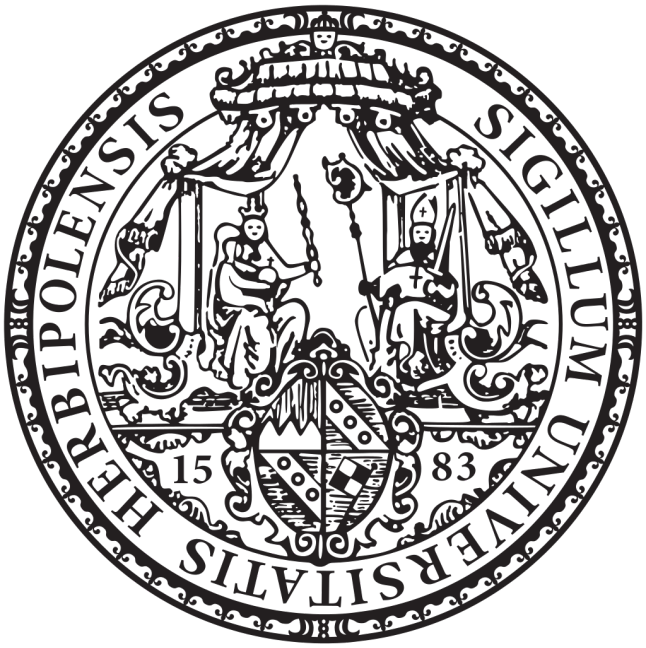
Julius-Maximilians-Universität Würzburg

Institut für Informatik
Lehrstuhl für Künstliche Intelligenz
und Angewandte Informatik

Bachelorarbeit

im Studiengang Informatik



zur Erlangung des akademischen Grades
Bachelor of Science

Automatisierte Kardiomegalie Erkennung durch auf Deep Learning basierter Objektdetektion

---

Autor: Nils Dienesch (nils.dienesch@stud-mail.uni-wuerzburg.de)
MatNr. 2334799

Abgabe: 03.08.2021

1. Betreuer: Prof. Dr. Frank Puppe
2. Betreuer: M. Sc. Amar Hekalo

Abstract

Im Bereich der Kardiomegalie Erkennung basierend auf Deep Learning, wurden bisher nur Neuronale Netze zur Klassifikation und zur Segmentierung evaluiert.[44] Beim Ansatz der Segmentierung werden aus ausgegebenen Masken der Querdurchmesser des Herzens und der Brust berechnet. Aus diesen beiden Werten ergibt sich der Herz-Thorax-Quotient, welcher ab einem Wert über 50% Kardiomegalie indiziert.[5] Doch für die Berechnung der Querdurchmesser wären auch Rahmen um die Organe ausreichend, welche von Objektdetektions Neuronalen Netzen ausgeben werden können. In bisherigen Veröffentlichungen zur automatischen Berechnung des Herz-Throax-Quotienten mithilfe Neuronaler Netze, wurde diese Option jedoch nicht betrachtet. In dieser Arbeit wird das vortrainierte Neuronale Netz Faster R-CNN[35] aus Detectron2[52] für die Erkennung von Herz und Lunge trainiert, um das Potential von Objektdetektoren in der Kardiomegalie Erkennung zu untersuchen. Auf einem privaten Testdatensatz, mit von einem Radiologen gemessenen Herz-Thorax-Quotienten, erreichte Faster R-CNN eine Sensitivity von 100% und eine Specificity von 82%. Dabei weisen die gemessenen Werte eine mittlere Differenz von $0,0291 \pm 0,0889$ zu den Werten des Radiologen auf. Des Weiteren wurde in Zusammenarbeit mit einem Radiologen der Uniklinik Würzburg die Anwendbarkeit des trainierten Netzes als Warnsystem im klinischen Alltag getestet. Dabei konnten vielversprechende Ergebnisse erzielt werden, welche zeigen, dass das Neuronale Netz bei einer schnellen Erkennung von Verdachtsfällen mit der Effektivität eines Radiologen mithalten kann. Um Vergleichswerte für den Ansatz der Segmentierung zu erhalten, wurde das U-Net[37] aus der Bachelorarbeit von Hasler[14] auf dem privaten Datensatz dieser Arbeit neu ausgewertet. Diese Werte konnten übertroffen werden. Ebenfalls wurde das Netz Mask R-CNN[16] aus Detectron2 trainiert, um zu testen, ob die Kombination aus Objektdetektion und Segmentierung weitere Verbesserungen erzielen kann. Ausschlaggebende Verbesserungen zeigten sich jedoch nicht.

Inhaltsverzeichnis

Abkürzungsverzeichnis

CNN	Convolutional Neuronales Netz
FAIR	Facebook AI Research
GPU	Graphic Processing Unit
HTQ	Herz-Thorax-Quotient
ID	Internal Diameter of Chest
IoU	Intersection over Union
JSON	JavaScript Object Notation
MLD	Midline to Left Heart Diameter
MRD	Midline to Right Heart Diameter
NN	Neuronales Netz
ReLU	Rectified Linear Unit
ResNet34	Residuales Netzwerk der Tiefe 34
RLE	Run Length Econding
RoI	Region of Interest
RPN	Region Proposal Netzwerk
RPN	Region Proposal Netzwerk
SGD	Stochastic Gradient Descent
UKW	Uniklinik Würzburg
UTF-8	8-Bit Universal Character Set Transformation Format

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Kardiomegalie ist eine Sammelbezeichnung für verschiedenste Ursachen, die zu einer Vergrößerung des Herzens führen. Gründe dafür können eine koronare Herzerkrankung, eine Herzklappenerkrankung, ein angeborener Herzfehler oder eine Herzrhythmusstörung sein. Des weiteren könnte auch eine Kardiomyopathie zugrunde liegen, welche unter anderem durch Gifte wie zum Beispiel Alkohol oder Stress herbeigeführt werden kann. Bekannt ist auch die durch Sport ausgelöste Kardiomegalie. Die Kardiomegalie wird meist erst erkannt, wenn erste Folgen auftreten. Dabei ist die Krankheit sehr gefährlich, denn sie kann zu Herzinsuffizienz führen. Diese ist für die Hälfte aller Betroffenen innerhalb von fünf Jahren tödlich. Eine frühzeitige Erkennung von Kardiomegalie ist deshalb sehr wichtig. Dabei spielen Röntgen Thorax Aufnahmen eine wichtige Rolle.[3] Sie sind kostengünstig, benötigen nur eine geringe Strahlendosis und stehen breit zur Verfügung.[44] Auf den Aufnahmen kann der Herz-Thorax-Quotient gemessen werden, welcher bei einem Wert ab 50% eine Kardiomegalie indiziert.[3] Fortschritte im Bereich des Deep Learning haben gezeigt, dass Herz und Lunge akkurat detektiert werden können und somit eine automatische Kardiomegalie Erkennung über den HTQ möglich ist. Nun liegt das Ziel darin, immer bessere Neuronale Netze zu trainieren, um Ärzte tatsächlich bei ihrer Arbeit unterstützen zu können.[5] Dabei liegt der Ansatz aktuell in der Bildsegmentierung, bei der Herz und Lunge pixelgenau maskiert werden, um deren Breiten zu bestimmen. Doch um diese zu bestimmen, würden genauso einfache Rahmen um die Organe ausreichen. Diese könnten durch Neuronale Objektdetektoren erzeugt werden. Trotzdem gibt es im Bereich der Kardiomegalie-Erkennung mit Objektdetektoren noch keine Veröffentlichungen. Dabei wäre alleine die Generierung der Daten, welche zum Trainieren der NN benötigt werden, deutlich einfacher. Das Ziel dieser Arbeit ist es deshalb, die Objektdetektion im Bereich der Kardiomegalie Erkennung zu untersuchen. In dieser Arbeit soll folgenden Fragen nachgegangen werden:

Frage 1: Kann die Kardiomegalie Erkennung durch Objektdetektion verbessert werden?

Frage 2: Enthalten beim Training der NN Masken komplementäre Informationen zu Boundingboxen, welche die Objektdetektion verbessern können?

Frage 3: Sollte zur Berechnung des HTQs die ganze Lunge oder die einzelnen Lungenflügel detektiert werden?

Frage 4: Wie relevant sind die erzielten Ergebnisse für den medizinischen Einsatz?

1.1 Hintergrund

Wie aus der Einleitung hervorgeht liegt der Fokus dieser Arbeit auf der Berechnung des HTQ auf einem Röntgenbild des menschlichen Brustkorbs. Aus diesem Grund soll der HTQ hier genauer erläutert werden. In Abbildung 1 ist der Midline to Right Heart

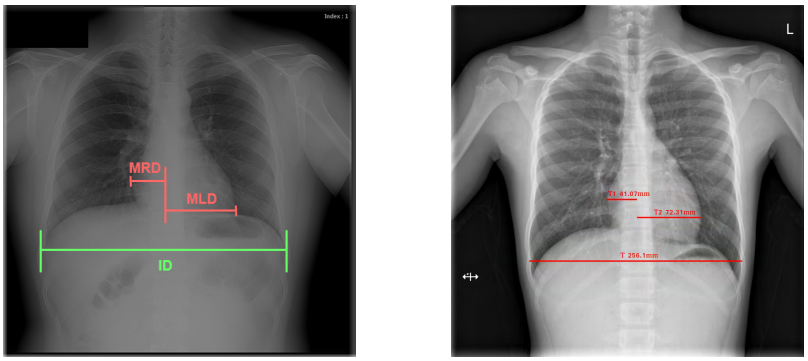


Abbildung 1: **Links:** Das Bild zeigt eine posterior-anterior Röntgenaufnahme des Thorax. Auf diesem ist der Querdurchmesser des Herzens (MRD+MLD) und der Querdurchmesser der Brust (ID) abgebildet.[5] **Rechts:** Eine Messung eines Radiologen.[24]

Diameter, der Midline to Left Heart Diameter und der Internal Diameter of Chest abgebildet, also der Querdurchmesser des Herzens von der Mittellinie zur rechten Grenze, der Querdurchmesser des Herzens von der Mittellinie zur linken Grenze und der Querdurchmesser der Brust. Im weiteren Verlauf der Arbeit werden die Abkürzungen der englischen Begriffe verwendet, da sie prägnanter und gleichzeitig aussagekräftiger sind.[5] Verwirren lassen sollte man sich nicht von den seitenverkehrten Röntgenbildern. Diese stellen eine posterior-anterior Projektion[3] dar, bei der die Röntgenstrahlen von hinten durch den Körper strahlen. Der HTQ berechnet sich durch folgenden Formel:

$$\text{HTQ} = \frac{\text{MRD} + \text{MLD}}{\text{ID}} \tag{1}$$

MRD+MLD und ID können dabei durch die Breite eines Rahmens um jeweils Herz und Lunge gemessen werden. Die separate Veranschaulichung von MRD und MLD dient dem Verständnis. Überschreitet der Wert des HTQs 50%, liegt bei dem Patienten Kardiomeglie vor.[5]

2 Grundlagen

2.1 Neuronale Netze

Oft werden Neuronale Netze mit einer Analogie zum Menschlichen Gehirn eingeführt. In dieser Arbeit dieser jedoch nicht aufgegriffen, da er schnell zu einer falschen Vorstellung eines NN führen kann. Stattdessen soll genauer auf die Mathematik hinter NN eingegangen werden. Grundsätzlich ist ein NN eine Funktion $f(x;\theta)$, welche die Funktion $y = f^*(x)$ approximiert. f^* stellt dabei die Funktion dar, welche für ein konkretes Problem zu jeder möglichen Eingabe x die korrekte Ausgabe y berechnet. Durch das Approximieren von f^* soll f dieses Problem möglichst gut lösen. Damit das NN lernen kann das Problem zu lösen, das heißt f^* besser zu approximieren, besitzt es die anpassbaren Parameter θ. Diese werden während des Lernprozesses eines NN, dem sogenannten Trainieren, angepasst. Dies soll an einem Beispiel verdeutlicht werden. Eine lineare Funktion, wie beispielsweise $f^*(x) = 5x + 2x$, ist bekannt. In einem NN sähe diese Funktion wie folgt aus: $f(x;\theta) = \theta_1 x + \theta_2 x$.[11, Kapitel 6] Durch das richtige Anpassen von θ kann $f^*(x) = 5x + 2x$ approximiert oder im Optimalfall mit $\theta_1 = 2$ und $\theta_2 = 5$ berechnet werden. Ein Neuronalen Netz muss jedoch viel komplexere Probleme lösen. Dazu braucht es auch komplexere Funktionen. In einem NN werden diese durch das Schachteln vieler Funktionen aufgebaut. $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$ Bei den einzelnen Funktionen $f^{(i)}(x)$ spricht man von den Schichten des Neuronalen Netzes. Durch dieses Vorgenen werden sehr große beziehungsweise tiefe Funktionen geschaffen, welche hunderte Millionen Parameter[22] enthalten und dadurch viele Informationen über das zu lösende Problem lernen können. Die Eingabe x sowie die Ausgaben der einzelnen Schichten des NN sind Vektoren. Die Funktion $f^{(i)}$ stellt also grundsätzlich eine Funktion dar, welche aus einem Vektor einen Ausgabevektor berechnet. In Abbildungen sind diese Vektorfunktionen oftmals durch Neuronen dargestellt. In diesem Fall stellt man sich die Vektorfunktion einer Schicht als parallele Funktionen vor, wobei jede einzelne Funktion als Neuron bezeichnet wird. Diese berechnen dann aus dem Eingabevektor parallel einfache Zahlen. In einem NN gehört zu einer Schicht eine nicht lineare Funktion, die Aktivierungsfunktion, welche $f^{(i)}$ umschließt. Dadurch wird auch $f^{(i)}$ zu einer nicht linearen Funktion. Würde jede Schicht schlichtweg eine lineare Funktion darstellen, wäre die Ausgabe von $f(x)$ nur eine lineare Transformation der Eingabe x. Wie man sich leicht vorstellen kann stellen Aufgaben[22] wie das Verstehen von Sprache oder das Erkennen von Objekten auf Bildern, allesamt Probleme, die Neuronale Netze lösen können, keine linearen Zusammenhänge dar. Die Aktivierung jeder Schicht mit einer nicht linearen Funktion macht dies möglich. In verschiedenen Darstellungen eines NN gehört diese entweder fest zu einer Schicht dazu oder wird als eigene Schicht interpretiert. Ein Beispiel für eine Aktivierungsfunktion stellt die Rectified Linear Unit, zu sehen in Abbildung 2, dar.[11, Kapitel 6] Bisher repräsentierte $f^{(i)}$ eine Schicht. Bisher

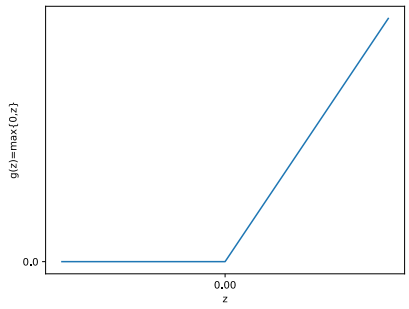


Abbildung 2: Die ReLU $g(z) = max\{0, z\}$ stellt heute den Standard unter den Aktivierungsfunktionen dar, denn ihre Einfachheit bietet verschiedene Vorteile.[11, Kapitel 6] Zu bedenken ist, dass das Neuronale Netz die für eine Schicht benötigte Funktion schlichtweg lernen kann.

blieb jedoch offen, welche Funktion $f^{(i)}$ repräsentiert. Dafür gibt es viele Möglichkeiten, doch die grundlegende ist die generalisierte lineare Funktion

$$f(x; W, b) = W^\top x + b \tag{2}$$

Diese wird auch als Lineare Schicht bezeichnet. Dabei ist $x \in \mathbb{R}^e$ die Eingabe, $W \in \mathbb{R}^{e \times n}$ die Gewichtsmatrix und $b \in \mathbb{R}^n$ der Bias, wobei W und b Parameter aus θ und damit lernbar sind. Die Ausgabe der Schicht ist also ein neuer Vektor der Größe n, wobei die Funktion, welche diesen berechnet, vollständige vom NN gelernt wird.[11, Kapitel 6][2, Kapitel 1.2]. Hierbei können die zwei unterschiedlichen Sichtweisen auf diese Schicht noch einmal verdeutlicht werden. Eben vorgestellt wurde eine Matrixoperation, welche man genauso als n parallel gewichtet Summen der Eingabe x auffassen kann. Diese stellen dann die Neuronen dar, von denen das $i$te die Funktion

$$\sum_j W_{ij} \cdot x_j + b_i \tag{3}$$

berechnet. Somit kann man sich ein NN wie in Abbildung 3 vorstellen. Auf dieser Grafik sind die Neuronen als Kreise dargestellt, welche die mit den Kantengewichten gewichtete Summe ihrer Eingabe berechnen. Die Kanten sind dabei mit den Gewichten aus W gewichtet und der Bias aus Gründen der Übersicht weggelassen. In diesem Fall wird die Aktivierungsfunktion nicht als eigenständige Schicht gezeigt. Die auf Abbildung 3 zu sehende Ausgabeschicht (output layer) stellt die letzte Schicht eines NN dar und hat die Aufgabe, die konkrete Ausgabe $\hat{y}$ zu berechnen. Im Folgenden bezeichnet $\hat{y}$ die Ausgabe eines Neuronalen Netzes also $f(x)$. Was alle weiteren Schichten berechnen wird alleine vom Netz kontrolliert, weshalb diese als versteckte Schichten bezeichnet werden.[11, Kapitel 6] Dies Schichten haben die Aufgabe die Eingabe nicht linear zu verzerren, sodass die Ausgabeschicht ein lineares Problem vor sich sieht.[22]

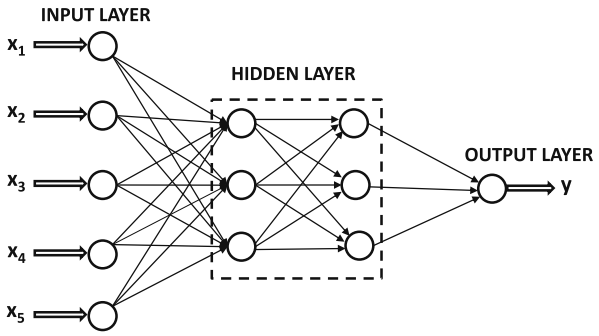


Abbildung 3: Die Darstellung[2, Kapitel 1.2] zeigt ein Neuronales Netz als Graph, wobei die Sicht auf die einzelnen Neuronen gewählt wird. Das abgebildete Netz enthält zwei versteckte Schichten und eine Ausgabeschicht bestehend aus einem Neuron. Das NN berechnet also aus fünf Eingabewerten einen Ausgabewert.

2.1.1 Ausgabeschichten

Eben wurde die Ausgabeschicht eingeführt, deren zentrale Aufgabe darin besteht die von den versteckten Schichten berechneten Werte in die erwünschte Ausgabe zu transformieren. Ein Beispiel für eine Ausgabeschicht stellt die Softmax Schicht dar. Diese berechnet aus den Ausgabewerten verschiedener Neuronen eine Wahrscheinlichkeitsverteilung. Das heißt jeder Wert muss zwischen 0 und 1 liegen und sämtliche Werte müssen sich zu 1 summieren.

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{4}$$

Vor allem bei der Problemstellung der Klassifizierung kommt diese zu Schicht zum Einsatz. Hierbei wird eine Eingabe einer von k Klassen zugeordnet, wozu eine lineare Schicht mit k Neuronen benutzt wird. Deren Ausgaben werden durch die die Softmaxfunktion normalisiert, wodurch die Ausgabe $\hat{y}_i = P(y = i|x)$ entsteht. Dabei bedeutet beispielsweise $f^*(x) = y = 0$, dass 0 die korrekte Klasse ist.[11, Kapitel 6]

2.1.2 Hyperparameter

Bei einem Neuronalen Netz stellen Hyperparameter Paramter dar, welche nicht in θ enthalten sind. Aus diesem Grund können sie nicht erlernt werden und müssen somit vor dem Training des NN festgelegt werden.[2, Kapitel 3.3.1]

2.2 Training

Damit ein Neuronales Netz eine Funktion lernen kann besitzt es die lernbaren Parameter θ. Der Prozess, diese von einem initialen Wert aus anzupassen, um die Funktion $f^*(x)$ möglichst gut zu approximieren, wird Training genannt. In diesem lernt das NN also ein gegebenes Problem möglichst gut zu lösen. Die initialen Werte können dabei

beispielsweise von einem früheren Training des NN stammen, dann ist das NN schon vortrainiert. Wichtig für diese Arbeit ist das sogenannte beaufsichtigte Lernen. Bei diesem stehen im Trainingsprozess viele Datenpaare aus Eingabebeispielen x und dazugehörigen korrekten Ausgaben $y = f^*(x)$ zur Verfügung. Die Kernidee des Trainings besteht nun darin das $\hat{y} = f(x)$ zu einem Eingabebeispiel zu berechnen und θ anschließend, basierend auf dem Vergleich zwischen $\hat{y}$ und $y = f^*(x)$, anzupassen. Dazu werden zwei Komponenten benötigt: Eine Funktion, welche $\hat{y}$ mit y vergleicht. Ein Mechanismus, welcher die Parameter des NN $f(x;\theta)$ so anpasst, dass anschließend $f^*(x)$ besser approximiert wird.[11, Kapitel 5]

2.2.1 Loss Funktion

Die soeben erwähnte Vergleichsfunktion wird als Loss Funktion L bezeichnet und berechnet, wie weit zwei Eingaben auseinander liegen. Ein einfaches, zur Veranschaulichung gut geeignetes Beispiel für L ist der Squared Loss $L(x) = (y - \hat{y})^2$. Dieser wird bei Regressionsproblemen eingesetzt.[2, Kapitel 1.2.1.5] Betrachtet man L nun als $(f^*(x) - f(x;\theta))^2$, erkennt man die entscheidende Eigenschaft der Loss Funktion. Sie misst den Abstand zwischen $f^*(x)$ und $f(x;\theta)$, wodurch eine Zielfunktion entsteht, welche minimiert werden kann, um die Gewichte θ anzupassen.[11, Kapitel 6]

2.2.2 Optimizer

Zum Minimieren der Loss Funktion wird ein sogenannte Optimizer genutzt. Bei NN kommen dabei vor allem Optimizer basierend auf dem Stochastic Gradient Descent Verfahren zum Einsatz. Bei SGD handelt es sich um eine Variante des Gradient Descent, welcher die Ableitung einer Funktion nutzt um diese zu minimieren.[11, Kapitel 5]

2.2.2.1 Gradient Descent

Sei die Funktion $f(x)$ gegeben, dann bezeichnet $f'(x)$ oder $\frac{dy}{dx}$ ihre Ableitung. Diese berechnet die Steigung der Funktion f am Punkt x. Somit kann mithilfe der Ableitung ermittelt werden, wie sich eine kleine Veränderung der Eingabe auf die Ausgabe auswirkt.

$$f(x + \epsilon) \approx f(x) + \epsilon \cdot f'(x) \qquad \text{(Satz von Taylor)} \tag{5}$$

Wird ϵ dabei klein genug gewählt, gilt

$$f(x - \epsilon \cdot sign(f'(x))) < f(x), \tag{6}$$

wobei

$$sign(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{7}$$

gilt. Aufgrund dieser Beobachtungen können wir x entgegengesetzt zum Vorzeichens

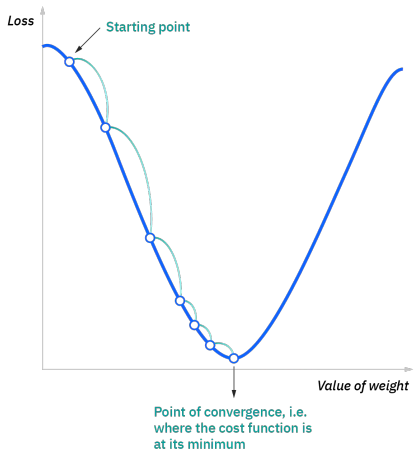


Abbildung 4: Dargestellt ist eine Visualisierung des Minimierungsverfahrens Gradient Descent in einem sehr einfachen Szenario. Auf der Y-Achse ist dabei der Loss und auf der X-Achse, sehr vereinfacht, der Wert der Gewichte θ abgetragen. Der Startpunkt markiert den initialen Wert der Gewichte und das Minimierungsziel stellt das globale Minimum der ausgebildeten Funktion dar.[7]

der Ableitung um ϵ verändern und minimieren dadurch schrittweise f, was in Abbildung 4 dargestellt wird. ϵ wird dabei als Lernrate bezeichnet. Problemstellen für dieses Verfahren sind Maxima, Terrassenpunkte und lokale Minima. An diesen Punkten ist die Ableitung der zu minimierenden Funktion gleich 0, wodurch kein Minimierungsschritt durchgeführt werden kann. Aufgrund dessen ist es kein realistisches Szenario, das globale Minimum beziehungsweise das optimale Resultat zu finden. Gleichzeitig machen jedoch verschiedene Verbesserungen das Verfahren Gradient Descent zunehmend robuster gegenüber den Problemstellen.
Da die Funktion des Neuronalen Netzes mehrere Eingabe hat, müssen alle partiellen Ableitungen $\frac{\partial}{\partial x_i} f(x)$ gebildet werden, um Gradient Descent anwenden zu können. Diese messen, wie sich f am Punkt x ändert, wenn nur die Eingabe x_i verändert wird. Der Gradient $\nabla_x f(x)$ bezeichnet den Vektor, welcher alle partiellen Ableitungen der Funktion $f(x)$ enthält. Dieser zeigt, stellt man sich die Funktion metaphorisch als Berge und Täler vor, bergauf, wohingegen der negative Gradient talabwärts gerichtet ist. Indem man dem negativen Gradienten schrittweise folgt, kann also eine Funktion mini-

miert werden. Dieses Vorgehen nennt sich Gradient Descent, mit dem die nächstbessere Eingabe

$$x' = x - \epsilon \nabla_x f(x) \tag{8}$$

schrittweise berechnen werden kann. ϵ stellt hierbei wieder die Lernrate dar. Mit dieser wird festgelegt, wie schnell in Richtung Tal gelaufen wird. Problemstellen ergeben sich hier, wenn alle Elemente des Gradienten gleich 0 sind.[11, Kapitel 4.3]

2.2.2.2 Stochastic Gradient Descent

Wie zu Beginn von Abschnitt 2.2 erklärt, werden für das Training eines Neuronalen Netzes zahlreiche Eingabebeispiele genutzt. Um aus diesen den Loss zu ermitteln, werden die Ausgaben der Loss Funktion für jedes Eingabebeispiel gemittelt. Die Loss Funktion für m Eingabebeispiele sieht damit wie folgt aus:

$$L(x, y, \theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta) \tag{9}$$

Die Loss Funktion L auf der rechten Seite der Gleichung, berechnet den Loss für ein einzelnes Eingabebeispiel. Diese könnte zum Beispiel der Squared Loss aus Abschnitt 2.2.1 sein. $x^{(i)}$ und $y^{(i)}$ stellen das i-te Eingabebeispiel aus x und die dazugehörige richtige Ausgabe aus y dar. Der Gradient würde über

$$\nabla L(x, y, \theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta) \tag{10}$$

berechnet werden. Ein Problem, welches hierbei aufkommt, ist die Tatsache, dass die Anzahl der Eingabebeispiele m beim Trainieren eines Neuronalen Netzes sehr groß ist. So wurden zum Trainieren der NN dieser Arbeit ungefähr 1000 Eingabebeispiele genutzt. Diese Zahl verdeutlicht, dass die Berechnung des Gradienten für einen Minimierungsschritt zu aufwendig ist. Die Idee von SGD ist nun anstatt bei jedem Minimierungsschritt alle Eingabebeispiele zu betrachten, nur eine Teilmenge in die Berechnung mit einzubeziehen. Diese Teilmenge wird für jeden Schritt gleichverteilt aus allen Eingabebeispielen gewählt und Minibatch oder einfach Batch genannt. Dessen Größe m' liegt zwischen einem und ein paar hundert Eingabebeispielen. Mithilfe eines Minibatches kann nun der Gradient über

$$g(\theta) = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta) \tag{11}$$

geschätzt werden. Mit dieser Schätzung kann dann genauso θ mittels $\theta - \epsilon g$ minimiert werden.[11, Kapitel 5.9] Für einen Minimierungschritt müssen also zunächst die Ausgaben $\hat{y}$ des NN zu einem Minibatch berechnet werden um einen Loss zu erhalten. Dieser Schritt nennt sich Forward Propagation. Anschließend muss der Gradient berechnet

werden, was die Aufgabe der sogenannten Back-Propagation ist. Zum Schluss können mithilfe von SGD die Gewichte angepasst werden.[11, Kapitel 6.5] Das Ausführen dieser drei Schritte für einen Batch wird als Iteration[52] bezeichnet. Ein Zyklus dieser drei Schritte durch alle Eingabebeispiele wird als Epoche[2, Kapitel 1.2.1] bezeichnet.

2.3 Overfitting

Bei dem eben vorgestellten Training eines Neuronalen Netzes liegt das Ziel darin, den Loss des NN auf den Trainingsdaten zu minimieren. Doch die Trainingsdaten stellen meist nur einen sehr kleinen Ausschnitt des Definitionsbereichs von $f^*(x)$ dar. Das heißt den Loss auf den Trainingsdaten zu minimieren heißt nicht direkt $f^*(x)$ besser zu approximieren.[2, Seite 25–26] Overfitting beschreibt den Fall indem der Unterschied zwischen dem Loss auf den Trainingsdaten und dem Loss auf den Testdaten zu groß wird.[11, Seite 110]

2.4 Datensätze

Die Menge der zur Verfügung stehenden Eingabebeispiele mit dazugehörigen korrekten Ausgaben wird als Datensatz bezeichnet. Um ein Neuronales Netz zu trainieren wird dieser in drei Datensätze aufgeteilt. Einen Teil der Daten wird dem Trainings-Datensatz zugeordnet, welcher zum Training, also für das Anpassen der Parameter θ des NN, genutzt wird. Weitere Daten erhält der Validations-Datensatz. Dieser wird zum Evaluieren verschiedener Hyperparameter-Kombinationen oder zum Erkennen von Overfitting verwendet. Die übrigen Daten fließen in den Test-Datensatz, welcher ausschließlich für das Testen des trainierten NN verwendet wird.[2, Seite 178] Dadurch hat der Test Datensatz keinen Einfluss auf das Neuronale Netz, wodurch die Leistung des NN auf zuvor nicht gesehen Eingaben getestet werden kann.[11, Seite 108]

2.5 Daten Augmentation

Wie gerade besprochen werden zum Trainieren von Neuronalen Netzen viele Eingabebeispiele mit den dazugehörigen, bekannten, korrekten Ausgabewerten benötigt. Allerdings stehen nicht immer genügend solcher Datenpaare zu einer konkreten Problemstellung zur Verfügung. Bei der Daten Augmentation werden durch Transformation der vorhanden Eingabebeispiele neue Datenpaare automatisch erzeugt. Diese gehören jedoch nicht zum Datensatz dazu. Da sie meist einfach zu berechnen sind, werden sie erst auf ein Eingabebeispiel angewandt, wenn es in das NN eingegeben wird. Wichtig bei der Wahl der Daten Augmentation ist nur Transformationen zu verwenden, welche den korrekten Ausgabewert erhalten. Daten Augmentationen sind für diese Arbeit von großer Bedeutung, da sie gerade im Bereich der Bildverarbeitung besonders gut funk-

tionieren. Beispiele für Augmentationen wären hierbei die Rotation oder das Spiegeln des Eingabebildes.[2, Kapitel 8.3.4]

2.6 Convolution

In 2.1 wurde die grundlegende Funktion einer Schicht, nämlich die Lineare Schicht, eines Neuronalen Netzes vorgestellt. Doch es gibt noch weitere Funktionen für die Schichten eines NN. Unter anderem die Convolution Schicht, welche für diese Arbeit von besonderer Bedeutung ist. Die Idee hinter dieser Schicht ist es, lokale Eigenschaften in der Eingabe zu finden und diese zu abstrakteren, höher geordneten Eigenschaften zu kombinieren. Um das zu realisieren bekommen die Neuronen der Convolution Schicht nicht wie in der Linearen Schicht die ganze Eingabe auf einmal zu sehen, sondern nur einen Teil auf einmal. Das zwingt die Neuronen dazu lokale Informationen zu verarbeiten, was beispielsweise besonders gut dazu geeignet ist, Muster in der Eingabe zu erkennen. Abstraktere Eigenschaften entstehen durch mehrere hintereinander geschaltete Convolution Schichten.[23] Da in dieser Arbeit für die Eingabe Bilder verwendet werden, welche durch eine zweidimensionale Matrix dargestellt werden können, wird im Folgenden die zweidimensionale Convolution erklärt und die Eingabe als Image I bezeichnet.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, i+n) K(m,n), \tag{12}$$

In Gleichung (12) ist die Kreuzkorrelation gezeigt. Diese stellt zwar nicht exakt die Convolution Operation dar, berechnet jedoch dasselbe und ist in vielen Bibliotheken für Neuronale Netze anstatt der Convolution implementiert. An dieser Stelle soll sie erklärt werden, da sie intuitiver zu verstehen ist. Im Folgenden wird sie aus Gründen der Einfachheit auch als Convolution bezeichnet. $K \in \mathbb{R}^{m \times n}$ aus Gleichung (12) ist der sogenannte Kernel, der eine Matrix darstellt und lernbar ist. Das heißt er ist in θ enthalten und wird zur Minimierung der Loss Funktion angepasst. Der Kernel ist viel kleiner als die Eingabematrix und kann somit an verschieden Stellen der Eingabe lokal angewandt werden. Wie aus 12 hervorgeht, erzeugt jede dieser Anwendungen des Kernels einen Wert in der Matrix S, der sogenannten Feature Map. Wie Eingabe, Kernel und Feature Map zusammenhängen ist in Abbildung 5 dargestellt. Zur Verdeutlichung der Convolution Operation wird die Berechnung eines Eintrags der Feature Map gezeigt.

$$\begin{aligned} S(0,0) = & I(0,0)K(0,0) + I(0,1)K(0,1) + I(0,2)K(0,2) \quad + \\ & I(1,0)K(1,0) + I(1,1)K(1,1) + I(1,2)K(1,2) \quad + \\ & I(2,0)K(2,0) + I(2,1)K(2,1) + I(2,2)K(2,2) \end{aligned} \tag{13}$$

Die Berechnung geht aus Gleichung (12) hervor. Um mehrere Eigenschaften auf einer Eingabe zu erkennen, werden in einer Convolution Schicht mehrere Feature Maps mit

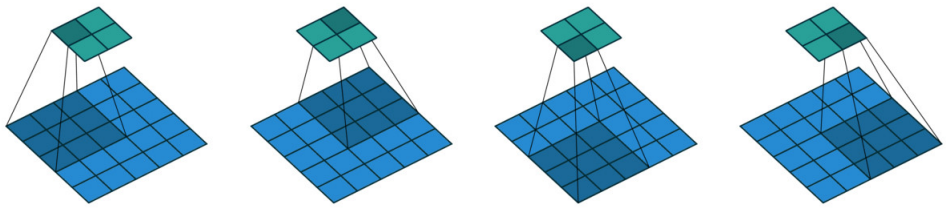


Abbildung 5: In der Abbilding wird gezeigt, welche Werte der Eingabematrix zu einem Wert der Feature Map beitragen. Die Größe des in dunkelblau dargestellten Kernels ist 3×3. Auffällig ist hier, dass i und j, also der Kernel, in der Eingabematrix jeweils um zwei Matrixeinträge verschoben werden. Das führt zu einer Verkleinerung der Feature Map.[41]

eigenem Kernel benutzt. Die Ausgabe der Schicht ist damit eine dreidimensionale Matrix, bestehend aus mehreren zweidimensionalen Feature Maps. Um zu verstehen, was

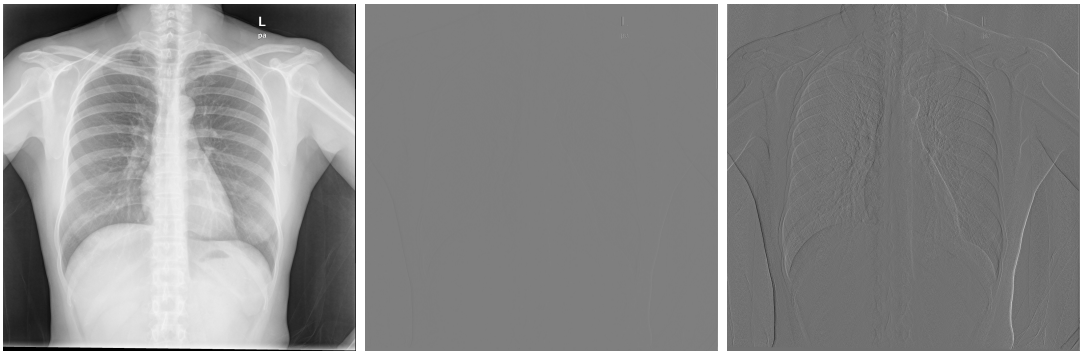


Abbildung 6: **Links:** Das Originalbild[19]. **Mitte:** Die Feature Map, die entsteht, wenn auf das Originalbild eine Convolution mit Kernel $[1, -1]$ angewandt wird. **Rechts:** Die Feature Map mit erhöhtem Kontrast.

die Convolution Schicht möglich macht, kann Abbildung 6 betrachtet werden. Um das mittlere sowie das rechte Bild aus dem Originalbild zu erhalten wurde eine Convolution mit dem Kernel $[1, -1]$ auf das Originalbild angewandt. Wie man erkennen kann, ist das dadurch erkannte Feature Kanten. Dieses könnte dazu genutzt werden, Objekte auf einem Eingabebild zu identifizieren. Wichtig zu verstehen ist, dass diese Operation genauso von einer Linearen Schicht durchgeführt werden kann. Dies würde jedoch eine Gewichtsmatrix W der Dimension Bildbreite $\times$ Bildhöhe $\times$ (Bildbreite $- 1$) $\times$ Bildhöhe benötigen. Im Vergleich braucht die Convolution eine Gewichtsmatrix aus zwei Elementen, den Kernel. Das verdeutlicht, dass die grundlegende Funktion aus Abschnitt 2.1 zwar jegliche Operationen lernen kann, jedoch nicht immer sinnvoll ist. Durch andere Schichtenfunktionen kann ein bestimmtes sinnvolles Verhalten erzwungen werden und muss nicht zunächst gelernt werden, wobei unnötiger Rechnenaufwand eingespart werden kann.[11, Kapitel 9]

2.6.1 Pooling

Pooling ist eine weitere Funktion, welche für eine Schicht eines Neuronalen Netzes genutzt werden kann. Sie ist unter dem Abschnitt Convolution aufgeführt, da sie typischerweise gemeinsam mit Convolution Schichten eingesetzt wird. Ihre Aufgabe besteht darin, die jeweiligen Eingabewerte durch eine Zusammenfassung aus aktuellem Eingabewert und benachbarten Eingabewerten zu ersetzen. Dabei können die direkt benachbarten Werte, aber auch weiter entfernte Nachbarn betrachtet werden. Stellt man sich die Eingabe wieder als Bild vor, dann beschreiben die Nachbarn eines Pixels seine umliegenden Pixel. Um eine Zusammenfassung aus mehreren Werten zu generieren, gibt es zwei Optionen. Das sogenannte Average Pooling berechnet eine Zusammenfassung durch das Mitteln aller betrachteten Werte. Das Max Pooling wählt das Maximum der betrachteten Werte. Durch das Überspringen einzelner Eingabewerte kann beim Pooling wie bei der Convolution die Eingabe verkleinert werden, was in Abbildung 5 zu sehen ist. Die Pooling Schicht ist eine der wenigen Schichten, welche keine Aktivierungsfunktion nutzt.[11, Kapitel 9.3]

2.7 Residuale Netzwerke

Neuronale Netze, in denen mehrere Convolution Schichten enthalten sind, werden Convolutional Neuronale Netze genannt. Zunächst sah es danach aus, dass tiefere CNNs bessere Ergebnisse in der Approximation von $f^*(x)$ erreichen. Mit tieferen CNNs sind hier CNNs mit mehr Convolution Schichten gemeint. Doch es hat sich gezeigt, dass

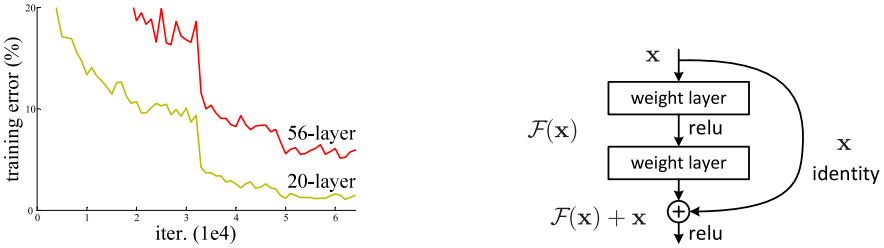


Abbildung 7: **Links:** Auf dem abgebildeten Graph ist der Loss auf der Y-Achse und die Iterationen auf der X-Achse abgetragen. Man erkennt, dass das tiefere CNN einen höheren Loss aufweist. **Rechts:** In der Grafik ist ein Residual Block mit der sogenannten Shortcut Verbindung abgebildet. Dieser stellt die Grundlage eines Residualen Netzes dar.[15]

es einen Punkt gibt, an dem tiefere CNNs wieder schlechter werden. In Abbildung 7 (links) ist dazu ein Beispiel zu sehen. Stellt man sich ein flaches und ein tiefes CNN vor, wobei das tiefere Netz eine schlechtere Approximation von $f^*(x)$ erreicht, dann lässt sich dabei ein Widerspruch erkennen. Das tiefere CNN könnte theoretisch in den übereinstimmenden Schichten die selben Gewichte lernen und in den zusätzlichen Schichten das Ergebnis nicht mehr verändern. Dadurch würde sowohl das Tiefe, als auch das Fla-

che CNN die selbe Funktion berechnen. Das nicht verändern des Ergebnisses stellt im mathematischen Sinne eine Identitätsabbildung dar. Dieses Verhalten zeigt sich, zumindest in absehbarer Zeit, jedoch nicht. Um dieses Problem zu lösen, führten He et al.[15] das Residuale Lernen für die Bilderkennung beziehungsweise allgemein für CNNs ein. Dazu entwickelten sie einen Residualen Baustein, welcher in Abbildung 7 (rechts) zu sehen ist. Dieser besteht aus Schichten mit lernbaren Parametern, wobei die Ausgabe einer früheren Schicht x am Ende des Blocks auf dessen Ausgabe aufaddiert wird. Aus diesem Grund lernen die Schichten eines Residualen Blocks nicht $\mathcal{F}(x) := \mathcal{H}(x)$ sondern $\mathcal{F}(x) := \mathcal{H}(x) - x$ zu berechnen. Die Autoren vermuten, dass das Residuale Netz dadurch besser lernen kann. Außerdem kann die Identitätsabbildung wesentlich einfacher erreicht werden. In Abbildung 8 ist der Vergleich zwischen einem 34 Schichte

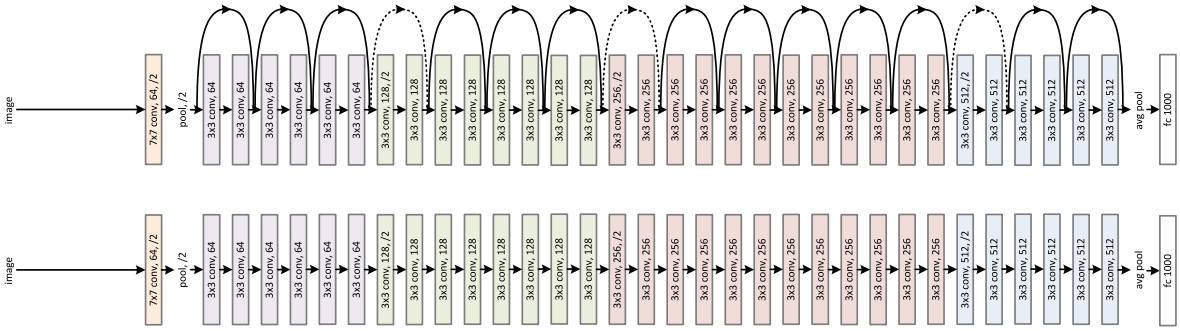


Abbildung 8: **Oben:** Ein Residuales Netz, welches an den Shortcut Verbindungen erkannt werden kann. **Unten:** Ein gewöhnliches CNN der Tiefe 34, wobei die einzelnen Schichten als Rechtecke dargestellt werden. Die Pfeile beschreiben den Informationsfluss.[15]

tiefen einfachen CNN und einem Residualen Netz zu sehen. Dieses zeichnet sich durch die sogenannten Shortcut Verbindungen aus, welche die Addition $\mathcal{F}(x) + x$ darstellen und durch schichtenüberspringende Pfeile gekennzeichnet werden. 7×7 beziehungsweise 3×3, was in den Convolution Schichten zu lesen ist, steht für die Größe des Kernels der jeweiligen Schicht. Die Werte $64, 128, 256, 512$ wiederum stehen jeweils für die Anzahl der Feature Maps einer Schicht. Problemstellen sind dort zu finden, wo eine Shortcut Verbindung zwei Schichten unterschiedlicher Größen verbindet. Diese werden durch eine Convolution Schicht mit einem 1×1 Kernel und der entsprechenden Anzahl an Feature Maps aufgelöst.[15] Im Folgenden sollen Residuale Netzwerke mit ResNet gefolgt von ihrer Tiefe abgekürzt werden. In der Arbeit wird des Weiteren von einem ResNeXt gesprochen. Dies ist ein weiteres Residuales CNN, das einen neuen, komplexeren Residualen Block einführt. In diesem werden parallele Schichtenberechnungen durchgeführt, welche vor dem Addieren der Shortcut Verbindung kombiniert werden.[54]

2.8 Feature Pyramid Netzwerk

Im folgenden Abschnitt wird ein Objektdetektor eingeführt, welcher Objekte auf Bildern erkennen kann. Um diesen dabei zu unterstützen, Objekte verschiedener Größe zu erkennen, werden so genante Feature Pyramids genutzt. Diese bestehen aus verschiedenen Repräsentationen der Eingabe mit unterschiedlichem Detail- beziehungsweise Abstraktionsgrad. In der Vergangenheit wurden diese Feature Pyramids aus Bildpy-

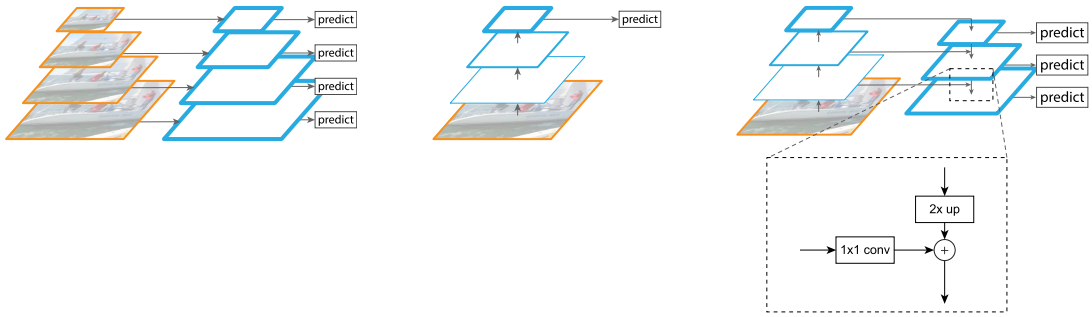


Abbildung 9: Die Abbildung zeigt verschiedene Feature Pyramids. Dabei ist die Eingabe orange und die berechneten Feature Maps blau umrandet dargestellt. Der jeweilige Pfeil auf predict gibt an, welche Feature Maps für die weitere Verarbeitung genutzt werden. **Links**: Hier ist die Feature Pyramid basierend auf Bildpyramiden abgebildet. **Rechts:** Eine Darstellung des Feature Pyramid Netzwerks. **Mitte:** Die hier abgebildete Pyramide ist keine Feature Pyramid, da hier nur die Feature Map einer Ebene genutzt wird.[26]

ramiden aufgebaut, wozu ein Beispiel links in Abbildung 9 zu sehen ist. Bei dieser wurden als Input verschiedene Skalierungen der Eingabe genutzt. Später machte die Nutzung von CNNs zum Generieren der Feature Maps, welche robuster gegenüber der Größe von Eigenschaften sind, eine Verwendung von einer einzigen Feature Map möglich. Das ressourcensparende Vorgehen ist in Abbildung 9 (mitte) zu sehen. Dabei ist zu erkennen, dass die einzelnen Feature Maps immer kleiner werden. Dies kann durch die in Abbildung 5 erwähnte Technik erreicht werden. Trotz der Robustheit der CNNs besitzt eine Feature Pyramid jedoch weiterhin mehr Information, da jede Feature Map verschiedener Detailgrade relevante und komplementäre Information enthält. Lin et al.[26] nutzen bestehende CNNs um eine Feature Pyramid zu generieren, welche genauso schnell berechnet werden kann wie die finale Feature Map aus Abbildung 9 (mitte). Das entwickelte Feature Pyramid Network ist in Abbildung 9 (rechts) gezeigt. Um dieses zu generieren werden zunächst Feature Maps eines CNNs unterschiedlicher Größe auswählt. Besitzt das CNN mehrere Schichten einer Größe, wird die Ausgabe der jeweils letzten Schicht einer Größe gewählt. Um anschließend die für die weitere Verarbeitung genutzten Feature Maps zu erhalten, werden die ausgewählten Feature Maps miteinander kombiniert. Dabei wird eine Feature Map mit der jeweils abstrakteren,

welche zuvor mit der nächster Nachbar Interpolation vergrößert wurde, elementweise addiert. Dieser Vorgang ist in Abbildung 9 (rechts) zu sehen.[26]

2.9 Faster R-CNN

Faster R-CNN[35] ist ein Objektdetektionsnetzwerk, welches den Vorläufer Fast R-CNN[10] um das Region Proposal Netzwerk aus Abschnitt 2.9.1 erweitert. Ein sogenannter Objektdetektor erhält ein Bild als Eingabe und gibt Rechtecke, definiert durch beispielsweise (x, y, w, h), um Objekte mit dazugehörigen Objektklassen aus. Eine Objektklasse kann hierbei beispielsweise ein Mensch oder ein Auto sein. (x, y) definiert die linke obere Ecke des Rechtecks um ein Objekt und (w, h) seine Höhe und Breite. Diese Rechtecke werden Boundingboxen genannt, zu deutsch Begrenzungsrahmen. Fast R-CNN erhält neben dem Bild noch sogenannte Objektvorschläge als Eingabe. Das sind rechteckige Bildausschnitte, welche ein Objekt enthalten können und vom NN genauer untersucht werden. Im Falle von Faster R-CNN[35] werden diese Vorschläge von dem RPN aus Abschnitt 2.9.1 berechnet. In Abbildung 10 ist der Aufbau des Fast R-CNN

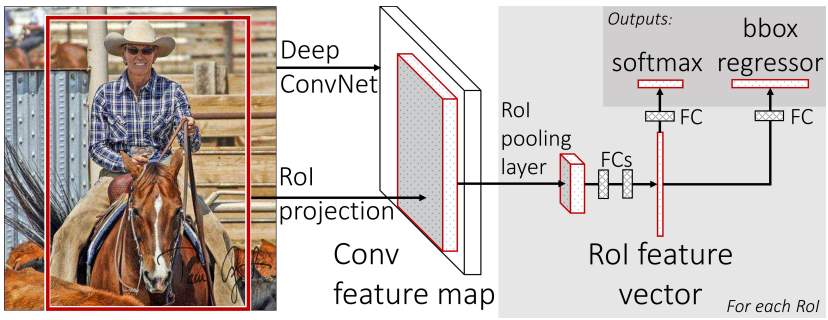


Abbildung 10: Die Abbildung zeigt den Aufbau von Fast R-CNN, dabei kann die Aufteilung in zwei Ausgabeschichten verdeutlicht werden. Der grau hinterlegte Abschnitt des NN wird auf jeden Objektvorschlag angewandt.[10]

Objektdetektors dargestellt. In der Grafik wird links das Eingabebild und ein exemplarischer Objektvorschlag gezeigt. Folgt man der Abbildung, wird das Bild zunächst durch ein CNN verarbeitet. Hierbei ist nur die finale Feature Map auf Abbildung 10 eingezeichnet. Anschließend berechnet die sogenannte Region of Interest Pooling Schicht aus jedem Objektvorschlag eine kleinere Feature Map fester Größe $H \times W$. Dazu wird der $h \times w$ große Ausschnitt eines Objektvorschlages aus der vom CNN ausgegebenen Feature Map in ein Gitter der Größe $\frac{h}{H} \times \frac{w}{W}$ eingeteilt. Anschließend wird aus jeder Zelle der größte Wert ausgewählt, wodurch die $H \times W$ große Feature Map entsteht. Jede dieser $H \times W$ großen Feature Maps wird dann separat durch weitere Schichten weiterverarbeitet. Dazu werden zunächst einige Lineare Schichten verwendet, welche in Abschnitt 2.1 genauer beschreiben sind. Deren Ausgabe wird dann von zwei unterschiedlichen Ausgabeschichten verwendet. Diese berechnen die Klasse und die Boundingbox zu dem gerade verarbeiteten Objektvorschlag. Die Klassen bestehen dabei aus

K vorgebenden Objekttypen (zum Beispiel Mensch oder Auto) und einer Hintergrundklasse. Wie in Abschnitt 2.1.1 erklärt, kann die Softmax Schicht verwendet werden, um die Wahrscheinlichkeit der Zugehörigkeit des erkannten Objekts zu einer der $K+1$ Klassen auszugeben. Um die Boundingboxen zu erhalten, wird eine Lineare Schicht mit $4 \cdot K$ Ausgaben verwendet. Diese berechnet für jede Klasse $t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$, was deren Boundingbox definiert. Um die beiden verschiedenen Ausgaben trainieren zu können, wird die Loss Funktion

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v) \tag{14}$$

minimiert. Diese setzt sich aus einer Loss Funktion $L_{cls}(p, u) = -log\, p_u$ für die Ausgabe der Klassen und einer Loss Funktion $L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t^u_i - v_i)$ für die Ausgabe der Boundingboxen zusammen. p_u steht dabei für die von der Softmax Schicht berechnete Wahrscheinlichkeit zur Korrekten Klasse u und t^u für die ausgegebene Boundingbox zu u. $v = (v_x, v_y, v_w, v_h)$ stellt die tatsächlich korrekte Boundingbox dar. Des weiteren ist $[u \geq 1]$ gleich 1, wenn die Bedingung wahr ist und andernfalls 0. Dabei ist $u = 0$ die Hintergrundklasse.

$$smooth_{L_1} = \begin{cases} 0,5x^2 & wenn\, |x| < 1 \\ |x| - 0,5 & sonst \end{cases}. \tag{15}$$

λ soll die beiden Loss Funktionen balancieren.[10]

2.9.1 Region Proposal Netzwerk

Regions- beziehungsweise Objektvorschläge sind Bereiche einer Eingabe, welche ein Objekt enthalten könnten. Manche Objektdetektoren sind auf diese Vorschläge angewiesen. Diese müssen jedoch in einem separaten Schritt berechnet werden, was solche Objektdetektoren langsam macht. Ren et al.[35] schlagen deshalb ein Region Proposal Netzwerk vor, welches das CNN eines Detektors mitbenutzt, um Regionsvorschläge zu generieren. In diesem wird ein sogenanntes Schiebefenster der Größe $n \times n$ über die Ausgabe der letzten Schicht des CNNs des Detektors geschoben. Dieser Vorgang ist in Abbildung 11 visuell dargestellt. Wird dabei ein Feature Pyramid Netzwerk aus Abschnitt 2.8 benutzt, wird ein Schiebefenster für jede Feature Map der Feature Pyramid verwendet. Das Schiebefenster generiert an jeder Position eine $n \times n$ große Matrix, welche in einem Vektor, dem intermediate layer, zusammengefasst wird. Dabei wird jeder intermediate layer einzeln weiterverarbeitet. Für die weitere Verarbeitung spielen ferner auch die in Abbildung 11 (rechts) zu sehenden k Anchorboxen eine Rolle. Dies sind Rechtecke, welche durch Größe und Seitenverhältnis definiert werden und Hyperparameter darstellen. Die Weiterverarbeitung erfolgt nun durch das gemeinsa-

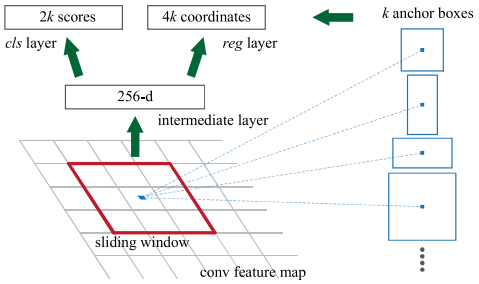


Abbildung 11: Die Abbildung zeigt das Schiebefenster eines RPN auf einer Feature Map eines CNN. Die Werte, welche in dem Schiebefenster liegen, werden zunächst in dem intermediate layer zusammengefasst und anschließend durch den cls layer und den reg layer weiterverarbeitet. Auf der Grafik ist außerdem zu erkennen, wie die Anchorboxen mittig auf das Schiebefenster gelegt werden.[35]

me Verarbeiten von intermediate layer und Anchorboxen durch zwei separate Lineare Schichten, die Box Regression Schicht und die Box Klassifikation Schicht. Diese werden genutzt, um für jede der k Anchorboxen an jeder Schiebefensterposition einen Regionsvorschlag zu berechnen. Dieser setzt sich aus einem konkreten Bildausschnitt und einer Wahrscheinlichkeit, ob in diesem Bildausschnitt ein Objekt vorhanden ist, zusammen. Der Bildausschnitt wird von der Box Regression Schicht berechnet, wobei sein Zentrum die Schiebefensterposition darstellt und seine Größe relativ zur jeweiligen Anchorbox berechnet wird. Die Box Klassifikation Schicht berechnet die Wahrscheinlichkeit für die Klassen Objekt und Nicht-Objekt. Umgesetzt wird dieses Verhalten durch eine Convolution Schicht mit einem Kernel der Größe $n \times n$ und zwei darauf folgenden, parallel angewandten Convolution Schichten mit einem Kernel der Größe 1×1.

An dieser Stelle folgt eine Ergänzung zu Abschnitt 2.9, denn durch das RPN muss Faster R-CNN[35] speziell trainiert werden. Der Grund dafür liegt in dem geteilten CNN, dem Backbone[16]. Da Faster R-CNN Objektvorschläge benötigt, wird zunächst ausschließlich das RPN trainiert, wobei die Regionsvorschläge festgehalten werden. Mit diesen wird anschließend Fast R-CNN eigenständig trainiert. Dadurch entstehen vorerst zwei Netze mit unterschiedlichen Parametern θ. Nun wird das RPN ein zweites Mal trainiert, wobei die Gewichte des separaten Fast R-CNN Trainings übernommen und nicht mehr geändert werden. Dadurch werden nur die zusätzlichen Schichten des RPNs angepasst. In einem letzten Trainingsschritt werden mit dem soeben erhaltenen Netzwerk die Linearen Schichten des Fast R-CNN angepasst, wozu alle weiteren Schichten des NN nicht mehr verändert werden.[35]

2.10 Mask R-CNN

Mask R-CNN[16] erweitert Faster R-CNN[35] um die Ausgabe von Segmentierungsmasken und stellt damit ein Instanz Segmentierungs Neuronales Netz dar. Die semantische

Segmentierung bezeichnet dabei die Problemstellung, jeden Pixel eines Bilds einzeln einer Klasse zuzuweisen. Dadurch ergeben sich Masken, welche man durch das Darstellen der Ausgabe als Bild visualisieren kann. Abbildung 12 zeigt den Aufbau von Mask

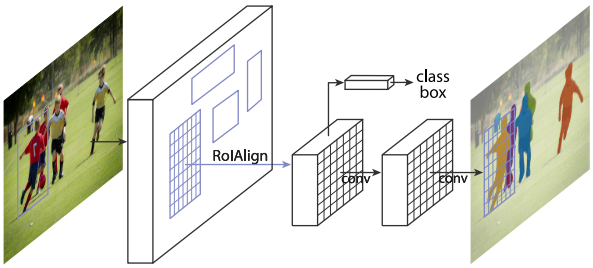


Abbildung 12: Dargestellt ist der Aufbau von Mask R-CNN, wobei die Ähnlichkeit zu Faster R-CNN auffällt. RoI Align ersetzt dabei das RoI Pooling und zusätzliche Convolution Schichten ermöglichen die Ausgabe von Segmentierungsmasken.[16]

R-CNN. Dieses erweitert Faster R-CNN nach dem RoI Pooling um einen parallelen Berechnungspfad. Dieser berechnet, basierend auf weiteren Convolution Schichten, für jede Klasse eine binäre Maske für die aktuelle Region of Interest. Anschließend wird die Maske der wahrscheinlichsten Klasse ausgewählt. Die essentielle Idee hierbei ist in einem zuvor durch das RPN ausgewählten Bildausschnitt ein einziges Objekt zu segmentieren. Dies ist der kritische Unterschied zu vorherigen Instanz Segmentierungs Netzen. Neben dieser zentralen Anpassung wurden noch weitere Änderungen, wie die RoI Align Schicht, an Faster R-CNN vorgenommen. Diese sollen jedoch nicht in dieser Arbeit thematisiert werden und können He et al.[16] entnommen werden.[16]

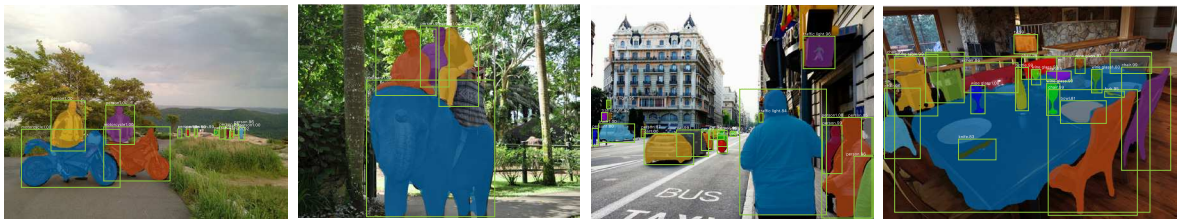


Abbildung 13: Abgebildet sind die von Mask R-CNN ausgegebenen Boundingboxen und Segmentierungsmasken mit dazugehöriger Klasse zu vier Eingabebildern.[16] Für die Ausgabe von Faster R-CNN[35] müssen sich die Masken weggedacht werden.

2.11 Metriken

Bei der Evaluation Neuronaler Netze werden die Ausgaben eines NN zu verschiedenen Eingabebeispielen mit korrekten Informationen verglichen. Dabei stehen Eingabebeispiele und dazugehörige soll Ausgaben in einem Testdatensatz (siehe Abschnitt 2.4) zur Verfügung. Verschiedene Metriken messen diesen Vergleich quantitativ.

2.11.1 Klassifikation

Um die Zuweisung zu Klassen, also die Aufgabe der Klassifikation, zu evaluieren, werden vor allem die Metriken Accuracy, Precision, Sensitivity, Specificity und der F1-Score verwendet. Die englischen Begriffe werden hier verwendet, da sie sich etabliert haben und somit eine bessere Vergleichbarkeit mit andern Veröffentlichungen gewährleisten. Um die vorgestellten Metriken zu berechnen werden die Ergebnisse der Klas-

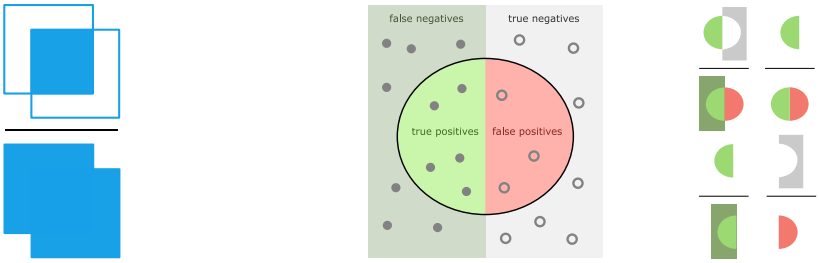


Abbildung 14: **Links:** Der Quotient der blau gefärbten Flächen stellt den Jaccard Score, beziehungsweise die Intersection over Union dar.[38] **Rechts:** Von links nach rechts und oben nach unten die vier Klassen, in die eine Klassifikation eingeordnet werden kann: Falsch positiv, Richtig negativ, Richtig positiv und Falsch negativ. Daneben ist die Berechnung der Metriken Accuracy, Precision, Sensivity und Spcificity in gleicher Reihenfolge visuell dargestellt.[49]

sifikation zunächst in vier Gruppen aufgeteilt. Diese sind in Abbildung 14 (rechts) zu sehen. Deren Bedeutung soll anhand der Klassifikation in Kardiomegaliepatient und gesunder Patient verdeutlicht werden. Hierbei wäre eine richtig-positive (TP) Klassifikation die Klassifikation eines Kardiomegaliepatienten als Kardiomegaliepatient. Eine falsch-positive (FP) Klassifikation wäre die Klassifikation eines gesunden Patienten als Kardiomegaliepatient. Dies lässt sich auf die beiden weiteren Klassen richtig negativ (TN) und falsch negativ (FN) übertragen.[33] Die Abkürzungen stammen dabei von den jeweiligen englischen Begriffen.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad Precision = \frac{TP}{TP + FP},$$

$$Sensivity = \frac{TP}{TP + FN}, \quad Specificity = \frac{TN}{FP + TN}, \tag{16}$$

$$F1 = \frac{2 \cdot Precision \cdot Sensivity}{Precision + Sensivity}$$

Die sich daraus ergebenden, genannten Metriken sind sowohl in Gleichung (16) als auch visuell in Abbildung 14 dargestellt. Die Accuracy misst also den Anteil der richtig Klas-

sifizierten Eingaben an allen Eingaben. Die weiteren Metriken ergeben sich daraus.[30, 33]

2.11.2 Boundingboxen

Um die Korrektheit einer Boundingbox messen zu können, werden die Metriken Jaccard Score, auch Intersection over Union genannt, und Dice Score genutzt.[33] Beide Metriken sind in Gleichung (17) gezeigt.

$$Jaccard = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}, \qquad Dice = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{17}$$

Dabei stellt TP bei einer Boundingbox die Fläche der Überschneidung von zu evaluierender und korrekter Boundingbox dar. TP+FN ist die Fläche der korrekten Boundingbox und TP+FP die Fläche der zu evaluierenden Boundingbox. Die Berechnung des Jaccard Scores ist zusätzlich in Abbildung 14 (links) visuell dargestellt. Der Dice Score unterscheidet sich nur marginal vom Jaccard Score. Dieser gewichtet schlichtweg die Überschneidung von zu evaluierender und korrekter Boundingbox zweifach.

2.11.3 Korrelationskoeffizient

Um zu messen, wie stark zwei Merkmale X und Y zusammenhängen, kann zunächst ein Streudiagramm genutzt werden. In diesem sind die Punkte (x_i, y_i) eingetragen. Wären X und Y gleich, ergäbe das Streudiagramm eine Gerade, da X und Y entsprechend gleich wachsen oder schrumpfen. Es kann also festgehalten werden, dass je genauer das Streudiagramm der Merkmale eine Gerade formt, desto stärker die beiden Merkmale zusammenhängen. Um diese Beobachtung in einer einfachen Zahl auszudrücken wird der Empirische Korrelationskoeffizient

$$r = r_{XY} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{18}$$

verwendet. Dabei stehen $\overline{x}$ und $\overline{y}$ jeweils für die Mittelwerte der Merkmale X und Y. Das Ergebnis des Korrelationskoeffizienten kann positiv oder negativ sein, je nachdem ob die Gerade im Streudiagramm steigt oder fällt. Man spricht entweder von einer positiven oder einer negativen Korrelation. Des weiteren gilt $-1 \leq r \leq 1$ und damit $0 \leq |r| \leq 1$. Je näher $|r|$ an 1 liegt, desto stärker ist der vorliegende Zusammenhang. Nach Fahrmeir et al.[8, Kapitel 3.4.1] stellt für genaue Messungen ein $|r| < 0,5$ eine schwache, ein $|r| \geq 0,8$ eine starke und Werte dazwischen eine mittlere Korrelation dar.[8, Kapitel 3.4.1]

3 Verwandte Arbeiten

3.1 Automatische Kardiomegalie Erkennung

In bisherigen Arbeiten zur automatischen Erkennung von Kardiomegalie auf Röntgenthorax-Aufnahmen mithilfe von Deep Learning wurden bisher zwei Typen von Neuronalen Netzen eingesetzt. Zum einen sind das Netze, welche die Kardiomegalie oder eine Reihe weiterer Erkrankungen direkt auf einer Röntgenthorax-Aufnahme erkennen. Zum andern sind das Netze, welche den HTQ auf der Basis von Segmentierungsmasken zu Herz und Lunge berechnen. Ein großer Vorteil des zweiten Ansatzes liegt in der besseren Erklärbarkeit der Ergebnisses. Anstatt ein Eingabebild schlichtweg zu klassifizieren, werden die Breite des Herzens und der Lunge ausgeben. Mit Hilfe dieser Messwerte kann dann, wie in Abschnitt 1.1 beschreiben, das Vorliegen von Kardiomegalie ermittelt werden. Sogancioglu et al.[44] verglichen die beiden Ansätze miteinander und konnten zudem bessere Ergebnisse bei der Kardiomegalie Erkennung durch Segmentierung von Herz und Lunge feststellen. Dabei erreicht ihr segmentierungsbasierter Ansatz eine Sensitivity von 97% und eine Specificity von 90%. Ihr klassifizierungsbasierter Ansatz erreicht dagegen nur eine Sensitivity von 81% und eine Specificity von 89%. Ein wichtiger Unterschied zwischen den Methoden ist, dass für das Training des klassifizierungsbasierten Ansatzes über 100-mal so viele Röntgenaufnahmen nötig waren. Für die Röntgenaufnahmen, welche für die Evaluation genutzt wurden, notierten jeweils zwei Radiologen den Querdurchmesser des Herzens und der Lunge. Für die zuvor vorgestellte Auswertung wurden jedoch nur die Annotationen eines erfahreneren Radiologen verwendet. Die Annotationen des zweiten Radiologen wurden genauso wie die NN anhand der Maße des erfahreneren Radiologen ausgewertet. Dabei erreicht der Radiologe eine Sensitivity von 91% und eine Specificity von 95%.[44] Noch bessere Ergebnisse konnte die automatische Kardiomegalie Erkennung von Li et al.[24] erzielen. Die Autoren verfolgten dabei den Ansatz der Segmentierungsmasken. Dafür nutzen Sie ein angepasstes U-Net[37], ein Neuronales Netz, welches für die Bildsegmentierung konstruiert wurde. Dieses geht in zwei Schritten vor, um die Aufgabenstellung der Segmentierung zu lösen. Zunächst berechnet ein CNN verschiedene Feature Maps, welche nacheinander immer abstraktere Eigenschaften des Eingabebildes repräsentieren. In einem zweiten Schritt kombiniert das Netz abstraktere mit weniger abstrakten Feature Maps, um wieder räumliche Informationen in die erkannten abstrakten Eigenschaften herzustellen. Zum Trainieren ihres angepassten U-Net sammelten die Autoren 5000 Röntgenaufnahmen des Thorax, zu denen Expertenmasken für Herzen und Lungenflügel erstellten. Da die Bilder verschiedenste Intensitäten hatten, wurde sie zuvor normalisiert. Daten Augmentation (siehe Abschnitt 2.5) wurde jedoch nicht benutzt, da diese in Tests die Ergebnisse nicht verbessern konnte. Allerdings wurde ein Nachbearbeitungsschritt angewandt, der die vom U-Net ausgegebenen Masken verbessert.

Tabelle 1: Evaluationsergebnisse von Li et al.[24]

	Accuracy	Sensitivity	Specificity	Differenz	KK (r^2)
Deep Learning	$95,3\%$	$97,2\%$	$92,7\%$	$0,0004 \pm 0,0133$	$0,965$
Radiologe	$93,8\%$	$91,4\%$	$97,1\%$	-0.0083 ± 0.0187	$0,929$

Die Zeile Deep Learning zeigt die Evaluationsergebnisse des NN und die Zeile Radiologe die eines unabhängigen Radiologen. KK kürzt den Korrelationskoeffizienten ab, der hier quadriert dargestellt ist (Bestimmtheitsmaß).[24]

Dieser ist nötig, da bei Segmentierungsnetzen oftmals die Ränder von Masken ungenau sind. Das liegt daran, dass diese NN jedem Pixel eine Klasse zuweisen, weshalb die Klassen bei jedem Pixel miteinander konkurrieren. Li et al.[24] nutzten als Nachbearbeitungsschritt ein von Krähenbühl und Koltun[20] vorgestelltes Fully Connected Conditional Random Field. Zum Testen des trainierten Netzes wurden noch einmal 500 Röntgenaufnahmen des Thorax gesammelt. Anschließend wurden auf diesen von vier Radiologen jeweils der Querdurchmesser von Herz und Lunge bestimmt. Durch Mitteln sowie einen Konsensmechanismus wurden die Werte von drei Radiologen zu einem Vergleichswert für die Evaluation kombiniert. Die Ergebnisse des vierten Radiologen wurden wie die Ergebnisse des angepassten U-Net anhand der Vergleichswerte der anderen Radiologen ausgewertet. Tabelle 1 zeigt die Auswertung, welche den aktuell letzten Stand der Technik darstellt. Bei 100 zufällig ausgewählten Bildern ermittelten die Autoren zusätzlich die Zeit, welche für die Bemessung eines Bildes nötig war. Beim Deep Learning Ansatz bewegte sich diese zwischen $0,69$ und $0,70$ Sekunden. Der Radiologe benötigte zischen $23,49$ und $27,44$ Sekunden für eine Aufnahme. Diese Werte demonstrieren, welche Zeit- und Arbeitsersparnis durch eine Automatisierte Kardiomegalie Erkennung möglich wäre.[24]
Auch Chamveha et al.[5] untersuchten die Kardiomegalie Erkennung mithilfe von Segmentierungs NN. Sie nutzen dabei ein U-Net[37] mit VGG16[43] Encoder. Für das Training wurden 383 Röntgenbilder mit Masken für die Lungen verwendet, welche aus den Datensätzen JSRT[42] und Montgomery County X-ray[19] stammen. 331 Bilder mit Herzmasken wurden genutzt, welche teilweise aus dem JSRT Datensatz stammen und teilweise aus dem NIH Chest X-ray[50] und CheXpert[18] Datensatz stammen. Zu letzteren beiden Datensätzen wurden die Masken für die Veröffentlichung erstellt.

$$g_{i,j} = \lfloor (L-1) \sum_{n=0}^{f_{i,j}} p_n \rfloor \tag{19}$$

Wie in Li et al.[24] wurden die Bilder normalisiert. Dies wurde hierbei durch einen Histogrammausgleich, zu sehen in Gleichung (19), erreicht. Dabei stellt $f_{i,j}$ die Intensität des Ursprungsbildes am Pixel (i, j) dar und p_n den Anteil der Pixel mit der Intensität n am Ursprungsbild. Zusätzlich wurden die Daten Augmentationen (siehe Abschnitt 2.5)

Tabelle 2: Von Experten bewertete Maße MRD, MLD und ID

	Korrekt	Inkorrekt	Accuracy
Kardiomegalie	385	106	$78,4\%$
Keine Kardiomegalie	397	134	$74,8\%$
Gesamt	782	240	$76,5\%$

Die Auswertung zeigt, ob die von einem NN ausgegeben Werte zu MRD, MLD und ID in einem medizinischen Bericht genutzt werden könnten.[5]

Rotieren, Anwenden von Gaußschem Rauschen und Anwenden von Gaußscher Unschärfe genutzt. Ein Nachbearbeitungsschritt wurde verwendet, um die vom NN ausgegebenen Masken zu verbessern. Für die Evaluation ihrer Ergebnisse nutzten die Autoren 1022 Bilder aus dem NIH Chest X-ray[50] und CheXpert[18] Datensatz. Auf diesen Bildern ließen sie das U-Net die Werte für MRD, MLD und ID berechnen, welche daraufhin von Experten bewertet wurden. In Tabelle 2 sind die Ergebnisse abgebildet. Dabei wurden die gemessenen Werte auf einer Röntgenaufnahme nur als korrekt angesehen, wenn diese ohne jegliche Anpassung in einem medizinischen Bericht verwendet hätten werden können. Eine wichtige Auswertung um die Verwendbarkeit der gemessen Werte in einem realen Anwendungsszenario einschätzen zu können. Sie stützt weiterhin den Ansatz der Segmentierung von Herz und Lunge, da so die genannten Messungen zur Verfügung stehen. In einer weiteren Auswertung wurde zudem konkret die Kardiomegalie Erkennung ausgewertet. Dazu wurden die Informationen über die Präsenz oder Absenz von Kardiomegalie auf Röntgenthorax-Aufnahmen aus den Datensätzen NIH Chest X-ray[50] und CheXpert[18] genutzt. Bei der Auswertung stellte sich heraus, dass diese Informationen vor allem in Grenzfällen fehlerbehaftet sind. Eine weitere wichtige Arbeit stellt die Bachelorarbeit von Nico Hasler[14] aus dem Wintersemester 2020/21 über die Kardiomegalie Erkennung dar. Diese wurde ebenfalls am Lehrstuhl VI der Universität Würzburg durchgeführt. In dieser verglich Hasler[14] Methoden der klassischen Bildverarbeitung mit Deep Learning Ansätzen, genauer der Bildsegmentierung durch Neuronale Netze. Dabei konnte er zeigen, dass NN den klassischen Methoden überlegen sind. Mit einem U-Net[37] und einem ResNet101 als Encoder erreicht er einen $F1 - Score$ von $0,55$ und eine Sensitivity von $0,65$ bei der Kardiomegalie Erkennung auf dem CheXpert Validationsset[18]. Dieses U-Net soll in dieser Arbeit als Vergleichsmodell für den Segmentierungs Ansatz fungieren. Im Bereich der Kardiomegalie Erkennung durch Objektdetektion von Organen sind keine verwandte Arbeiten vorhanden, was sich durch diese Arbeit ändern soll.

Tabelle 3: Die besten drei Ergebnisse der RSNA Pneumonia Challenge auf dem privaten Testdatensatz.[9]

Team Name	mAP
Ian Pan and Alexandre Cadrin-Chênevert	0.25475
Dmytro Poplavskiy	0.24781
Phillip Cheng	0.23908

3.2 Objektdetektion auf Röntgenthorax-Aufnahmen

2018 wurde die Radiological Society of North America Pneumonia (Lungenentzündung) Detection Challenge auf Kaggle[1] abgehalten, bei der Lungentrübungen auf Röntgen-Thorax-Aufnahmen lokalisiert werden sollten. Die in Tabelle 3 abgebildeten Top 3 Ergebnisse wurden hauptsächlich durch Ensembles aus RetinaNets erzielt.[9, 28] Auch in der Fremdkörpererkennung auf Röntgenthorax-Aufnahmen wurden Objektdetektoren getestet. Le et al.[21] nutzen den Objektdetektor YOLOv4[4] um Fremdkörper zu lokalisieren, damit sie zur weiteren Verarbeitung separiert werden können. Auf dem Chexphoto[32] Datensatz erzielen sie damit eine mAP@0,5 von $0,61$ und einen F1-Score von $0,73$. Wichtig zu erwähnen ist, dass Chexphoto[32] aus echten und durch Bildbearbeitung künstlich erstellten Smartphonefotografien des CheXpert[18] Datensatzes besteht, was die Aufgabe erschwert. Santosh et al.[40] nutzen für die Aufgabe der Fremdkörperdetektion einen Faster-RCNN Detektor, der mit einem F1-Score von $0,91$ auf dem Indiana CXR Datensatz den letzten Stand der Technik übertrifft. Einem weiteren, schon sehr ähnlichen Problem zur Herz und Lungenflügel Detektion widmen sich Wessel et al.[51]. Sie erkennen Rippen mithilfe von Mask-RCNN mit einem Residual Netz der Tiefe 50 als Backbone und erhalten somit Boundingboxen und Masken. Die Boxen erreichen dabei einen durchschnittlichen Dice Wert von $0,846$ und die Masken einen Dice Wert von $0,733$. Fair vergleichen könnte man diese Werte jedoch nur, wenn auch der Dice Wert der aus den Masken berechneten Boundingboxen zur Verfügung

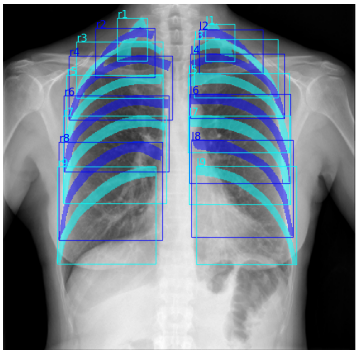


Abbildung 15: Eine Ausgabe des von Wessel et al.[51] vorgestellten Modells.

[1] https://www.kaggle.com

stünde. Trotzdem zeigen Wessel et al.[51], dass Informationen aus Boundingboxen und Masken bestehende Segmentierungs NN schlagen können.

3.3 Implementation

3.3.1 PyTorch

PyTorch[29] ist ein Deep Learning Framework wie beispielsweise Tensorflow[1]. Die

Abbildung 16: Logos der vorgestellten Deep Learning Frameworks PyTorch und Tensorflow.[34, 46]

Logos der beiden Frameworks sind in Abbildung 16 zu sehen. Bisher lag der Ansatz von Deep Learning Frameworks darin, vor jeglicher Berechnung einen sogenannten statischen Datenflussgraphen aufzubauen. Dieser enthält alle nötigen Berechnungen für ein Neuronales Netz und kann somit wiederholt auf dessen Eingaben angewandt werden. PyTorch nutzt dagegen eine dynamische Ausführung der Berechnungen und zeigt, dass dies fast ohne Leistungseinbuße möglich ist. Das Ziel dabei ist das Framework einfacher nutzbar zu machen. Dies wird durch einen imperativen Programmierstiel erreicht, welcher an Python angelehnt ist. Das macht die Implementierung flexibler und einfacher zu Debuggen. Um bei der Leistung keine Einbußen machen zu müssen, ist PyTorch hauptsächlich in C++ implementiert und ermöglicht das Nutzen einer Graphic Processing Unit.[29]

3.3.2 Detectron2

Detectron2[52] ist eine Programmbibliothek von Facebook AI Research, implementiert in PyTorch[29]. Sie stellt vortrainierte Neuronale Netze für die Bildverarbeitung zur Verfügung, zu denen unter anderem Faster R-CNN und Mask R-CNN gehören. Dabei verfolgt die Bibliothek einen modularen Aufbau, welcher Anpassungen einfach möglich macht. Detectron2 ist genau wie sein Vorgänger Detectron Open Source und kann von GitHub.com[2] heruntergeladen werden. Offiziell wird die Bibliothek jedoch nur für Linux und Mac OS unterstützt. Die Verwendung unter Windows (bestätigt für die Nutzung ohne GPU)[3] ist jedoch genauso möglich. Dazu muss Detectron2 wie auf Readthedocs.io[4] beschrieben aus dem Quellcode gebaut werden, wozu Microsofts

[2]https://github.com/facebookresearch/detectron2
[3]Ist CUDA auf dem Rechner installiert muss, für eine Installation ohne GPU, in setup.py die Abfrage ob CUDA verfügbar ist auf False gesetzt werden.
[4]https://detectron2.readthedocs.io/tutorials/install.html

Tabelle 4: Die Ergebnisse verschiedener Neuronaler Netze aus Detectron2 auf dem COCO[27] Datensatz.

Name	Traingzeit	Inferenzzeit	Speicher	box AP	mask AP
Faster R-CNN (R50)	0,209	0,038	3,0	40,2%	–
Faster R-CNN (R101)	0,286	0,051	4,1	42,0%	–
Faster R-CNN (X101)	0,638	0,098	6,7	43,0%	–
Mask R-CNN (R50)	0,261	0,043	3,4	41,0%	37,2%
Mask R-CNN (R101)	0,340	0,056	4,6	42,9%	38,6%
Mask R-CNN (X101)	0,690	0,103	7,2	44,3%	39,5%

Abgebildet sind Ergebnisse der in dieser Arbeit verwendeten Neuronalen Netze aus Detectron2 auf dem COCO[27] Datensatz. Dabei sind die verwendeten Backbones in Klammern angegeben, wobei R für ResNet[15] und X für ResNeXt[54] steht. Die jeweiligen Trainings der NN umfassten dabei ungefähr 37 Epochen und wurden auf acht NVIDIA V100 GPUs durchgeführt. Gezeigt ist die Verarbeitungszeit eines Batches in Sekunden während des Trainings. Daneben die Inferenzzeit, welche in Sekunden misst, wie lange das Netz benötigt, um eine Ausgabe für ein Eingabebild zu berechnen. Anschließend ist der Speicherbedarf des Netzes während des Trainings in Gigabyte angegeben und schlussendlich das Evaluationsergebnis der Average Precision von Boundingboxen und Segmentierungsmasken.[52] Um einen Vergleichswert für die abgebildet Werte zu haben, soll die Boundingbox AP von Dual-Swin-L[25] vorgestellt werden. Diese stellte am 29.07.2021 nach https://paperswithcode.com/sota/object-detection-on-coco die höchste AP von 60,1% auf dem COCO Datensatz, in der Kategorie ohne extra Trainingsdaten, dar.

Visual C++ Build Tools benötigt werden. Detectron2 musste von Grund auf neu geschrieben werden, wodurch Verbesserungen auf niedrigster Ebene möglich waren. Aus diesem Grund sind Implementationen aktueller Neuronaler Netzwerkarchitekturen von hoher Qualität entstanden. Was diese leisten können ist Tabelle 4 zu entnehmen.[53]

4 Vorgehen

4.1 Vorbereitung

Der erste Schritt dieser Arbeit bestand darin alle nötigen Programme für das weitere Vorgehen einzurichten. Python 3.9 bildet dabei die Grundlage des Projekts. Quelltext, welcher für das Trainieren der Modelle benutzt wurde, ist jedoch auch Python 3.8 lauffähig.

```
1 torch==1.8.1+cu102
2 torchaudio==0.8.1
3 torchvision==0.9.1+cu102
4
5 opencv-python==4.5.2.54
6
7 detectron2==0.4+cu102
```

Quelltext 1: Pip Pakete, welche zur Benutzung des Projekts benötigt werden. torch, torchaudio und torchvision müssen dabei mit dem Befehl von PyTorch.org und detectron2 mit dem Befehl von https://detectron2.readthedocs.io/en/latest/tutorials/install.html installiert werden.

Die nötigen Python Pakete wurden mit Pip installiert und sind in Quelltext 1 aufgelistet. Damit für das Training der Neuronealen Netze die GPU verwendet werden kann, muss auch CUDA von NVIDIA.com[5] installiert werden. Auf Linux und Mac OS kann Detectron2[52] direkt installiert werden, auf Windows muss Abschnitt 3.3.2 beachtet werden. Zum erstellen von Boundingboxen wurde LabelImg[6] und zum Erstellen von Masken GIMP[7] genutzt.

4.2 Datengenerierung

4.2.1 Verwendete Datensätze

Um die verwendeten Neuronalen Netze zu trainieren, wurden in dieser Arbeit die im Folgenden beschriebenen Datensätze verwendet. Der JSRT-Datensatz (Japanese Society of Radiological Technology)[42] enthält 247 Röntgenthorax-Aufnahmen und wurde ursprünglich für die Lungenrundherd Erkennung zusammengestellt. Für die Veröffentlichung von Ginneken, Stegmann und Loog[47] wurden zusätzlich Herzen und Lungen manuell segmentiert. Des Weiteren wurde der Shenzhen Chest X-ray Datensatz (Shenzhen No.3 People's Hospital)[19], welcher aus 662 Röntgenthorax-Aufnahmen besteht, genutzt. Zu diesem veröffentlichten Stirenko et al.[45] 566 segmetierte Lungen auf Kaggle[8]. Auch der Montogomery County Chest X-ray Datensatz (Department of Health

[5] https://developer.nvidia.com/cuda-downloads
[6] https://github.com/tzutalin/labelImg
[7] https://www.gimp.org
[8] https://www.kaggle.com/yoctoman/shcxr-lung-mask

and Human Services, Montgomery County)[19], der 138 Röntgenthorax-Aufnahmen mit dazugehörigen Lungenmasken enthält, wurde in den Trainingsdatensatz übernommen. Sowohl der Shenzhen Chest X-ray[19], als auch der Montogomery County Chest X-ray[19] Datensatz wurden ursprünglich für die computergestützte Erkennung von Lungenerkrankungen wie der Tuberkulose erstellt. Mit diesen drei Datensätzen bestand der Trainingsdatensatz dieser Arbeit aus 1047 Röntgenthorax-Aufnahmen, auf denen 951 Lungen und 247 Herzen manuell segmentiert vorlagen. Während der Durchführung seiner Bachelorarbeit segmentierte Hasler[14], welcher in seiner Arbeit die gleichen Datensätze verwendete, die fehlenden Herzmasken manuell. Diese wurden in dieser Arbeit weiter verwendet.

4.2.2 Datenaufbereitung

In seiner Bachelorarbeit verwendete Hasler[14] quadratische Bilder als Eingabe. Dazu skalierte er die Bilder auf ein quadratisches Format, was für die Berechnung des HTQ kein Problem darstellt. Aus diesem Grund lagen seine Herzmasken jedoch in einem quadratischen Seitenverhältnis vor. Da für diese Arbeit die in Abschnitt 2.9.1 angesprochen Anchorboxen gewählt werden mussten, war es wichtig, das Seitenverhältnis der Bilder beizubehalten. Somit liegen die Lungen und Herzen in ihrer natürlichen Form vor, was es erleichtert Boxen zu definieren, welche das ganze Spektrum an Organformen abbilden. Die von Hasler[14] erstellten Masken mussten also zunächst zurückskaliert werden. Der nächste Schritt bestand darin die fehlenden 96 fehlenden Lungenmasken zu segmentieren, wozu die Software GIMP verwendet wurde. Dabei wurden die Lungenflügel genau segmentiert, um die Datenqualität des Datensatzes nicht zu verschlechtern. Anschließend wurden sowohl auf den Herzmasken von Hasler[14], als auch auf den Lungenmasken von Stirenko et al.[45] Fehler korrigiert. Hierbei sei angemerkt, dass die Maskenqualität des JSRT[42] und Montogomery County Chest X-ray Datensatzes[19] generell besser ist. Neben weniger Fehlern in den Masken sind die Masken der beiden Datensätze auch genauer. Die verschiedenen Fehlertypen, welche auf den Herzmasken

Abbildung 17: Zu sehen sind Fehlertypen, welche in den Masken gefunden wurden. **Links:** Eine dritte separate Maske (unten links). **Mitte links:** Schlitze in der Maske. **Mitte rechts:** Ein Loch in der Maske. **Rechts:** Hörner oben am Herz.[45, 14]

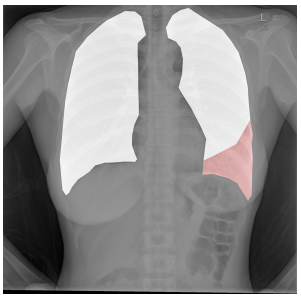


Abbildung 18: Stirenko et al.[45] übersahen auf ein paar wenigen Lungenmasken ganze Teile der Lunge. Die Abbildung[19] zeigt ein Beispiel, auf dem der übersehene Teil der Lunge rot eingezeichnet ist.

von Hasler[14] und den Lungenmasken von Stirenko et al.[45] verbessert wurden, sind in Abbildung 17 und Abbildung 18 gezeigt. Haslers[14] Daten wiesen dabei ausschließlich die in Abbildung 17 rechts zu sehenden Hörner bei einigen Herzmasken auf. Während diese auf den Originalbildern kaum zu erkennen sind, hob sie die Skalierung der Maske und das anschließende Setzen aller Pixel größer 0 auf 255 deutlich hervor. Die Lungenmasken zeigten verschiedene Fehler, welche so gut wie möglich korrigiert wurden. Drei dieser Fehler werden beispielhaft in Abbildung 17 gezeigt. Auf dem Bild links ist unten links eine kleine separate Maske zu sehen. In der Abbildung daneben fallen oben rechts und unten links Schlitze in der Maske auf und in der darauffolgenden Abbildung ist oben links ein Loch zu erkennen. Ein deutlich gravierender Fehler ist in Abbildung 18 dargestellt. In diesem Fall wurde ein Teil der Lunge übersehen, was von einem Radiologen bestätigt wurde. Der Fehler konnte jedoch nur auf vier Röntgenaufnahmen beobachtet werden und wurde nur bei den Boundingboxen berücksichtigt.

4.2.3 Boundingboxen

Objektdetektoren und Instanz Segmentierungsnetze, welche in dieser Arbeit verwendet wurden, benötigen Boundingboxen (siehe Abschnitt 2.9 und Abschnitt 2.10). Ein wesentlicher Bestandteil dieser Arbeit war es also, Boundingboxen für die zuvor genannten Datensätze zu erstellen. Die Lungenmasken aus dem JSRT[42], dem Shenzhen Chest X-ray[19] und dem Montogomery County Chest X-ray[19] Datensatz wurden von Experten erstellt. Gleiches gilt auch für die Herzmasken des JSRT Datensatzes. Da dieses Expertenwissen genutzt werden muss, wurden diese Masken durch einen Algorithmus in Boundingboxen konvertiert. Zum Speichern der Boundingboxen wurde dabei das Pascal VOC Format benutzt. Die verbleibenden 800 Herzmasken wurden von Hasler[14], einem Laien, erstellt und enthalten somit kein Expertenwissen. Aus diesem Grund konnten die Boundingboxen für diese Masken, für einen unabhängigen und fairen Vergleich zwischen Segmentierungnetzen und Objektdetektionsnetzen, manuell erstellt werden. Dabei half die Beratung eines Radiologen und eines Medizinstudenten. Erstellt wurden die Boxen mit der Software LabelImg, welche diese unter dem Pascal VOC Format abspeichern kann. Da es auf manchen Bildern schwer war, das Herz un-

ter dem Oberbauch zu erkennen, wurden diese Bilder von einem Radiologien überprüft und kommentiert. Drei dieser Kommentare sind in Abbildung 19 zu den entsprechen

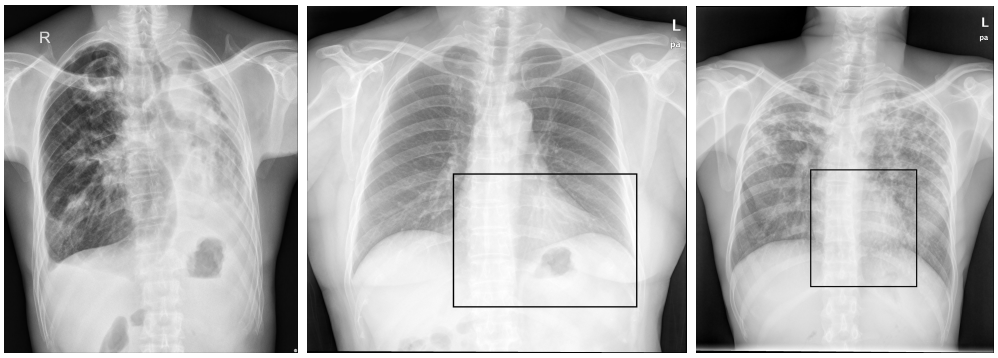


Abbildung 19: Die Bilder[19] zeigen Röntgenthorax-Aufnahmen mit der dazu erstellten vorläufigen Boundingbox um das Herz. Ein Radiologe kommentierte diese wie folgt: **Links:** Vielleicht ganz Ausschluss, da eher ungeeignet für Messungen. **Mitte:** Deutlich zu viel Lunge auf der linken Patientenseite mit erfasst, zu viel Oberbauchanteile mit erfasst. **Rechts:** Zu viel Lunge auf der linken Patientenseite mit erfasst.

Bildern gezeigt. Zusätzlich sind die ursprünglichen verbesserungswürdigen Boundingboxen eingezeichnet. Die vom Radiologen genannten Makel wurden korrigiert und fünf Bilder aussortiert.

4.2.4 Datensatz

In einem letzten Schritt wurden die gesammelten und erstellten Daten in einem eigenen Datensatz zusammengefasst. Davor wurde jedoch noch eine letzte Änderung an den Daten vorgenommen. Die Herzmasken von Hasler[14] wurden mit den für diese Arbeit erstellten Boundingboxen beschnitten, sodass keine Herzmaske größer als die dazugehörige Bounindbox ist. Der danach entstandene Datensatz enthält 1042 Röntgenthorax-Aufnahmen mit dazugehörigen Masken und Boundingboxen für Herzen und Lungenflügel. Dabei wurden die Lungenflügel jeweils einzeln und zusammen, als Lungenmaske, abgespeichert. Für die Lungenmaske wurden die beiden Masken der Lungenflügel in einer Maske kombiniert und die dazugehörige Boundingbox definiert sich durch die linke obere Ecke des linken und die rechte untere Eckte des rechten Lungenflügel. Ein Beispiel aus dem Datensatz dieser Arbeit ist in Abbildung 20 gezeigt. In der Grafik links kann man erkennen, dass Herzmaske und Herz Boundingbox voneinander abweichen. Das liegt daran, dass sie unabhängig voneinander erstellt wurden. Wie in Abschnitt 2.4 beschreiben, werden für das Training eines Neuronalen Netzes zwei Datensätze benötigt. Dafür wurde der 1042 Röntgenbilder umfassende Datensatz dieser Arbeit in einem Trainingsdatensatz der Größe 942 und einen Validationsdatensatz der Größe 100 aufgeteilt. Dabei wurden für den Validierungsdatensatz die ersten

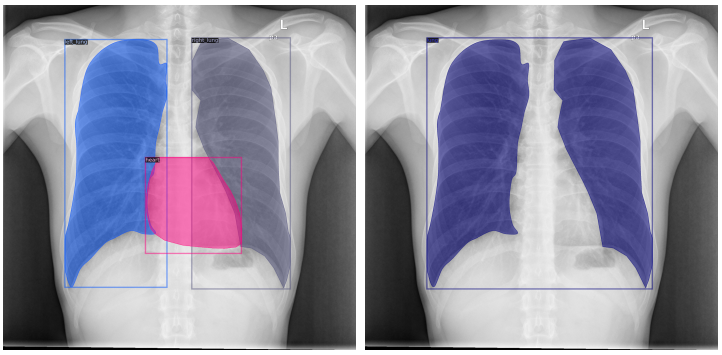


Abbildung 20: Eine Röntgenthorax-Aufnahme aus dem Shenzhen Chest X-ray Datensatz[19] mit dazugehörigen Segmentierungsmasken und Boundingboxen. Die Lungenmaske und Boundingbox ist zur besseren Übersicht separat abgebildet.

62 Bilder des Shenzhen Chest X-ray Datensatzes[19], die ersten 25 Bilder des JSRT Datensatzes[42] und die ersten 13 Bilder aus dem Montogomery County Chest X-ray Datensatz[19] verwendet. Die jeweilige Anzahl der Bilder repräsentiert dabei den Anteil des jeweiligen Datensatzes am gesamten Datensatz. In dieser Arbeit wurden NN aus FAIR's Detectron2[52] verwendet. Die Programmbibliothek nimmt dabei einen Datensatz als Liste von Dictionaries entgegen, wobei ein Dictionary die Informationen zu einem Bild enthält.

```
[{
  "file_name": "roentgenaufnahme.png",
  "height": 2725,
  "width": 2450,
  "image_id": 0,
  "annotations": [
    {
      "bbox": [911, 1251, 1905, 2346],
      "bbox_mode": 0,
      "category_id": 3,
      "segmentation": {
        "size": [2725, 2450],
        "counts": "bji[25ld2:E;F:E;F:E;F:E;F:E;F:F:E;F:E>Cb0..."
      }
    }, ...
  ]
}]
```

Quelltext 2: Der Quelltext zeigt das Datensatzformat, welches Detectron2[52] entgegennimmt, für ein Bild. Dieses muss dabei separat gespeichert werden und unter "file_name" gefunden werden können. Nach der ersten Annotation bei ... können weitere Objekte in dem Bild mit dazugehörigen Masken und Boxen folgen.[36]

Ein Dictionary Eintrag ist in Quelltext 2 dargestellt. Dabei lässt sich erkennen, dass sich dieses Format optimal als JavaScript Object Notation Datei speichern lässt. Dieses Dateiformat kann beispielsweise durch die Python Bibliothek json wie benötigt eingelesen werden. In jedem Eintrag zu einem Bild ist dessen Dateipfad und seine Annotationen enthalten. Diese stellen in dieser Arbeit die Boundingboxen und Segmentierungsmasken zu Herz, Lunge, linkem und rechtem Lungenflügel dar. Die Masken werden dabei in "counts", codiert durch COCO's[27] Run Length Econding, angegeben.[36] Um diesen in der JSON Datei abspeichern zu können, wird das RLE in das 8-Bit Universal Character Set Transformation Format, bekannt als UTF-8, kodiert. Bei der Erstellung des Datensatzes sind Skripte entstanden, welche generell für die Generierung eines Datensatzes im Detectron2[52] Format genutzt werden können. Die Methode create_detectron2_dataset in Processing data.detectron2_format erstellt einen solchen Datensatz. Dabei muss eine Methode geschrieben werden, welche einen einzelnen Datensatzeintrag erstellt. Dazu kann sich an der bestehenden Methode in data.detectron2_format orientiert werden.

4.2.5 Privater Datensatz der Uniklinik Würzburg

Für die Auswertung der Ergebnisse wurde für diese Arbeit ein privater Datensatz der Uniklinik Würzburg[9] erstellt. In diesem sind 108 Röntgenthorax-Aufnahmen enthalten, wobei 58 Aufnahmen Kardiomegalie zeigen. Zu diesen Aufnahmen erstellte ein Radiologe der Uniklinik Würzburg Segmentierungsmasken und maß separat den HTQ. Genauso wie der Trainings- und Validierungsdatensatz wurde dieser Datensatz, der Testdatensatz, mit den Vorverarbeitungsskripten in das benötigte Detectron2 Format gebracht. Die Boundingboxen wurden dabei automatisch aus den Masken generiert. Die manuell gemessen HTQs wurden separat in einem eigenen Format abgespeichert.

4.3 Implementierung der Neuronalen Netze

In dieser Arbeit sollen die Neuronalen Netze Faster R-CNN[35] und Mask R-CNN[16] mit verschiedenen Backbones trainiert und verglichen werden. Eine Übersicht über die verwendeten NN ist in Tabelle 4 zu sehen. Um diese zu trainieren, wurden deren Implementationen von Detectron2[52] genutzt. Damit dabei das gewünschte Trainingsverhalten erzielt werden konnte, wurde dessen DefaultTrainer angepasst. Dieser stellt die Standard Trainingsschleife von Detectron2 dar. Die Trainingsschleife ist dabei der Quelltext, der eine Iteration beim Training verarbeitet und deshalb immer wieder ausgeführt wird. Für zukünftige Arbeiten sollte festgehalten werden, dass das Schreiben einer eigenen Trainingsschleife die bessere Wahl dargestellt hätte. Diese wird wie eine

[9] Universitätsklinikum Würzburg, Josef-Schneider-Straße 2, 97080 Würzburg

PyTorch[29] Trainingsschleife geschrieben, wobei eine Vorlage von FAIR[10] zur Verfügung gestellt wird. Dies ermöglicht eine deutlich höhere Flexibilität und des Weiteren ist PyTorch wesentlich ausführlicher dokumentiert als Detectron2. In dieser Arbeit wurde zunächst das Trainingsskript von DeepLab[52][11] übernommen und weiter angepasst. Das Laden der Datensätze konnte einfach umgesetzt werden, da diese wegen der Vorverarbeitungsschritte schon im richtigen Format vorlagen. Allerdings musste die Komponente Mapper[36], welche die Daten aus dem Datensatz zur Laufzeit lädt, angepasst werden. Das war nötig, um die selbst geschriebenen Daten Augmentationen möglich zu machen (siehe Abschnitt 4.4.2). Um neues Verhalten zum DefaultTrainer hinzuzufügen, welcher eine Python Klasse darstellt, können seine Methoden überschrieben werden. Des weiteren führt Detectron2[52] das sogenannte Hook System ein. Dieses erlaubt es vor und nach jedem Step zusätzlichen eigenen Code auszuführen. Dabei stellt ein Step die Forward- und Backpropagtion (siehe Abschnitt 2.2.2.2) eines Batches dar.[52, 36] Da der DefaultTrainer die Auswertung des Loss auf einem Validierungsdatensatz nicht vorsieht, wurde diese Auswertung mithilfe des Hook Systems für diese Arbeit implementiert. Dabei erfolgt eine Auswertung immer dann, wenn die Gewichte θ des NN abgespeichert werden. Das Speicherintervall kann in Detectron2 selbst festgelegt werden. Der Hook wurde dabei auf Grundlage eines Github Issue[12] zu Detectron2, einer Kaggle Challenge[13] und dem Quelltext für die Auswertung von Detectron2[52] implementiert. Für die Auswertung der NN auf dem Testdatensatz, was von Detectron2 unterstützt wird, musste das DatasetEvaluator Interface[36] implementiert werden. Faster R-CNN[35] (Abschnitt 2.9) und Mask R-CNN[16] (Abschnitt 2.10) geben zu einer Objektklasse mehrere Boundingboxen und Masken aus. Das liegt daran dass, wie in Abbildung 13 zu sehen, mehrere Objekte einer Klasse auf einem Bild vorkommen können. Im Szenario der Kardiomegalie Erkennung liegen auf einem Eingabebild jedoch nur ein Herz und eine Lunge vor. Da die genannten NN dennoch weiterhin versuchen, mehrere Herzen und Lungen zu detektieren, werden die Organe mit der höchsten Konfidenz ausgewählt. Die Konfidenz meint hierbei die Wahrscheinlichkeit, welche durch den in Abschnitt 2.1.1 beschriebenen Softmax entstehen. Dies setzt der JsonDumpEvaluator um, welcher in dieser Arbeit das DatasetEvaluator Interface implementiert. Er speichert zu jedem Bild und jedem Organ die Boundingbox und Maske mit der höchsten Konfidenz in einer JSON Datei ab. Dabei evaluiert er gleichzeitig die gemittelte IoU[14] und den gemittelten Dice Score (siehe Abschnitt 2.11.2) der vier Objektklassen. Alle für das Training geschriebenen Erweiterungen und Anpassungen des DefaultTrainers sind generisch gehalten und können genauso für andere Projekte genutzt werden.

[10] https://github.com/facebookresearch/detectron2/blob/master/tools/plain_train_net.py (30.07.2021)
[11] https://github.com/facebookresearch/detectron2/tree/master/projects/DeepLab (30.07.2021)
[12] https://github.com/facebookresearch/detectron2/issues/810 (30.07.2021)
[13] https://www.kaggle.com/corochann/vinbigdata-detectron2-train (30.07.2021)
[14] Implementiert nach https://youtu.be/XXYG5ZWtjj0 (30.07.2021)

Ausschließlich der JsonDumpEvaluator müsste für andere Einsatzgebiete um die Non Maximum Suppression erweitert werden. Sollen andere Datensätze verwendet werden, müssen diese in Detection train.py eingetragen werden. Trainings- und Testdatensatz müssen anschließend noch in der Konfiguration (siehe Abschnitt 4.4) angegeben werden. Der Name des Validierungsdatensatzes muss der LossEvalHook Klasse in Detection train.py build_hooks übergeben werden. Zum Starten des Trainings wird der Befehl aus Quelltext 3 genutzt.

```
1 python train.py --config-file faster_rcnn_r_101_fpn_3x --num-gpus 1
    OUTPUT_DIR output
```

Quelltext 3: Hier ist der Befehl zum starten des Trainingsskript dieser Arbeit gezeigt.

Dabei können wie bei OUTPUT_DIR output, getrennt durch Leerzeichen, weitere Parameter in dem Aufruf festgelegt werden.

4.4 Hyperparameteranpassung

Um ein Neuronales Netz trainieren zu können, müssen wie in Abschnitt 2.1.2 erwähnt einige Parameter zuvor festgelegt werden. In Detectron2[52] werden diese in einer CfgNode[36] eingetragen, welche im Folgenden als Konfiguration bezeichnet wird. Bevor auf das Finden geeigneter Hyperparameter eingegangen wird, soll die Verwendung des für diese Arbeit geschriebenen Konfiguration Quelltextes erklärt werden. Da vortrainierte Neuronale Netze verwendet werden und deren Parametereinstellungen vorwiegend beibehalten wurden, wurde das Festlegen der Konfiguration in zwei Schritte unterteilt. Um den Aufrufbefehl (siehe Quelltext 3) kurz zu halten, wurde eine Methode implementiert, welche aufgrund einer definierten Netzbeschreibung (zum Beispiel faster_rcnn_r_101_fpn_3x für das NN Faster R-CNN[35] mit Backbone ResNet101[15]) die entsprechende Konfiguration aus Detectron2 lädt. Diese enthält bisher Abkürzungen für die Netze dieser Arbeit, kann jedoch beliebig erweitert werden. Durch die zweite Methode Detection card_detection.config.add_base können Hyperparametern neue Werte zugewiesen werden.

4.4.1 Hyperparameter des Netzes

Der Pixel Durchschnitt $[123.675, 116.280, 103.530]$ und die Pixel Standardabweichung $[58.395, 57.120, 57.375]$ bezeichnen den Mittelwert und die Standardabweichung mit denen die Pixel der Eingabebilder von Detectron2[52] normalisiert werden. Da jeder Pixel aus einem Rot, Grün und Blau Kanal besteht, müssen dabei drei Werte angeben werden. Die aufgelisteten Werte ergeben sich aus dem Durchschnitt der Pixelwerte des ImageNet[6] Datensatzes, welcher genutzt wurde um die verwendeten Backbones vor zu trainieren.[36] Pham, Pham und Dang[31] nutzten Detectron2 für das Erkennen von Straßenschäden. Sie experimentierten dabei mit eigenen Pixel Mittelwerten und dazu-

gehörigen Standardabweichungen. Diese konnten ihr Ergebnis jedoch nicht verbessern. Eine mögliche Erklärung der Autoren ist, dass das NN mit anderen Werten vortrainiert wurde und dadurch auf eine andere Normalisierung eingestellt ist. Aus diesem Grund wurden auch in dieser Arbeit die Werte von ImageNet[6] übernommen. Aus der Erklärung von Pham, Pham und Dang[31] wird des weiteren gefolgert, die NN dieser Arbeit mit farbigen Eingabebildern zu trainieren. Die Röntgenbilder sind zwar schwarzweiß und bestehen somit aus nur einem Farbkanal, können jedoch durch das Verwenden von drei identischen Kanälen als farbig aufgefasst werden. Ein weiterer denkbarer Vorteil dieser Vorgehensweise sind mehr Eingabewerte. Dabei wird in Abschnitt 4.4.2 darauf eingegangen, wie mit Hilfe von addiertem Rauschen vorgegangen wird damit diese nicht identisch sind. In einem auf Faster R-CNN[35] basierenden NN müssen die Anchorboxen aus Abschnitt 2.9.1 gewählt werden. Diese definieren sich durch Größe und Seitenverhältnis, wobei Detectron2[52] die Größe als Wurzel der Fläche und das Seitenverhältnis als Höhe durch Breite entgegennimmt. Pham, Pham und Dang[31] konnten in ihrer Arbeit feststellen, dass das möglichst genaue anpassen dieser Anker auf die zu erwartenden Boundingboxen das Training deutlich verkürzt, das heißt es werden weniger Iterationen benötigt. Aus diesem Grund wurden die Boundingboxen des Trainingsdatensatzes dieser Arbeit genauer untersucht. Dazu wurden zu jedem Bild und Organ die Wurzel der Fläche als Anteil an dem ganzen Bild und das Seitenverhältnis ausgeben. Wie diese Ausgabe aussah ist in Abbildung 21 beispielhaft für das Herz

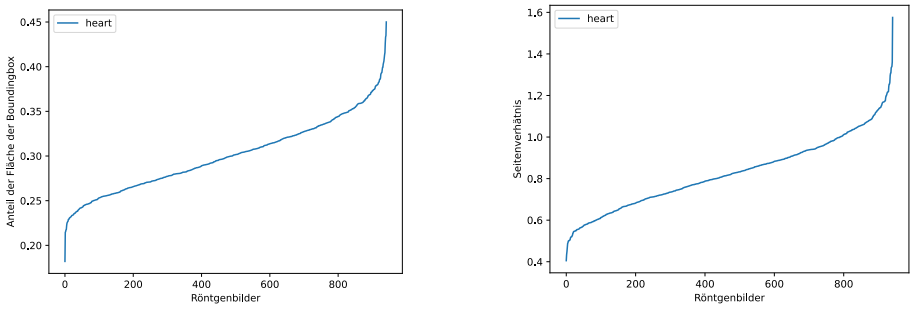


Abbildung 21: Um den Trainingsdatensatz besser zu verstehen wurden die Wurzeln der Flächen der Boundingboxen zu Herz, Lunge und Lungenflügeln und deren Seitenverhältnisse in einem Graphen dargestellt. In dieser Grafik ist die Ausgabe zu der Herz Boundingbox zu sehen, wobei die jeweiligen Werte sortiert entlang der X-Achse gezeigt werden.

zu sehen. Um die Größen für die Anchorboxen festzulegen wurden wie in [31] nicht notwendige Werte aus den Standardwerten[36] $[32, 64, 128, 256, 512]$ weggelassen. Dazu wurden repräsentative Minima und Maxima aus den ausgebenden Graphen (siehe Abbildung 21) zu den Flächen gewählt und mit den möglichen Wurzeln der Flächen der Eingabebilder multipliziert. Da keiner der entstandenen Werte kleiner als 64 war, ergaben sich die Werte $[64, 128, 256, 512]$ für die Flächen der Anchorboxen. Zum Ermitteln der Seitenverhältnisse wurden zu jedem Organ ein (Lunge) bis drei (Lungenflügel)

repräsentative Werte aus den Graphen der Seitenverhältnisse gewählt. Dabei wurde sich am Median der gesamten Daten und dem Median der jeweils linken und rechten Hälfte des Graphen orientiert. Die so entstandenen neun Seitenverhältnisse wurden, aufgrund der teilweisen starken Ähnlichkeit, auf die Werte $[0.7, 1.05, 1.85, 2.15, 2.5]$ reduziert. Des Weiteren nutzt Faster R-CNN RoIs (siehe Abschnitt 2.9), beziehungsweise Objektvorschläge. Wie viele davon für ein Eingabebild generiert werden sollen, muss in Detectron2[52] angeben werden. Der Standard[36] liegt hierbei bei 512, wobei für diese Arbeit nur 128 benutzt wurden. Dies spart Rechenleistung und auf den Röntgenbildern müssen schlussendlich nur vier Boxen detektiert werden. Der Wert von 128 wird unter anderem auch im Detectron2 Tutorial[15] und im Detectron2 Projekt TridentNet[16] verwendet.[52]

4.4.1.1 Hyperparameter des Optimizters

Detectron2[52] nutzt den Optimizer SGD, welcher in Abschnitt 2.2.2.2 besprochen wurde. Anstatt eine Lernrate über das ganze Training hinweg zu nutzten wird jedoch ein Learningrate Scheduler, auf deutsch Lernraten Planer, genutzt. Dieser passt die Lernrate nach einem festgelegten Plan während des Trainings an. In Abbildung 22 ist die

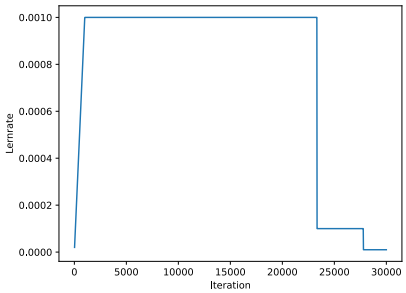


Abbildung 22: Der Graph zeigt den Verlauf der Lernrate während des Trainings. Diese verläuft nich konstant, sonder glieder sich in drei Phasen. Zu beginn wird die Lernrate graduell aufgewärmt und zum Ende Schrittweise verkleinert.[13, 55]

Lernrate eines Trainingsdurchlaufs dieser Abrbeit dargestellt. Dabei wächst die Lernrate zunächst linear bis eine feste Lernrate, die Basislernrate, erreicht wird. Dieses Vorgehen wird graduelles Aufwärmen genannt und wurde von Goyal et al.[13] vorgeschlagen. Durch das Nutzen einer weniger aggressiven Lernrate zu Beginn des Trainings sollen Trainings Instabilitäten vermieden werden.[13, 12] Ein weiteres auffälliges Merkmal in Abbildung 22 ist das Abfallen der Lernrate zum Ende des Trainings, was Lernraten Decay genannt wird. Dabei wird zu festgelegen Iterationen die Lernrate mit einem Faktor kleiner eins, in Detectron2 dem Gamma, multipliziert. Die allgemeine Erklärung, wieso

[15] https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5 (30.07.2021)
[16] https://github.com/facebookresearch/detectron2/tree/master/projects/TridentNet (30.07.2021)

Lernraten Decay genutzt wird, ist um den Optimizer zum Ende des Trainings beim konvergieren zu unterstützen. Konvergieren meint dabei ein nahe gelegenes Minimum zu findet.[55] You et al.[55] halten diese Erklärung jedoch für unzureichend. Sie erklären, dass bei einem Training mit einer höheren Lernrate besser mit allgemeinen Daten umgegangen werden kann und zum Ende des Trainings mit einer kleineren Lernrate komplexere Muster gelernt werden. Die Parameter für das graduelle Aufwärmen wurden bei den Standardeinstellungen belassen. Um geeignete Werte für den Lernraten Decay zu finden wurden die Detectron2[52] Projekte von Facebook[17] und die Arbeiten von Vishwakarma et al.[48] und Pham, Pham und Dang[31] studiert. Die Studie ergab, dass die optimalen Iterationen für einen Lernraten Decay bei ungefähr $77,\overline{7}\%$ und $92,6\%$ der gesamten Iterationen liegen und ein Faktor von $0,1$ oder $0,05$ gewählt werden sollte. Einen weiteren Hyperparameter stellt die Batchgröße dar, welche für den SGD (siehe Abschnitt 2.2.2.2) gewählt werden muss. Diese wurde so groß gewählt wie der Speicher der verwendeten GPU NVIDIA GeForce RTX 2080 Ti zuließ. Dabei ist zu bedenken, dass das NN für die Berechnungen auch auf der GPU gespeichert werden muss. Der hierfür benötigte Speicherplatz je Netz kann Tabelle 4 entnommen werden. Wegen des hohen Speicherplatzverbrauchs des ResNeXt101[54] konnten bei NN mit diesem Backbone nur zwei Bilder in einem Minibatch verwendet werden. Alle weiteren Netze ließen vier Bilder in einem Batch zu. Um eine optimale Lernrate zu finden, wurden ausgehend vom Standard $0,02$ verschiedene Lernraten getestet. Dazu wurde Mask

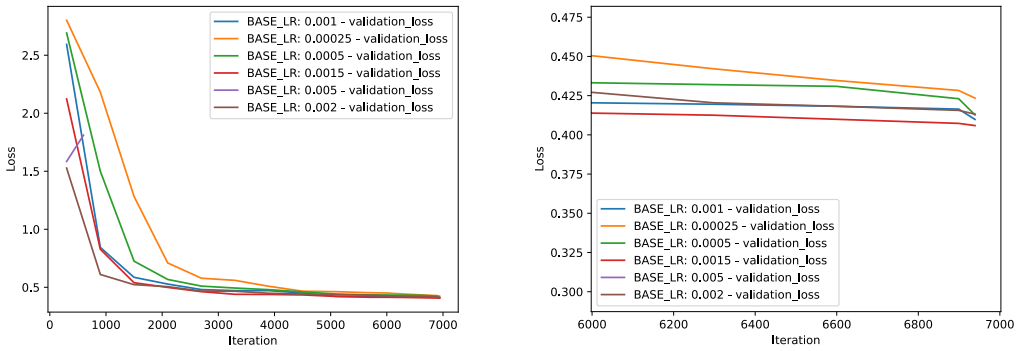


Abbildung 23: Die beiden Graphen zeigen den gemittelten Loss aller Batches des Validierungsdatensatzes zu verscheidenden Iterationen der ersten Trainingsdurchläufe dieser Arbeit, bei denen verschiedene Lernraten getestet wurden. Der Graph rechts stellt einen Ausschnitt des linken Graphen dar und zeigt die beste Lernrate von $0,0015$.

R-CNN[16] 6940 Iterationen lang mit dem Backbone ResNet101[15] und den Lernraten aus Abbildung 23 trainiert. Anschließend wurde der über alle Batches gemittelte Loss des Validierungsdatensatzes begutachtet. Das beste Ergebnis konnte dabei die Lernrate $0,0015$ erzielen. Größere Lernraten, wie beispielsweise $0,005$ brachen nach wenigen

[17] https://github.com/facebookresearch/detectron2/tree/master/projects (30.07.2021)

Iteration ab. Durch zwei weitere Trainings, zu sehen in Abbildung 24 (links), wurde

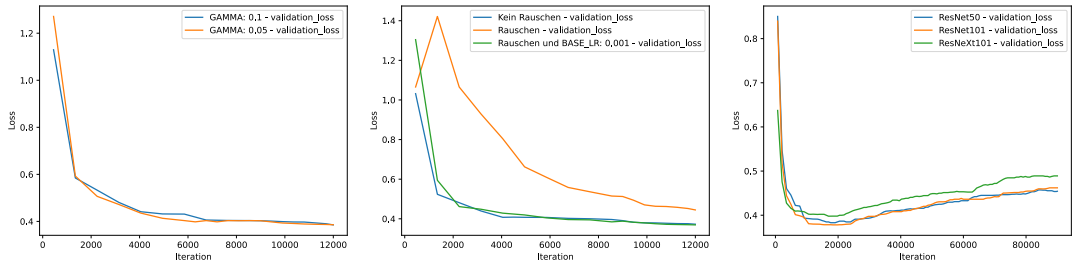


Abbildung 24: Die Graphen zeigen den gemittelten Loss aller Batches des Validierungsdatensatzes im Verlauf verschiedener Trainings. **Links:** Zwei Trainingsdurchläufe mit unterschiedlichen Gammas **Mitte:** Der Vergleich von Trainings mit und ohne Gaußschem Rauschen als Daten Augmentation. Ist die Lernrate nicht angegeben, liegt sie bei $0,0015$. **Rechts:** Drei sehr lange Trainingsdurchläufe mit unterschiedlichen Backbones. Ab Iteration 20000 ist das sogenannte Overfitting (siehe Abschnitt 2.3) zu erkennen.

sich für das Gamma $0,1$ entscheiden.

4.4.2 Daten Augmentierung

Die Basis aller im Weiteren aufgeführten Augmentierungen bildet das Ändern der Größe der Eingabebilder. Für diese Arbeit wird die kleinere Seite jedes Eingabebilds auf entweder $640, 672, 704, 736, 768$ oder 800 Pixel skaliert, wobei das Seitenverhältnis beibehalten wird. Zusätzlich ist die größere Seite des Eingabebilds auf 1333 Pixel limitiert. Da weitere Augmentationen entweder vor oder nach dem Skalieren angewendet werden, wurde in Detection die Methode card_detection.data.build_aug implementiert. Diese nimmt zwei Listen mit Augmentationen entgegen und fügt die Skalierung in der Mitte ein. Um eigene Augmentationen zu verwenden, müssen diese in Detection train.py in einer der Listen von build_aug eingetragen werden. Wie in Abschnitt 3 erwähnt, bestehen große Unterscheide zwischen den einzelnen Datensätzen. Um diese auszugleichen wurde sich in dieser Arbeit für die Daten Augmentationen Zuschneiden und Kontrastanpassung entschieden. Für die Kontrastanpassung wird ImageEnhance aus der Python Bibliothek Pillow genutzt, wobei die Anwendungsstärke gleichverteilt zufällig aus dem Bereich $0,9$ bis $1,2$ gewählt wird. Für die Zuschneide Augmentation wurde die von Detectron2[52] zur Verfügung gestellte RandomCrop Augmentation erweitert. Diese legt nun um jede Boundingbox, die zu einem Eingabebild angegeben ist, ein Padding mit definierbarer Größe und beschneidet ein Bild nie kleiner. Padding, zu deutsch Polsterung, meint in diesem Kontext die Vergrößerung aller Boundingboxen um eine Anzahl an Pixeln. Eine weitere Datenaugmentation, welche Gaußsches Rauschen auf ein Bild addiert, wurde implementiert und getestet. Diese wird auch von Chamveha et al.[5] genutzt. Der Grund für diese Augmentation liegt darin, dass NN

anfällig gegenüber Störungen in der Eingabe sind. Dabei können diese Störungen so unauffällig sein, dass ein menschlicher Betrachter sie nicht einmal war nimmt. Die sogenannten Adversarial Perturbations verändern ein Eingabebild minimal, sodass sich die Ausgabe eines trainierten NN fälschlicherweise ändert. Diese minimale Änderung kann als addiertes Rauschen gesehen werden.[39] Rusak et al.[39] zeigen, dass das einfache Addieren von Gaußschem Rauschen während des Trainings einen effektiven Schutz vor diesen Störungen darstellt. Die Verteilung des Rauschens wurde dabei wie in [39] für ein ResNet50[15] vorschlagen gewählt. In dieser Arbeit wird ein Mittelwert von 0 und eine Standardabweichung von $0,5$ genutzt. Um die Ergebnisse auf rauschfreien Bildern nicht zu verschlechtern, schlagen die Autoren der Arbeit[39] zudem vor, nur 50% der Bilder zu augmentieren. Dieser Vorschlag wird auch in dieser Arbeit umgesetzt. Um das Addieren von Gaußschem Rauschen zu testen, wurden verschiedene Trainings mit dem im vorherigen Abschnitt angepassten Mask R-CNN[16] durchgeführt. Die Ergebnisse sind in Abbildung 24 (Mitte) zu sehen. Dabei ist ein Trainingsdurchlauf mit der beschriebenen Augmentation in orange und ein Durchlauf ohne in blau gezeigt. Bei dem Training mit addiertem Gaußschem Rauschen kam es zu einer Instabilität zu Beginn des Trainings. Um diese zu vermeiden wurde die Basislernrate (siehe Abschnitt 4.4.1.1) auf $0,001$ verkleinert. Dies löste die Instabilität auf und erzielte schlussendlich das beste Ergebnis. Gerade auch wegen der von Rusak et al.[39] versprochenen zusätzlichen Robustheit gegenüber Bildstörungen wurde das Addieren von Gaußschem Rauschen beibehalten.

4.4.3 Backbone

Alle Trainings wurden bisher mit einem Mask R-CNN[16] und dem Backbone ResNet101[15] durchgeführt. Detectron2[52] stellt jedoch auch vortrainierte NN mit den Backbones ResNet50 und ResNeXt101 zur Verfügung (siehe Tabelle 4). Weitere Trainings sollten deshalb klären, welcher Backbone die besten Ergebnisse erzielt. Zusätzlich wurden die NN dabei sehr lange trainiert, um herauszufinden wann es zum Overfitting der Netze kommt. Die Graphen mit dem über alle Batches gemittelten Loss des Validierungsdatensatzes zu den Trainings sind in Abbildung 24 (rechts) zu sehen. Zu erkennen ist, dass der Loss des ResNet101 am tiefsten fällt. Die finale Trainingsdauer wurde auf 20000 Iterationen festgelegt. Diese wurde so ermittelt, dass der zweite Lernraten Decay zu dem Zeitpunkt stattfindet an dem die Lernrate wieder ansteigt. Die Ergebnisse, welche mit den gefunden Hyperparamtern erzielt wurden, sind in Abschnitt 5 unter Mask R-CNN nachzulesen.

4.4.4 Faster R-CNN

Bisher wurden alle Trainings mit dem Instanz Segmentierungs NN Mask R-CNN[16] durchgeführt. Von wesentlicher Bedeutung für diese Arbeit ist jedoch genauso der Ob-

jektdetektor Faster R-CNN[35]. Das Finden der Hyperparamter für dieses NN soll nun etwas kompakter dargestellt werden. Zunächst wurde ein Faster R-CNN mit den für das Mask R-CNN ermittelten Hyperparametern trainiert. Dieses erzielt so gute Leistungen, dass es in den Ergebnissen als Faster R-CNN (1) besprochen werden soll. Jedoch sollten weitere Hyperparameter Kombinationen getestet werden um zu validieren, dass die Ergebnisse nicht weiter verbessert werden können. Erste Trainings zeigten, dass die Hyperparameter des Mask R-CNN weiterhin die besten Ergebnisse erzielen. Da es jedoch zu einigen Instabilitäten zu Beginn verschiedener Trainings kam, wurde die Basislernrate weiter auf $0,0005$ gesenkt. Mit einem langen Training, zu sehen in Abbildung 25 (blau), wurde die Anzahl der zu trainierenden Iterationen (38120) ermittelt. Dabei wurde genauso wie in Abschnitt 4.4.3 vorgegangen. Da weiterhin Instabilitäten zu Beginn des Trainings auftraten, wurde die Lernrate ein weiteres Mal auf $0,00025$ gesenkt. Das im Folgenden trainierte NN Faster R-CNN (2) erzielte den besten Loss auf dem Validierungsdatensatz. Wie in Abschnitt 5 zu sehen ist, konnte es jedoch keine großen Erfolge in der Evaluation erzielen. Das bisher erfolgreichste Faster R-CNN in der Evaluation blieb Faster R-CNN(1), welches mit einer größeren Lernrate kürzer trainierte. Der Gedanke für die finalen beiden Trainings war es, die insgesamt zu trainierenden Iterationen (38120) zu verkürzen. Deshalb wurden sie auf den Wert gesetzt, an dem das lange Training stagnierte. Dies passierte bei ungefähr 30000 Iterationen. Mit der neuen Trainingslänge wurde nun einmal mit einer Lernrate von $0,00025$ trainiert und wegen der Stärke von Faster R-CNN (1) noch einmal mit einer Lernrate von $0,001$. Dabei schnitt das Training mit einer Lernrate von $0,001$, welches in Abbildung 25 (grün) gezeigt ist, am besten ab. Das entstandene NN wird als Faster R-CNN (3) bezeichnet.

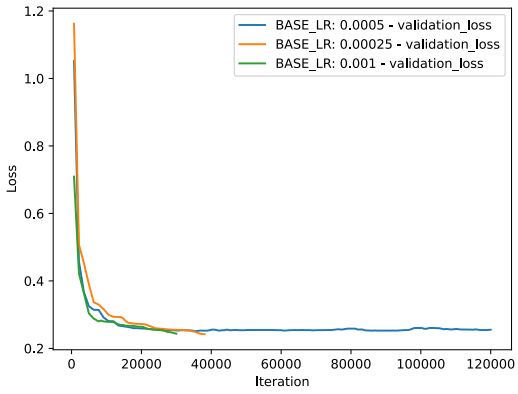


Abbildung 25: Der Graph zeigt den gemittelten Loss aller Batches auf dem Validierungsdatensatz während des Trainings verschiedener NN. Mit Hilfe des langen Trainings, dargestellt in blau, wurden die Trainingsdauern der finalen Faster R-CNN Netze festgelegt. Für das orange Netz wurde der Zeitpunkt des Overfittings und für das grüne Netz der Zeitpunkt des Stagnieren des blauen Graphen gewählt.

5 Evaluation

Zu Beginn dieser Arbeit wurde verdeutlicht, dass zum Berechnen des HTQ nur Boundingboxen um Herz und Lunge nötig sind. Aus diesem Grund wurden die von Mask R-CNN[16] ausgegebenen Masken als Boundingboxen aufgefasst und entsprechend evaluiert. Dabei steht im folgenden (Box) für die Auswertung der Boundingboxen und (Maske) für die Auswertung der Boundingbox, welche aus den ausgegebenen Masken erstellt wurden. Eine Beispielausgabe der NN dieser Arbeit ist in Abbildung 26

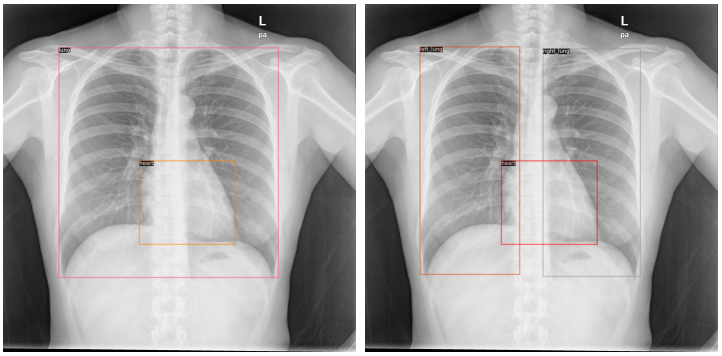


Abbildung 26: Auf den Röntgenaufnahmen[19] sind die von Mask R-CNN und Faster R-CNN ausgegeben Boundingboxen zu sehen. Dabei wurden die Masken von Mask R-CNN als Boundingboxen aufgefasst und wie abgebildet ausgegeben. Die Bilder verdeutlichen die zwei möglichen Berechnungen das HTQ.

gezeigt. Dabei sind auf den Bildern jeweils die beiden möglichen Berechnungen des HTQs gezeigt. Einmal wird dabei die Lungenbox und einmal die Boxen der Lungenflügel verwendet um den Querdurchmesser der Brust zu berechnen. Durch diese unterschiedlichen Berechnung, soll Frage drei aus Abschnitt 1 beantworteten werden. Im Folgenden werden die Ergebnisse dieser Arbeit vorgestellt, welche später im Abschnitt 6 diskutiert werden sollen. Ein Überblick der evaluierten Netze ist in Tabelle 5

Tabelle 5: Die Hyperparameter der evaluierten NN

Name	Lernrate	Iter. gesamt	Decay Schritte	Gamma	Loss
Mask R-CNN	0,001	20000	15555 & 18520	0,1	0,371
Faster R-CNN (1)	0,001	20000	15555 & 18520	0,1	0,251
Faster R-CNN (2)	0,00025	38120	29649 & 35299	0,1	0,242
Faster R-CNN (3)	0,001	30000	23333 & 27780	0,1	0,244

In der Tabelle sind nur die wichtigsten Unterschiede der NN abgebildet. Weitere Hyperparamter sind in Abschnitt 4.4 nachzulesen. Iter. kürzt in der Tabelle dabei Iterationen ab. Da der über alle Batches gemittelte finale Loss des Validierungsdatensatzes zuvor nur ungefähr zu erkennen war, ist dieser ebenfalls in der Tabelle abgebildet. Wichtig ist dabei Mask R-CNN[16] nicht mit Faster R-CNN[35] zu vergleichen, da bei Mask R-CNN ein zusätzlicher Loss für die Segmentierungsmasken verwendet wird.[16]

Tabelle 6: Der Jaccard Score (IoU) der NN.

Name	Linker Lungenflügel	rechter Lungenflügel	Lunge	Herz
Mask R-CNN (Box)	0,9130	0,8974	0,9498	0,6697
Mask R-CNN (Maske)	0,8910	0,8796	0,9137	0,6619
Faster R-CNN (1)	0,9186	0,9006	0,9514	0,6892
Faster R-CNN (2)	0,9161	0,8991	0,9495	0,6761
Faster R-CNN (3)	0,9149	0,8991	0,9505	0,6865
U-Net	–	–	0,9426	0,7049

Tabelle 7: Der Dice Score (IoU) der NN.

Name	Linker Lungenflügel	Rechter Lungenflügel	Lunge	Herz
Mask R-CNN (Box)	0,9487	0,9393	0,9738	0,7996
Mask R-CNN (Maske)	0,9360	0,9286	0,9533	0,7940
Faster R-CNN (1)	0,9525	0,9452	0,9747	0,8141
Faster R-CNN (2)	0,9535	0,9443	0,9737	0,8045
Faster R-CNN (3)	0,9502	0,9409	0,9743	0,8121
U-Net	–	–	0,9695	0,8237

zu sehen. Für die Auswertung wurde der private Datensatz der Uniklinik Würzburg (siehe Abschnitt 4.2.5) genutzt. Dieser enthält 108 Röntgen Thorax Aufnahmen auf denen Herzen und Lungenflügel von einem Radiologen maskiert wurden. Des weiteren stehen neben den Röntgenaufnahmen auch manuelle Messungen des HTQ eines Radiologen zur Verfügung. Als Referenz für die Berechnung des HTQ durch Segmentierungsnetze ist in allen folgenden Tabellen die Auswertung von Hasler[14] U-Net[37] auf dem Datensatz der Uniklinik gezeigt. Zunächst soll der Jaccard und Dice Score der einzelnen Modelle vorgestellt werden. Die Ergebnisse aller vier Objekttypen ist in den Tabellen 6 und 7 zu sehen, wobei für das U-Net nur Werte für die Lungen und Herzen eingetragen sind. Das liegt daran, dass dieses nur die Klasse Lunge und Herz detektiert. Sehr auffällig auf den beiden Tabellen ist das schlechte Abschneiden der Herz Boundingboxen. Der Grund dafür liegt in den sich unterscheidenden Trainings- und Testdaten zu den Herzen. Die Masken und somit die Boundingboxen des Radiologen sind oben deutlich höher als die in Abbildung 20 gezeigten Trainingsmasken und Boxen. Für die folgenden Auswertung ist es wichtig zu erwähnen, dass die NN dieser Arbeit auf einigen Bilder zu einem Objekttyp keine Boundingbox oder Maske ausgaben. Bei den vorgestellten Netzen fehlten nur Masken zu linkem und rechtem Lungenflügel. Bei der Berechnung konkreter Werte wurde dies, wenn möglich, ignoriert oder es wurde der entsprechende Wert auf Null gesetzt. Fehlte also zum Beispiel der rechte Lungenflügel, wurde der ID nur aus dem linken Lungenflügel berechnet. In Tabelle 8 ist zu

Tabelle 8: Abgebildet ist die Korrelation zu den jeweils korrekten Werten (oben) und dem Jaccard Score (unten), gemessen mit dem Empirischen Korrelationskoeffizienten.

Name	HTQ (L)	HTQ (LF)	MLD	MRD	ID (L)	ID (LF)
Mask R-CNN (Box)	0,9379	0,6606	0,9096	0,9239	0,9831	0,866
	−0,1587	−0,7171	0,0426	0,0189	0,3262	0,6826
Mask R-CNN (Maske)	0,7308	0,6326	0,8795	0,9218	0,8878	0,8646
	−0,3928	−0,6527	0,0445	0,0371	0,6478	0,6808
Faster R-CNN (1)	0,8757	0,6722	0,807	0,9353	0,9883	0,8868
	−0,1207	−0,5921	0,0881	0,0161	0,3404	0,6349
Faster R-CNN (2)	0,9166	0,9201	0,8797	0,9164	0,9716	0,9849
	−0,1383	−0,4166	0,121	0,0431	0,3419	0,4556
Faster R-CNN (3)	0,9355	0,6719	0,8997	0,9248	0,9816	0,8682
	−0,0844	−0,6731	0,1032	0,0143	0,4218	0,7146
U-Net	0,8415	–	0,8062	0,9325	0,8958	–
	−0,1946	–	−0,0561	−0,0892	0,1437	–

(L) steht hierbei für die Berechnung des HTQ aus Lunge und Herz und (LF) für die Berechnung des HTQ aus den beiden Lungenflügeln und dem Herz.

jedem NN oben die Korrelation (siehe Abschnitt 2.11.3) der Ausgabewerte zu den korrekten Wert aus dem Testdatensatz gezeigt. Dabei wurde für die HTQs der manuell gemessene Wert des Radiologen verwendet und die weiteren Werte aus den Masken des Radiologen berechnet. Dadurch sagt die Auswertung aus, wie ähnlich die berechneten Werte der Netze zu denen eines Experten sind. Dabei weisen alle Faster R-CNN[35] nach Fahrmeir et al.[8, Kapitel 3.4.1] eine starke Korrelation auf. Zu jedem Netz ist unten ebenfalls die Korrelation der ausgegeben Werte zu der IoU zu sehen. Dabei wurde die jeweilige IoU durch das Mitteln der IoUs aller für die Berechnung des jeweiligen Werts nötigen Boundingboxen ermittelt. Diese Auswertung zeigt, wie wenig die Metrik Jaccard Score über das Ergebnis der HTQ Berechnung aussagt. Denn während für die Berechnung von MLD, MRD und ID nur die Breite einer Boundingbox von Bedeutung ist, spielt bei der IoU noch der Faktor Höhe eine Rolle. Dies zeigt sich vor allem an den unterschiedlich eingerahmten Herzen. Um herauszufinden, wie gut die genaue Berechnung der gezeigten Werte tatsächlich ist, wurde die mittlere Differenz zwischen den Werten des NN und den Werten des Radiologen berechnet (Ausgabe − Wert des Radiologen). In Tabelle 9 ist diese zusammen mit der dazugehörigen Standardabweichung (unten) abgebildet. Aus der Tabelle ist zu entnehmen, dass der Querdurchmessers der Lunge (ID) besser von der Lungen Boundingbox berechnet wird. Dies wirkt sich direkt auf die HTQ Berechnung aus. Am besten konnte in dieser Auswertung Faster R-CNN (1) abschneiden. Nun soll die tatsächliche Kardiomegalie Erkennung anhand der Klassifikationsmetriken aus Abschnitt 2.11.1 ausgewertet werden. Dabei wurden drei unterschiedliche Auswertungen durchgeführt, um die Ergebnisse im nächsten Abschnitt besser diskutieren zu können. In allen drei Auswertungen, wurde der HTQ in

Tabelle 9: Abgebildet ist die mittlere Differenz zu den jeweils korrekten Werten (oben) mit der dazugehörigen Standardabweichung (unten).

Name	HTQ (L)	HTQ (LF)	MLD	MRD	ID (L)	ID (LF)
Mask R-CNN	$0,0161\pm$	$0,0301\pm$	$-0,0024\pm$	$0,0015\pm$	$-0,0013\pm$	$0,0136\pm$
(Box)	$0,0294$	$0,0915$	$0,0209$	$0,0147$	$0,0136$	$0,051$
Mask R-CNN	$0,0142\pm$	$0,0218\pm$	$-0,0094\pm$	$-0,0018\pm$	$-0,0119\pm$	$-0,0173\pm$
(Maske)	$0,0679$	$0,0928$	$0,024$	$0,0151$	$0,0424$	$0,0514$
Faster R-CNN (1)	$0,009\pm$	$0,018\pm$	$-0,0090\pm$	$0,0024\pm$	$-0,0013\pm$	$-0,0083\pm$
	$0,041$	$0,0769$	$0,0298$	$0,0136$	$0,0114$	$0,0414$
Faster R-CNN (2)	$0,0177\pm$	$0,0183\pm$	$-0,0047\pm$	$0,0051\pm$	$-0,0011\pm$	$-0,002\pm$
	$0,0345$	$0,0338$	$0,0239$	$0,0154$	$0,0174$	$0,013$
Faster R-CNN (3)	$0,0154\pm$	$0,0291\pm$	$-0,0023\pm$	$0,0018\pm$	$-0,0003\pm$	$-0,0091\pm$
	$0,0299$	$0,0889$	$0,022$	$0,0147$	$0,0138$	$0,0502$
U-Net	$-0,0143\pm$	–	$-0,0146\pm$	$-0,0043\pm$	$0,0086\pm$	–
	$0,046$	–	$0,0299$	$0,0142$	$0,0337$	–

(L) steht hierbei für die Berechnung des HTQ aus Lunge und Herz und (LF) für die Berechnung des HTQ aus den beiden Lungenflügeln und dem Herz.

die Klassen Kardiomegalie (HTQ$> 0,5$) und keine Kardiomegalie (HTQ$\leq 0,5$) eingeteilt. Wie zuvor erwähnt, wurden von den Modellen Mask R-CNN und Faster R-CNN zu manchen Organen auf den 108 Bildern des Testdatensatzes keine Boundingboxen ausgeben. In der ersten Auswertung, abgebildet in Tabelle 10, wurden Röntgenbilder in eine dritte Klasse eingeteilt, wenn eine zur Berechnung des HTQ nötige Box fehlte. Dabei wurde zwischen der Berechnung des HTQ aus Lunge und Lungenflügel unterschieden. Diese Evaluation ist jedoch nur aus Gründen der Vollständigkeit enthalten. Da Domänenwissen vorhanden ist, kann folgender Nachbearbeitungsschritt eingeführt werden. Bei einer HTQ Berechnung, bei der eine Boundingbox fehlt, wird der HTQ auf eins gesetzt. Dies begründet sich darin, dass das Ziel der NN die Erkennung einer folgenschweren Krankheit ist: Es ist also viel wichtiger einen Verdachtsfall zu erkennen, als einen gesunden Patienten als krank einzustufen. Alle folgenden Auswertungen wurden mit dem beschriebenen Nachbearbeitungsschritt durchgeführt. In Tabelle 11 ist die Auswertung der Kardiomegalie Erkennung noch einmal anhand der manuell gemessenen HTQs des Radiologen zu sehen. In Tabelle 12 ist die selbe Auswertung anhand von HTQs, welche aus den Masken das Radiologen berechnet wurden, abgebildet. Dabei fällt ein Unterschied auf, welcher im nächsten Abschnitt genauer diskutiert werden soll. Ebenfalls lassen sich folgende Beobachtungen machen: Das Faster R-CNN mit dem besten Loss auf dem Validierungsdatensatz schneidet in der Auswertung am schlechtesten ab. Es lässt sich nicht feststellen, ob die Detektion der Lunge oder der einzelnen Lungenflügel den ID besser berechnet. Mask R-CNN erzielt, obwohl es sowohl Boundingboxen als auch Masken zum Training nutzte, keine besseren Ergebnisse

Tabelle 10: Die Auswertung der Kardiomegalie Erkennung mit einer dritten Klasse für Röntgenbilder mit fehlenden Boundingboxen.

Name	Accuracy	Precission	Sensitivity	Specificity	F1-Score
Mask R-CNN (Box)	0,8981	0,8507	0,9828	0,8	0,912
	0,8889	0,8485	0,9655	0,8	0,9032
Mask R-CNN (Maske)	0,9167	0,8769	0,9828	0,84	0,9268
	0,8704	0,8333	0,9483	0,78	0,8871
Faster R-CNN (1)	0,9167	0,8889	0,9655	0,86	0,9256
	0,9074	0,875	0,9655	0,84	0,9180
Faster R-CNN (2)	0,8981	0,8507	0,9828	0,8	0,912
	0,8889	0,8382	0,9828	0,78	0,9048
Faster R-CNN (3)	0,8981	0,8507	0,9828	0,8	0,912
	0,8981	0,8615	0,9655	0,82	0,9106
U-Net	0,9074	0,9	0,9310	0,88	0,9153

Zu jedem Netz ist die HTQ Berechnung aufgrund des IDs der Lungen Boundingbox oben gezeigt. Darunter befindet sich die Auswertung des Kardiomegalie, basierend auf dem ID der Lungenflügel. Da es sich hierbei um die Auswertungen von mehr als zwei Klassen handelt, stellen Precision und Sensitivity die jeweilige Auswertung auf der Kardiomegalie Klasse dar. Specificity meint hier die Sensitivity der Klasse keine Kardiomegaflie.

Tabelle 11: Die Auswertung der Kardiomegalie Erkennung, mit dem beschriebenen Nachbearbeitungsschritt, anhand der vom Radiologen gemessenen HTQs.

Name	Accuracy	Precission	Sensitivity	Specificity	F1-Score
Mask R-CNN (Box)	0,8981	0,8507	0,9828	0,8	0,912
	0,9074	0,8529	1	0,8	0,9206
Mask R-CNN (Maske)	0,9167	0,8769	0,9828	0,84	0,9268
	0,8889	0,8382	0,9828	0,78	0,9048
Faster R-CNN (1)	0,9167	0,8889	0,9655	0,86	0,9256
	0,9167	0,8769	0,9828	0,84	0,9268
Faster R-CNN (2)	0,8981	0,8507	0,9828	0,8	0,912
	0,8889	0,8382	0,9828	0,78	0,9048
Faster R-CNN (3)	0,8981	0,8507	0,9828	0,8	0,912
	0,9167	0,8657	1	0,82	0,928
U-Net	0,9074	0,9	0,9310	0,88	0,9153

Zu jedem Netz ist die HTQ Berechnung aufgrund des IDs der Lungen Boundingbox oben gezeigt. Darunter befindet sich die Auswertung der Kardiomegalie, basierend auf dem ID der Lungenflügel.

Tabelle 12: Die Auswertung der Kardiomegalie Erkennung, mit dem beschriebenen Nachbearbeitungsschritt, anhand der aus den Masken des Radiologen berechneten HTQs.

Name	Accuracy	Precission	Sensitivity	Specificity	F1-Score
Mask R-CNN (Box)	0,9259	0,9403	0,9403	0,9024	0,9403
	0,9352	0,9412	0,9552	0,9024	0,9481
Mask R-CNN (Maske)	0,9259	0,9538	0,9254	0,9268	0,9394
	0,9167	0,9265	0,9403	0,878	0,9333
Faster R-CNN (1)	0,9444	0,9841	0,9254	0,9756	0,9538
	0,9259	0,9538	0,9254	0,9268	0,9394
Faster R-CNN (2)	0,9259	0,9403	0,9403	0,9024	0,9403
	0,9167	0,9265	0,9403	0,8780	0,9333
Faster R-CNN (3)	0,9259	0,9403	0,9403	0,9024	0,9403
	0,9444	0,9552	0,9552	0,9268	0,9552
U-Net	0,8981	0,9667	0,8657	0,9512	0,9134

Zu jedem Netz ist die HTQ Berechnung aufgrund des IDs der Lungen Boundingbox oben gezeigt. Darunter befindet sich die Auswertung der Kardiomegalie, basierend auf dem ID der Lungenflügel

als die Objektdetektoren. Das Segmentierungsnetz U-Net wird von Mask R-CNN und Faster R-CNN übertroffen. Die aufgezählten Beobachtungen werden im nächsten Kapitel genauer diskutiert. Im Gespräch mit einem Radiologen der Uniklinik Würzburg wurde nach einer Auswertung gesucht, welche die Neuronalen Netze unter möglichst realen Bedingungen testet. Dabei stellte sich heraus, das im Klinikalltag der HTQ auf Röntgen Thorax Aufnahmen zunächst subjektiv von einem Radiologen geschätzt wird. Sollte sich dabei eine Auffälligkeit ergeben, wird der HTQ genaue gemessen. Der Einsatz eines NN als solches Warnsystem wäre nach dem Radiologen der UKW eine Möglichkeit dieses in den realen Einsatz zu integrieren. Um auszuwerten, wie gut ein solches Warnsystem mit den Netzen dieser Arbeit funktionieren würde, sollten die vorgestellten NN die 108 Röntgenbilder des Testdatensatzes in drei Schweregrade einteilen: Keine Kardiomegalie (HTQ $\leq 0,45$), Kardiomegalie Warnung ($0,45 <$ HTQ $\leq 0,55$) und Kardiomegalie (HTQ $> 0,55$). Der Radiologe der UKW simulierte für diese Arbeit die subjektive Einschätzung der Gruppe durch einen Blick auf die jeweiligen Röntgen Thorax Aufnahmen. Beide Schweregradeinteilungen wurde mit der korrekten Einteilung verglichen, welche anhand der gemessenen HTQs aus dem privaten Datensatz der UKW ermittelt wurde. Die Ergebnisse von Faster R-CNN (1) und Faster R-CNN (3) werden in Tabelle 13 mit der Einschätzung eines Radiologen in der Gruppeneinteilung verglichen. Dabei wurde bei den NN weiterhin der genannte Nachbearbeitungsschritt angewandt. Für diese Auswertung wurden sowohl Faster R-CNN (1), als auch Faster R-CNN (3) ausgewählt, da beide Netze unterschiedliche Stärken zeigten. Bei Faster R-

Tabelle 13: Gezeigt sind die Ergebnisse der Schweregradeinteilung, in der NN mit der subjektiven Einschätzung eines Radiologen verglichen wurden.

Name	Schweregrad	Precission	Sensitivity
Radiologe	Keine Kardiomegalie	$0,9231$	$0,6$
	Kardiomegalie Warnung	$0,7674$	$0,9167$
	Kardiomegalie	$0,9615$	$0,9615$
Faster R-CNN (1)	Kardiomegalie Warnung	1	$0,45$
	Kardiomegalie Warnung	$0,6939$	$0,9444$
	Kardiomegalie	$0,96$	$0,9231$
Faster R-CNN (3)	Kardiomegalie	1	0.6
	Kardiomegalie Warnung	$0,7727$	$0,9444$
	Kardiomegalie	$0,9615$	$0,9615$

CNN (1) liegt diese in der Genauigkeit der Messung und bei Faster R-CNN (3) in der Kardiomegalie Klassifikation. In der Schweregradeinteilung erreichte Faster R-CNN (1) eine Accuracy von $0,8426$, der Radiologe eine Accuracy von $0,8796$ und Faster R-CNN (3) eine Accuracy von $0,8889$. Eine genauere Auswertung findet sich in Tabelle 13, in der weiterhin Faster R-CNN (3) die besten Ergebnisse zeigt. Generell fallen die hohen Sensitivitywerte der Klassen Kardiomegalie Warnung und Kardiomegalie positiv auf. Sollte ein Patient also Kardiomegalie haben, wird vor dieser in den aller meisten Fällen auch gewarnt. Ein genauerer Blick in die Daten zeigt ebenfalls, dass die Warnung bei einem HTQ $> 0,45$ sehr fein eingestellt ist. Beispielsweise Faster R-CNN (3) warnt bei acht Aufnahmen, welche nach der Messung von keiner klinischen Relevanz sind.

6 Diskussion und Ausblick

6.1 Widersprüche in der Evaluation

Im letzten Kapitel wurden verschiedene Auswertungen durchgeführt. Dabei sind einige widersprüchliche Ergebnisse entstanden, welche in diesem Abschnitt untersucht werden sollen. Zunächst fällt auf, dass das Neuronale Netz Faster R-CNN (2) in den Auswertungen auf dem Testdatensatz keine Spitzenergebnisse erzielt. Dabei schnitt dieses Netz am besten beim Loss des Validierungsdatensatzes ab. Eine mögliche Erklärung ist das sogenannte Overfitting auf dem Validierungsdatensatz (laut Hekalo). Ist dieser zu ähnlich zum Trainingsdatensatz kann er keine gute Aussage über ungesehene Eingaben machen und zeigt das Overfitting zu spät an. Hierbei war es im Nachhinein ein Fehler Röntgenaufnahmen aller Datensätze für den Validierungsdatensatz zu wählen. Ein besseres Vorgehen, wäre das Nutzen des Montogomery County Chest X-ray Datensats[19] gewesen. Des weiteren kann in der Auswertung festgestellt werden, dass der Korrelationskoeffizient stark schwangt und teilweise kaum mit den Ergebnissen der Kardiomegalie Klassifikation zusammenhängt. Nach Held[17] wird der Empirische Korrelationskoeffizient durch extreme Werte und Ausreißer stark beeinflusst. Betrachtet man Abbildung 27 scheinen diese die Schwankungen zu verursachen.

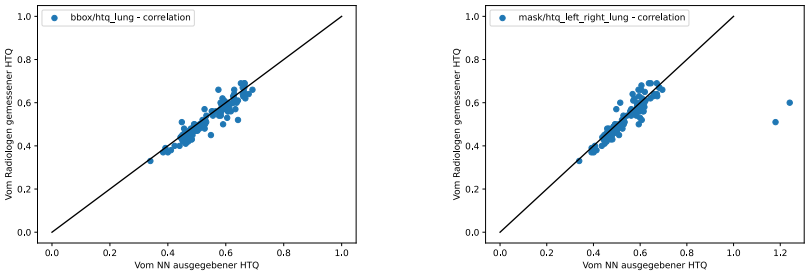


Abbildung 27: **Links:** Der beste Korrelationskoeffizient des HTQs aus Tabelle 8 (0,9379). **Rechts:** Der schlechteste Korrelationskoeffizient des HTQs aus Tabelle 8 (0,6326).

Während die Streudiagramme sehr ähnlich aussehen, ist links ein Korrelationskoeffizient von 0,9379 und rechts ein Korrelationskoeffizient von 0,6326 zu sehen. Dabei unterscheiden sich die beiden Diagramme im Wesentlichen nur um zwei Ausreißer im rechten Diagramm. Diese kommen zustande, da die NN zu manchen Organen des Datensatzes keine Boundingboxen ausgeben. Während das Nicht-Betrachten dieser Werte keine sinnvolle Option darstellt, wäre der Rangkorrelationskoeffizient eine mögliche alternative zum Empirischen Korrelationskoeffizienten gewesen. Aus dem Grund der Vergleichbarkeit zu anderen Veröffentlichungen wurde jedoch der Empirische Korrelationskoeffizienten beibehalten. Ebenfalls fällt der Unterschied zwischen den Tabellen 11 und 12 auf. In diesen wurde die Kardiomegalie Erkennung einmal anhand manuell gemessener HTQs und einmal anhand von HTQs, welche sich aus den korrekten Masken

ergeben, ausgewertet. Dabei wurden sowohl die Werte, als auch die Masken von demselben Radiologen erstellt. Aus diesem Grund wurde der Radiologe gebeten, sich die beiden HTQs zu einer Röntgen Thorax Aufnahme noch einmal anzusehen. Dabei zeigte sich zwar eine generelle Übereinstimmung, jedoch wiesen einzelne Werte deutliche Unterschiede auf. Dabei fand der Radiologe Fehler bei beiden Werten. Die gemessenen HTQs waren dabei eher zu klein und die aus den Masken berechneten HTQs zu groß. In Grenzfällen fallen diese Unterscheide ins Gewicht. Li et al.[24] kombinieren die Messergebnisse von drei Radiologen, um Vergleichswerte für ihre Arbeit zu erhalten. Dies scheint ein wichtiger Schritt zu sein, um aussagekräftige Vergleichswerte zu erhalten.

6.2 Verbesserungen durch Objektdetektion

Die wichtigste Frage dieser Arbeit ist, wie Objektdetektoren (Faster R-CNN[35]) im Vergleich zu Segmentierungs NN abschneiden. Wie in Tabelle 14 zu sehen, erreichen die Objektdetektoren in dieser Arbeit bessere Ergebnisse als das Vergleichssegmentierungsnetz U-Net von Hasler[14]. Des weiteren erzielen sie ähnliche Ergebnisse wie der letzten Stand der Technik. Das zeigt, dass Objektdetektoren die selbe Eignung für die Kardiomeglie Erkennung haben, wie die bisher genutzten Segmentierungsnetze. Dabei soll hervorgehoben werden, dass das Erstellen von Trainingsbeispielen für die Objektdetektion einen viel kleineren Aufwand darstellt. Währen für die Segmentierung genaue Masken gezeichnet werden müssen, reichen der Objektdetektion einfache Rahmen. Eine Etablierung der Objektdetektion für die Kardiomeglie Erkennung würde deshalb größeren Datensätze ermöglichen. Diese könnten in Zukunft die NN weiter verbessern.

Tabelle 14: Eine Übersicht der Ergebnisse dieser Arbeit.

Name	Accuracy	Sensitivity	Specificity	F1	KK	Differenz
Mask R-CNN (Maske)	$0,9167$	$0,9828$	$0,84$	$0,9268$	$0,7308$	$0,0142 \pm 0,0679$
	$0,8889$	$0,9828$	$0,78$	$0,9048$	$0,6326$	$0,0218 \pm 0,0928$
Faster R-CNN (2)	$0,8981$	$0,9828$	$0,8$	$0,912$	$0,9166$	$0,0177 \pm 0,0345$
	$0,8889$	$0,9828$	$0,78$	$0,9048$	$0,9201$	$0,0183 \pm 0,0338$
Faster R-CNN (3)	$0,8981$	$0,9828$	$0,8$	$0,912$	$0,9355$	$0,0154 \pm 0,0299$
	$0,9167$	1	$0,82$	$0,928$	$0,6719$	$0,0291 \pm 0,0889$
U-Net	$0,9074$	$0,9310$	$0,88$	$0,9153$	$0,8415$	$-0,0143 \pm 0,046$
Faster R-CNN (3)	$0,9259$	$0,9403$	$0,9024$	$0,9403$	–	–
(HTQ aus Masken)	$0,9444$	$0,9552$	$0,9268$	$0,9552$	–	–
State of the Art	$0,953$	$0,972$	$0,927$	–	$0,982$	$0,0004 \pm 0,0133$
Radiologe	$0,8796$	$0,9391$	–	–	–	–
Faster R-CNN (3)	$0,8889$	$0,953$	–	–	–	–

Zu jedem Objektdetektor ist die HTQ Berechnung aufgrund des IDs der Lungen Boundingbox oben gezeigt. Darunter befindet sich die Auswertung der Kardiomegalie, basierend auf dem ID der Lungenflügel. Dabei wurde der in Abschnitt 5 beschriebene Nachbearbeitungsschritt angewendet. KK meint den Korrelationskoeffizienten, welcher die Korrelation zu den vom Radiologen gemessenen HTQs misst. Die Differenz zeigt die mittlere Differenz und die dazugehörige Standardabweichung von ausgegebenen und von einem Radiologen gemessenen HTQs. State of the Art zeigt die Ergbnisse von Li et al.[24]. Die letzten beiden Werte stellen die Auswertung der Einteilung in Schweregrade dar. Die Sensitivity ist dabei der Durchschnitt der Sensitivity von den Klassen Kardiomegalie Warnung und Kardiomegalie.

6.3 Instanz Segmentierung

Um zu untersuchen, ob Masken komplementäre Informationen enthalten, welche die NN verbessern können, wurde das Instanz Segmentierungs NN Mask R-CNN[16] trainiert. Dieses gibt Boundingboxen und Segmentierungsmasken aus und enthält deshalb zu beiden Informationen. Wie man aus Tabelle 14 entnehmen kann, zeigt Mask R-CNN auch mit die besten Ergebnisse. Allerdings zeigt Mask R-CNN damit auch keine Verbesserung, welche den enormen zusätzlichen Aufwand für die Datengenerierung rechtfertigen würde. Da Mask R-CNN wie Faster R-CNN[35] auf Regionsvorschläge (siehe Abschnitt 2.10) angewiesen ist, sind deutliche Verbesserungen vermutlich nicht möglich. In Zukunft sollten also Objektdetektionsnetze mit Segmentierungsnetzen verglichen werden.

6.4 Lunge oder Lungenflügel

Eine weitere Frage dieser Arbeit stellte dar, wie der Querdurchmesser der Brust am besten detektiert werden kann. Dazu wurde die Detektion der vollständigen Lunge mit der Detektion der Lungenflügel verglichen. Im Fall der Lungenflügel ergibt sich der ID aus linker Kante des linken Lungenflügels und rechter Kante des rechten Lungenflügels. Die von den NN dieser Arbeit ausgegebenen Boundingboxen sind in Abbildung 26 gezeigt. Betrachtet man Tabelle 14, lässt sich die gestellte Frage nicht wirklich beantworten. Während die Ergebnisse ohne den in Abschnitt 5 beschrieben Nachbearbeitungsschritt für die Detektion der Lunge sprechen, gleichen sich die Ergebnisse mit der Nachbearbeitung an. Das essentielle Problem stellt hier die fehlende Ausgabe von ein bis zwei Boundingboxen der Lungenflügel dar. Durch das Vorgehen bei der Auswertung von Differenz und Korrelation ergeben sich bei fehlenden Lungenflügelboxen sehr große Werte für den HTQ. Dies führt zu einem schlechteren Korrelationskoeffizienten (siehe Abschnitt 6.1) und einer größeren, schlechteren durchschnittlichen Differenz. Die Auswertung von Faster R-CNN (2), welches zu jedem Objekt des Testdatensatzes eine Boundingbox ausgibt, zeigt, dass es kaum einen Unterschied zwischen den beiden ID Berechnungen gibt. In Zukunft muss deshalb den fehlenden Ausgaben nachgegangen werden, um die Frage der ID Berechnung beantworten zu können. Generell bestünde jedoch auch die Möglichkeit einen Konsensmechanismus zwischen den beiden ID Berechnungen einzuführen und somit beide Ausgaben zu nutzen. Dabei könnte beispielsweise die Konfidenz der einzelnen Ausgaben genutzt werden. Das Fehlen einer Boundingbox würde durch solch einen Mechanismus nicht weiter ins Gewicht fallen. Anzumerken ist, dass dabei keine zusätzliche Arbeit bei der Datengenerierung anfallen würde, da die Boundingboxen der Lungen aus den Lungenflügeln berechnet werden können.

6.5 Medizinischer Einsatz

Um zu klären, wie relevant die in dieser Arbeit erzielten Ergebnisse für den medizinischen Einsatz sind, wurde eine von einem Radiologen vorgeschlagene Auswertung durchgeführt (siehe Abschnitt 5). Diese betrachtete die NN als ein Warnsystem, welches Ärzte auf Kardiomegalie Verdachtsfälle hinweisen soll. In dieser Arbeit konnte gezeigt werden, dass ein automatisches Warnsystem angetrieben von Deep Learning ähnlich effektiv wie schnelle visuelle Einschätzungen eines Radiologen sind (siehe Tabelle 14), welche bisher genutzt werden, um Kardiomegalie Verdachtsfälle zu erkennen. Der Radiologe, welcher die Auswertung vorgeschlagen hatte, schätzte die erzielten Ergebnisse wie folgt ein: „Eine Bestimmung des HTQs mittels neuronaler Netze kann für klinisch[] tätige Radiologen ein Hilfsmittel in der täglichen Diagnostik werden, da so eine schnelle Vorauswahl zwischen Patienten mit und ohne Kardiomegalie möglich ist. So kann die Konzentration schneller auf die pathologischen Befunde gelenkt werden. Auch der Einbau einer Gruppe zwischen „unauffällig“und „ Kardiomegalie“ hat sich als hilfreich erwiesen, da diese Patienten so nochmals für eine weiterführende, genauere Beurteilung markiert wurden. Auch im Hinblick auf die Zuordnung der Patienten zu den einzelnen Gruppen durch das neuronale Netz und einen Radiologen war zu einem Großteil eine Übereinstimmung erkennbar, sodass von einer guten Korrelation des Netzes mit der klinische Einschätzung ausgegangen werden kann.“

6.6 Fazit und Ausblick

In der Arbeit wurde der Ansatz der Objektdetektion und der Instanz Segmentierung mit dem der Segmentierung in der Kardiomegalie Erkennung verglichen. Dabei konnten sowohl mit der Instanz Segmentierung und der Objektdetektion gute Ergebnisse erzielt werden. Dazu zählen unter anderem die hohen Sensitivities, welche von den Neuronalen Netzen dieser Arbeit erreicht wurden. Tabelle 14 zeigt, dass das Segmentierungsnetz U-Net[37] aus Haslers[14] Bachelorarbeit übertroffen wurde und wie ähnlich die Ergebnisse zum letzten Stand der Technik (State of the Art) des Ansatzes der Segmentierung sind. Ebenfalls kann Tabelle 14 entnommen werden, wie gut die Ergebnisse zu den Vergleichswerten von Experten korrelieren. Die Korrelation wird durch die Auswertung für den medizinischen Einsatz zusätzlich unterstrichen. Allerdings bleiben Fragen in der Arbeit offen und der letzte Stand der Technik, aufgestellt von Segmentierungs NN[24], wird nicht übertroffen. Besonders die fehlenden Ausgaben zu den Lungenflügeln stellen ein offenes Problem dar. Des weiteren müssen Entscheidungen bei der Hyerparameteranpassung überdacht und verbessert werden. Dabei sollte vor allem der Backbone ResNeXt101[54] noch einmal auf einer GPU mit mehr Speicherplatz getestet werden. Das schlechte Abschneiden des komplexeren Backbones könnte an der limitierten Minibatchgröße gelegen haben. Ebenfalls wäre das Testen eines Histogrammausgleichs zur Normalisierung der Eingabebilder[5] und weiterer Daten Augmentierungen wich-

tig. Eine Normalisierung der Eingabebilder könnte getestet werden. Besonders interessant wäre die Auswertung eines Konsensmechanisums für die Berechnung des IDs, welcher in Abschnitt 6.4 vorgeschlagen wird. Für eine folgende Masterarbeit würde sich das weitere Verbessern eines Objektdetektors und die anschließende Integration in das Medizinische System der Uniklinik Würzburg anbieten. Dabei könnten neben Faster R-CNN[35] auch neuere und gegebenenfalls bessere Objektdetektoren ausgewertet werden.
Abschließend kann gesagt werden, dass Objektdetektoren ein hohes Potential für die Kardiomegalie Erkennung haben. Gerade durch die einfacher zu erstellenden Trainingsdaten, können diese in Zukunft für entscheidende Verbesserungen und den kritischen realen Einsatz sorgen.

Literaturverzeichnis

[1] Martín Abadi u. a. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Charu C Aggarwal u. a. "Neural networks and deep learning". In: *Springer* 10 (2018), S. 978–3.

[3] Hina Amin und Waqas J Siddiqui. "Cardiomegaly". In: *StatPearls [internet]* (2020).

[4] Alexey Bochkovskiy, Chien-Yao Wang und Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[5] Isarun Chamveha u. a. "Automated cardiothoracic ratio calculation and cardiomegaly detection using deep learning approach". In: *arXiv preprint arXiv:2002.07468* (2020).

[6] Jia Deng u. a. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, S. 248–255.

[7] IBM Cloud Education. *Gradient Descent*. 2020. URL: https://www.ibm.com/cloud/learn/gradient-descent (besucht am 27. 07. 2021).

[8] Ludwig Fahrmeir u. a. "Regressionsanalyse". In: *Statistik*. Springer, 2016, S. 437–475.

[9] Tatiana Gabruseva, Dmytro Poplavskiy und Alexandr Kalinin. "Deep learning for automatic pneumonia detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, S. 350–351.

[10] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, S. 1440–1448.

[11] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[12] Akhilesh Gotmare u. a. "A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation". In: *arXiv preprint arXiv:1810.13243* (2018).

[13] Priya Goyal u. a. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).

[14] Nico Hasler. "Bildsegmentierung mit Methoden der klassischen Bildverarbeitung und des Deep Learning im Vergleich amAnwendungsbeispiel der Kardiomegalie-Erkennung". Bachelorarbeit. Julius-Maximilians-Universität Würzburg, 2021.

[15] Kaiming He u. a. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 770–778.

[16] Kaiming He u. a. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 2961–2969.

[17] Ulrike Held. "Tücken von Korrelationen: die Korrelationskoeffizienten von Pearson und Spearman". In: *Swiss Medical Forum*. Bd. 10. 38. EMH Swiss Medical Publishers. 2010, S. 652–653.

[18] Jeremy Irvin u. a. “Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison”. In: *Proceedings of the AAAI conference on artificial intelligence*. Bd. 33. 01. 2019, S. 590–597.

[19] Stefan Jaeger u. a. “Two public chest X-ray datasets for computer-aided screening of pulmonary diseases”. In: *Quantitative imaging in medicine and surgery* 4.6 (2014), S. 475.

[20] Philipp Krähenbühl und Vladlen Koltun. “Efficient inference in fully connected crfs with gaussian edge potentials”. In: *Advances in neural information processing systems* 24 (2011), S. 109–117.

[21] Hieu X Le u. a. “A novel approach to remove foreign objects from chest X-ray images”. In: *arXiv preprint arXiv:2008.06828* (2020).

[22] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), S. 436–444.

[23] Yann LeCun u. a. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), S. 541–551.

[24] Zhennan Li u. a. “Automatic cardiothoracic ratio calculation with deep learning”. In: *IEEE Access* 7 (2019), S. 37749–37756.

[25] Tingting Liang u. a. “CBNetV2: A Composite Backbone Network Architecture for Object Detection”. In: *arXiv preprint arXiv:2107.00420* (2021).

[26] Tsung-Yi Lin u. a. “Feature pyramid networks for object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 2117–2125.

[27] Tsung-Yi Lin u. a. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, S. 740–755.

[28] Ian Pan, Alexandre Cadrin-Chênevert und Phillip M Cheng. “Tackling the radiological society of north america pneumonia detection challenge”. In: *American Journal of Roentgenology* 213.3 (2019), S. 568–574.

[29] Adam Paszke u. a. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019), S. 8026–8037.

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

[31] Vung Pham, Chau Pham und Tommy Dang. “Road damage detection and classification with detectron2 and faster r-cnn”. In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, S. 5592–5601.

[32] Nick A Phillips u. a. “Chexphoto: 10,000+ smartphone photos and synthetic photographic transformations of chest x-rays for benchmarking deep learning robustness”. In: *arXiv preprint arXiv:2007.06199* (2020).

[33] David MW Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. In: *arXiv preprint arXiv:2010.16061* (2020).

[34] PyTorch. *PyTorch logo white*. URL: https://commons.wikimedia.org/w/index.php?curid=88256697 (besucht am 22. 07. 2021).

[35] Shaoqing Ren u. a. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015), S. 91–99.

[36] detectron2 contributors Revision 52c81d75. *detectron2's documentation*. 2019. URL: https://detectron2.readthedocs.io/en/latest/index.html (besucht am 29. 07. 2021).

[37] Olaf Ronneberger, Philipp Fischer und Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, S. 234–241.

[38] Adrian Rosebrock. *Intersection over union, IoU for object detection*. 2016. URL: http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection (besucht am 19. 07. 2021).

[39] Evgenia Rusak u. a. “A simple way to make neural networks robust against diverse image corruptions”. In: *European Conference on Computer Vision*. Springer. 2020, S. 53–69.

[40] KC Santosh u. a. “Deep Neural Network for Foreign Object Detection in Chest X-Rays”. In: *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE. 2020, S. 538–541.

[41] Irhum Shafkat. *Intuitively understanding convolutions for deep learning*. 2018. URL: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 (besucht am 28. 07. 2021).

[42] Junji Shiraishi u. a. “Development of a digital image database for chest radiographs with and without a lung nodule: receiver operating characteristic analysis of radiologists' detection of pulmonary nodules”. In: *American Journal of Roentgenology* 174.1 (2000), S. 71–74.

[43] Karen Simonyan und Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[44] Ecem Sogancioglu u. a. “Cardiomegaly detection on chest radiographs: Segmentation versus classification”. In: *IEEE Access* 8 (2020), S. 94631–94642.

[45] Sergii Stirenko u. a. “Chest X-ray analysis of tuberculosis by deep learning with segmentation and augmentation”. In: *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*. IEEE. 2018, S. 422–428.

[46] TensorFlow. *Vectors combined, edited – Begoon*. https://github.com/valohai/ml-logos/blob/master/tensorflow-text.svg and https://github.com/valohai/ml-logos/blob/master/tensorflow-tf.svg. URL: https://github.com/tensorflow/tensorflow (besucht am 22. 07. 2021).

[47] Bram Van Ginneken, Mikkel B Stegmann und Marco Loog. “Segmentation of anatomical structures in chest radiographs using supervised methods: a comparative study on a public database”. In: *Medical image analysis* 10.1 (2006), S. 19–40.

[48] Rahul Vishwakarma und Ravigopal Vennelakanti. “CNN Model & Tuning for Global Road Damage Detection”. In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, S. 5609–5615.

[49] Fean Doe - Modified version from Walber. *Precision and Recall*. URL: https://commons.wikimedia.org/wiki/File:Precisionrecall.svg (besucht am 19. 07. 2021).

[50] Xiaosong Wang u. a. "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 2097–2106.

[51] Jöran Wessel u. a. "Sequential rib labeling and segmentation in chest X-ray using Mask R-CNN". In: *arXiv preprint arXiv:1908.08329* (2019).

[52] Yuxin Wu u. a. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[53] Yuxin Wu u. a. *Detectron2: A PyTorch-based modular object detection library*. 2019. URL: https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/ (besucht am 21. 07. 2021).

[54] Saining Xie u. a. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 1492–1500.

[55] Kaichao You u. a. "How does learning rate decay help modern neural networks?" In: *arXiv preprint arXiv:1908.01878* (2019).

Anhang

In dieser Arbeit sind zwei Faster R-CNN[35] entstanden, welche unterschiedliche Stärken zeigen und deshalb in folgenden Arbeiten berücksichtigt werden sollen. Faster R-CNN (1), zu finden in TrainingHistory\23-09.07.2021, zeigte die besten Ergebnisse in der Kardiomegalie Erkennung mit der Berechnung des ID aus der Lungenbox. Diese wurden zu allen Bildern ausgegeben. Faster R-CNN (3), zu finden in TrainingHistory\36-14.07.2021, erzielte die insgesamt besten Ergebnisse. Dabei wurde der ID aus den einzelnen Lungenflügeln berechnet, welche nicht zu allen Bilder ausgegeben wurden. In diesen Fällen, muss der HTQ in einem Nachbearbeitungsschritt auf 1 gesetzt werden.

7 USB-Speicherstick mit Ausarbeitung und Programmcode

Der Arbeit ist ein USB-Speicherstick mit digitaler Fassung der Arbeit beigelegt. Zudem befinden sich auf dem USB-Speicherstick der Programmcode und der erstellte Datensatz.

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

$Unterschrift$: $Ort, Datum$: