088/1742-6596/1142/1/012012

Amiriparian, S., Gerczuk, M., Ottl, S., Cummins, N., Freitag, M., Pugachevskiy, S., Baird, A., & Schuller, B. (2017, August 24). *Snore Sound Classification Using Image-Based Deep Spectrum Features*. https://doi.org/10.21437/Interspeech.2017-434

Aodha, O. M., Gibb, R., Barlow, K. E., Browning, E., Firman, M., Freeman, R., Harder, B., Kinsey, L., Mead, G. R., Newson, S. E., Pandourski, I., Parsons, S., Russ, J., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Girolami, M., Brostow, G., & Jones, K. E. (2018). Bat detective—Deep learning tools for bat acoustic signal detection. *PLOS Computational Biology*, *14*(3), e1005995. https://doi.org/10.1371/journal.pcbi.1005995

Aykanat, M., Kılıç, Ö., Kurt, B., & Saryal, S. (2017). Classification of lung sounds using convolutional neural networks. *EURASIP Journal on Image and Video Processing*, *2017*(1), 65. https://doi.org/10.1186/s13640-017-0213-2

Boddapati, V., Petef, A., Rasmusson, J., & Lundberg, L. (2017). Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, *112*, 2048–2056. https://doi.org/10.1016/j.procs.2017.08.250

Böttcher, W., Bunne, C., & von Stetten, J. (2019). Gradientenabstiegsverfahren. In K. Kersting, C. Lampert, & C. Rothkopf (Hrsg.), *Wie Maschinen lernen: Künstliche Intelligenz verständlich erklärt* (S. 171–180). Springer Fachmedien. https://doi.org/10.1007/978-3-658-26763-6_22

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A.,

Brownlee, J. (2019, 27. Februar) *How to use Learning Curves to Diagnose Machine Learning Model Performance.* machinelearningmastery. Abgerufen am 28. Juli 2022, von https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

Chiao, J.-Y., Chen, K.-Y., Liao, K. Y.-K., Hsieh, P.-H., Zhang, G., & Huang, T.-C. (2019). Detection and classification the breast tumors using mask R-CNN on sonograms. *Medicine*, *98*(19), e15200. https://doi.org/10.1097/MD.0000000000015200

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

Dominguez-Morales, J. P., Jimenez-Fernandez, A. F., Dominguez-Morales, M. J., & Jimenez-Moreno, G. (2018). Deep Neural Networks for the Recognition and Classification of Heart Murmurs Using Neuromorphic Auditory Sensors. *IEEE Transactions on Biomedical Circuits and Systems*, *12*(1), 24–34. https://doi.org/10.1109/TBCAS.2017.2751545

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale* (arXiv:2010.11929). arXiv. https://doi.org/10.48550/arXiv.2010.11929

Dudenredaktion (o. D.). In *Duden online.* Abgerufen am 28. Juli 2022, von https://www.duden.de/rechtschreibung/Sonagramm

Fujioka, T., Mori, M., Kubota, K., Oyama, J., Yamaga, E., Yashima, Y., Katsuta, L., Nomura, K., Nara, M., Oda, G., Nakagawa, T., Kitazume, Y., & Tateishi, U. (2020). The Utility of Deep Learning in Breast Ultrasonic Imaging: A Review. *Diagnostics*, *10*(12), 1055. https://doi.org/10.3390/diagnostics10121055

Gavali, P., & Banu, J. S. (2019). Chapter 6—Deep Convolutional Neural Network for Image Classification on CUDA Platform. In A. K. Sangaiah (Hrsg.), *Deep Learning and Parallel Computing Environment for Bioengineering Systems* (S. 99–122). Academic Press. https://doi.org/10.1016/B978-0-12-816718-2.00013-0

Glielmo, A., Husic, B. E., Rodriguez, A., Clementi, C., Noé, F., & Laio, A. (2021). Unsupervised Learning Methods for Molecular Simulation Data. *Chemical Reviews*, *121*(16), 9722–9758. https://doi.org/10.1021/acs.chemrev.0c01195

Guo, J., Han, K., Wu, H., Tang, Y., Chen, X., Wang, Y., & Xu, C. (2022). *CMT: Convolutional Neural Networks Meet Vision Transformers* (arXiv:2107.06263). arXiv. https://doi.org/10.48550/arXiv.2107.06263

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., Yang, Z., Zhang, Y., & Tao, D. (2022). A Survey on Vision Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2022.3152247

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

Hurst, J., Biedermann, M., Dietz, C., Dietz, M., Reers, H., Karst, I., Petermann, R., Schorcht, W., & Brinkmann, R. (2020). Wind energy production in forests and bat conservation – an overview of the current state of knowledge and suitable methods for monitoring and measures. In C. C. Voigt (Hrsg.), *Evidenzbasierter Fledermausschutz in Windkraftvorhaben* (S. 29–54). Springer. https://doi.org/10.1007/978-3-662-61454-9_2

Image Classification on ImageNet. (o. D.). paperswithcode. Abgerufen am 28. Juli 2022, von https://paperswithcode.com/sota/image-classification-on-imagenet.

ImageNet. (o. D.). image-net. Abgerufen am 27. Juli 2022, von https://www.image-net.org/index.php

Jmour, N., Zayen, S., & Abdelkrim, A. (2018). Convolutional neural networks for image classification. *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 397–402. https://doi.org/10.1109/ASET.2018.8379889

Kiencke, U. (2009). *Signalverarbeitung Zeit-Frequenz-Analyse und Schätzverfahren*. Oldenbourg.

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980

Krizhevsky, A. (2014). *One weird trick for parallelizing convolutional neural networks* (arXiv:1404.5997). arXiv. https://doi.org/10.48550/arXiv.1404.5997

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

Kruszynski, C., Bailey, L. D., Bach, L., Bach, P., Fritze, M., Lindecke, O., Teige, T., & Voigt, C. C. (2022). High vulnerability of juvenile Nathusius' pipistrelle bats (Pipistrellus nathusii) at wind turbines. *Ecological Applications*, *32*(2), e2513. https://doi.org/10.1002/eap.2513

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, *405*(2), 442–451. https://doi.org/10.1016/0005-2795(75)90109-9

Models and pre-trained weights - Torchvision 0.13 documentation. (o. D.). pytorch. Abgerufen am 19. Juli 2022, von https://pytorch.org/vision/stable/models.html#classification

Mohsen, H., El-Dahshan, E.-S. A., El-Horbaty, E.-S. M., & Salem, A.-B. M. (2018). Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, *3*(1), 68–71. https://doi.org/10.1016/j.fcij.2017.12.001

Müller, R., Ritz, F., Illium, S., & Linnhoff-Popien, C. (2021). Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning. *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, 49–56. https://doi.org/10.5220/0010185800490056

Nath, S. S., Mishra, G., Kar, J., Chakraborty, S., & Dey, N. (2014). A survey of image classification methods and techniques. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 554–557. https://doi.org/10.1109/ICCICCT.2014.6993023

Oppermann, A. (2021) *Accuracy, Precision, Recall, F1-Score und Specificity*. artemoppermann. Abgerufen am 28. Juli 2022, von https://artemoppermann.com/de/accuracy-precision-recall-f1-score-und-specificity/

Paaß, G., & Hecker, D. (2021). *Künstliche Intelligenz: Was steckt hinter der Technologie der Zukunft?* (1. Aufl. 2020 Edition). Springer Vieweg.

Parsons, S., & Jones, G. (2000). Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of Experimental Biology*, *203*(17), 2641–2656. https://doi.org/10.1242/jeb.203.17.2641

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained Models for Natural Language Processing: A Survey. *Science China Technological Sciences*, *63*(10), 1872–1897. https://doi.org/10.1007/s11431-020-1647-3

Releases · pytorch/vision. (o. D.). GitHub. Abgerufen am 23. Juli 2022, von https://github.com/pytorch/vision/releases

Sahu, P., Chug, A., Singh, A., Singh, D., & Singh, R. (2020). *Implementation of CNNs for Crop Diseases Classification: A Comparison of Pre-trained Model and Training from Scratch*. 206. https://doi.org/10.22937/IJCSNS.2020.20.10.26

Sarraf, S., & Tofighi, G. (2017). *Classification of Alzheimer's Disease Structural MRI Data by Deep Learning Convolutional Neural Networks* (arXiv:1607.06583). arXiv. http://arxiv.org/abs/1607.06583

Scheid, S., & Vogl, S. (2021). *Data Science: Grundlagen, Methode und Modelle der Statistik*. Carl Hanser Verlag GmbH Co KG.

Schmarje, L., Santarossa, M., Schröder, S.-M., & Koch, R. (2021). A Survey on Semi-, Self- and Unsupervised Learning for Image Classification. *IEEE Access*, *9*, 82146–82168. https://doi.org/10.1109/ACCESS.2021.3084358

Schwab, E., Pogrebnoj, S., Freund, M., Flossmann, F., Vogl, S., & Frommolt, K.-H. (2022). Automated bat call classification using deep convolutional neural networks. *Bioacoustics*, *0*(0), 1–16. https://doi.org/10.1080/09524622.2022.2050816

Schreiber, M. (2016). *Abschaltzeiten für Windkraftanlagen zur Vermeidung und Verminderung von Vogelkollisionen.* https://www.fachagentur-windenergie.de/fileadmin/files/Veranstaltungen/Runder_Tisch_Vermeidungsmassnahmen/1._Runder_Tisch_24.02.2016/Studie_Abschaltzeiten_Dr._Schreiber_LKR_Osnabarueck_2016.pdf

Steinwendner, J., & Schwaiger, R. (2019). *Neuronale Netze programmieren mit Python*. Rheinwerk Verlag GmbH.

Unmanned Aircraft Systems (*Releases*.). trade. Abgerufen am 28. Juli 2022, von https://www.trade.gov/unmanned-aircraft-systems#:~:text=Data%20Collection-,Overview,Sytems%20(RPAS)%20and%20drones

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need* (arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Voigt, C. C., Lehnert, L. S., Petersons, G., Adorf, F., & Bach, L. (2015). Wildlife and renewable energy: German politics cross migratory bats. *European Journal of Wildlife Research*, *61*(2), 213–219. https://doi.org/10.1007/s10344-015-0903-y

Yaqub, M., Feng, J., Zia, M. S., Arshid, K., Jia, K., Rehman, Z. U., & Mehmood, A. (2020). State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images. *Brain Sciences*, *10*(7), 427. https://doi.org/10.3390/brainsci10070427

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., & Wu, Y. (2022). *CoCa: Contrastive Captioners are Image-Text Foundation Models* (arXiv:2205.01917). arXiv. https://doi.org/10.48550/arXiv.2205.01917

Yu, T., & Zhu, H. (2020). *Hyper-Parameter Optimization: A Review of Algorithms and Applications* (arXiv:2003.05689). arXiv. https://doi.org/10.48550/arXiv.2003.05689

Yu, W., Yang, K., Bai, Y., Yao, H., & Rui, Y. (2014). *Visualizing and Comparing Convolutional Neural Networks* (arXiv:1412.6631). arXiv. https://doi.org/10.48550/arXiv.1412.6631

Zeng, Y., Mao, H., Peng, D., & Yi, Z. (2019). Spectrogram based multi-task audio classification. *Multimedia Tools and Applications*, *78*(3), 3705–3722. https://doi.org/10.1007/s11042-017-5539-3

Zhang, J., Li, C., Yin, Y., Zhang, J., & Grzegorzek, M. (2022). Applications of artificial neural networks in microorganism image analysis: A comprehensive review from conventional multilayer perceptron to popular convolutional neural network and potential visual transformer. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-022-10192-7

Ziegler, D., Wu, J., Winter, C., … Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, *33*, 1877–1901. https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., & Chand, P. (2020). A Tiny CNN Architecture for Identifying Bat Species from Echolocation Calls. *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*, 81–86. https://doi.org/10.1109/AI4G50087.2020.9311084

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Künstliche neuronale Netze für die Bildklassifikation von Fledermausrufen

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Datum	Unterschrift